What are Pros and Cons? Stance Detection and Summarization on Feature Request

Yawen Wang[1,2,3], Junjie Wang[1,2,3,*], Hongyu Zhang[4], Kairui Wang[1,2,3], Qing Wang[1,2,3]
[1] *State Key Laboratory of Intelligent Game, Beijing, China*
[2] *Science and Technology on Integrated Information System Laboratory, Institute of Software Chinese Academy of Sciences, Beijing, China*
[3] *University of Chinese Academy of Sciences, Beijing, China*
[4] *Chongqing University, Chongqing, China*
{yawen2018, junjie, wangkairui2021, wq}@iscas.ac.cn, hyzhang@cqu.edu.cn

***Abstract*—BACKGROUND: In an online issue tracking system, e.g., GitHub Issue Tracker, feature requests and the associated comment stream provide valuable crowd-generated knowledge for requirements elicitation. To decide whether a feature request should be accepted or not, stakeholders need to identify the comments for/against the feature and understand the two-sided opinions, which is time- and effort-consuming considering the abundant information embedded in lengthy comment stream per feature request. AIMS: This paper proposes VOTEBOT for automatically detecting stance (for/against) and summarizing the related opinions on a feature request, which can facilitate the decision making (i.e., voting) of feature requests. To our best knowledge, such an approach is previously unexplored for crowd-based requirements elicitation. METHOD: VOTEBOT is a relation-aware approach, which incorporates three types of relations among the comments or among the comment sentences to better understand the discussions about feature requests. Specifically, it extracts the reply-to relation among the comments, and incorporates it into a BERT-based classifier for stance detection. It also designs a graph-based ranking algorithm, and incorporates semantic relevance and argumentative relations for stance summarization. RESULTS: The automatic evaluation on 250 feature requests with 6,598 comments from five GitHub projects, and the evaluation with practitioners on five new projects, show the promising results. CONCLUSIONS: VOTEBOT is effective in stance detection and stance summarization, and potentially useful for understanding feature requests and associated discussions in real-world practice.**

***Index Terms*—Feature Request, Stance Detection, Stance Summarization**

I. INTRODUCTION

The degree of success and failure of a software system depends upon the level and quality of services it provides, as required by its users and stakeholders. Requirements elicitation is the practice of identifying and collecting the requirements from the stakeholders of the system to be developed, which plays a significant role in the overall quality of requirements engineering process. Crowd-generated content is an essential source of knowledge that can be utilized to create a wider perspective. Utilizing this information can bring big benefits and enhancement to the requirements elicitation activity.

The feature requests in Open Source Software (OSS) issue tracking systems, e.g., GitHub Issue Tracker, Google Code Issue Tracker, and Bugzilla, are a commonly-utilized knowledge source for crowd-based requirements elicitation. Once a feature request is submitted, OSS stakeholders would be involved in asynchronous discussions by leaving comments to express their opinions about whether the target feature should be accepted or not. However, the discussions about feature requests could become very lengthy and difficult to follow, especially for complex features which might have a major impact on the project. Additionally, OSS stakeholders with mixed backgrounds often need to undergo extensive discussions before reaching a common ground, which aggravates the problem of information overload in feature discussion. A previous study [1] on 82 GitHub projects revealed that on average, there are more than 170 issue reports (including feature requests) per project, and each issue report contains over 20 comments embedding abundant information of diverse types, and 10 more participants involved in the discussions.

A typical feature request is as demonstrated in Figure 1 (Angular #40904[1]), where different stakeholders might express their stance (for/against) on the feature request with detailed comments. The feature request on GitHub is usually assigned with a tag like *vote required*, which asks the stakeholders' opinions about whether or not the feature should be accepted. For an effective voting, the stakeholders usually need to glance over all previous discussions, identify the comments for/against the feature (i.e., stance detection), and summarize the opinions on each side (i.e., stance summarization). Since the two-sided opinions are buried in lengthy discussion threads and mixed with other information, such process is quite time- and effort-consuming. This study aims to develop an automated approach for detecting the comments' stance and summarizing the related opinions that are for/against a certain feature request, which is previously unexplored to our best knowledge. We name the approach VOTEBOT, mimicking a bot that can automatically conduct the rebuttal speech in a debate for a feature request and facilitate the voting process.

* Corresponding author.

[1]https://github.com/angular/angular/issues/40904/

978-1-6654-5223-6/23/$31.00 © 2023 IEEE

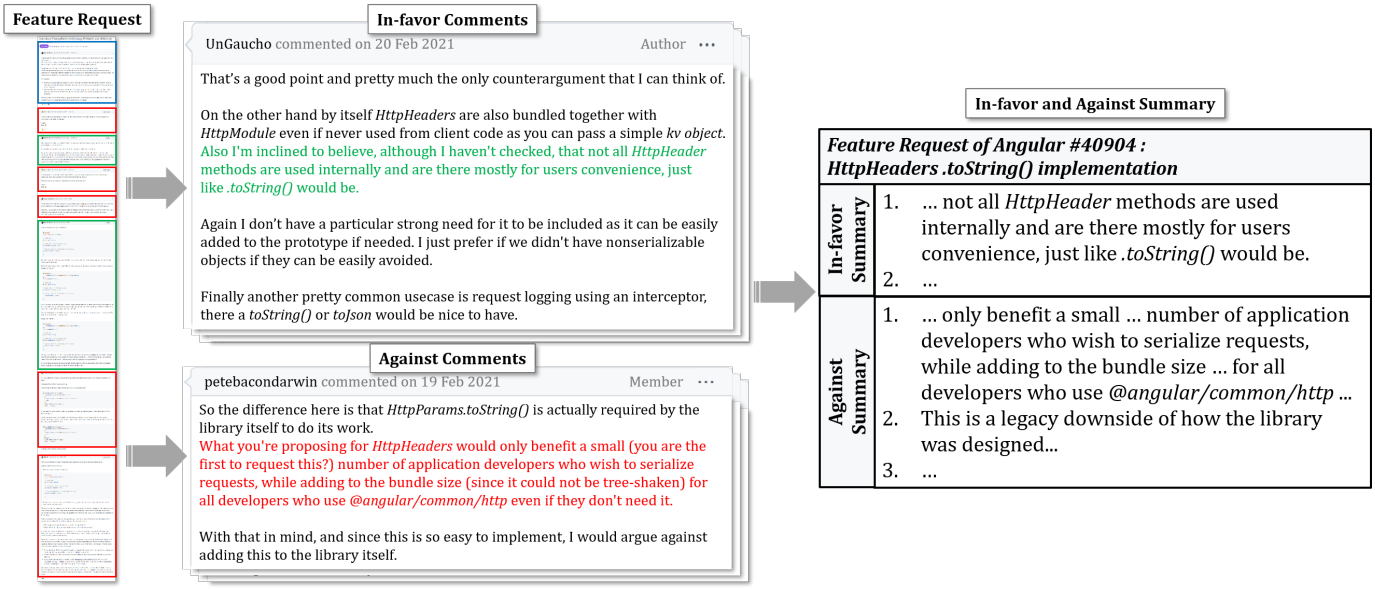

Fig. 1. An example of stance detection and summarization for the feature request *HttpHeaders .toString() implementation* in *Angular*.

There are two main challenges for designing VoteBot:

(1) The stance polarity is buried in the comment stream, while the term-based classification in previous studies is unworkable. Previous studies on issue/review classification mainly take effect with the indicative terms (e.g., agree, good) or publicly-available attributes (e.g., review type) [1], [2]. Yet in our scenario, comments with positive/negative indicative terms might be in completely opposite stance. For example, as shown in Figure 1, the comment in favor of the feature request (in green color) has such indicative words like 'not' which express the negative sentiment and can easily be classified as the against stance. Besides, previous studies reveal that measures of stance and sentiment are only 60% correlated [3], which further indicates the uniqueness of stance detection. To tackle this issue, we incorporate the reply-to relations in a discussion thread, which capture the structural logic associations among the comments, to improve the stance detection.

(2) The two-sided opinion summary is tightly related to the context of target features and argumentation structures, hence existing approaches solely based on the content itself would fail to work. Existing text summarization approaches typically represent the documents as a graph based on the semantic similarity among its contained sentences, and those with higher centrality are selected as the summary [4], [5]. Yet in feature request discussions, the sentences which do not convey opinions for/against the target features should not be included into the summary. Besides, when the stakeholders express an opinion, some sentences are utilized for claiming the major standpoints, while others are premises of the points, which takes different weights in contributing to the opinion elaboration. This motivates us to incorporate the semantic relation between comment sentences and the proposed feature, and the argumentative relations among comment sentences, to facilitate the stance summarization.

