

Singapore Management University

Institutional Knowledge at Singapore Management University

Research Collection School Of Computing and Information Systems

School of Computing and Information Systems

1-2022

Automating app review response generation based on contextual knowledge

Cuiyun GAO

Wenjie ZHOU

Xin XIA

David LO

Singapore Management University, davidlo@smu.edu.sg

Qi XIE

See next page for additional authors

Follow this and additional works at: https://ink.library.smu.edu.sg/sis_research



Part of the [Databases and Information Systems Commons](#), [OS and Networks Commons](#), and the [Software Engineering Commons](#)

Citation

GAO, Cuiyun; ZHOU, Wenjie; XIA, Xin; LO, David; XIE, Qi; and LYU, Michael R.. Automating app review response generation based on contextual knowledge. (2022). *ACM Transactions on Software Engineering and Methodology*. 31, (1), 1-36.

Available at: https://ink.library.smu.edu.sg/sis_research/7668

This Journal Article is brought to you for free and open access by the School of Computing and Information Systems at Institutional Knowledge at Singapore Management University. It has been accepted for inclusion in Research Collection School Of Computing and Information Systems by an authorized administrator of Institutional Knowledge at Singapore Management University. For more information, please email cherylds@smu.edu.sg.

Author

Cuiyun GAO, Wenjie ZHOU, Xin XIA, David LO, Qi XIE, and Michael R. LYU

Automating App Review Response Generation Based on Contextual Knowledge

CUIYUN GAO, Harbin Institute of Technology (Shenzhen), China

WENJIE ZHOU, Southwest Minzu University, China

XIN XIA, Monash University, Australia

DAVID LO, Singapore Management University, Singapore

QI XIE*, 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 audience volumes and app revenue. To ensure good user experience and assist app development, several prior studies resort to analysis of app reviews, a type of app 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 paper, we propose a novel end-to-end 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 11.53% 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.

ACM Reference Format:

Cuiyun Gao, Wenjie Zhou, Xin Xia, David Lo, Qi Xie, and Michael R. Lyu. 2020. Automating App Review Response Generation Based on Contextual Knowledge. 1, 1 (October 2020), 24 pages. <https://doi.org/10.1145/nnnnnnn.nnnnnnn>

*Corresponding author.

Authors' addresses: Cuiyun Gao, gaocuiyun@hit.edu.cn, Harbin Institute of Technology (Shenzhen), China; Wenjie Zhou, zhouwenjie2@stu.swun.edu.cn, Southwest Minzu University, China; Xin Xia, xin.xia@monash.edu, Monash University, Australia; David Lo, davidlo@smu.edu.sg, Singapore Management University, Singapore; Qi Xie, qi.xie.swun@gmail.com, Southwest Minzu University, China; Michael R. Lyu, lyu@cse.cuhk.edu.hk, The Chinese University of Hong Kong, Hong Kong, China.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2020 Association for Computing Machinery.

XXXX-XXXX/2020/10-ART \$15.00

<https://doi.org/10.1145/nnnnnnn.nnnnnnn>

1 INTRODUCTION

According to the report released by [6], there are over five 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 [7]. 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 realize, test, optimize, maintain and categorize apps (see e.g., [18, 22, 26, 29, 48]) for ensuring good user experience.

The app stores such as Google Play and App Store also allow developers to respond to the reviews [3, 5], and encourage them to conduct review response promptly and precisely for creating a better user experience and improving app ratings. A recent study by Hassan et al. [30] 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 responding. McIlroy et al. [40] 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 of the advantage of review response, developers of many apps never respond to the reviews [30, 40]. 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 [11]. It is labor-intensive and time-consuming for developers to respond to each piece of review. Therefore, the prior work [23] initiates automating the review response process.

Review response generation can be analogical to social dialogue generation [37, 56] 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. [23] proposed a Neural Machine Translation (NMT) [50]-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) [16] 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, RRGen highly relies on the performance of the external tools such as SURF [48] for determining pre-defined keywords and SentiStrength [51] for estimating review sentiment. This weakens the flexibility and generalization of RRGen, e.g., when keywords in the reviews are not in the pre-defined keyword dictionary. Second, RRGen presents the similar 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 [12, 58, 60].

To alleviate the above limitations, we propose a novel neural architecture namely Contextual knowledge-based app Review response 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 [9]. Based

★★★★★ **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.

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 [34] 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 [46] 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 11.53% in terms of BLEU-4 score [44] (An accuracy measure that is widely used to evaluate text generation systems). Human study with 20 programmers through Tencent Online Questionnaire [4] further confirms that CoRe can generate a more relevant and accurate response than RRGGen.

The remainder of this paper 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 paper.

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

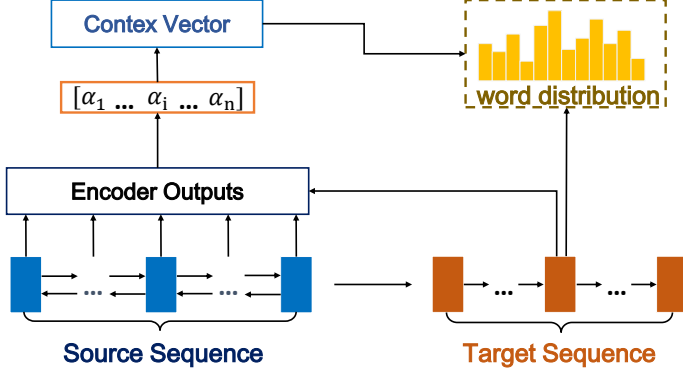


Fig. 2. Graphical illustration of the attentional bi-LSTM encoder-decode model.

