Essential Sentences for Navigating Stack Overflow Answers

Sarah Nadi
University of Alberta
Edmonton, AB, Canada
nadi@ualberta.ca

Christoph Treude
University of Adelaide
Adelaide, SA, Australia
christoph.treude@adelaide.edu.au

***Abstract*—Stack Overflow (SO) has become an essential resource for software development. Despite its success and prevalence, navigating SO remains a challenge. Ideally, SO users could benefit from highlighted navigational cues that help them decide if an answer is relevant to their task and context. Such navigational cues could be in the form of *essential sentences* that help the searcher decide whether they want to read the answer or skip over it. In this paper, we compare four potential approaches for identifying essential sentences. We adopt two existing approaches and develop two new approaches based on the idea that contextual information in a sentence (e.g., "if using windows") could help identify essential sentences. We compare the four techniques using a survey of 43 participants. Our participants indicate that it is not always easy to figure out what the best solution for their specific problem is, given the options, and that they would indeed like to easily spot contextual information that may narrow down the search. Our quantitative comparison of the techniques shows that there is no single technique sufficient for identifying essential sentences that can serve as navigational cues, while our qualitative analysis shows that participants valued explanations and specific conditions, and did not value filler sentences or speculations. Our work sheds light on the importance of navigational cues, and our findings can be used to guide future research to find the best combination of techniques to identify such cues.**

I. Introduction

With more than 18 million questions and 28 million answers as of October 2019, Stack Overflow (SO) has become a critical knowledge asset for the software development industry. However, navigating the amount of data available on Stack Overflow is challenging. While lots of previous work focused on identifying the most relevant threads for a given query [1], [2], [3], [4], navigation is not complete after a suitable thread has been identified. In a survey with 72 developers from two IT companies, Xu et al. [5] identified "the answers in long posts are hard to find" as one of the challenges. Almost 6.5 million questions (37% of all questions) have more than one answer, and the average length of an answer is 789 characters. Identifying useful information from this amount of data is not trivial. Thus, in this paper, we focus on the next stage of the navigation process. Specifically, we assume that the user has already identified the relevant threads and now needs to navigate them to identify relevant information.

Stack Overflow threads that contain many and/or long answers often discuss alternatives or opinions [6], [7], [8]. A user browsing such threads may want to quickly spot certain information about an answer to decide if it fits their needs. We refer to this information as *essential sentences*, because clearly seeing these sentences can help the user quickly decide if they should skip over this answer or carefully read it. The question is how can we identify such essential sentences?

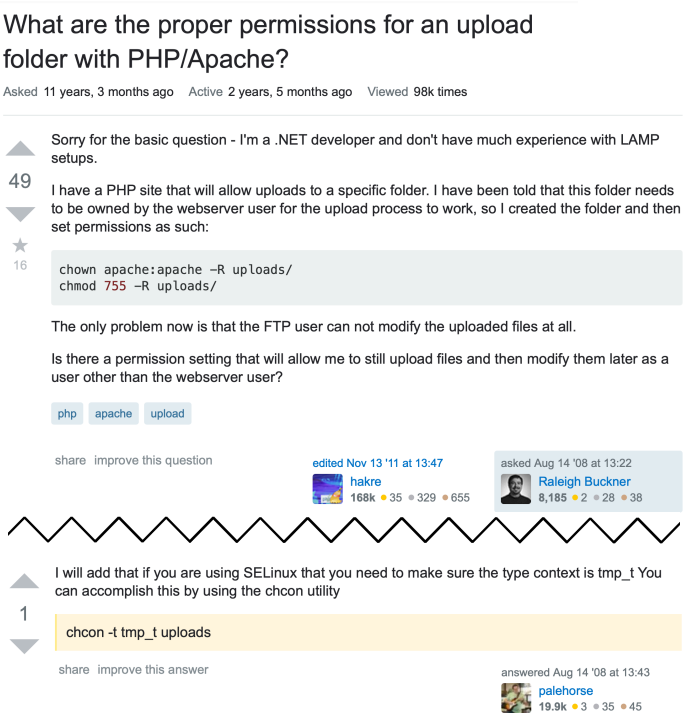


Fig. 1: Motivating Example from SO Thread 10990. The sentence mentioning SELinux is an essential sentence for navigating this thread. Our work compares four approaches for automatically identifying such sentences.

We could think of two possibilities from existing work. The first, `wordpattern`, is a pattern-based approach used to identify indispensable information in software documentation [9]. The second is `lexrank`, which is a standard text summarization algorithm that identifies the most important sentence(s) in a document. If we assume that the developer has a certain context in mind when navigating information (e.g., a particular platform, technology stack, or non-functional requirement), then another possibility for identifying essential sentences is looking for conditional sentences. For example,

Figure 1 shows a screenshot of Stack Overflow question 10990 along with one of its answers 11025. At the time of writing, this answer appears as the sixth out of eight answers in Stack Overflow's default ordering of answers by votes, i.e., the user would have to scroll past five other answers, including an accepted one, to see the sentence which indicates that this answer might be relevant to their particular context: For users using Security-Enhanced Linux (SELinux), it is important to know that type context needs to be `tmp_t` as part of setting the permissions for an upload folder. Thus, *"I will add that if you are using SELinux..."* is an essential sentence which if highlighted could help with navigating the information.

In this paper, we compare four approaches for identifying essential sentences in Stack Overflow answers. The first two are `wordpattern` and `lexrank` mentioned above. The other two approaches are both built on the above intuition that contextual information is often expressed as a condition in the sentence. Thus, the third approach, `simpleif`, simply relies on identifying any sentence with an if condition, or in other words *conditional sentences*. The fourth approach `contextif` aims to identify the subset of conditional sentences that specifically express a relevant condition or that provide contextual information, such as "*..if you are using SELinux*" from Figure 1. We evaluate 76 sentences identified by the four approaches through a survey with 43 participants. Our survey answers the following research questions:

- RQ_1 What navigation challenges do SO users face?
- RQ_2 Can highlighting navigational cues help SO users?
- RQ_3 Are essential sentences identified by different approaches perceived as helpful navigational cues?
- RQ_4 What information is available in essential sentences?

Our participants indicate that sifting through information on Stack Overflow to find the most suitable answer they can adapt to their specific context is a challenge. The results also suggest that highlighting relevant information, including contextual information, may help. Our quantitative results show that while `lexrank` has the highest percentage of highly rated sentences, each technique finds a different set of highly rated sentences. This suggests that there is no single technique that is clearly suitable for identifying essential sentences. However, our qualitative analysis shows that the majority of positively rated essential sentences contained explanations or specific conditions, which can help future work create more techniques for identifying essential sentences for navigation cues.

II. Related Work

To the best of our knowledge, there is no existing work on the extraction of essential sentences to be used as navigational cues. However, there is a lot of relevant work on identifying various types of information in Stack Overflow or software documentation, which we review in the following.

a) Identifying relevant information on Stack Overflow: Gottipati et al. [10] argue that it is hard for developers to find relevant answers in question-and-answer forums. They introduced a semantic search engine to recover relevant answers. The motivation of our work follows a similar thought, but at a finer level of granularity: Not only is it hard to find relevant answers on SO, but it is also challenging to navigate the information in these answers. Nasehi et al. [11] found that explanations accompanying SO code examples are as important as the examples themselves, further motivating our work on helping users navigate these textual explanations to identify relevant information.

