Automated Summarization of Stack Overflow Posts

Abstract—**Software developers often resort to Stack Overflow (SO) to fill their programming needs. Given the abundance of relevant posts, navigating them and comparing different solutions is tedious and time-consuming. Recent work has proposed to automatically summarize SO posts to concise text to facilitate the navigation of SO posts. However, these techniques rely only on information retrieval methods or heuristics for text summarization, which is insufficient to handle the ambiguity and sophistication of natural language.**

This paper presents a deep learning based framework called ASSORT for SO post summarization. ASSORT includes two complementary learning methods, ASSORT_S and ASSORT_{IS}, to address the lack of labeled training data for SO post summarization. ASSORT_S is designed to directly train a novel ensemble learning model with BERT embeddings and domain-specific features to account for the unique characteristics of SO posts. By contrast, ASSORT_{IS} is designed to reuse pre-trained models while addressing the domain shift challenge when no training data is present (i.e., zero-shot learning). Both ASSORT_S and ASSORT_{IS} outperform six existing techniques by at least 13% and 7% respectively in terms of the F1 score. Furthermore, a human study shows that participants significantly preferred summaries generated by ASSORT_S and ASSORT_{IS} over the best baseline, while the preference difference between ASSORT_S and ASSORT_{IS} was small.

Index Terms—**Stack Overflow, Text Summarization, Deep Learning**

I. INTRODUCTION

Online Q&A forums such as Stack Overflow (SO) have become an integral part of modern programming workflow [1]–[4]. However, locating essential information in online posts can be time-consuming since there are often multiple relevant posts to consider, and some posts can be lengthy. This is confirmed by a recent survey with 72 professional software developers [5]. Participants complained that sifting through many online posts was cognitively demanding and wished to get tool support for quickly navigating online posts.

Generating concise summaries of SO posts is a promising way to facilitate the navigation of SO posts [5]–[8]. Nadi and Treude experimented with four approaches to capture the gist of a SO post by extracting essential sentences from it [6]. They conducted a user study with 43 developers and confirmed that seeing essential sentences in a SO post indeed increased developers' confidence when assessing the relevance and quality of the post. However, they also found that selecting sentences only based on heuristics or information retrieval (IR) methods were not sufficient. Indeed, these approaches are inherently limited since they lack the flexibility of handling ambiguous or sophisticated narratives in natural language.

The Natural Language Processing (NLP) community has made significant progress in text summarization using deep learning (DL) [9]–[13]. Those DL-based approaches require to be trained with massive parallel corpora of text documents and summaries. For example, the CNN/DailyMails dataset [14] contains 286K pairs of news articles and human-written summaries. However, no such large datasets exist for SO posts, except for a recent dataset named SOSum [15]. While SOSum contains manually curated summaries for 2,278 SO answer posts, it is still much smaller compared with general-domain corpora such as CNN/DailyMails. Furthermore, no experiment has been done to prove that this dataset is sufficient to train a DL model for summarizing SO posts.

To bridge the gap, we propose a framework for SO post summarization, ASSORT (Automated Summarization of Stack OveRflow posT). ASSORT provides a novel supervised model ASSORT_S to account for the unique characteristics of SO posts. First, ASSORT_S uses a combination of BERT embeddings and domain-specific features to characterize sentences in a SO post. Second, since answers to different types of questions have different linguistic norms, we train a question classifier and separate models with the same BERT-based architecture for different types of questions. Third, to account for the uncertainty of the question classifier, the outputs of these models are then ensembled based on the probability distribution of the question classifier, as shown in Figure 2.

Furthermore, since acquiring labeled training data is costly, ASSORT also includes an indirect supervision method called ASSORT_{IS}, which does not need to be trained with any labeled SO data. ASSORT_{IS} makes use of supervision signals from pre-trained models in another domain, such as news articles, for SO post summarization. To handle domain shift issues, ASSORT_{IS} uses a pre-trained Natural Language Inference (NLI) model to trace back to key sentences in the original SO post based on the initially generated summary. Compared with the initially generated summary, which may contain inaccurate terminologies and narratives due to domain shift, extracting aligned sentences from the original post can more accurately capture the gist of the post.

We evaluate both learning methods in ASSORT with the comparison to a BERT-based text summarization model [16], which is fine-tuned with the same SO training data as ASSORT_S. We also select three heuristics-based methods and one unsupervised learning method from prior work [6], [17], [18] as baselines. We find that both ASSORT_S and ASSORT_{IS} outperform all baselines by at least 13% and 7% in terms of the F1 score. This implies that only finetuning a general model with a relatively small dataset such as SOSum [15] is not sufficient. To improve the training efficiency, it is necessary to account for the unique characteristics of SO posts. Furthermore, since ASSORT_{IS} is not trained on any SO data, it achieves lower accuracy than ASSORT_S. However, when less

than 20% of the original training data are available, ASSORT$_{IS}$ achieves better summarization accuracy than ASSORT$_S$. This implies that indirect supervision remains a promising yet cheap alternative for building generalizable models for specific domains when there is a lack of labeled training data.

We conduct a qualitative user study with 12 participants to evaluate the quality of summaries generated by different techniques. Compared with the best baseline model [16], the majority of participants (76%-85%) preferred summaries generated by ASSORT$_S$ or ASSORT$_{IS}$ in terms of usefulness, comprehensiveness, and conciseness. Participants found the summaries generated by ASSORT$_S$ more concise and useful in practice, while they found the summaries generated by AS-SORT$_{IS}$ more comprehensive and provided a better overview. Overall, neither ASSORT$_S$ or ASSORT$_{IS}$ really triumph over each other according to user feedback.

In summary, this work makes the following contributions:

- We design a new supervised model that generates concise and self-contained summaries of SO posts. This model accounts for the uniqueness of SO posts and achieves state-of-the-art accuracy in SO post summarization.
- We propose a new indirect supervision method to further tackle the challenge of lacking labeled training data in SO post summarization. We demonstrate the feasibility of developing models with acceptable accuracy but with no cost of data labeling.
- We conduct a comprehensive evaluation of the proposed learning methods with six comparison baselines, an ablation study, and a qualitative user study.

The rest of the paper is organized as follows. Section II describes a motivating example for generating post summaries to facilitate the navigation of SO posts. Section III defines the SO post summarization task. Sections IV and V describe the supervised and indirect supervision models respectively. Section VI describes the evaluation design and setup. Section VII describes the evaluation results. Section VIII discusses the implications, threats to validity, and future work. Section IX describes the related work. Section X concludes this work. Section XI describes the data availability.

II. MOTIVATING EXAMPLE

This section illustrates how summarizing SO posts helps developers quickly navigate SO posts and get an overview of various answers given by other developers. Suppose Alex is an Android developer and she wants to transfer key-value pairs between two layers of a Spring MVC framework. She knows both `HashMap` and `HashTable` can serve as the data structure for this task. Yet she is not sure about the pros and cons of these two APIs. So she searches online.

The first search result from Google is a Stack Overflow question—*"What are the differences between a HashMap and a HashTable in Java?"*[1] There are 35 answer posts to this question. Alex finds it time-consuming to read all of them.

---

[1] https://stackoverflow.com/questions/40471

So she decides to first read the accepted answer. The accepted answer (Post 40878) points out three major differences between `HashMap` and `HashTable`: (1) `HashTable` is synchronized, whereas `HashMap` is not; (2) `HashTable` does not allow `null` keys or values; (3) `HashMap` has the flexibility to be replaced with `LinkedHashMap`. Alex finds this answer helpful, but she is not sure how comprehensive this answer is.

Alex starts reading other highly voted answers to check if they include any important information not covered by the accepted answer. However, she finds herself submerged by the abundant information in those posts. Since many posts are lengthy with details that she does not care about, Alex spends most of her time locating helpful information in those posts. For example, Figure 1 shows the second answer (Post 41454) in this thread. It is a long post with 4 code snippets. Yet the gist of it is a simple message—*the synchronization in `HashTable` is not sufficient, and synchronized `HashMap` and `ConcurrentMap` are better choices*. Without any tool support, Alex has to read the entire post linearly to get this key message, which can take quite a few minutes.