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 $\mathbf{w} = (w_1, w_2, \dots, w_n)$ (n is the number of input tokens) are fed one-by-one into the encoder (a single-layer bidirectional GRU [16] as shown in Figure 2), producing a sequence of encoder hidden states $\mathbf{h} = (h_1, h_2, \dots, h_n)$. 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 $\{y_1, \dots, y_{t-1}\}$, 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 [13]:

$$c_t = \sum_j^n \alpha_{tj} h_j, \quad (1)$$

$$\alpha_{tj} = \text{softmax}(e_{tj}),$$

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 fed through two linear layers to generate the vocab distribution:

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

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 model is trained to minimize the negative log likelihood:

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

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

2.2 Pointer-Generator Model

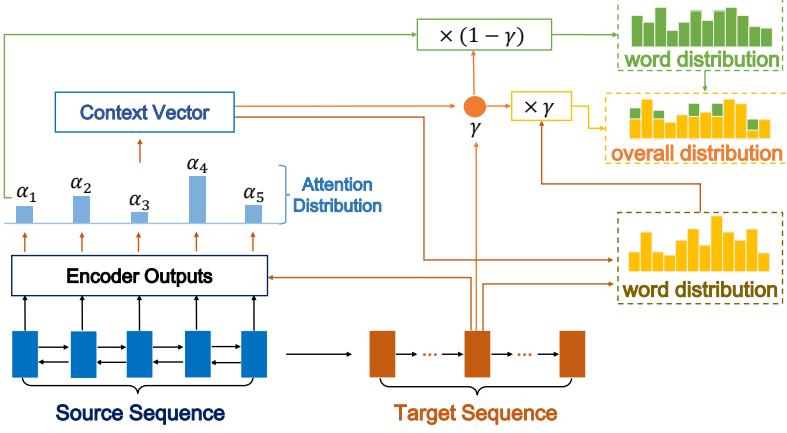


Fig. 3. Graphical illustration of the pointer-generator model.

Pointer-generator networks [46, 53] 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 point-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 :

$$\gamma_t = \sigma(\omega_c^T c_t + \omega_s^T s_t + \omega_w^T w_t + b_{ptr}), \quad (4)$$

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:

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

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 equations (3).

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 [9], 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

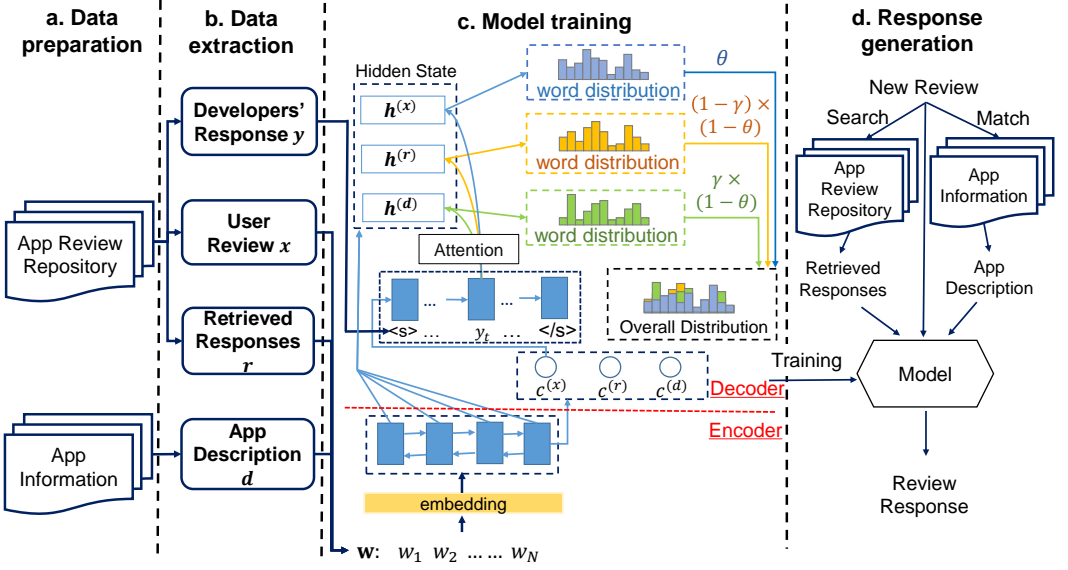


Fig. 4. Overall architecture of CoRe.

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.

3.1 Source Sequence Encoding

Let $\mathbf{w} = (w_1, w_2, \dots, w_n)$ 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.

