Automatic Extraction of Opinion-based Q&A from Online Developer Chats

Preetha Chatterjee
University of Delaware
Newark, DE, USA
preethac@udel.edu

Kostadin Damevski
Virginia Commonwealth University
Richmond, VA, USA
kdamevski@vcu.edu

Lori Pollock
University of Delaware
Newark, DE, USA
pollock@udel.edu

***Abstract*—Virtual conversational assistants designed specifically for software engineers could have a huge impact on the time it takes for software engineers to get help. Research efforts are focusing on virtual assistants that support specific software development tasks such as bug repair and pair programming. In this paper, we study the use of online chat platforms as a resource towards collecting developer opinions that could potentially help in building opinion Q&A systems, as a specialized instance of virtual assistants and chatbots for software engineers. Opinion Q&A has a stronger presence in chats than in other developer communications, thus mining them can provide a valuable resource for developers in quickly getting insight about a specific development topic (e.g., *What is the best Java library for parsing JSON?*). We address the problem of opinion Q&A extraction by developing automatic identification of opinion-asking questions and extraction of participants' answers from public online developer chats. We evaluate our automatic approaches on chats spanning six programming communities and two platforms. Our results show that a heuristic approach to opinion-asking questions works well (.87 precision), and a deep learning approach customized to the software domain outperforms heuristics-based, machine-learning-based and deep learning for answer extraction in community question answering.**

***Index Terms*—opinion question-answering system, public chats, opinion-asking question, answer extraction**

I. Introduction

Recognizing the increasing capabilities of virtual assistants that use conversational artificial intelligence (AI) (e.g., chatbots, voice assistants), some researchers in software engineering are working towards the development of virtual assistants to help programmers. They have conducted studies to gain insights into the design of a programmer conversational agent [1], proposed techniques to automatically detect speech acts in conversations about bug repair to aid the assistant in mimicking different conversation types [2], and designed virtual assistants for API usage [3].

While early versions of conversational assistants were focused on short, task-oriented dialogs (e.g., playing music or asking for facts), more sophisticated virtual assistants deliver coherent and engaging interactions by understanding dialog nuances such as user intent (e.g., asking for opinion vs knowledge) [4]. They integrate specialized instances dedicated to a single task, including dialog management, knowledge retrieval, opinion-mining, and question-answering [5]. To build virtual assistants for software engineers, we need to provide similar specialized instances based on the available information from software engineers' daily conversations. Recent studies indicate that online chat services such as IRC, Slack, and Gitter are increasingly popular platforms for software engineering conversations, including both factual and opinion information sharing and now playing a significant role in software development activities [6]–[9]. These conversations potentially provide rich data for building virtual assistants for software engineers, but little research has explored this potential.

In this paper, we leverage the availability of opinion-asking questions in developer chat platforms to explore the feasibility of building opinion-providing virtual assistants for software engineers. Opinion question answering (Opinion QA) systems [10]–[13] aim to find answers to subjective questions from user-generated content, such as online forums, product reviews, and discussion groups. One type of virtual assistant that can benefit from opinions are Conversational Search Assistants (CSAs) [24]. CSAs support information seekers who struggle forming good queries for exploratory search, e.g., seeking opinions/recommendations on API, tools, or resources, by eliciting the actual need from the user through conversation. Studies indicate developers conducting web searches or querying Q&A sites for relevant questions often find it difficult to formulate good queries [25], [26]. Wizard of Oz studies have explicitly shown the need for opinions within CSAs [27]. A key result of our paper is the availability of opinions on chat platforms, which would enable the creation of a sizable opinion Q&A corpus that could actually be used by CSAs. The opinion Q&A corpus generated from chats by our technique can be used in a few different ways to build a CSA: 1) matching queries/questions asked to the CSA with questions from the corpus and retrieving the answers; 2) summarizing related groups of opinion Q&A to generate (e.g., using a GAN) an aggregate response for a specific software engineering topic.

Opinion extraction efforts in software engineering have focused on API-related opinions and developer emotions from Q&A forums [14]–[18], developer sentiments from commit logs [19], developer intentions from emails and issue reports [20], [21] and detecting software requirements and feature requests from app reviews [22], [23]. These studies suggest that, beyond reducing developers' effort of manual searches on the web and facilitating information gathering, mining of opinions could help in increasing developer productivity, improving code efficiency [16], and building better recommendation systems [17].

Findings from our previous exploratory study [9] of Slack conversations suggests that developer chats include opinion expression during human conversations. Our current study (in Section II) of 400 developer chat conversations selected from six programming communities showed that 81 (20%) of the chat conversations start with a question that asks for opinions (e.g., "Which one is the best ORM that is efficient for large datasets?", "What do you think about the Onyx platform?"). This finding shows much higher prevalence of questions asking for opinions in chats than the 1.6% found in emails [20] and 1.1% found in issue reports [21].

Thus, we investigate the problem of opinion Q&A extraction from public developer chats in this paper. We decompose the problem of opinion Q&A extraction into two subproblems: (1) identifying questions where the questioner asks for opinions from other chat participants (which we call posing an opinion-asking question), and (2) extracting answers to those opinion-asking questions within the containing conversation.

Researchers extracting opinions from software-related documents have focused on identifying sentences containing opinions, using lexical patterns [20] and sentiment analysis techniques [16], [17], [28]. However, these techniques are not directly applicable to identifying opinion-asking questions in chats for several reasons. Chat communities differ in format, with no formal structure and informal conversation style. The natural language text in chats could follow different syntactic patterns and contain incomplete sentences [9], which could potentially inhibit automatic mining of opinions.

Outside the software engineering domain, researchers have addressed the problem of answer extraction from community question-answering (CQA) forums by using deep neural network models [29]–[31], and syntactic tree structures [32], [33]. Compared to CQA forums, chats contain rapid exchanges of messages between two or more developers in short bursts [9]. A question asked at the start of a conversation may be followed by a series of clarification or follow-up questions and their answers, before the answers to the original question are given. Moreover, along with the answers, conversations sometimes contain noisy and unrelated information. Therefore, to determine the semantic relation between a question and answer, understanding the context of discussion is crucial.

We are the first to extract opinion Q&A from developer chats, which could be used to support SE virtual assistants as well as chatbots, programmer API recommendation, automatic FAQ generation, and in understanding developer behavior and collaboration. Our automatic opinion Q&A extraction takes a chat conversation as input and automatically identifies whether the conversation starts with an opinion-asking question, and if so, extracts one or more opinion answers from the conversation. The major contributions of this paper are:

- For opinion-asking question identification, we designed a set of heuristics, learned from the results from our preliminary chat analysis, to determine if the leading question in a chat conversation asks for opinions.
- For automatic answer extraction, we built upon related work on non-SE artifacts to create a deep learning approach customized to the SE domain. We compare against heuristics, machine learning combining features, and a deep learning technique based on the context of the discussion in community question answering. This answer extraction model could potentially be leveraged to extract answers from other types of questions in chats.
- We evaluated our techniques on developer conversations from six different programming communities on two different platforms, Slack and IRC. Our evaluation results show that we can automatically identify opinion-asking questions and extract their corresponding answers within a chat conversation with a precision of 0.87 and 0.77, respectively.
- We publish the dataset and source code [1] to facilitate the replication of our study and its application in other contexts.

II. Opinion-asking questions in developer online communications

Since developer chats constitute a subclass of developer online communications, we began by investigating whether we could gain insights from work by others on analyzing the opinion-asking questions in other kinds of developer online discussions (emails, issue reports, Q&A forums).

Emails. The most closely related work, by Di Sorbo et al. [20], proposed an approach to classify email sentences according to developers' intentions (feature request, opinion asking, problem discovery, solution proposal, information seeking and information giving). Their taxonomy of intentions and associated linguistic patterns have also been applied to analyze user feedback in app reviews [34], [35].

In their taxonomy, Di Sorbo et al. define "opinion asking" as: *requiring someone to explicitly express his/her point of view about something (e.g., What do you think about creating a single webpage for all the services?.* They claim that sentences belonging to "opinion asking" may emphasize discussion elements useful for developers' activities; and thus, make it reasonable to distinguish them from more general information requests such as "information seeking". Of their manually labelled 1077 sentences from mailing lists of Qt and Ubuntu, only 17 sentences (1.6%) were classified as "opinion asking", suggesting that opinion-asking questions are infrequent in developer emails.

Issue Reports. To investigate the comprehensiveness and generalizability of Di Sorbo et al.'s taxonomy, Huang et al. [21] manually labelled 1000 sentences from issue reports of four projects (TensorFlow, Docker, Bootstrap, VS Code) in GitHub. Consistent with Di Sorbo et al.'s findings, Huang et al. [21] reported that only 11 (1.1%) sentences were classified as "opinion asking". Given this low percentage and that

[1] Replication Package: https://tinyurl.com/y3qth6s3

Table I: Example of Opinion-asking Question And Answers on Slack #python-dev channel

Ques: hello everyone, I've requirement where dataset is large like 10 millions records, i want to use django rest framework in order to provide that data. Question: which one is the best ORM which is efficient for large datasets? 1. Django ORM 2. SQL alchemy

Ans 1: SQLalchemy is more performant especially if you're using Core.
Ans 2: yea, you can mix sqlalchemy and django orm. it's all just python at the end of the day. however, if you swap out one for the other you lose all the benefits of the django orm and how it integrates with the framework.
Ans 3: If performance is a factor than use SQLAlchemy core to work on this large data set If deadlines are more of a factor that use Django ORM since you're already use DRF. Just make sure to use eager loading on relationships where you can to optimize queries.

