Automating App Review Response Generation Based on Contextual Knowledge

CUIYUN GAO, Harbin Institute of Technology, Shenzhen, China
WENJIE ZHOU, The Key Laboratory for Computer Systems of State Ethnic Affairs Commission, Southwest Minzu University, China
XIN XIA, Software Engineering Application Technology Lab, Huawei, China
DAVID LO, Singapore Management University, Singapore
QI XIE, The Key Laboratory for Computer Systems of State Ethnic Affairs Commission, Southwest Minzu University, China
MICHAEL R. LYU, The Chinese University of Hong Kong, China

User experience of mobile apps is an essential ingredient that can influence the user base and app revenue. To ensure good user experience and assist app development, several prior studies resort to analysis of app reviews, a type of repository that directly reflects user opinions about the apps. Accurately responding to the app reviews is one of the ways to relieve user concerns and thus improve user experience. However, the response quality of the existing method relies on the pre-extracted features from other tools, including manually labelled keywords and predicted review sentiment, which may hinder the generalizability and flexibility of the method. In this article, we propose a novel neural network approach, named CoRe, with the contextual knowledge naturally incorporated and without involving external tools. Specifically, CoRe integrates two types of contextual knowledge in the training corpus, including official app descriptions from app store and responses of the retrieved semantically similar reviews, for enhancing the relevance and accuracy of the generated review responses. Experiments on practical review data show that CoRe can outperform the state-of-the-art method by 12.36% in terms of BLEU-4, an accuracy metric that is widely used to evaluate text generation systems.

CCS Concepts: • **Software and its engineering** → *Context specific languages*; • **Computing methodologies** → *Machine learning approaches;*

Additional Key Words and Phrases: User reviews, retrieved responses, app descriptions, pointer-generator network

This work was supported by the National Natural Science Foundation of China under project No. 62002084, the National Natural Science Foundation of China under project No. 61502401, the Research Grants Council of the Hong Kong Special Administrative Region, China (CUHK 14210717), the National Research Foundation, Singapore under its Industry Alignment Fund — Pre-positioning (IAF-PP) Funding Initiative.

Authors' addresses: C. Gao, Harbin Institute of Technology, Shenzhen, China; email: gaocuiyun@hit.edu.cn; W. Zhou and Q. Xie (corresponding author), The Key Laboratory for Computer Systems of State Ethnic Affairs Commission, Southwest Minzu University, Sichuan, China; emails: zhouwenjie2@stu.swun.edu.cn, qi.xie.swun@gmail.com; X. Xia, Software Engineering Application Technology Lab, Huawei, China; email: xin.xia@acm.org; D. Lo, Singapore Management University, Singapore; email: davidlo@smu.edu.sg; M. R. Lyu, The Chinese University of Hong Kong, Hong Kong, China; email: lyu@cse.cuhk.edu.hk.

1049-331X/2021/10-ART11 $15.00
https://doi.org/10.1145/3464969

ACM Reference format:
Cuiyun Gao, Wenjie Zhou, Xin Xia, David Lo, Qi Xie, and Michael R. Lyu. 2021. Automating App Review Response Generation Based on Contextual Knowledge. *ACM Trans. Softw. Eng. Methodol.* 31, 1, Article 11 (October 2021), 36 pages.
https://doi.org/10.1145/3464969

1 INTRODUCTION

According to the report released by Reference [8], there are over 5 billion mobile users worldwide, with global internet penetration standing at 57%. For these app users, they could choose the apps for usage from a vast number of mobile apps; for example, Google Play and Apple's App Store provide 2.5 million and 1.8 million apps, respectively [9]. An essential factor for apps to be successful is to guarantee the quality of app functionalities and ensure good user experience. User reviews, which serve as a communication channel between users and developers, can reflect immediate user experience, including app bugs and features to add or modify. Recent research has leveraged natural language processing and machine learning techniques to extract useful information from user reviews to help developers test, optimize, maintain, and categorize apps (see e.g., References [21, 26, 30, 33, 62]) for ensuring good user experience.

The app stores such as Google Play and App Store also allow developers to respond to the reviews [5, 7] and encourage them to respond to reviews promptly and precisely for creating a better user experience and improving app ratings. A recent study by Hassan et al. [34] confirmed the positive effects of review reply. Specifically, they found that responding to a review increases the chances of a user updating their given rating by up to six times in comparison with no response. McIlroy et al. [51] discovered that users change their ratings 38.7% of the time following a developer response, with a median increase of 20% in the rating. Despite the advantage of review response, developers of many apps never respond to the reviews [34, 51]. One major reason is the plentiful reviews received for the mobile apps, e.g., the Facebook app on Google Play collects thousands of reviews per day [13]. It is labor-intensive and time-consuming for developers to respond to each piece of review. Therefore, the prior work [27] initiates automating the review response process.

Review response generation can be analogical to social dialogue generation [41, 71] in the natural language processing field. Different from social dialogue generation, app review-response generation is more domain-specific or even app-specific, and hence, its performance strongly relies on the establishment of the domain knowledge. For example, the response for the review of one app may not be applicable for the review of another app even though the reflected issues are similar. As illustrated in Figure 1, both review instances are complaining about the Internet connection issue, but developers' suggested solutions are different. For the UC browser app, the developer suggests to clear cache while for the PicsArt photo editor app, the developers undertake to simplify the options of save and share edits.

To automatically learn the domain-specific knowledge, Gao et al. [27] proposed a **Neural Machine Translation (NMT)** [64]-based neural network, named RRGen, which can encode user reviews with an embedding layer and decode them into developers' response through a **Gated Recurrent Unit (GRU)** [19] model with attention mechanism. External review attributes, including review length, rating, predicted sentiment, app category, and pre-defined keywords, are adopted to better encode the semantics of user reviews. Although good performance is demonstrated, the design of RRGen exhibits two main limitations. First, the performance of the external tools such as SURF [62] (for determining pre-defined keywords) and SentiStrength [65] (for estimating review sentiment) may impact the results of RRGen. For example, when the keywords in the reviews are not in the pre-defined keyword dictionary provided by SURF, RRGen would fail to capture the

★★★★★ **User name1 26/12/2016**

After recent updation, my uc browser wasn't work properly especially uc news tool. Whenever I did refresh, it displayed need internet connection all time. Please fix my issue.

Developer1 27/12/2016

Hi my dear so sorry for the problem. Do it happen in all news or individual one? Can you open another sitting in uc browser? We suggest you clear cache, turn on off cloud boost, change the access point and retry.

(a) One review instance of the UC Browser app.

★☆☆☆☆ **User name2 21/12/2016**

It was perfect I could take a photo then edit it and save it to move it to my pc or to send it to a friend via bluetooth but now if I'm not connecting to the internet it won't let me do anything!

Developer2 22/12/2016

We're sad to hear that you feel this way as our social network is getting bigger and more people are sharing their edits with us. We will simplify the option of save and share edits make them happen simultaneously .

(b) One review instance of the PicsArt Photo Editor app.

Fig. 1. Review instances from two separate apps. The underlined texts highlight the main issues reported in reviews and corresponding suggested solutions from developers.

semantics of the review. Second, RRGen faces a common problem of NMT-based approaches, i.e., they generally prefer high-frequency words in the corpus and the generated responses are often generic and not informative [14, 73, 76].

To alleviate the above limitations, we propose a novel neural architecture, namely, **C**ontextual kn**o**wledge-based app **R**eview r**e**sponse generation **(CoRe)**, built upon official app descriptions and responses of retrieved similar reviews from the training corpus. For mitigating the first limitation, we incorporate app descriptions, which usually contain sketches of app functionalities [11]. Based on app descriptions, the neural model can learn to pay attention to app functionality-related words in the reviews, without feeding pre-defined keywords into the model. For relieving the second limitation, we involve responses of similar reviews based on **Information Retrieval (IR)**-based approach. The IR-based approach [38] has proven useful in leveraging the responses of similar conversations for producing relevant responses, so the IR-based retrieved responses are highly probable to contain the words in the expected responses (including the low-frequency ones). To incorporate the words in the retrieved responses, CoRe utilizes pointer-generator network [59] to adaptively copy words from the responses instead of simply from a fixed vocabulary obtained from the training corpus.

Experiments based on 309,246 review-response pairs from 58 popular apps show that CoRe significantly outperforms the state-of-the-art model by 12.36% in terms of BLEU-4 score [55] (an accuracy measure that is widely used to evaluate text generation systems). Human study with 20 programmers through Tencent Online Questionnaire [6] further confirms that CoRe can generate

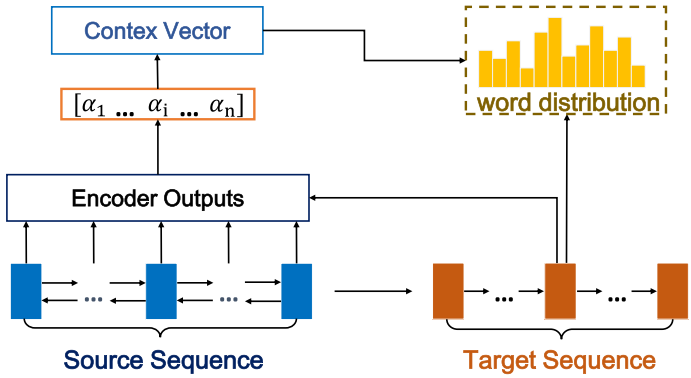


Fig. 2. Graphical illustration of the attentional bi-LSTM encoder-decode model.

a more relevant and accurate response than RRGen. We release the whole replication package through this link,[1] including the dataset, experimental configurations, and source code.

The remainder of this article is organized as follows: Section 2 introduces the background of our work. Section 3 illustrates the proposed approach. Section 4 and Section 5 detail our experimental settings and the experimental results, respectively. Section 6 describes the human evaluation results. Section 7 discusses the advantages of the proposed approach and threats to validity. Section 8 surveys the related work. Section 9 concludes the article.

2 BACKGROUND

In this section, we introduce the background knowledge of the proposed approach, including attentional encoder-decoder model and pointer-generator model.

2.1 Attentional Encoder-decoder Model

Encoder-decoder model, also called sequence-to-sequence model, has demonstrated the ability to model the variable-length input and output, e.g., words and sentences. Figure 2 illustrates the architecture of the attentional encoder-decoder model. Generally, tokens of the source sequence $\mathbf{w} = (w_1, w_2, \ldots, w_n)$ (n is the number of input tokens) are fed one-by-one into the encoder (a single-layer bidirectional GRU [19], as shown in Figure 2), producing a sequence of encoder hidden states $\mathbf{h} = (h_1, h_2, \ldots, h_n)$. On each step t, the decoder (a single-layer unidirectional GRU) is often trained to predict the next word y_t based on the context vector $\mathbf{c}$ and previously predicted words $\{y_1, \ldots, y_{t-1}\}$, and has decoder state s_t. The context vector c_t depends on a sequence of encoder hidden states $\mathbf{h}$ and is computed as a weighted sum of the hidden states [15]:

$$\begin{aligned} c_t &= \sum_{j}^{n} \alpha_{tj} h_j, \\ \alpha_{tj} &= \text{softmax}(e_{tj}), \end{aligned} \tag{1}$$

where e_{tj} measures the similarity degree between the input hidden state h_j and decoder state s_{t-1}. The attention weight α_t can be viewed as a probability distribution over the source words, and higher probabilities render the decoder pay more attention to the corresponding input during producing the next word. The context vector is then concatenated with the decoder state s_t and

[1] https://bit.ly/3kv6WEl.

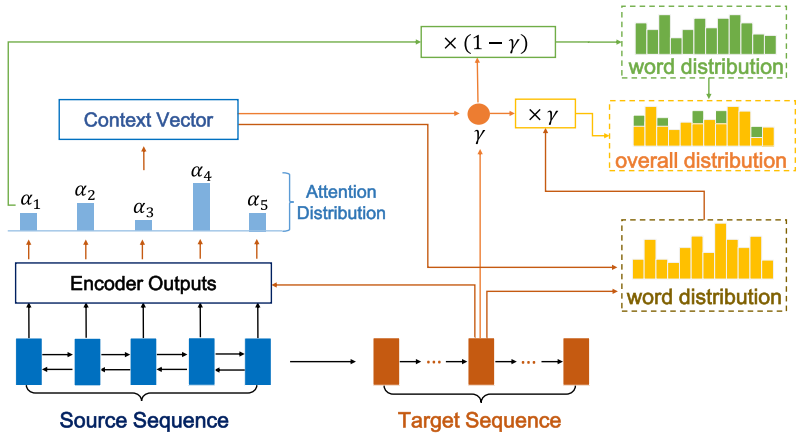


Fig. 3. Graphical illustration of the pointer-generator model.

fed through two linear layers to generate the vocab distribution:

$$P_t^{\text{vocab}}(w) = \text{softmax}(v'(v[s_t, c_t] + b) + b'), \tag{2}$$

where v, v', b, and b' are learnable parameters, and P_t^{vocab} is a probability distribution over all the words in the vocabulary. The decoder state s_t depends on the last decoder state s_{t-1} and the previous decoder output y_{t-1} at a decoder step t, where $s_{t=1}$ is the encoder's final hidden state and $y_{t=1}$ is a special character that indicates the beginning of a sentence. The model is trained to minimize the negative log likelihood:

$$\text{loss} = \min \frac{1}{N} \sum_i -\log P(y_i|x_i), \tag{3}$$

where each (x_i, y_i) is a (source sequence, target sequence) pair from the training set.

2.2 Pointer-generator Model

Pointer-generator networks [59, 68] allow sequence-to-sequence models to predict words during decoding by either copying words via pointing or generating words from a fixed vocabulary. Figure 3 depicts the architecture of the pointer-generator model. As can be seen, besides computing the context vector c_t and attention weight α_t, the generation probability $\gamma_t \in [0, 1]$ for step t is calculated for the context vector c_t, the decoder state s_t, and the decoder input w_t:

$$\gamma_t = \sigma\left(\omega_c^\intercal c_t + \omega_s^\intercal s_t + \omega_w^\intercal w_t + b_{ptr}\right), \tag{4}$$

where vectors ω_c, ω_s, ω_w, and scalar b_{ptr} are learnable parameters. σ is the sigmoid function. γ_t can be regarded as an indicator of which source the predicted word comes from. The probability distribution over the *overall vocabulary* is computed as:

$$P_t(w) = \gamma_t \cdot P_t^{\text{vocab}}(w) + (1 - \gamma_t) \cdot \sum_{i:w_i=w} \alpha_{ti}. \tag{5}$$

If w is an **out-of-vocabulary (OOV)** word, then $P_t^{\text{vocab}}(w)$ is zero. In this way, point-generator models are able to generate OOV words. The loss function is the same as described in Equation (3).