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

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

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

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

3.2 Contextual Knowledge Integration

Different from the basic pointer-generator network [46], 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:

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

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

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

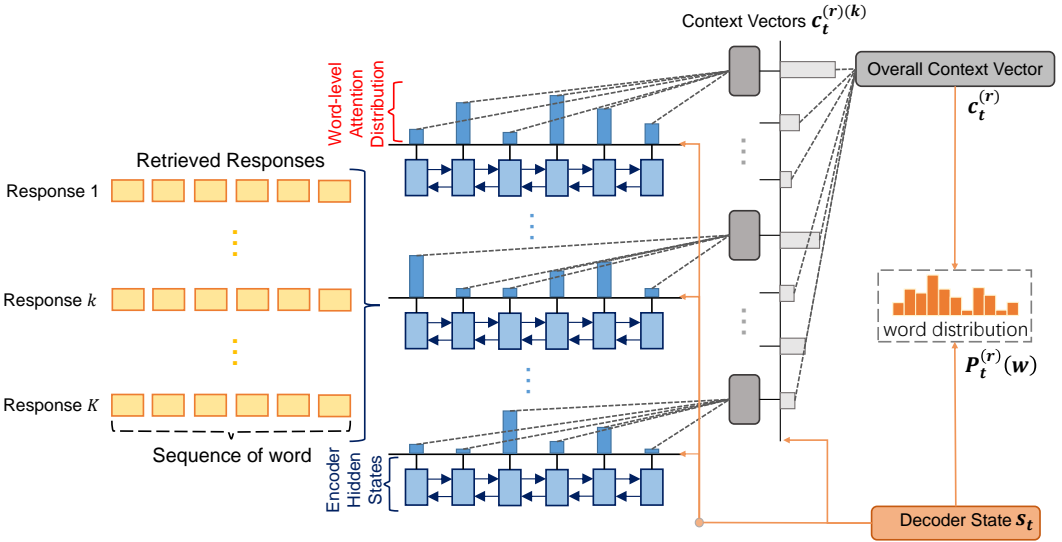


Fig. 5. Illustration of the hierarchical pointer network for copying tokens from the retrieved K responses.

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:

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

$$\alpha_t^{(r)}, c_t^{(r)} = \text{Attention}([c_t^{(r)(1)}, \dots, c_t^{(r)(K)}], s_t), \quad (12)$$

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

where $\mathbf{h}^{(r)(k)}$ is the output of the encoder for the response of the top k -th 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).

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

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

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 .

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

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 end-to-end with the negative log-likelihood loss function of

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

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) [16] which is a popular basic encoder-decoder model and performs well in many text generation tasks [17, 57]. The hidden units of GRUs are set as 200 and word embeddings are initiated with pre-trained 100-dimensional GloVe vectors [1]. 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 at Section 5.3. The whole model is trained using the minibatch Adam [36], a stochastic optimization approach which 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 are most frequently appeared in the training set.

For implementation, we use PyTorch [2], an open-source deep learning framework. We train our model in a server with Intel(R) Xeon(R) Gold 6230 CPU @ 2.10GHz, Tesla T4 16G. The training lasts ~8 hours with three epochs.

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 [44]. 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 [23]. 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. 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 [23], 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 learnt by the basic attentional encoder-decoder model [59], 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 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.

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) [31, 35, 37, 60]. 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.

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

where j indexes all possible n-grams, and $h(j, y)$ or $h(j, \hat{y})$ indicate the number of j -th 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 favours shorter generated sentences, BLEU-N introduces a brevity penalty.

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

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 judgements than other evaluation metrics [38].

4.3 Baseline Approaches

We compare the performance of the proposed CoRe with a random selection approach, the basic attentional encoder-decoder model [13], and the state-of-the-art approach for review response generation [23], namely RRGen. We elaborate on the first and last baselines as below.

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

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.

Table 1. Comparison results with baseline approaches. **Bold** figures highlight better results. p_n indicates the n -gram precision computed in Equation (18). Statistical significance results are indicated with $*$ (p -value <0.01).

Model	BLEU-4	p_1	p_2	p_3	p_4
Random	6.55*	27.64*	6.90*	3.55*	2.78*
NMT	21.61*	40.55*	20.75*	16.78*	15.47*
RRGen	36.17*	53.24*	35.83*	31.73*	30.04*
CoRe	40.34	57.17	40.24	35.96	34.11

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 [23], and another competing approach, NMT [13], 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 illustrate 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 three baselines by 11.53%~5.16 times. 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 13.55% regarding the accuracy of 4-gram prediction.

We then use Wilcoxon signed-rank test [55] to verify whether the increase is significant, and Cliff’s Delta (or d) to measure the effect size [8]. 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.

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, presenting only 20.1 in terms of the BLEU-4 metric. 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.

Table 2. Contrastive experiments with individual extension removed.

Model	BLEU-4	p_1	p_2	p_3	p_4
CoRe	40.34	57.17	40.24	35.96	34.11
-Retrieval	38.65	54.73	37.91	33.71	31.90
-Description	38.58	54.00	36.71	32.55	30.71
Only review (NMT)	21.61	40.55	20.75	16.78	15.47

