Context-Aware Model for Mining User Intentions from App Reviews

Jinwei Lu[†], Yimin Wu[†§], Jiayan Pei[‡], Zishan Qin[†], Shizhao Huang[‡] and Chao Deng[§]
[†]School of Computer Science and Engineering, South China University of Technology, Guangzhou, China
[‡]School of Software Engineering, South China University of Technology, Guangzhou, China
[§]School of Computer Science and Engineering, Guangdong Ocean University at yangjiang, Yangjiang, China
cskilljl@mail.scut.edu.cn, csymwu@scut.edu.cn, seasensio@mail.scut.edu.cn,
csqzs@mail.scut.edu.cn, se_hsz@mail.scut.edu.cn, dengchao@gdou.edu.cn

***Abstract*—Due to the highly competitive and dynamic mobile application (app) market, app developers need to release new versions regularly to improve existing features and provide new features for users. To accomplish the maintenance and evolution tasks more effectively and efficiently, app developers should collect and analyze user reviews, which contain a rich source of information from user perspective. Although there are many approaches based on intention mining that can automatically predict the intention of reviews for better understanding valuable information, those approaches are limited since contextual information of the whole review text may be lost. In this paper, we propose Mining Intention from App Reviews (MIAR), a novel deep learning model to predict the intention of app reviews automatically. We adopt a Contextual Feature Extractor to capture the context semantic information and fuse it with the local feature through a fusion mechanism. The experiment results demonstrate that MIAR has made significant improvement over the baseline approaches in Precision, Recall, and F1-score evaluation metrics, achieving state-of-the-art performance in this task. Our model also performs well in other intention mining tasks, proving its generalization ability and robustness.**

***Keywords*—Software Maintenance; Deep Learning; App Review; Intention Mining**

I. Introduction

With the highly competitive and dynamic app market, it is essential for app developers to regularly release new versions to fix bugs and offer new features to users. In order to understanding the changing user needs which are related to app maintenance and evolution, many app stores offer a review module to users, which provide a communication channel to users and developers [1]. However, manually processing such user reviews is a big challenging. Many useless reviews do not express any valuable information, or unstructured reviews use non-technical language, making review processing task become time-consuming and error-prone. Therefore, many researchers have developed a variety of approaches to automatically identifying relevant user reviews [2] [3], or by clustering and prioritizing user reviews to find the most crucial topics [4].

Apart from previous approaches which are helpful to cope with large amounts of user review, it is also important for app developers to understand the intention that users expressed implicitly in reviews [5], which could provide valuable information in detecting relevant content to accomplish several maintenance and evolution tasks. To help identify user intentions more accurately, many researchers leveraged a process called intention mining to analyze and filter user review sentences. While these researches have performed well in classifying app reviews into different user intentions, their methods based on linguistic patterns matching may waste a lot of time and manual inspection in pattern identification. Moreover, it is still difficult to accomplish this task when reviews are unstructured or intentions are implicit, which will be hard for these methods to distinguish the relevant components and patterns for classification.

In recent years, some researchers adopt deep learning to overcome this shortcoming. With the strong non-linear fitting ability, deep learning can effectively extract semantic information. Some researchers have adopted deep learning to identify intentions from various communication channels [6] [7]. Taking Huang *et al.*'s [7] work as an example, their approach builds a convolutional neural network (CNN) to classify sentences from issue tracking systems. However, they only adopt neural network to learn local representation of sentence, which ignores the contextual information of the text.

This paper conducts research based on Huang *et al.*'s work. In order to leverage intention mining to accomplish review mining task, we propose a novel model, named Mining Intention from App Reviews, which can detect intentions via neural network. To better understand the contextual information in the review, we build a context-aware model based on BiLSTM [8] that can deeply learn the global representation. To evaluate the proposed model, we conduct experiments on the review dataset to compare with state-of-the-art method, and results demonstrate the superior performance of our proposed method.

The main contributions of this paper are as follows:

- We propose a novel intention mining model for app review classification task, which does not rely on the linguistic patterns and greatly reduce the manual effort.
- We implement a contextual feature extractor to capture the global feature for better gaining the context sematic information and understanding user intentions.
- We evaluate our approach on several relevant datasets, and the result shows that our model not only outperforms previous methods in app review domain, but also have a good performance in other software engineering domains.