Like many other developers, Alex only reads a handful of answers and returns to her own code due to her limited time budget [2], [19], [20]. This inevitably makes her overlook answers that are not highly ranked but contain useful information that she is unaware of. For example, the 20th answer (Post 37031553) mentions that `HashMap` has $O(log(n))$ complexity and is faster than `HashTable`. None of the top five answers mentions this. If performance is a top concern to Alex, reading this post can help Alex make a more informed decision. However, without any tool support, Alex is unlikely to reach this deep in the thread practically.

ASSORT helps Alex by automatically summarizing each answer post into concise text, so Alex can get a quick overview of many posts and prioritize which posts to spend more time on. In this way, she can make a more informed decision on which API to use. Specially, we build a Chrome extension for Stack Overflow, which highlights summative sentences from each post and renders a list of post summaries, as shown in Figure 1. By looking at the navigation panel, Alex quickly realizes that three answers mention `HashTable` is synchronized. By contrast, the other four answers suggest staying away from `HashTable`. Alex is interested in the second answer since its author expressed a strong opinion against `HashTable`. After clicking on it, Alex jumps to the corresponding answer, where three summative sentences have been highlighted as navigation cues. The first sentence points out that the synchronization in `HashTable` is not sufficient, while the second and third sentences propose synchronized `HashMap` and `ConcurrentMap` as alternatives. Alex feels that these highlighted sentences summarize the gist of the answer well, so she returns to the navigation panel to go deeper in the thread. As she scrolls down the panel, she notices the summary of the 20th answer—$O(log(n))$ *for `HashMap` vs.* $O(n)$ *in `HashTable`*. This indicates that `HashMap` has lower time complexity than `HashTable`. Alex dives into this
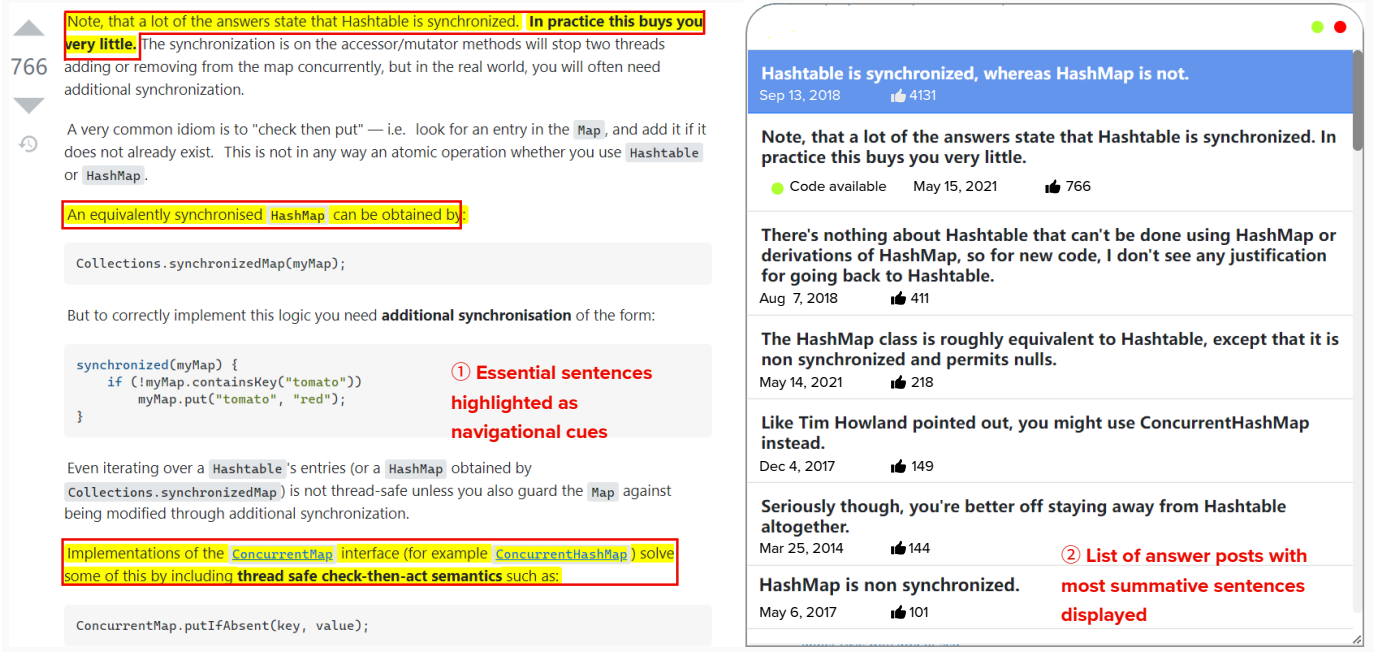
Fig. 1: A Chrome extension built upon ASSORT. On the left side, summative sentences are highlighted in yellow. On the right side, a navigation panel provides a bird's-eye view of the thread by showing the first sentence in the summary of each answer.

answer for details, as performance is always a top concern for her. Without ASSORT, this information would have been buried deep in the thread and unlikely to be discovered. Being aware of this information, Alex now feels more confident in making an informed decision between these two APIs.

III. TASK DEFINITION

We introduce the definition of *Stack Overflow Post Summarization* as follows: Given a SO answer post consisting of N sentences, the goal is to select a small set of sentences to form a succinct and self-contained summary. Essentially, this can be viewed as a contextualized sentence classification task where a sentence can either be in or not in the summary.

This task is also known as *Extractive Summarization* (ES) in NLP. It is in contrast to another type of summarization called *Abstractive Summarization* (AS). Instead of selection summative sentences from the original document, AS generates new text to summarize the document [21]. Compared with ES, a unique challenge in AS is that distorted or fabricated text is likely to be introduced during text generation [22], [23]. Several studies have shown that factual inconsistency occurs in up to 30% of abstractive summaries [24]–[27]. Furthermore, since most abstractive summarization models are pre-trained on news articles or general-domain corpora, such errors may become more prevalent when reusing those models on SO posts due to domain shift.

In this work, we focus on extractive summarization as the first step towards SO post summarization while leaving the more challenging abstractive summarization task as future work. In Section IV, we first propose a supervised learning method with a novel model architecture tailored for SO posts.
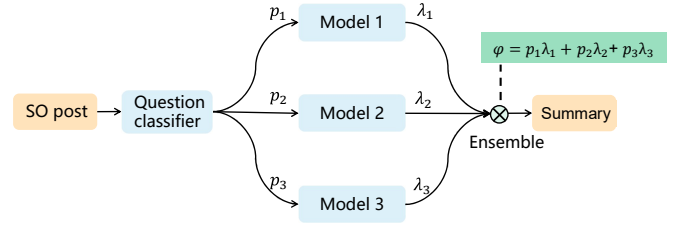

Fig. 2: An overview of ASSORT$_S$

Then in Section V, we propose an indirect supervision method that reuses pre-trained models from another domain while addressing domain shift issues via natural language inference.

IV. SUPERVISED POST SUMMARIZATION

ASSORT includes a supervised learning method named AS-SORT$_S$. As shown in Figure 2, ASSORT$_S$ takes three phases to summarize a SO answer post. Since answers to different types of questions follow different linguistic patterns, ASSORT$_S$ first predicts the type of the SO question (Phase I). To account for the uncertainty of the question classifier, the answer post is fed into three sentence classification models separately, each of which is trained for one type of SO questions (Phase II). Finally, ASSORT$_S$ ensembles the predictions of these models based on the likelihood of the question type to generate the final summary (Phase III).

