Discovering, Explaining and Summarizing Controversial Discussions in Community Q&A Sites

Xiaoxue Ren*†‖, Zhenchang Xing ¶, Xin Xia‡✓, Guoqiang Li§, Jianling Sun*
*College of Computer Science and Technology, Zhejiang University, Hangzhou, China
†Ningbo Research Institute, Zhejiang University, Ningbo, China
‖PengCheng Laboratory, Shenzhen, China
¶Research School of Computer Science, Australian National University, Canberra, Australia
‡Faculty of Information Technology, Monash University, Melbourne, Australia
§School of Software, Shanghai Jiao Tong University, Shanghai, China
xxren@zju.edu.cn, zhenchang.Xing@anu.edu.au, xin.xia@monash.edu.au, li.g@sjtu.edu.cn, sunjl@zju.edu.cn

***Abstract*—Developers often look for solutions to programming problems in community Q&A sites like Stack Overflow. Due to the crowdsourcing nature of these Q&A sites, many user-provided answers are wrong, less optimal or out-of-date. Relying on community-curated quality indicators (e.g., accepted answer, answer vote) cannot reliably identify these answer problems. Such problematic answers are often criticized by other users. However, these critiques are not readily discoverable when reading the posts. In this paper, we consider the answers being criticized and their critique posts as controversial discussions in community Q&A sites. To help developers notice such controversial discussions and make more informed choices of appropriate solutions, we design an automatic open information extraction approach for systematically discovering and summarizing the controversies in Stack Overflow and exploiting official API documentation to assist the understanding of the discovered controversies. We apply our approach to millions of java/android-tagged Stack overflow questions and answers and discover a large scale of controversial discussions in Stack Overflow. Our manual evaluation confirms that the extracted controversy information is of high accuracy. A user study with 18 developers demonstrates the usefulness of our generated controversy summaries in helping developers avoid the controversial answers and choose more appropriate solutions to programming questions.**

***Index Terms*—Controversial discussion, Stack Overflow, Open information extraction, Sentence embedding**

I. Introduction

In community Q&A sites like Stack Overflow, users commonly rely on community-curated quality indicators, such as the answer accepted by the question asker and the community votes on the answer, to choose the appropriate solution to the question, especially when the question has multiple answers. However, we observe that many answers, even the accepted or high-vote answers, may be wrong, less optimal or out-of-date. Such answer issues may be warned in the comments on the answers and/or other answers to the same question.

Table I presents such as example. The question asks How do I fix android.os.NetworkOnMainThreadException?. The accepted answer suggests "Run your code in AsyncTask", As of 11th May, 2019, this question has been viewed 1,090,842 times and the accepted answer has received 2413 votes. This indicates that the solution in the accepted answer could be adopted by many developers in their applications. Unfortunately, this accepted answer violates an official API caveat of using AsyncTask, as pointed out by a comment on the accepted answer and the other answer that receives only 144 votes and is ranked 4th by community votes. In this work, we refer to answers being criticized as controversial answers and comments/answers criticizing other answers as critique posts. Critique posts sometimes reference the links of official API documents or discuss (may paraphrase) API caveats in official documentation to support or explain their critiques. A controversial answer, its critique posts (one or more) and relevant explanatory API links/caveats (if any) constitute a controversial discussion thread.

There has been much work on investigating question and answer quality in community Q&A sites [1]–[4], but controversial discussions have not been systematically studied. We conduct an empirical study of controversial discussions in java/android-tagged Stack Overflow questions and answers (see details in Section II). Among 2,469,536 java/android-tagged questions in the latest Stack Overflow data dump (released on 4th March, 2019), we find that 18.85% questions have at least one controversial answer. Among all 3,899,653 answers, 21.48% of them are controversial. We found that 410,063 controversial discussion threads are API related. Among these 410,063 API-related controversial discussion threads, our explain-controversy approach (see Section III-C) identifies that 28.17% threads discuss relevant API caveats in official Java and Android documentation and that 35.62% threads reference the links of official API documents.

The large scale of controversial discussions in Stack Overflow is unsurprising in that there is almost no barrier to participate in Stack Overflow question answering. Due to the lack of certain API caveat knowledge, users are likely to contribute some solutions that they believe to be good but actually violate API usage caveats (e.g., Table I), or are less optimal (e.g., Table II), or are out-of-date (e.g., Table III) Other users may warn readers of such problematic answers by criticizing them in answer comments and other answers. Although such critiques signal the potential controversies of

✓Corresponding author.

certain solutions, they may be overshadowed by other answer-quality signals (e.g., the answer acceptance by the question asker or the high votes by many less knowledgeable users).

In this paper, we design an automatic open information extraction (OpenIE) approach for systematically discovering, explaining and summarizing controversies in community Q&A discussions. Our approach detects critique posts by mining critique sentences. It infers controversial answers being criticized by either explicit answer references or a text-summary-and-matching based method. A controversial answer and its critique posts are summarized as a controversial discussion thread. Our approach detects API mentions in a controversial answer or a critique post against an API inventory crawled from official API reference. For the mentioned APIs, it further extracts the discussed API caveats and the referenced API document links in the controversial answer and the critique posts, as official references to explain the discovered controversies. We apply the API caveat mining approach [5] to extract API caveat sentences from official API documentation, and design a weighted word-embedding based sentence matching method which is robust to match the paraphrased API-caveat explanations in the answers/comments and the original API-caveat sentences in official documentation, even in face of the lexical gap between the sentences.

We apply our approach to Java/Android-tagged Stack Overflow questions and answers in the latest Stack Overflow data dump. We use the statistical sampling method [6] to evaluate the accuracy of each information extraction step of our approach at 5% error margin and 95% confidence level. The identified critique posts and controversial answers reach the accuracy 99.5% and 95.1% respectively. The accuracy of identifying API-related controversial answers and critique posts is 99.2%. The accuracy of extracting the API-caveat sentences from official Java/Android API documentation is 99.2%, which is consistent with the results reported in [5]. The accuracy of extracting the discussed API-caveat sentences in the controversial answers and critique posts is 90.4%.

We conduct a user study in which 18 Java developers are asked to identify the most appropriate solutions to eight Stack Overflow questions by reading the answers to these questions on Stack Overflow. 18 participants are divided into three equal-size groups: the first group (G_1) read answers without any controversial reminders; the second group (G_2) read answers with only a general warning statement on the controversial answers; the third group (G_3) read answers augmented with the detailed controversy summary produced by our approach (see Section III-D). The third group achieves the highest correctness. Five participants in G_3 answer six or seven questions correctly, while seven participants in G_1 and G_2 answer four or less questions correctly. Compared with the group G_2, the group G_3 completes the task 15.4% faster.

In this paper, we make the following contributions:

- To the best of our knowledge, we are the first to systematically investigate the controversial discussions in Stack Overflow, including their scale, impact, typical cases and connection to API documentation.
- We design an automatic approach for discovering and summarizing controversial discussions in Stack Overflow and exploiting official API links/caveats as reference to explain the controversies.
- We conduct extensive experiments to evaluate the accuracy of our approach for controversy-related information extraction, as well as the usefulness of the generated controversy summary for identifying the most appropriate solutions in crowd answers to programming questions.
- We release the source code of our approach and the dataset of our evaluation and userstudy[1] to help other researchers replicate and extend our study.

II. Empirical Study of Controversial Discussions in Stack Overflow

Our empirical study on controversial discussions uses the latest Stack Overflow data dump (released on 4th March, 2019). In this study, we examine java/android-tagged questions and the answers to these questions. Our study reveals key insights of controversial discussions, including their scale, impact, typical cases and connection to API documentation.