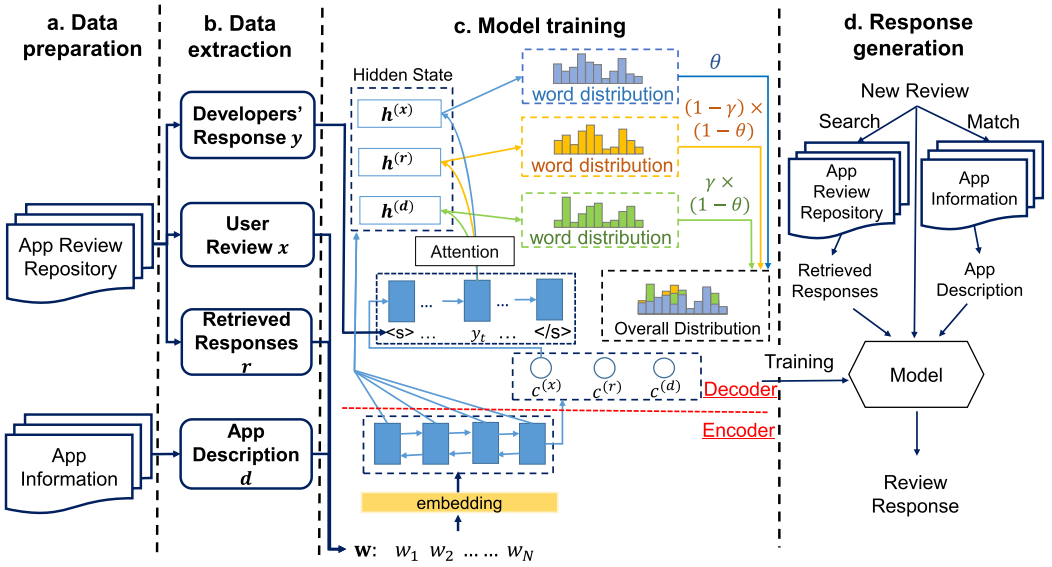


Fig. 4. Overall architecture of CoRe.

3 METHODOLOGY

This section describes our proposed model CoRe, which builds upon the basic pointer-generator model. Besides user reviews, two types of contextual knowledge, including app descriptions and responses of the retrieved similar reviews from the training corpus, are regarded as the source sequence. The developers' responses are treated as the target sequence. App descriptions generally describe apps' functionalities [11], so with app descriptions integrated, the words related to app functionalities are prone to be captured. Semantically similar reviews are involved, since the semantics of the corresponding responses tend to be identical. For each piece of review, the semantic distances with other reviews in the training set are computed as the cosine similarity between the unigram tf-idf representations, and only the responses of the top K reviews with highest similarity scores are considered for the response generation.

The overall architecture of the proposed model is illustrated in Figure 4. CoRe is mainly composed of four stages: Data preparation, data extraction, model training, and response generation. We first preprocess the app reviews, their responses, and app descriptions collected from Google Play. The processed data are then parsed into a parallel corpus of user reviews, corresponding responses, the retrieved responses, and app description. Based on the parallel corpus, we build and train a pointer-generator-based model with the contextual knowledge holistically considered. The details are elaborated in the following subsections.

3.1 Source Sequence Encoding

Let $\mathbf{w} = (w_1, w_2, \ldots, w_n)$ be a sequence of source tokens, which can be the input review $\mathbf{x}$, app description $\mathbf{d}$, or the response for each of top K retrieved similar reviews $\mathbf{r}^{(k)}, 1 \leq k \leq K$. We first obtain a trainable embedded representation of each token in the sequence and then adopt bi-GRU to encode the sequence of the embedding vectors:

$$e^{(x)}, \mathbf{h}^{(x)} = \text{bi-GRU}(\mathbf{x}), \tag{6}$$

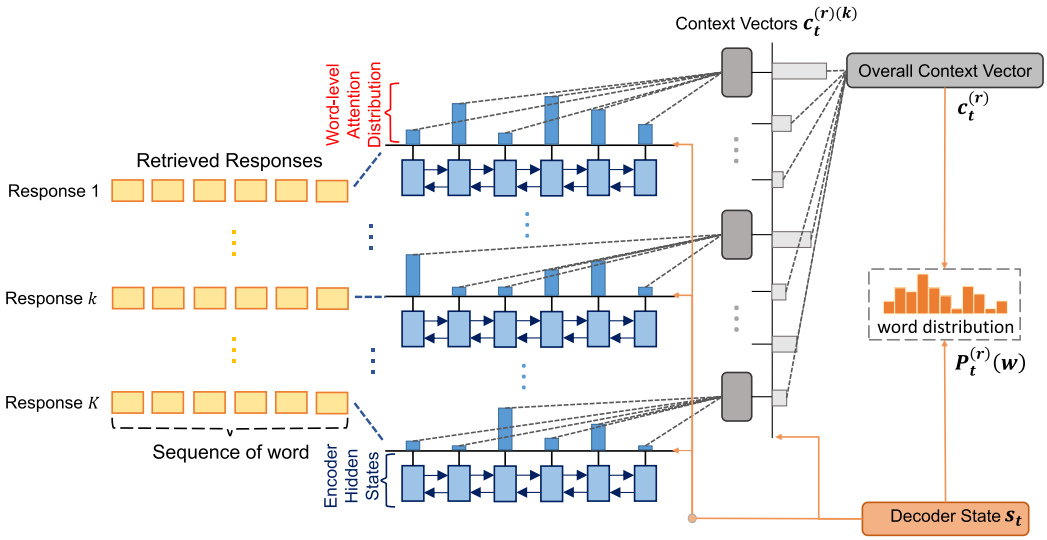


Fig. 5. Illustration of the hierarchical pointer network for copying tokens from the retrieved K responses.

$$e^{(d)}, \mathbf{h}^{(d)} = \text{bi-GRU}(\mathbf{d}), \tag{7}$$

$$e^{(r)(k)}, \mathbf{h}^{(r)(k)} = \text{bi-GRU}(\mathbf{r}^{(k)}), \tag{8}$$

where e^{Δ} and $\mathbf{h}^{\Delta} = (h_1, h_2, \ldots, h_n)$ denote the final hidden state of the bi-LSTM and outputs of bi-LSTM at all steps, where $\Delta \in [(x), (d), (r)(1), \ldots, (r)(k), \ldots, (r)(K)]$.

3.2 Contextual Knowledge Integration

Different from the basic pointer-generator network [59], CoRe also allows integrating tokens from the contextual information besides the input reviews. At decoder step t, the decoder state s_t is used to attend over the app description tokens and the retrieved response tokens to produce a probability distribution over the tokens appearing in the description and retrieved responses, respectively. These distributions are then integrated with the attention distribution obtained by the decoder over the fixed vocabulary to compute an overall distribution.

3.2.1 Copying Tokens from App Description. Similar to the basic attentional encoder-decoder model, we encode the description tokens $\mathbf{d}$ and apply attention to the encoder outputs at a decoder step t. This produces the attention weights $\alpha_t^{(d)}$ and a representation of the entire context $c_t^{(d)}$. The context vector is then employed to obtain the probability distribution $P_t^{(d)}(w)$ over the tokens in the app description:

$$\alpha_t^{(d)}, c_t^{(d)} = \text{Attention}(\mathbf{h}^{(d)}, s_t) \tag{9}$$

$$P_t^{(d)}(w) = g\left(s_t, y_{t-1}, c_t^{(d)}\right), \tag{10}$$

where $\mathbf{h}^{(d)}$ indicates the encoder outputs as computed in Equation (7) and g is a non-linear mapping function.

3.2.2 Copying Tokens from Responses of the Retrieved Reviews. To integrate the responses of the K retrieved reviews, we adapt the hierarchical pointer network, as shown in Figure 5, for involving

tokens from multiple extracted responses. Based on the token-level representations $\mathbf{h}^{(r)(k)}$, the decoder state s_t is used to attend over the tokens in each retrieved response:

$$\alpha_t^{(r)(k)}, c_t^{(r)(k)} = \text{Attention}(\mathbf{h}^{(r)(k)}, s_t), \tag{11}$$

$$\alpha_t^{(r)}, c_t^{(r)} = \text{Attention}\left(\left[c_t^{(r)(1)}, \ldots, c_t^{(r)(K)}\right], s_t\right), \tag{12}$$

$$P_t^{(r)}(w) = g\left(s_t, y_{t-1}, c_t^{(r)}\right), \tag{13}$$

where $\mathbf{h}^{(r)(k)}$ is the output of the encoder for the response of the top kth retrieved reviews. The context vector $c_t^{(r)}$ for all the retrieved responses are obtained based on the context vectors of all the K responses, following the Equation (12). $P_t^{(r)}(w)$ means the probability distribution over the tokens in the retrieved K responses.

3.2.3 Attention Fusion. We first fuse the two vocabulary distributions $P_t^{(d)}(w)$ and $P_t^{(r)}(w)$, which represent the probabilities of copying tokens from the app description and retrieved responses, respectively. We compute the fused attention vector using the decoder state s_t, the overall app description representation $c_t^{(d)}$, and overall retrieved response representation $c_t^{(r)}$ (Equation 14). The computed attention weight γ_t is adopted to combine the two copying distributions as Equation (15):

$$\gamma_t, c_t^{\text{fuse}} = \text{Attention}\left(\left[c_t^{(d)}, c_t^{(r)}\right], s_t\right), \tag{14}$$

$$P_t^{\text{fuse}}(w) = \gamma_t \cdot P_t^{(d)}(w) + (1 - \gamma_t) \cdot P_t^{(r)}(w). \tag{15}$$

The overall distribution $P_t(w)$ for the training vocabulary at each decoder step t is calculated based on the context vector c_t^{fuse} of the two contextual sources and decoder state s_t:

$$\begin{aligned} \theta_t &= \sigma\left(\omega_f^\intercal c_t^{\text{fuse}} + \omega_s^\intercal s_t + \omega_x^\intercal x_t + b_{\text{ptr}}\right), \\ P_t(w) &= \theta_t \cdot P_t^{\text{vocab}}(w) + (1 - \theta_t) \cdot P_t^{\text{fuse}}(w), \end{aligned} \tag{16}$$

where ω_f, ω_s, ω_x, and b_{ptr} are learnable parameters, x_t is the decoder input, and $P_t^{\text{vocab}}(w)$ indicates the vocabulary distribution based on the input reviews only (referring to Equation (2)).

3.3 Model Training and Validation

3.3.1 Training. We train the whole network with the negative log-likelihood loss function of

$$J_{\text{loss}}(\Theta) = -\frac{1}{|y|}\sum_{t=1}^{|y|} \log\left(p_t\left(y_t | y < t, \mathbf{x}, \mathbf{d}, \left\{\mathbf{r}^{(k)}\right\}_{k=1}^{K}\right)\right), \tag{17}$$

for a training sample $(\mathbf{x}, \mathbf{y}, \mathbf{d}, \{\mathbf{r}^{(i)}\}_{i=1}^{K})$ where Θ denotes all the learnable model parameters. The attentional encoder-decoder model has various implementations. We adopt bidirectional **Gated Recurrent Units (GRUs)** [19], which is a popular basic encoder-decoder model and performs well in many text generation tasks [20, 72]. The hidden units of GRUs are set as 200 and word embeddings are initiated with pre-trained 100-dimensional GloVe vectors [3]. The maximum sequence lengths for reviews, app descriptions, and retrieved responses are all defined as 200. We save the model every 200 batches. The number of retrieved responses, the dropout rate, and the number of hidden layers are defined as 4, 0.1, and 1, respectively. Details of parameter tuning are discussed in Section 5.3. The whole model is trained using the minibatch Adam [40], a stochastic

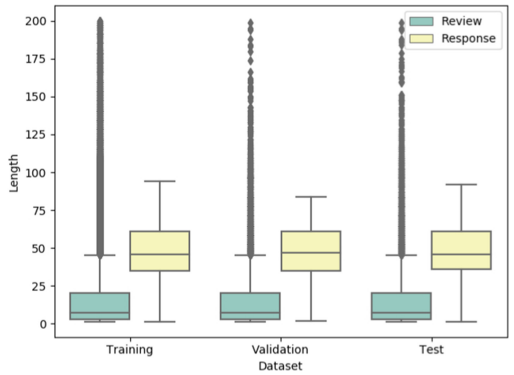


Fig. 6. Length distribution of the reviews and responses in the benchmark dataset.

optimization approach that can automatically adjust the learning rate. The batch size is set as 32. During training the neural networks, we limit the source and target vocabulary to the top 10,000 words that most frequently appear in the training set.

For implementation, we use PyTorch [4], an open-source deep learning framework. We train our model in a server with Intel(R) Xeon(R) Gold 6230 CPU @ 2.10 GHz, Tesla T4 16 G. The training lasts ~8 hours with three epochs following the setting in Reference [27].

3.3.2 Validation. We evaluate on the test set after the batch during which the trained model shows an improved performance on the validation set regarding BLEU score [55]. The evaluation results are the highest test score and corresponding generated response. We use the same GPU as used in training, and the testing process cost around 30 minutes.

4 EXPERIMENTAL SETUP

In this section, we elaborate on the setup of our experiments, including experimental dataset, the evaluation metric, and baseline approaches.

4.1 Experimental Dataset

We perform experiments for verifying the effectiveness of the proposed model on the recently released review response dataset [27]. The dataset includes 309,246 review-response pairs from 58 popular apps, with 279,792, 14,727, and 14,727 pairs in the training, validation, and test sets, respectively, following 8:1:1 random split. The statistics of the lengths of app reviews and responses are illustrated in Figure 6. The average review length is ~15 with maximum at around 200 on the training, validation, and test sets. The maximum word number of user reviews may be attributed to the length limit of Google Play reviews [2]. We also observe that the length distributions of the reviews/responses are relatively consistent among the training, validation, and test sets. In this work, we set the maximum sequence length as 200 following the prior study cited in Reference [27] to ensure fair comparison. We also discuss the impact of the maximum sequence lengths in Section 7.

Besides the review-response pairs, we crawled the corresponding app descriptions from Google Play for the 58 subject apps. For the app descriptions, we remove all special characters such as "★" and conduct similar preprocessing steps as the review preprocessing steps [27], including lowercase and lemmatization. After the basic preprocessing, we observe that the maximum, median, and minimum lengths of the app descriptions are 625, 300, and 43 words, respectively, with the

average length at 314. Since the semantics of long input texts are difficult to be effectively learned by the basic attentional encoder-decoder model [75], we reduce the input description lengths by manually filtering out the sentences irrelevant to the app features/functionalities (e.g., the sentences explicitly encouraging users to download the apps, "*download the highest rated travel app now and join thousands of bookers like you finding unmissable hotel deals!*"). The pruning process is conducted by the first two authors together. One author first extracts the informative description sentences for each app. We find that although the official app descriptions vary in content and length, they present similar structure, i.e., introduction about the app functionality in general first and then about specific functions and advantages, finally encouraging users to download and use the apps. So, one author extracts one or two sentences from the general introduction, keeps the sentences that describe specific functions, and extracts one or two sentences from the conclusion words. The other author then checks the extracted sentences and marks the cases (4/58, 6.9%) for which he disagrees with the first author, e.g., the other author finds that some sentences that are irrelevant to the app functionality are included in the pruned description. For the disagreement cases, the two authors further discuss to reach a consensus. The pruning process costs us around 1.5 hours for the 58 subject apps. The maximum, median, and minimum lengths of the reduced descriptions are 198, 151, and 43 words, respectively, with the average length at 146. For the retrieved responses of CoRe as input, only the responses in the training set are considered instead of those in the validation or test sets.