"opinion asking" could be a sub-category of "information seeking", they merged these two categories in their study. To broaden their search of opinions, Huang et al. introduced a new category "aspect evaluation", defined as: *express opinions or evaluations on a specific aspect (e.g., "I think BS3's new theme looks good, it's a little flat style.", "But I think it's cleaner than my old test, and I prefer a non-JS solution personally.?).*" They classified 14-20% sentences as "aspect evaluation". Comparing the two definitions and their results, it is evident that although opinions are expressed widely in issue reports, questions asking for others' opinions are rare.

Chats. Chatterjee et al.'s [9] results showing potentially more opinions in Slack developer chats motivated us to perform a manual study to systematically analyze the occurrence of opinion-asking questions and their answers in developer chats.

Dataset: To create a representative analysis dataset, we identified chat groups that primarily discuss software development topics and have a substantial number of participants. We selected three programming communities with an active presence on Slack. Within those selected communities, we focused on public channels that follow a Q&A format, i.e., a conversation typically starts with a question and is followed by a discussion potentially containing multiple answers or no answers. Our analysis dataset of 400 Slack developer conversations consists of 100 conversations from Slack Pythondev#help channel, 100 from clojurians#clojure, 100 from elmlang#beginners, and 100 from elmlang#general, all chosen randomly from the dataset released by Chatterjee et al. [36].

Procedure: Using the definition of opinion-asking sentences proposed by Di Sorbo et al. [20], two annotators (authors of this paper) independently identified conversations starting with an opinion-asking question. We also investigated if those questions were answered by others in a conversation. The authors annotated a shared set of 400 conversations, which indicates that the sample size is sufficient to compute the agreement measure with high confidence [37]. We computed Cohen's Kappa inter-rater agreement between the 2 annotators, and found an agreement of 0.74, which is considered to be sufficient (> 0.6) [38]. The annotators further discussed their annotations iteratively until all disagreements were resolved.

Observations: We observed that out of our 400 developer conversations, 81 conversations (20%) start with an opinion-asking question. There are a total of 134 answers to those 81 opinion-asking questions, since each conversation could contain no or multiple answers.

Table I shows an opinion-asking question (*Ques*) and its answers (*Ans*) extracted from a conversation on #python-dev channel. Each of the answers contain sufficient information as a standalone response, and thus could be paired with the question to form separate Q&A pairs. Given that conversations are composed of a sequence of utterances by each of the people participating in the conversation in a back and forth manner, the Q&A pairs are pairs of utterances.

Summary: Compared to other developer communications, conversations starting with opinion-asking questions in developer chats are much more frequent. Thus, chats may serve as a better resource to mine for opinion Q&A systems.

III. Automatically Extracting Opinion Q&A from Developer Chats

Figure 1 describes the overview of our approach, ChatEO, to automatically *E*xtract *O*pinion Q&A from software developer *Chat*s. Our approach takes a developer chat history as input and extracts opinion Q&A pairs using the three major steps: (1) Individual conversations are extracted from the interleaved chat history using conversation disentanglement. (2) Conversations starting with an opinion-asking question are identified by applying textual heuristics. (3) Possibly multiple available answers to the opinion-asking question within the conversation are identified using a deep learning-based approach.

A. Conversation Disentanglement

Since utterances in chats form a stream, conversations often interleave such that a single conversation thread is entangled with other conversations. Hence, to ease individual conversation analysis, we separate, or disentangle, the conversation threads in each chat log. The disentanglement problem has been widely addressed by researchers in the context of IRC and similar chat platforms [9], [36], [39]–[41]. We used the best available disentanglement approaches proposed for Slack and IRC chat logs, respectively, in this paper.

- Slack chat logs: We used a subset of the publicly available disentangled Slack chat dataset[2] released by Chatterjee et al. [36] since their modified disentanglement algorithm customized for Slack developer chats achieved a micro-averaged F-measure of 0.80.
- IRC chat logs: We used Kummerfeld et al.'s [41] technique, a feed-forward neural network model for conversation disentanglement, trained on 77K manually annotated IRC utterances, and achieving 74.9% precision and 79.7% recall.

In the disentangled conversations, each utterance contains a unique conversation id and metadata including timestamp and author information.

[2]https://www.zenodo.org/record/3627124

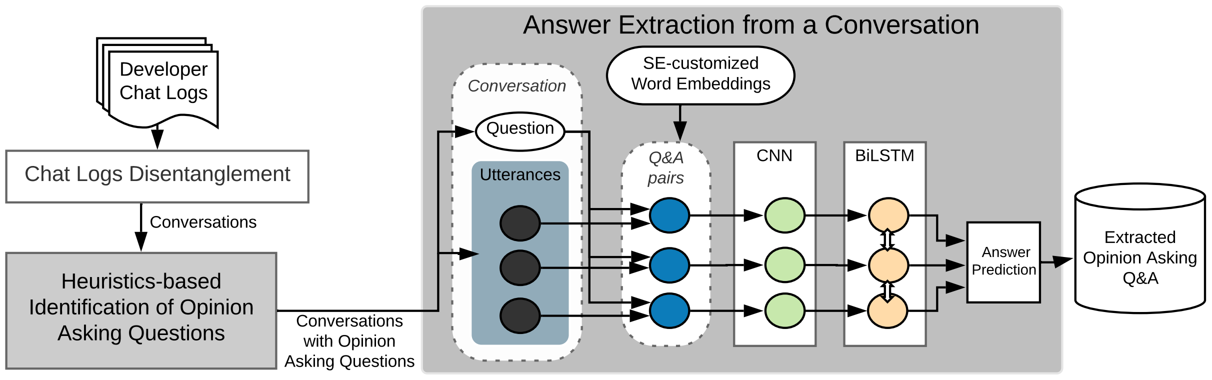
Developer Chat Logs
Chat Logs Disentanglement
Conversations
Heuristics-based Identification of Opinion Asking Questions
Conversations with Opinion Asking Questions
Answer Extraction from a Conversation
Conversation
Question
Utterances
SE-customized Word Embeddings
Q&A pairs
CNN
BiLSTM
Answer Prediction
Extracted Opinion Asking Q&A

Figure 1: Overview of ChatEO Automatic Extraction of Opinion Q&A from Developer Chats

B. Heuristics-based Identification of Opinion-asking Question

Di Sorbo et al. [20] claim that developers tend to use recurrent linguistic patterns within discussions about development issues. Thus, using natural language parsing, they defined five linguistic patterns to identify opinion-asking questions in developer emails. First, to investigate the generalizability of Di Sorbo et al.'s linguistic patterns, we used their replication package of DECA[3] to identify opinion-asking questions in developer chats. We found that, out of 400 conversations in our manual analysis dataset, only 2 questions were identified as opinion-asking questions by DECA. Hence, we conducted a deeper analysis of opinion-asking questions in our manual dataset in Section II, to identify additional linguistic patterns that represent opinion-asking questions in developer chats.

We followed a qualitative content analysis procedure [42], where the same two annotators (authors of this paper) first independently analyzed 100 developer conversations to understand the structure and characteristics of opinion-asking questions in chats. The utterances of the first speaker in each conversation were manually analyzed to categorize if they were asking for an opinion. When an opinion-asking question was manually identified, the annotator identified the parts of the utterance that contributed to that decision and identified part-of-speech tags and recurrent keywords towards potential linguistic patterns. Consider the question in Table I. First, the annotator selects the text that represents an opinion-asking question, in this case: "which one is the best ORM which is efficient for large datasets?". 'ORM" is noted as a noun referring to the library, and "best" as an adjective related to the opinion about ORM. Thus, a proposed linguistic pattern to consider is: "*Which one* <*verb_to_be*> [*positive_adjective*] [*rec_target_noun*] <*verb_phrase*>?".

Throughout the annotation process, the annotators wrote memos to facilitate the analysis, recording observations on types of opinions asked, observed linguistic patterns of opinion-asking questions, and researcher reflections. The two annotators then met and discussed their observations on common types of questions asking for opinions, which resulted in a set of preliminary patterns for identifying opinion-asking questions in developer chats. Using the preliminary patterns, the two authors then independently coded the rest of the opinion-asking questions in our manual analysis dataset, after which they met to further analyze their annotations and discuss disagreements. Thus, the analysis was performed in an iterative approach comprised of multiple sessions, which helped in generalizing the hypotheses and revising the linguistic patterns of opinion-asking questions.

[3]https://www.ifi.uzh.ch/en/seal/people/panichella/tools/DECA.html

Table II: Most Common Pattern for opinion-asking Question Identification in Chats.