DOI reference number: 10.18293/SEKE2022-084

The rest of the paper is organized as follows. Section 2 briefly presents the related work of our study. Section 3 introduces the overall framework and technical details of our approach. Section 4 describes the experimental settings and presents the experiment results. Finally, Section 5 concludes the paper and outlines future work.

II. Related Work

App review is essential to app developers, as it contains valuable insights that can help successfully accomplish app maintenance and evolution tasks. Pagano and Maalej [1] identified 17 topics in user feedback by manually investigating the content of selected user reviews, which included the topics of *feature request* or *bug report* that could be mined for requirements-related information.

Many previous researchers used machine learning methods based on linguistic rules or heuristics patterns to extract such information. Iacob and Harrison [2] extracted feature requests from app reviews utilizing linguistic rules and used Latent Dirichlet Allocation (LDA) to group them. AR-Miner, which was proposed by Chen *et al.* [4], also employed LDA to group informative reviews. Maalej and Nabil [3] generated a list of keywords to be applied for the classification task. Then, they applied various machine learning methods to classify reviews. Panichella *et al.* [5] hypothesized that understanding the intention in a review has an important role in extracting useful information for developers. Therefore, they leveraged intention mining, which was proposed by Di Sorbo *et al.* [9], to catch useful contents. They merged three techniques to mining user intention for classify app reviews into the categories which are relevant to user intention. After that, they proposed AR-Doc [10], which is based on J48 algorithm, to use linguistic patterns for classification. In a later work, Di Sorbo *et al.* [11] proposed SURF to summarize app review for software change recommendation. Palomba *et al.* [12] proposed ChangeAdvisor to extract the intention of reviews to analyze potential app evolution. These tools leveraged classifier proposed by Panichella *et al.* [5], which means that intention mining can be a fundamental component to support complicated tasks. In order to minimize the manual effort of relevant pattern tagging, Di Sorbo [13] proposed NEON to automatically mine linguistic rules for review analysis and classification. However, these methods based on syntax analyzing or linguistic patterns matching limit the ability to extract semantic information from reviews, which means that the classifier could not understand the implicit intentions and just use the explicit feature to identify the category of app reviews. Therefore, to solve this problem, we leverage deep learning to model high-level abstractions in data by building neural networks with multiple layers.

In recent years, some researchers have explored the possibility of applying deep learning for intention mining. Stanik *et al.* [14] used a simple CNN-based model to classify user feedback for software development. Huang *et al.* [7] also proposed a CNN-based approach, which improves Di Sorbo *et al.*'s approach [9] and the other automated sentence classification approaches by a substantial margin. However, in their work, deep learning approaches were based on the local feature extraction model, which learns the representation of the receptive field and only calculates the relevance between adjacent n-gram elements. Hence these methods can not sufficiently consider the contextual information of the whole text, which is essential to predicting the intention of app review. Therefore, based on deep learning, we adopt the global feature extraction mechanism to learn the contextual information, hoping to achieve better performance in the review intention mining task.

III. Approach

According to Panichella *et al.* [5], user intention categories of app review can be defined as the following four classes:

- *Information Giving (IG)* : sentences that inform or update users or developers about an aspect related to the app.
- *Information Seeking (IS)* : sentences related to attempts to obtain information or help from other users or developers.
- *Feature Request (FR)* : sentences expressing ideas, suggestions or needs for improving or enhancing the app or its functionalities.
- *Problem Discovery (PD)* : sentences describing issues with the app or unexpected behaviors.

The intention of a review is predicted to be one of the four classes mentioned above. We use raw text sentences of a review as the input sequence. Suppose the input text sequence is $R = \{w_1, w_2, \ldots, w_n\}$,where *n* is the sequence length.

Figure 1 presents the overall framework of MIAR, which is mainly composed of the following five modules: (1) *Word Embedding Layer*, (2) *Local Feature Extractor*, (3) *Contextual Feature Extractor*, (4) *Fusion Layer*, and (5) *Prediction Layer*.