In this paper, we propose VoteBot which automatically conducts stance detection and stance summarization on discussions related to feature requests, to facilitate the decision making of feature requests. It takes a natural-language feature request involving feature description and related comment stream as input, detects the stance (i.e., pro/con) and extracts the summary sentence for each comment, and concatenates the comment summary sentences of two polarities to generate a pro summary and a con summary, respectively. VoteBot is a relation-aware approach, which incorporates three types of relations among the comments or among the comment sentences to better understand the discussions about feature requests. For stance detection, it extracts the explicit and implicit reply-to relations among the comments, and incorporates them into the BERT-based classifier. For stance summarization, it designs an extractive summarization method based on graph-based ranking, and incorporates the semantic relevance with the feature description and the argumentative relations among the comment sentences.

We evaluate VoteBot on 250 features requests involving 6,598 comments from five projects hosted on GitHub. For stance detection, the overall precision and recall achieved by VoteBot is 79.1% and 78.0%, respectively, outperforming three baselines and two variants. VoteBot also achieves 81.20% accuracy for stance summarization, outperforming four commonly-used baselines and three variants. As for the evaluation on two stance summaries, VoteBot achieves 60.1 of ROUGE-1 for pro summaries and 65.3 of ROUGE-1 for con summaries, outperforming two state-of-the-art (SOTA) baselines. We further conduct an evaluation with practitioners with VoteBot on five new projects to examine its usefulness in real-world practice.

The main contributions of this paper are as follows:

- A novel approach for stance detection and stance summarization to understand discussions about feature requests, which is unexplored previously to our best knowledge and can motivate the follow-up studies.
- A tool named VoteBot including a module incor-

porating the reply-to relations to improve BERT-based classifier on stance detection, and a module incorporating the semantic relevance and argumentative relations to improve graph-based ranking on stance summarization.
- An automatic evaluation on 250 feature requests involving 6,598 comments from five projects on GitHub with affirmative performance, and a human evaluation to explore VOTEBOT's usefulness in real-world practice.
- Publicly accessible source code, data and results[2].

II. BACKGROUND

A. Stance Detection

The task of stance detection aims to automatically identify the author's standpoint/attitude (e.g. pro, con, or neutral, etc.) expressed in text towards a specific proposition, topic, or target [6]. This task differs from the task of sentiment analysis, which aims to categorize texts according to a notion of polarity (e.g., positive, negative or neutral). The sentiment polarity is usually explicitly expressed in a text, while the stance polarity that the text holds with respect to a topic is generally more abstract and might not be directly mentioned in the text [7]. Sometimes texts can transmit a negative sentiment, but be in favor of the target topic. Previous studies [3] show that measures of stance and sentiment are only 60% correlated.

Stance detection originates from mining political debates [8], and has attracted increasing attention on detecting user stance from social media in recent years, which is mainly boosted by the tweet stance detection competition and datasets released in SemEval-2016 [9]. One main challenge is that the targets for stance detection may not appear in the text unlike sentiment analysis in which the aspects or targets are explicitly presented, resulting in the lack of contextual information for understanding the targets [10]. To capture the contextual information of the targets, most existing works leverage deep learning classifier (e.g., TextCNN [11]) with attention mechanism to connect targets with their respective contexts, or introducing external knowledge to address this issue (e.g., aligning words with Wiki concepts [12]). In this work, we conduct stance detection on feature requests to identify the stance of comments towards a proposal related to the software, and tackle it by introducing contextual information based on characteristics of OSS issue reports.

B. Stance Summarization

In this work, we define the task of stance summarization, which aims to aggregate and summarize the key opinions based on stance from discussions related to a feature request. Thus readers can immediately view most relevant opinions for each stance (e.g., pro/con). The difference to conventional summarization task is that stance summarization takes texts which are discussions about feature requests involving a feature description and a steam of comments, summarizes comment by comment to output summaries by each stance.

In this context, we summarize each comment following the paradigm of extractive summarization to determine a sentence's salience indicating whether including the sentence in the summary. It is based on graph-based ranking algorithms, which conventionally represent a document D consisting of n sentences $\{s_1, s_2, ..., s_n\}$ as a graph $G = (V; E)$, where V is the set of vertices denoting sentences, and E is the set of edges denoting relevance between sentences [4], [5]. The weight e_{ij} between node pairs $<v_i, v_j>$is typically a measure of similarity between two sentences $<s_i, s_j>$(e.g. cosine distance between their representations). The centrality for sentence s_i can be defined as: $centrality(s_i) = \sum_{j \neq i}^{n} e_{ij}$ These algorithms select the most salient sentences (i.e., with the highest centrality) as summaries based on the assumption that the more similar a sentence is to other sentences in the texts, the more important it is [13].

III. APPROACH

In this paper, we propose VOTEBOT which automatically conducts stance detection and stance summarization on discussions related to feature requests. Figure 2 presents its overview. VOTEBOT first preprocesses the natural-language feature request (including feature description and related comment stream). Then VOTEBOT deals with each comment to assign stance polarities, and extract summary sentences. For stance detection, it extracts the explicit and implicit reply-to relations among the comments, and incorporates it into the BERT-based classifier. For stance summarization, it designs an extractive summarization method based on graph-based ranking, and incorporates semantic relevance between comment sentences and feature description, and the argumentative relations among comment sentences. Finally, VOTEBOT concatenates the comment summary sentences of two polarities to produce a pro summary and a con summary, respectively, which serves as the rebuttal speech in the debate for a feature request and facilitate the voting process.

A. Data Preprocessing

The raw feature requests on GitHub are written in natural language, submitted by stakeholders with different backgrounds, containing two parts of feature description and related comment stream. This leads to the frequent occurrences of uninformative words, such as repetitive quotations, html tags, etc., in these texts. We therefore conduct the preprocessing for the feature description and related comments.

First, we *filter* the uninformative comments such as robot comments, which would not contribute to the opinion expression. We also remove the html tags and unrecognizable codes in the texts. Second, we *format* the words to alleviate the influence of word morphology by performing lowercase and lemmatization on all words in the text with Spacy[3]. Third, we *simplify* the informative but non-natural-language texts (excluding emoticons) by replacing them with special tokens, to unify the model inputs and also avoid the internal

[2]https://github.com/KeyL99/VoteBot/

[3]https://spacy.io/

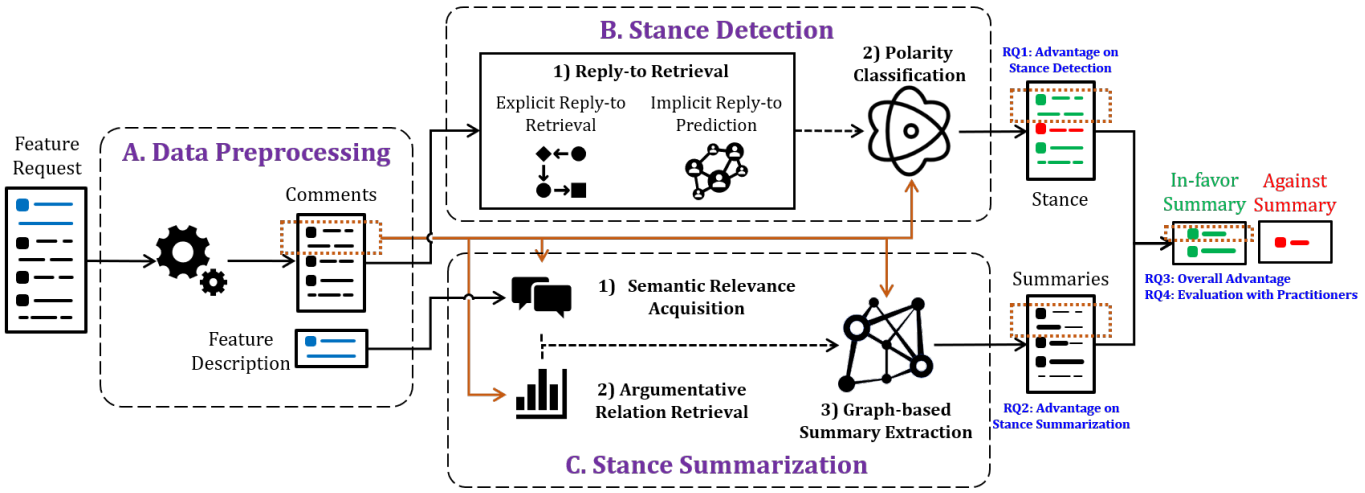

Fig. 2. The overview of VOTEBOT.

words confusing the model, e.g., replacing quoting texts with *<ref>*, code snippets with *<code>*, hyperlinks with *<url>*, and pictures/GIFs with *<pic>*.