A. Question Classification

While manually inspecting SO answer posts, we observed that answers to different kinds of questions have different

linguistic forms. For example, answers to a how-to question often contain a step-by-step solution, while answers to a conceptual question often contain a definition or description of the concept. Based on this insight, we decide to first categorize SO posts based on their question types and then train separate post summarization models for different types of questions. We follow the SO question taxonomy from prior work [15], [28]–[31] and consider three representative types of SO questions—*how-to questions*, *conceptual questions*, and *bug fixing questions*. How-to questions ask for instructions for achieving a task, e.g., *"how do I undo the most recent local commits in Git?"*. Conceptual questions ask for clarifications on a concept, e.g., *"what are metaclasses in Python?"*. Bug fixing questions ask for solutions to fix some issues, e.g. *"git is not working after macOS Update"*.

We develop a Support Vector Machine (SVM) classifier to categorize SO posts based on their question titles. We train it with a combination of 506 classified SO questions from SOSum [15] and 365 questions labeled by ourselves. In total, our dataset includes 301 how-to questions, 305 conceptual questions, and 265 bug fixing questions. Despite the simplicity of SVM, this classifier achieves reasonable accuracy (78%) with 8:1:1 train/dev/test data split and 10-fold validation. To further account for the potential misclassification of the SVM classier, ASSORT_S adopts an ensemble mechanism that incorporates the probability distribution of different types of questions predicted by the SVM classifier (Section IV-C). An ablation study confirms the benefit of incorporating question classification into ASSORT_S, improving its F1-score by 9% (Section VII-B).

B. Summative Sentence Identification

We train separate models to identify summative sentences for the three representative types of SO questions. These models share the same model architecture but are trained on answer posts to different types of questions to capture unique linguistic norms and writing styles of each question category. We explain the model architecture below.

Figure 3 gives an overview of the model architecture. ASSORT_S performs hybrid learning by combining the semantic representation from a BERT model and a multifaceted set of domain-specific features. Each sentence in the given post is encoded into a vector embedding, which is then fed into a feedforward neural network (FNN) to decide whether the sentence is a summative sentence.

In this work, we use a BERT model that is fine-tuned with 152 million sentences from Stack Overflow [32]. Given a sentence from a SO post, ASSORT_S computes its embedding from BERT by averaging the embedding vectors of all tokens in the sentence. The incorporation of deep contextualized sentence embeddings helps ASSORT_S capture the semantics of a given sentence. This is confirmed by the ablation study in Section VII-B, where the inclusion of BERT increases the F1 score of our model by 17%.

While the contextualized embeddings from BERT are effective in capturing sentence semantics, they are not sufficient to
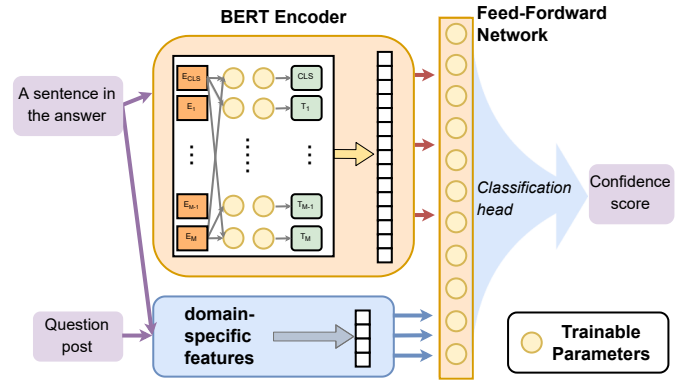

Fig. 3: The architecture of the sentence classifier

capture specific features, such as bold text, which human readers often rely on to identify summative sentences. Therefore, we further design a set of domain-specific features to capture the semantic, structural, and stylistic information cues in SO posts. We elaborate on these features below.

First, we design five **semantic features** to select summative sentences based on their content.

- *Entity Overlap.* If a sentence mentions a software entity that is also mentioned in the question title or in a SO tag of the question, this sentence may provide some directly relevant information for the question. In this work, we use SEthesaurus [33] to identify software entities in a sentence. Let E_q be the combination of software entities in the question title and the SO tags in the question. Let E_s be the set of software entities mentioned in a sentence in the answer post. The entity overlap is computed as $|E_q| \cap |E_s|/|E_q|$.
- *Comparative Adjective.* This feature captures whether a sentence contains a comparative adjective. Sentences containing a comparative adjective often contain information that helps readers compare two APIs or two solutions, e.g., *"the stack is faster because all free memory is always contiguous."*
- *Superlative Adjective.* This feature captures whether a sentence contains a superlative adjective. Similar to comparative adjectives, sentences with superlative adjectives often indicate answerers' strong inclination for or against an approach, API, or bug fix, e.g., *"there is no doubt that application/json is the best MIME type for a JSON response."*
- *Imperative Sentence.* This feature captures whether a sentence is an imperative sentence. Imperative sentences often contain instructions to accomplish a programming task or to fix a bug, e.g., *"use git revert commit-id."*
- *Linguistic Patterns.* As we label SO posts for training ASSORT_S, we summarize 19 phrases that may imply important information in an answer (Table I). Each linguistic pattern corresponds to a dimension in the sentence embedding. If a linguistic pattern is matched, then the corresponding dimension is set to 1, otherwise 0.

TABLE I: Linguistic patterns for sentences that may contain important information

No.	Phrase	No.	Phrase
1	However, ...	11	In practice, ...
2	First, ...	12	In fact, ...
3	In short, ...	13	Otherwise, ...
4	In this case, ...	14	If you care, ...
5	In general, ...	15	In contrast, ...
6	Finally, ...	16	On the other hand, ...
7	Then, ...	17	Below is ...
8	Alternatively, ...	18	Additionally, ...
9	In other words, ...	19	Furthermore, ...
10	In addition, ...		

Second, we design two **structural features** to select summative sentences based on their relations with other sentences and codes in the post.

- *Sentence Position.* This feature captures the position of a sentence in a given post. This feature is designed based on our observation that the leading sentences in many SO posts can serve as a good summary of the post.
- *Code Adjacency.* This feature indicates whether a sentence is around a code snippet. It is designed based on our observation that sentences around a code snippet enclosed by a `pre` tag often contain information cues for a programming task solution or a bug fix.

Third, we design three **stylistic features** to capture formatting styles that are used to highlight important information in a SO post.

- *Inline Code.* This feature indicates whether a sentence contains an inlined code fragment. This feature is designed based on our observation that, for how-to questions and bug-fixing questions, answerers sometimes suggest an alternative API, pinpoint a problematic piece of code, or provide a short code fragment as a solution.
- *Bold Text.* This feature indicates whether a sentence contains bold text. In SO posts, answerers often highlight important terms or statements in bold to draw attention from readers.
- *Step in a List.* This feature indicates whether a sentence is the first sentence of an item in a bulleted or numbered list. This feature is designed based on our observation that for how-to questions, many answerers typically provide a list of steps to accomplish a task.

The ablation study (Section VII-B) confirms the usefulness of these domain-specific features. Specifically, including these features leads to an increase of 6% in the F1 score.

C. Ensemble Inference

To account for the uncertainty of question classification in Phase I, we design an ensemble mechanism that merges the predictions from the three sentence classifiers to make the final decision about whether a sentence should be included in the summary.

The ensemble mechanism takes two input: (1) the softmax probabilities of k categories ($k = 3$ in this case) predicted by the question classifier, $p_1, ..., p_k$; (2) the probabilities of a sentence being a summative sentence from each sentence classifier, $\lambda_1, ..., \lambda_k$. The final score ϕ of a sentence is defined as: $\phi = \sum_{i=1}^{k} p_i \lambda_i$. If a sentence has a final score greater than a threshold θ, it is selected as a summative sentence. We experimentally determine θ to be 0.5 on a validation set of 303 posts (10% of the dataset) with the objective of maximizing the F-1 score. After every sentence in an answer post is classified, ASSORT collects all the selected sentences and outputs them as an extractive summary for the answer post.