The scale of controversies is significant. The data dump contains 2,469,536 java/android-tagged questions. These questions have 3,899,653 answers which have 5,525,257 comments. Using our controversy-discovery approach (see Section III-B1), we find that 465,399 (18.85%) questions have at least one controversial answer. 837,719 (21.48%) answers are controversial as they have been criticized by at least one comment or answer. We find 408,683 critique answers and 858,072 critique comments. 484,285 (57.81%) controversial answers have two or more critique comments/answers.

Wrong answer, less optimal answer and out-of-date answer are typical types of controversies. During the mining of sample critique sentences (see Section III-B) and the evaluation of our controversy-discovery approach (see Section IV), we empirically observe three typical types of controversies: wrong answer, less optimal answer and out-of-date answer. The accepted answer in Table I is a wrong answer because the suggested solution violates the API caveat of using ***AsyncTask***. The critique comment and answer in Table I paraphrase the API caveat of ***AsyncTask*** to explain their critiques. The accepted answer in Table II is a less optimal answer. It has two critique comments: one suggests "This answer doesn't help as others do. It shouldn't be accepted"; the other suggests "***ListPopupWindow*** is better option" and refers the readers to the API reference of ***ListPopupWindow*** for more detail. Table III shows an example of partially controversial answer. The answer is mostly correct, but the use of one API ***setDrawerListener*** is out-of-date because ***setDrawerListener*** is deprecated. The critique comment/answer points out this API deprecation and suggests an alternative API. The critique answer also references relevant API link as supporting resource.

[1]The replication package can be downloaded at: https://github.com/goodchar/Controversy-summary-of-Stack-Overflow-posts

Community-curated quality indicators (e.g., accepted answer and vote) are not reliable to identify controversial answers. Only 12,733 (1.52%) controversial answers have negative votes. 446,504 (53.30%) controversial answers are marked as accepted answer and/or have the highest positive votes among all answers to a question. Our user study reveals that information seekers on Stack Overflow often trust these community-curated answer-quality indicators when no direct controversy reminders are present (see Section V-B).

The presence of critique posts does not mean that the controversies are easily discoverable and accessible to information seeker. By comparing the vote number of the controversial answers in the two data dumps (released on 18th March, 2018 and on 4th March, 2019), we find that the vote number of 529,271 (63.18%) controversial answers increases over time. Take the controversial accepted answer in Table I as an example. The vote on this answer was 2131 as of 18th March, 2018, and increased to 2379 one year later, and further increased to 2413 as of 4th May, 2019. The increasing vote on the controversial answers indicates that many developers still regard the controversial answers as good solutions to their problems (there could be even more developers who adopt the controversial solutions but do not make a vote), even though these answers have been explicitly criticized by some comments on them or other answers.

Official API documentation is often referenced to explain or backup the critiques. Among 837,719 controversial discussion threads (each is uniquely identified by a controversial answer), 410,063 (48.95%) of them are API related (i.e., mentioning at least one Java/Android API). Among these API-related controversial discussion threads, 146,064 (35.62%) explicitly reference the links of official Java and Android API documents. In addition, using our API-caveat discussion extraction method (see Section III-C), we find that 115,496 (28.17%) controversial discussion threads discuss some API caveats from official API documentation, even though they may not explicitly reference the API document links.

III. Approach

The large-scale controversies in Stack Overflow and the interference of these controversies on developers motivate us to design an approach that turns latent controversial discussions into salient, concise, semi-structured controversy warnings to help developers notice the controversies and judge their validity. As shown in Fig. 1, the raw input to our approach includes Stack Overflow posts (i.e., answers and comments) and official API documentation. Our approach has three main steps: discover controversies including controversial answers and critique posts (Section III-B), explain API-related controversies in terms of API document links referenced and API caveats discussed in controversial discussions (Section III-C), and summarize controversies as salient, concise, semi-structured controversy warnings (Section III-D).

TABLE I
Wrong Answer (Violating API Caveat)

Question	How do I fix android.os.NetworkOnMainThreadException?
Viewed times	1,090,842 (as of 11th May, 2019)
Accepted but controversial answer (vote=2431): This exception is thrown when an application attempts to perform a networking operation on its main thread. Run your code in AsyncTask ... ***Critique comment on accepted answer:*** This is exactly the wrong answer...AsyncTask should not be used for network activity, because it's tied to the activity ... ***Critique answer (vote=144):*** The accepted answer has some significant downsides. It is not advisable to use AsyncTask for networking ...AsyncTask's created as non-static inner classes have an implicit reference to the enclosing Activity object ... This reference prevents the Activity from being garbage collected until the AsyncTask's background work completes ... these short-term memory leaks can become a problem ... ***Explanatory API caveat of AsyncTask:*** When using a subclass of AsyncTask to run network operations, ... don't create a memory leak where the Activity that is referenced by the AsyncTask is destroyed before the AsyncTask finishes its background work.	

TABLE II
Less Optimal Answer

Question	How to set Spinner default value to null?
Viewed times	109,613 (as of 11th May, 2019)
Accepted but controversial answer (vote=78): Only if there is no data. If you have 1+ items in the SpinnerAdapter, the Spinner will always have a selection... ***Critique comment-1 on accepted answer: (vote=2)*** ListPopupWindow is better option in this case. By replacing spinner from textView and calling ListPopupWindow.show() on textView click shows the same behaviour with no drawback of default item selection. For more detail see this https://developer.android.com/reference/android/widget/ListPopupWindow.html. ***Critique comment-2 on accepted answer:*** This answer doesn't help as others do. It shouldn't be accepted ***Explanatory API document link:*** https://developer.android.com/reference/android/widget/ListPopupWindow.html.	

TABLE III
Out-of-Date Answer

Question	How to detect that the DrawerLayout started opening?
Viewed times	38,850 (as of 11th May, 2019)
Accepted but controversial answer (vote=68): There are 2 possible ways to do that: 1) Use onDrawerSlide(View drawerView, float slideOffset) callback ... mDrawerLayout.setDrawerListener ... 2) Use onDrawerStateChanged(int newState) callback ... mDrawerLayout.setDrawerListener ... ***Critique comment on accepted answer:*** setDrawerListener is deprecated. Changed to addDrawerListener. ***Critique answer (vote=32):*** Currently accepted answer by Pavel Dudka is already deprecated. Please use mDrawerLayout.addDrawerListener() method instead to set a listener https://developer.android.com/reference/android/support/v4/widget/DrawerLayout.html ***Explanatory API caveat of setDrawerListener:*** setDrawerListener (DrawerLayout.DrawerListener listener) ... was deprecated in API level 24.1.0. Use addDrawerListener (DrawerListener) ***Explanatory API document link:*** https://developer.android.com/reference/android/support/v4/widget/DrawerLayout.html	