4.2 Evaluation Metric

BLEU is a metric widely used in natural language processing and software engineering fields to evaluate generative tasks (e.g., machine translation, dialogue generation, and code commit message generation) [35, 39, 41, 76]. It calculates the frequencies of the co-occurrence of n-grams between the ground truth $\hat{y}$ and the generated sequence y to judge their similarity:

$$p_n(y, \hat{y}) = \frac{\sum_j \min(h(j, \hat{y}), h(j, y))}{\sum_j h(j, \hat{y})}, \tag{18}$$

where j indexes all possible n-grams, and $h(j, y)$ or $h(j, \hat{y})$ indicate the number of jth n-grams in the generated sequence y or the ground truth $\hat{y}$, respectively. To avoid the drawbacks of using a precision score, namely, it favors shorter generated sentences, BLEU-N introduces a brevity penalty:

$$\text{BLEU-N} := b(y, \hat{y}) \exp\left(\sum_{n=1}^{N} \beta_n \log p_n(y, \hat{y})\right), \tag{19}$$

where $b(y, \hat{y})$ is the brevity penalty and β_n is a weighting parameter. We use corpus-level BLEU-4, i.e., $N = 4$, as our evaluation metric, since it is demonstrated to be more correlated with human judgments than other evaluation metrics [46].

Rouge [44] is a set of metrics used for evaluating the quality of generated sentences. Rouge-N mainly counts the recall rate of N-grams contained in candidate and reference sentences. When the unigram and bigram are extracted from sentences, the corresponding evaluation metric are Rouge-1 and Rouge-2. Rouge-L is a F-measure based on the **Longest Common Subsequence (LCS)** between a candidate and target sentence, where the LCS is a set of words appearing in two sentences in the same order.

METEOR [16] creates an explicit alignment between the candidate and target responses, which is based on exact token matching. Given the alignments, the METEOR score is the harmonic mean of precision and recall between the generated and ground truth texts [46].

4.3 Baseline Approaches

We compare the performance of the proposed CoRe with a random selection approach, a retrieval-based model [47], the basic attentional encoder-decoder model (NMT) [15], a reinforcement learning-based model for pull request summary generation [48], and the state-of-the-art approach for review response generation [27], namely, RRGen. We elaborate on the first and last baselines below.

Random Selection: The approach randomly picks one response in the training set as the response to a review in the test set.

NNGen: NNGen is chosen as one baseline, since it shows better performance than the basic NMT model [39] in producing commit messages for code changes. NNGen computes the cosine similarity between new reviews and the reviews in the training set based on the "bags of words" representations [50]. The **nearest neighbor (NN)** algorithm is adopted to retrieve the most relevant five reviews. NNGen finally regards the response of the training review with the highest BLEU-4 score as the result.

PRSummarizer: It is a sequence-to-sequence model for generating text summarization of the pull requests in software project. PRSummarizer adopts the pointer generator to learn to copy words from the source sequence to cope with out-of-vocabulary words in software artifacts. It also designs a special loss function to bridge the gap between the training loss function and the evaluation metric Rouge.

RRGen: It is the state-of-the-art approach for automating review reply generation. RRGen explicitly combines review attributes, such as review length, rating, predicted sentiment and app category, and occurrences of specific keywords into the basic attentional encoder-decoder (NMT) model.

5 EXPERIMENTAL RESULTS

In this section, we elaborate on the results of the evaluation of CoRe through experiments and compare it with the state-of-the-art tool, RRGen [27], and another competing approach, NMT [15], to assess its capability in accurately responding to user reviews. Our experiments are aimed at answering the following research questions:

RQ1: What is the performance of CoRe in responding to user reviews?
RQ2: What is the impact of the involved contextual knowledge on the performance of CoRe?
RQ3: How accurate is CoRe under different parameter settings?

5.1 RQ1: What Is the Performance of CoRe in Responding to User Reviews?

Table 1 illustrates the comparison results with the baseline approaches. As can be seen, the proposed CoRe shows the best performance among all the approaches. Specifically, CoRe outperforms the baselines by 12.36%∼5.20 times in terms of the BLEU-4 metric. From the p_n scores, we can observe that the responses produced by CoRe consist of more similar n-grams comparing to the ground truth. For example, CoRe increases the performance of the baselines by at least 15.68% with respect to the accuracy of 4-gram prediction. For the Rouge and METEOR scores, CoRe also shows superior performance than all the baselines.

We then use Wilcoxon signed-rank test [70] to verify whether the increase is significant and Cliff'd Delta (or d) to measure the effect size [10]. The significance test result (p-value < 0.01) and large effect size on the metrics ($|d| > 0.474$) of CoRe and RRGen indicate that the proposed model can generate more accurate and relevant responses to user reviews.

Table 1. Comparison Results with Baseline Approaches

Model	BLEU-4	p_1	p_2	p_3	p_4	Rouge-1	Rouge-2	Rouge-L	METEOR
Random	6.55*	27.64*	6.90*	3.55*	2.78*	24.82*	5.81*	22.86*	18.71*
NNGen	14.09*	34.48*	13.85*	9.78*	8.59*	32.49*	12.31*	25.78*	24.93*
PRSummarizer	3.75*	13.05*	4.39*	2.3*	1.48*	18.26*	6.36*	19.22*	23.39*
NMT	21.61*	40.55*	20.75*	16.78*	15.47*	36.13*	15.47*	35.80*	29.27*
RRGen	36.17*	53.24*	35.83*	31.73*	30.04*	45.81*	29.82*	45.49*	42.16*
CoRe	**40.64**	**57.78**	**40.70**	**36.47**	**34.75**	**53.55**	**37.10**	**52.28**	**49.34**

Bold figures highlight better results. p_n indicates the n-gram precision computed in Equation (18). Statistically significant results are indicated with *(p-value < 0.01).

Table 2. Contrastive Experiments with Individual Knowledge Source Removed, where "-Retrieval" and "-Description" Indicate the CoRe with the Retrieved Responses and App Description, Respectively, Removed

Model	BLEU-4	p_1	p_2	p_3	p_4	Rouge-1	Rouge-2	Rouge-L	METEOR
CoRe	**40.64**	**57.78**	**40.70**	**36.47**	**34.75**	**53.55**	**37.10**	**52.28**	**49.34**
-Retrieval	38.65	54.73	37.91	33.71	31.90	52.39	36.12	51.31	49.26
-Description	38.58	54.00	36.71	32.55	30.71	52.36	35.85	51.05	48.54
Only review (NMT)	21.61	40.55	20.75	16.78	15.47	36.13	15.47	35.80	29.27

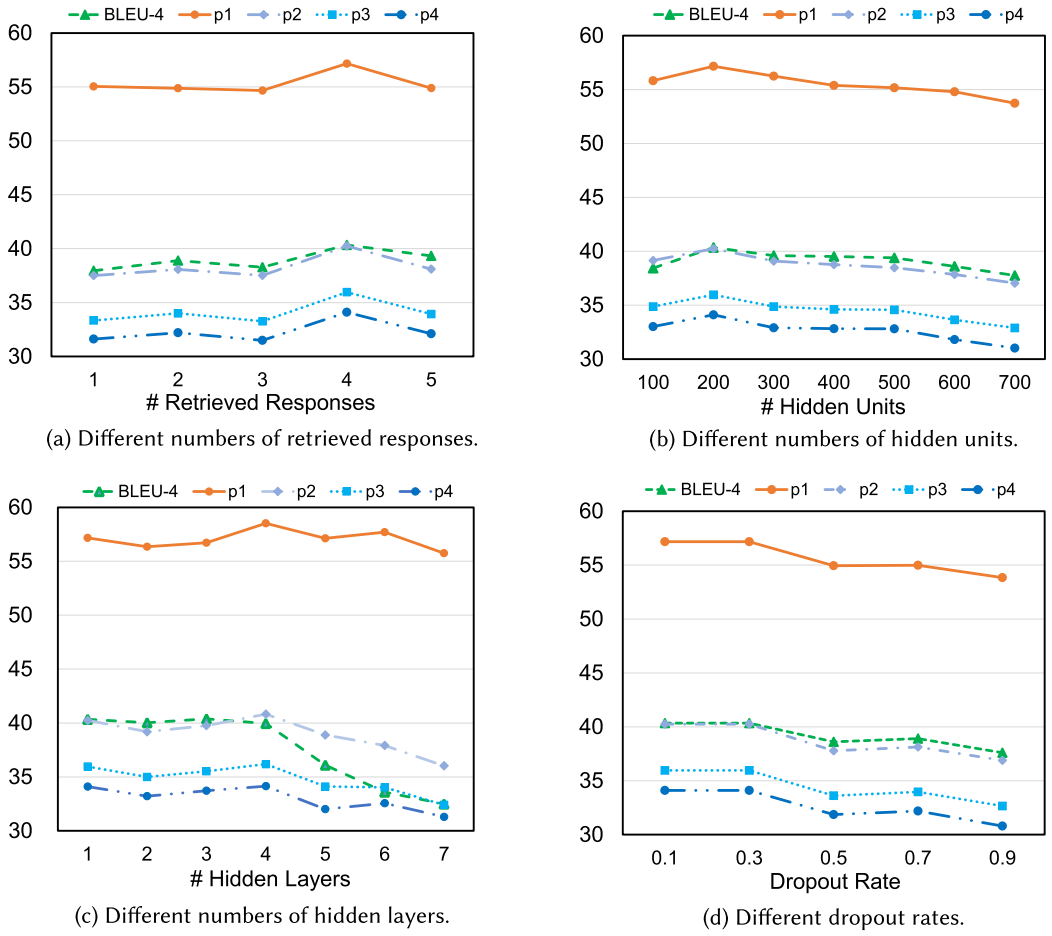
5.2 RQ2: What Is the Impact of the Involved Contextual Knowledge on the Performance of CoRe?

We analyze the impact of the involved contextual knowledge, including app description and the retrieved responses, on the model performance. We perform contrastive experiments in which only a single source of contextual information is considered in the basic attentional encoder-decoder model. Table 2 illustrates the results.

The integration of both app description and the retrieved responses presents the highest improvements. With either type of contextual information individually combined, the model achieves comparative performance, i.e., ~38 and ~54 in terms of BLEU-4 and p_1 scores, respectively. However, without the contextual information included, the performance shows dramatic decline, achieving only 21.61, 35.80, and 29.27 in terms of the BLEU-4, Rouge-L, and METEOR metrics, respectively. This implies the importance of integrating contextual knowledge for accurate review response generation, and each type of the considered contextual knowledge is helpful for improving the generation accuracy. We analyze deeper into the advantage carried by the contextual knowledge in Section 7.1.

5.3 RQ3: How Accurate Is CoRe under Different Parameter Settings?

We also analyze the impact of different parameter settings on the model performance. Specifically, we compare the accuracy of CoRe under varied parameters, including the number of retrieved responses, the number of hidden units, the number of hidden layers, dropout rate, and the dimension of word embeddings. Grid search [42] is a common strategy to determine the optimal values of the hyper-parameters for a given model. Since multiple parameters of CoRe are required to be fine-tuned and each trial consumes ~8 hours (cf., Section 3.3), we initialize the parameters following the prior work cited in Reference [27] to save time for parameter tuning. Specifically, we first set the number of hidden units, word embedding dimension, number of hidden layers, and dropout rate as 200, 100, 1, and 0.1, respectively. Based on these initial parameters, we first conduct grid search to find the optimal number of retrieved responses. With the number of retrieved



(a) Different numbers of retrieved responses.
(b) Different numbers of hidden units.
(c) Different numbers of hidden layers.
(d) Different dropout rates.

Fig. 7. Model performance under different parameter settings.

Table 3. Impact of Different Dimensions of Word Embeddings on the Performance of CoRe

Dimension of Word Embedding	BLEU-4	p_1	p_2	p_3	p_4	Rouge-1	Rouge-2	Rouge-L	METEOR
25	40.15	56.87	39.45	35.16	33.34	52.53	35.97	51.41	48.63
50	38.04	54.13	37.30	33.12	31.31	51.59	35.13	50.72	48.13
100	**40.64**	**57.78**	**40.70**	**36.47**	**34.75**	**53.55**	**37.10**	**52.28**	**49.34**
200	39.09	55.20	38.42	34.13	32.27	52.29	36.18	51.16	48.77

responses determined, we then use grid search to detect the optimal values of the other parameters one-by-one.

Figure 7 and Table 3 show the influence of different parameter settings on the model performance. We observe that the accuracy of the model varies as the parameters change.

 # Retrieved Responses: As can be seen in Figure 7(a), with the number of retrieved responses increasing from 1 to 5, the BLEU-4 score fluctuates slightly, and when the number of retrieved responses is set as 4, CoRe achieves the best performance. This indicates that more retrieved

responses could be helpful for generating more accurate responses. However, since the relevance between the retrieved response and the review reduces as the number of retrieved responses increases, considering too many responses may bring interference to the final output.

 # Hidden Units: As shown in Figure 7(b), more hidden units may not be beneficial for improving accuracy. When the number of hidden units is larger than 200, the model performance exhibits a downward trend. Thus, we define the number of hidden units as 200 during the evaluation.

 # Hidden Layers: Figure 7(c) depicts the variations of the model performance as the number of hidden layers increases. We can observe that the performance of CoRe shows a dramatic decrease when the layer number is greater than 4 in terms of the BLEU-4 score, and variations as the layer number increases from 1 to 4 are not obvious, ranging from 39.96 to 40.34 regarding the BLEU-4 value. With more hidden layers, both model training and testing time will increase; thus, we set the number of hidden layers as 1 during the evaluation.

Dropout Rate: As can be seen in Figure 7(d), as the dropout rate grows, the model accuracy presents a decline trend, which implies that large dropout rates could greatly reduce the knowledge learned by the previous layer, leading to poor generation performance. To reduce the information loss during the forward and backward propagation and avoid overfitting, the dropout rate is set as 0.1.

Dimension of Word Embedding: We compare the model performance under the four different dimensions of word embeddings provided by GloVe [3], and the results are illustrated in Table 3. As can be seen, CoRe achieves the poorest accuracy when the dimension of word embedding equals to 50 and the best when set as 100. The performance decreases as the embedding dimension increases to 200, which indicates that more dimension may not be useful for enhancing the accuracy of the response generation. In this work, we set the dimension of word embeddings as 100.

6 HUMAN EVALUATION

In this section, we conduct human evaluation to further validate the effectiveness of the proposed CoRe. The human evaluation is conducted through online questionnaire. We invite 20 participants totally, including 15 postgraduate students, four undergraduate students, and one senior researcher, all of whom are not co-authors and major in computer science. Among the participants, 12 of them have industrial experience in software development for at least a year. 95% (19/20) of the participants read the user reviews or developers' replies before downloading or updating an app. Besides, 75% (15/20) of them have written at least one app review. The statistics indicate that a majority of the participants are familiar with app reviews. Moreover, for the 58 subject apps, which are from 16 app categories according to Google Play, 95% (19/20) raters have usage experience with apps from more than five of the categories, and 70% (14/20) have used apps from more than 12 app categories, which implies that the participants are likely familiar with the subject apps.