Ye et al. [12] and Zou et al. [13] investigated the types of interrogatives (e.g., how-to, what, and where) in questions and answers and leverage this information to re-rank search results. Bagheri and Ensan [14] propose a semantic tagging approach based on Wikipedia data to improve Stack Overflow tags, eventually improving the search process. Soliman et al.'s [15] approach on the other hand is more domain-specific, focusing on how architects search for architecturally relevant information on Stack Overflow. CROKAGE by Silva et al. [16] takes the description of a programming task and provides a comprehensive solution for this task by searching multiple threads. In contrast, our work focuses on the navigation of a single thread, with the goal of identifying navigational cues. Finally, Opiner [17] and POME [8] identify Stack Overflow sentences that specifically contain opinions about Application Programming Interfaces (APIs), with the end-goal of recommending an API that matches a specific aspect (e.g., performance). In our work, we focus on sentences that help developers navigate answers regardless of whether they contain a positive or negative sentiment.

b) Identifying relevant information in software documentation: In an approach to bridge documentation from different sources, Treude and Robillard [18] augmented API documentation with insight sentences from Stack Overflow, i.e., sentences that are related to a particular API type and that provide insights not contained in the API documentation of that type. While some insight sentences may also be navigational cues, the application is fundamentally different: we focus on highlighting navigational cues in Stack Overflow threads instead of extracting them to complement a different information source; we do not rely on API documentation to identify sentences. However, given the somewhat similar goals, we use their same baselines. Specifically, we use the `wordpattern` approach by Robillard and Chhetri [9], who argued that information useful to programmers can be buried in irrelevant text, making it difficult to discover. To overcome that, they detected fragments of API documentation potentially important to a programmer, based on a tool which supports both information filtering and discovery. We also use the same `lexrank` summarization technique they compare to.

Outside of SO, Petrosyan et al. [19] use supervised text classification based on linguistic and structural features to discover tutorial sections explaining a given API type. Their approach uses supervised text classification. Jiang et al. [20] presented a similar approach, discovering two important indicators to complement traditional text-based features, namely co-occurring APIs and knowledge-based API extensions. They later refined their work [21] by identifying APIs in tutorial fragments and replacing ambiguous pronouns and variables

with related ontologies and API names. Specifically focusing on the challenge of navigating software documentation, Treude et al. [22] automatically extracted tasks from software documentation and suggested them to developers. Tian et al. [23] developed a bot to answer API questions given API documentation as an input. Zhong et al. [24] inferred specifications from API documentation by detecting actions and resources through machine learning.

Different to these approaches which are generally aimed at more structured documentation, such as API specifications, we focus on identifying navigational cues in SO threads.

III. Identifying Essential Sentences

In this paper, we compare four techniques for identifying essential sentences in Stack Overflow answers. Essential sentences can be used as navigational cues that allow a user to easily find the information they are looking for.

The first two techniques we use are `wordpattern` and `lexrank`, which have been used as the baselines in previous work on identifying insight sentences for API documentation [18]. Since both techniques try to identify various forms of valuable information, which may in turn be relevant for navigating Stack Overflow threads, we evaluate their application in our context of identifying essential sentences.

We also develop two new techniques that rely on the concept of conditions in sentences. The first, `simpleif`, simply identifies all sentences with the word "if", regardless of the contents of the condition. The second technique, `contextif`, finds conditional sentences whose conditional phrase contains relevant technical context. We develop several heuristics to determine whether the condition in the sentence is useful. We now present the details of all four techniques, first discussing the common pre-processing steps we apply for all of them.

A. Common Pre-processing Steps

Given a SO thread, we use `BeautifulSoup` [25] to process each answer's html code. We identify all paragraphs by searching for `<p>` tags. For each paragraph, we replace all html links (identified by the `<a>` tag) with the word `LINK` to enable the processing of the sentence through natural language processing tools later. For similar reasons, we also replace all in-text code (identified by the `<code>` tag) with the word `CW`. Afterwards, we use the Stanford CoreNLP toolkit [26] to identify the sentences in each paragraph. All techniques work on the same set of identified sentences. When parsing a paragraph using CoreNLP, we use the part-of-speech tagger (pos) and parse annotator, which provide full syntactic analysis using both the constituent and the dependency representations. Such annotators extract the tree structure of the sentence, allowing us to later identify any conditional phrases.

B. Technique 1: `wordpattern`

As a potential technique for identifying essential sentences, we use the work by Robillard and Chhetri [9], who developed a set of 360 word patterns for identifying sentences containing *indispensable* knowledge in documentation pages. The word patterns they created typically consist of a set of words and a code word that must appear in a sentence. An example of such a word pattern is {should, value, CW, be}. This pattern indicates that these three words must appear in the sentence, as well as any code word. An example matching this pattern is *"Providing the `contentType` parameter with the value of `json` will tell jQuery that the response should be json and it should auto parse it before giving it to a callback."*

Given a sentence, as identified through the common pre-processing steps, we lemmatize all words in the sentence to increase the chances of a pattern match. Since some users may use code words such as a class or a function name in their text without necessarily formatting it as code using the `<code>` html tag, we also use the list of regular expressions from Treude and Robillard's work [18] to identify additional code elements in the sentence and replace them with `CW` to further increase the chance of matching code words. We then look for the existence of any word pattern by searching for *all* the words in the pattern list, including `CW`. If the sentence matches any of the given 360 patterns, we mark this sentence as a `wordpatttern` sentence.

C. Technique 2: `lexrank`

Text summarization identifies the most important sentences from a given document [27]. We could think of navigational cues as sentences that contain important information relevant to their surrounding sentences, and thus interpret the task as a text summarization task. Lexrank [27] is a commonly used unsupervised text summarization approach. We use `lexrank` here as another potential technique for identifying navigational cues in a thread. We apply the common pre-processing steps and then lemmatize all words in the sentences. Finally, we pass the whole pre-processed thread to an existing open-source implementation of lexrank [28], and indicate the number of sentences it should return (1 sentence, see Section IV-A). For example, for thread 50957609, lexrank identifies the sentence "*You're not looping over an Array, you are looping over the properties of an Object with a for...in loop*".

D. Technique 3: `simpleif`

Essential sentences may contain contextual information which may often be expressed in the form of conditions, such as that shown in the example in the introduction. Thus, as a third potential technique for identifying essential sentences, we propose `simpleif`, which identifies all sentences that simply have the word "if" in them and thus contain a conditional phrase. We apply the common pre-processing steps and then keep sentences with the word "if" in them. An example `simpleif` sentence from thread 52853048 is *"If not, you could insert an extra trailing value, e.g. null, if it doesn't hurt."*

E. Technique 4: `contextif`

`simpleif` finds all conditional sentences. In `contextif`, the goal is to automatically identify conditional sentences that carry technical context and are useful. We do this based on a set of heuristics that we identified through manually analyzing 118 randomly selected conditional sentences from our corpus.