B. Stance Detection

As shown in Figure 2, VOTEBOT conducts the stance detection with two steps, i.e., reply-to retrieval and polarity classification. In detail, it first retrieves the reply-to relations among the comments, and incorporates them with the comments' textual information to fine-tune the *BERT* (Bidirectional Encoder Representations from Transformers) [14] model for classifying the polarity of each comment.

1) ***Reply-to Retrieval****:* During the discussions about feature requests, it usually involves the reply-to relations among the comments, which might implies the structural logic associations during the discussion. There are explicit reply-to relations, e.g., directly quoting someone's words or referring to someone. Meanwhile, the OSS stakeholders can also implicitly reply to other's comments with no explicit indicative words. To handle this situation, we employ the pattern matching to retrieve the explicit reply-to relation, and utilize the conversation disentanglement model for implicit reply-to prediction.

Pattern matching for explicit reply-to retrieval. We retrieve the explicit reply-to relations with regular expressions by matching specific patterns. For example, we identify the token "@" or indicative token of quoting words, and then match the username or quotation to retrieve the explicit relations. Although the explicit reply-to relations can be 100 percent retrieved by pattern matching, the comments with explicit reply-to relations only occupies about 28% in our experimental data (the data shown in Table I), which motivates us to further retrieve the implicit ones.

Conversation disentanglement model for implicit reply-to prediction. The conversation disentanglement task is for identifying separate conversations in a stream of messages, where a chain of messages having reply-to relations are treated as a separate conversation [15]. We borrow this idea to predict implicit reply-to relations within a stream of comments with the model of *irc-disentanglement* [16]. It scores the potential antecedents in a reply-to relation with a feedforward model based on textual features (e.g., word overlap and context), plus sentence embeddings calculated by averaging the vector from GloVe [17] of each word. We use the model released by the original paper, which has been trained on the 77,563 messages from *#Ubuntu* Internet Relay Chat channel and *#Linux* channel. In our context, we treat all the comments of a feature request as input. For each comment, we use the trained *irc-disentanglement* model to predict which one the current comment is most likely to reply to.

After implicit reply-to prediction, 73% comments' reply-to relations (either or both explicit and implicit ones) in our experimental data can be retrieved. Note that, the reply-to relations (even not 100% retrieved and accurate) provide VOTEBOT more clues for the stance detection, and the experimental evaluation in Section V-A also demonstrates its effectiveness. Another issue is that the *irc-disentanglement* model might make wrong predictions, which leads to conflicts in comments with both explicit and implicit reply-to relations. The conflicting ones occupies about 5% of the comments with explicit reply-to relations, which to some extent indicates an acceptable accuracy on implicit reply-to prediction. To tackle this issue, we keep the reply-to relations retrieved explicitly as higher priority, i.e., removing the predicted implicit ones which are conflicting with explicit ones.

2) ***Polarity Classification****:* We incorporate the reply-to relations with the comment texts into the *BERT* model [14] for classifying the polarity of each comment. Each comment has parent comments (i.e., comments that are replied to by the current comment) and child comments (i.e., comments that reply to the current comment) after reply-to retrieval, except the first and last ones which only has child or parent comments. Furthermore, we observe that the comment stance is correlated with the role of the commenter (e.g., contributor, collaborator), and the stance switch is often accompanied by the switch of commenter's role. For example, author as the proposer of feature request surely holds the pro stance to the feature. They would elaborate supporting claims in their comments, among which stance is consistent and hardly

switches. While commenters in other roles (e.g., contributors) would provide opposite claims in their comments. In our experimental data, we statistically find that if stance switch occurs in two comments having reply-to relations, 63% of them also having a switch of the commenter's role. Therefore we utilize the commenter's role, i.e., the commenter's roles of the current and the most recent parent/child comment, to represent reply-to relations of the current comment.

The commenter belongs to one of the following roles: *Author*, *Member*, *Contributor*, *Collaborator* and *None*. We add a special tag *Null* to complement the missing reply-to relations. The commenter's roles are discrete values, which need to be embedded into continuous vectors (denoted as r for current role, r_p for parent role, and r_c for child role) to feed them into the model. To this end, we use the embedding layer to represent each value with a 130-dimensional vector, which can be trained jointly with the whole model.

For each comment, we first feed the comment text into the *BERT* to obtain a 768-dimensional vector (denoted as v) which embeds the comment's semantics. Then we concatenate v, r_p, r and r_c ($v \bigoplus r_p \bigoplus r \bigoplus r_c$) to obtain a 1,158-dimensional ($768 + 130 \times 3$) vector (denoted as v'), which embeds the semantics and reply-to relations of current comment simultaneously. Finally, the concatenated vector v' is fed into a dropout layer to avoid over-fitting, and a fully-connected layer which computes the probability vector of polarity labels (i.e., pro/neutral/con). Since this task is a classification task, we use the commonly-used cross entropy as the loss function.

The hyper-parameters in the model are tuned carefully with a greedy strategy to obtain best performance. Given a hyper-parameter P and its candidate values, we perform automated tuning for n iterations, and choose the values which leads to the best performance as the tuned value of P. After tuning, the learning rate is set as 10^{-3}. The optimizer is Adam algorithm [18]. We use the mini-batch technique for speeding up the training process with batch size 32. The drop rate is 0.5, which means 50% of neuron cells will be randomly masked to avoid over-fitting. We implement this module with PyTorch[4].

C. Stance Summarization

VOTEBOT conducts the stance summarization for each comment with three steps, i.e., semantic relevance acquisition, argumentative relation retrieval, and graph-based summary extraction, as shown in Figure 2. We treat this task as an extractive summarization task, i.e., select the most informative sentence as the summary. In detail, it first acquires the semantic relevance of each sentence in a comment with the feature description, and retrieve the argumentative relations among the sentences within a comment. Then together with the textual information of each sentence in a comment, the knowledge is incorporated into the graph-based ranking model for extracting the summary sentence.

1) ***Semantic Relevance Acquisition****:* Different from the general summarization task, the stance summarization involves both the feature description which acts as the discussion topic to some extent, and the discussions (i.e., comment stream). Hence the summary should have high relevance with the proposed feature, and the sentences which are deviated from the feature description would be less likely included in the summary. We therefore acquire the semantic relevance of each sentence in the comment with the feature description, which would be incorporated into the summary extraction model for boosting the performance. More specifically, we firstly encode each comment sentence and the feature description into fixed-dimensional vectors using publicly-released pre-trained BERT [14] model. Then we calculate the cosine similarity score between two sentence vectors as the semantic relevance between comment sentences and feature request.

2) ***Argumentative Relation Retrieval****:* Argumentative relation measures the functional role and contribution of every sentence to the whole documents [19]. A typical taxonomy of argumentative relation includes *MajorClaim*, *Claim*, and *Premise*, where major claim presents the main opinion, while the premise provides the supports for the validity of the claim [20]. We take the argumentative relations into consideration because we observe that the summary sentences usually share similar positional patterns with some argumentative relations. For example, summary sentence might occur more often at the beginnings or endings of comment paragraphs, which is similar as *MajorClaim*.

Specifically, we apply the model of *DiSA* [19] to identify the argumentative relations (i.e., *MajorClaim*, *Claim*, *Premise* and *Other*) of sentences within a comment. *DiSA* is designed as a neural network with self-attentions, and adopted three types of sentence positions (i.e., *local position*, *paragraph position* and *global position*) for positional encoding, which sufficiently portrays the positional characteristics of the argumentative relations, especially for *MajorClaim* which usually conveys the main ideas of documents. We first pre-train *DiSA* on the open-source data provided by the original paper, and then fine-tune it on 800 comments, which are randomly sampled from the issue discussion data of previous work [1] and manually annotated with argumentative relation labels. Note that this data is not included in our experimental dataset. With the fine-tuned model, we take the probability where the argumentative relation is predicted as *MajorClaim* to improve stance summarization.