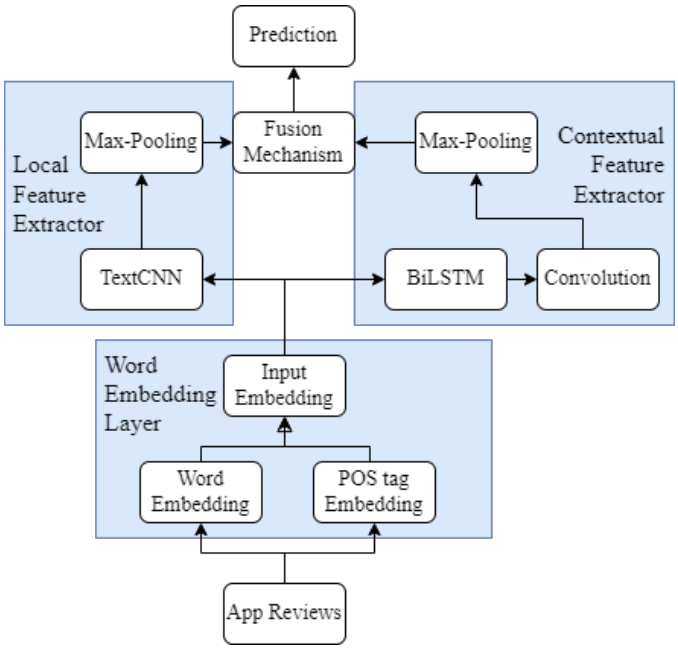

Fig. 1. The framework of MIAR

A. Word Embedding Layer

In this Layer, we leverage Word Embedding technique to transform words into the corresponding vector representations. Each word w_i in the input sequence is transformed into a vector representation $x_i \in \mathbb{R}^d$ through the pre-trained word embeddings. In our work, we use the pre-trained *GloVe* word embeddings with 300 dimensions [15]. Then, we train word vectors to obtain word embeddings. Of course, all kinds of word embedding methods can be employed in this process.

Moreover, inspired by previous work [16], making good use of POS tag can benefit semantic understanding by extracting explicit lexical information. Therefore, we add POS tag information into word vector to augment its ability of feature representation. Specifically, each type of POS tag is initialized as a random vector with uniform distribution and optimized during training. Hence, each word can be represented as:

$$x_i = [x_i^e \oplus x_i^p] \tag{1}$$

Where $\oplus$ is the concatenation operator, x_i^e and x_i^p denotes the corresponding word embedding and the embedding of the POS tag of the word, respectively.

Therefore, the review sequence containing *n* words can be converted to corresponding matrix representation, which is the input of the two feature extractors of the model:

$$Rx = x_1 \oplus x_2 \oplus \ldots \oplus x_n \tag{2}$$

B. Local Feature Extractor

In this module, we adopt TextCNN [17], which is a type of CNN for sentence classification, to extract the local feature in the text sequence. The convolutional layer receives the matrix representation Rx and performs convolution operation on it using different filters. Each filter is also a matrix, denoted as F, having the same width as the matrix R. The purpose of each filter with height *f* is to capture the semantic feature of each n-gram sequence in the sentence through a convolution operation, which is computed as follows:

$$C_i = F \cdot R_{i:i+f-1} + b_i, \forall i \in \{1, 2, \ldots, n-f+1\} \tag{3}$$

Where $R_{i:i+f-1}$ represents the sub-matrix of R from the *i* th row to the (*i+n-1*) th row, which represents the vectors of *f* continuous words (n-gram), and b_i is a bias value.

The *j* th filter is applied repeatedly to each n-grams in the sentence with convolution operation and produces a vector:

$$F_j = [C_1, C_2, \ldots, C_{n-f+1}], \forall j \in \{1, 2, \ldots, m\} \tag{4}$$

Where *n* is the height of matrix R_x. In order to help the extractor to learn enough semantic features in different granularities, multiple filters with various heights are used.