Each participant is invited to read 25 user reviews and judge the quality of the responses generated by CoRe, RRGen, and the official app developers. Each of them will be paid 10 USD upon completing the questionnaire.

6.1 Survey Design

We randomly selected 100 review-response pairs and split them evenly into four groups, where each group consists of 25 review-response pairs. We create an online questionnaire for each group and ensure that each group is assessed by five different participants. In the questionnaire, each question describes one review-response pair, comprising one piece of user review, the developers' response, and its responses generated by RRGen and CoRe. The order of the responses is randomly swapped for each review.

User Review: It is the best photo editing app in google store. I've been using <app> since last year. Bu I still didn't know about saving picture with its actual size. Always the picture is compressed to a smaller size after editing. Is there any way to save picture with high quality? Or else please enable that feature! Thank you <app> team for this awesome app.

Response 1: Hi <user>, thanks for your honest feedback. We do have the option to change the image quality size and it's located in the <app> setting max image size and pick the high res. Have you find it? If no, contact our team at <email> and they will provide a detailed step by step instruction.

Response 2: Hi <user>, thanks for the review. We'd appreciate it if you could contact us at <email> with your issue and some great suggestions so we can improve your future experience with <app>.

Response 3: Hey <user>, thanks for your honest review! You can solve this issue by going to your device's setting about the maximum image size and clicking on the preferring image size. If the problem still continues, please email us at <email>.

Note: *This is a photography app, and the user rating is five stars. In the sentences, the symbols <app>, <user>, <email>, <digit> denote app name, user name, email address and one digit, respectively.*

	Very Dissatisfied				Very Satisfied
Response 1's Relevance	○	○	○	○	○
Response 1's Accuracy	○	○	○	○	○
Response 1's Fluency	○	○	○	○	○
:	:	:	:	:	:

Your Preference Rank of the Three Responses: _____

Fig. 8. An example of questions in our questionnaires. Response 1, 2, and 3 correspond to the developer's response, the response produced by RRGen, and the output of CoRe, respectively. The two-dot symbols indicate the simplified rating schemes for Response 2 and 3.

Following Reference [27], the quality of the responses is evaluated from three aspects, including "*grammatical fluency*," "*relevance*," and "*accuracy*." We explained the three aspects at the beginning of each questionnaire: The metric "*grammatical fluency*" measures the degree of the readability of the response; the metric "*relevance*" estimates the extent of semantic relevance between the user review and response; and the metric "*accuracy*" relates to the extent of the response accurately replying to the review. All the three aspects are scored based on 1–5 scale (1 for completely not satisfying the rating scheme and 5 for fully satisfying the rating scheme). Besides the three aspects, each participant is asked to rank the three responses based on their overall preference. The "*preference rank*" score is evaluated on 1–3 scale (1 for the most preferred and 3 for the least preferred). Figure 8 shows a sample question in our questionnaire. The participants are not aware of which response is written by developers or which one is generated by which model. They are asked to complete the online questionnaires separately.

6.2 Results

We finally received 500 sets of scores totally and five sets of scores for each review-response pair from the human evaluation. Each set contains scores regarding the four metrics, including "*grammatical fluency*," "*relevance*," "*accuracy*," and "*preference rank*," for the responses of CoRe, RRGen, and official developers. The participants spent 1.72 hours to complete the questionnaire on average, with the median completion time of 1.40 hours. We compute the agreement rate on the four aspects given by the participants, illustrated in Figure 9. As can be seen, 78.3%, 74.0%, 72.7%, and 65.0% of the total 100 review-response pairs received at lease three identical scores regarding the "*grammatical fluency*," "*relevance*," "*accuracy*," and "*preference rank*" metrics, respectively. Besides, 7.3%, 6.7%, 8.3%, and 10.0% of the pairs are rated with consistent scores from the five annotators in

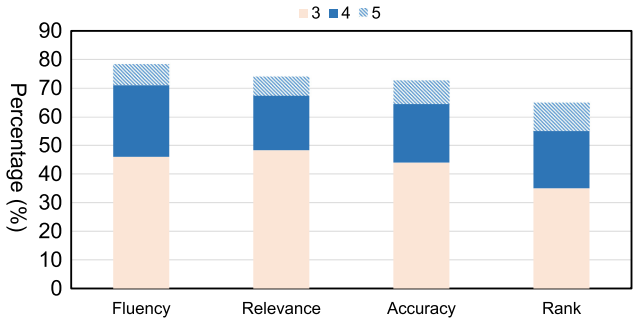


Fig. 9. Agreement rate among the participants in the human evaluation. The horizontal axis and vertical axis indicate different evaluation metrics, and the percentages of 3/4/5 participants giving the same scores, respectively.

Table 4. Comparison Results Based on Human Evaluation

	Grammatical Fluency	Relevance	Accuracy	Preference Rank
RRGen	3.58*	2.93*	2.89*	2.59*
CoRe	4.19	4.06	4.00	1.79
Developer	**4.32**	**4.56**	**4.03**	**1.60**

Average scores are computed and **bold** indicates top scores. Two-tailed t-test results between CoRe and RRGen are indicated with *(p-value < 0.01).

terms of the respective metrics. This indicates that the participants achieved reasonable agreement on the quality of the generated responses.

Table 4 and Figure 10 depict the results of human evaluation. As can be seen, the responses from official developers received the best scores from the participants among all the three responses and with respect to all the metrics. In terms of grammatical fluency, the average scores of the response generated by CoRe and the developers' response are rather close, i.e., 4.19 and 4.32, respectively. As shown in Figure 10(a), most participants gave the responses generated by RRGen a 3-star rating, while CoRe receives more 4/5-star ratings. This indicates that CoRe can produce more grammatically fluent responses than RRGen. Regarding the relevance, the responses generated by RRGen are rated much poorer than those output by CoRe. Combined with Figure 10(b), we can observe that the more than half (62.5%) of the participants enter ratings lower than 4 for the responses generated by RRGen, and the number of 4/5-star ratings for the responses produced by CoRe is 1.15 times than those for the responses of RRGen. Developers' responses receive the most 5-star ratings compared to the generated responses. This implies that the responses output by CoRe tends to be more relevant to the reviews than those generated by RRGen. In terms of the "*accuracy*" metric, we find that the average scores for the responses output by CoRe and the developer's responses are much close, i.e., 4.00 and 4.03, respectively. As illustrated in Figure 10(c), the responses generated by CoRe received slightly more 4/5-star ratings than the developers' responses (391 vs. 384), and 1.22 times than the responses generated by RRGen (176). The results demonstrate that CoRe can produce accurate responses to the user reviews, which is also reflected in the distributions of the "*preference rank*" scores, as shown in Figure 10(d). We discover that most participants rank the responses output by RRGen as the least preferred (69.6%) and the developers' responses as the most favored (53.0%). Also, the responses of CoRe present similar preference score as the developers'

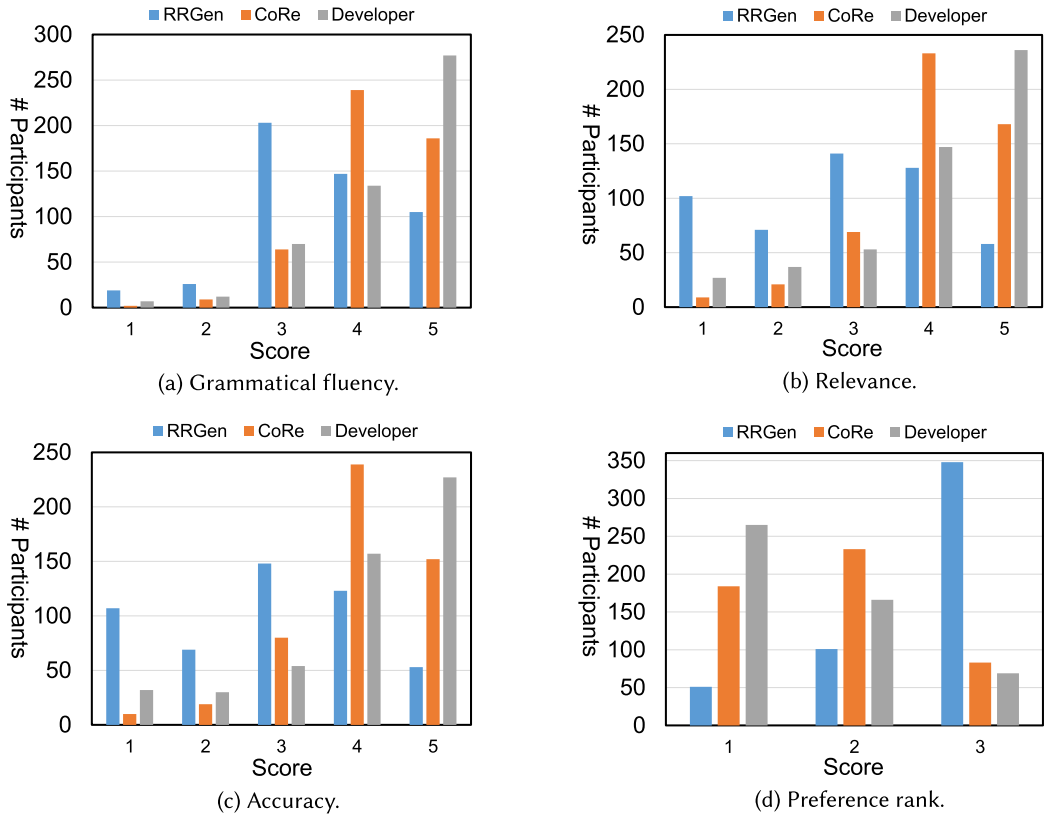


Fig. 10. Human evaluation results. For the metrics "*grammatical fluency*," "*relevance*," and "*accuracy*," the higher the scores, the better; while for the metric "*preference rank*," the lower the scores, the better. The vertical axis indicates the number of participants giving the scores.

responses on average, i.e., 1.79 vs. 1.60 (as shown in Table 4). The human study further validates the effectiveness of the proposed CoRe for review response generation.

7 DISCUSSION

In this section, we discuss the advantages and limitations of our approach, as well as threats to the validity of our findings.

7.1 Why Does Our Model Work?

We have conducted a deep analysis on the advantages of combining app descriptions and retrieved responses for review response generation in CoRe.

7.1.1 App Descriptions. App descriptions generally contain keywords related to main app features, aiming at convincing users to download the apps and facilitating user search through app stores. By considering app descriptions, CoRe can recognize the topics/functionalities discussed by users more accurately. For example, it can learn that the review "*It lose your full charge.*" is related to the "*power save mode*" in the app and generate response providing the solution "*trying different save mode*," as shown in Figure 11, while the response generated by RRGen is rather in

User Review: It's fake app. Don't download it, friend. It lost your full charge. It finished your full charge in <digit> minutes.

Developer's Response: Hi, thanks for your feedback. Our test and feedback from many users show that this product is usually quite helpful and we do believe it can help you as well. We recommend you try different power save modes to find the one that fits you best. If there's still problem, you can email us at <email>, we are ready to help you any way!

RRGen: Hi <user>, this certainly sounds like a frustrate experience. We want to look into this issue for you. Please send a quick note to <url> contact so we can connect.

CoRe: Hi, thanks for your feedback. Our test and feedback from many users show the product is helpful generally. We do believe it can help you as well. We recommend you to try different save modes to find the one that fits you best. If there's still problem, you can email us at <email>, we are ready to help you any way we can!

Retrieved Response-1: Hi, we're sorry to hear you didn't like du battery saver. We're always look to improve so if you have any suggestions, you're welcome to send them to me at <email> and we can discuss them. Thanks!

Retrieved Response-2: Hi, thanks for downloading and support. Any questions or requests, we will be ready to help. Look forward to <user> five-star rating and wish you a nice day!

App Description: ..... With Du Battery Saver's smart preset battery power management mode, you can solve battery problem and extend your battery life. Du Battery Saver is the simplest and easiest way to keep your android phone work well when you need it, and protects against poor charge, battery hog apps, and overlooks device setting that shorten your battery life .

Fig. 11. A user review with the generated response where CoRe can generate responses based on the app description. The fonts in red are indicative of the partial topical words in corresponding texts. We only illustrate the responses of the top two retrieved reviews here for saving space.

general purpose and not topically relevant to the review. Figure 12 visualizes the latent alignment over the user review/app description based on the attention weights α_{tj} from Equation (1) and $\alpha_t^{(r)}$ from Equation (9), respectively. Each column indicates the word distribution over the user review/app description during response generation, which implies the importance of the words in the user review/app description when generating the target word in the response. We can observe the obvious correlation between the word "*mode*" (in the app description) and "*save mode*" (in the response), and relatively weak correlations between "*charge*"/"*minute*" (in the review) and "*save mode*" (in the response). This illustrates that CoRe can build implicit relations between the topical words in app descriptions and corresponding responses, which can help generate relevant and accurate response given a review.