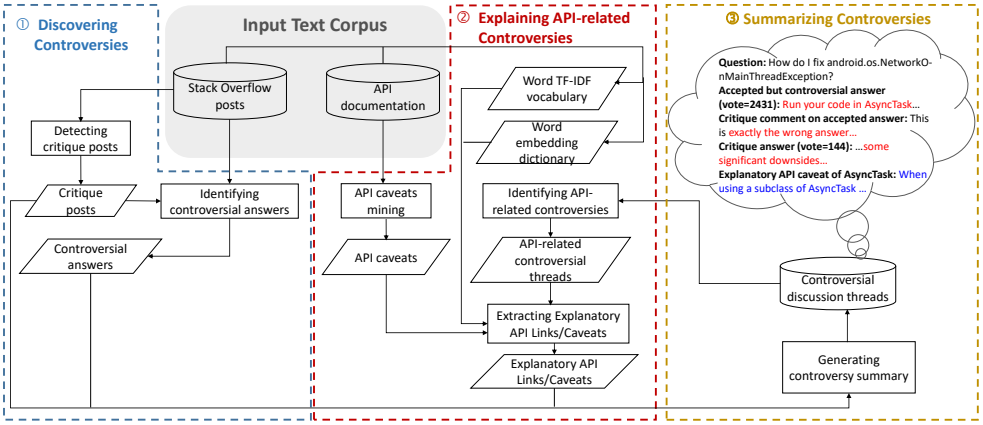

Fig. 1. The Framework of Our Approach

A. Input Documentation

Our approach discovers the controversies in Stack Overflow posts (i.e., answers and comments) and finds the explanations of the controversies in official API documentation. We use Stack Overflow data because Stack Overflow is the most popular social information seeking platform for computer programming. We focus on java/android-tagged Stack Overflow questions in this work. Stack Overflow post content (including answer body and comment content) can be directly retrieved from the Stack Overflow data dump (the latest data dump released on 4th March, 2019 is used in this study).

For the official API documentation, we consider both API references and API tutorials. We crawl Java and Android API references from Android API reference and Java SDK API specification respectively, and crawl Java and Android API tutorials from Android Developer Guides and Java Tutorials respectively. We discard web page navigation content and retain the main API reference and tutorial content.

Both Stack Overflow post content and API reference/tutorial content are contained in webpages. We follow the steps that are commonly used for preprocessing web content [7], [5], [8], [9]. We preserve textual content but remove HTML tags. Stand-alone code snippets (e.g., those in <pre> tag in Stack Overflow posts) are removed because our focus is on controversial discussions in text. But code elements in natural language sentences are kept in order to retain the sentence integrity. The text content is then tokenized. Software text contains API mentions, such as ***AsyncTask***, ***ListPopupWindow***, ***mDrawerLayout.addDrawerListener()*** and ***addDrawerListener(DrawerListener)*** in the answer/comment/API-caveat examples in Table I, Table II and Table III. API tokens usually contain special characters such as ".", "()", "[]", "_". In face of these special characters, a general text tokenizer usually breaks one API token into several tokens, such as "addDrawerListener", "(", "DrawerListener", ")" for the API token "addDrawerListener(DrawerListener)". This will disturb the sentence integrity. Therefore, we adopt the software-specific tokenizer [9] for extracting API mentions in natural language sentences. Our software-specific tokenizer retains the integrity of API tokens during text tokenization. After tokenization, we use Stanford CoreNLP [10] to split texts into sentences, which are the basic text units in our work.

B. Discovering Controversies

First, our approach detects critique posts by mining critique sentences, and then it infers controversial answers being criticized based on explicit answer references in critique posts and a text-summary-and-matching method.

1) Detecting critique posts: Two of our authors randomly select 5000 answers and 5000 comments to read, from which we identify in total 3206 critique sentences with Cohen's Kappa more than 0.800. All these critique sentences contain some critique indicators. We classify the critique indicators into three categories: judgment, sentiment and opinion. Table IV lists the top-4 most frequent critique indicators for each category identified during our manual post examination.

On Stack Overflow, the similar sentence semantics are often expressed in many different ways [11]. Table V shows variant critique sentences with similar critique semantics.We can see that simply using a set of critique indicators to pattern-match critique sentences likely misses many critique posts in which the critiques are not expressed in the exact form of these critique indicators. To overcome this issue, we compile the 3206 critique sentences identified in our manual examination into a corpus of sample critique sentences. If a sentence is similar enough (the similarity score, which is computed with our pre-trained domain-specific word embedding dictionary (see part 2 in Fig. 1) > 0.8 in this work) to at least one sentence in the corpus of sample critique sentences, this sentence will be considered as a critique sentence. The sentence similarity is computed using the sentence matching method described in Section III-C3. If an answer or a comment contains at least one critique sentence, this answer or comment will be identified as a critique post.

2) Identifying controversial answers: Some critique posts have explicit references to the controversial answers being

TABLE IV
EXAMPLES OF CRITIQUE INDICATORS

Category	Top-4 Frequent Critique Indicators
Judgement	wrong answer, out of date, incorrect, not work
Sentiment	stupid, unfortunately, dislike, disagree
Opinion	better answer, a little bit vague, not think so, not sure

TABLE V
EXAMPLES OF VARIANT CRITIQUE SENTENCES

Critique Semantics	Variant Critique Sentences
Wrong Answer	Unfortunately the accepted answer is wrong.
	ERROR Make sure the Cursor is initialized correctly before accessing data from it.. checking in API 19.
	Doesn't work for me.
Less Optimal Answer	Rather than creating a Dialog with a list of Intent options, it is much better to use Intent.createChooser...
	Not the Perfect Solution.
	I'm not satisfied with it.
Out-of-date Answer	recyclerView.getChildPosition(v); is depricated now use recyclerView. indexOfChild(v);
	These answers were outdated for me, so here's how it worked out.
	It is long outmoded, and java.time the modern Java date and time API is so much nicer to work with..

criticized. If a critique post is a comment, the answer being commented is the controversial answer. We observe that a critique answer often references the answer being criticized by "this accepted answer" (e.g., Table I) or the author of the answer (e.g., Table III). If such explicit answer references exist in a critique answer, we analyze the answer metadata to identify the referenced controversial answer.

Even when a critique answer does not explicitly reference the controversial answer, we observe that a critique answer has to reference the content of the controversial answer and explains what problems it has and why. This implicit content reference usually makes the relevance between the critique answer and the controversial answer higher than their relevance to other answers (if any). Based on this observation, we develop a text-summary-and-matching method to identify the controversial answer that is not explicitly referenced in the critique answer. Note that when a question has only two answers, we heuristically identify the answer other than the critique answer as the controversial answer without the need to check explicit reference or content relevance.

Our text-summary-and-matching method first use TextRank [12] to extract the most important sentences (up to 5 sentences) in the critique answer and in each of the other answers to the same question. TextRank ranks the sentence importance by applying the PageRank algorithm [13] to a sentence correlation graph for a piece of text. Next, given the critique answer and one of the other answers to the same question, we use the sentence matching method described in Section III-C3 to compute the sentence similarity between each pair of the importance sentences of the two answers. And then, we use the bipartite graph matching algorithm [14] to determine the optimal sentence-matching pairs between the two answers. Finally, we average the sentence similarity of the optimal sentence-matching pairs as the similarity of the two answers. The answer with the highest similarity to the critique answer is identified as the corresponding controversial answer.

C. Explaining API-related Controversies

As an Q&A site for computer programming, many Stack Overflow posts are API related. As shown in Table I, Table II and Table III, when discussing API-related controversies, critique posts often reference official API document links or discuss API caveats from official API documentation to support their critiques. Furthermore, controversial answers (e.g., the accepted answer in Table I) may discuss some API usage highly relevant to certain API caveats in official documentation. Such API-link references and API-caveat discussions are an important resource for understanding the controversies and determining what problems a controversial answer may have and why. Therefore, our approach identifies API-related controversial answers and critique posts, and then extract explanatory API links/caveats from these answers/posts and the corresponding official documentation.