After performing the convolution operation, a pooling layer is applied to reduce the number of parameters and the computation cost. Specifically, the pooling layer applies a 1-max pooling function to the vector F_j received from each filter. Then outputs of *m* filters are concatenated as a high-level feature vector, which can catch semantic features of different n-grams in the input, representing the local feature vector *LF*:

$$LF = maxpooling([F_1; F_2; \ldots; F_m]) \tag{5}$$

C. Contextual Feature Extractor

To solve the problem of Local Feature Extractor that is not sensitive to the sequential information which is essential to understand the semantic relation and implicit intention, we build a Contextual Feature Extractor to extract the contextual information and generate the global representation of the input sequence. Here, we adopt BiLSTM to incorporate the contextual information into the original representation of each token in input sequence. BiLSTM is composed of a forward and a backward LSTM. Through its three gate structures, LSTM can solve the long-term dependence information very well, which means that bi-directional semantic dependencies within the review can be well captured. We concatenate the outputs of the two LSTM to generate the augmented vector of a token, which incorporates the contextual information into the token representation. This process can be represented as:

$$\overrightarrow{h}_i = LSTM(\overrightarrow{h}_{i-1}, x_i), \forall i \in \{1, 2, \ldots, n\} \tag{6}$$

$$H_i = [\overrightarrow{h}_i; \overleftarrow{h}_i], \forall i \in \{1, 2, \ldots, n\} \tag{7}$$

Where $\overrightarrow{h}_i$ is the hidden state of the forward LSTM in the time step *i*, $\overleftarrow{h}_i$ represents the backward, x_i is the input of LSTM in the time step i, and H_i is the contextual representation.

To condense the rich information extracted by BiLSTM, we add a convolutional layer to abstract the contextual feature. The feature vector transferred from BiLSTM does not contain sequential information incorporated into representation, which is suitable for convolutional network to perform feature condensing. Then a max-pooling layer is leveraged to conduct feature dimension reduction and generate the global representation. This global feature vector contains high concentration contextual information that can help mining intention better. The global feature vector *GF* can be calculated as follow:

$$HC_i = F_h \cdot H_{i:i+f-1} + b_i, \forall i \in \{1, 2, \ldots, n-f+1\} \tag{8}$$

$$F_j = [HC_1, HC_2, \ldots, HC_{n-f+1}], \forall j \in \{1, 2, \ldots, m\} \tag{9}$$

$$GF = maxpooling([F_1; F_2; \ldots; F_m]) \tag{10}$$

Where $H_{i:i+f-1} = [H_i, \ldots, H_{i+f-1}]$, *f* is filter size, *m* is the number of filters, and b_i is a bias value. Then the final representation is passed to the fusion layer for feature fusion.

D. Fusion Layer

In the Fusion Layer, *LF* and *GF* are integrated to generate the intention representation. We fuse the features as follow:

$$\tilde{X}_i^1 = \alpha LF_i + (1-\alpha) GF_i \tag{11}$$

$$\tilde{X}_i^2 = \beta LF_i - (1-\beta) GF_i \tag{12}$$

$$\tilde{X}_i^3 = \gamma GF_i - (1-\gamma) LF_i \tag{13}$$

Where $+$ represents feature augment computation, which highlights the similarity between two vectors, and $-$ denotes the difference between two vectors. α, β, and γ are the weighting factors which can be tuned for controlling the fusion degree

of each feature. We concatenate the results obtained by the above three methods and input them into another single-layer feed-forward network F to compute the output:

$$\tilde{X}_i^1 = F([\tilde{X}_i^1; \tilde{X}_i^2; \tilde{X}_i^3]) \tag{14}$$

Where $[;]$ refers to the concatenation operation. Then the final feature representation is passed to the Prediction Layer.

E. Prediction Layer

In the Prediction Layer, for vectors obtained from last layer, we use a multi-layer feed-forward network L to get feature vectors. Then a softmax function is applied to normalize the values so that the output vector can represent the probability of the input sequence belonging to one specific category:

$$\sigma(V_i) = \frac{e^{V_i}}{\sum_{j=1}^{K} e^{V_j}}, \forall i \in 1, 2, \ldots, K \tag{15}$$

Finally, we use the cross-entropy function to measure the loss between the prediction result and the ground truth:

$$Loss = -(y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})) \tag{16}$$

IV. Experiment

In this section, we conduct some experiments for answering the following research questions.

A. Research Questions

RQ1: How effective is MIAR for predicting user intention of different categories in app reviews?