Pattern code: P_ANY_ADJ
Description: question starting with "any" and followed by a positive adjective and target noun.
Rule: Any [rec_positive_adj] [rec_target_noun] verb_phrase
Definitions:
[rec_positive_adj] = "good","better","best","right","optimal",...
[rec_target_noun] = "project","api","tutorial","library",...
Example: *Any good examples or things I need to be looking at?*

Our manual analysis findings from 400 Slack conversations showed that an opinion-asking question in a developer chat is a question occurring primarily at the beginning of a conversation and could exhibit any of these characteristics:

- Expects subjective answers (i.e., opinions) about APIs, libraries, examples, resources, e.g., "Is this a bad style?","What do you think?"
- Asks for which path to take among several paths, e.g., "Should I use X instead of Y?"
- Asks for an alternative solution (other than questioner's current solution), e.g., "Is there a better way?"

Thus, we extended Di Sorbo et al.'s linguistic pattern set for identifying opinion-asking questions, by adding 10 additional linguistic patterns. Table II shows the most common pattern, P_ANY_ADJ, in our dataset with its description and example question. Most of the patterns utilize a combination of keywords and part-of-speech-tagging. The annotators curated sets of keywords in several categories, e.g., [rec_target_noun], [rec_verbs], [rec_positive_adjective] related to nouns, verbs, and adjectives, respectively. The complete set of patterns and keywords list are available in our replication package.

C. Answer Selection from a Conversation

We build upon work by Zhou et al. [29], who designed R-CNN, a deep learning architecture, for answer selection in community question answering (CQA). Since R-CNN was designed for application in CQA for the non-SE domain[4], we customize for application in chats for software engineering and then compare to the non-customized R-CNN in our evaluation. We chose to build on R-CNN because other answer extraction models [43]–[45] only model the semantic relevance between questions and answers. In contrast, R-CNN models the semantic links between successive candidate answers in a discussion thread, in addition to the semantic relevance between question and answer. Since developer chats often contain short and rapid exchanges of messages between participants, understanding the context of the discussion is crucial to determine the semantic relation between question and answer. Hence, we adapt R-CNN to extract the relevant answer(s) to an opinion-asking question based on the context of the discussion in a conversation.

Zhou et al. [29] regarded the problem of answer selection as an answer sequence labeling task. First, they apply two convolution neural networks (CNNs) to summarize the meaning of the question and a candidate answer, and then generate the joint representation of a Q&A pair. The learned joint representation is then used as input to long short-term memory (LSTM) to learn the answer sequence of a question for labeling the matching quality of each answer.

To design ChatEO, we make the following adaptations to account for both the SE domain content and specifically software-related chats. First, we preprocess the text, apply a software-specific word-embedding model, and use those embeddings as input to a CNN to learn joint representation of a Q&A pair. We use TextCNN [46] since text in chat (utterances) are much shorter compared to CQA (post). The representations from the CNN are then passed as input to Bidirectional LSTM (BiLSTM) instead of LSTM to improve prediction of the answers from a sequence of utterances in a conversation. We detail ChatEO answer extraction as follows:

1) Preprocessing: To help ChatEO with the semantics of the chat text, the textual content in the disentangled conversations is preprocessed. We replace url, user mentions, emojis, and code with specific tokens *'url', 'username', 'emoji'*, and *'code'* respectively. To handle the informal style of communication in chats, we use a manual set of common phrase expansions (e.g., "you've" to "you have"). We then convert the text to lowercase.

2) SE-customized Word Embeddings: Text in developer chats and other software development communication can differ from regular English text found in Wikipedia, news articles, etc. in terms of vocabulary and semantics. Hence, we trained custom GloVe vectors on the most recent Stack Overflow data dump (as of June, 2020) to more precisely capture word semantics in the context of developer communications. To train GloVe vectors, we performed standard tokenization and preprocessing on each Stack Overflow post's title and text and trimmed extremely rarely occurring words (vocabulary minimum threshold of 100 posts; window size of 15 words). Our word embedding model thus consists of 123,995 words, where each word is represented by a 200-dimensional word vector. We applied this custom word embedding model to each word in each utterance of a conversation.

3) Convolutional Neural Networks: In natural language analysis tasks, a sentence is encoded before it is further processed in neural networks. We leverage the sentence encoding technique from TextCNN [46]. TextCNN, also used for other dialog analysis tasks [47], is a classical technique for sentence modeling which uses a shallow Convolution Neural Network (CNN) that combines n-gram information to model compact text representation. Since an utterance in a chat is typically short (<25 words on average), we take each utterance as a sentence and apply word embedding, multiple convolution, and max-pooling operations.

The input for TextCNN is the distributed representation of an utterance, created by mapping each word index into its pre-trained embeddings. Each utterance is padded to the same length n with zero vectors. Let $z_j \in \mathbb{R}^d$ denote the d-dimensional embedding for the jth word in an utterance. Thus, an utterance of length n can be represented by: $z_{1:n} = z_1 \oplus z_2 \oplus \ldots z_n$, where $\oplus$ is the concatenation operator. To gather local information, convolution is achieved by applying a fixed length sliding window (kernel) $w^m \in \mathbb{R}^{h\times d}$, on each word position i such that $n-h+1$ convolutional units in the mth layer are generated by: $c_i^m = \sigma\left(w^m \cdot z_{i:i+h} + b^m\right), i = 0, 1, \ldots, n-h+1$, where h is the size of convolution kernel, σ is the activation function, and b^m is the bias factor for the mth layer. The convolution layer is followed by a max pooling layer, which can select the most effective information with the highest value. The flattened output vectors for each kernel after max-pooling are concatenated as the final output.

4) Bidirectional LSTM: The task of identifying answers in a conversation requires capturing the context and flow of information among the utterances inside a conversation. Hence, we use Bidirectional Long Short Term Memory (BiLSTM) [48], where the utterances of a conversation are considered as sequential data. The input to our BiLSTM is a sequence of utterance representations created by TextCNN. Variations of LSTM, widely used by researchers for answer extraction tasks [30], [49], are capable of modeling semantic links between continuous text to perform answer sequence learning.

LSTM [50] uses a gate mechanism to filter relevant information and capture long-term dependencies. An LSTM cell comprises of input gate (i), forget gate (f), cell state (c), and output gate (o). The outputs of LSTM at each time step h_t can be computed by the following equations:

$$\begin{bmatrix} i_t \\ f_t \\ \tilde{c}_t \\ o_t \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \tanh \\ \sigma \end{bmatrix} \left(W \begin{bmatrix} x_t \\ h_{t-1} \end{bmatrix} + b \right)$$

$$c_t = \tilde{c}_t \odot i_t + c_{t-1} \odot f_t$$

[4]http://alt.qcri.org/semeval2015/task3/

$$h_t = o_t \odot \tanh(c_t)$$

where x_t is the ith element in the input sequence; W is the weight matrix of LSTM cells; b is the bias term; σ denotes sigmoid activation function, and $tanh$ denotes hyperbolic tangent activation function; $\odot$ denotes element-wise multiplication.

BiLSTM processes a sequence on two opposite directions (forward and backward), and generates two independent sequences of LSTM output vectors. Hence, the output of a BiLSTM at each time step is the concatenation of the two output vectors from both directions, $h = [\overrightarrow{h} \oplus \overleftarrow{h}]$, where $\overrightarrow{h}$ and $\overleftarrow{h}$ denote the outputs of two LSTM layers respectively, and $\oplus$ is the concatenation operator.

IV. Evaluation Study

We designed our evaluation to analyze the effectiveness of the pattern-based identification of opinion-asking questions (*RQ1*) and of our answer extraction technique (*RQ2*).

A. Metrics

We use measures that are widely used for evaluation in machine learning and classification. To analyze whether the automatically identified instances are indeed opinion-asking questions and their answers, we use precision, the ratio of true positives over the sum of true and false positives. To evaluate how often a technique fails to identify an opinion-asking question or its answer, we use recall, the ratio of true positives over the sum of true positives and false negatives. F-measure combines these measures by harmonic mean.

B. Evaluation Datasets

We established several requirements for dataset creation to reduce bias and threats to validity. To curate a representative analysis dataset, we identified chat groups that primarily discuss software development topics and had a substantial number of participants. To ensure the generalizability of our techniques, we chose two separate chat platforms, Slack and IRC, which are currently the most popular chat platforms used by software developers. We selected six popular programming communities with an active presence on Slack or IRC. We believe the communities are representative of public software-related chats in general; we observed that the structure and intent of conversations are similar across all 6 communities.

To collect conversations on Slack, we downloaded the developer chat conversations dataset released by Chatterjee et al. [36]. To gather conversations on IRC, we scraped publicly available online chat logs[5]. After disentanglement, we discarded single-utterance conversations, and then created two separate evaluation datasets, one for opinion-asking question identification and a subset with only chats that start with an opinion-asking question, for answer extraction. We created our evaluation datasets by randomly selecting a representative portion of the conversations from each of the six programming communities.

[5]https://echelog.com/

Table III shows the characteristics of the collected chat logs, and our evaluation datasets, where *#OAConv* gives the number of conversations that we identified as starting with an opinion-asking question using the heuristics described in section III-B, per community.

The question identification evaluation dataset consists of a total of 400 conversations, 5153 utterances, and 489 users. Our question extraction technique is heuristics-based, requiring conversations that do not start with an opinion-asking question. Thus, we randomly chose 400 from our 45K chat logs for a reasonable human-constructed goldset.