1) Identifying API-Related Controversies: If a controversial answer and/or its critique posts mention some API(s) (Java and Android APIs in this work), the controversy is considered as API related. To detect API mentions in Stack Overflow posts, we built an API inventory that stores the fully-qualified names of all Java/Android SDK APIs and the corresponding URLs of these APIs in official API reference websites (https://developer.android.com/reference/classes for Android SDK and https://docs.oracle.com/javase/8/docs/api/ for Java SDK).

Then, we detect API mentions in two ways. First, if a token in the post matches an API name in the API inventory, this token is considered as an API mention. As the examples in Table I, Table II and Table III show, it is common that a class is mentioned without package name, and a method is mentioned without package/class name or parameters. Therefore, we perform the partial name matching between the post tokens and the API names. Second, a post may not explicitly mention the API names but reference the API links in the API reference website. Therefore, we also examine each hyperlink in the post. If the domain name of a hyperlink matches the domain name of the API reference website, this hyperlink is considered as an API mention. We then search the API hyperlink in the API inventory to find the corresponding API.

2) Extracting Explanatory API Links/Caveats: Extracting explanatory API document links is straightforward. The API hyperlinks identified in the last step are explanatory API links relevant to the controversy. However, extracting explanatory API caveats is challenging because of the lexical gap between the paraphrased API-caveat explanations in Stack Overflow posts and the original API-caveat sentences in official documentation. Table VI shows some examples. Such lexical gaps render keyword-matching based method ineffective to determine the relevance between the paraphrased API-caveat explanations in Stack Overflow posts and the original API-caveat sentences in official documentation. Therefore, we

develop a weighted word-embedding based sentence matching method (see Section III-C3) to battle the lexical gap between the paraphrased and original API-caveat descriptions.

We adopt the API-caveat mining method developed by Li et al. [5] to extract API caveat sentences from Java and Android API documentation. Li et al. define an API caveat as a sentence that specifies some constraint, contract or guideline of API usage. They develop a set of sentence syntactic patterns for extract three general categories and 10 subcategories of API caveats - explicit (error/exception, recommendation, alternative, imperative, note), restricted (conditional, temporal), and generic (affirmative, negative, emphasis).

For each mentioned API in a Stack Overflow post, we rank the sentences in the post by their relevance to each of this API's caveat sentence mined by [5]. Although an API can have many caveats, we observe that usually only one API caveat is relevant to a specific question. Therefore, we use the pair of post sentence and official API-caveat sentence with the highest relevance score (the similarity metric > 0.8) as the explanatory API caveat for a particular API involved the controversy, for example, the sentences highlighted in blue in Table I. If a post mentions multiple APIs, we will find one explanatory API caveat for each mentioned API.

3) Weighted Word-Embedding Based Sentence Matching: We compute the relevance between a sentence in a Stack Overflow post and an official API caveat sentence based on word embeddings [15] and Term-Frequency-Inverse-Document-Frequency (TF-IDF) [16]. This sentence-matching method is also used to identify critique sentences similar to sample critique sentences in Section III-B1 and to identify the controversial answers that are not explicitly referenced in the critique posts in Section III-B2.

Word embedding techniques [17]–[19] assume that words appear in similar context tend to have similar meanings [20], [21]. By word embedding, words are no longer represented as sparse one-hot vectors, but are mapped to a real-valued dense vector space. Each dimension of a word vector represents a latent semantic or syntactic feature of the word. In this work, we adopt the continuous skip-gram model [17], [22], which is one of the state-of-the-art algorithms for learning word embeddings using a neural network model. We train the continuous skip-gram model using the text corpus of Stack Overflow posts and API documentation built in Section III-A. Following the previous work to learn word embeddings from Stack Overflow posts and API documentation [23], we set the dimension of word embeddings at 200.

TD-IDF is a statistical metric to evaluate the importance of a word in a document [24], [25]. By TD-IDF, the importance of a word is proportional to the number of times it appears in the document, but is inverse-proportional to the frequency of its occurrence in the corpus. We compute the TF-IDF metric of each word in the text corpus of Stack Overflow posts and API documentation built in Section III-A. Each Stack Overflow answer, comment or API document is a document. We assign a weight to a word t in a document d by $score_t = tf_{t,d} * idf_t$. $tf_{t,d}$ is computed by $n_{t,d} / \sum_k n_{k,d}$ where $n_{t,d}$ is the term frequency of the word t in the document d. We normalize $n_{t,d}$ by the total number of words $\sum_k n_{k,d}$ in the document. This prevents $tf_{t,d}$ from skewing toward long files, because the same word tends to have a higher frequency in a long document than in a short one, regardless of its importance. idf_t is computed by $log(|D|(1 + |d \in D : t \in d|)$ where $|D|$ is the total number of documents in the corpus and $|d \in D : t \in d|$ is the number of documents where the word t appears.

Finally, we compute the sentence embedding v_s of a sentence as the TD-IDF weighted average of the word embeddings of the words in the sentence, i.e, $v_s = \frac{1}{|s|} \sum_{t \in s} score_t * v_t$, where t is a word in the sentence s, $score_t$ is t's TD-IDF weight, v_t is t's word vector, and $|s|$ is the number of words in the sentence. We use the cosine similarity of the sentence embeddings of two sentences to measure the relevance between the two sentences.

D. Summarizing Controversies

The current form of the controversies in Stack Overflow is latent. First, critique comments/answers may be buried in many other comments/answers. Second, critique answers and controversial answers have no direct links in between. Third, critique sentences, controversy-related content in between critique posts and controversial answers, and explanations of API caveats are hidden in text. Such latent information organization makes it difficult to discover and understand the controversies in Stack Overflow Q&A discussions.

To facilitate the discovery and understanding of controversial discussions, we design a controversy-oriented way to organize and present controversial discussions. First, a controversial answer, its critique posts and relevant official API links/caveats (if any) are summarized in a semi-structured controversial discussion thread (similar to the examples shown in Table I, Table II and Table III). We use the text-summary-and-matching method described in Section III-B2 to extract relevant sentence pairs in between the controversial answer and its critique posts. Based on these relevant sentence pairs, we compose an excerpt for the controversial answer and each critique post. These excerpts help users quickly review the controversy-related content in the controversial answer and the critique posts. The controversial discussion thread lists all explanatory API document links as supporting resources to understand the controversies. If any explanatory API caveats are found in a post, the official API caveats will be listed under the post, and the corresponding explanations of the API caveats in the post will be added to the post excerpt.

When users are viewing a Stack Overflow question, our browser-plugin tool can display the generated controversy summary as an explicit and concise controversy warning to help users choose appropriate solution to the question. The critique sentences in the critique posts will be highlighted to attract the users' attention to the controversies. The API-caveat explanations in the post (if any) will also be highlighted to help users quickly judge the validity of the controversy against the relevant official API caveats.

TABLE VI
EXAMPLES OF LEXICAL GAP BETWEEN PARAPHRASED AND ORIGINAL API CAVEATS

NO.	API Caveats in Stack Overflow Posts	Original API caveats in API Documentation
1	... if you want to be able to add or remove elements from the returned list in your code, you'll need to wrap it in a new ArrayList. Otherwise you'll get an UnsupportedOperationException.	To implement a modifiable collection, the programmer must additionally override this class's add method (which otherwise throws an UnsupportedOperationException), and the iterator returned by the iterator method must additionally implement its remove method.
2	NoSuchMethodError happens if one class expects a method in another class (and was compiled with that method in place), but at runtime the other class does not have that method.	NoSuchMethodError Thrown if an application tries to call a specified method of a class (either static or instance), and that class no longer has a definition of that method.
3	And if we want an item to occupy full span we need to set isFullSpan when creating render info:	boolean isFullSpan() Returns whether this View occupies all available spans or just one.
4	The WifiP2pManager includes APIs that allow you to: Initialize your application for P2P connections by calling initialize()	The application needs to do an initialization with initialize (Context, Looper, WifiP2pManager.ChannelListener) before doing any p2p operation.