7.1.2 Retrieved Responses. NMT-based approaches tend to prefer high-frequency words in the corpus, and the generated responses are often generic and not informative [14, 73, 76]. For example, they may fail for the responses containing low-frequency words. In our experiment, we find that 51,364/309,246 (16.61%) responses in the corpus contain low-frequency words (frequency ≤ 100). Since similar reviews based on IR-based methods are generally related to the same semantics, their responses could be semantically related and the words in the expected responses (including the low-frequency ones) are also highly probable to appear in them. For example, for the review in Figure 13, we retrieve most similar reviews with respect the semantics (i.e., tf-idf representations in the article) from the training corpus. We only present the responses of the top two similar reviews here for saving space. We can see that the low-frequency words "*localize*" and "*rss*" (which is an abbreviation of Really Simple Syndication, a web feed that allows users to access updates of websites in a standardized format) also appears in the retrieved

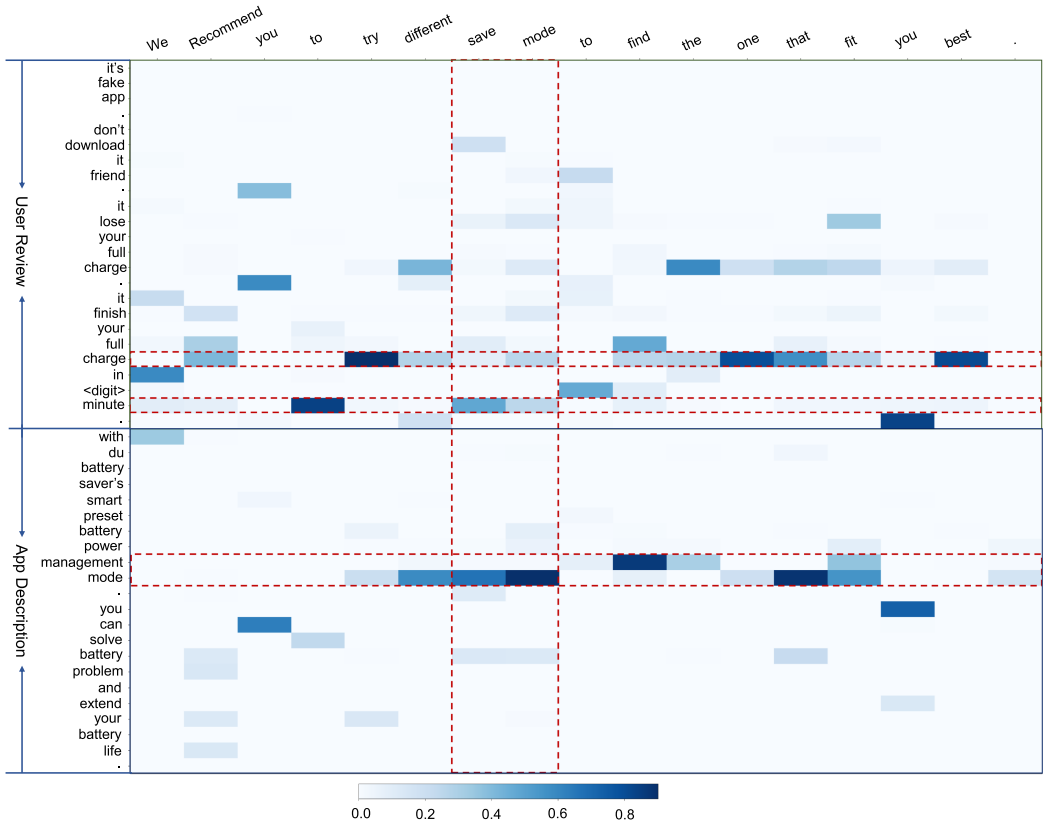


Fig. 12. A heatmap representing the alignment between the user review (left-top)/app description (left-bottom) and generated response by CoRe (top). The columns represent the distribution over the user review/app description after generating each word in the response. Darker colors indicate higher attention weights and manifest a stronger correlation between the target word and source word. Red dotted rectangles highlight partial topical words in corresponding texts.

response (i.e., Retrieved response-2). The words are ignored by RRGen but correctly predicted by CoRe, since they appear in the retrieved responses and are effectively captured during attention fusion (Section 3.2). In contrast, the response generated by RRGen is topically irrelevant to the review, supposing the review is talking about "*ads*." This exhibits that the retrieved responses in CoRe are helpful for generating the responses with low-frequency words.

7.2 Automating the Combination of App Descriptions in CoRe

In CoRe, the input app descriptions require manual filtering to remove the sentences irrelevant to the app functionalities. To render CoRe a fully end-to-end framework, we propose to automatically extract keywords from app descriptions for replacing the manual filtering process. Specifically, we adopt the RAKE algorithm [58] for automatic keywords extraction, with detailed steps shown in Algorithm 1. RAKE first collects a set of candidate keywords at phrase delimiters and stop word position and builds a graph of word co-occurrence for the candidate keywords (Lines 2 to 3). RAKE ranks the candidate keywords based on the degree and frequency of the vertices in the graph (Lines 4 to 9). An example of the process is shown in Figure 14. We then replace the app

User Review: Please add Hindi language.

Developer's Response: Thanks for your feedback. We are working to localize for more region. In the meantime, Please use search or add rss url to connect to your prefer content. If you have any more questions, please contact us within the <app> or email <email>.

RRGen: Thanks for your feedback. Sorry to hear that you are unhappy with the ads . We would like to hear more about this issue. Please contact us within the <app> or email <email> and our team will be happy to address this issue and hopefully resolve it for you.

CoRe: Thanks for your feedback. We are working to localize for more regions. In the meantime, please use search or add rss url to connect to your prefer content. If you have any other questions or comments, please email <email>, thanks!

Retrieved Response-1: Hi <user>! We don't offer the <app> in Hindi at this time. Very sorry!

Retrieved Response-2: We are working to localize content guide for more regions. In the meantime, please use search or add rss url to connect to your prefer content. Thanks for feedback.

App Description: Discover quality content for all your interest from personalize news, entertainment, tech and sport article to trend lifestyle magazine. .....Our editor blends expert voice and curated source, and even recommends story they know you'll like , so you can sit back and flip through what matter most to you. You can even customize your personalize newsstand pick your passion and create a smart magazine for each topic fee.

Fig. 13. A user review with the generated response where CoRe can generate responses with low-frequency words. The fonts in red are indicative of the low-frequency words (frequency $\leq$ 100), and the double-underlined words mean they are topically irrelevant to the user review. Responses of the retrieved top two reviews and the app description are also illustrated.

ALGORITHM 1: RAKE Algorithm.

Data: App description d.
Result: Extracted keywords.

Split d into an array of words $arr(d)$;
Split d into a set of candidate keywords at phrase delimiters and stop word position, denoted as $\Lambda = (\lambda_1, \lambda_2, \ldots)$;
Build a graph of word co-occurrence for the candidate keyword set Λ;
for w *in* $arr(d)$ **do**
    Compute the frequency $freq(w)$ and degree $deg(w)$, where $freq(w)$ is the number of candidate keywords containing w and $deg(w)$ is the total number of words in the candidate keywords containing w;
end
for λ *in* Λ **do**
    Compute the ranking score $s_\lambda = \sum_{w \in \lambda} \frac{deg(w)}{freq(w)}$;
end
Return the first third of the ranked keyword candidates.

description sentences d with the corresponding extracted keywords as the input of CoRe during experimentation and denote the new approach as $\text{CoRe}_{\text{rake}}$. The results are shown in Table 5. As can be found, with only the keywords extracted from descriptions considered, $\text{CoRe}_{\text{rake}}$ already outperforms RRGen in terms of all evaluation metrics. Although the results of $\text{CoRe}_{\text{rake}}$ are lower than those of CoRe, they are rather close with respect to the Rouge-1 and METEOR scores. The

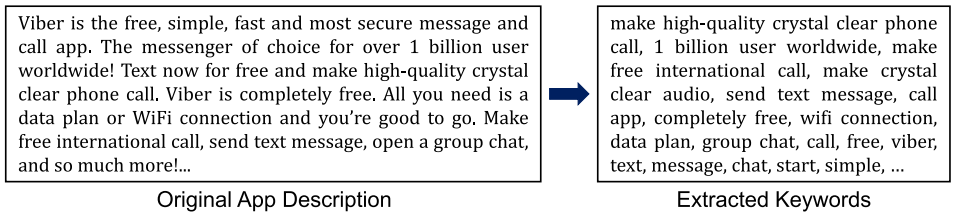


Fig. 14. Example of the extracted keywords for the Viber app using RAKE [58].

Table 5. Results of CoRe with the Keywords Automatically Extracted from App Descriptions

Model	BLEU-4	p_1	p_2	p_3	p_4	Rouge-1	Rouge-2	Rouge-L	METEOR
RRGen	36.17	53.24	35.83	31.73	30.04	45.81	29.82	45.49	42.16
$\text{CoRe}_{\text{rake}}$-Retrieval	37.31	53.17	36.43	32.33	30.55	50.97	34.80	50.17	47.21
$\text{CoRe}_{\text{rake}}$	39.80	56.23	38.99	34.76	32.91	53.23	36.36	51.86	49.12
CoRe	**40.64**	**57.78**	**40.70**	**36.47**	**34.75**	**53.55**	**37.10**	**52.28**	**49.34**

"$\text{CoRe}_{\text{rake}}$" indicates CoRe with the keywords extracted from app descriptions as input, and "$\text{CoRe}_{\text{rake}}$-Retrieval" means $\text{CoRe}_{\text{rake}}$ without the retrieved responses considered, i.e., only considering the extracted keywords.

Table 6. Results of CoRe with Different Similarity Measurement Approaches Involved During Retrieving Similar Reviews

Model	BLEU-4	p_1	p_2	p_3	p_4	Rouge-1	Rouge-2	Rouge-L	METEOR
RRGen	36.17	53.24	35.83	31.73	30.04	45.81	29.82	45.49	42.16
$\text{CoRe}_{\text{GloVe}}$	39.42	55.46	38.55	34.46	32.76	53.15	36.86	52.07	49.34
$\text{CoRe}_{\text{BERT}}$	39.60	55.46	38.89	34.72	32.86	**53.63**	**37.52**	**52.69**	**50.09**
CoRe	**40.64**	**57.78**	**40.70**	**36.47**	**34.75**	53.55	37.10	52.28	49.34

"$\text{CoRe}_{\text{GloVe}}$" and "$\text{CoRe}_{\text{BERT}}$" indicate the CoRe with GloVe-based and BERT-based similarity measurement approaches, respectively.

results indicate the effectiveness of the keywords that are automatically extracted from app descriptions for the review response generation task, and we will further enhance the keyword extraction component in the future.

7.3 Analysis of Different Retrieval Approaches

During retrieval of semantically similar reviews, we adopt the unigram tf-idf based similarity assessment method, since it has been proven effective for other text retrieval tasks [47, 74]. However, the tf-idf representations are based on word frequencies, which may not well capture the semantics of low-frequency words. Thus, we also explore more advanced similarity measurement approaches, namely, word embeddings [52] and BERT [22]. Specifically, for the word embedding-based approach, we compute the average 50-dimensional GloVe vectors [3] of the words in each review as the review representation; while for the BERT-based approach, we first use the pre-trained BERT to obtain 768-dimensional vectors of the reviews and then reduce the dimension to 50 by using the **PCA (principle component analysis)** method [66]. We define the dimension as 50 to save the time needed for computing millions of cosine similarities between the review representations. For each approach, we take responses of the top 4 similar reviews as the input of CoRe. The results are shown in Table 6. We observe that for all similarity measurement approaches, CoRe outperforms the state-of-the-art baseline, RRGen. BERT-based approach shows better performance than the GloVe-based approach with respect to all the evaluation metrics,

Table 7. Percentage Analysis for the Reviews of which the Similarity Scores with the Corresponding Most Similar Reviews in the Training Dataset Are below a Similarity Threshold

Dataset	Similarity Threshold								
	<0.1	<0.2	<0.3	<0.4	<0.5	<0.6	<0.7	<0.8	<0.9
Training	2.02%	2.43%	6.83%	22.70%	39.10%	50.05%	57.30%	62.95%	67.52%
Validation	1.88%	1.90%	4.50%	19.30%	35.80%	47.44%	59.98%	65.43%	70.06%
Test	2.12%	2.16%	4.85%	18.90%	35.60%	47.35%	59.95%	65.36%	69.59%

and even better than the tf-idf based approach considering the Rouge and METEOR scores. This indicates that the similarity assessment method can impact the model performance, and BERT-based method has an edge over the GloVe-based method. We will explore how to better retrieve similar reviews for more accurate response generation in the future work.

7.4 Analysis of the Retrieved Responses

Due to the limited apps considered, the retrieved reviews may be irrelevant to the input reviews, in which case the incorporated responses of CoRe could introduce bias into the corresponding generated results. We analyze the similarity scores between the retrieved most similar reviews and the given reviews, as shown in Table 7. Typically, values of cosine similarity greater than 0.5 mean that the examined vector representations have strong semantic relevancy [56]. According to Table 7, more than 60% of the reviews possess similarity scores larger than 0.5 with the most similar retrieved reviews, and only a small percentage of the reviews have similarity scores less than 0.1 on the training, validation, and test sets, respectively. We then conduct performance analysis on the CoRe considering various similarity scores between the input reviews and retrieved reviews. Specifically, we exclude the retrieved responses from the input of CoRe if the largest similarity scores are lower than specific thresholds, i.e., only the app description is considered for response generation. The results are shown in Table 8. We can observe that removing highly irrelevant responses, e.g., with similarity values less than 0.2, results in better performance than removing more relevant responses, e.g., with similarity scores less than 0.9. The results demonstrate that incorporating relevant responses can benefit the model performance. However, excluding the responses with similarity scores less than 0.1 does not boost performance. And still, involving the responses without considering the similarity values achieves the best performance, which indicates that the retrieved responses are helpful for the task despite the low relevancy. The phenomenon may be because the retrieved responses could provide other hints such as the patterns or sentiment of the generated responses besides topics/semantics of the given reviews.

7.5 Dataset Partition

In this article, we split dataset into training, validation, and test sets by following the prior study on review response generation [27], i.e., splitting the dataset by review instead of by app. The dataset partition strategy might cause information leakage from the training set apps into the validation or test sets. Thus, we also evaluate the performance of CoRe based on the dataset split by app instead of by review. Specifically, we randomly select 80%, 10%, and 10% of the studied apps as training set, validation set, and test set, respectively. To ensure the reliability of the results, the random selection process is conducted three times, and the average is computed. We also compare CoRe with the best retrieval-based approach NNGen and the best generation-based approach RRGen. The comparison results are shown in Table 9. We find that CoRe outperforms both baseline models, and the performance of all the models decreases significantly, comparing to the performance when splitting dataset by review. A reduced performance is reasonable, since less knowledge of the apps

Table 8. Results of CoRe with the Retrieved Responses Excluded when Their Largest Similarity Scores Are below a Threshold

Threshold	BLEU-4	p_1	p_2	p_3	p_4	Rouge-1	Rouge-2	Rouge-L	METEOR
0.0	**40.64**	**57.78**	**40.70**	**36.47**	**34.75**	**53.55**	**37.10**	**52.28**	**49.34**
0.1	40.50	57.69	40.57	36.33	34.60	53.05	36.67	51.68	49.24
0.2	40.50	57.70	40.57	36.33	34.60	53.05	36.68	51.68	49.24
0.3	40.43	57.67	40.56	36.32	34.60	52.97	36.62	51.62	49.15
0.4	40.14	57.42	40.39	36.19	34.47	52.49	36.30	51.18	48.65
0.5	39.79	56.82	39.89	35.75	34.04	51.89	35.89	50.66	48.19
0.6	39.60	56.41	39.55	35.45	33.74	51.53	35.67	50.38	47.94
0.7	39.47	56.14	39.30	35.22	33.51	51.32	35.51	50.21	47.78
0.8	39.45	55.98	39.16	35.08	33.36	51.22	35.44	50.13	47.73
0.9	39.38	55.85	39.03	34.95	33.22	51.13	35.36	50.05	47.66

The threshold "0.0" corresponds to CoRe with no retrieved responses removed.

Table 9. Comparison Results of Different Approaches Based on the Dataset Split by App

Model	BLEU-4	p_1	p_2	p_3	p_4	Rouge-1	Rouge-2	Rouge-L	METEOR
NNGen	3.07	31.57	6.76	1.78	0.52	28.33	6.11	19.80	18.32
RRGen	3.46	24.58	5.12	1.45	0.79	27.59	5.98	26.91	22.52
CoRe	7.14	30.06	9.39	4.84	2.80	29.58	9.94	27.53	24.51

in the test set will be learned during training. The phenomenon indicates that generating responses for the reviews of unknown apps is more challenging, and we will try to tackle the challenge in future work.