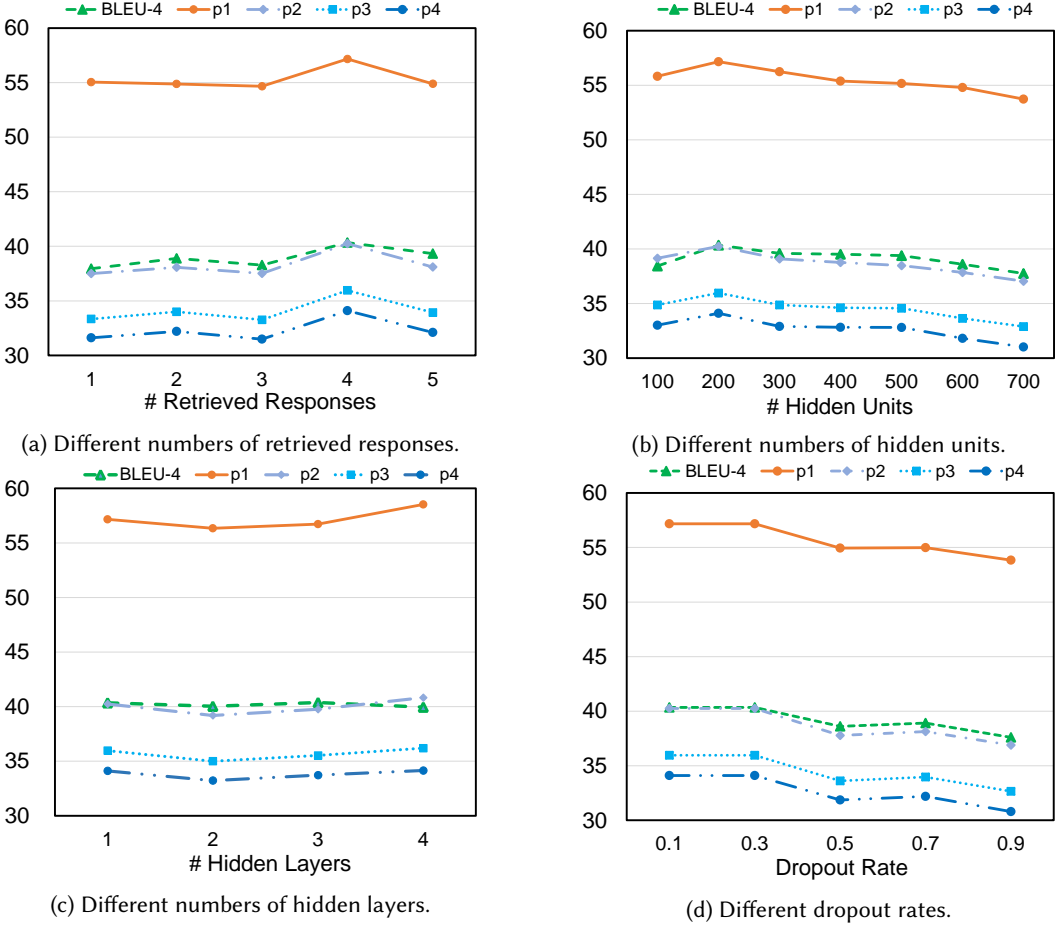


Fig. 6. Model performance under different parameter settings.

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. Figure 6 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.

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
25	40.15	56.87	39.45	35.16	33.34
50	38.04	54.13	37.30	33.12	31.31
100	40.34	57.17	40.24	35.96	34.11
200	39.09	55.20	38.42	34.13	32.27

Retrieved Responses: As can be seen in Figure 6 (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 6 (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 6 (c) depicts the variations of the model performance as the number of hidden layers increase. We can observe that the variations are not obvious, ranging from 39.96 to 40.34 in terms of BLEU-4 score. Since with more hidden layers, both model training and testing time will increase, we set the number of hidden layers as 1 during the evaluation.

Dropout Rate: As can be seen in Figure 6 (d), as the dropout rate grows, the model accuracy presents a decline trend, which implies that large dropout rates could greatly reduce the knowledge learnt 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 [1] 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 defined 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 bachelors 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. 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 if 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

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

question describes one review-response pair, comprising one piece of user review, the developers' response, and its responses generated by RRGGen and CoRe. The order of the responses are randomly disrupted for each review.

Following [23], 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 the preference. The "preference rank" score is evaluated on 1-3 scale (1 for the most preferred and 3 for the lease preferred). Figure 7 shows one example of questions 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, RRGGen, and official developers. The participants spent 1.72 hours on completing the questionnaire on average, with the median time cost at 1.40 hours. We compute the agreement rate on the four

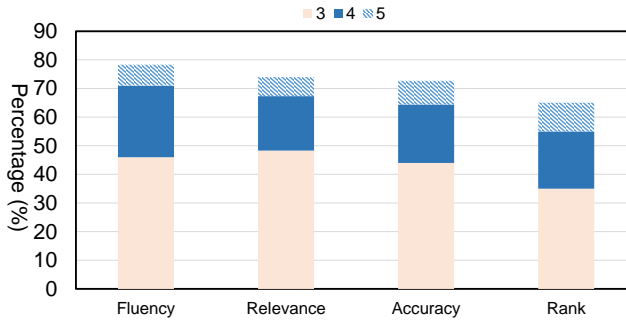


Fig. 8. Agreement rate among the participants in the human evaluation. The horizontal axis and vertical axis indicate different evaluation metrics and the percentages of 3/4/5 participants giving the same scores, respectively.

aspects given by the participants, illustrated in Figure 8. As can be seen, 78.3%, 74.0%, 72.7% and 65.0% of the total 100 review-response pairs received at least 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 terms of the respective metrics. This indicates that the participants achieved reasonable agreement on the quality of the generated responses.

Table 4 and Figure 9 depict the results of human evaluation. As can be seen, the responses from official developers receive 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 9 (a), most participants give 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 9 (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 comparing to the generated responses. This implies that the responses output by CoRe tend 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 9 (c), the responses generated by CoRe receive slightly more 4/5-star ratings than the developers’ responses (391 v.s. 384), and 1.22 times than the responses generated by RRGen (176). The result 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 9 (d). We can 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%), and the responses of CoRe present similar preference score as the developers’ responses on average, i.e., 1.79 v.s. 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, limitations, and threats of our model.