IV. ACCURACY OF CONTROVERSY EXTRACTION METHODS

Our approach is an open information extraction (OpenIE) method to extract controversy-related knowledge from textual Q&A discussions and API documentation. In this section, we evaluate the accuracy of the four key steps of our approach: detecting critique posts, identifying controversial answers, identifying API-related controversies, and extracting explanatory API caveats. We also evaluate the accuracy of the adopted API-caveat mining method [5] on our dataset.

A. Experimental Dataset and Evaluation Methods

We use the latest Stack Overflow data dump released on 4th March, 2019 in this study. This data dump has 2,469,536 java/android-tagged questions, 3,899,653 answers to these questions, and 5,525,257 comments on the answers. Using our controversy extraction method, we identify 837,719 controversial answers, 408,683 critique answers and 858,072 critique comments, 410,063 API-related controversial discussion threads, and 115,496 API-related answers and comments containing explanatory API caveats. Using the API caveat mining method [5], we extract 389,871 API caveat sentences from Java and Android API documentation.

As each step of our approach extracts a large number of data instances, we adopt a statistical sampling method [6] to examine the minimum number MIN of data instances for each step. This sampling method ensures that the estimated accuracy is in a certain error margin at a certain confidence level. We use the error margin 0.05 at 95% confidence level in our evaluation. Given the large number of data instances, MIN is approximately 384 at this statistical setting. Therefore, we examine 384 data instances for each step. For the identification of controversial answers, we consider only the controversial answers that are not explicitly referenced by the critique posts, because those that are explicitly referenced by the critique posts can be identified without errors. We do not need to examine the extracted explanatory API links because they are identified by straightforward domain name matching.

The two authors independently evaluate the accuracy of a data instance (binary decision: whether a critique post points out a controversy, whether a controversial answer is the actual answer being criticized by a critique post, whether an API-related answer or comment mentions the identified APIs, whether the identified explanatory API caveat is discussed in an answer or comment, and whether an API caveat from official documentation reveals certain API usage constraint, contract or guideline). We compute Cohen's Kappa [26] to evaluate the inter-rater agreement. For the data instances that the two authors disagree, they have to discuss and come to a final decision. Based on the final decisions, we compute the accuracy of each information extraction step.

Table VII reports the accuracy results. The columns $Acc1$ and $Acc2$ show the accuracy results determined by the two annotators independently, and the column $AccF$ is the final accuracy after resolving the disagreements. The column $Kappa$ is the inter-rater agreement.

B. Results - Discovering Controversies

The first two rows in Table VII report the accuracy of detecting critique posts and the accuracy of identifying controversial answer being criticized by a critique post. For the detected critique posts, the final accuracy is 99.5% and the Cohen's Kappa is 0.80 which indicate substantial agreement between the labeling decisions of the two annotators. Detecting critique posts is accurate because we manually identify a set of high-quality sample critique sentences. Furthermore, our sentence matching method can accurately identify variant critique sentences such as those in Table V. A small number of errors are caused by some complex sentences, such as the sentence "the User might not want to agree to those and didn't update intentionally.". Although "not agree to" is a common critique indicator, this sentence is not a critique sentence.

For the accuracy of identifying controversial answers, the final accuracy is 95.1% and the Cohen's Kappa is 0.76 which indicate substantial agreement between the two annotators. This result shows that our text-summary-and-matching method can accurately identify the controversial answers that are not explicitly referenced in the critique posts. Most matching errors are caused by the short controversial answers and/or the short critique answers. These short answers are usually not self-contained, but refer readers to some other online materials for more details. As such, they do not have enough content for reliable matching.

TABLE VII
ACCURACY RESULTS OF CONTROVERSY EXTRACTION STEPS

OpenIE Output	Acc1	Acc2	AccF	Kappa
CritiquePost	98.9%	99.5%	99.5%	0.80
ControversialAnswer	95.8%	94.0%	95.1%	0.76
APIRelatedPost	99.5%	99.2%	99.2%	0.80
APICaveatDiscussed	87.2%	91.4%	90.4%	0.73
APICaveatinOfficialDoc	99.0%	99.2%	99.2%	0.86

C. Results - Explaining Controversies

The last three rows in Table VII report the accuracy of identifying API-related answers/comments, the accuracy of identifying explanatory API caveats discussed in an answer or comment, and the accuracy of extracting API caveats from official documentation. For the accuracy of identifying API-related answers/comments, the Cohen's Kappa between the two annotators is 0.80, which indicates substantial agreement between the two annotators. The final accuracy after resolving the disagreements is 99.2%. This suggests that our API-mention detection method can accurately detect the mentions of Java and Android APIs in Stack Overflow discussions. A small number of errors are caused by ambiguous API class or method names, such as "job", "observable', "database" which can either be API mentions or common words.

For the accuracy of identifying explanatory API caveats discussed in an answer/comment, the Cohen's Kappa between the two annotators is 0.73, which indicates substantial agreement. The final accuracy is 90.4%. This high accuracy suggests that our sentence matching method can accurately identify API caveats discussed in an answer/comment even in face of the lexical gap between the paraphrased API-caveat explanations in the answer/comment and the original API caveat sentences in official documentation. The matching errors are usually caused by long sentences which have complex sentence semantics, in which some parts of the sentences match certain API caveats, but the whole sentences are not completely about the matched API caveats.

For the accuracy of extracting API caveat sentences from official API documentation, the Cohen's Kappa between the two annotators is 0.86, which indicates the almost perfect agreement. The final accuracy is 99.2%. This result is consistent with the accuracy result reported in Li et al. [5], and confirms again the effectiveness of the carefully-designed caveat-indicating sentence patterns in [5].

Our open information extraction methods can accurately extract controversy-related knowledge from textual Q&A discussions and API documentation.

V. USEFULNESS OF CONTROVERSY SUMMARY

Our empirical study in Section II reveals that the scale of controversial discussions in Stack Overflow is significant. We also observe that these controversies may not be easily discoverable by community-curated quality factors and critiques when developers seek solutions to their problems on Stack Overflow. Having evaluated the accuracy of our approach for extracting controversy-related knowledge, we now would like to investigate the usefulness of our generated controversy summary, compared with the existing community-curated quality indicators and critiques, for helping developers notice the controversies in Stack Overflow discussions and select the most appropriate solutions to Stack Overflow questions in face of such controversial discussions.

A. User Study Design

1) Experimental Tasks: As our study focuses on the interference of controversial discussions on the developers' information seeking on Stack Overflow, we select experimental tasks from our dataset of controversial discussion threads. For the purpose of evaluating the usefulness of the generated controversy summary, we follow the following selection criteria. First, the view counts of the questions should be 5000 or above, which indicates that developers frequently encounter similar problems. Second, the questions should have different numbers of answers. Third, the controversial answers to the questions should have positive votes, which indicates that they have been adopted by some developers. Fourth, the controversial answers should cover different types of controversies (wrong answer, less optimal answer and out-of-date answer). Fifth, the controversial answers should be criticized by different means (comments, answers or both), and the critique posts may or may not suggest the remedies. Finally, the controversy can be concerned about the whole answer or only some parts of the answer. According to these selection criteria, we select eight Stack Overflow questions as our experimental tasks (see Table VIII). We select more Android-related questions than Java-related questions, because Android APIs are relatively more challenging to use than Java APIs for ordinary developers.