3) ***Graph-based Summary Extraction****:* Up to now, for each sentence in a comment, we have obtained the semantic relevance with the feature description (denoted as sr) and the argumentative relation within the comment (denoted as ar). Then we construct a graph for each comment, where nodes are sentences and edges are the similarity between the two involved sentences. Specifically, we use the pre-trained BERT [14] model to encode sentences into continuous representations. After normalizing the similarities following the previous practice [4] to alleviate the effect of absolute values, we obtain the similarity scores e_{ij} between sentence s_i and s_j. The centrality for sentence s_i is finally specified as:

[4]https://pytorch.org/

$$centrality(s_i) = (\lambda \sum_{j \neq i}^{n} e_{ij} + (1-\lambda) sr_i) \cdot ar_i \quad (1)$$

where we regard $\sum_{j \neq i}^{n} e_{ij}$ as semantic similarities between s_i and other sentences within the comment, and sr_i as semantic relevance between s_i and feature description. λ denotes the weights for two similarities. The final centrality is the product of argumentative relation probability ar_i and the sum of two similarities, and the sentence with the highest centrality would be the summary of the current comment. The hyper-parameter λ is tuned experimentally on a validation set consisting of comments and corresponding summaries. We implement this module of VOTEBOT with PyTorch.

With two modules above, VOTEBOT detects stance and summarizes each comment in a feature discussion, then concatenates the comment summary sentences of two polarities to produce a pro summary and a con summary, respectively.

IV. EXPERIMENTAL DESIGN

A. Research Questions

We firstly evaluate the effectiveness of VOTEBOT's two individual modules (i.e., stance detection and stance summarization) respectively, and then the VOTEBOT's overall performance (i.e., two-polarity summary), by comparison with corresponding baselines. We also evaluate the necessity of introducing additional relation-aware information into two modules by comparison with variants, which are modified from the VOTEBOT's two modules, respectively. To this end, we set up four research questions. **RQ1** conducts comparison with three commonly-used baselines and two variants to demonstrate the advantage of VOTEBOT on stance detection, and the contribution of reply-to relations. **RQ2** compares with four commonly-used and SOTA baselines and three variants to verify the effectiveness of VOTEBOT on stance summarization, and the contribution of semantic relevance and argumentative relation. **RQ3** verifies the quality of two-polarity summary produced by VOTEBOT via comparison with two SOTA baselines. **RQ4** conducts an evaluation with practitioners to further demonstrate the usefulness of VOTEBOT in real-world practice. The four RQs are as follows.

RQ1 (Advantage on Stance Detection): How effective is VOTEBOT on stance detection, and what is the contribution of reply-to relations to performance?

RQ2 (Advantage on Stance Summarization): How effective is VOTEBOT on stance summarization, and what is the contribution of semantic relevance and argumentative relation to performance?

RQ3 (Overall Advantage): How effective is VOTEBOT on summarizing two-polarity opinions?

RQ4 (Evaluation with Practitioners): How useful is VOTEBOT in real-world practice?

B. Data Preparation

We use the issue reports of five projects from GitHub in our experiments. All five projects are popular and widely-used by a large number of users with at least 30,000+ stars. We first crawl the issue reports and corresponding comments which are tagged by the label *feature request* from GitHub with the tool *PyGitHub*[5], which is a Python library to access to GitHub resources. For each project, we randomly select 50 closed issues with over 20 comments. We include them in dataset because the comment stream is more lengthy, and the opinions are stated more completely in such issues, which would facilitate a fuller evaluation on VOTEBOT. Finally, we construct a dataset including 250 issue reports (50 for each project) with 6,598 comments in total as shown in Table I. Then we manually label the data for further experiments following the process as described below.

TABLE I
EXPERIMENTAL DATASET.

Project	#Feature Request	#Comment
Angular [21]	50	684
Godot [22]	50	2,768
PowerShell [23]	50	866
TypeScript [24]	50	1,195
Vue [25]	50	1,085
Total	250	6,598

For each feature request, we label each comment by assigning one of pro, con and neutral stance labels (we deem the comment in favor of feature request as pro stance), and select one sentence as the summary if the comment is labeled as pro or con. We concatenate the summary sentences of pro and con comments to create the ground-truth summary of two polarities for a feature request. To guarantee the accuracy of the labeling outcomes, we invite three annotators who have rich web development experience to conduct annotation following a two-round process: Two annotators firstly label the comments of one project independently. Second, the third annotator compares the labeling results, finds the difference, and organizes a face-to-face discussion among them three to determine the final label. The agreement between two annotators in the first round reaches 0.79 of Cohen's Kappa for stance detection task, and 0.73 of Cohen's Kappa for stance summarization task. After two rounds of labeling, a common consensus is reached for every comment.

We have 250 feature requests with corresponding 6,598 comments in total. After labeling, we obtain 877 comments labeled as pro, 770 as con, and the remaining 4,951 are neutral ones. Note that we put the comments which are not expressing the stance towards feature request and unrelated to the discussion topics in the neutral category as well.

C. Baselines

*1) **Baselines for RQ1**:* Existing works usually tackle stance detection based on basic deep neural network, and introduce external knowledge according to specific application contexts. We choose three typical deep neural networks as baselines.

Stanford CoreNLP [26]: A method built on top of a recursive neural network, which classifies the text's sentiment on a five-value scale. We consider very positive and positive as pro, very negative and negative as con, and neutral unchanged.

TextCNN [27]: The convolutional neural network for text classification which has been proven to be useful in many

[5]https://github.com/PyGithub/PyGithub/

natural language processing (NLP) tasks. We train it on our comment data with stance polarity labels.

BERT Classifier [14]: The BERT classifier utilizing a BERT model for representing sentences and a multi-layer perceptron for computing the probabilities of stance polarities. We fine-tune it on our comment data with stance polarity labels.

*2) **Baselines for RQ2**:* We select four commonly-used and SOTA summarization techniques as baselines.

Random: A simple baseline by randomly selecting one sentence as the summary of the comment.

Lead-K: A commonly-used baseline for extractive summarization by selecting the first K (K=1 in our context) sentences as the summary of the comment.

TextRank [28]: An unsupervised algorithm which represents sentences as nodes in a graph with undirected edges whose weights are computed based on sentence similarity. We treat the sentence with the highest score as the comment summary.

PacSum [4]: The SOTA centrality-based model which employs BERT to better capture sentential meaning, and builds graphs with directed edges. We select the sentence with the highest centrality as the comment summary.

*3) **Baselines for RQ3**:* Since there are no existing approaches directly for this task, we utilize two SOTA approaches for similar tasks as baselines.

OpenIE [29]: The SOTA approach for systematically discovering, explaining and summarizing controversies in community Q&A discussions. We employ its first step, which employs the sentence matching for discovering controversies, to conduct the stance detection, and the third step, which utilizes TextRank for summarizing controversies, to conduct the stance summarization.

ChatEO [30]: The SOTA approach for automatically extracting opinion Q&A from software developer chats. We employ its second step, which applies textual heuristics for opinion-asking questions identification, to conduct the stance detection, and the third step, which uses a deep learning-based approach for answer extraction from conversations, to conduct the stance summarization.

D. Experimental Setup

Overall Setup. In our labeled dataset, each comment has the labels of stance polarity and summary sentence, and each feature request includes a stream of comments, and the ground-truth summary of two stance polarities by respectively concatenating the summary sentences of them. We conduct VOTEBOT's experimental evaluation on this dataset with the same 10-fold cross validation in RQ1-RQ3. We randomly divide the dataset into ten folds in feature request level, use eight of them as training set, one as validation set, and one as testing set. In each fold, we gather and balance the comment data with stance polarity labels in the training set to train the stance detection module, and tune the hyper-parameters with the comment data in the validation set; Simultaneously we gather the comment data with summary sentences in training set plus validation set to tune the hyper-parameters of unsupervised stance summarization module; Finally, VOTEBOT with two trained modules assigns polarity labels and extracts summary sentences comment by comment in testing set, and generates a pro summary and a con summary for each feature request via concatenating the comment summary sentences of two polarities. By comparing with the ground-truths labels, we can evaluate the respective performance of two tasks in comment level (RQ1 and RQ2), and the overall performance in feature request level (RQ3). The above process is repeated ten times, and the average metric is treated as the final performance.