7.6 Study on the Impact of Trivial Responses

Trivial responses such as "Thank you" are not informative for users, and it makes little sense to learn from or produce such responses through machine learning methods. Similar to Reference [47], we manually derive some common patterns of trivial responses by skimming the responses in the benchmark dataset. Table 10 presents our trivial response patterns. The proportions of the identified trivial responses are 7,614 (2.72%), 415 (2.82%), and 398 (2.70%) in the training, validation, and test sets, respectively. After removing the trivial responses, the performance of CoRe and the baseline approaches are shown in Table 11. We choose the best retrieval-based model, NNGen, and the best generation model, RRGen, as the baselines. We can find that removing the trivial responses slightly reduces CoRe's performance but increases the results of NNGen and RRGen in terms of the Rouge and METEOR metrics. With respect to the BLEU-4 scores, although the performance of RRGen and CoRe decreases, the decreasing degree is marginal. The phenomenon may be attributed to the small percentage of the trivial responses in the benchmark dataset. Still, among all the models, CoRe achieves the best results considering all evaluation metrics, which further indicates the effectiveness of the proposed approach.

7.7 Analysis of Using BERT as Word Representations

In the work, we initiate the word embeddings with pre-trained GloVe vectors [3] in CoRe, which may not be the optimal choice. We further analyze the model performance when using more advanced pre-trianed model BERT [22]. We first adopt the original pre-trained BERT model [1] to obtain a 768-dimensional vector for each word in the vocabulary and then utilize PCA to reduce the dimension to 25, 50, 100, and 200 for fair comparison with the GloVe vectors. The results are shown in Table 12. We can find that CoRe achieves different performance with different dimensions and

Table 10. Our Trivial Response Patterns

you are [always] welcome [<user>]
why? [what is wrong?]
we'll fix this [<user>]
thank so much for your review [<user>]
thank you [<user>]
hello [<user>], any problem?
we be glad you be enjoy the <app>
thank [<user>/you] for [your] review/suggestion/comment/feedback

"[]" Means Optional, "/" Refers to "or," "<user>" Indicates the User Name, and "<app>" Is the App Name.

Table 11. Results on the Dataset with the Trivial Responses Removed

Threshold	BLEU-4	p_1	p_2	p_3	p_4	Rouge-1	Rouge-2	Rouge-L	METEOR
NNGen	14.91 (↑ 0.82)	36.02 (↑ 1.54)	14.71 (↑ 0.86)	10.41 (↑ 0.63)	9.19 (↑ 0.60)	34.38 (↑ 1.89)	13.17 (↑ 0.86)	27.00 (↑ 1.22)	25.99 (↑ 1.06)
RRGen	35.99 (↓ 0.18)	53.84 (↑ 0.60)	35.73 (↓ 0.10)	31.56 (↓ 0.17)	29.88 (↓ 0.16)	48.41 (↑ 2.60)	30.96 (↑ 1.14)	47.54 (↑ 2.05)	43.41 (↑ 1.25)
CoRe	**40.32** (↓ 0.32)	**56.25** (↓ 1.53)	**39.50** (↓ 1.20)	**35.41** (↓ 1.06)	**33.60** (↓ 1.15)	**53.39** (↓ 0.16)	**36.88** (↓ 0.22)	**52.12** (↓ 0.16)	**49.37** (↑ 0.03)

The value inside each bracket indicates the change compared to using the whole responses.

Table 12. Results of Using BERT to Initialize the Word Embeddings

Dimension	BLEU-4	p_1	p_2	p_3	p_4	Rouge-1	Rouge-2	Rouge-L	METEOR
25	39.01	55.35	38.23	34.02	32.17	52.46	35.87	51.21	48.65
50	**39.93**	**56.24**	**39.15**	**34.88**	**33.09**	53.00	36.38	51.85	48.63
100	39.23	55.19	38.54	34.30	32.48	**53.48**	**37.34**	**52.41**	**49.75**
200	39.68	55.55	38.91	34.77	33.00	53.35	37.22	52.31	49.47

shows relatively more promising results when the dimension is set to 50 or 100. Comparing with the word embedding-based word representations, BERT-based approach (100-dimension) achieves slightly better results in terms of the Rouge and METEOR metrics, but performs worse in terms of the BLEU-4 score. Thus, we suppose that BERT-based word representations are also helpful for accurate review response generation, but its advantage is not large as compared to the word embedding-based representations.

7.8 Analysis of the Vocabulary Size

In this work, we directly follow the defined vocabulary size in Reference [27] for fair comparison, but the parameter could also impact the model performance [45]. So, we analyze the model performance when the considered vocabulary size ranges from 2,000 to 14,000 for the task. As shown in Table 13, we can observe that CoRe could achieve better performance in terms of the BLEU-4, Rouge and METEOR metrics, with a relatively small vocabulary size such as 4,000 or 6,000. This may be because the top 4,000 words already cover a significant proportion (~99.17%) in terms of the word frequencies, in spite of only accounting for a small percentage of the whole vocabulary size (6.49%). With only the top 2,000 words involved, CoRe performs much worse

Table 13. Influence of the Vocabulary Size on the Model Performance

Vocab Size	BLEU-4	p_1	p_2	p_3	p_4	Rouge-1	Rouge-2	Rouge-L	METEOR
2,000	39.33	55.71	38.55	34.31	32.47	52.64	36.11	51.43	48.60
4,000	**41.06**	57.17	40.33	36.06	34.19	53.93	37.73	52.96	49.91
6,000	40.83	56.82	40.10	35.92	33.98	**54.21**	**37.84**	**53.33**	**50.38**
8,000	40.61	57.11	39.99	35.65	33.73	53.28	36.80	52.34	49.18
10,000	40.64	**57.78**	**40.70**	**36.47**	**34.75**	53.55	37.10	52.28	49.34
12,000	40.26	56.42	39.50	35.28	33.42	53.63	37.33	52.41	49.80
14,000	40.34	56.16	39.58	35.44	33.60	53.95	37.74	52.90	50.34

than CoRe involving more top words, which indicates the importance of considering vocabulary of sufficient size for the task.

7.9 Analysis of the Incorporation of Different Input Sources

CoRe can automatically learn the degree of the three input sources, i.e., user reviews, app descriptions, and retrieved responses, are considered. According to Equation (15), CoRe captures the incorporation weights of the vocabulary distributions from app descriptions d and retrieved responses r through the parameter γ_t at a decoder step t. CoRe also obtains the combined degrees of the user reviews via the parameter θ_t in Equation (16). We use an example, as shown in Figure 15, to elaborate the capacity of CoRe in adaptively combining the three input sources. We can observe that some words in the response output by CoRe are from the app description and some from the retrieved responses, with the words highlighted in Figure 15(a). We then normalize and visualize the attention distribution weights for the review, app description, and retrieved responses, as illustrated in Figure 15(b). We can find that the three input sources contribute to the generation of the response words in varied degrees. For example, the app description plays the greatest role in generating the words "*local*" and "*international*"; while the retrieved responses help most to produce the words "*homescreen*" and "*arrow*." The example indicates that CoRe can integrate the three input sources with differing weights during the generation of responses.

7.10 Analysis of the Human Evaluation Results

The human evaluation results show that the responses generated by CoRe are more preferred by the participants than the developers' responses. We find that the responses produced by CoRe receive better ratings from the participants than developers' responses for 34, 34, 35, and 35 of the studied reviews in terms of the average "*grammatical fluency*," "*relevance*," "*accuracy*," and "*preference rank*" metric values, respectively. The first and second authors then perform an elaborative analysis of the cases and discover that CoRe's responses describe more details or more concrete topics than the corresponding developers' responses. Figure 16 depicts two examples. In the first example, we can find that CoRe's response is clearly related to the user-concerned topic "*watermark*," while the developer's response does not address the user issue concretely and seems like a bot message. For the second example, CoRe's response provides an explanation about how the user can provide more detailed information, i.e., by providing a "*screenshot*," while the developer's response simply asks the user to forward more details without any further suggestion. The finding also inspires us to generate responses containing more detailed descriptions in the future work.

7.11 Analysis of the Defined Maximum Sequence Length

In this work, we define the maximum sequence length for reviews and responses as 200 following the prior work cited in Reference [27]. However, the maximum sequence length setting could

User Review: Sleazy news and naked celebrity aggregate when star magazine just doesn't show enough skin.

Developer's Response: sorry to hear you're dissatisfied with our content. As a news aggregator our news providers are local and international and treat topic in various ways and have their own views. You can dislike this content category by that small three dot icon under article title on news fee list and this kind of content won't appear anymore .

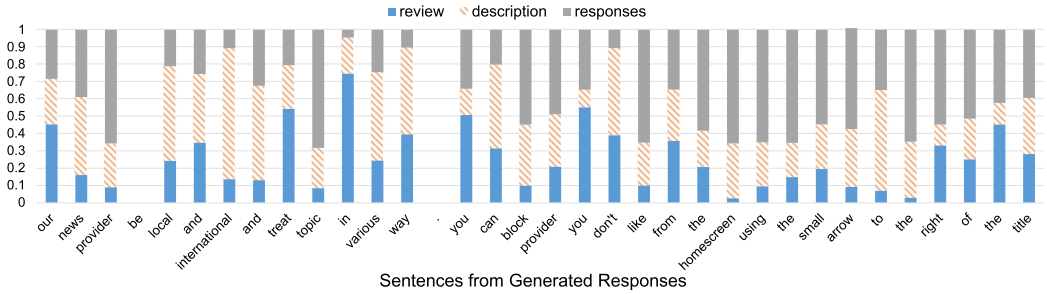
CoRe: Hi <user> thank for your review and support! Our news providers are local and international and treat topic in various way. You can also block provider you don't like from the homescreen using the small arrow to the right of the title. Thank for your feedback, hope this is helpful.

Retrieved Response-1: Hi <user> sorry you didn't like the homescreen. To only get news you be interest in you can add or remove topic you like to the my topic screen. You can also block provider you don't like from the homescreen using the small arrow to the right of the title. thank for your feedback hope this be helpful news republic.

Retrieved Response-2: Hi there thanks for your review and support! We'd like to invite you to join our exclusive user community to get early access to new feature and give us feedback.

App Description: News republic a news app power by artificial intelligence, deliver daily break news, world local news, and article directly to you! Receive international and local news for free! News republic provides the latest news and complete article from 500 authorize local and international source. easy to customize to give you access to the article of the most famous source!.....

(a) A user review and the generated response by CoRe. The corresponding top two retrieved responses and app description are also illustrated. The words highlighted in red indicate partial words appearing in both app description/retrieved responses and the review.



(b) The normalized attention distribution weights among the three input sources based on the learnt γ_t in Equ. (15) and θ_t in Equ. (16).

Fig. 15. An example for illustrating the capability of CoRe in incorporating the three input sources differently.

impact the performance of the basic NMT model [15]. Generally, the maximum sequence lengths for both source and target sequences are set between 50 to 100 [15, 39, 60]. To analyze the influence of different sequence lengths on model performance, we set the maximum sequence length at 100 tokens for the target responses, since 100 is the largest preferred maximum length used by NMT in natural language translation [39] and all the responses in the dataset have fewer than 100 tokens. For the review sequences, we choose two settings: one is 200 and another is 100. We pick these two settings because 100% and 98.9% reviews in the dataset are with fewer than 200 and 100 tokens, respectively. The results are illustrated in Table 14. We can find that the maximum sequence lengths can impact the model performance. For example, with the maximum sequence lengths set at 100 for both reviews and responses, the BLEU-4 score shows a slight increase compared to setting

User Review: It used to be great but now they have this annoying watermark! The best part of this <app> is that there is no watermark! Now you have to download a whole other <app> and open it to get rid of it but only for a limited time! At least you can remove it for a while which is why it isn't one star yet.

Developer's Response: Hello, if you enjoy using <app>, would you mind giving us <digit> star rating? It will encourage us to make the <app> better, thank you for supporting! <app> holds two online contest. Welcome to show yourself with creative idea! Top winner of each contest will earn cash prize.

CoRe: Hello, you can remove the watermark for free just by downloading sponsor's <app> or you can buy our pro version. Our team is supported by advertisement. Hope you understand, thank you!

(a) The first example.

User Review: It has a little problem with mi note <digit> handset.

Developer's Response: We want you to know that we are working hard to improve <app> for you. Please forward us more details about your issue to <email>. The more you describe, the better the support we will give to you. Hope to hear from you soon! Thank you very much!

CoRe: Hi, sorry for the inconvenience. Could you please describe your problem more specifically such as a screenshot and send them to <email>? That will help us to locate your issue better. Thanks very much!

(b) The second example.

Fig. 16. Two examples about the responses generated by CoRe are rated higher than the developers' responses during the human evaluation.

Table 14. Influence of the Maximum Sequence Length Settings on the Model Performance

Review Seq. Length	Response Seq. Length	BLEU-4	p_1	p_2	p_3	p_4	Rouge-1	Rouge-2	Rouge-L	METEOR
100	100	**40.73**	56.59	39.86	35.83	34.06	53.26	**37.18**	51.91	49.28
200	100	39.75	55.98	38.95	34.77	32.93	53.09	36.56	51.75	49.26
200	200	40.64	**57.78**	**40.70**	**36.47**	**34.75**	**53.55**	37.10	**52.28**	**49.34**

both parameters as 200. Since many configurations of the parameter are possible, optimizing the maximum review/response lengths for generating responses is an area of future work.

7.12 Analysis on the Impact of Contextual Information on the Generated Responses

According to Table 2, CoRe with only the app description achieves similar performance as CoRe with only the retrieved responses. To investigate further the difference between the responses produced by the two approaches, the first two authors manually check the generated responses by the two approaches. We find that CoRe with only the app description tends to integrate simple reference of the app functionalities in the generated responses, while CoRe with only the retrieved responses could generate responses with similar patterns as the retrieved ones. An example is illustrated in Figure 17. We can observe that by combining the app description, the generated response contains sentences related to the app functionalites, e.g., "*clean cache, clean junk file, ...*" When integrating the retrieved responses, CoRe would learn to capture the topics/pattern from the retrieved responses, e.g., "*We notice that you ... We'd like to know if you encounter any issue when use or the reason you dislike it*" in the Retrieved Response-1, for the response generation. Although

User Review: Waste time in my life.

Developer's Response: Hi. Thanks for using our app. We notice that you may have encountered some issues which make you do not like <app>. Would you mind telling us the details? And the reason why you dislike? Feel free to contact us via e-mail <email> many thanks!

CoRe with Only Description: Hi <app> is a free phone cleaner which commits to protection, clean cache, clean junk file, save battery and speed up your <app> phone. We notice that you may have encountered some issue which make you do not like <app>. Would you mind tell us the reason why you dislike via <email> many thanks.

CoRe with Only Retrieval: Hi. Thanks for using our apps. We notice that you may have encountered some issues which make you do not like <app>. Would you mind telling us the details? <user> the reason why you dislike? Feel free to contact us via e-mail <email> many thanks!

Retrieved Response-1: Hi. We notice that you rate us <digit> star without any complaint about our app. We'd like to know if you encounter any issue when use or the reason you dislike it. We would love to obtain feedback from you. Please send us e-mail <email> your advice will be greatly appreciated.