V. POST SUMMARIZATION VIA INDIRECT SUPERVISION

While supervised learning can achieve superior performance, obtaining a large amount of labeled data is often costly, especially in specific domains such as software engineering. Therefore, we propose an indirect supervision approach, ASSORT$_{IS}$, to overcome this limit. Instead of acquiring labeled data for direct supervision, ASSORT$_{IS}$ uses supervision signals from pre-trained models in another domain, such as news article summarization. To address the challenge of data shift in cross-domain transfer, we use a pre-trained Natural Language Inference (NLI) model to select summative sentences in the original post based on the summary generated by the pre-trained text summarization model. Figure 4 provides an overview of our approach.
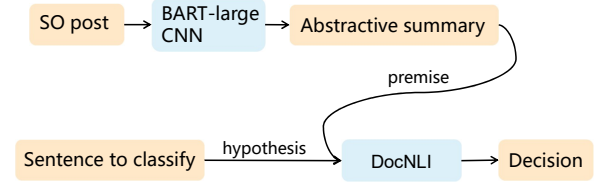

Fig. 4: An overview of ASSORT$_{IS}$

In this work, we use BART-large-CNN [34] as the pre-trained text summarization model. Unlike our supervised model in Section IV, BART-large-CNN is an abstractive summarization model. In other words, BART-large-CNN generates a summary in its own words via an autoregressive decoding process, rather than selecting summative sentences in the input document. Specifically, BART-large-CNN is based on BART [10], a denoised auto-encoder that encodes an input document to a high-dimensional embedding for text reconstruction. In BART-large-CNN, the initial BART model is fine-tuned to generate text summaries using the CNN/DailyMail corpus [14], which contains 300K news articles and their human-written summaries. In this work, we select BART-large-CNN, since it is a representative and strong summarization model that outperforms other transformer-based models such as [13] by up to 6 points on CNN/DailyMail in ROUGE-based metrics [10].

While BART-large-CNN is demonstrated effective on news article summarization, it suffers from domain-shifting issues when applied to a different domain, such as SO post summarization. Furthermore, recent studies have shown that ab-

TABLE II: An abstractive summary generated by BART-large-CNN and the summary generated by our approach

Original post [Post 438095]
There is a potential problem here with regard to workflow persistence.
If you create long running workflows that are persisted in a database, these workflows are not reloaded into memory until there is some external event that reloads them. As there they are responsible for triggering the event themselves but cannot until they are reloaded. And we have a catch 22 :-(.
The proper way to do this is using an external service. And while this might feel like dividing the code into two places it really isn't. The reason is that the workflow is responsible for the big picture, IE what should be done. And the runtime service is responsible for the actual implementation or how it should be done. That way you can change the how without changing the why and when part. **[Source of the ineloquent sentence in BART-large-CNN's summary]**

Abstractive summary generated by BART-large-CNN
There is a potential problem with regard to workflow persistence. If you create long running workflows that are persisted in a database they will not be able to restart until there is an external event reloading them. The proper way of doing this is using an external service which can change the how part without changing what it should be done or when it's better than having two separate processes for different tasks as each one does their own thing but doesn't agree on exactly who must carry out its function at any given time. **[broken English]**

Summary generated by our indirect supervision approach
There is a potential problem here with regard to workflow persistence. The proper way to do this is using an external service. That way you can change the how without changing the why and when part.

stractive summarization models may introduce factual inconsistencies in generated text compared with original text [22], [23]. Table II shows an abstractive summary generated by BART-large-CNN. The last sentence in it (highlighted in red) is exceptionally long and contains broken English with some hallucinated information.

Despite the broken English, this sentence is not fully made up by BART-large-CNN. It is based on two sentences in the original post (highlighted in green). In fact, these two sentences provide a good summary of the last paragraph in the original post. Though BART-large-CNN recognizes them, it fails to generate an eloquent summary due to domain shift.

To address this issue, we propose associating abstractive summaries to the original post's corresponding sentences using a pre-trained Natural Language Inference (NLI) model. The task of NLI [35] studies to what extent a premise entails, contradicts, or remains neutral with a hypothesis. For example, suppose we have a hypothesis sentence, "*a kid ate a fruit*", and a premise sentence, "*a boy ate an apple*". An NLI model will predict that the premise entails the hypothesis. However, if the hypothesis is changed to "*a kid ate a banana*", the prediction

will becomes to neutral or contradiction.

We use a pre-trained NLI model to decide which sentence in the original SO post is entailed by the abstractive summary and thus should be included in the final summary. Specifically, we use a RoBERTa model that is pre-trained on DocNLI [36], a document-level NLI dataset. DocNLI covers multiple text genres, such as news, fiction, and conversations, with multi-sentence (i.e., document) premises and single-sentence hypotheses. Given a summary generated by BART-large-CNN, our approach checks the logical relationship between the summary (i.e., the premise) and every sentence in the original post (i.e., the hypotheses). The DocNLI model will then produce a probability distribution over the three relationships—*entail*, *contradict*, and *neutral*. Our approach selects a sentence as part of the final summary if the entailment probability is the higher than the other two probabilities.

VI. EVALUATION

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

- RQ1: How effective is our supervised model, ASSORT_S, in SO post summarization?
- RQ2: To what extent does each component in ASSORT_S contribute to its effectiveness?
- RQ3: How does our indirectly supervised model, ASSORT_{IS}, compare to our surprised model?
- RQ4: How do real programmers perceive the summaries given by our models?

A. Experiment Setup

In addition to the 2,278 labeled posts from SOSum [15], we further manually labeled 785 answer posts following the same labeling procedure as described in [15]. We use these 3,063 posts for training and evaluation. These posts are from 785 SO questions, including 254 *how-to* questions, 322 *conceptual* questions, and 209 *bug-fixing* questions.

To train ASSORT_S, for each type of questions, we randomly split the corresponding answer posts into training, development, and test sets with a split ratio of 8:1:1. Then, we train a summative sentence identification model for answers to each type of question following the model architecture in Figure 3. Each model was trained with a batch size of 512 in 150 epochs. We use the Adam optimizer [37] for training and the learning rate is set to $1e^{-5}$.

To design an ablation study to answer RQ2, we create four variants of ASSORT_S with one key component removed in each variant. The four key components are: (1) the BERT embeddings, (2) the domain-specific features, (3) the question classifier, (4) the ensemble mechanism. Specifically, when the question classifier is ablated, ASSORT_S uses a sentence classifier trained on all types of questions to predict the final score of a sentence. When the ensemble mechanism is ablated, ASSORT_S first predicts the question type of a given post and then only uses the sentence classifier for the predicted question type to identify summative sentences, rather than using all three classifiers.

To answer RQ3, we first compare ASSORT_{IS} against AS-SORT_S on a full training setting where we train ASSORT_S with the entire training set. Then, we compare them in various low-resource settings where only a subset of the original training data is available. In those low-resource settings, we randomly select SO posts from the original dataset to make subsets of training data. Since ASSORT_{IS} does not require any training data, the low-resource baselines are created to make a fair comparison between ASSORT_S and ASSORT_{IS} in conditions where training data are scarce.

B. Compared Baselines

We select a state-of-the-art extractive summarization model on the general domain [16] and also fine-tune it on the same training set of 2,424 SO posts as ASSORT_S. Furthermore, we select three heuristics-based methods and one unsupervised method that can perform sentence-level summarization from prior work [6], [17], [18]. These four methods have also been experimented in [6]. We do not compare with paragraph-level summarization techniques [5], [38], since it is not a head-to-head comparison.

(1) *BertSum* [16] is an extractive summarization model that first uses BERT to encode sentences and then uses a transformer to select summative sentences [16]. It outperforms several previous techniques [11], [39]–[41] on two popular text summarization datasets—NYT [42] and CNN/DailyMail [14]. In this experiment, we use the checkpoint of *BertSum* that has the best performance on CNN/DailyMail.

(2) *BertSum (fine-tuned)* [16] is a fine-tuned version of *BertSum*. It is fine-tuned with the training data of ASSORT_S, including 2,424 SO posts and their summaries.