(1) Conditional Phrase Extraction: Given a `simpleif` sentence, we extract the conditional phrase from the parse tree produced by CoreNLP. Specifically, we look for subordinate clauses (SBAR) with an "if" in the left subtree, and then extract the conditional phrase from the simple declarative clause that is in the corresponding right subtree. For example, the conditional phrase of *"...., which will outperform the sync method if your server is under load"* is "*if your server is under load*." We then identify all nouns in the conditional phrase, *load* and *server* in the example sentence. We also treat any code elements in the sentence, again identified by the regular expressions from Treude and Robillard [18], as nouns. We then compare this noun list to the set of all Stack Overflow tags, with the goal of ensuring that the condition has some technical context. If any of the nouns are also SO tags, then we proceed to the next steps. Otherwise, we discard this sentence. In our example sentence, both nouns are also SO tags.

(2) Grammatical-relationship filtering: Based on our manual analysis, we noticed that the sentence's grammar structure can often be an indication of its usefulness. We thus follow intuition from Wang et al. [29] about using grammatical dependencies to identify higher quality sentences from Stack Overflow. We keep only sentences whose conditional phrase (1) has a verb phrase where the verb has a dependency on a noun (e.g., *if you want a good UI*) or (2) where the "if" has a direct dependency on a noun (e.g., *if file exists*).

(3) Final heuristic filtering: As a final step, we further remove sentences: ending in a question mark, with first person or "you" references but no modal verb (e.g., "if you look at the data"), containing unsure phrases (e.g., *I'm not sure if that's...*) or with conditions surrounded by parentheses. An example `contextif` sentence from thread 52703976 is *"If your expected value is an array, consider using map, especially if there will always be a value"*.

IV. Evaluation Setup

In this section, we describe the setup we use to evaluate the sentences identified by each of the four techniques.

A. Thread Selection

We select `json` as our subject domain, because it is a general data exchange format that is used in multiple programming languages and technologies. This increases our chance to get meaningful ratings, as opposed to using a specific technology that only few participants would be familiar with. We use the StackExchange API to find `json`-tagged threads that (1) have a question score of zero or more to filter out questions that have been explicitly marked as negative and (2) have been asked between March 29, 2018 to March 29, 2019.

The search returned 29,420 threads. We consider only threads that have a minimum of two answers, to ensure there are at least two alternative answers such that highlighting essential sentences makes sense. This leaves us with 7,920 threads. We processed 19,427 answers for these threads, with a mean of 2.45 answers per thread, and some threads having up to 13 answers. We run all techniques on all text from these answers, processing a total of 68,331 sentences. In the end, we identified 1,200 `wordpattern` sentences, 3,441 `simpleif` sentences, and 761 `contextif` sentences.

Since evaluating thousands of sentences in a survey is infeasible, we sample the threads for evaluation. To avoid being biased towards one technique, our criteria for selecting a thread for evaluation is that all techniques detected at least one sentence in that thread. `lexrank` needs the number of sentences to select for a summary, which guarantees that it will always identify at least one sentence per thread. Thus, we look only at the results of `wordpattern`, `contextif`, and `simpleif` for our sample selection. We identified 79 threads for which all three techniques detected at least one sentence. From these, we randomly select 20 threads for evaluation. To avoid bias, we want to balance the number of sentences being evaluated for each technique. Therefore, we first randomly select 20 threads. We then look at the number of total sentences identified from each of the three techniques in those 20 threads. If there is a big imbalance (e.g., a thread with 6 `wordpattern` sentences vs. 3 `contextif` and 3 `simpleif` or a thread with 5 `simpleif` sentences and only one sentence from each of the other techniques), we replace the thread with another randomly selected one until we get a reasonable and balanced number of sentences in each technique. To avoid over-burdening participants, we also focus on threads with at most a total of 5 sentences selected by all approaches. To determine the number of sentences to provide to the `lexrank` algorithm, we use the median number of sentences detected by each technique for these 20 threads. Since this median is 1 sentence, we configure `lexrank` to select one sentence per thread.

Our final 20 selected threads, along with their descriptive statistics, are available on our artifact page [30]. The minimum number of answers per thread was 2, median 3, and maximum 7. Table I shows the number of sentences identified by each technique across the 20 selected threads. Note that the same sentence may be identified by multiple techniques. In total, we have 76 unique sentences for evaluation in the survey. As shown in Figure 2, only 13 out of the 76 unique sentences are identified by more than one technique. Furthermore, the majority of sentences identified by each category of techniques are unique to that technique, suggesting that these techniques detect different kinds of information.

B. Survey Design

We design a custom web application that shows highlighted sentences within the context and GUI of a Stack Overflow thread. Participants see a page that contains the full thread, but without the extra controls/buttons/information on SO pages to reduce unrelated clutter. When they start the survey, participants are provided with an information page that describes the study (including ethics clearance information) and then proceed to an instructions page that explains their task. We now describe the flow of the survey.

c) Background Questions: Participants first see the background questions page, with the following questions.

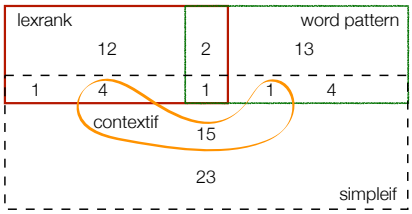


Fig. 2: Venn diagram illustrating the overlap between the 76 sentences identified across the four techniques

(BQ_1) *Is developing software part of your job?* Yes, no
(BQ_2) *What is your job title?* Free text
(BQ_3) *For how many years have you been developing software?* Free text
(BQ_4) *What is your area of software development?* Free text
(BQ_5) *How would you rate your json expertise?* No experience at all using json, Beginner, Intermediate, Expert
(BQ_6) *Have you used Stack Overflow to search for information before?* Yes, no
(BQ_7) *Have you contributed to Stack Overflow before? (questions, answers, comments, discussion etc.)* Yes, no

d) Sentence Review Questions: After answering the background questions, participants proceed to review three threads plus one quality gate thread (explained shortly). The sentences identified by the four techniques are highlighted in each thread. All sentences are highlighted in the same color and format, regardless of the technique that detected them to avoid any bias. Thus, each participant evaluates sentences from all four techniques without actually knowing which technique they are evaluating. Once participants click on a highlighted sentence, the following questions appear in the right margin beside the sentence (as shown in Figure 3).

(SR_1) *Which of the following statements best describes this highlighted sentence?* (a) The sentence is meaningful and provides important/useful information needed to correctly accomplish the task in question[1], (b) The sentence is meaningful, but does not provide any important/useful information to correctly accomplish the task in question, (c) The sentence does not make sense to me. Note that we ask this question to differentiate wording/grammar issues from actual usefulness.
(SR_2) *Given this highlighted sentence, indicate whether you agree with the following statement "When reading this thread, I would like to be able to quickly locate this sentence"* (a) strongly agree, (b) agree, (c) neither agree or disagree, (d) disagree, (e) strongly disagree.
(SR_3) *Given this highlighted sentence, indicate whether you agree with the following statement "Highlighting this sentence helps me navigate to relevant solutions and disregard irrelevant solutions"* (a) strongly agree, (b) agree, (c) neither agree or disagree, (d) disagree, (e) strongly disagree.
(SR_4) *Please justify your above ratings.* Free text.