Table 4. Comparison results based on human evaluation. Average scores are computed and **bold** indicates top scores. Two-tailed t-test results between CoRe and RRGGen are indicated with $*(p\text{-value}<0.01)$.

	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

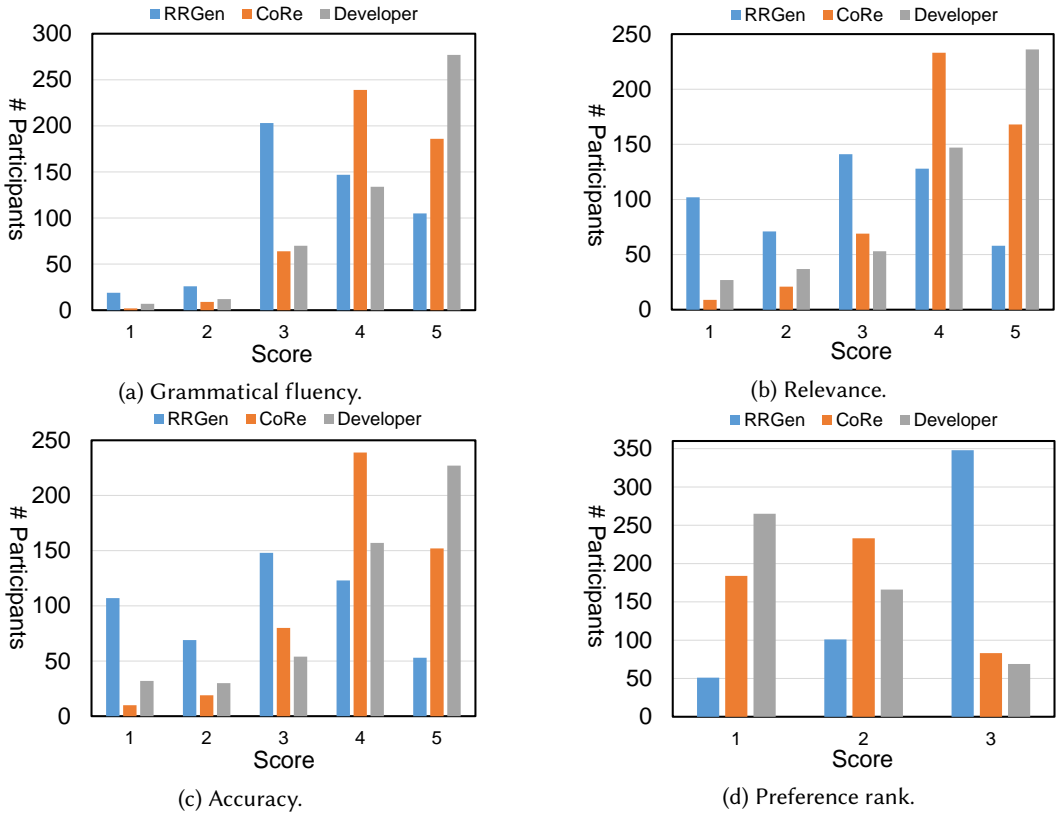


Fig. 9. Human evaluation results. For the metrics “grammatical fluency”, “relevance” and “accuracy”, the higher scores the better; while for the metric “preference rank”, the lower scores the better. The vertical axis indicates the number of participants giving the scores.

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/functionality discussed by users more accurately. For example, it can learn that the review “It lose your full charge.” is related to

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

Fig. 10. 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.

the “*power save mode*” in the app, and generate response providing the solution “*trying different save mode*”, as shown in Figure 10; while the response generated by RRGen is rather in general purpose and not topically relevant to the review. Figure 11 visualizes the latent alignment over the user review/app description based on the attention weights α_{tj} from Equation (1) and $\alpha_t^{(r)}$ 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 [12, 58, 60]. 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 \leq 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 12, we retrieve most similar reviews with respect the semantics (i.e., tf-idf representations in the paper) 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