The evaluation dataset for answer extraction consists of a total of 2001 conversations, 23,972 utterances, and 3160 users. Our machine-learning-based answer extraction requires conversations starting with a question, and a dataset large enough for training. Thus, 2001 conversations starting with opinion-asking questions were used.

RQ1. How effective is ChatEO in identifying opinion-asking questions in developer chats?

Gold Set Creation: We recruited 2 human judges with (3+ years) experience in programming and in using both chat platforms (Slack and IRC), but no knowledge of our techniques. They were provided a set of conversations where each utterance includes: the unique conversation id, anonymized name of the speaker, the utterance timestamp, and the utterance text. Using Di Sorbo et al.'s [20] definition of opinion-asking questions i.e., *requiring someone to explicitly express his/her point of view about something (e.g., What do you think about creating a single webpage for all the services?)*, the human judges were asked to annotate only the utterances of the first speaker of each conversation with value '1' for opinion-asking question, or otherwise '0'.

The judges annotated a shared set of 400 conversations, of which they identified 69 instances i.e., utterances of the first speaker of each conversation, containing opinion-asking questions (AngularJS: 10, C++: 10, OpenGL: 5, Python: 15, Clojurians: 12, Elm: 17). We computed Cohen's Kappa inter-rater agreement between the 2 judges, and found an agreement of 0.76, which is considered to be sufficient (> 0.6) [38], while the sample size of 400 conversations is sufficient to compute the agreement measure with high confidence [37]. The two judges then iteratively discussed and resolved all conflicts to create the final gold set.

Comparison Techniques: Researchers have used sentiment analysis techniques [16], [17] and lexical patterns [20] to extract opinions from software-related documents. Thus, we selected two different approaches, i.e., pattern-matching approaches and sentiment analysis, as comparison techniques to ChatEO. We evaluated three well-known sentiment analysis techniques SentiStrength-SE [51], CoreNLP [52], and NLTK [53] with their default settings. Since opinions could have positive/negative polarities, for the purpose of evaluation, we consider a leading question in a conversation identified with either positive or negative sentiment as opinion-asking. DECA [20] is a pattern-based technique that uses Natural Language

Table III: Evaluation Dataset

Samples(OA Question Identification) created from *Chat Logs(#Conv)*, *Samples(Answer Extraction)* created from *Chat Logs(#OAConv)*

Source	**Duration**	**Chat Logs**		**Samples**					
				OA Question Identification			**Answer Extraction**		
		#Conversations	#OAConv	#Conversations	#Utterances	#Users	#Conversations	#Utterances	#Users
angularjs (IRC)	Apr2014-Jun2020	181662	9175	80	1812	90	891	11218	1109
c++ (IRC)	Oct2018-Jun2020	95548	841	60	630	62	127	2017	181
opengl (IRC)	Jul2005-Jun2020	135536	9198	60	698	54	258	3176	390
pythondev (Slack)	Jul2017-Jun2019	8887	707	80	848	93	160	1604	320
clojurians (Slack)	Jul2017-Jun2019	7918	562	60	570	81	156	1422	314
elmlang (Slack)	Jul2017-Jun2019	22150	1665	60	595	109	409	4535	846
Total		**451701**	**22148**	**400**	**5153**	**489**	**2001**	**23972**	**3160**

Table IV: Opinion-Asking Question Identification Results

Technique	**P**	**R**	**F**	**Example FP**	**Example TP**
SentiStrength-SE [51]	0.18	0.31	0.23	*...I'm having a weird issue with my ng-cli based angular app...Is there any potential issue, because my Model is called ”Class” and/or the method is called ”toClass”?*	*Anyone know a good cross platform GUI library that (preferably) supports CMake? I'd rather not use Qt if I don't have to because I don't want to use the qt MOC*
CoreNLP [52]	0.19	0.73	0.30	*why does '(read-string ”07”)' return '7', but '(read-string ”08”)' throws an exception?*	*any suggestions for cmake test running strategies? or how to organize tests expected to succeed or fail?*
NLTK [53]	0.18	0.78	0.30	*Hi, is there a way to expose a class and a function using c style declaration : void foo(MyClass bar)*	*Does anyone know of a good way to ”sandbox” the loading of namespaces?*
DECA [20]	1.00	0.01	0.02	-	*..seems lambda with auto argument type provide separate templated methods per used type; am i right?..*
ChatEO	**0.87**	**0.49**	**0.62**	*can someone tell me why does this give an error ?*	*What is the best way in your opinion to convert input (file, XML, etc.) to PDF with precision to 1 mm?*

Parsing to classify the content of development emails according to their purpose. We used their tool to investigate the use of their linguistic patterns to identify opinion-asking questions in developer chats. We do not compare with Huang et al.'s CNN-based classifier of intention categories [21], since they merged "opinion asking" with the "information seeking" category.

Results: Table IV presents precision (P), recall (R), F-measure (F), and examples of False Positives (FP) and True Positives (TP) for ChatEO and our comparison techniques for opinion-asking question identification on our evaluation dataset.

ChatEO achieves a precision, recall, and F-measure of 0.87, 0.49 and 0.62, respectively. Results in Table IV indicate that ChatEO achieves an overall better precision (except DECA) and F-measure, compared to all the comparison techniques. With high precision, when ChatEO identifies an opinion-asking question, the chance of it being correct is higher than that identified by other techniques. We aim for higher precision (with possible lower recall) in identifying opinion-asking questions, since that could potentially contribute to the next module of ChatEO i.e., extracting answers to opinion-asking questions.

Some of the opinion-asking instances that ChatEO wasn't able to recognize lacked presence of recurrent linguistic patterns such as *“”How does angular fit in with traditional MVC frameworks like .NET MVC and ruby on rails? Do people generally still use an MVC framework or just write a web api?”*. Some FNs also resulted from instances where the opinion-asking questions were continuation of a separate question such as *“Is there a canvas library where I can use getImageData to work with the typed array data? Or is this where I should use ports?”*.

We observe that the sentiment analysis tools show a high recall at the expense of low precision, with an exception of SentiStrengthSE, which exhibits lower values for both precision and recall. The *“Example FP”* column in Table IV indicates that sentiment analysis tools are often unable to catch the nuances of SE-specific keywords such as 'expose', 'exception'. Another example, *“What is the preferred way to distribute python programs?”*, which ChatEO is able to correctly identify as opinion-asking, is labelled as neutral by all the sentiment analysis tools. The same happens for the instance *“how do I filter items in a list when displaying them with ngFor? Should I use a filter/pipe or should I use ngIf in the template?”*. ChatEO is able to recognize that this is asking for opinions on what path to take among two options, while the sentiment analysis tools classify this as neutral. Note that this just indicates that these tools are limited in the context of identifying opinion-asking questions, but could be indeed useful for other tasks (e.g., assessing developer emotions).

DECA [20] identified only one instance to be opinion-asking, which is a true positive, hence the precision is 1.00. Apart from this, it was not able to classify any other instance as opinion-asking, hence the low recall (0.01). On analyzing DECA's classification results, we observe that, out of 69 instances in the gold set, it could not assign any intention category to 17 instances. This is possibly due to the informal communication style in chats, which is considerably different than emails. Since an utterance could contain more than one sentence, DECA often assigned multiple categories (e.g., *information seeking, feature request, problem discovery*) to each instance. The most frequent intention category observed was *information seeking*. During the development phase, we explored additional linguistic patterns, but they yielded more FPs. This is a limitation of using linguistic patterns, as they are restrictive when expressing words that have different meaning in different contexts.

ChatEO opinion-asking question identification significantly outperforms an existing pattern-based technique that was designed for emails [20], as well as sentiment analysis tools [51]–[53] in terms of F-measure.

RQ2. How effective is ChatEO in identifying answer(s) to opinion-asking questions in a developer conversation?

Gold Set Creation: Similar to RQ1, we recruited 2 human judges with (3+ years) experience in programming and in using both chat platforms (Slack and IRC), but no knowledge of our techniques. The gold set creation for answer annotation was conducted in two phases, as follows:

- Phase-1 (Annotation): The human judges were provided a set of conversations with annotation instructions as follows: *Mark each utterance in the conversation that provides information or advice (good or bad) that contributes to addressing the opinion-asking question in a way that is understandable/meaningful/interpretable when read as a standalone response to the marked opinion-asking question (i.e., the answer should provide information that is understandable without reading the entire conversation). Such utterances should not represent answer(s) to follow-up questions in a conversation. An answer to an opinion-asking question could also be a yes/no response. There could be more than one answer provided to the opinion-asking question in a conversation.*
- Phase-2 (Validation): The purpose of Phase-2 was two-fold: (1) measure validity of Phase-1 annotations, and (2) evaluate if an answer would match an opinion-asking question out of conversational context, such that the Q&A pair could be useful as part of a Q&A system. Therefore, for Phase-2 annotations, we ensured that the annotators read only the provided question and answers, and not the entire conversations from which they were extracted. The Phase-1 annotations from the first annotator were used to generate a set of question and answers, which were used for Phase-2 annotations by the second annotator, and vice-versa. For each utterance provided as an answer to an opinion-asking question, the annotators were asked to indicate ("yes/no") if the utterance represents an answer based on the guidelines in Phase-1. Additionally, if the annotation value was "no", the annotators were asked to state the reason.