[1]Refers to the question being asked in the current SO thread

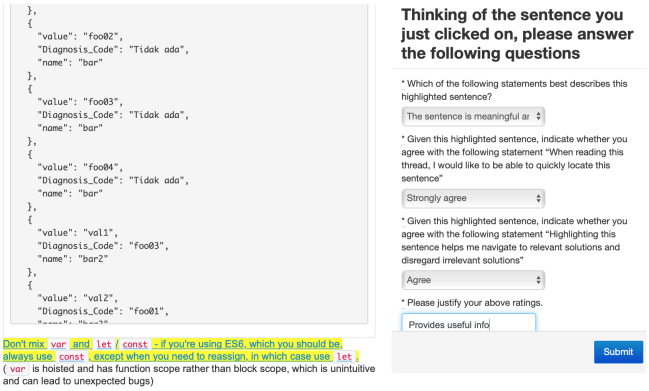


Fig. 3: Sentence questions displayed to survey participants

As a *quality gate*, we fix one thread that we selected beforehand with the following sentence highlighted: *"Hope this helps."* We expect participants who are not randomly answering the questions to rate this sentence negatively since it does not provide any useful information. The quality gate thread appears in a random order for each participant and looks the same as any of the other threads.

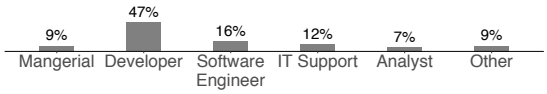
e) Exit Questions: After evaluating four threads, participants proceed to the following final set of exit questions.

(EQ_1) *Thinking of the highlighted sentences you observed, and specifically the ones you thought were useful/important, are there any specific properties of these sentences that affected your rating?* Free form
(EQ_2) *Thinking of the highlighted sentences you observed, and specifically the ones you thought were NOT useful/important, are there any specific properties of these sentences that affected your rating?* Free form
(EQ_3) *Thinking of your general experience as a software developer: after you have identified a relevant Stack Overflow thread that you need to look at, what challenges (if any) do you encounter in navigating the information in the thread?* Free form
(EQ_4) *Assuming Stack Overflow could highlight certain information for you in a given thread, what kind of information would you like to see?* Free form
(EQ_5) *Some of the sentences you saw were conditional sentences (i.e., sentences with an if clause) that contained conditions related to programming languages, technologies, operating systems, or situations a developer would face. Do you think highlighting such sentences could be useful to find relevant information in a thread more quickly?* Free form

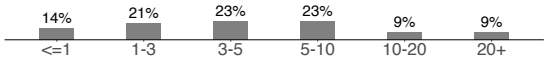
While the threads each participant sees are randomly selected from the thread pool and appear in a random order, we use a balancing algorithm to ensure that we get at least 3 ratings per thread. Thus, our algorithm tries to select from the threads with the lowest number of responses first.

C. Participant Recruitment

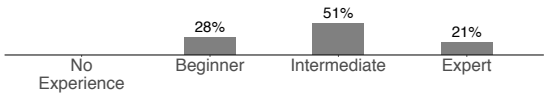
We used Amazon Mechanical Turk [31] (MT) to recruit participants, using the premium option of "Employment Industry – Software & IT Services" as a required qualification



(a) Job title



(b) Years of software development



(c) What is your experience using json?

Fig. 4: Participant Background

and a compensation of USD$3. Our web application generates a unique token for each participant to submit on MT. We accepted only responses which provide this token and where the participant has (at least) minimal knowledge of JSON (BQ_5) and has used SO to search for information before (BQ_6). Each participant can answer the survey only once.

We also performed post-filtering of the responses to ensure that we consider results only from participants who understood the task. We use our quality gate thread to filter out participants. Specifically, we look at the answers to (SR_3) for the sentence "*Hope this helps.*" and filter out all participants who did not answer "Disagree" or "Strongly disagree".

To ensure that there are no issues with our survey application, we first conducted a pilot study with 5 MT participants. We fixed some technical issues that arose from the pilot. We do not include the pilot data in the results presented in this paper, and only base our results on the full run.

D. Data Analysis

Qualitative Analysis: For RQs 1, 2, and 4, we use qualitative methods, specifically open coding [32], to analyze participants' responses to the exit questions, as well as the information contained in positively rated sentences. For each analysis, one author first manually created meaningful codes [32] for a random sample of the data. Then, using these defined codes, both authors coded 10% of the data points using the defined coding scheme. If our inter-rater reliability, calculated using Fuzzy Kappa [33] to account for potentially multiple labels per data point, was high ($>= 0.75$), the two authors continued coding the rest of the data. If not, they discussed the codes first and found potentials for code merging.

Quantitative Analysis: To answer RQ2, we need to compare the ratings of (SR_1), (SR_2), and (SR_3) for the different techniques. A high median rating suggests that the majority of participants thought that this sentence is "good" according to the criteria the question asks about. Thus, to evaluate the effectiveness of each technique, we look at the percentage of sentences identified by that technique which received a high rating. We use the question nature to determine what a "high rating" means. For Question (SR_1), we convert the choices to ratings, where 1 corresponds to not meaningful, 2 corresponds to meaningful but not useful/helpful, and 3 corresponds to meaningful and useful/helpful. Since we are interested in meaningful and helpful sentences, we consider a sentence with a median score equal to 3 as a highly rated sentence. For Questions (SR_2) and (SR_3), we convert the choices to ratings from 1 (Strongly disagree) to 5 (Strongly agree). For those questions, we are interested in sentences where participants at least agree with the statement. Thus, we consider sentences with median score $>= 4$ as highly rated.

V. Survey Summary Statistics

We accepted submissions from 59 participants. We filter out 16 participants based on our quality gate thread, and use the remaining 43 participants for our results.

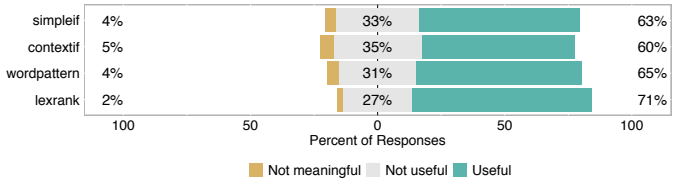
Figure 4 shows the distribution of the background of the 43 survey participants. We group related job titles under common categories. As Figures 4a and 4b show, our participants work in various occupations related to the technology sector and have a wide range of years of software development experience. Figure 4c shows that the majority of our participants had intermediate knowledge of json; recall that we did not accept submissions from participants with no json experience.