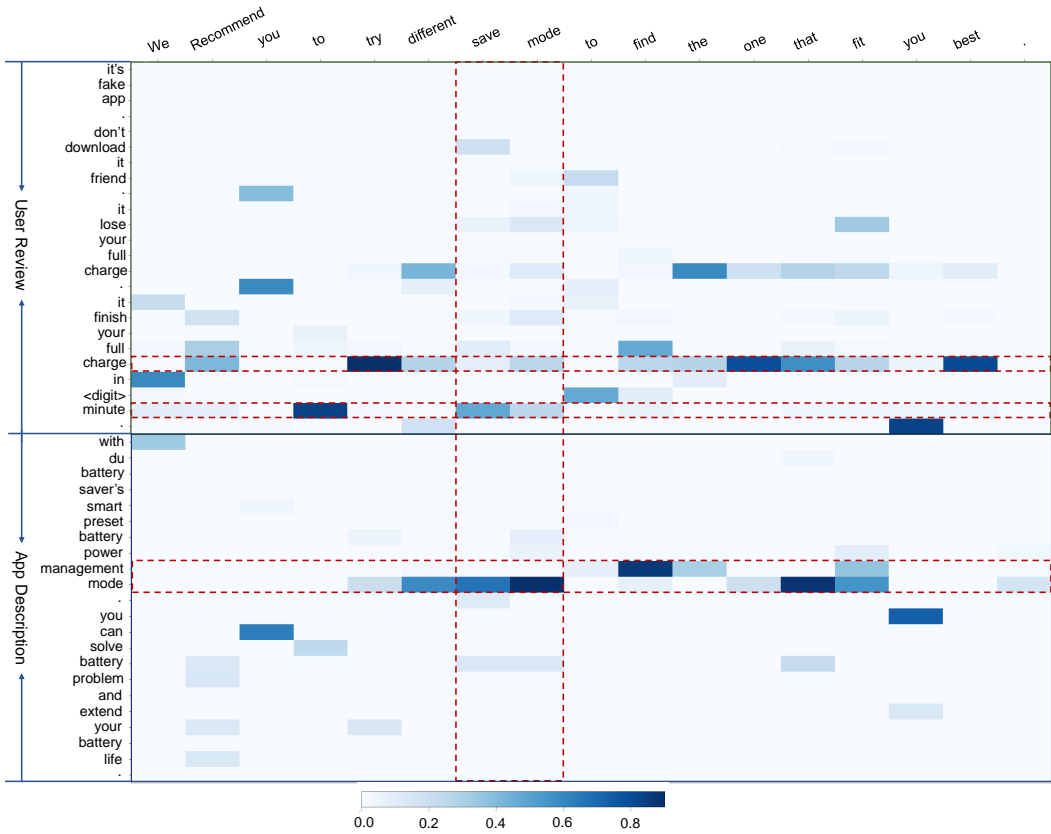


Fig. 11. A heatmap representing the alignment between the user review (left-top)/app description (left-bottom) and generated response by CoRe (top). The columns represent the distribution over the user review/app description after generating each word in the response. Darker colors indicate higher attention weights and manifest a stronger correlation between the target word and source word. Red dotted rectangles highlight partial topical words in corresponding texts.

websites in a standardized format) also appears in the retrieved response (i.e., Retrieved response-2). The words are ignored by RRGGen 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 RRGGen 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 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 type, we refer readers to the work [30, 49] on summarizing which review features spur the responses. 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

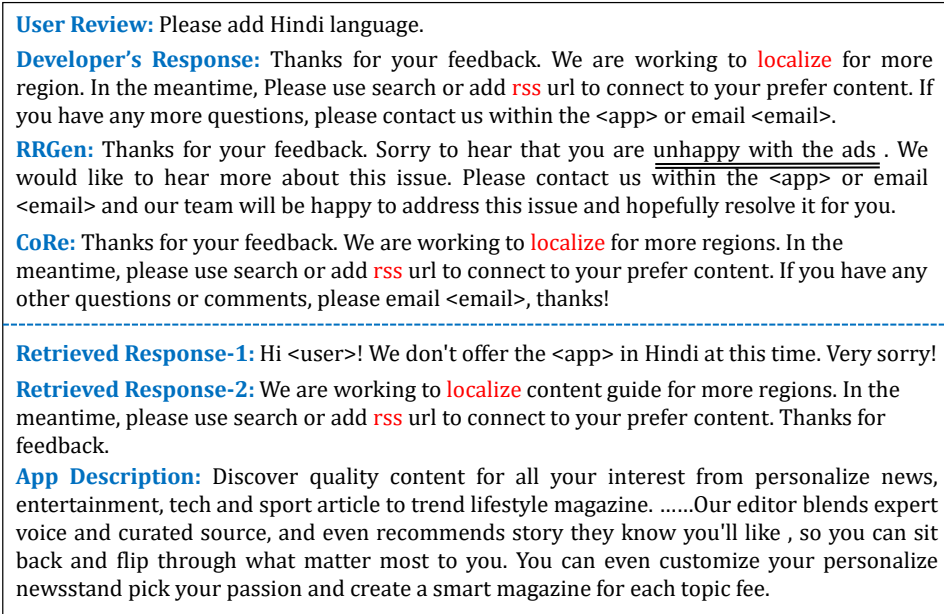


Fig. 12. A user review with the generated response where CoRe can generate responses with low-frequency words. The fonts in red are indicative of the low-frequency words (frequency \leq 100) and the double-underlined words mean they are topically irrelevant to the user review. Responses of the retrieved top two reviews and the app description are also illustrated.

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 which indicate the extent of accuracy to reply to the review (as shown in Figure 9). 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 [19, 54]. 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 a SVM model on the other 9 folds, and test the SVM model on the one fold. We finally obtained the test results for every fold. Table 5 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

Table 5. 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

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.

7.3 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 RRGGen 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 [41]. 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 [54], we investigate the light-weight tf-idf approach in the paper. We will explore the impact of different retrieval approaches and datasets on automatic review response generation in the future.
- (3) Bias in manual inspection. 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 that 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 short text conversational.

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 [10] and feature-improving process [21]. The essence of app review mining lies in the effective extraction and summarization of the useful information from app reviews. Jacob et al. [33] 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, Jacob and Harrison [32]

use pre-defined linguistic rules for retrieving feature requests from app reviews. Di Sorbo et al. [48] 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. [52], [15], [21], and [54], etc., employ unsupervised clustering methods to prioritize user reviews for better app release planning. Nayebi, Farrahi, and Ruhe [42] 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 [25, 27, 28, 39]. For example, Guzman et al. [28] 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 [27] 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 [20]. Oh et al. et al. [43] discover that users tend to take a passive action such as uninstalling apps when their inquires (e.g., user reviews) would take long time to be responded or receive no response. Srisopha et al. [49] 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. [40] and Hassan et al. [30]'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. [23] propose an NMT-based approach named RRGGen 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 [47]. 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) [14, 45], and has been boosted by the success of deep learning models [50] and reinforcement learning approaches [37]. Gao et al. [24] 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 [58]. 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 paper 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 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 plan to address them in the future.