The judges annotated a total of 2001 conversations, of which they identified a total of 2292 answers to opinion-asking questions (AngularJS: 1001, C++: 133, OpenGL: 263, Python: 165, Clojurians: 197, Elm: 533). We found that the first annotator considered 94.6% of annotations of the second annotator as valid, while the second annotator considered 96.2% annotations of the first annotator as valid. We also noticed that the majority of disagreements were due to the answer utterances containing incomplete or inadequate information to answer the marked opinion-asking question when removed from conversational context (e.g., "and then you can replace your calls to 'f' with 'logArgs2 f' without touching the function..."), and human annotation errors such as marking an utterance as an answer when it just points to other channels.

Comparison Techniques: Since we believe this is the first effort to automatically extract answers to opinion-asking questions from developer chats, we chose to evaluate against heuristic-based and feature-based machine learning classification as well as the original R-CNN deep learning-based technique on which we built ChatEO.

Heuristic-based (HE): Intuitively, the answer to an opinion-asking question might be found based on its location in the conversation, the relation between its content and the question, or the presence of sentiment in the answer. We investigated each of these possibilities separately and in combination.

- *Location:* Based on the intuition that a question might be answered immediately after it is asked during a conversation, we compare against the naive approach of identifying the next utterance after the leading opinion-asking question as an answer.
- *Content:* Traditional Q&A systems have often aimed to extract answers based on semantic matching between question and answers [43], [54]. Thus, to model content-based semantic relation between question and answer, we compare the average word embedding of the question and answer texts. Using our word embedding model described in III-C2, we extract utterances with considerable similarity ($\geq$ 0.5) to the opinion-asking question as answers.
- *Sentiment:* Previous researchers have leveraged sentiment analysis to extract relevant answers in non-factoid Q&A systems [55]–[58]. Thus, based on the intuition that the answer to an opinion-asking question might exhibit sentiment, we use CoreNLP [52] to extract utterances bearing sentiment (positive or negative) as answers. We explored other sentiment analysis tools (e.g., SentiStrength-SE [51], NLTK [53]); however, we do not discuss them, since they yielded inferior results.

Machine Learning-based (ML): We combine location, content, sentiment attributes as features of a machine learning (ML)-based classifier. We explored several popular ML algorithms (e.g., Support Vector Machines (SVM), Random Forest) using the Weka toolkit [59], and observed that they yielded nearly similar results. We report the results for SVM.

Deep Learning-based (DL): We present the results for both R-CNN and ChatEO implemented as follows.

- *RCNN:* We implemented R-CNN [29] for developer chats using open-source neural-network library Keras [60]. R-CNN used word embeddings pre-trained on their corpus. Similarly, we trained custom GloVe vectors on our chat corpus for our comparison.
- *ChatEO:* We also implemented ChatEO using Keras [60]. We used grid-search [61] to perform hyper-parameter tuning. First, to obtain sufficient semantic information at the utterance level, we use three convolution filters of size 2, 3, and 4, with 50 (twice the average length of an utterance) feature maps for each filter. The pool sizes of convolution are (2,1), (2,1), (3,1), respectively. Then, a BiLSTM layer with 400 units (200 for each direction) is used to capture the contextual information in a conversation. Finally, we use a linear layer with sigmoid activation function to predict the probability scores of binary classes (answer and non-answer). We use binary-cross-entropy as the loss function, and Adam optimization algorithm for gradient descent.
  To avoid over-fitting, we apply a dropout [62] of 0.5 on the TextCNN embeddings, i.e., 50% units will be randomly omitted to prevent complex co-adaptations on the training

Table V: Answer Extraction Results on Held-out Test Set
HE: Heuristic-based, ML: Machine Learning-based, DL: Deep Learning-based

Technique		**AngularJS**			**C++**			**OpenGL**			**Python**			**Clojurians**			**Elm**			**Overall**		
		P	R	F	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
HE	Location(L)	0.66	0.47	0.55	0.68	**0.77**	0.72	0.69	**0.74**	0.71	0.59	0.62	0.60	0.52	0.40	0.46	0.65	0.45	0.53	0.63	0.55	0.59
	Content(C)	0.03	0.06	0.04	0.05	0.14	0.07	0.06	0.18	0.09	0.11	0.18	0.14	0.09	0.11	0.10	0.14	0.15	0.14	0.07	0.13	0.09
	Sentiment(S)	0.06	0.16	0.09	0.05	0.23	0.08	0.09	0.31	0.13	0.11	0.24	0.15	0.07	0.14	0.09	0.10	0.18	0.13	0.07	0.20	0.11
ML	L+C+S	0.70	0.47	0.56	0.78	**0.77**	**0.77**	0.70	0.73	0.71	0.62	0.62	0.62	0.60	0.41	0.49	0.75	0.47	0.58	0.69	0.55	0.61
DL	R-CNN	0.70	0.46	0.55	0.78	0.74	0.76	0.72	0.70	0.71	0.64	0.61	0.62	0.62	0.38	0.48	0.73	0.39	0.51	0.70	0.52	0.60
	ChatEO	**0.80**	**0.53**	**0.64**	**0.81**	0.72	**0.77**	**0.77**	0.68	**0.72**	**0.76**	**0.65**	**0.70**	**0.66**	**0.42**	**0.51**	**0.81**	**0.56**	**0.66**	**0.77**	**0.59**	**0.67**

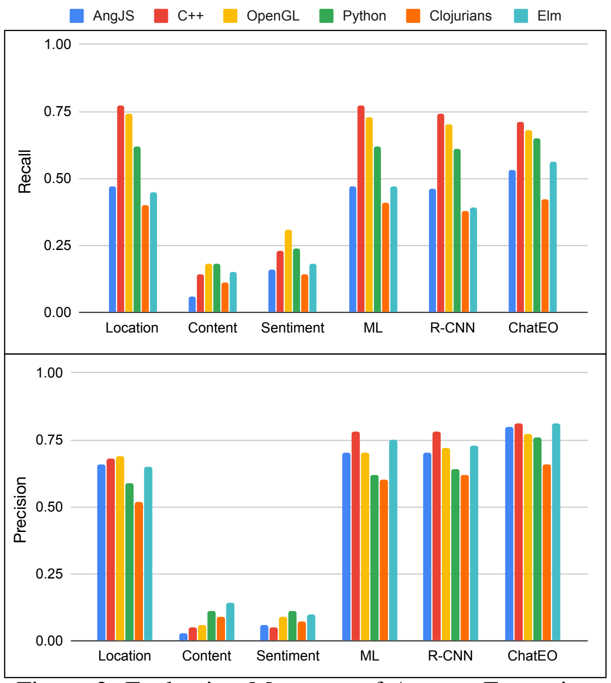

Figure 2: Evaluation Measures of Answer Extraction

data. Additionally, we use a recurrent dropout of 0.1 in the LSTM units. We also use early stopping [63], i.e., if the performance of the classifier did not improve for 10 epochs, the training process is stopped. ChatEO answer extraction takes approximately 36 minutes to train and test 2001 conversations on a system with 2.5 GHz Intel Core i5 processor and 8GB DDR3 RAM.

Evaluation Process: Evaluation of ChatEO and the comparison techniques was conducted using the evaluation dataset of 2000 conversations, described in Table III. We created a test set of 400 conversations (adhering to commonly recognized train-test ratio of 80-20%). We ensured that it contains similar number of instances from each programming community: 140 (70*2) conversations from Angular JS and Python, 260 (65*4) conversations from C++, OpenGL, Clojurians, and Elm.

Results: Table V presents precision (P), recall (R), and F-measure (F) for ChatEO and our comparison techniques for automatic answer extraction on our held-out test set of 400 conversations, which contains a total of 499 answers. The best results for P, R, F across all techniques are highlighted in bold.

Overall, Table V shows that ChatEO achieves the highest overall precision, recall, and F-Measure of all techniques, with 0.77, 0.59, and 0.67, respectively. Overall, ChatEO identified 370 answers in the test set, out of which 285 are true positives. R-CNN and the ML-based classifier perform next best and similar to each other in precision and F-measure. The better performance of ChatEO suggests that capturing contextual information through BiLSTM can benefit answer extraction in chat messages, and that using a domain-specific word embedding model (trained on software-related texts) accompanied with hyper-parameter tuning, is essential to adapting deep learning models for software-related applications. Figure 2 shows that the performance of ChatEO is consistent (76-81% precision) across all communities, except Clojure. One possible reason for this is, answers are often provided in this chat community in the form of code snippets along with little to no natural language text, which makes it difficult for ChatEO to model the semantic links.

ChatEO's overall recall of 0.59 indicates that it is difficult to identify all relevant answers in chats even with complex dialog modeling. In fact, the recall of ChatEO is lower than the heuristic-based technique *location* for C++ and OpenGL. Upon deeper analysis, we observed that these two communities contain less answers (one answer per opinion-asking question on average) compared to the other communities, and the answer often resides in the utterance occurring immediately after the first speaker.