As per our load balancing algorithm, all threads received at least 3 ratings. The median number of ratings per thread was 7, with a maximum of 10. The median number of ratings per sentence per technique was also 7, and the distribution of the number of ratings per sentence across the four techniques was similar, giving us confidence that the number of ratings will not implicitly bias our results towards a certain technique.

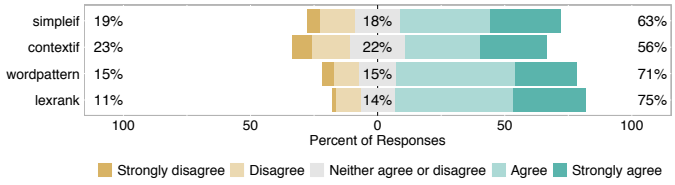
VI. RQ1: Navigation Challenges

In RQ_1, we want to understand what kind of navigation challenges users face, if any. We look at the SO navigation challenges participants mention in (EQ_3). The codes we use for these responses, along with the number of responses per code, are: *too much information to navigate* (13), *no problems* (11), *not easy to adapt information to my specific problem* (6), *outdated information* (5), *SO feature* (2), *need to combine multiple solutions from different threads* (2), *duplicate content* (1), and *archived threads* (1). Our inter-rater agreement for this coding task was 0.76.

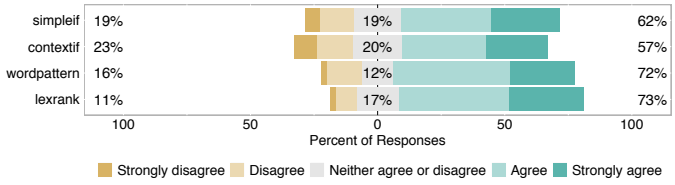
While 11 participants have no problems navigating Stack Overflow, more participants complain about the amount of information they need to navigate (e.g., "*It's often finding the useful info. There are some threads where there's so much useless stuff*"). This further motivates our research for helping developers navigate Stack Overflow. The other challenges mentioned are also quite interesting. For example, the difficulty in adopting solutions to a user's specific context or the need to combine multiple solutions from different threads (e.g., "*I usually have to piece together multiple stack overflow threads to find a solution to a particular situation*"). Highlighting the context under which a solution is relevant may help alleviate the former challenge.



(a) (SR_1)



(b) (SR_2)



(c) (SR_3)

Fig. 5: Distribution of responses per technique per question

Answer to RQ1: Challenges in SO navigation include sifting through lots of information, adapting solutions to the user's context, and combining solutions from multiple threads.

VII. RQ2: Potential of Highlighting Information

In RQ2, we explore the potential of highlighting information on SO. To do so, we analyze the responses to two questions: (EQ_4) where we explicitly asked participants what information they want highlighted and (EQ_5) which explicitly asks participants whether they think highlighting conditional information could be helpful. Note that we put (EQ_5) as the last question of the survey to avoid any bias.

The codes we use for the responses in (EQ_4) are: *most relevant solution* (12), *direct answer* (10), *relevant explanations* (8), *tips* (4), *code* (4), *confirmed information* (2), *step-by-step solution* (2), *no highlighting* (2), and *summary* (1). Our inter-rater agreement for this coding task was 0.85. Naturally, most participants wanted the most-relevant solution (e.g., "*I would like the most relevant solution to be highlighted*") or a direct answer (e.g., "*Any links to programs or interfaces that are the direct solution*") to be highlighted. It is interesting to note here that the most-relevant solution to the question in the thread may not necessarily be the most relevant one to the user. Thus, the term "most relevant" is relative to the specific problem the user is looking to solve, which may have slight differences to the one in the thread. The third code in our list shows that not only are direct solutions relevant, the explanations that are needed to understand them are very important to participants. Previous work investigating code examples also found similar results about the importance of explanations [11].

For (EQ_5), twenty participants said yes to highlighting conditional information; 11 said maybe. The following are the codes for participant justifications (if any): *yes, good to know cases* (9), *yes, provides easy/quick navigation* (6), *yes, depends on stack* (4), *yes, with granularity selection* (1), and *no, may miss big picture* (1). Our inter-rater agreement was 0.82. These comments suggest that the majority of participants think that highlighting conditional sentences is a good idea (e.g., "*Knowing a solution is specific to a technology, etc. will let me quickly decide if I should continue reading or look elsewhere*"). Participants say that it is good to know the different cases when finding a solution, that it makes navigation easier, and that such highlighting is useful if the user's technology stack matches the highlighted information. Such comments match our intuition and motivation for this work. One participant had an interesting suggestion of allowing users to select the granularity level of the highlighting they want (e.g., programming languages, operating systems etc.): "*It might be nice to have some granularity to the highlighting that a user could specify (i.e., Do you want to highlight x, y, z?)*".

Answer to RQ2: Participants would find highlighting the most relevant solution useful. Highlighting conditional information may be useful, especially to understand the various cases for a solution.

VIII. RQ3: Technique Comparison

To compare the performance of the four techniques for RQ3, we perform a quantitative analysis of the ratings from Questions (SR_1), (SR_2), and (SR_3). Figure 5 shows the distribution of responses for the sentences identified by each technique for each question. We use a two-sided unpaired Wilcoxon signed-rank test to compare the rating distribution of the various techniques and calculate effect size using $r = Z/sqrt(N)$ [34]. We use a Benjamini & Hochberg (BH) p-value adjustment measure to account for multiple comparisons, and use $\alpha = 0.05$. Table I also shows the percentage of the identified sentences that were highly rated by participants.

General Quality of Sentences: In (SR_1), we asked participants to select a statement that best describes the highlighted sentence. While we found no statistically significant differences between the rating distributions of all four techniques, Table I shows that `simpleif` had the highest number of highly rated sentences (32). This is expected given that `simpleif` naturally identifies more sentences given its simple criteria. However, a higher percentage of the sentences identified by `lexrank` end up being highly rated by participants, in this case seen as meaningful and useful.

Desire to Locate Sentences: In (SR_2), we ask participants if they would like to quickly locate the highlighted sentence when navigating the thread. Again, we find no statistically significant differences between the distribution of ratings across the techniques. However, as shown in Table I, the number of highly rated sentences in each technique differs between (SR_1) and (SR_2). We point out that the highly rated sentences in (SR_2) are not necessarily a subset of those in (SR_1) due to the nature of the question. Question (SR_1) asked participants to rate the sentence on its own rather than for

TABLE I: Number of sentences identified by each technique, and percentage of those sentences that were highly rated