MIAR adopts deep learning to automatically mining review intention based on two feature extractors and the fusion mechanism, which is much different from the previous works. To investigate the effectiveness of our approach, we compare the performance of MIAR with the baseline from Panichella *et al.*'s work [5]. Moreover, we also conduct experiments with other three intention mining models including AR-Doc [10], DECA [9], and Huang *et al.*'s CNN-based model [7]. To present results more accurately, we keep results of all models to three decimal places.

RQ2: How much influence do the techniques or modules we proposed contribute to the improvement of MIAR?

Three important techniques we proposed, including POS tag embedding (POS), Contextual Feature Extractor (CFE), and Fusion Layer (FL), could help MIAR to capture semantic information and identify intention from app reviews better. To evaluate their contributions, we conduct an ablation study to demonstrate. We take turns to remove one of the three techniques and compare the revised model with the original model on the *F1-score*. Specifically, when we remove CFE, we must remove FL simultaneously, since the fusion mechanism needs global feature generated by CFE to perform the fusion process. Thus, we perform the following ablation studies, consisting in (1) only removing POS; (2) only removing FL and replacing by concatenation; and (3) removing the CFE.

RQ3: Does MIAR work well in intention mining tasks from other software engineering domains?

TABLE I
Details of Datasets

Domain	IG	IS	FR	PD	Other	Total
Review	583	101	218	488	31	1421
Issue	1,328	962	536	762	397	3,985
Email	167	264	187	170	0	788

To explore the generalization ability of MIAR, we apply MIAR to emails mining task [9] and issue mining task [7] for comparison. Following Huang *et al.*'s work, we conduct the following studies, including (1) only using issue dataset (Issue); (2) only using email dataset (Email); (3) using issue dataset as training set and email dataset as test set (I to E); (4) using email dataset as training set and issue dataset as test set (E to I). The results of baseline models are from paper [7].

B. Dataset

We carry out experiments on the review dataset built by Panichella *et al.* [5]. They sampled 1421 review sentences out of 7696 reviews and manually labeled the sample according to the categories of their intention taxonomy, which includes four intention classes and *Other* class. We use 3-folds cross-validation to carry out our experiments on this dataset.

Moreover, to evaluate the generalization ability of MIAR, we also run experiments on two intention mining datasets proposed by Di Sorbo *et al.* [9] and Huang *et al.* [7]. We also carry out 3-folds cross-validation as Huang *et al.* [7] to evaluate the performance of MIAR. The detailed data of these three datasets is presented in Table I.

C. Evaluation Metric

We use the same evaluation metrics as Huang *et al.* [7] to evaluate MIAR's performance.

Precision represents the proportion of samples predicted to be positive that are truly positive samples.

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive} \tag{17}$$

Recall represents the proportion that the positive samples are predicted to be positive correctly.

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative} \tag{18}$$

F1-score is the weighted average of *Precision* and *Recall*. This metric takes into account both the *Precision* and *Recall* of the model. In the multi-class classification task, *F1-score* are computed for each class and then averaged via arithmetic mean to get *Macro-F1*.

$$F1 - score = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{19}$$

D. Implementation Details

MIAR is implemented based on the PyTorch [18] framework, experimented on an Nvidia 1080Ti GPU. For the Local Feature Extractor, we set two different kernel sizes, which are 2 and 3, respectively, and 100 feature maps for each kernel.

TABLE II
MAIN RESULT

	IG			IS			FR			PD					
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	Avg-P	Avg-R	Macro-F1
Baseline	0.680	**0.904**	0.776	0.712	0.684	0.698	**0.704**	0.225	0.341	**0.875**	0.776	0.823	0.743	0.647	0.659
DECA	0.327	0.730	0.451	0.723	0.640	0.679	0.516	0.281	0.364	0.733	0.810	0.769	0.575	0.615	0.566
Huang	0.677	0.788	0.728	0.793	0.396	0.528	0.512	0.401	0.450	0.745	0.717	0.731	0.682	0.575	0.609
AR-Doc	0.456	0.747	0.566	0.683	0.633	0.657	0.625	0.385	0.476	0.751	0.808	0.779	0.629	0.643	0.620
MIAR	**0.750**	0.880	**0.810**	**0.871**	**0.692**	**0.772**	0.578	**0.646**	**0.610**	0.829	**0.836**	**0.833**	**0.757**	**0.764**	**0.756**