Retrieved Response-2: Hi there. We are sorry for the inconvenience. This issue has our full attention. Could you please send us some screenshots and detail your step which feature be use before this happen phone model version <digit> <app> version. Please contact us via feedback at the top right of <digit> <app> or email <email>

App Description: Clean master, one of the best optimization tool with space cleaner and antivirus for android device ... Clean master key function junk cleaner junk file, free antivirus, wifi security, boost mobile, battery saver, ...

Fig. 17. One example for illustrating the difference between the responses generated by the CoRe without the retrieved responses and the CoRe without the app description. The red fonts indicate the words describing the app functionalites.

the two generated responses are generally similar in semantics, they are slightly different in the descriptions.

7.13 Developer Survey

To obtain developers' opinions about the generated responses by CoRe, we tried to get in touch with the developers of the studied apps through the official emails collected from Google Play. Since some of the 58 subject apps could not be accessed in Google Play, we only collected the developers' emails for 43 apps. We then sent emails to these app developers, in which we provided two examples of the generated responses of the corresponding apps and asked their opinions about the quality of the generated responses. Unfortunately, no feedback was received within two weeks. So, we contacted 12 industry practitioners from the engineering department in several IT companies such as ByteDance and Pinduoduo and asked their help to assess the quality of the generated responses.

Specifically, we design a short survey that contains two questions that ask them about their background, six questions about the quality of the generated responses, and one question about their willingness of adopting our tool in practical development. The 12 participants include six developers, five algorithm engineers, and one test engineer. Around 83.3% (10/12) have more than one year of software engineering experience. For each question about rating the quality of the generated response, we provide a randomly selected review and the corresponding response produced by CoRe and ask the participants to rate the quality on a 1–5 Likert scale [43] (5 for excellent, 4 for good, 3 for undecided, 2 for marginal, and 1 for poor). We only ask six questions about the quality of the generated responses for the purpose of saving the participants' time. The collected results show that 83.3%~91.7% of the participants rate each of the six generated responses as good or excellent. For the last question, the willingness to adopt our tool is also scored on 1–5 Likert scale

Table 15. Results of CoRe with Retrieved Reviews Incorporated

Model	BLEU-4	p_1	p_2	p_3	p_4	Rouge-1	Rouge-2	Rouge-L	METEOR
NMT	**21.61**	**40.55**	**20.75**	**16.78**	**15.47**	36.13	15.47	35.80	29.27
NMT+Reviews	21.12	39.82	20.31	16.37	15.02	**39.59**	**20.53**	**37.97**	**34.50**
CoRe-Retrieval	**38.65**	54.73	37.91	**33.71**	**31.90**	**52.39**	**36.12**	**51.31**	**49.26**
CoRe-Retrieval +Reviews	38.22	**55.51**	**37.98**	33.66	31.77	51.53	34.98	50.32	47.16
CoRe-Description	**38.58**	54.00	36.71	32.55	30.71	**52.36**	**35.85**	**51.05**	**48.54**
CoRe-Description +Reviews	38.41	**54.87**	**37.64**	**33.40**	**31.54**	51.94	35.46	50.68	47.74
CoRe	**40.64**	**57.78**	**40.70**	**36.47**	**34.75**	**53.55**	37.10	**52.28**	49.34
CoRe+Reviews	40.26	56.29	39.47	35.31	33.48	53.54	**37.17**	52.26	**49.75**

User Review: Lot of ad!

CoRe: Thank for your message. Smart charge can help auto clean battery drain <app> when charge. To be able to offer our <app> for free, advertising revenue is the only way to keep our start up run. We will balance the proportion of the ads. Hope that you're having a good day!

CoRe with Retrieved Reviews: Hi pal. Haha I know you like this little flashlight! And could I ask for a 5-star rating? You know your support is the best award for this little flashlight thanks. Pal.

Retrieved Review-1: It is lot of help

Retrieved Review-2: Not lot of action.

Retrieved Review-3: Easy to use thank a lot.

Retrieved Review-4: Too bad because I use it a lot.

Fig. 18. An example of the generated responses with the retrieved responses involved.

(5 for strong agreement, 4 for agreement, 3 for undecided, 2 for disagreement, and 1 for strong disagreement). We find that 83.3% of the participants express that they strongly agree or agree to use CoRe in practical development. The results demonstrate that the quality of the responses generated by CoRe is acceptable and potentially beneficial for developers in practice.

7.14 Analysis of Incorporating the Retrieved Reviews

The design of CoRe considers the retrieved responses during the response generation. However, the retrieved similar reviews may also be useful, since they could provide hints about the described topics. We then conduct experiments by considering user reviews as additional contextual knowledge. The results are illustrated in Table 15. As can be seen, with the similar reviews incorporated, CoRe's performance does not improve substantially and becomes worse in terms of the BLEU-4 and Rouge metrics. CoRe with only app description and CoRe with only the retrieved responses also demonstrate the same trends, i.e., integrating the similar reviews does not benefit the model performance. Although combining NMT with the similar reviews enhances the results of NMT in terms of the Rouge and METEOR scores, the BLEU-4 score is not improved. The unimproved results may be attributed to the fact that the retrieved similar reviews may involve unrelated topics, which could introduce noise. As shown in Figure 18, the retrieved reviews are obviously irrelevant to the given review, so the generated response with the incorporated review does not correctly answer the given review. Since the relevancy between the retrieved reviews and given reviews could affect the model performance, we will investigate how to retrieve and employ relevant reviews for more accurate response generation in the future work.

Table 16. Predicted Results of the Cross Evaluation of the Quality Assurance Filter

Predicted Results		Actual Labels	
		Not Bad	Bad
Predicted Labels	Not Bad	73	15
	Bad	5	7

7.15 Limitations of CoRe

Although the proposed CoRe enhances the performance of review reply generation, CoRe does not handle two case types well, including the reviews that do not require responses and the reviews with poor responses generated by CoRe. For the first case, we refer readers to the work on summarizing which review features spur the responses [34, 63]. In this work, we are more focused on the subsequent behavior for developers, i.e., responding to the reviews requiring responses. For the second case type, we design a quality assurance filter based on the manually annotated review-response pairs in Section 6 to automatically learn the cases in which the proposed CoRe does not perform well. The poorly generated responses can be delegated to developers for further inspection before posting.

Filter Design: The proposed quality assurance filter contains three main steps. We first prepare the gold set for filter training. We employ the involved reviews and the corresponding responses generated by CoRe in the human evaluation as our gold set. Each review and the corresponding generated message are associated with scores that indicate the extent of accuracy to reply to the review (as shown in Figure 10). To be conservative, we labeled the reviews that receive the "*accuracy*" score of one, two, or three from one annotator as "bad" and all the other reviews as "not bad." Then, we extract the unigram tf-idf representations of the reviews as the features, since tf-idf has been widely used in natural language processing for feature representation [23, 69]. We finally train a Gaussian kernel SVM using **stochastic gradient descent (SGD)** as the learning algorithm based on the dataset of reviews and their labels. The trained SVM will be adopted to predict whether the CoRe model generates a "bad" response for a user review.

Filter Performance: We split gold set into 10 folds based on stratified shuffle. For each fold, we train an SVM model on the other 9 folds and test the SVM model on the 1 fold. We finally obtained the test results for every fold. Table 16 shows the predicts of all the folds. In terms of detecting reviews for which the CoRe model will generate "bad" responses, the filter has 83.0% precision and 93.6% recall. Furthermore, it can reduce 31.8% of the "bad" responses. The results demonstrate the usefulness of the proposed filter component for detecting the poorly generated responses. We also deployed the trained filter to the test set used in Section 5 and observed that the model performance showed 40.55 in terms of BLEU-4 score with 2,106/14,727 (14.3%) "bad" responses removed, which is slightly higher than the BLEU-4 score (40.34) reported in our earlier experiment using all the test samples. Developers can focus on examining the "bad" responses during using the proposed CoRe model. For the other reviews, developers can directly adopt the responses generated by CoRe.

Discussion on the Bad Cases. As can be seen in Table 5, 22 responses are labeled as "bad" by the quality assurance filter. The first two authors then skim through the "bad" cases to further inspect the reasons behind the "bad" performance. We find that most responses are semantically consistent with the given reviews, but may dissatisfy users for the following three reasons:

(1) It asks users to raise the review rating: In the 22 "bad" responses, 5 are about asking users to increase the rating, e.g., "*Hi there, thanks a lot for your continuous support. If you rate <app> with 5 stars, it will really encourage us. Thank you.*." Such responses generally respond to positive

feedback, such as "*It's the best camera app.*" We find that among the 5 positive reviews, 4 are rated below 4 stars. The responses reflect the importance of ratings to developers. Although asking users to increase their given rating in responses is common for app developers [34], such responses may not actually satisfy the users. For example, according to Reference [34], only 2.4% of users increase the rating after a developer asks for a rating increase.

(2) It asks for more details about the reported issues based on template: We find that asking users for more details about the reported issues accounts for the largest proportion (59.1%, 13/22) among the responses, which is similar to the finding in Reference [34]. Such responses seem to be template-based, e.g., "*Thanks for your feedback. We always try to improve to keep our user happy. If you have any suggestion for us on how we can do better, please let us know. Thanks.*." Without specific details included, users may feel unsatisfied with the responses.

(3) It provides insufficient details for solutions to the provided issues: For the other four cases, they are all about providing guidance to users to solve their issues. The solutions may not be sufficient for users to solve the problems. For example, one review complained that "*...I still didn't know about saving picture with its actual size. Always the picture is compressed to a smaller size after editing...*," the generated response is "*...You can solve this issue by going to your device's setting about the maximum image size and clicking on the preferred image size....*" Although a solution is provided in the generated response, the users may still be unclear about the detailed steps for the setting. More detailed steps for solving the reported issues could satisfy the users more.

7.16 Threats to Validity

There are three main threats to the validity of our study.

(1) The scale of dataset. We directly use the publicly released data of RRGen provided by their authors. The data include only review-response pairs of 58 free apps from Google Play Store. The limited categories and number of studied apps may influence the generalization of the proposed CoRe. Since the dataset is the only one with huge quantities of review-response pairs at this time, we will eliminate this threat as soon as larger-scale datasets are publicly available.

(2) The retrieved reviews may not always present high similarities. One of the reasons may be the similarity measurement approach is simply based on tf-idf representations, in which the tf-idf may not be the best approach to represent the semantics of the review texts [52]. Another reason is the available review-response pairs may be limited. Since involving more complex approach for retrieving similar reviews could increase the burden of model training and the effectiveness of tf-idf in review representation has already been demonstrated in Reference [69], we investigate the lightweight tf-idf approach in the article. We will explore the impact of different retrieval approaches and datasets on automatic review response generation in the future.

(3) Bias in manual inspection. We invite 20 participants to evaluate the quality of 100 randomly selected review-response pairs. We cannot guarantee that the judgments fully reflect ordinary app users' perceptions of the responses. However, the participants also belong to ordinary app users and are familiar with most studied app categories. Additionally, a majority of them have the experience of skimming app reviews or developers' replies, which makes us more confident about their judgments. Besides, the results of the human evaluation can be impacted by the participants' experience and their understanding of the evaluation metrics. To mitigate the bias in manual inspection, we ensure that each review-response pair was evaluated by five different participants. Besides, we randomly disrupt the order of the three types of responses for each review, so the influence of participants' prior knowledge about the response orders is eliminated.

8 RELATED WORK

We split the related work into three categories: (1) the work that conducts app review mining; (2) the work that analyzes user-developer dialogue; and (3) the work that generates conversational short text.

8.1 App Review Mining

App reviews are a valuable resource provided directly by the customers, which can be exploited by app developers during the bug-fixing [12] and feature-improving process [25]. The essence of app review mining lies in the effective extraction and summarization of the useful information from app reviews. Iacob et al. [37] manually label 3,278 reviews of 161 apps into nine classes and discover that 23.3% of the feedback constitutes requirements from users, e.g., various issues encountered by users. Due to the ever-increasing amount of reviews, previous studies resort to generic NLP techniques to automate the information extraction process. For example, Iacob and Harrison [36] use pre-defined linguistic rules for retrieving feature requests from app reviews. Di Sorbo et al. [62] build a two-level classifiers to summarize the enormous amount of information in user reviews, where user intentions and review topics are, respectively, classified. Developers can learn feature requests and bug reports more quickly when presented with the summary. References [67], [18], [25], and [69], and so on, employ unsupervised clustering methods to prioritize user reviews for better app release planning. Nayebi, Farrahi, and Ruhe [53] adopt app reviews besides other release attributes for predicting release marketability and determining which versions to be released.

Another line of work on app review mining is about predicting user sentiment towards the app features or functionalities [29, 31, 32, 49]. For example, Guzman et al. [32] use topic modeling techniques to group fine-grained features into more meaningful high-level features and then predict the sentiment associated with each feature. Instead of treating reviews as bags-of-words (i.e., mixed review categories), Gu and Kim [31] only consider the reviews related to aspect evaluation and then estimate the aspect sentiment based on a pattern-based parser.

8.2 Analysis of User-developer Dialogue

Analysis of user developer dialogue explores the rich interplay between app customers and their developers [24]. Oh et al. [54] discover that users tend to take a passive action such as uninstalling apps when their request (e.g., user reviews) would take long time to be responded or receive no response. Srisopha et al. [63] investigate which features of user reviews spur developers' responses and find that ratings, review length, and the proportions of positive and negative words are the most important features to predict developer responses. Both McIlroy et al. [51] and Hassan et al. [34]'s studies observe the positive impact of developers' responses on user ratings, for example, users would change their ratings 38.7% of the time following a response. To alleviate the burden in the responding process, Gao et al. [27] propose an NMT-based approach named RRGen for automatically generating the review responses.

8.3 Short Text Conversation Generation

Short text conversation is one of the most challenging natural language processing problems, involving language understanding and utilization of common sense knowledge [61]. Short text conversation can be formulated as a ranking or a generation problem. The former formulation aims at learning the semantic matching relations between conversation histories and responses in the knowledge base, and retrieving the most relevant responses from the base for the current conversation. Ranking-based approaches have the advantage of returning fluent and informative

responses, but may fail to return any appropriate responses for those unseen conversations. The generation-based formulation treats generation of conversational dialogue as a data-driven **statistical machine translation (SMT)** [17, 57] and has been boosted by the success of deep learning models [64] and reinforcement learning approaches [41]. Gao et al. [28] perform a comprehensive survey of neural conversation models in this area. The major problem of the generation-based approaches is that the generated responses are often generic and not informative due to the lack of grounding knowledge [73]. In this work, we propose to integrate contextual knowledge, including app descriptions and retrieved responses, for accurate review response generation.

9 CONCLUSIONS AND FUTURE WORK

This article proposes CoRe, a novel framework aiming at automatically generating accurate responses for user reviews and thereby ensuring a good user experience of the mobile applications. We present that employing app descriptions and the responses of similar user reviews in the training corpus, as contextual knowledge is beneficial for generating high-quality responses. Both automated quantitative evaluation and human evaluation show that the proposed model CoRe significantly outperforms the baseline models. The encouraging experimental results demonstrate the importance of involving contextual knowledge for accurate review response generation. We also analyze the advantages and limitations in this work, and we plan to address the latter in the future.

ACKNOWLEDGMENTS

Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore. This work is a non-Huawei service achievement.