Technique	Number of Sentences	(SR_1) Highly Rated	(SR_2) Highly Rated	(SR_3) Highly Rated
`simpleif`	49	32 (65 %)	30 (61 %)	29 (59 %)
`contextif`	20	13 (65 %)	9 (45 %)	11 (55 %)
`wordpattern`	21	16 (76 %)	17 (81 %)	16 (76 %)
`lexrank`	20	16 (80 %)	14 (70 %)	16 (80 %)
Total Unique Sentences	76	51 (67 %)	49 (64 %)	48 (63 %)

a specific purpose/goal, while (SR_2) specifically asks about the goal of quickly locating a given sentence. One example of a sentence that was positively rated in (SR_1) but negatively rated in (SR_2) is *"And if so it returns the default value"* from answer 50523464. This sentence is part of a paragraph that explains the logic of a piece of code that preceded it. While the sentence is meaningful and may be helpful as an explanation of the problem and provided code, someone navigating this thread would not be keen about quickly locating this sentence per se since it does not give any information about the context/topic/value of the answer. On the other hand, an example of a sentence that was less positively rated in (SR_1) but positively rated in (SR_2) is *"You're not looping over an Array, you are looping over the properties of an Object with a for...in loop.'* from thread 50958104. While this sentence does not provide a direct solution to the problem, it explains why the problem occurs. As one participant elaborates *"The highlighted sentence gives an explanation to the problem but is not suggesting any solution to the problem. But, still highlighting it might help me in navigating to that solution which might also contain the actual solution to the problem."*

Table I shows that `wordpattern` has the highest proportion of highly rated sentences for (SR_2). It is also worth noting that `contextif` has a smaller number of highly rated sentences when compared to `simpleif`. Given that `contextif` is a subset of `simpleif`, this suggests that some of our filtering criteria may have been too strict, resulting in filtering out sentences that are valued by participants. We discuss this phenomenon in more detail in Section X, as well as look more closely at the overlap between the sentences identified by the four techniques.

Helpfulness In Identifying Relevant Solutions: In (SR_3), we ask participants whether the given sentence helps them quickly identify relevant solutions and disregard irrelevant ones. This is the most important question for our goal of providing useful navigational cues to Stack Overflow users. Table I shows the percentage of sentences identified by each technique that were highly rated by participants. Again, there may be sentences that are highly rated in this question but not in other ones. For example, the following sentence *"But if you have an `include` property in your `tsconfig.json`:"* from thread 51494250 has a median rating $>= 4$ for (SR_3) but not (SR_2). As one participant explains their rating for this sentence, *"This explains one situation in which this person's solution might be helpful"*. Thus, when specifically asked if this sentence would help users quickly identify relevant solutions and disregard irrelevant ones, more participants rated this sentence more positively.

Following this logic, as shown in Table I, we can see that the number of `contextif` sentences that were positively rated increased between (SR_2) and (SR_3), suggesting that these conditional sentences do indeed serve the purpose of differentiating between solutions better. However, `lexrank` still outperforms the other techniques in terms of having the highest ratio of its sentences rated highly. Note that we find a statistically significant difference between the rating distribution of `contextif` and `lexrank` ($p = 0.045$), but with a very small [35] effect size of 0.155.

Answer to RQ3: When considering a median rating of at least "Agree", a higher percentage of `lexrank`'s sentences are perceived as helpful navigational cues.

IX. RQ4: Helpful Navigation Information

We now present the results of our qualitative analysis to answer RQ4. We present the details of the final set of codes used, the inter-rater agreement for each question, and the number of instances per code. For all questions, we find that the majority of instances were assigned exactly one code, with a max number of codes of 3.

Sentence Analysis: To understand the information participants look for in navigation cues, we look at the 48 unique highly rated sentences for Question (SR_3). Our goal is to identify the kind of information in the sentence, or its role in the thread. Thus, we also use participants' rating justifications from (SR_4) to guide us during the process.

The set of codes we use, along with the number of instances per code in parentheses is as follows: *explanation* (23), *specific condition* (19), *API note* (17), *direct solution* (5), *label* (5), and *other* (1). Our inter-rater agreement kappa score is 0.78. Looking at the top three codes, an example of an explanation sentence is *"If it returned a falsy value (i.e. 0) then the value of the other operand of || will be returned (which is sorting by timestamp)."* Here, the poster is explaining some concept related to the question, or in other cases related to the provided solution. For explanation sentences, participants use justifications that involve the words "explain", e.g., *"That's the simplest and clear[est] explanation."* An example of a sentence highlighting a specific condition is *"You can use session storage if you want the data to be retrieved once PER SESSION or local storage if you want to have better control of the data's 'expiration'."* Participants provided justifications for this category of sentences such as *"This single sentence answers the question thoroughly, with two separate options."* Sentences containing API notes typically provided extra information about a specific method. An example is *"Another thing to notice, is that `require` is synchronous, so if your JSON is specially large, the first time you instantiate `MyClass` the event loop will be blocked."* One of the participants comments on this sentence saying, *"This*

type of information would allow me to avoid a future bug or unexpected behavior, and directly relates to the question."

General reasons for high ratings: In (EQ_1), we asked participants about what generally made them rate a sentence highly. Recall that participants answer this question at the end of the survey, allowing them to reflect on all sentences they evaluated. The set of codes we use, along with the number of instances per code in parentheses is as follows: *direct solution* (16), *explanation* (13), *relevance of info* (11), *code/lib* (6), *well-written* (5), *SO info* (2), *alternate solution* (1), and *warning* (1). Our inter-rater agreement kappa score is 0.86.

Our codes show that participants explicitly mention the existence of a direct solution or explanation the most. The quality of the sentence itself, in terms of how well-written or concise it is, also affected their rating. Several participants find that a sentence is useful when it mentions a specific library, or piece of code (e.g., an API). It is interesting to see that only one participant explicitly mentions alternatives, while based on our previous analysis of what information is contained in highly rated sentences, 19 out of the 48 analyzed sentences contained some form of a specific condition. We speculate that participants do not necessarily refer to these as conditional sentences, or alternative solutions, but instead find them useful because they mention particular contexts, whether programming languages, library names, or APIs.

General reasons for low ratings: We also look at the answers to (EQ_2), which asked about general reasons for low ratings. The set of codes we use, along with the number of instances per code in parentheses is as follows: *not relevant to main issue* (23), *filler sentences* (7), *no meaningful information* (6), *speculation* (4), *not standalone* (2), *formatting* (1), *complicated* (1), *contains no code* (1), *generic* (1), and *other* (1). Our inter-rater agreement kappa score is 0.87.

As can be seen, sentences that did not contain information relevant to the main issue being asked in the thread, such as direct solutions, were seen as not useful. Filler sentences are farewell sentences or just social comments, such as people saying good bye or good luck at the end of a post. Sentences with no meaningful information of how to apply the given information to solve the problem were also rated low.

Answer to RQ4: The majority of highly rated sentences in (SR_3) contain explanations or specific conditions. The most mentioned factor affecting participants' sentence ratings is whether they directly solve the main issue in question.

X. Discussion

Our goal in this work is to investigate what kind of information is helpful for developers to navigate through potential solutions on Stack Overflow, and possible techniques to identify this information. Our work is the first to investigate essential sentences for the purpose of navigating technical information on Stack Overflow. Our qualitative results suggest that we are on the right track in thinking that contextual information may be useful. We believe that techniques that identify and highlight such contextual information can provide valuable navigation

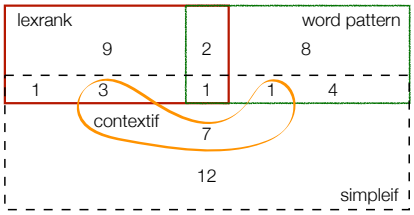


Fig. 6: Venn diagram illustrating the techniques corresponding to the 48 highly rated sentences in Question (SR_3)

cues on Stack Overflow. Identifying conditions and contexts related to a query can also help users contextualize their search and reach their target thread more easily. For example, if we identify that there is a big variation in answers to a given task based on the programming language or operating system, we can prompt the user to select one of the existing contexts during their search. Finally, this information can also help with automated thread or answer tagging on Stack Overflow, which would make both manual navigation as well as automated search engines more effective.