(3) *wordpattern* identifies essential sentences in a SO post using a set of 360 word patterns. These patterns are initially designed by Robillard and Chhetri [17] to identify sentences containing indispensable knowledge in API documentations.

(4) *simpleif* is a technique proposed by Nadi and Treude [6]. It is designed based on the insight that essential sentences may contain contextual information expressed in the form of conditions. Thus, *simpleif* identifies all sentences that have the word "if" in them as essential sentences.

(5) *contextif* is another technique proposed by Nadi and Treude [6]. It uses a set of heuristics to identify essential sentences that carry technical context and are useful.

(6) *lexrank* is a commonly used unsupervised text summarization approach [18]. It uses a stochastic graph-based method to compute the relative importance of sentences in a document and generates an extractive summary by selecting the top k sentences. We use the default k value, 5, in an open-source implementation of *lexrank* [43].

C. Evaluation Metrics

We use three metrics—precision, recall, and F1 to measure the effectiveness of our methods and the comparison baselines in SO post summarization. Each metric is calculated at the sentence level. Given a set of summative sentences in a set of SO posts G, let M be the set of summative sentences selected by an extractive summarization model. The precision of the model is calculated as $|G \cap M|/|M|$. And the recall of the model is defined as $|G \cap M|/|G|$. Furthermore, we measure the F1 score, which combines the precision and recall of a model into a single metric by taking their harmonic mean.

We make sure both ASSORT_S and ASSORT_{IS} and all the baselines are evaluated with the same test set (i.e., 304 posts and their summaries) to make the comparison fair. A 10-fold validation is performed when calculating the metrics for our approaches and all the baselines.

D. User Study Design

To answer RQ4, we conduct a within-subjects user study to evaluate the summary quality. We recruit 12 graduate students through the department mailing list from an R1 university. Participants have an average of four years of programming experience. We randomly selected 40 answer posts from our dataset, including 13 *how-to questions*, 13 *conceptual questions*, and 14 *bug-fixing questions*. In each user study, we select 10 out of the 40 answer posts and ask the participant to review their summaries. We counterbalance the post assignment so that each post is evaluated by three different participants.

For each answer post, participants first report their expertise of the concepts in the question on a 7-point scale. 1 means "Haven't even heard of it before" and 7 means "I am an expert". Then, they will be provided with the question post, the answer post, and the summaries generated by ASSORT_S, ASSORT_{IS}, and *BertSum (fine-tuned)*. Specifically, we select *BertSum (fine-tuned)* as our baseline in the user study since it performs the best among all six baselines in the quantitative experiment. After reading all three summaries of a post, the participants evaluate the quality of these summaries by answering the following five multiple-choice questions.

(1) *Which summary provides the most helpful information?*
(2) *Which summary provides the best overview of the post?*
(3) *Which summary provides the most comprehensive information?*
(4) *Which summary is the most concise without being incomplete?*
(5) *Which summary do you prefer to see in practice?*

At the end of the user study, participants answer several open-ended questions about whether they wish to see SO post summaries when browsing SO and what kinds of characteristics an ideal SO summary should possess.

VII. RESULTS

A. RQ1: SO Post Summarization Accuracy

Table III shows the summarization accuracy of ASSORT_S and the baselines after 10-fold validation is performed. ASSORT_S achieves the best results in all three metrics. Furthermore, the domain shift from general text corpora to the SO post corpus is non-trivial. The original *BertSum* model, which is trained on news articles and their summaries from CNNDailymails, only achieves an F1 score of 53%. While

TABLE III: Comparison of ASSORT_S and baselines

	Precision	Recall	F1
ASSORT_S ★	**0.73**	**0.69**	**0.71**
BertSum ★	0.47	0.60	0.53
BertSum (fine-tuned) ★	0.51	0.67	0.58
wordpattern ◇	0.40	0.03	0.06
simpleif ◇	0.39	0.15	0.21
contextif ◇	0.39	0.15	0.22
lexrank ✩	0.61	0.45	0.52

★: DL-based, ◇: heuristics-based, ✩: unsupervised

TABLE IV: Contribution of each component in ASSORT_S

	Precision	Recall	F1
ASSORT_S	**0.73**	**0.69**	**0.71**
—w/o BERT	0.61	0.48	0.54
—w/o Domain-specific features	0.70	0.61	0.65
—w/o Ensemble	0.68	0.66	0.67
—w/o Question classifier	0.61	0.63	0.62

fine-tuning *BertSum* with 2,424 labeled SO posts increases the F1 score from 53% to 58%, it is still 13% below ASSORT_S. Given that *BertSum* also uses BERT for sentence encoding, this result implies that incorporating domain-specific features and ensembling based on question types indeed improve the effectiveness of SO post summarization.

B. RQ2: Ablation Study

Table IV shows the ablation study results. On the one hand, ablating the BERT embedding leads to the largest accuracy degradation, 17% in the F1 score. This indicates that incorporating deep contextualized embeddings from a language model is critical for a domain-specific task such as SO post summarization. On the other hand, only including BERT is also not sufficient. Removing each of the other three components, which are specifically designed to account for the unique characteristics of SO posts, leads to a non-trivial decrease in the F1 score. Specifically, the removal of domain-specific features decreases the F1 score by 6%. This indicates that deep contextualized embeddings alone cannot cover the unique structural and linguistic patterns that distinguish SO posts from general text data. Removing the question classifier decreases the F1 score by 9%. This indicates that accounting for different linguistic norms in answers to different types of questions indeed helps. Removing the ensemble mechanism leads to a decrease of 4% in the F1 score. This indicates that the ensemble mechanism can help to alleviate the influence of question classification errors.

C. RQ3: Supervision vs. Indirect Supervision

Figure 5 compares the accuracy of ASSORT_S and ASSORT_{IS} when various amounts of training data are available. Given that ASSORT_{IS} only uses pre-trained models and does not require any training data, the accuracy of ASSORT_{IS} is constant (65%) in these settings. When all training data (i.e., 2,424 posts and their summaries) is available, ASSORT_S outperforms ASSORT_{IS} by 6%. This indicates that directly

training a supervised model is a better choice when there are sufficient training data. However, when 20% or less of the original training data is available, ASSORT_{IS} outperforms ASSORT_S. Therefore, when the training data is small, indirect supervision can be a better option than directly training a model with small training data. Furthermore, with an F1 score of 65%, ASSORT_{IS} outperforms all six comparison baselines by 7% to 59%. This result is significant since the best comparison baseline is fine-tuned on all training data. This implies that indirect supervision can be a promising yet low-cost alternative compared with unsupervised approaches and model fine-tuning.
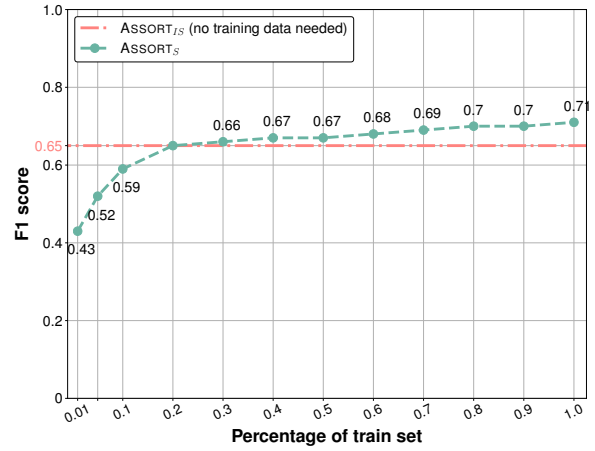

Fig. 5: The F1 score of the direct vs. indirect supervision methods when various amounts of training data are available.

D. RQ4: Human Evaluation

Figure 6 shows the choice of participants over the summaries generated by ASSORT_S, ASSORT_{IS}, and *BertSum (fine-tuned)* in five aspects. Overall, participants strongly preferred summaries generated by ASSORT_S or ASSORT_{IS} over *Bert-Sum (fine-tuned)*. Between ASSORT_S and ASSORT_{IS}, more participants found summaries generated by ASSORT_S more helpful, concise, and practical. Yet more participants found summaries generated by ASSORT_{IS} more comprehensive and providing a better overview of the post.