The *location* heuristic exhibits significantly better performance than content or sentiment heuristics. Of the 400 conversations, 278 have at least one answer occurring immediately after the utterances of the first speaker. Neither *content* or *sentiment* is a strong indicator of answers, with precision of 0.07 and F-measure of 0.09 and 0.11, respectively. These heuristics cannot distinguish the intent of a response. Consider, "***Q:*** *Hi, Clojure intermediate here. What is the best way to read big endian binary file?;* ***R:*** *do you need to make something that parses the binary contents, does it suffice to just get the bytes in an array?*". Both *content* and *sentiment* marked this response as an answer, without being able to understand that this is a follow-up question.

Combining the heuristics as features to SVM, the precision (and thus F-Measure) improved slightly over the *location* heuristic with .06 increase in precision and .02 in F-measure. As expected, *location* as a feature shows the highest information gain. We investigated several classifier parameters (e.g., kernal and regularization parameters), but observed that the classification task was not very sensitive to parameter choices, as they had little discernible effect on the effectiveness metrics (in most cases ≤ 0.01). Since our dataset is imbalanced, with considerably low ratio of *answers* to other utterances, we explored over-sampling (SMOTE) techniques. No significant improvements occurred. ML-based classification may be improved with more features and feature engineering.

ChatEO answer extraction shows improvement over heuristics-based, ML-based, and existing deep learning-based [29] techniques in terms of precision, recall, and F-measure.

V. Threats to Validity

Construct Validity: A threat to construct validity might arise from the manual annotations for creating the gold sets. To limit this threat, we ensured that our annotators have considerable experience in programming and in using both chat platforms and that they followed a consistent annotation procedure piloted in advance. We also observed high values of Cohen's Kappa coefficient, which measures the inter-rater agreement for opinion-asking questions. For answer annotations, we conducted a two-phase annotation procedure to ensure the validity of the selected answers.

Internal Validity: Errors in the automatically disentangled conversations could pose a threat to internal validity affecting misclassification. We mitigated this threat by humans, without knowledge of our techniques, manually discarding poorly disentangled conversations from our dataset. In all stages of the pipeline of ChatEO, we aimed for higher precision over recall as the quality of information is more important than missing instances; chat datasets are large with many opinions so our achieved recall is sufficient to extract a significant number of opinion Q&A. Other potential threats could be related to evaluation bias or errors in our scripts. To reduce these threats, we ensured that the instances in our development set do not overlap with our train or test sets. We also wrote unit tests and performed code reviews.

External Validity: To ensure generalizability of our approach, we selected the subjects of our study from the two most popular software developer chat communities, Slack and IRC. We selected statistically representative samples from six active communities which represent a broad set of topics related to each programming language. However, our study's results may not transfer to other chat platforms or developer communications. Scaling to larger datasets might also lead to different evaluation results. Our technique of identifying opinion-asking questions could be made more generalizable by augmenting the set of identified patterns and vocabulary terms.

VI. Related Work

Mining Opinions in SE. In addition to the related work discussed in Section II, significant work has focused on mining opinions from developer forums. Uddin and Khomh [16] designed Opiner, which uses keyword-matching along with a customized Sentiment Orientation algorithm to summarize API reviews. Lin et al. [17] used patterns to identify and classify opinions on APIs from Stack Overflow. Zhang et al. [64] identifies negative opinions about APIs from forums. Huang et al. [65] proposed an automatic approach to distill and aggregate comparative opinions of comparable technologies from Q&A websites. Ren et al. [66] discovered and summarized controversial (criticized) answers in Stack Overflow, based on judgment, sentiment and opinion. Novielli et al. [18], [67] investigated the role of affective lexicon on the questions posted in Stack Overflow.

Researchers have also analyzed opinions in developer emails, commit logs, and app reviews. Xiong et al. [68] studied assumptions in OSS development mailing lists. Sinha et al. [19] analyzed developer sentiment in Github commit logs. Opinions in app reviews [22], [23], [34], [69] have been mined to help app developers gather information about user requirements, ideas for improvements, and user sentiments about specific features. To the best of our knowledge, our work is the first to extract opinion Q&A from developer chats.

Extracting Q&A from Online Communications. Outside the SE domain, researchers have proposed techniques to identify Q&A pairs in online communications (e.g., Yahoo Answers). Shrestha et al. [70] used machine learning approaches to automatically detect Q&A pairs in emails. Cong et al. [71] detected Q&A pairs from forum threads by using Sequential Pattern Mining to detect questions, and a graph-based propagation method to detect answers in the same thread.

Recently, researchers have focused on answer selection, a major subtask of Q&A extraction, which aims to select the most relevant answers from a candidate answer set. Typical approaches for answer selection model the semantic matching between question and answers [31], [43]–[45]. These approaches have the advantage of sharing parameters, thus making the model smaller and easier to train. However, they often fail to capture the semantic correlations embedded in the response sequence of a question. To overcome such drawbacks, Zhou et al. [29] designed a recurrent architecture that models the semantic relations between successive responses, as well as the question and answer. Xiang et al. [49] investigated an attention mechanism and context modeling to aid the learning of deterministic information for answer selection. Wang et al. [30] proposed a bilateral multi-perspective matching model in which Q&A pairs are matched on multiple levels of granularity at each time-step. Our model belongs to the same framework which captures the contextual information of conversations in extracting answers from developer chats.

Most of the these techniques for Q&A extraction were designed for general online communications and not specifically for software forums. Gottipati et al. [72] used Hidden Markov Models to infer semantic tags (e.g., question, answer, clarifying question) of posts in the software forum threads. Henß et al. [73] used topic modeling and text similarity measures to automatically extract FAQs from software development discussions (mailing lists, online forums).

Analyzing Developer Chats. Wood et al. [2] created a supervised classifier to automatically detect speech acts in developer Q&A bug repair conversations. Shi et al. [47] use deep Siamese network to identify feature-request from chat conversations. Alkadhi et al. [74], [75] showed that machine learning can be leveraged to detect rationale in IRC messages. Chowdhury and Hindle [76] exploit Stack Overflow discussions and YouTube video comments to automatically filter off-topic discussions in IRC chats. Romero et al. [8] developed a chatbot that detects a troubleshooting question asked on Gitter and provides possible answers retrieved from querying similar Stack Overflow posts. Compared to their work, we are automatically identifying opinion-asking questions and their answers provided by developers in chat forums.

Chatterjee et al.'s [9] exploratory study on Slack developer chats suggested that developers share opinions and interesting insights on APIs, programming tools and best practices, via conversations. Other studies have focused on learning developer behaviors and how chat communities are used by development teams across the globe [6], [7], [77]–[81].

VII. Conclusions and Future Work

In this paper, we present and evaluate ChatEO, which automatically identifies opinion-asking questions from chats using a pattern-based technique, and extracts participants' answers using a deep learning-based architecture. This research provides a significant contribution to using software developers' public chat forums for building opinion Q&A systems, a specialized instance of virtual assistants and chatbots for software engineers. ChatEO opinion-asking question identification significantly outperforms existing sentiment analysis tools and a pattern-based technique that was designed for emails [20]. ChatEO answer extraction shows improvement over heuristics-based, ML-based, and an existing deep learning-based technique designed for CQA [29]. Our replication package can be used to verify our results [url].

Our immediate next steps focus on investigating machine learning-based techniques for opinion-asking question identification, and attention-based LSTM network for answer extraction. We will also expand to a larger and more diverse developer chat communication dataset. The Q&A pairs extracted using ChatEO could also be leveraged to generate FAQs, provide tool support for recommendation systems, and in understanding developer behavior and collaboration (asking and sharing opinions).

Acknowledgment

We acknowledge the support of the National Science Foundation under grants 1812968 and 1813253.

References

[1] S. K. Kuttal, J. Myers, S. Gurka, D. Magar, D. Piorkowski, and R. Bellamy, "Towards designing conversational agents for pair programming: Accounting for creativity strategies and conversational styles," in *2020 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 2020, pp. 1–11.

[2] A. Wood, P. Rodeghero, A. Armaly, and C. McMillan, "Detecting speech act types in developer question/answer conversations during bug repair," in *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE 2018. New York, NY, USA: Association for Computing Machinery, 2018, p. 491?502. [Online]. Available: https://doi.org/10.1145/3236024.3236031

[3] Z. Eberhart, A. Bansal, and C. McMillan, "The apiza corpus: Api usage dialogues with a simulated virtual assistant," 2020.

[4] A. Ram, R. Prasad, C. Khatri, A. Venkatesh, R. Gabriel, Q. Liu, J. Nunn, B. Hedayatnia, M. Cheng, A. Nagar, E. King, K. Bland, A. Wartick, Y. Pan, H. Song, S. Jayadevan, G. Hwang, and A. Pettigrue, "Conversational ai: The science behind the alexa prize," 2018.

[5] I. Itkin, A. Novikov, and R. Yavorskiy, "Development of intelligent virtual assistant for software testing team," in *2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, 2019, pp. 126–129.