Instead of re-inventing the wheel, we looked for existing techniques that may potentially find relevant information for our purposes, namely `lexrank` and `wordpattern`. Our results show that both techniques are indeed promising, since a high proportion of the sentences they identify is highly rated by participants. Additionally, given the idea of contextual information, we focused on conditional sentences since conditions may likely signal specific contexts. In the `contextif` technique, we designed some heuristics to identify meaningful conditions. However, the results show that our heuristics were rather strict, where they filtered out many `simpleif` sentences that ended up being highly rated by participants. We plan to investigate if ignoring parts of speech in the condition helps and if other means for differentiating technology related conditions can be used [36], [37], [38].

Not all sentences that help users navigate a thread are necessarily providing context information. For example, the following sentence, detected by `lexrank`, "*You are trying to cast to* `[String: Any]`*, but you have an array of* `[String: Any]` *because your response enclosed in* `[]` *brackets.*" was highly rated. Since it does not contain any conditional phrases, it was not picked up by `simpleif` or `contextif`. However, participants found it relevant since it provided a direct explanation of the problem being faced in this thread. We conclude that there may be different types of navigation issues at hand. The first is a user who knows the particular thread that describes their exact problem and just wants to quickly find (and understand) the direct solution. In this case, a sentence such as the above one is what they need. The second is a user who is facing a similar problem but in a very specific context and is browsing threads to try to find someone who shares this context. In this case, sentences that contextualize the solution (and thus likely contain some form of condition) are more useful. Our survey did not differentiate these types of users since we did not provide them a concrete task to do. Thus, we conclude that both types of information

are useful, but may depend on the use case.

Figure 6 illustrates this point further. We show all the 48 sentences that were highly rated in Question (SR_3), and the corresponding techniques that identified them, including any overlap. As can be seen, the majority (29/48 = 60%) of these highly rated sentences are conditional sentences (i.e., `simpleif` sentences), while the remaining sentences are identified by `wordpattern` or `lexrank`. Additionally, with the exception of a very small effect size for the difference between `lexrank` and `contextif`, we found no statistically significant difference between the four techniques. Thus, we can conclude that there is no single technique that is clearly a "winner", and a future direction could be investigating meaningful combinations as well as additional techniques.

Finally, it is worth noting that the highly rated sentences did not necessarily exist in the accepted answer for the thread or in the highest scoring answers. For example, the sentence *"I'd only put* `require` *inside the constructor if it was a dynamic dependency, [...]"* had a median rating of 5, even though the corresponding answer's score was 1. This supports our belief that developers often look for information relevant to their context, which may exist in various answers in a thread.

XI. Threats to Validity

Internal Validity: There may be additional essential sentences useful for navigational cues that none of the techniques detected. Thus, our comparison of techniques is a relative one with respect to the four examined techniques.

The choice of the threshold used for the definition of "highly rated" affects the results we obtain. We chose a threshold that indicates a high rating, based on the meaning of each question.

Participants may provide random answers to the survey questions, including the quality gate question in which it will be correct in 40% of the time. This is always a threat to any survey, and we tried to mitigate it as much as possible.

We rely on the Stanford CoreNLP toolkit for the processing of sentences. Any inaccuracies in it may lead to inaccuracies in the presented techniques.

When sampling from Stack Overflow, we did not apply any filtering based on answer score, because an essential sentence does not necessarily exist only in highly scored answers. As discussed in Section X, we found highly rated sentences in answers with score 1. Additionally, we found no Spearman rank correlation between any of the ratings and answer score.

Construct Validity: The two existing techniques we used, `wordpattern` and `lexrank`, were not designed for the purpose of SO navigation cues. Our goal is to investigate how applicable existing techniques are to identifying navigational cues and to identify future steps.

Our survey participants were not provided with a concrete task to think of when evaluating threads. This was intentional since creating a concrete task with a specific context that does not bias the results towards the conditional sentences in the thread is difficult. Without a concrete task/context, a controlled experiment setting to compare the highlighting of essential sentences to current navigation techniques is infeasible. Participants might rate the sentences differently if they were provided with a specific context. Our survey mitigates that by asking about different aspects of the sentences, as well as eliciting free-form feedback that we qualitatively analyzed.

To ensure that we can statistically compare the techniques and to avoid burdening participants, we evaluated threads with a similar number of extracted sentences by each technique. This sampling may mean that we evaluated on a unique subset of threads since the majority of threads had sentences identified by only some of the techniques.

External Validity: There are other language patterns that encode conditional information, e.g., "For Linux, use ..." Thus, the current `simpleif` and `contextif` sentences do not necessarily represent all contextual information. To investigate the idea of using conditional information for navigational cues, we started with the simple form of sentences with "if".

Our results are based on the evaluation of 20 json threads, with 76 highlighted sentences, by 43 participants, and may not generalize beyond that. That said, our survey participants have diverse background, with varying expertise levels with json as well as varying occupations and programming experience, which gives us confidence that our results are not biased towards the views of a particular population sample.

XII. Conclusion

Given the amount of information available on Stack Overflow, identifying which part of which answer is suitable for a user's task and context is difficult. In this paper, we compared four techniques to identify essential sentences providing navigational cues for guiding users to the most suitable (part of an) answer. Our results show that a high percentage of the sentences identified by a text summarization technique, `lexrank`, are highly rated. However, we also show that the other techniques find many additional highly rated sentences, which suggests that there is no clear silver bullet. We find that most of the highly rated sentences contain explanations or specific conditions. The results support our intuition that conditional or contextual information can help users quickly navigate information on Stack Overflow. We plan to investigate additional heuristics to further differentiate between useful contextual information and irrelevant conditions. We publicly share all the code and data from our work [30], which also includes the collected sentence ratings from our survey evaluation that can be used to guide future work.

Acknowledgements

Thanks to Benyamin Noori for early investigation of conditional sentences and to Samer Al Masri for implementing the survey website. This research was undertaken, in part, thanks to funding from the Canada Research Chairs program and the Australian Research Council's Discovery Early Career Researcher Award (DECRA) funding scheme (DE180100153). This work was inspired by the International Workshop series on Dynamic Software Documentation, held at McGill's Bellairs Research Institute.

REFERENCES

[1] L. Ponzanelli, G. Bavota, M. Di Penta, R. Oliveto, and M. Lanza, "Mining StackOverflow to turn the IDE into a self-confident programming prompter," in *Proceedings of the Working Conference on Mining Software Repositories*, 2014, pp. 102–111.