We run all baselines and variants of VOTEBOT following the same 10-fold cross-validation above. In **RQ1**, we conduct ablation experiment with two variants, i.e., VOTEBOT$_{exp}$ which only retrieves explicit reply-to relations, and VOTEBOT$_{imp}$ which only retrieves implicit reply-to relations. The baseline of BERT classifier is also treated as basic version with no reply-to relations. In **RQ2**, we conduct the ablation experiment with three variants, i.e., VOTEBOT$_{sr}$ which only introduces semantic relevance, VOTEBOT$_{ar}$ which only introduces argumentative relation, and VOTEBOT$_{base}$ which introduces neither of them. In **RQ3**, we demonstrate the performance of VOTEBOT on summarizing the opinions for two stance polarities by running two baselines. In **RQ4**, we apply VOTEBOT on feature requests from five new projects: React, Bootstrap, Flutter, Electron and Kubernetes. All projects have 80,000+ more stars signifying their popularity. Note that they are different from our experimental projects, and all data do not appear in our training/testing data. Then we recruit eight participants to assess VOTEBOT from four criteria: *Informativeness*, *Polarity*, *Coherence* and *Usefulness*.

E. Evaluation Metrics

For **RQ1**, since stance detection task is a multi-class classification task, we use *Precision* and *Recall* to evaluate the performance for each polarity label. Precision is the ratio of the number of correct predictions to the total number of predictions. Recall is the ratio of the number of correct predictions to the total number of ground-truth samples.

For **RQ2**, since the stance summarization task is to select one sentence as the comment summary, we use *Accuracy* to evaluate the performance by calculating the ratio of the number of correct selections to the total number of samples.

For **RQ3**, we evaluate summarization quality automatically using ROUGE F1 [31] by comparing the ground-truth and predicted summaries of two stance polarities, respectively [4], [5]. We report unigram and bigram overlap (i.e., ROUGE-1 and ROUGE-2) as a means of assessing informativeness, and the longest common subsequence (i.e., ROUGE-L) as a means of assessing fluency.

V. RESULTS AND ANALYSIS

A. Answering RQ1 (Advantage on Stance Detection)

Table II presents the precision and recall of VOTEBOT on stance detection by comparison with three baselines and two variants. We first look at the overall performance comparison with baselines (i.e., CoreNLP, TextCNN and BERT) as shown in the first three columns of Table II. We can observe that

TABLE II
EVALUATION (PRECISION AND RECALL) ON STANCE DETECTION (RQ1).

Stance	CoreNLP		TextCNN		BERT		VoteBot$_{exp}$		VoteBot$_{imp}$		VoteBot	
	P	R	P	R	P	R	P	R	P	R	P	R
Pro	39.2%	42.8%	54.8%	57.6%	52.3%	63.0%	81.1%	69.5%	76.4%	74.9%	81.2%	73.6%
Neutral	33.4%	43.4%	74.5%	77.0%	75.8%	86.1%	78.0%	87.8%	77.1%	87.5%	78.1%	86.0%
Con	38.6%	24.9%	61.3%	54.3%	69.7%	40.5%	74.6%	74.6%	73.7%	71.7%	76.8%	75.8%
Overall	39.3%	40.4%	63.5%	63.1%	65.7%	62.2%	78.0%	76.4%	77.2%	76.8%	**79.1%**	**78.0%**

TABLE III
EVALUATION (ACCURACY) ON STANCE SUMMARIZATION (RQ2).

Project	Random	LEAD-K	TextRank	PacSum	VoteBot$_{base}$	VoteBot$_{sr}$	VoteBot$_{ar}$	VoteBot
Angular	42.78%	46.46%	42.78%	50.14%	48.44%	59.21%	78.19%	**82.44%**
Godot	34.85%	40.15%	45.20%	44.19%	42.93%	54.29%	75.51%	**79.80%**
PowerShell	33.83%	35.32%	39.41%	40.52%	38.66%	49.44%	72.49%	**80.67%**
TypeScript	36.70%	33.71%	39.33%	41.95%	40.07%	53.56%	74.91%	**80.52%**
Vue	39.50%	44.48%	44.48%	46.13%	45.03%	57.18%	77.62%	**82.60%**
Average	37.53%	40.02%	42.24%	44.59%	43.03%	54.74%	75.74%	**81.20%**

VOTEBOT outperforms all baselines on all metrics in a large margin. The overall precision and recall reach 79.1% and 78.0%, respectively, which outperforms the best baseline of BERT by 20.40% in precision and 25.40% in recall. The performance of CoreNLP is not good although it is a deep learning-based method. This is because it is originally trained on a movie review corpus to detect sentiment of the text, which cannot work well in the context of stance detection. The performance of TextCNN is roughly in the middle. This is because although it is trained on our data, the simple neural network cannot capture the textual semantics accurately. BERT which is fine-tuned on our data, achieves the best performance among baselines, but there is still a margin compared with VOTEBOT. BERT can be treated as a basic version of VOTEBOT, the performance difference indicates that simply turning the stance detection into a text classification task, and constructing a deep learning model only with textual semantics cannot work well.

As for the performance across stance labels, we can observe that VOTEBOT achieves the highest precision on pro labels and the highest recall on neutral labels, and the highest F1 appears on neutral labels as well. This indicates the semantics of neutral comments can be easily distinguished from pro/con comments by the model, while it is hard to distinguish them between pro and con comments, which further indicates the difficulties of the this task. We also examine VOTEBOT's performance across projects, which is better on Angular and Vue than that on other projects. After checking results manually, we find that there are more explicit reply-to relations among comments of these two projects. This indicates that better reply-to retrieval would boost the performance, which proves the contribution brought by reply-to relations to some extent.

Furthermore, we compare the performance between VOTEBOT and its two variants to verify the influence of reply-to relations. The comparison between the basic version BERT and VOTEBOT$_{exp}$/VOTEBOT$_{imp}$ indicates adding the reply-to relations can boost the overall performance no matter they are explicit or implicit, especially the big improvement on pro/con labels. The comparison between VOTEBOT and VOTEBOT$_{exp}$/VOTEBOT$_{imp}$ indicates the contribution of explicit and implicit reply-to relations overlaps to some extent, i.e., the increased performance by each type is not simply added up to the the whole model's performance. This is reasonable considering the retrieved implicit reply-to relations overlaps with some explicit ones as well.

Overall, VOTEBOT achieves the best performance on stance detection from perspectives of both labels and projects. The incorporation of reply-to relations improves the performance to varying degrees.

B. Answering RQ2 (Advantage on Stance Summarization)

Table III presents the accuracy of VOTEBOT on stance summarization by comparison with four baselines and three variants. We first look at the performance comparison with baselines as shown in the first four columns of Table III. We can observe that VOTEBOT outperforms all baselines on both project and average accuracy, which reaches 81.20% accuracy on average. The random baseline obtains the worst performance and sets a lower limit for other methods. The LEAD-K baseline obtains the second worst performance, which indicates the summary sentences in our dataset are not simply the first sentences of the comment. The TextRank and PacSum are both graph and centrality-based methods, but they cannot work well without introducing semantic relevance and argumentative relation. Another observation is that the performance across projects of VOTEBOT is more stable than baselines. This is because the incorporated semantic relevance and argumentative relation are project-independent features.

Then we turn to the performance comparison between VOTEBOT and its three variants. VOTEBOT$_{base}$ obtains the worst performance, because it only takes the structures among comment sentences when computing centrality, and cannot select the most salient sentence well. VOTEBOT$_{sr}$ only introduces semantic relevance between comment sentence and feature description on top of VOTEBOT$_{base}$, leading to 27.21% performance improvement. The comparison between VOTEBOT$_{base}$ and VOTEBOT$_{ar}$ indicates that only introducing the argumentative relation, which implies the positional patterns, would improve the performance by 76.02%. The different improvement between two above variants illustrates that the argumentative relation can bring more contribution compared with semantic relevance. While VOTEBOT which introduces both features achieves the best performance with the

TABLE IV
EVALUATION (ROUGE-1/2/L) ON OVERALL PERFORMANCE (RQ3).