Since each answer post and its summaries were reviewed by three participants, we further analyzed the consistency among those participants when they answered each multiple-choice question. In 60% of the cases, the three participants assigned to the same post consistently chose summaries generated by ASSORT over *BertSum (fine-tuned)* in a multiple-choice question. In 91% of cases, at least two out of three participants consistently chose either ASSORT_S or ASSORT_{IS} over *BertSum (fine-tuned)*. However, the choices between ASSORT_S and ASSORT_{IS} were not very consistent among participants assigned to the same post. In only 40% of cases, all three participants consistently chose ASSORT_S or ASSORT_{IS}. This
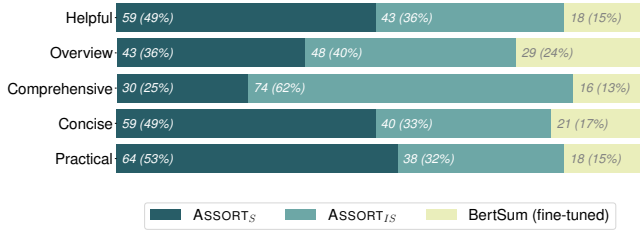
Fig. 6: Participants' choices over different types of summaries in five aspects
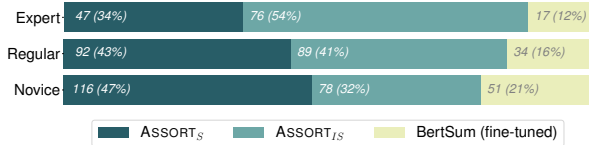

Fig. 7: Choices of participants with different levels of expertise

implies the participants chose between ASSORT_S and ASSORT_{IS} largely based on their personal preference. Neither ASSORT_S or ASSORT_{IS} really triumph over each other.

We also investigated how participants' expertise on concepts in a SO question affects their preferences over post summaries. As described in Section VI-D, participants first reported their expertise on a 7-point scale. We categorized their expertise as "novice" if their rating is 1-2, "regular" if their rating is 3-5, and "expert" if their rating is 6-7. Figure 7 shows the choices of participants with different levels of expertise. We observed an tendency of favoring ASSORT_S among novices while an tendency of favoring ASSORT_{IS} among experts. One plausible reason for this is that novices prefer to see short summaries and feel overwhelmed if a summary contains too much information, while experts prefer to see more comprehensive information and feel less overwhelmed. Pearson's Chi-square test of independence shows that the choice difference among participants with different levels of expertise for post summaries is statistically significant ($p = 0.0006$).

In the final survey, 84% of participants confirmed they would like to see post summaries when browsing SO posts. For example, P4 elaborates, "*seeing a concise yet informative summary can help me quickly decide whether a post is worth reading or not.*" Another participant (P10) said, "*most information in a long post is useless to me, what I want is the solution and solution only*".

VIII. DISCUSSION

Our experiment shows that only fine-tuning a general text summarization model such as [16] is insufficient. Our supervised approach, ASSORT_S, achieves significantly higher summarization accuracy by incorporating specific designs for SO posts, such as question classification and domain-specific features. This may carry a more general implication—as we reuse models from NLP or ML, we should consider renovating

their model architectures and adapting them to account for the unique characteristics of SE tasks.

Our indirect supervision approach, ASSORT_{IS}, is proposed to address the challenge of lacking labeled data in SO post summarization. ASSORT_{IS} has achieved reasonable accuracy and is proven to be a better choice when labeled data is insufficient, e.g., less than 485 posts in our experiment setting. Furthermore, our user study shows that while summaries generated by ASSORT_{IS} and ASSORT_S were deemed good in different aspects (e.g., *comprehensiveness* vs. *conciseness*), there was no significant preference difference in general. This implies that ASSORT_{IS} could be an acceptable yet low-cost solution in practice.

Both ASSORT_S and ASSORT_{IS} can be generalized to summarize other types of SE documents, such as API documentation, tutorials, and bug reports. Since ASSORT_{IS} only uses pre-trained models, it can be reused as-is. By contrast, ASSORT_S would benefit from several extensions, e.g., re-training a topic classifier rather than a question classifier, re-designing some domain-specific features based on the linguistic norms in the target corpus. Furthermore, since ASSORT_{IS} does not require any labeled training data, it can also be used as a starting point to explore the feasibility and opportunities of summarizing other types of SE documents.

Threats to validity. In terms of internal validity, labeling summative sentences in SO posts is a subjective task. Therefore, different labelers will not necessarily select the same set of summative sentences from the same post. In this work, two labelers expanded SOSum with 785 answer posts retrieved under 365 SO questions. We strictly follow the labeling procedure established in [15]. The Cohen's kappa score is 0.72 on the initial labeling results before discussion, which indicates moderate agreement. Most disagreements were resolved after discussion. In the final dataset, only those sentences that both labelers agreed on are labeled as summative sentences.

In terms of external validity, since our dataset is constructed by retrieving posts under the most popular questions, it inevitably favors popular tags, such as Java and Python. Our dataset contains 1,089 unique tags, while there are more than 50K tags on Stack Overflow. The accuracy of ASSORT_S may drop a bit with unfamiliar topics due to unseen terminologies. Furthermore, the accuracy of ASSORT_S may also decrease when reused as-is for other types of SE documents. This is because ASSORT_S includes unique designs for SO posts, which is not applicable to other types of SE documents.

In terms of construct validity, our question classifier does not always make the correct prediction. Sometimes, the boundary between different question categories can be blurred with questions like *"How to fix headers-already-sent error in PHP?"*. This question should be classified as *bug-fixing*, since it mentions an error. However, it is classified as *how-to* by our question classifier since the question title contains a phrase *"how to"*. To mitigate this threat, ASSORT_S includes an ensemble mechanism that merges predictions from separate summative sentence prediction models.

Future work. Currently, ASSORT_S only uses a feedforward

neural network as the classification head on top of BERT. Prior work [16] has shown that the choice of classification heads has an influence on model accuracy. It would be worthwhile to further experiment with other types of classification heads, such as transformers and LSTMs. Furthermore, we have only considered single-document summarization in this work. It would be interesting to investigate how to perform multi-document summarization on a thread of SO posts. The challenge of multi-document summarization is not only to shorten the text, but also to organize information around the key aspects to represent diverse views. Compared with extractive summarization, abstractive summarization may be a more preferred solution, since it is capable of rephrasing sentences from different posts and generating gradual transitions. Diversity-based ranking algorithms such as Maximal Marginal Relevance (MMR) [44] can also be considered here.

IX. RELATED WORK

SO post summarization. Several approaches have been presented to summarize SO posts to concise texts to facilitate SO post navigation [5]–[7], [38]. AnswerBot extracts summative paragraphs from SO posts based on features such as information entropy and paragraph position [5]. CraSolver uses a similar multi-factor ranking mechanism to summarize bug solutions [38]. Both AnswerBot and CraSolver generate summaries at the paragraph level. By contrast, our approach generates more fine-grained sentence-level summaries of SO posts. Nadi and Treude experimented with four different IR-based approaches to select essential sentences from SO posts [6]. They conducted a survey with 43 developers and found that while participants would like to get navigation support for browsing SO, the IR-based approaches failed to provide such support. Motivated by these findings, we propose a novel DL-based framework that can more accurately identify summative sentences in SO posts in this work.

Text summarization for bug reports. There are several summarization techniques for other types of SE documents such as bug reports [45]–[49]. Liu et al. designed a new metric called *believability score*, which measures the degree to which a sentence is supported or refuted by other comments. Based on the idea of *believability score*, they further developed BugSum, which selects summative sentences in a bug report [46]. Rastkar et al. created a small dataset with human-annotated summaries for 36 bug reports and used it to train a Logistic Regression classifier with 24 explicit features to identify summative sentences in a bug report [47]. Mani et al. experimented with four unsupervised models— Centroid, MMR, Grasshopper, and Diverse Rank—on the task of bug report summarization [48]. Unlike these techniques, we proposed two learning paradigms for building DL models for SO post summarization.