[2] L. B. L. de Souza, E. C. Campos, and M. d. A. Maia, "Ranking crowd knowledge to assist software development," in *Proceedings of the International Conference on Program Comprehension*, 2014, pp. 72–82.

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

[4] R. F. G. Silva, C. K. Roy, M. M. Rahman, K. A. Schneider, K. Paixao, and M. de Almeida Maia, "Recommending comprehensive solutions for programming tasks by mining crowd knowledge," in *Proceedings of the International Conference on Program Comprehension*. Piscataway, NJ, USA: IEEE Press, 2019, pp. 358–368.

[5] B. Xu, Z. Xing, X. Xia, and D. Lo, "AnswerBot: Automated generation of answer summary to developers' technical questions," in *Proceedings of the International Conference on Automated Software Engineering*, 2017, pp. 706–716.

[6] G. Uddin and F. Khomh, "Automatic mining of opinions expressed about APIs in Stack Overflow," *IEEE Transactions on Software Engineering*, 2019, to appear.

[7] A. Zagalsky, D. M. German, M.-A. Storey, C. G. Teshima, and G. Poo-Caamaño, "How the R community creates and curates knowledge: an extended study of Stack Overflow and mailing lists," *Empirical Software Engineering*, vol. 23, no. 2, pp. 953–986, 2018.

[8] B. Lin, F. Zampetti, G. Bavota, M. Di Penta, and M. Lanza, "Pattern-based mining of opinions in Q&A websites," in *Proceedings of the International Conference on Software Engineering*. IEEE Press, 2019, pp. 548–559.

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

[10] S. Gottipati, D. Lo, and J. Jiang, "Finding relevant answers in software forums," in *Proceedings of the International Conference on Automated Software Engineering*, 2011, pp. 323–332.

[11] S. M. Nasehi, J. Sillito, F. Maurer, and C. Burns, "What makes a good code example?: A study of programming Q&A in StackOverflow," in *Proceedings of the International Conference on Software Maintenance*, 2012, pp. 25–34.

[12] T. Ye, B. Xie, Y. Zou, and X. Chen, "Interrogative-guided re-ranking for question-oriented software text retrieval," in *Proceedings of the International Conference on Automated Software Engineering*, 2014, pp. 115–120.

[13] Y. Zou, T. Ye, Y. Lu, J. Mylopoulos, and L. Zhang, "Learning to rank for question-oriented software text retrieval (t)," in *Proceedings of the International Conference on Automated Software Engineering*, 2015, pp. 1–11.

[14] E. Bagheri and F. Ensan, "Semantic tagging and linking of software engineering social content," *Automated Software Engineering*, vol. 23, no. 2, pp. 147–190, 2016.

[15] M. Soliman, A. Rekaby Salama, M. Galster, O. Zimmermann, and M. Riebisch, "Improving the search for architecture knowledge in online developer communities," in *Proceedings of the International Conference on Software Architecture*, 2018, pp. 186–195.

[16] R. F. Silva, C. K. Roy, M. M. Rahman, K. A. Schneider, K. Paixao, and M. de Almeida Maia, "Recommending comprehensive solutions for programming tasks by mining crowd knowledge," in *Proceedings of the International Conference on Program Comprehension*, 2019, pp. 358–368.

[17] G. Uddin and F. Khomh, "Opiner: an opinion search and summarization engine for APIs," in *Proceedings of the International Conference on Automated Software Engineering*, 2017, pp. 978–983.

[18] C. Treude and M. P. Robillard, "Augmenting API documentation with insights from Stack Overflow," in *Proceedings of the International Conference on Software Engineering*, 2016, pp. 392–403.

[19] G. Petrosyan, M. P. Robillard, and R. De Mori, "Discovering information explaining API types using text classification," in *Proceedings of the International Conference on Software Engineering - Volume 1*, 2015, pp. 869–879.

[20] H. Jiang, J. Zhang, X. Li, Z. Ren, and D. Lo, "A more accurate model for finding tutorial segments explaining APIs," in *Proceedings of the International Conference on Software Analysis, Evolution, and Reengineering*, 2016, pp. 157–167.

[21] H. Jiang, J. Zhang, Z. Ren, and T. Zhang, "An unsupervised approach for discovering relevant tutorial fragments for APIs," in *Proceedings of the International Conference on Software Engineering*, 2017, pp. 38–48.

[22] C. Treude, M. P. Robillard, and B. Dagenais, "Extracting development tasks to navigate software documentation," *IEEE Transactions on Software Engineering*, vol. 41, no. 6, pp. 565–581, 2015.

[23] Y. Tian, F. Thung, A. Sharma, and D. Lo, "APIBot: Question answering bot for API documentation," in *Proceedings of the International Conference on Automated Software Engineering*, 2017, pp. 153–158.

[24] H. Zhong, L. Zhang, T. Xie, and H. Mei, "Inferring resource specifications from natural language API documentation," in *Proceedings of the International Conference on Automated Software Engineering*, 2009, pp. 307–318.

[25] "Beautifulsoup python library," https://www.crummy.com/software/BeautifulSoup/bs4/doc/.

[26] "Stanford CoreNLP," https://stanfordnlp.github.io/CoreNLP/.

[27] G. Erkan and D. R. Radev, "Lexrank: Graph-based lexical centrality as salience in text summarization," *Journal of Artificial Intelligence Research*, vol. 22, no. 1, pp. 457–479, 2004.

[28] "Open-source lexrank implementation," https://github.com/linanqiu/lexrank.

[29] E. Wong, J. Yang, and L. Tan, "Autocomment: Mining question and answer sites for automatic comment generation," in *Proceedings of the International Conference on Automated Software Engineering*. IEEE, 2013, pp. 562–567.

[30] "Online artifact page," https://doi.org/10.6084/m9.figshare.10005515.

[31] "Amazon mechanical turk," https://www.mturk.com/.

[32] K.-J. Stol, P. Ralph, and B. Fitzgerald, "Grounded theory in software engineering research: A critical review and guidelines," in *Proceedings of the International Conference on Software Engineering*, 2016, pp. 120–131.

[33] A. P. Kirilenko and S. Stepchenkova, "Inter-coder agreement in one-to-many classification: Fuzzy kappa," *PLOS ONE*, vol. 11, pp. 1–14, 2016.

[34] A. Field, J. Miles, and Z. Field, *Discovering statistics using R*. Sage publications, 2012.

[35] S. S. Sawilowsky, "New effect size rules of thumb," *Journal of Modern Applied Statistical Methods*, vol. 8, no. 2, p. 26, 2009.

[36] D. Yan and S. Guo, "Leveraging contextual sentences for text classification by using a neural attention model," *Computational Intelligence and Neuroscience*, vol. 2019, 2019.

[37] S. W. K. Chan and J. Franklin, "Dynamic context generation for natural language understanding: A multifaceted knowledge approach," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 33, no. 1, pp. 23–41, 2003.

[38] M. Nassif, C. Treude, and M. Robillard, "Automatically categorizing software technologies," *IEEE Transactions on Software Engineering*, 2018, to appear.