Project	OpenIE						ChatEO						VoteBot					
	ROUGE-1		ROUGE-2		ROUGE-L		ROUGE-1		ROUGE-2		ROUGE-L		ROUGE-1		ROUGE-2		ROUGE-L	
	pro	con	pro	con	pro	con	pro	con	pro	con	pro	con	pro	con	pro	con	pro	con
Angular	29.8	43.9	13.1	27.0	27.5	41.2	54.3	37.1	43.0	26.1	53.0	35.7	62.3	64.9	55.2	57.9	61.2	63.8
Godot	39.1	48.8	24.8	31.3	37.2	46.4	56.8	45.7	45.0	33.8	55.5	44.5	62.9	67.1	56.6	57.3	62.0	66.1
PowerShell	29.4	40.0	17.6	26.5	26.8	37.8	44.2	40.5	31.8	28.9	41.8	38.0	60.2	63.7	52.3	55.4	59.3	62.3
TypeScript	30.7	39.8	17.3	23.4	28.3	37.0	52.9	48.7	43.1	37.9	51.7	47.6	53.2	59.3	44.6	51.1	51.6	58.2
Vue	29.2	49.8	13.7	34.6	26.3	47.4	51.3	38.7	40.1	26.2	49.9	36.8	60.9	70.4	52.9	62.8	59.5	69.3
Overall	31.7	44.7	17.3	28.7	29.3	42.1	52.1	42.3	40.8	30.7	50.6	40.7	**60.1**	**65.3**	**52.5**	**57.1**	**58.9**	**64.2**

improvement of 88.71%, 48.34% and 7.21% compared with VoteBot$_{base}$, VoteBot$_{sr}$ and VoteBot$_{ar}$, respectively.

Overall, VoteBot outperforms four baselines on stance summarization. Incorporating semantic relevance and argumentative relation boosts the performance further.

C. Answering RQ3 (Overall Advantage)

Table IV presents the ROUGE-1/2/L of VoteBot and two baselines to evaluate the quality of produced pro and con summaries. The final performance is influenced by the results of both modules. Namely, the approach might correctly classify the polarity but select the wrong summary sentence, or correctly select the summary sentence but assign the wrong polarity. Both situations would harm the result's correctness.

VoteBot achieves the best performance for both pro and con summaries on all metrics, which indicates that our approach can conduct both stance detection and stance summarization accurately. VoteBot achieves the overall ROUGE-1 of 60.1 and 65.3 on pro and con summaries respectively, which is a promising performance compared with similar extractive summarization approaches [4], [5]. It improves the best ROUGE-1 on pro summaries (52.1 from ChatEO) by 15.36%, and improves the best ROUGE-1 on con summaries (44.7 from OpenIE) by 46.09%. We check the results from each approach, and observe there are many cases two baselines classify the wrong stance polarity or/and select the wrong sentence as the summary, which both harm the accuracy of the final summary generation. OpenIE usually extracts the wrong sentences from correctly classified comments, which indicates its stance summarization module cannot work well. While ChatEO usually extracts the summary sentences correctly, but tends to assign it to wrong stance polarities, which indicates the poor capability of its stance detection module.

Overall, the quality of pro and con summaries depends on the accuracy of both modules. VoteBot outperforms two baselines on all metrics.

D. Answering RQ4 (Evaluation with Practitioners)

Participants. Since the five projects for human evaluation are mostly web development frameworks, we recruit participants with at least three years of web development experience working with at least three of these five projects. Finally, eight participants are recruited, including three PhD. students, three professional developers and two senior researchers.

Procedure. First, we crawl the most recent 5 closed issue reports which are tagged by *feature request* from GitHub for each project (totally 25 issue reports). Then the annotators who label the experimental dataset conduct the manual stance detection and summarization following the same process as described in Section IV-B, to produce the gold reference summaries. Second, we apply Best-Worst Scaling [32] for human evaluation, referring to the practice in previous summarization study [33]. It is a less labour-intensive alternative to paired comparisons which has been shown to produce more reliable results than rating scales [34]. More concretely, each of eight participants receives a questionnaire containing all 25 feature requests, and each feature request involves four summaries produced from gold reference, VoteBot, OpenIE, and ChatEO, respectively. The four summaries are in random order, and the participants are not informed which summary is produced by which method. Each participant is asked to decide the best and the worst summary from four competing summaries of 25 issue reports according to four criteria: (1) *Informativeness*: Whether the summary covers the informative opinions the comment expects to express? (2) *Polarity*: Whether the summary is classified into correct polarities? (3) *Coherence*: Whether the summary is coherent and easy to read? (4) *Usefulness*: Whether the summary can help readers understand the pros and cons about a new feature request quickly without viewing the original information? Third, we collect the feedback and calculate the score of each method for every criterion as the percentage of times it is selected as best minus the percentage of times it is selected as worst. We take the average score from eight participants as the final score, which ranges from -100 (worst) to +100 (best).

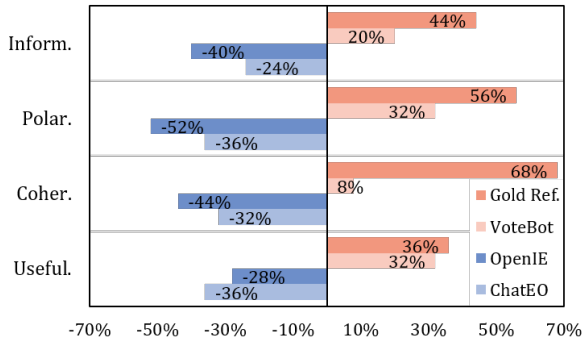

Fig. 3. Results of evaluation with practitioners (RQ4).

Results. Figure 3 presents the results of human evaluation. Participants favored our approach over two baselines (i.e., OpenIE and ChatEO) across all criteria. The biggest difference appears on the polarity criterion, because two baselines usually assign wrong polarity labels to summary sentences. Gold reference summaries are generally preferred over our

model, where the biggest gap appears in terms of coherence, because VOTEBOT creates the final summaries via mechanically concatenating the summary sentences, which neglects the coherence among summary sentences. Overall, the small difference on the usefulness criterion verify the effectiveness of VOTEBOT on summarizing the opinions of two polarities.

VI. THREATS TO VALIDITY.

The ***external threats*** concern the generality of the proposed approach. We train and evaluate VOTEBOT on the dataset consisting of feature requests from five projects on GitHub. The selected projects are all the widely-used ones with at least 30,000+ stars, which relatively reduces this threat. In addition, we apply VOTEBOT on five new projects and conduct an evaluation with practitioners. The results are promising, which verifies its generality to some extent. Even so, VOTEBOT still has some limitations on generality. It is primarily workable on issue reports about feature requests, which have distinct stance polarities. Additionally, it mainly targets textual comments in natural language, lacking the capability of understanding data in other formats mixed in comments, e.g., code snippets and images. The idea of incorporating the relation-aware contextual information to better understand discussion-style comment streams, and the unsupervised working manners of summarization module can be adapted to other contexts easily.

Regarding ***internal threats***, we reuse the source code from original paper (e.g., *OpenIE* and *ChatEO*), or the open source implementations (e.g., *CoreNLP, TextCNN and BERT*) for the baselines, which ensures the accuracy of experimental results.

The ***construct validity*** of this study mainly questions the evaluation metrics. We utilize ROUGE F1 to evaluate the quality of summaries, which is commonly-used automatic metrics for evaluating summarization systems [4], [5].

VII. RELATED WORK

Stance Detection: In NLP community, there has been a growing interest in stance detection on social media since the SemEval-2016 dataset [9] is released, and the models are improved from rule-based algorithms and feature-based machine learning approaches [35], [36] to deep neural networks (e.g., RNNs, CNNs, LSTMs and their modified versions) along with attention mechanism [11], [37], [38] to learn latent representation and relationship between target and text. Another line of works supplement the contextual semantic information by external knowledge bases, i.e., commonsense knowledge bases [39], Wiki concepts [12], the semantic lexicon and the emotional lexicon [40], and by fine-tuning pre-trained language models (e.g., BERT) [41]. In this work, we detect the stance of comments in feature requests by introducing contextual reply-to relations to boost the performance.

Sentiment analysis as a closely related task to stance detection, has been applied on different Software Engineering (SE) artifacts, such as technical artifacts (e.g., issues [42] and commit messages [43]) and crowd-generated content (e.g., forum messages [44] and users' reviews [45]–[47]), to support different tasks. There are several sentiment analysis tools available applied to SE applications. Most prior works leverage sentiment analysis tools (e.g., NLTK [48], CoreNLP [26]) not designed for software-related textual documents, thus leading to the poor accuracy when applied in different contexts. The recent trend is to customize existing sentiment analysis tools to properly work on SE datasets (e.g., SentiStrength-SE [49]). Other previous works related to text classification in SE include classifying app reviews [2], [50], bug reports [51], [52], etc. The approaches for sentiment analysis and SE text classification cannot work well in our context, because the stance might not be conveyed explicitly in the texts.