2) Experimental Procedure: In our study, we ask the study participants to read the Q&A discussions on one selected question at a time on the Stack Overflow website, and determine the most appropriate solution they would like to adopt to solve the question. The participants can access online materials (e.g., API documentation) referenced in the Q&A discussions to understand and judge the validity of certain solutions. Each question is allocated up to eight minutes, and the whole study completes in about an hour. We use screen-casting software to record the participants' task completion process. In post-experiment analysis, we determine to use screencasts to evaluate our results. We also interview each participant to collect the rationale behind his/her choices of certain solutions.

3) Participants and Experimental Groups: We recruit 18 developers from an IT company that have over 2000 developers. These 18 developers have 1 to 5 years (on average 3.2 years) of Java and Android development experience on either commercial or open-source projects. Based on the development experience of these developers, we divide them into three "comparable" groups: G_1, G_2 and G_3. Each group has six developers. All groups read the Q&A discussions on Stack Overflow. G_1 is the control group. Its participants do not know the study is about the controversies in Stack

TABLE VIII
EIGHT EXPERIMENTAL TASKS IN OUR USER STUDY

Question	Selection Criteria					
	View Times	Answers	Votes for Controversial Answer	Controversy Type	Critique Posts (Comment or answer)	Complete or Partial Controversy
Q1: How to log JSON responses in Dropwizard (Jersey)	6,800	5	10	Out-of-date	Both	Complete
Q2: How to enable / disable a Preference?	6,062	2	12	Less optimal	Answer	Complete
Q3: Android center view in FrameLayout doesn't work	83,354	5	34	Less optimal	Both	Complete
Q4: Determine when a ViewPager changes pages	96,726	4	108	Out-of-date	Both	Complete
Q5: Set EditText cursor color	244,947	17	1007	Wrong	Comment	Complete
Q6: Running cordova build android - unable to find attribute android:fontVariationSettings and android:ttcIndex	40,922	24	100	Less optimal	Both	Partial
Q7: Difference between Date class in Package java.util & Package java.sql	10,297	5	16	Out-of-date	Answer	Partial
Q8: Software keyboard resizes background image on Android	56,789	13	179	Less optimal	Both	Complete

TABLE IX
RESULTS OF USER STUDY

Metrics	Group	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Average
Correctness	G_1	2	1	2	2	1	1	3	1	27.1%
	G_2	3	4	4	4	3	2	5	1	54.2%
	G_3	6	5	5	6	4	4	6	2	79.2%
	$Average_1$	61.1%	55.6%	61.1%	66.7%	**44.4%**	**38.9%**	77.8%	**22.2%**	-
Completion Time	G_1	212.3 (25.4)	195.2 (26.6)	260.8 (70.5)	233.0 (33.5)	463.3 (58.9)	364.8 (108.1)	312.5 (61.5)	488.0 (71.7)	311.3 (57.0)
	G_2	275.2 (67.3)	245.3 (34.0)	381.5 (51.1)	315.3 (47.8)	556.0 (38.5)	573.8 (33.5)	477.7 (77.3)	564.7 (36.7)	432.7 (48.3)
	G_3	220.7 (26.1)	203.8 (25.6)	304.3 (67.5)	215.7 (26.5)	469.5 (40.6)	444.7 (53.7)	407.5 (62.9)	460.3 (28.5)	340.8 (41.4)
	$Average_2$	236.1 (39.6)	214.8 (28.7)	315.6 (63.0)	254.7 (35.9)	**496.3** (46.0)	**461.1** (65.1)	399.2 (67.2)	**491.0** (45.7)	-

Overflow, and they use the standard Stack Overflow user interface without any annotation of controversial answers and critique posts. G_2 and G_3 are the experiment groups with different levels of controversy warnings. For G_2, the detected controversial answers are annotated with a general warning "This answer may be controversial", but no details about critique posts and explanatory API links/caveats are provided. For G_3, the detected controversial answers are annotated with the detailed controversy summary generated by our approach (see Section III-D). Setting up G_2 and G_3 allows us to confirm the advantages of detailed controversy summary over general controversy warning.

4) Evaluation Metrics: We use the two metrics to evaluate the participants' performance: task completion time and answer correctness. Task completion time evaluates how fast a participant can find the solution that they believe to be the most appropriate one to an experimental question. From the screencasts of the participants' task completion process, we determine the time.

Answer correctness evaluates whether the solution submitted by a participant is actually the most appropriate solution to the question. The two authors collaboratively determine the ground-truth solution to each experimental question by carefully reading the whole Q&A discussion on the question and any referenced online materials. The ground-truth answer to a question can be in three forms: a single correct answer, the combination of several answers when some parts of an answer is controversial, or no appropriate answer when the controversy has been pointed out but no remedy has been suggested. If the submitted solution to a question matches the ground-truth solution, the participant get 1 mark, otherwise 0 mark. We use Wilcoxon signed-rank test [27] with Bonferroni correction [28] to determine if the performance difference across different groups is statistically significant, We consider one group performs better than the other at the confidence level of 95%, if the corresponding Wilcoxon signed-rank test result (i.e., $p - value$) is less than 0.05. We also use the Cliff's delta (δ) [29] to quantify the amount of difference between the two groups. The amount of difference is considered negligible ($|\delta| < 0.147$),small ($0.147 \leqslant |\delta| < 0.33$), moderate ($0.33 \leqslant |\delta| < 0.474$), or large ($|\delta| \geqslant 0.474$), respectively.

B. Results - Answer Correctness

In Table IX, the first row shows the answer correctness results. It lists the number of correct answers by each group on each question. The last column shows the answer correctness rate for each group (out of 48 answers submitted by six participants on eight questions). The group G_1 that complete the tasks without any controversy warnings achieve the average answer correctness rate 27.1%. Except for Q7, the other seven questions have only one or two correct answers (that is, only one or two participants answer them correctly). Compared with G_1, the group G_2 that complete the tasks with general controversy warnings achieve relatively better average answer correctness rate 54.2%. Except for Q6 and Q8, three or more participants answer the other six questions correctly. The group G_3 that complete the task with detailed controversy summaries achieves the highest average answer correctness rate 79.2%.

All six participants answer Q1/Q4/Q7 correctly, and four or five participants answer Q2/Q3/Q5/Q6 correctly.

The Wilcoxon signed-rank test shows that the differences of the answer correctness rates between G_1 and G_2, G_1 and G_3, and G_2 and G_3 are statistically significant at $p-value < 0.05$. The Cliff's delta of G_1 versus G_2 and G_1 versus G_3 are more than 0.600, which indicate that some forms of explicitly controversy warnings can significantly improves answer correctness by a large margin, compared with relying on only community-curated quality indicators and critiques. Furthermore, the Cliff's delta of G_2 and G_3 is also more than 0.600, which indicates that our detailed controversy summary can significantly improves answer correctness by a large margin, compared with just general controversy warning.