TABLE III
COMPARING THE F1-SCORE OF MIAR AND THE REVISED MODELS

	IG	IS	FR	PD	Macro-F1
MIAR	**0.810**	**0.772**	**0.610**	**0.833**	**0.756**
MIAR(-POS)	0.761	0.757	0.570	0.787	0.719
MIAR(- FL)	0.748	0.724	0.531	0.752	0.689
MIAR(- CFE -FL)	0.703	0.664	0.488	0.728	0.646

TABLE IV
PERFORMANCE OF DIFFERENT MODELS IN OTHER DATASETS

		IG	IS	FR	PD	Macro-F1
Issue	DECA	0.293	0.511	0.420	0.601	0.456
	Huang	0.805	0.904	0.794	0.820	0.831
	MIAR	**0.842**	**0.980**	**0.846**	**0.829**	**0.874**
Email	DECA	0.743	0.874	0.789	0.879	0.821
	Huang	0.785	0.883	0.793	**0.890**	0.838
	MIAR	**0.804**	**0.954**	**0.842**	0.818	**0.854**
I to E	DECA	**0.743**	0.874	**0.789**	**0.879**	**0.821**
	Huang	0.562	0.808	0.579	0.760	0.678
	MIAR	0.659	**0.943**	0.693	0.811	0.776
E to I	DECA	0.293	0.511	0.420	0.601	0.456
	Huang	0.487	0.678	0.579	0.520	0.566
	MIAR	**0.684**	**0.890**	**0.670**	**0.746**	**0.747**

For the Contextual Feature Extractor, we use BiLSTM with 700 units as the encoder. Adam optimizer [19] with an initial learning rate of 0.001 is applied. We release the source code of MIAR and hope to facilitate future researches.[1]

E. Result

RQ1: How effective is MIAR for predicting user intention of different categories in app reviews?

Table II presents the experiment results. The best results are highlighted in bold. We can see that MIAR achieves the best results for all the four intention classes, with an average of 75.7%, 75.4%, and 75.6% in *precision*, *recall*, and *F1-score*. Although Baseline method has a better performance of *precision* in some classes, the *F1-score* of this model is approximately 9.7% below than that of MIAR, which is limited by its low *recall*, which is 11.7% below than MIAR. This result indicates that using the approach based on linguistic pattern can predict intentions precisely, but this method may be confused by some ambiguous patterns which can appear in different intentions. Moreover, these tools relies heavily on manual effort. The lack of relevant linguistic pattern for matching can seriously degrade the performance of these tools. In contrast, deep learning methods, which do not rely on the fixed pattern, can model the high-level abstractions of reviews to understand the whole sentence and achieve higher score.

The result of *FR* is lower than other classes, which indicates that the semantic information of this class is more confused for understanding, or the expression is more implicit. So that Baseline and DECA are difficult to identify patterns for classification, which leads to a poor result (34.1% and 36.4%). This result reflect that this class limits the overall performance of all approaches included MIAR, which still have a major improvement of *F1-score* (at least 13.4%) compared with other approaches. For the remaining three classes, MIAR's improvement is also significant $(8.2\%-24.4\%)$ compared with Huang *et al.*'s approach. This result indicates that extracting contextual features and semantic relations is essential for understanding and identifying user intentions, especially in the noisy and informal communication environment.

RQ2: How much influence do the techniques or modules we proposed contribute to the improvement of MIAR?

The experimental results are shown in Table III. After removing POS or FL, respectively, the revised models' performance decreases in some degree ($3.7\%and6.7\%$, in terms of *Macro-F1*, respectively). However, after removing the CFE, the model's performance decreases more obviously $(11.0\%$ in *Macro-F1*), especially in predicting the *Feature Request* class (12.2% in terms of *F1-score*). This experiment results prove the importance of the CFE, which plays an essential role in predicting review intention. Certainly, POS and FL also improve our model's performance.

RQ3: Does MIAR work well in intention mining tasks from other software engineering domains?