General text summarization. Many neural-based text summarization techniques have been proposed recently [11], [12], [16], [50]–[52]. For example, Liu et al. presented a BERT-based model and trained it on CNN/DailyMails [16]. Narayan et al. presented a CNN-based model that summarizes a single news article into a one-liner and trained it with XSum [50]. In another work, Narayan et al. framed the task of text summarization as a ranking problem and proposed a global optimization method for training [11]. Dong et al. framed text summarization as a contextual bandit problem, in which each document is considered as a context and each combination of selected sentences is an action to take [12]. While these models perform well on news articles, reusing these models in another domain, such as SO posts, is not an easy task due to domain shift. In this work, we propose ASSORT_S, a supervised model that accounts for unique characteristics of SO posts and demonstrate that it significantly outperforms a fine-tuned version of *BertSum* (Section VII-A).

Other tool support for information seeking in SO. In addition to text summarization, many other kinds of tool support have been proposed to facilitate information seeking in SO [53]–[67]. For example, SeaHawk [53] is an Eclipse plugin that integrates SO into an IDE. Chatbot4QR [68] expands a user query with tags of similar SO questions in order to improve search results. Ye et al. [55] propose a re-ranking mechanism for SO search results of a user-defined query based on different search focus users may have. They identify linguistic patterns in different types of queries and train separate models based on query types to re-rank search results. SISE uses linguistic patterns to identify sentences that mention an API in SO posts and augments API documents with those sentences [60].

X. CONCLUSION

In this work, we propose two complementary learning methods for automated post summarization in Stack Overflow (SO). ASSORT_S is a supervised method that accounts for unique characteristics in SO posts via question classification, domain-specific features, and ensemble inference. ASSORT_{IS}, on the other hand, is an indirectly supervised method that does not require any labeled training data by leveraging a pre-trained model from another domain while addressing the domain shift issue via natural language inference. Both ASSORT_S and ASSORT_{IS} significantly outperform six existing techniques in terms of summarization accuracy. ASSORT_{IS} is demonstrated to be a promising yet low-cost solution in a low-resource setting. Furthermore, a user study shows that participants consistently favored summaries generated by ASSORT_S or ASSORT_{IS} over the best baseline model, while the preference difference between ASSORT_S and ASSORT_{IS} was small. In the future, we plan to extend ASSORT to support multi-document summarization on the entire discussion thread and also apply ASSORT to other types of SE documents.

XI. DATA AVAILABILITY

Our code and data have been made available in an anonymous GitHub repository for review.[2]

---

[2]https://anonymous.4open.science/r/ASSORT-Automatic-Summarization-of-Stack-Overflow-Posts-4FE3

REFERENCES

[1] J. Brandt, P. J. Guo *et al.*, "Two studies of opportunistic programming: interleaving web foraging, learning, and writing code," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2009, pp. 1589–1598.

[2] X. Xia, L. Bao *et al.*, "What do developers search for on the web?" *Empirical Software Engineering*, vol. 22, no. 6, pp. 3149–3185, 2017.

[3] Y. Wu, S. Wang *et al.*, "How do developers utilize source code from stack overflow?" *Empirical Software Engineering*, vol. 24, no. 2, pp. 637–673, 2019.

[4] R. Abdalkareem, E. Shihab *et al.*, "What do developers use the crowd for? a study using stack overflow," *IEEE Software*, vol. 34, no. 2, pp. 53–60, 2017.

[5] B. Xu, Z. Xing *et al.*, "Answerbot: Automated generation of answer summary to developers' technical questions," in *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2017, pp. 706–716.

[6] S. Nadi and C. Treude, "Essential sentences for navigating stack overflow answers," in *2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2020, pp. 229–239.

[7] R. F. Silva, C. K. Roy *et al.*, "Recommending comprehensive solutions for programming tasks by mining crowd knowledge," in *2019 IEEE/ACM 27th International Conference on Program Comprehension (ICPC)*. IEEE, 2019, pp. 358–368.

[8] Q. Huang, X. Xia *et al.*, "Api method recommendation without worrying about the task-api knowledge gap," in *2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2018, pp. 293–304.

[9] D. Khashabi, S. Min *et al.*, "Unifiedqa: Crossing format boundaries with a single qa system," *EMNLP - findings*, 2020.

[10] M. Lewis, Y. Liu *et al.*, "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," *arXiv preprint arXiv:1910.13461*, 2019.

[11] S. Narayan, S. B. Cohen *et al.*, "Ranking sentences for extractive summarization with reinforcement learning," *arXiv preprint arXiv:1802.08636*, 2018.

[12] Y. Dong, Y. Shen *et al.*, "Banditsum: Extractive summarization as a contextual bandit," *arXiv preprint arXiv:1809.09672*, 2018.

[13] Y. Liu and M. Lapata, "Text summarization with pretrained encoders," *arXiv preprint arXiv:1908.08345*, 2019.

[14] "Document summarization on cnn/daily mail," https://paperswithcode.com/sota/document-summarization-on-cnn-daily-mail, 2022, accessed: 2022-3-29.

[15] B. Kou, Y. Di, M. Chen, and T. Zhang, "Sosum: A dataset of stack overflow post summaries," in *2022 IEEE/ACM 19th International Conference on Mining Software Repositories (MSR)*. IEEE, 2022, pp. 247–251.

[16] Y. Liu, "Fine-tune bert for extractive summarization," *arXiv preprint arXiv:1903.10318*, 2019.

[17] M. P. Robillard and Y. B. Chhetri, "Recommending reference api documentation," *Empirical Software Engineering*, vol. 20, no. 6, pp. 1558–1586, 2015.

[18] G. Erkan and D. R. Radev, "Lexrank: Graph-based lexical centrality as salience in text summarization," *CoRR*, vol. abs/1109.2128, 2011. [Online]. Available: http://arxiv.org/abs/1109.2128

[19] J. Brandt, P. J. Guo, J. Lewenstein, M. Dontcheva, and S. R. Klemmer, "Two studies of opportunistic programming: interleaving web foraging, learning, and writing code," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2009, pp. 1589–1598.

[20] X. Xia, L. Bao, D. Lo, P. S. Kochhar, A. E. Hassan, and Z. Xing, "What do developers search for on the web?" *Empirical Software Engineering*, vol. 22, no. 6, pp. 3149–3185, 2017.

[21] S. Gupta and S. K. Gupta, "Abstractive summarization: An overview of the state of the art," *Expert Systems with Applications*, vol. 121, pp. 49–65, 2019.

[22] Y. Huang, X. Feng, X. Feng, and B. Qin, "The factual inconsistency problem in abstractive text summarization: A survey," *arXiv preprint arXiv:2104.14839*, 2021.

[23] W. Kryściński, B. McCann, C. Xiong, and R. Socher, "Evaluating the factual consistency of abstractive text summarization," *arXiv preprint arXiv:1910.12840*, 2019.

[24] Z. Cao, F. Wei, W. Li, and S. Li, "Faithful to the original: Fact aware neural abstractive summarization," in *thirty-second AAAI conference on artificial intelligence*, 2018.

[25] W. Kryściński, N. S. Keskar, B. McCann, C. Xiong, and R. Socher, "Neural text summarization: A critical evaluation," *arXiv preprint arXiv:1908.08960*, 2019.

[26] B. Goodrich, V. Rao, P. J. Liu, and M. Saleh, "Assessing the factual accuracy of generated text," in *proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 166–175.

[27] T. Falke, L. F. Ribeiro, P. A. Utama, I. Dagan, and I. Gurevych, "Ranking generated summaries by correctness: An interesting but challenging application for natural language inference," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 2214–2220.