Summarization in SE: In SE community, there are many studies focusing on summarizing various software artifacts. For example, summarizing bug reports with a classifier based on conversational data [53], [54], with an unsupervised approach based on noise reducer [55], and with heuristic rules [56], etc. Other artifacts include summarizing comments on Q&A forum with NLP techniques [57], summarizing app reviews based on a scoring mechanism [58], summarizing opinions in API reviews [59], and summarizing solutions for crash bugs [60], etc. Another line of studies focus on mining opinions in SE, including identifying and classifying opinions on APIs from Stack Overflow [61], identifying negative opinions about APIs from forums [44], distilling and aggregating comparative opinions of comparable technologies from Q&A websites [62], discovering and summarizing controversial (criticized) answers in Stack Overflow [29], extracting opinion-based Q&A from developer chats [30], and analyzing opinions in developer emails [63]. Different from the above studies, our work aims at summarizing the comments in feature requests, and producing summaries by the stance.

VIII. CONCLUSION

To enhance the decision-making process for accepting feature requests, this paper proposes an automated approach for conducting stance detection and summarization on discussions related to feature requests. The proposed system, called VOTEBOT, consists of two modules: one incorporates reply-to relations to enhance the BERT-based classifier for stance detection, and the other incorporates semantic relevance and argumentative relations to improve the graph-based ranking for stance summarization. We have evaluated VOTEBOT on 250 feature requests comprising 6,598 comments from five projects hosted on GitHub. Additionally, we conducted further evaluations with practitioners on five new projects. The results confirm the effectiveness and usefulness of VOTEBOT in both experimental setups and real-world practice. In future research, we plan to explore the representation and integration of code snippets and images within comment texts, aiming to gain a better understanding of discussions about feature requests.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China Grant No.62232016, No.62072442 and No.62002348, and Youth Innovation Promotion Association Chinese Academy of Sciences.

REFERENCES

[1] D. M. Arya, W. Wang, J. L. C. Guo, and J. Cheng, "Analysis and detection of information types of open source software issue discussions," in *Proceedings of the 41st International Conference on Software Engineering, ICSE 2019, Montreal, QC, Canada, May 25-31, 2019*, 2019, pp. 454–464.

[2] W. Maalej and H. Nabil, "Bug report, feature request, or simply praise? on automatically classifying app reviews," in *23rd IEEE International Requirements Engineering Conference, RE 2015, Ottawa, ON, Canada, August 24-28, 2015*, 2015, pp. 116–125.

[3] P. Sobhani, S. M. Mohammad, and S. Kiritchenko, "Detecting stance in tweets and analyzing its interaction with sentiment," in *Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics, *SEM@ACL 2016, Berlin, Germany, 11-12 August 2016*. The *SEM 2016 Organizing Committee, 2016.

[4] H. Zheng and M. Lapata, "Sentence centrality revisited for unsupervised summarization," in *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*. Association for Computational Linguistics, 2019, pp. 6236–6247.

[5] X. Liang, S. Wu, M. Li, and Z. Li, "Improving unsupervised extractive summarization with facet-aware modeling," in *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021*, ser. Findings of ACL, vol. ACL/IJCNLP 2021. Association for Computational Linguistics, 2021, pp. 1685–1697.

[6] S. Somasundaran and J. Wiebe, "Recognizing stances in online debates," in *ACL 2009, Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the AFNLP, 2-7 August 2009, Singapore*. The Association for Computer Linguistics, 2009, pp. 226–234.

[7] J. Kobbe, I. Hulpus, and H. Stuckenschmidt, "Unsupervised stance detection for arguments from consequences," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*. Association for Computational Linguistics, 2020, pp. 50–60.

[8] M. Thomas, B. Pang, and L. Lee, "Get out the vote: Determining support or opposition from congressional floor-debate transcripts," in *EMNLP 2006, Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing, 22-23 July 2006, Sydney, Australia*. ACL, 2006, pp. 327–335.

[9] S. M. Mohammad, S. Kiritchenko, P. Sobhani, X. Zhu, and C. Cherry, "Semeval-2016 task 6: Detecting stance in tweets," in *Proceedings of the 10th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2016, San Diego, CA, USA, June 16-17, 2016*. The Association for Computer Linguistics, 2016, pp. 31–41.

[10] Y. Jiang, J. Gao, H. Shen, and X. Cheng, "Few-shot stance detection via target-aware prompt distillation," in *SIGIR '22: The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, July 11 - 15, 2022*. ACM, 2022, pp. 837–847.

[11] T. Hercig, P. Krejzl, B. Hourová, J. Steinberger, and L. Lenc, "Detecting stance in czech news commentaries," in *Proceedings of the 17th Conference on Information Technologies - Applications and Theory (ITAT 2017), Martinské hole, Slovakia, September 22-26, 2017*, ser. CEUR Workshop Proceedings, vol. 1885. CEUR-WS.org, 2017, pp. 176–180.

[12] K. Hanawa, A. Sasaki, N. Okazaki, and K. Inui, "Stance detection attending external knowledge from wikipedia," *J. Inf. Process.*, vol. 27, pp. 499–506, 2019.

[13] Y. Dong, A. Romascanu, and J. C. K. Cheung, "Discourse-aware unsupervised summarization for long scientific documents," in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021*. Association for Computational Linguistics, 2021, pp. 1089–1102.

[14] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, 2019, pp. 4171–4186.

[15] D. Shen, Q. Yang, J. Sun, and Z. Chen, "Thread detection in dynamic text message streams," in *SIGIR 2006: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Seattle, Washington, USA, August 6-11, 2006*. ACM, 2006, pp. 35–42.

[16] J. K. Kummerfeld, S. R. Gouravajhala, J. Peper, V. Athreya, R. C. Gunasekara, J. Ganhotra, S. S. Patel, L. C. Polymenakos, and W. S. Lasecki, "A large-scale corpus for conversation disentanglement," in *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*. Association for Computational Linguistics, 2019, pp. 3846–3856.

[17] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*. ACL, 2014, pp. 1532–1543.

[18] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[19] W. Song, Z. Song, R. Fu, L. Liu, M. Cheng, and T. Liu, "Discourse self-attention for discourse element identification in argumentative student essays," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*. Association for Computational Linguistics, 2020, pp. 2820–2830.

[20] C. Stab and I. Gurevych, "Identifying argumentative discourse structures in persuasive essays," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*. ACL, 2014, pp. 46–56.

[21] https://github.com/angular/angular/.

[22] https://github.com/godotengine/godot/.

[23] https://github.com/PowerShell/PowerShell/.

[24] https://github.com/microsoft/TypeScript/.

[25] https://github.com/vuejs/vue/.

[26] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*. ACL, 2013, pp. 1631–1642.

[27] Y. Kim, "Convolutional neural networks for sentence classification," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*. ACL, 2014, pp. 1746–1751.

[28] R. Mihalcea and P. Tarau, "Textrank: Bringing order into text," in *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing , EMNLP 2004, A meeting of SIGDAT, a Special Interest Group of the ACL, held in conjunction with ACL 2004, 25-26 July 2004, Barcelona, Spain*. ACL, 2004, pp. 404–411.

[29] X. Ren, Z. Xing, X. Xia, G. Li, and J. Sun, "Discovering, explaining and summarizing controversial discussions in community q&a sites," in *34th IEEE/ACM International Conference on Automated Software Engineering, ASE 2019, San Diego, CA, USA, November 11-15, 2019*. IEEE, 2019, pp. 151–162.

[30] P. Chatterjee, K. Damevski, and L. L. Pollock, "Automatic extraction of opinion-based q&a from online developer chats," in *43rd IEEE/ACM International Conference on Software Engineering, ICSE 2021, Madrid, Spain, 22-30 May 2021*. IEEE, 2021, pp. 1260–1272.

[31] C. Lin and E. H. Hovy, "Automatic evaluation of summaries using n-gram co-occurrence statistics," in *Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, HLT-NAACL 2003, Edmonton, Canada, May 27 - June 1, 2003*. The Association for Computational Linguistics, 2003.

[32] J. Louviere, T. Flynn, and A. A. J. Marley, *Best-worst scaling: Theory, methods and applications*. Cambridge University Press, 01 2015.