REFERENCES

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2021. BERT repository. Retrieved from https://github.com/huggingface/transformers.
[2] Google Inc. 2021. Character limit in Google Play. Retrieved from https://support.google.com/googleplay/android-developer/answer/9859152.
[3] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. Retrieved from https://nlp.stanford.edu/projects/glove/.
[4] Facebook Community Bot. 2021. PyTorch. Retrieved from https://github.com/pytorch/pytorch.
[5] Apple Inc. 2021. Ratings, reviews, and responses in App Store. Retrieved from https://developer.apple.com/app-store/ratings-and-reviews/.
[6] Tencent Inc. 2021. Tencent online questionnaire. Retrieved from https://wj.qq.com/.
[7] Google Inc. 2021. View and analyze your app's ratings and reviews. Retrieved from https://support.google.com/googleplay/android-developer/answer/138230?hl=en.
[8] SOKO Media. 2019. App download and usage statistics (2019). Retrieved from https://www.businessofapps.com/data/app-statistics/.
[9] Statista Inc. 2019. Number of apps available in leading app stores. Retrieved from https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/.
[10] S. Ejaz Ahmed.2006. Effect sizes for research: A broad application approach. *Technometrics* 48, 4 (2006), 573.
[11] Afnan A. Al-Subaihin, Federica Sarro, Sue Black, Licia Capra, Mark Harman, Yue Jia, and Yuanyuan Zhang. 2016. Clustering mobile apps based on mined textual features. In *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. ACM, 38:1–38:10.
[12] Nasir Ali, Yann-Gaël Guéhéneuc, and Giuliano Antoniol. 2013. Trustrace: Mining software repositories to improve the accuracy of requirement traceability links. *IEEE Trans. Softw. Eng.* 39, 5 (2013), 725–741.
[13] App Revenue. 2018. Mobile app usage. Retrieved from https://www.statista.com/topics/1002/mobile-app-usage/.
[14] Philip Arthur, Graham Neubig, and Satoshi Nakamura. 2016. Incorporating discrete translation lexicons into neural machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 1557–1567.

[15] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations*.

[16] Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization@ACL 2005*, Jade Goldstein, Alon Lavie, Chin-Yew Lin, and Clare R. Voss (Eds.). Association for Computational Linguistics, 65–72.

[17] Grace Chen, Emma Tosch, Ron Artstein, Anton Leuski, and David R. Traum. 2011. Evaluating conversational characters created through question generation. In *Proceedings of the 24th International Florida Artificial Intelligence Research Society Conference*.

[18] Ning Chen, Jialiu Lin, Steven C. H. Hoi, Xiaokui Xiao, and Boshen Zhang. 2014. AR-miner: Mining informative reviews for developers from mobile app marketplace. In *Proceedings of the 36th International Conference on Software Engineering*, Pankaj Jalote, Lionel C. Briand, and André van der Hoek (Eds.). ACM, 767–778.

[19] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processin*. 1724–1734.

[20] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR* abs/1412.3555 (2014).

[21] Adelina Ciurumelea, Andreas Schaufelbühl, Sebastiano Panichella, and Harald C. Gall. 2017. Analyzing reviews and code of mobile apps for better release planning. In *Proceedings of the IEEE 24th International Conference on Software Analysis, Evolution and Reengineering*. 91–102.

[22] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, 4171–4186.

[23] Yuanrui Fan, Xin Xia, David Lo, and Ahmed E. Hassan. 2020. Chaff from the wheat: Characterizing and determining valid bug reports. *IEEE Trans. Softw. Eng.* 46, 5 (2020), 495–525.

[24] Anthony Finkelstein, Mark Harman, Yue Jia, William J. Martin, Federica Sarro, and Yuanyuan Zhang. 2017. Investigating the relationship between price, rating, and popularity in the BlackBerry World app store. *Inf. Softw. Technol.* 87 (2017), 119–139.

[25] Cuiyun Gao, Baoxiang Wang, Pinjia He, Jieming Zhu, Yangfan Zhou, and Michael R. Lyu. 2015. PAID: Prioritizing app issues for developers by tracking user reviews over versions. In *Proceedings of the 26th IEEE International Symposium on Software Reliability Engineering*. IEEE Computer Society, 35–45.

[26] Cuiyun Gao, Jichuan Zeng, Michael R. Lyu, and Irwin King. 2018. Online app review analysis for identifying emerging issues. In *Proceedings of the 40th International Conference on Software Engineering*, Michel Chaudron, Ivica Crnkovic, Marsha Chechik, and Mark Harman (Eds.). ACM, 48–58.

[27] Cuiyun Gao, Jichuan Zeng, Xin Xia, David Lo, Michael R. Lyu, and Irwin King. 2019. Automating app review response generation. In *Proceedings of the 34th IEEE/ACM International Conference on Automated Software Engineering*. IEEE, 163–175.

[28] Jianfeng Gao, Michel Galley, and Lihong Li. 2019. Neural approaches to conversational AI. *Found. Trends Inf. Retr.* 13, 2-3 (2019), 127–298.

[29] Necmiye Genc-Nayebi and Alain Abran. 2017. A systematic literature review: Opinion mining studies from mobile app store user reviews. *J. Syst. Softw.* 125 (2017), 207–219.

[30] G. Grano, A. Ciurumelea, S. Panichella, F. Palomba, and H. C. Gall. 2018. Exploring the integration of user feedback in automated testing of Android applications. In *Proceedings of the IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER'18)*. 72–83. DOI: https://doi.org/10.1109/SANER.2018.8330198

[31] Xiaodong Gu and Sunghun Kim. 2015. What parts of your apps are loved by users? (T). In *Proceedings of the 30th IEEE/ACM International Conference on Automated Software Engineering*. 760–770.

[32] Emitza Guzman and Walid Maalej. 2014. How do users like this feature? A fine grained sentiment analysis of app reviews. In *Proceedings of the IEEE 22nd International Requirements Engineering Conference*, Tony Gorschek and Robyn R. Lutz (Eds.). IEEE Computer Society, 153–162.

[33] Mark Harman, Afnan A. Al-Subaihin, Yue Jia, William Martin, Federica Sarro, and Yuanyuan Zhang. 2016. Mobile app and app store analysis, testing and optimisation. In *Proceedings of the International Conference on Mobile Software Engineering and Systems*. 243–244.

[34] Safwat Hassan, Chakkrit Tantithamthavorn, Cor-Paul Bezemer, and Ahmed E. Hassan. 2018. Studying the dialogue between users and developers of free apps in the Google Play Store. *Empir. Softw. Eng.* 23, 3 (2018), 1275–1312.

[35] Xing Hu, Ge Li, Xin Xia, David Lo, and Zhi Jin. 2018. Deep code comment generation. In *Proceedings of the 26th Conference on Program Comprehension*, Foutse Khomh, Chanchal K. Roy, and Janet Siegmund (Eds.). ACM, 200–210.

[36] Claudia Iacob and Rachel Harrison. 2013. Retrieving and analyzing mobile apps feature requests from online reviews. In *Proceedings of the 10th Working Conference on Mining Software Repositories*. 41–44.
[37] Claudia Iacob, Varsha Veerappa, and Rachel Harrison. 2013. What are you complaining about?: A study of online reviews of mobile applications. In *Proceedings of the 27th International BCS Human Computer Interaction Conference*.
[38] Zongcheng Ji, Zhengdong Lu, and Hang Li. 2014. An information retrieval approach to short text conversation. *CoRR* abs/1408.6988 (2014).
[39] Siyuan Jiang, Ameer Armaly, and Collin McMillan. 2017. Automatically generating commit messages from diffs using neural machine translation. In *Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering*, Grigore Rosu, Massimiliano Di Penta, and Tien N. Nguyen (Eds.). IEEE Computer Society, 135–146.
[40] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*, Yoshua Bengio and Yann LeCun (Eds.).
[41] Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016. Deep reinforcement learning for dialogue generation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Jian Su, Xavier Carreras, and Kevin Duh (Eds.). The Association for Computational Linguistics, 1192–1202.
[42] Petro Liashchynskyi and Pavlo Liashchynskyi. 2019. Grid search, random search, genetic algorithm: A big comparison for NAS. *arXiv preprint arXiv:1912.06059* (2019).
[43] Rensis Likert. 1932. A technique for the measurement of attitudes. *Archives of Psychology* 140:5–55.
[44] Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Proceedings of the Text summarization branches out (ACL-04)*. 8:74–81.
[45] Chao Liu, Cuiyun Gao, Xin Xia, David Lo, John Grundy, and Xiaohu Yang. 2020. On the replicability and reproducibility of deep learning in software engineering. *arXiv preprint arXiv:2006.14244* (2020).
[46] Chia-Wei Liu, Ryan Lowe, Iulian Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How NOT to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Jian Su, Xavier Carreras, and Kevin Duh (Eds.). The Association for Computational Linguistics, 2122–2132.
[47] Zhongxin Liu, Xin Xia, Ahmed E. Hassan, David Lo, Zhenchang Xing, and Xinyu Wang. 2018. Neural-machine-translation-based commit message generation: How far are we? In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*. 373–384.
[48] Zhongxin Liu, Xin Xia, Christoph Treude, David Lo, and Shanping Li. 2019. Automatic generation of pull request descriptions. In *Proceedings of the 34th IEEE/ACM International Conference on Automated Software Engineering*. IEEE, 176–188.
[49] Washington Luiz, Felipe Viegas, Rafael Odon de Alencar, Fernando Mourão, Thiago Salles, Dárlinton Carvalho, Marcos André Gonçalves, and Leonardo C. da Rocha. 2018. A feature-oriented sentiment rating for mobile app reviews. In *Proceedings of the World Wide Web Conference on World Wide Web*. 1909–1918.
[50] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.
[51] Stuart McIlroy, Weiyi Shang, Nasir Ali, and Ahmed E. Hassan. 2017. Is it worth responding to reviews? Studying the top free apps in Google Play. *IEEE Softw.* 34, 3 (2017), 64–71.
[52] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 27th Annual Conference on Neural Information Processing Systems*. 3111–3119.
[53] Maleknaz Nayebi, Homayoon Farrahi, and Guenther Ruhe. 2017. Which version should be released to app store? In *Proceedings of the ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, Ayse Bener, Burak Turhan, and Stefan Biffl (Eds.). IEEE Computer Society, 324–333.
[54] Jeungmin Oh, Daehoon Kim, Uichin Lee, Jae-Gil Lee, and Junehwa Song. 2013. Facilitating developer-user interactions with mobile app review digests. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, Wendy E. Mackay, Stephen A. Brewster, and Susanne Bødker (Eds.). ACM, 1809–1814.
[55] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics* 311–318.
[56] Gerasimos Razis, Georgios Theofilou, and Ioannis Anagnostopoulos. 2021. Latent Twitter image information for social analytics. *Information* 12, 2 (2021), 49.
[57] Alan Ritter, Colin Cherry, and William B. Dolan. 2011. Data-driven response generation in social media. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 583–593.
[58] Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. 2010. Automatic keyword extraction from individual documents. *Text Mining: Applic. Theor.* 1 (2010), 1–20.

[59] Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, Regina Barzilay and Min-Yen Kan (Eds.). Association for Computational Linguistics, 1073–1083.

[60] Rico Sennrich, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, and Maria Nadejde. 2017. Nematus: A toolkit for neural machine translation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, Andre Martins and Anselmo Peñas (Eds.). Association for Computational Linguistics, 65–68.

[61] Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*. The Association for Computer Linguistics, 1577–1586.

[62] Andrea Di Sorbo, Sebastiano Panichella, Carol V. Alexandru, Junji Shimagaki, Corrado Aaron Visaggio, Gerardo Canfora, and Harald C. Gall. 2016. 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*. 499–510.

[63] Kamonphop Srisopha, Devendra Swami, Daniel Link, and Barry W. Boehm. 2020. How features in iOS app store reviews can predict developer responses. In *Proceedings of the Evaluation and Assessment in Software Engineering Conference*, Jingyue Li, Letizia Jaccheri, Torgeir Dingsøyr, and Ruzanna Chitchyan (Eds.). ACM, 336–341.

[64] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the Annual Conference on Neural Information Processing Systems*. 3104–3112.

[65] Mike Thelwall, Kevan Buckley, Georgios Paltoglou, Di Cai, and Arvid Kappas. 2010. Sentiment in short strength detection informal text. *J. Assoc. Inf. Sci. Technol.* 61, 12 (2010), 2544–2558.

[66] Michael E. Tipping and Christopher M. Bishop. 1999. Probabilistic principal component analysis. *J. Roy. Statist. Soc.: Series B (Statist. Methodol.)* 61, 3 (1999), 611–622.

[67] Lorenzo Villarroel, Gabriele Bavota, Barbara Russo, Rocco Oliveto, and Massimiliano Di Penta. 2016. Release planning of mobile apps based on user reviews. In *Proceedings of the 38th International Conference on Software Engineering*, Laura K. Dillon, Willem Visser, and Laurie Williams (Eds.). ACM, 14–24.

[68] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Proceedings of the Annual Conference on Neural Information Processing Systems*. 2692–2700.

[69] Phong Minh Vu, Tam The Nguyen, Hung Viet Pham, and Tung Thanh Nguyen. 2015. Mining user opinions in mobile app reviews: A keyword-based approach (T). In *Proceedings of the 30th IEEE/ACM International Conference on Automated Software Engineering*, Myra B. Cohen, Lars Grunske, and Michael Whalen (Eds.). IEEE Computer Society, 749–759.

[70] Frank Wilcoxon. 1992. Individual comparisons by ranking methods. In *Breakthroughs in Statistics*. Springer, 196–202.

[71] Sixing Wu, Ying Li, Dawei Zhang, Yang Zhou, and Zhonghai Wu. 2020. Diverse and informative dialogue generation with context-specific commonsense knowledge awareness. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault (Eds.). Association for Computational Linguistics, 5811–5820.

[72] Zhizheng Wu and Simon King. 2016. Investigating gated recurrent networks for speech synthesis. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 5140–5144.

[73] Liu Yang, Junjie Hu, Minghui Qiu, Chen Qu, Jianfeng Gao, W. Bruce Croft, Xiaodong Liu, Yelong Shen, and Jingjing Liu. 2019. A hybrid retrieval-generation neural conversation model. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. ACM, 1341–1350.

[74] Xinli Yang, David Lo, Xin Xia, Lingfeng Bao, and Jianling Sun. 2016. Combining word embedding with information retrieval to recommend similar bug reports. In *Proceedings of the 27th IEEE International Symposium on Software Reliability Engineering*. IEEE Computer Society, 127–137.

[75] Yong Yu, Xiaosheng Si, Changhua Hu, and Jianxun Zhang. 2019. A review of recurrent neural networks: LSTM cells and network architectures. *Neural Comput.* 31, 7 (2019), 1235–1270.

[76] Jingyi Zhang, Masao Utiyama, Eiichiro Sumita, Graham Neubig, and Satoshi Nakamura. 2018. Guiding neural machine translation with retrieved translation pieces. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 1325–1335.

Received October 2020; revised March 2021; accepted May 2021