[28] L. B. De Souza, E. C. Campos *et al.*, "Ranking crowd knowledge to assist software development," in *Proceedings of the 22nd International Conference on Program Comprehension*, 2014, pp. 72–82.

[29] C. Treude, O. Barzilay *et al.*, "How do programmers ask and answer questions on the web?(nier track)," in *Proceedings of the 33rd international conference on software engineering*, 2011, pp. 804–807.

[30] C. Rosen and E. Shihab, "What are mobile developers asking about? a large scale study using stack overflow," *Empirical Software Engineering*, vol. 21, no. 3, pp. 1192–1223, 2016.

[31] M. Allamanis and C. Sutton, "Why, when, and what: analyzing stack overflow questions by topic, type, and code," in *2013 10th Working conference on mining software repositories (MSR)*. IEEE, 2013, pp. 53–56.

[32] J. Tabassum, M. Maddela, W. Xu, and A. Ritter, "Code and named entity recognition in stackoverflow," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2020. [Online]. Available: https://www.aclweb.org/anthology/2020.acl-main.443/

[33] C. Chen, Z. Xing, and W. Ximing, "Unsupervised software-specific morphological forms inference from informal discussions," in *The 39th International Conference on Software Engineering, Buenos Aires, Argentina*. IEEE, 2017.

[34] "Bart large cnn," https://huggingface.co/facebook/bart-large-cnn, 2022, accessed: 2022-1-7.

[35] I. Dagan, O. Glickman, and B. Magnini, "The pascal recognising textual entailment challenge," in *Machine learning challenges workshop*. Springer, 2005, pp. 177–190.

[36] W. Yin, D. Radev, and C. Xiong, "Docnli: A large-scale dataset for document-level natural language inference," *arXiv preprint arXiv:2106.09449*, 2021.

[37] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[38] H. Wang, X. Xia *et al.*, "Automatic solution summarization for crash bugs," in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 2021, pp. 1286–1297.

[39] Q. Zhou, N. Yang, F. Wei, S. Huang, M. Zhou, and T. Zhao, "Neural document summarization by jointly learning to score and select sentences," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2018, pp. 654–663.

[40] A. See, P. J. Liu, and C. D. Manning, "Get to the point: Summarization with pointer-generator networks," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017, pp. 1073–1083.

[41] A. Celikyilmaz, A. Bosselut, X. He, and Y. Choi, "Deep communicating agents for abstractive summarization," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2018, pp. 1662–1675.

[42] E. Sandhaus, "The new york times annotated corpus," *Linguistic Data Consortium, Philadelphia*, vol. 6, no. 12, p. e26752, 2008.

[43] "Stack exchange data explorer," https://pypi.org/project/lexrank/, 2022, accessed: 2022-8-15.

[44] J. Carbonell and J. Goldstein, "The use of mmr, diversity-based reranking for reordering documents and producing summaries," in *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, 1998, pp. 335–336.

[45] S. Rastkar, G. C. Murphy, and G. Murray, "Summarizing software artifacts: a case study of bug reports," in *2010 ACM/IEEE 32nd*

International Conference on Software Engineering, vol. 1. IEEE, 2010, pp. 505–514.

[46] H. Liu, Y. Yu et al., BugSum: Deep Context Understanding for Bug Report Summarization. New York, NY, USA: Association for Computing Machinery, 2020, p. 94–105. [Online]. Available: https://doi.org/10.1145/3387904.3389272

[47] S. Rastkar, G. C. Murphy et al., "Automatic summarization of bug reports," IEEE Transactions on Software Engineering, vol. 40, no. 4, pp. 366–380, 2014.

[48] S. Mani, R. Catherine, V. S. Sinha, and A. Dubey, "Ausum: approach for unsupervised bug report summarization," in Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering, 2012, pp. 1–11.

[49] X. Li, H. Jiang, D. Liu, Z. Ren, and G. Li, "Unsupervised deep bug report summarization," in 2018 IEEE/ACM 26th International Conference on Program Comprehension (ICPC). IEEE, 2018, pp. 144–14 411.

[50] S. Narayan, S. B. Cohen et al., "Don't give me the details, just the summary!" Topic-aware Convolutional Neural Networks for Extreme Summarization. In, 2018.

[51] S. Verma and V. Nidhi, "Extractive summarization using deep learning," arXiv preprint arXiv:1708.04439, 2017.

[52] D. Miller, "Leveraging bert for extractive text summarization on lectures," arXiv preprint arXiv:1906.04165, 2019.

[53] L. Ponzanelli, A. Bacchelli, and M. Lanza, "Seahawk: Stack overflow in the ide," in 2013 35th International Conference on Software Engineering (ICSE). IEEE, 2013, pp. 1295–1298.

[54] L. Ponzanelli, G. Bavota et al., "Mining stackoverflow to turn the ide into a self-confident programming prompter," in Proceedings of the 11th Working Conference on Mining Software Repositories, 2014, pp. 102–111.

[55] T. Ye, B. Xie, Y. Zou, and X. Chen, "Interrogative-guided re-ranking for question-oriented software text retrieval," in Proceedings of the 29th ACM/IEEE international conference on Automated software engineering, 2014, pp. 115–120.

[56] Y. Zou, T. Ye, Y. Lu, J. Mylopoulos, and L. Zhang, "Learning to rank for question-oriented software text retrieval (t)," in 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE). IEEE, 2015, pp. 1–11.

[57] M. Soliman, A. R. Salama, M. Galster, O. Zimmermann, and M. Riebisch, "Improving the search for architecture knowledge in online developer communities," in 2018 IEEE International Conference on Software Architecture (ICSA). IEEE, 2018, pp. 186–18 609.

[58] M. M. Rahman and C. Roy, "Effective reformulation of query for code search using crowdsourced knowledge and extra-large data analytics," in 2018 IEEE International Conference on Software Maintenance and Evolution (ICSME). IEEE, 2018, pp. 473–484.

[59] H. Li, S. Li et al., "Improving api caveats accessibility by mining api caveats knowledge graph," in 2018 IEEE International Conference on Software Maintenance and Evolution (ICSME). IEEE, 2018, pp. 183–193.

[60] C. Treude and M. P. Robillard, "Augmenting api documentation with insights from stack overflow," in 2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE). IEEE, 2016, pp. 392–403.

[61] H. Li, S. Li, J. Sun, Z. Xing, X. Peng, M. Liu, and X. Zhao, "Improving api caveats accessibility by mining api caveats knowledge graph," in 2018 IEEE International Conference on Software Maintenance and Evolution (ICSME). IEEE, 2018, pp. 183–193.

[62] S. Subramanian, L. Inozemtseva et al., "Live api documentation," in Proceedings of the 36th International Conference on Software Engineering, 2014, pp. 643–652.

[63] G. Uddin and F. Khomh, "Opiner: an opinion search and summarization engine for apis," in 2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE). IEEE, 2017, pp. 978–983.

[64] B. Lin, F. Zampetti et al., "Pattern-based mining of opinions in q&a websites," in 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE). IEEE, 2019, pp. 548–559.

[65] T. Zhang, G. Upadhyaya, A. Reinhardt, H. Rajan, and M. Kim, "Are code examples on an online q&a forum reliable?: a study of api misuse on stack overflow," in 2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE). IEEE, 2018, pp. 886–896.

[66] A. Reinhardt, T. Zhang et al., "Augmenting stack overflow with api usage patterns mined from github," in Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 2018, pp. 880–883.

[67] J. Zhou and R. J. Walker, "Api deprecation: a retrospective analysis and detection method for code examples on the web," in Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering, 2016, pp. 266–277.

[68] N. Zhang, Q. Huang, X. Xia, Y. Zou, D. Lo, and Z. Xing, "Chatbot4qr: Interactive query refinement for technical question retrieval," IEEE Transactions on Software Engineering, vol. 48, no. 4, pp. 1185–1211, 2022.