[6] B. Lin, A. Zagalsky, M.-A. Storey, and A. Serebrenik, "Why Developers Are Slacking Off: Understanding How Software Teams Use Slack," in *Proceedings of the 19th ACM Conference on Computer Supported Cooperative Work and Social Computing Companion*, ser. CSCW '16 Companion. New York, NY, USA: ACM, 2016, pp. 333–336. [Online]. Available: http://doi.acm.org/10.1145/2818052.2869117

[7] E. Shihab, Z. M. Jiang, and A. E. Hassan, "On the Use of Internet Relay Chat (IRC) Meetings by Developers of the GNOME GTK+ Project," in *Proceedings of the 2009 6th IEEE International Working Conference on Mining Software Repositories*, ser. MSR '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 107–110. [Online]. Available: http://dx.doi.org/10.1109/MSR.2009.5069488

[8] R. Romero, S. Haiduc, and E. Parra, "Experiences building an answer bot for gitter," in *Proceedings of the 2nd International Workshop on Bots in Software Engineering*, ser. BotSE '20, 2020.

[9] P. Chatterjee, K. Damevski, L. Pollock, V. Augustine, and N. Kraft, "Exploratory Study of Slack Q&A Chats as a Mining Source for Software Engineering Tools," in *Proceedings of the 16th International Conference on Mining Software Repositories (MSR'19)*, May 2019. [Online]. Available: https://doi.org/10.1109/MSR.2019.00075

[10] Peng Jiang, Hongping Fu, Chunxia Zhang, and Zhendong Niu, "A framework for opinion question answering," in *2010 6th International Conference on Advanced Information Management and Service (IMS)*, 2010, pp. 424–427.

[11] A. Balahur, E. Boldrini, A. Montoyo, and P. Martínez-Barco, "Opinion question answering: Towards a unified approach," in *ECAI*, 2010.

[12] M. Wan and J. McAuley, "Modeling ambiguity, subjectivity, and diverging viewpoints in opinion question answering systems," in *2016 IEEE 16th International Conference on Data Mining (ICDM)*, 2016, pp. 489–498.

[13] X. Su, G. Gao, and Y. Tian, "A framework to answer questions of opinion type," in *2010 Seventh Web Information Systems and Applications Conference*, 2010, pp. 166–169.

[14] M. M. Rahman, C. K. Roy, and D. Lo, "RACK: Automatic API Recommendation Using Crowdsourced Knowledge," in *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, vol. 1, March 2016, pp. 349–359.

[15] C. Chen, S. Gao, and Z. Xing, "Mining Analogical Libraries in Q A Discussions – Incorporating Relational and Categorical Knowledge into Word Embedding," in *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, vol. 1, March 2016, pp. 338–348.

[16] G. Uddin and F. Khomh, "Automatic summarization of api reviews," in *Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering*, ser. ASE 2017. IEEE Press, 2017, p. 159?170.

[17] B. Lin, F. Zampetti, G. Bavota, M. Di Penta, and M. Lanza, "Pattern-based mining of opinions in q&a websites," in *Proceedings of the 41st International Conference on Software Engineering*, ser. ICSE

'19. IEEE Press, 2019, p. 548?559. [Online]. Available: https://doi.org/10.1109/ICSE.2019.00066

[18] N. Novielli, F. Calefato, and F. Lanubile, "Towards Discovering the Role of Emotions in Stack Overflow," in *Proceedings of the 6th International Workshop on Social Software Engineering*, ser. SSE 2014. New York, NY, USA: ACM, 2014, pp. 33–36. [Online]. Available: http://doi.acm.org.udel.idm.oclc.org/10.1145/2661685.2661689

[19] V. Sinha, A. Lazar, and B. Sharif, "Analyzing developer sentiment in commit logs," in *2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR)*, 2016, pp. 520–523.

[20] A. D. Sorbo, S. Panichella, C. A. Visaggio, M. D. Penta, G. Canfora, and H. C. Gall, "Development Emails Content Analyzer: Intention Mining in Developer Discussions (T)," in *Proceedings of the 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, ser. ASE '15. Washington, DC, USA: IEEE Computer Society, 2015, pp. 12–23. [Online]. Available: https://doi-org.udel.idm.oclc.org/10.1109/ASE.2015.12

[21] Q. Huang, X. Xia, D. Lo, and G. C. Murphy, "Automating intention mining," *IEEE Transactions on Software Engineering*, pp. 1–1, 2018.

[22] E. Guzman and W. Maalej, "How do users like this feature? a fine grained sentiment analysis of app reviews," in *2014 IEEE 22nd International Requirements Engineering Conference (RE)*, 2014, pp. 153–162.

[23] L. V. G. Carreño and K. Winbladh, "Analysis of user comments: An approach for software requirements evolution," in *2013 35th International Conference on Software Engineering (ICSE)*, 2013, pp. 582–591.

[24] J. S. Culpepper, F. Diaz, and M. D. Smucker, "Research frontiers in information retrieval: Report from the third strategic workshop on information retrieval in lorne (swirl 2018)," *SIGIR Forum*, vol. 52, no. 1, p. 34?90, Aug. 2018. [Online]. Available: https://doi.org/10.1145/3274784.3274788

[25] L. Nie, H. Jiang, Z. Ren, Z. Sun, and X. Li, "Query expansion based on crowd knowledge for code search," *IEEE Transactions on Services Computing*, vol. 9, no. 5, pp. 771–783, Sept 2016.

[26] N. Rao, C. Bansal, T. Zimmermann, A. H. Awadallah, and N. Nagappan, "Analyzing web search behavior for software engineering tasks," 2020.

[27] A. Vtyurina, D. Savenkov, E. Agichtein, and C. L. A. Clarke, "Exploring conversational search with humans, assistants, and wizards," in *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, ser. CHI EA '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 2187?2193. [Online]. Available: https://doi.org/10.1145/3027063.3053175

[28] G. Uddin and F. Khomh, "Automatic mining of opinions expressed about apis in stack overflow," *IEEE Transactions on Software Engineering*, pp. 1–1, 2019.

[29] X. Zhou, B. Hu, Q. Chen, B. Tang, and X. Wang, "Answer sequence learning with neural networks for answer selection in community question answering," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Beijing, China: Association for Computational Linguistics, Jul. 2015, pp. 713–718. [Online]. Available: https://www.aclweb.org/anthology/P15-2117

[30] Z. Wang, W. Hamza, and R. Florian, "Bilateral multi-perspective matching for natural language sentences," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, ser. IJCAI?17. AAAI Press, 2017, p. 4144?4150.

[31] G. Shen, Y. Yang, and Z.-H. Deng, "Inter-weighted alignment network for sentence pair modeling," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, Sep. 2017, pp. 1179–1189. [Online]. Available: https://www.aclweb.org/anthology/D17-1122

[32] S. Joty, A. Barrón-Cedeño, G. Da San Martino, S. Filice, L. Màrquez, A. Moschitti, and P. Nakov, "Global thread-level inference for comment classification in community question answering," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, Sep. 2015, pp. 573–578. [Online]. Available: https://www.aclweb.org/anthology/D15-1068

[33] A. Moschitti, S. Quarteroni, R. Basili, and S. Manandhar, "Exploiting syntactic and shallow semantic kernels for question answer classification," in *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*. Prague, Czech Republic: Association for Computational Linguistics, Jun. 2007, pp. 776–783. [Online]. Available: https://www.aclweb.org/anthology/P07-1098

[34] A. Di 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 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, ser. FSE 2016. New York, NY, USA: ACM, 2016, pp. 499–510. [Online]. Available: http://doi.acm.org/10.1145/2950290.2950299

[35] S. Panichella, A. Di Sorbo, E. Guzman, C. A. Visaggio, G. Canfora, and H. C. Gall, "ARdoc: App Reviews Development Oriented Classifier," in *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, ser. FSE 2016. New York, NY, USA: ACM, 2016, pp. 1023–1027. [Online]. Available: http://doi.acm.org.udel.idm.oclc.org/10.1145/2950290.2983938

[36] P. Chatterjee, K. Damevski, N. Kraft, and L. Pollock, "Software-related Slack Chats with Disentangled Conversations," in *Proceedings of the 17th International Conference on Mining Software Repositories (MSR'20)*, May 2020.

[37] M. A. Bujang and N. Baharum, "A simplified guide to determination of sample size requirements for estimating the value of intraclass correlation coefficient: a review." *Archives of Orofacial Science*, vol. 12, no. 1, 2017.

[38] J. R. Landis and G. G. Koch, "The measurement of observer agreement for categorical data," *biometrics*, pp. 159–174, 1977.

[39] M. Elsner and E. Charniak, "You talking to me? a corpus and algorithm for conversation disentanglement," in *Proc. Association of Computational Linguistics: Human Language Technology*, 2008, pp. 834–842.

[40] D. C. Uthus and D. W. Aha, "Multiparticipant chat analysis: A survey," *Artificial Intelligence*, vol. 199, pp. 106–121, 2013.

[41] J. K. Kummerfeld, S. R. Gouravajhala, J. J. Peper, V. Athreya, C. Gunasekara, J. Ganhotra, S. S. Patel, L. Polymenakos, and W. S. Lasecki, "A large-scale corpus for conversation disentanglement," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, July 2019, pp. 3846–3856. [Online]. Available: https://www.aclweb.org/anthology/P19-1374.pdf

[42] P. Runeson, M. Host, A. Rainer, and B. Regnell, *Case Study Research in Software Engineering: Guidelines and Examples*, 1st ed. Wiley Publishing, 2012.