The $Average_1$ row shows the answer correctness rate for each question (out of the 18 answers). Among the eight questions, Q5/Q6/Q8 have relatively lower answer correctness rates. These three questions have large numbers of answers, which make it difficult to determine the most appropriate solutions to the questions. Especially for Q8, it involves more than 3 controversial discussion threads, which make it even more challenging to determine the appropriate solution in between alternative answers. However, with the help of either general controversy warnings or detailed controversy summaries, more participants in G_2 and G_3 answer these three questions correctly, compared with G_1. For the other five questions that have small numbers of answers, the generated controversy warnings are even more effective in help the participants G_2 and G_3 avoid the controversial answers and choose the correct solutions. In contrast, the participants in G_1 mainly base their choices of appropriate answers on the accepted answers and the answers with the highest votes, but fail to notice the critique comments or other critique answers pointing out the controversies in the selected answers. This leads to the G_1's very low answer correctness rate.

C. Results - Task Completion Time

The second row in Table IX shows the average task completion time of each group on each question and on all eight questions (last column). The numbers in brackets are standard deviation. The G_1 participants take the shortest time to complete the tasks, the G_2 participants take the longest time, and the task completion time of the G_3 participants is in between. The G_1 participants mostly select the accepted answer or the highest-vote answers as their solutions, without much checking of other answers. So they complete the tasks faster. In contrast, some forms of controversy warnings make the G_2 and G_3 participants more cautious in selecting solutions. They will spend more time on checking the controversy and other answers. Our explicit, concise controversy summaries can facilitate such checking. As such, the G_3 participants can complete the tasks faster than those in G_2.

The Wilcoxon signed-rank test shows that the differences of the task completion time between G_1 and G_2, G_1 and G_3, and G_2 and G_3 are statistically significant at $p - value < 0.05$. The Cliff's delta of G_1 versus G_2 and G_1 versus G_3 are more than 0.500, which indicate that some forms of controversy warnings can "force" the developers spend significantly more time to investigate the controversies and select the appropriate solutions, compared with relying on only community-curated quality indicators and critiques. The Cliff's delta of G_2 and G_3 is 0.350, which indicates that our detailed controversy summary can save some investigation time by a moderate margin, compared with just general controversy warning.

Comparing the average task completion time on each question (the last row $Average_2$ in Table IX), we can see that the questions (Q5/Q6/Q8) with large numbers of answers take longer time to complete than those with small numbers of answers. However, without the help of some forms of controversy warnings, spending more time on a question does not necessarily lead to higher answer correctness rate. The G_1 participants comment that they often go back to the accepted answers or the highest-vote answers after scanning many other answers to Q5/Q6/Q8, because they are often overwhelmed by the information in all these answers.

Direct controversy warnings on the controversial answers can more effectively raise developers' attention to the potential issues in the answers, compared with latent critique posts. Both general controversy warnings and detailed controversy summaries can reduce the chance of selecting inappropriate solutions. Furthermore, detailed controversy summaries can further facilitate the investigation of the controversies than general controversy warnings.

VI. Threats to Validity

Threats to internal validity The threats to internal validity relate to errors in our experimental data, tool implementation and personal bias in user studies. To avoid errors in experimental data, we carefully tested our data processing tool and verify the accuracy of our tool implementation by manually examining a large number of data instances outputted by each step of our tool. To reduce the personal bias in the manual examination of the extracted controversy information, the two authors annotate the data instances independently and the Cohen's Kappas indicate the substantial or almost perfect agreement between the two annotators.

Threats to external validity The threats to external validity relate to the generalizability of our experiment results and findings. In this work, we study the controversies in Stack Overflow and focus on java/android-related questions and answers, because Stack Overflow is the most popular Q&A site for computer programming and Java and Android are among the most discussed programming techniques on Stack Overflow. However, further studies are needed to validate and generalize our findings to other domain-specific Q&A sites and the Q&A discussions on other programming techniques. Furthermore, our user study is small scale. More user evaluation is needed to confirm and improve the usefulness of our generated controversy summaries. We will release our webbrowser-plugin for annotating Stack Overflow posts with controversy summaries for public evaluation.

VII. Related work

Software developers learn to use APIs from both *official API documentation* and *crowd documentation* such as Stack Overflow questions and answers. Official API documentation is often criticized as being incomplete or stale, while crowd documentation enter as a rescue to keep official documentation up-to-date [30]–[32] or enrich official documentation with crowd insights [33], [34]. Different from these studies, our work focuses on making the latent controversies in Q&A sites explicit and facilitating the understanding of the discovered controversies with relevant API links/caveats.

A recent work by Li et al. [5] points out that high-quality, well-maintained official documentation do exist, and they mine a large number of API usage caveats that developers should be aware of from official Android API documentation. They also find that many API caveats are either directly quoted or paraphrased to help answer questions in Stack Overflow. Our empirical study shows that official API caveats are also often used to explain the controversies. We adopt the API-caveat mining method [5] for building a knowledge base of official API caveats, from which we identify explanatory API caveats in the controversial discussions.

Many information extraction techniques have been developed to extract software engineering knowledge embedded in software text, such as API extraction [35], [36], mining API usage examples [30], [37]. To battle the information overload problem in Q&A sites, text summarization techniques have been developed to generate query-specific post summary [38] or technology comparison report [39], [40]. Different from these works, our work distill controversy-related knowledge from community Q&A discussions and exploit the discovered controversies as a new means of judging the answer quality.

Word embedding techniques have recently been adopted to improve document retrieval [15], [37], to find domain-specific synonyms [41], to recommend analogical libraries and APIs [42], [43]. Traceability recovery in software text has been widely studied [30], [35], [44]–[47]. In this work, we develop a weighted word-embedding based sentence matching approach to recover the traceability links between the controversial answers and the critique posts and between the API caveats discussed in the Stack Overflow posts and the API caveats in official documentation.

VIII. Conclusion and Future Work

In this paper, we systematically investigate the controversial discussions in Stack Overflow. Our study shows that there exists a large scale of controversies in Stack Overflow, involving many wrong answers, less optimal answers and out-of-date answers. Unfortunately, neither community-curated answer-quality indicators (e.g., accepted answer, answer vote) can reliably identify such controversial answers, nor are existing critiques readily discoverable and accessible to developers who seek solutions to their programming problems in community Q&A sites . Consequently, many controversial solutions have been adopted by developers as evident in the high votes on many controversial solutions. To reduce the negative impact of such controversial answers on developers, we design an open information extraction method that turns latent controversial discussions into salient, concise and semi-structured controversy warnings. Our evaluation validates the practicality of our approach for extracting controversy-related knowledge from a large corpus of textual Q&A discussions and API documentation, as well as the usefulness of the controversy summaries generated by our approach for helping developers make more informed choices of appropriate solutions to programming questions in community Q&A sites.

IX. Acknowledgements

We would like to thank Li et al. for sharing their tools and dataset. We also appreciate the reviewers for their insightful comments to help us improve this paper. Xin Xia is the corresponding author. This research was partially supported by the National Key Research and Development Program of China (2018YFB1003904) and NSFC Program (No. 61972339).

References

[1] C. Shah and J. Pomerantz, "Evaluating and predicting answer quality in community qa," in *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2010, pp. 411–418.

[2] H. Zhang, S. Wang, T.-H. P. Chen, Y. Zou, and A. E. Hassan, "An empirical study of obsolete answers on stack overflow," *IEEE Transactions on Software Engineering*, 2019.

[3] P. Arora, D. Ganguly, and G. J. Jones, "The good, the bad and their kins: Identifying questions with negative scores in stackoverflow," in *2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. IEEE, 2015, pp. 1232–1239.