[33] R. K. Amplayo, S. Angelidis, and M. Lapata, "Unsupervised opinion summarization with content planning," in *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*. AAAI Press, 2021, pp. 12 489–12 497.

[34] S. Kiritchenko and S. M. Mohammad, "Best-worst scaling more reliable than rating scales: A case study on sentiment intensity annotation," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 2: Short Papers*. Association for Computational Linguistics, 2017, pp. 465–470.

[35] H. Bøhler, P. Asla, E. Marsi, and R. Sætre, "Idi$@$ntnu at semeval-2016 task 6: Detecting stance in tweets using shallow features and glove vectors for word representation," in *Proceedings of the 10th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2016, San Diego, CA, USA, June 16-17, 2016*. The Association for Computer Linguistics, 2016, pp. 445–450.

[36] M. Tutek, I. Sekulic, P. Gombar, I. Paljak, F. Culinovic, F. Boltuzic, M. Karan, D. Alagic, and J. Snajder, "Takelab at semeval-2016 task 6: Stance classification in tweets using a genetic algorithm based ensemble," in *Proceedings of the 10th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2016, San Diego, CA, USA, June 16-17, 2016*. The Association for Computer Linguistics, 2016, pp. 464–468.

[37] J. Du, R. Xu, Y. He, and L. Gui, "Stance classification with target-specific neural attention," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*. ijcai.org, 2017, pp. 3988–3994.

[38] R. Seoh, I. Birle, M. Tak, H. Chang, B. Pinette, and A. Hough, "Open aspect target sentiment classification with natural language prompts," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*. Association for Computational Linguistics, 2021, pp. 6311–6322.

[39] R. Liu, Z. Lin, Y. Tan, and W. Wang, "Enhancing zero-shot and few-shot stance detection with commonsense knowledge graph," in *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021*, ser. Findings of ACL, vol. ACL/IJCNLP 2021. Association for Computational Linguistics, 2021, pp. 3152–3157.

[40] B. Zhang, M. Yang, X. Li, Y. Ye, X. Xu, and K. Dai, "Enhancing cross-target stance detection with transferable semantic-emotion knowledge," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*. Association for Computational Linguistics, 2020, pp. 3188–3197.

[41] K. Popat, S. Mukherjee, A. Yates, and G. Weikum, "STANCY: stance classification based on consistency cues," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*. Association for Computational Linguistics, 2019, pp. 6412–6417.

[42] F. Jurado and P. R. Marín, "Sentiment analysis in monitoring software development processes: An exploratory case study on github's project issues," *J. Syst. Softw.*, vol. 104, pp. 82–89, 2015.

[43] E. Guzman, D. Azócar, and Y. Li, "Sentiment analysis of commit comments in github: an empirical study," in *11th Working Conference on Mining Software Repositories, MSR 2014, Proceedings, May 31 - June 1, 2014, Hyderabad, India*. ACM, 2014, pp. 352–355.

[44] Y. Zhang and D. Hou, "Extracting problematic API features from forum discussions," in *IEEE 21st International Conference on Program Comprehension, ICPC 2013, San Francisco, CA, USA, 20-21 May, 2013*. IEEE Computer Society, 2013, pp. 142–151.

[45] L. V. G. Carreño and K. Winbladh, "Analysis of user comments: an approach for software requirements evolution," in *35th International Conference on Software Engineering, ICSE '13, San Francisco, CA, USA, May 18-26, 2013*. IEEE Computer Society, 2013, pp. 582–591.

[46] E. Guzman and W. Maalej, "How do users like this feature? A fine grained sentiment analysis of app reviews," in *IEEE 22nd International Requirements Engineering Conference, RE 2014, Karlskrona, Sweden, August 25-29, 2014*. IEEE Computer Society, 2014, pp. 153–162.

[47] S. Panichella, A. D. Sorbo, E. Guzman, C. A. Visaggio, G. Canfora, and H. C. Gall, "How can i improve my app? classifying user reviews for software maintenance and evolution," in *2015 IEEE International Conference on Software Maintenance and Evolution, ICSME 2015, Bremen, Germany, September 29 - October 1, 2015*. IEEE Computer Society, 2015, pp. 281–290.

[48] C. J. Hutto and E. Gilbert, "VADER: A parsimonious rule-based model for sentiment analysis of social media text," in *Proceedings of the Eighth International Conference on Weblogs and Social Media, ICWSM 2014, Ann Arbor, Michigan, USA, June 1-4, 2014*. The AAAI Press, 2014.

[49] M. R. Islam and M. F. Zibran, "Sentistrength-se: Exploiting domain specificity for improved sentiment analysis in software engineering text," *J. Syst. Softw.*, vol. 145, pp. 125–146, 2018.

[50] S. Panichella, A. D. Sorbo, E. Guzman, C. A. Visaggio, G. Canfora, and H. C. Gall, "How can i improve my app? classifying user reviews for software maintenance and evolution," in *2015 IEEE International Conference on Software Maintenance and Evolution, ICSME 2015, Bremen, Germany, September 29 - October 1, 2015*, 2015, pp. 281–290.

[51] H. Liu, M. Shen, J. Jin, and Y. Jiang, "Automated classification of actions in bug reports of mobile apps," in *ISSTA '20: 29th ACM SIGSOFT International Symposium on Software Testing and Analysis, Virtual Event, USA, July 18-22, 2020*, 2020, pp. 128–140.

[52] W. Junjie, C. Qiang, W. Song, and W. Qing, "Domain adaptation for test report classification in crowdsourced testing," in *ICSE'17*, 2017, pp. 83–92.

[53] S. Rastkar, G. C. Murphy, and G. Murray, "Summarizing software artifacts: a case study of bug reports," in *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1, ICSE 2010, Cape Town, South Africa, 1-8 May 2010*. ACM, 2010, pp. 505–514.

[54] ——, "Automatic summarization of bug reports," *IEEE Trans. Software Eng.*, vol. 40, no. 4, pp. 366–380, 2014.

[55] S. Mani, R. Catherine, V. S. Sinha, and A. Dubey, "AUSUM: approach for unsupervised bug report summarization," in *20th ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE-20), SIGSOFT/FSE'12, Cary, NC, USA - November 11 - 16, 2012*. ACM, 2012, p. 11.

[56] R. Lotufo, Z. Malik, and K. Czarnecki, "Modelling the 'hurried' bug report reading process to summarize bug reports," *Empir. Softw. Eng.*, vol. 20, no. 2, pp. 516–548, 2015.

[57] E. Wong, J. Yang, and L. Tan, "Autocomment: Mining question and answer sites for automatic comment generation," in *2013 28th IEEE/ACM International Conference on Automated Software Engineering, ASE 2013, Silicon Valley, CA, USA, November 11-15, 2013*. IEEE, 2013, pp. 562–567.

[58] A. D. Sorbo, S. Panichella, C. V. Alexandru, J. Shimagaki, C. A. Visaggio, G. Canfora, and H. C. Gall, "What would users change in my app? summarizing app reviews for recommending software changes," in *Proceedings of the 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering, FSE 2016, Seattle, WA, USA, November 13-18, 2016*. ACM, 2016, pp. 499–510.

[59] G. Uddin and F. Khomh, "Automatic summarization of API reviews," in *Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering, ASE 2017, Urbana, IL, USA, October 30 - November 03, 2017*. IEEE Computer Society, 2017, pp. 159–170.

[60] H. Wang, X. Xia, D. Lo, J. C. Grundy, and X. Wang, "Automatic solution summarization for crash bugs," in *43rd IEEE/ACM International Conference on Software Engineering, ICSE 2021, Madrid, Spain, 22-30 May 2021*. IEEE, 2021, pp. 1286–1297.

[61] B. Lin, F. Zampetti, G. Bavota, M. D. Penta, and M. Lanza, "Pattern-based mining of opinions in q&a websites," in *Proceedings of the 41st International Conference on Software Engineering, ICSE 2019, Montreal, QC, Canada, May 25-31, 2019*. IEEE / ACM, 2019, pp. 548–559.

[62] 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, ASE 2018, Montpellier, France, September 3-7, 2018*. ACM, 2018, pp. 214–224.

[63] Z. Xiong, P. Liang, C. Yang, and T. Liu, "Assumptions in OSS development: An exploratory study through the hibernate developer mailing list," in *25th Asia-Pacific Software Engineering Conference, APSEC 2018, Nara, Japan, December 4-7, 2018*. IEEE, 2018, pp. 455–464.