REFERENCES

- [1] [n.d.]. GloVe: Global Vectors for Word Representation. <https://nlp.stanford.edu/projects/glove/>.
- [2] [n.d.]. PyTorch. <https://pytorch.org/>.
- [3] [n.d.]. Ratings, reviews, and responses in App Store. <https://developer.apple.com/app-store/ratings-and-reviews/>.
- [4] [n.d.]. Tencent online questionnaire. <https://wj.qq.com/>.
- [5] [n.d.]. View and analyze your app's ratings and reviews. <https://support.google.com/googleplay/android-developer/answer/138230?hl=en>.
- [6] 2019. App Download and Usage Statistics (2019). <https://www.businessofapps.com/data/app-statistics/>.
- [7] 2019. Number of apps available in leading app stores. <https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/>.
- [8] S. Ejaz Ahmed. 2006. Effect Sizes for Research: A Broad Application Approach. *Technometrics* 48, 4 (2006), 573.
- [9] Afnan A. Al-Subaihini, 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, ESEM 2016, Ciudad Real, Spain, September 8-9, 2016*. ACM, 38:1–38:10.
- [10] Nasir Ali, Yann-Gaël Guéhéneuc, and Giuliano Antoniol. 2013. Trustrace: Mining Software Repositories to Improve the Accuracy of Requirement Traceability Links. *IEEE Trans. Software Eng.* 39, 5 (2013), 725–741.
- [11] App Revenue [n.d.]. Mobile app usage. <https://www.statista.com/topics/1002/mobile-app-usage/>.
- [12] Philip Arthur, Graham Neubig, and Satoshi Nakamura. 2016. Incorporating Discrete Translation Lexicons into Neural Machine Translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*. 1557–1567.
- [13] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.
- [14] Grace Chen, Emma Tosch, Ron Artstein, Anton Leuski, and David R. Traum. 2011. Evaluating Conversational Characters Created through Question Generation. In *Proceedings of the Twenty-Fourth International Florida Artificial Intelligence Research Society Conference, May 18-20, 2011, Palm Beach, Florida, USA*.
- [15] 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 *36th International Conference on Software Engineering, ICSE '14, Hyderabad, India - May 31 - June 07, 2014*, Pankaj Jalote, Lionel C. Briand, and André van der Hoek (Eds.). ACM, 767–778.
- [16] Kyunghyun Cho, Bart van Merriënboer, Çağlar 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 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*. 1724–1734.
- [17] Junyoung Chung, Çağlar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *CoRR* abs/1412.3555 (2014).
- [18] Adelina Ciurumelea, Andreas Schaufelbühl, Sebastiano Panichella, and Harald C. Gall. 2017. Analyzing reviews and code of mobile apps for better release planning. In *IEEE 24th International Conference on Software Analysis, Evolution and Reengineering, SANER 2017, Klagenfurt, Austria, February 20-24, 2017*. 91–102.
- [19] Yuanrui Fan, Xin Xia, David Lo, and Ahmed E. Hassan. 2020. Chaff from the Wheat: Characterizing and Determining Valid Bug Reports. *IEEE Trans. Software Eng.* 46, 5 (2020), 495–525.
- [20] 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.
- [21] 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 *26th IEEE International Symposium on Software Reliability Engineering, ISSRE 2015, Gaithersbury, MD, USA, November 2-5, 2015*. IEEE Computer Society, 35–45.
- [22] 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, ICSE 2018, Gothenburg, Sweden, May 27 - June 03, 2018*, Michel Chaudron, Ivica Crnkovic, Marsha Chechik, and Mark Harman (Eds.). ACM, 48–58.