[43] M. Tan, B. Xiang, and B. Zhou, "Lstm-based deep learning models for non-factoid answer selection," *ArXiv*, vol. abs/1511.04108, 2015.

[44] A. Rücklé and I. Gurevych, "Representation learning for answer selection with LSTM-based importance weighting," in *IWCS 2017 — 12th International Conference on Computational Semantics — Short papers*, 2017. [Online]. Available: https://www.aclweb.org/anthology/W17-6935

[45] Z. Zhao, H. Lu, V. W. Zheng, D. Cai, X. He, and Y. Zhuang, "Community-based question answering via asymmetric multi-faceted ranking network learning," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, ser. AAAI?17. AAAI Press, 2017, p. 3532?3538.

[46] Y. Kim, "Convolutional neural networks for sentence classification," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1746–1751. [Online]. Available: https://www.aclweb.org/anthology/D14-1181

[47] L. Shi, M. Xing, M. Li, Y. Wang, L. Shoubin, and Q. Wang, "Detection of Hidden Feature Requests from Massive Chat Messages via Deep Siamese Network," in *Proceedings of the 42nd International Conference on Software Engineering*, ser. ICSE '20. New York, NY, USA: ACM, 2020.

[48] A. Graves, A. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 6645–6649.

[49] Y. Xiang, Q. Chen, X. Wang, and Y. Qin, "Answer selection in community question answering via attentive neural networks," *IEEE Signal Processing Letters*, vol. 24, no. 4, pp. 505–509, 2017.

[50] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, p. 1735?1780, Nov. 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735

[51] M. R. Islam and M. F. Zibran, "Leveraging automated sentiment analysis in software engineering," in *Proceedings of the 14th International Conference on Mining Software Repositories*, ser. MSR ?17. IEEE

Press, 2017, p. 203?214. [Online]. Available: https://doi.org/10.1109/MSR.2017.9

[52] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. 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*. Seattle, Washington, USA: Association for Computational Linguistics, Oct. 2013, pp. 1631–1642. [Online]. Available: https://www.aclweb.org/anthology/D13-1170

[53] C. J. Hutto and E. Gilbert, "Vader: A parsimonious rule-based model for sentiment analysis of social media text," in *ICWSM*, 2014.

[54] W. Wu, X. Sun, and H. Wang, "Question condensing networks for answer selection in community question answering," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, Jul. 2018, pp. 1746–1755. [Online]. Available: https://www.aclweb.org/anthology/P18-1162

[55] Q. Ye, K. Misra, H. Devarapalli, and J. T. Rayz, "A sentiment based non-factoid question-answering framework," in *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, 2019, pp. 372–377.

[56] L.-W. Ku, Y.-T. Liang, and H.-H. Chen, "Question analysis and answer passage retrieval for opinion question answering systems," in *International Journal of Computational Linguistics & Chinese Language Processing, Volume 13, Number 3, September 2008: Special Issue on Selected Papers from ROCLING XIX*, Sep. 2008, pp. 307–326. [Online]. Available: https://www.aclweb.org/anthology/O08-5003

[57] F. Eskandari, H. Shayestehmanesh, and S. Hashemi, "Predicting best answer using sentiment analysis in community question answering systems," in *2015 Signal Processing and Intelligent Systems Conference (SPIS)*, 2015, pp. 53–57.

[58] S. Moghaddam and M. Ester, "Aqa: Aspect-based opinion question answering," in *2011 IEEE 11th International Conference on Data Mining Workshops*, 2011, pp. 89–96.

[59] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: An update," *SIGKDD Explor. Newsl.*, vol. 11, no. 1, pp. 10–18, Nov. 2009. [Online]. Available: http://doi.acm.org/10.1145/1656274.1656278

[60] F. Chollet *et al.* (2015) Keras. [Online]. Available: https://github.com/fchollet/keras

[61] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, no. null, p. 281?305, Feb. 2012.

[62] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, p. 1929?1958, Jan. 2014.

[63] L. Prechelt, "Early stopping - but when?" in *Neural Networks: Tricks of the Trade, volume 1524 of LNCS, chapter 2*. Springer-Verlag, 1997, pp. 55–69.

[64] Y. Zhang and D. Hou, "Extracting problematic api features from forum discussions," in *2013 21st International Conference on Program Comprehension (ICPC)*, May 2013, pp. 142–151.

[65] 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*, ser. ASE 2018. New York, NY, USA: Association for Computing Machinery, 2018, p. 214?224. [Online]. Available: https://doi.org/10.1145/3238147.3238208

[66] X. Ren, Z. Xing, X. Xia, G. Li, and J. Sun, "Discovering, explaining and summarizing controversial discussions in community q&a sites," in *Proceedings of the 34th IEEE/ACM International Conference on Automated Software Engineering*, ser. ASE ?19. IEEE Press, 2019, p. 151?162. [Online]. Available: https://doi.org/10.1109/ASE.2019.00024

[67] N. Novielli, F. Calefato, and F. Lanubile, "The Challenges of Sentiment Detection in the Social Programmer Ecosystem," in *Proceedings of the 7th International Workshop on Social Software Engineering*, ser. SSE 2015. New York, NY, USA: ACM, 2015, pp. 33–40. [Online]. Available: http://doi.acm.org.udel.idm.oclc.org/10.1145/2804381.2804387

[68] Z. Xiong, P. Liang, C. Yang, and T. Liu, "Assumptions in oss development: An exploratory study through the hibernate developer mailing list," in *2018 25th Asia-Pacific Software Engineering Conference (APSEC)*, 2018, pp. 455–464.

[69] M. Goul, O. Marjanovic, S. Baxley, and K. Vizecky, "Managing the enterprise business intelligence app store: Sentiment analysis supported requirements engineering," in *2012 45th Hawaii International Conference on System Sciences*, 2012, pp. 4168–4177.

[70] L. Shrestha and K. McKeown, "Detection of question-answer pairs in email conversations," in *Proceedings of the 20th International Conference on Computational Linguistics*, ser. COLING '04. Stroudsburg, PA, USA: Association for Computational Linguistics, 2004. [Online]. Available: https://doi.org/10.3115/1220355.1220483

[71] G. Cong, L. Wang, C.-Y. Lin, Y.-I. Song, and Y. Sun, "Finding question-answer pairs from online forums," in *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '08. New York, NY, USA: ACM, 2008, pp. 467–474. [Online]. Available: http://doi.acm.org/10.1145/1390334.1390415

[72] S. Gottipati, D. Lo, and J. Jiang, "Finding relevant answers in software forums," in *Proceedings of the 2011 26th IEEE/ACM International Conference on Automated Software Engineering*, ser. ASE '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 323–332. [Online]. Available: http://dx.doi.org/10.1109/ASE.2011.6100069

[73] S. Henß, M. Monperrus, and M. Mezini, "Semi-automatically extracting faqs to improve accessibility of software development knowledge," in *Proceedings of the 34th International Conference on Software Engineering*, ser. ICSE '12. Piscataway, NJ, USA: IEEE Press, 2012, pp. 793–803. [Online]. Available: http://dl.acm.org/citation.cfm?id=2337223.2337317

[74] R. Alkadhi, T. Lata, E. Guzmany, and B. Bruegge, "Rationale in Development Chat Messages: An Exploratory Study," in *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*, May 2017, pp. 436–446.

[75] R. Alkadhi, M. Nonnenmacher, E. Guzman, and B. Bruegge, "How do developers discuss rationale?" in *2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, vol. 00, March 2018, pp. 357–369. [Online]. Available: doi.ieeecomputersociety.org/10.1109/SANER.2018.8330223

[76] S. A. Chowdhury and A. Hindle, "Mining StackOverflow to Filter Out Off-Topic IRC Discussion," in *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*, May 2015, pp. 422–425.

[77] E. Shihab, Z. M. Jiang, and A. E. Hassan, "Studying the Use of Developer IRC Meetings in Open Source Projects," in *2009 IEEE International Conference on Software Maintenance*, Sept 2009, pp. 147–156.

[78] M. S. Elliott and W. Scacchi, "Free software developers as an occupational community: Resolving conflicts and fostering collaboration," in *Proceedings of the 2003 International ACM SIGGROUP Conference on Supporting Group Work*, ser. GROUP '03. New York, NY, USA: ACM, 2003, pp. 21–30. [Online]. Available: http://doi.acm.org/10.1145/958160.958164

[79] S. Panichella, G. Bavota, M. D. Penta, G. Canfora, and G. Antoniol, "How developers' collaborations identified from different sources tell us about code changes," in *2014 IEEE International Conference on Software Maintenance and Evolution*, Sept 2014, pp. 251–260.

[80] E. Paikari and A. van der Hoek, "A framework for understanding chatbots and their future," in *Proceedings of the 11th International Workshop on Cooperative and Human Aspects of Software Engineering*, ser. CHASE '18. New York, NY, USA: ACM, 2018, pp. 13–16. [Online]. Available: http://doi.acm.org.udel.idm.oclc.org/10.1145/3195836.3195859

[81] C. Lebeuf, M. D. Storey, and A. Zagalsky, "How Software Developers Mitigate Collaboration Friction with Chatbots," *CoRR*, vol. abs/1702.07011, 2017. [Online]. Available: http://arxiv.org/abs/1702.07011