[4] Y. Yao, H. Tong, T. Xie, L. Akoglu, F. Xu, and J. Lu, "Detecting high-quality posts in community question answering sites," *Information Sciences*, vol. 302, pp. 70–82, 2015.

[5] 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.

[6] R. Singh and N. S. Mangat, *Elements of survey sampling*. Springer Science & Business Media, 2013, vol. 15.

[7] S. P. Ponzetto and M. Strube, "Deriving a large scale taxonomy from wikipedia," pp. 1440–1445, 2007.

[8] ——, "Knowledge derived from wikipedia for computing semantic relatedness," *Journal of Artificial Intelligence Research*, vol. 30, no. 1, pp. 181–212, 2007.

[9] D. Ye, Z. Xing, C. Y. Foo, Z. Q. Ang, J. Li, and N. Kapre, "Software-specific named entity recognition in software engineering social content," vol. 1, pp. 90–101, 2016.

[10] C. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. Bethard, and D. McClosky, "The stanford corenlp natural language processing toolkit," in *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, 2014, pp. 55–60.

[11] D. Ye, Z. Xing, C. Y. Foo, J. Li, and N. Kapre, "Learning to extract api mentions from informal natural language discussions," in *2016 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2016, pp. 389–399.

[12] R. Mihalcea and P. Tarau, "Textrank: Bringing order into text," in *Proceedings of the 2004 conference on empirical methods in natural language processing*, 2004.

[13] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web." Stanford InfoLab, Tech. Rep., 1999.

[14] J. E. Hopcroft and R. M. Karp, "An n^5/2 algorithm for maximum matchings in bipartite graphs," *SIAM Journal on computing*, vol. 2, no. 4, pp. 225–231, 1973.

[15] X. Ye, H. Shen, X. Ma, R. C. Bunescu, and C. Liu, "From word embeddings to document similarities for improved information retrieval in software engineering," pp. 404–415, 2016.

[16] J. L. Neto, A. D. Santos, C. A. Kaestner, N. Alexandre, D. Santos *et al.*, "Document clustering and text summarization," 2000.

[17] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *neural information processing systems*, pp. 3111–3119, 2013.

[18] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv: Computation and Language*, 2013.

[19] J. P. Turian, L. Ratinov, and Y. Bengio, "Word representations: A simple and general method for semi-supervised learning," pp. 384–394, 2010.

[20] Z. S. Harris, "Distributional structure," *Word*, vol. 10, no. 2-3, pp. 146–162, 1954.

[21] Y. Goldberg and O. Levy, "word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method," *arXiv preprint arXiv:1402.3722*, 2014.

[22] D. Soutner and L. Muller, "Continuous distributed representations of words as input of lstm network language model," pp. 150–157, 2014.

[23] G. Chen, C. Chen, Z. Xing, and B. Xu, "Learning a dual-language vector space for domain-specific cross-lingual question retrieval," *automated software engineering*, pp. 744–755, 2016.

[24] T. Hong, C. Lin, K. Yang, and S. Wang, "Using tf-idf to hide sensitive itemsets," *Applied Intelligence*, vol. 38, no. 4, pp. 502–510, 2013.

[25] J. Martineau and T. Finin, "Delta tfidf: An improved feature space for sentiment analysis," 2009.

[26] J. R. Landis and G. G. Koch, "An application of hierarchical kappa-type statistics in the assessment of majority agreement among multiple observers," *Biometrics*, pp. 363–374, 1977.

[27] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics bulletin*, vol. 1, no. 6, pp. 80–83, 1945.

[28] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[29] N. Cliff, *Ordinal methods for behavioral data analysis*. Psychology Press, 2014.

[30] S. Subramanian, L. Inozemtseva, and R. Holmes, "Live api documentation," in *Proceedings of the 36th International Conference on Software Engineering*. ACM, 2014, pp. 643–652.

[31] P. W. McBurney and C. McMillan, "Automatic documentation generation via source code summarization of method context," in *Proceedings of the 22nd International Conference on Program Comprehension*. ACM, 2014, pp. 279–290.

[32] H. Jiang, L. Nie, Z. Sun, Z. Ren, W. Kong, T. Zhang, and X. Luo, "Rosf: Leveraging information retrieval and supervised learning for recommending code snippets," *IEEE Transactions on Services Computing*, 2016.

[33] P. Moslehi, B. Adams, and J. Rilling, "On mining crowd-based speech documentation," in *Proceedings of the 13th International Conference on Mining Software Repositories*. ACM, 2016, pp. 259–268.

[34] L. Ponzanelli, A. Bacchelli, and M. Lanza, "Leveraging crowd knowledge for software comprehension and development," in *Software Maintenance and Reengineering (CSMR), 2013 17th European Conference on*. IEEE, 2013, pp. 57–66.

[35] B. Dagenais and M. P. Robillard, "Recovering traceability links between an api and its learning resources," in *Proceedings of the 34th International Conference on Software Engineering*. IEEE Press, 2012, pp. 47–57.

[36] Y. Lu, G. Li, R. Miao, and Z. Jin, "Learning embeddings of api tokens to facilitate deep learning based program processing," in *International Conference on Knowledge Science, Engineering and Management*. Springer, 2016, pp. 527–539.

[37] Q. Huang, X. Xia, Z. Xing, D. Lo, and X. Wang, "Api method recommendation without worrying about the task-api knowledge gap," in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*. ACM, 2018, pp. 293–304.

[38] B. Xu, Z. Xing, X. Xia, and D. Lo, "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.

[39] Y. Huang, C. Chen, Z. Xing, T. Lin, and Y. Liu, "Tell them apart: distilling technology differences from crowd-scale comparison discussions," in *Proceedings of the 33rd ACM/IEEE international conference on automated software engineering. ACM*, 2018, pp. 214–224.

[40] D. Kavaler and V. Filkov, "Determinants of quality, latency, and amount of stack overflow answers about recent android apis," *PloS one*, vol. 13, no. 3, p. e0194139, 2018.

[41] C. Chen, Z. Xing, and X. Wang, "Unsupervised software-specific morphological forms inference from informal discussions," in *Proceedings of the 39th International Conference on Software Engineering*. IEEE Press, 2017, pp. 450–461.

[42] C. Chen and Z. Xing, "Similartech: Automatically recommend analogical libraries across different programming languages," in *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*. ACM, 2016, pp. 834–839.

[43] C. Chen, Z. Xing, Y. Liu, and K. L. X. Ong, "Mining likely analogical apis across third-party libraries via large-scale unsupervised api semantics embedding," *IEEE Transactions on Software Engineering*, 2019.

[44] S. P. Reiss, "Semantics-based code search," in *Proceedings of the 31st International Conference on Software Engineering*. IEEE Computer Society, 2009, pp. 243–253.

[45] V. Raychev, M. Vechev, and E. Yahav, "Code completion with statistical language models," in *Acm Sigplan Notices*, vol. 49, no. 6. ACM, 2014, pp. 419–428.

[46] B. Dit, M. Revelle, M. Gethers, and D. Poshyvanyk, "Feature location in source code: a taxonomy and survey," *Journal of software: Evolution and Process*, vol. 25, no. 1, pp. 53–95, 2013.

[47] A. Panichella, B. Dit, R. Oliveto, M. Di Penta, D. Poshyvanyk, and A. De Lucia, "How to effectively use topic models for software engineering tasks? an approach based on genetic algorithms," in *Proceedings of the 2013 International Conference on Software Engineering*. IEEE Press, 2013, pp. 522–531.