- [23] Cuiyun Gao, Jichuan Zeng, Xin Xia, David Lo, Michael R. Lyu, and Irwin King. 2019. Automating App Review Response Generation. In *34th IEEE/ACM International Conference on Automated Software Engineering, ASE 2019, San Diego, CA, USA, November 11-15, 2019*. IEEE, 163–175.
- [24] Jianfeng Gao, Michel Galley, and Lihong Li. 2019. Neural Approaches to Conversational AI. *Found. Trends Inf. Retr.* 13, 2-3 (2019), 127–298.
- [25] Necmiye Genc-Nayebi and Alain Abran. 2017. A systematic literature review: Opinion mining studies from mobile app store user reviews. *Journal of Systems and Software* 125 (2017), 207–219.
- [26] 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 *IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER'18)*. 72–83. <https://doi.org/10.1109/SANER.2018.8330198>
- [27] Xiaodong Gu and Sunghun Kim. 2015. "What Parts of Your Apps are Loved by Users?" (T). In *30th IEEE/ACM International Conference on Automated Software Engineering, ASE 2015, Lincoln, NE, USA, November 9-13, 2015*. 760–770.
- [28] Emitza Guzman and Walid Maalej. 2014. How Do Users Like This Feature? A Fine Grained Sentiment Analysis of App Reviews. In *IEEE 22nd International Requirements Engineering Conference, RE 2014, Karlskrona, Sweden, August 25-29, 2014*, Tony Gorschek and Robyn R. Lutz (Eds.). IEEE Computer Society, 153–162.
- [29] Mark Harman, Afnan A. Al-Subaihini, 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, MOBILESoft '16, Austin, Texas, USA, May 14-22, 2016*. 243–244.
- [30] 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. *Empirical Software Engineering* 23, 3 (2018), 1275–1312.
- [31] 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, ICPC 2018, Gothenburg, Sweden, May 27-28, 2018*, Foutse Khomh, Chanchal K. Roy, and Janet Siegmund (Eds.). ACM, 200–210.
- [32] 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, MSR '13, San Francisco, CA, USA, May 18-19, 2013*. 41–44.
- [33] Claudia Iacob, Varsha Veerappa, and Rachel Harrison. 2013. What are you complaining about?: a study of online reviews of mobile applications. In *BCS-HCI '13 Proceedings of the 27th International BCS Human Computer Interaction Conference, Brunel University, London, UK, 9-13 September 2013*. 29.
- [34] Zongcheng Ji, Zhengdong Lu, and Hang Li. 2014. An Information Retrieval Approach to Short Text Conversation. *CoRR* abs/1408.6988 (2014).
- [35] 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, ASE 2017, Urbana, IL, USA, October 30 - November 03, 2017*, Grigore Rosu, Massimiliano Di Penta, and Tien N. Nguyen (Eds.). IEEE Computer Society, 135–146.
- [36] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.).
- [37] Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016. Deep Reinforcement Learning for Dialogue Generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, Jian Su, Xavier Carreras, and Kevin Duh (Eds.). The Association for Computational Linguistics, 1192–1202.
- [38] 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 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, Jian Su, Xavier Carreras, and Kevin Duh (Eds.). The Association for Computational Linguistics, 2122–2132.
- [39] 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 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*. 1909–1918.
- [40] 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 Software* 34, 3 (2017), 64–71.
- [41] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe,*

- Nevada, United States. 3111–3119.
- [42] Maleknaz Nayebi, Homayoon Farrahi, and Guenther Ruhe. 2017. Which Version Should Be Released to App Store?. In *2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM 2017, Toronto, ON, Canada, November 9-10, 2017*, Ayse Bener, Burak Turhan, and Stefan Biffl (Eds.). IEEE Computer Society, 324–333.
- [43] Jeungmin Oh, Daehoon Kim, Uichin Lee, Jae-Gil Lee, and Junehwa Song. 2013. Facilitating developer-user interactions with mobile app review digests. In *2013 ACM SIGCHI Conference on Human Factors in Computing Systems, CHI '13, Paris, France, April 27 - May 2, 2013, Extended Abstracts*, Wendy E. Mackay, Stephen A. Brewster, and Susanne Bødker (Eds.). ACM, 1809–1814.
- [44] 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, July 6-12, 2002, Philadelphia, PA, USA*. 311–318.
- [45] Alan Ritter, Colin Cherry, and William B. Dolan. 2011. Data-Driven Response Generation in Social Media. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, 27-31 July 2011, John McIntyre Conference Centre, Edinburgh, UK, A meeting of SIGDAT, a Special Interest Group of the ACL*. 583–593.
- [46] 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, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, Regina Barzilay and Min-Yen Kan (Eds.). Association for Computational Linguistics, 1073–1083.
- [47] 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, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*. The Association for Computer Linguistics, 1577–1586.
- [48] 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, FSE 2016, Seattle, WA, USA, November 13-18, 2016*. 499–510.
- [49] Kamonphop Srisopha, Devendra Swami, Daniel Link, and Barry W. Boehm. 2020. How features in iOS App Store Reviews can Predict Developer Responses. In *EASE '20: Evaluation and Assessment in Software Engineering, Trondheim, Norway, April 15-17, 2020*, Jingyue Li, Letizia Jaccheri, Torgeir Dingsøyr, and Ruzanna Chitchyan (Eds.). ACM, 336–341.
- [50] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*. 3104–3112.
- [51] Mike Thelwall, Kevan Buckley, Georgios Paltoglou, Di Cai, and Arvid Kappas. 2010. Sentiment in short strength detection informal text. *JASIST* 61, 12 (2010), 2544–2558.
- [52] Lorenzo Villaruel, 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, ICSE 2016, Austin, TX, USA, May 14-22, 2016*, Laura K. Dillon, Willem Visser, and Laurie Williams (Eds.). ACM, 14–24.
- [53] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer Networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*. 2692–2700.
- [54] 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 *30th IEEE/ACM International Conference on Automated Software Engineering, ASE 2015, Lincoln, NE, USA, November 9-13, 2015*, Myra B. Cohen, Lars Grunske, and Michael Whalen (Eds.). IEEE Computer Society, 749–759.
- [55] Frank Wilcoxon. 1992. Individual comparisons by ranking methods. In *Breakthroughs in statistics*. Springer, 196–202.
- [56] 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, ACL 2020, Online, July 5-10, 2020*, Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault (Eds.). Association for Computational Linguistics, 5811–5820.
- [57] Zhizheng Wu and Simon King. 2016. Investigating gated recurrent networks for speech synthesis. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2016, Shanghai, China, March 20-25, 2016*. IEEE, 5140–5144.
- [58] 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, CIKM 2019, Beijing, China, November 3-7, 2019*. ACM, 1341–1350.
- [59] 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.

- [60] Jingyi Zhang, Masao Utiyama, Eiichiro Sumita, Graham Neubig, and Satoshi Nakamura. 2018. Guiding Neural Machine Translation with Retrieved Translation Pieces. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*. 1325–1335.