Table IV presents the experiment results. For Issue, Email and E to I, MIAR outperforms other models, which include the state-of-the-art model on these datasets. For I to E, MIAR has a performance degradation, due to the poor performance in *IG* and *FR*, which are also the bottleneck of other models. This experimental result may cause by the gap between different communication channels. The discussion from issue is more similar to app review in the statement, which has more verbal and indirect expression that understanding the contextual information is more important. Moreover, MIAR achieves better results in all the four experiments than Huang *et al.*'s model,

[1]Our model is openly available in https://anonymous.4open.science/r/MIAR/

TABLE V
CASE STUDY

Case	Review	AR-Doc	Huang	MIAR	Label
1	**Crashing Bug** Normally I will be able to find a work Around but I couldn't get a ROM to run.	PD	PD	FR	FR
2	And I'm also not sure how to search for hybrid cards am I **missing something** here	PD	PD	IS	IS

which shows that MIAR has good generalization ability that can be applied to other software engineering domains.

F. Case Study

To investigate how our architecture makes a difference in details, we visualize two examples from different classes in Table V. The most important phrases extracted by CNN-based architectures are highlighted in bold, and we underline the important contextual information extracted by CFE.

In case (1), AR-Doc captures the linguistic pattern: "*someone get something*", and this pattern generally apply to express some problems. So that the linguistic pattern belongs to *PD*, leading to the wrong predicting. Huang *et al.*'s approach extracts the most important phrase "*Crashing Bug*", which generally indicates some bugs. Thus their model misclassifies this review into *PD*. In contrast, MIAR not only captures the important local feature"*Crashing Bug*", but also extracts some contextual information: "*couldn't get*" and "*run*", which belong to long distance dependency and could be ignored by Local Feature Extractor. These contextual features can provide more semantic information for predicting the correct label.

In case (2), both AR-Doc and Huang *et al.*'s approach are disturbed by the influential local feature "*missing something*", which is inclined to report some errors appeared in the app, while MIAR can consider the helpful contextual feature "*not sure*" and "*search for*", which is effective on understanding the implicit intention and predicting *IS* correctly. In short, with the help of CFE, MIAR can extract more useful information and provide to the classifier for efficiently mining intentions.

V. CONCLUSION AND FUTURE WORK

In this paper, we propose a deep learning model MIAR based on the feature fusion mechanism to predict the user intention from app review, which can reduce the manual effort and better help developers obtain useful information for software maintenance and evolution. The experiment results show MIAR's effectiveness and consistency in predicting review intention, outperforming some baseline models in previous works. Moreover, in some other intention mining tasks, MIAR can also achieve state-of-the-art performance, which proves its generalization ability. We will explore the application that identify the intention of other written communication channels from software engineer domain in future work.

REFERENCES

[1] D. Pagano and W. Maalej, "User feedback in the appstore: An empirical study," in *2013 21st IEEE international requirements engineering conference (RE)*. IEEE, 2013, pp. 125–134.

[2] C. Iacob and R. Harrison, "Retrieving and analyzing mobile apps feature requests from online reviews," in *2013 10th working conference on mining software repositories (MSR)*. IEEE, 2013, pp. 41–44.

[3] W. Maalej and H. Nabil, "Bug report, feature request, or simply praise? on automatically classifying app reviews," in *2015 IEEE 23rd international requirements engineering conference (RE)*. IEEE, 2015, pp. 116–125.

[4] N. Chen, J. Lin, S. C. Hoi, X. Xiao, and B. Zhang, "Ar-miner: mining informative reviews for developers from mobile app marketplace," in *Proceedings of the 36th international conference on software engineering*, 2014, pp. 767–778.

[5] S. Panichella, A. Di Sorbo, E. Guzman, C. A. Visaggio, G. Canfora, and H. C. Gall, "How can i improve my app? classifying user reviews for software maintenance and evolution," in *2015 IEEE international conference on software maintenance and evolution (ICSME)*. IEEE, 2015, pp. 281–290.

[6] M. Haering, C. Stanik, and W. Maalej, "Automatically matching bug reports with related app reviews," in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 2021, pp. 970–981.

[7] Q. Huang, X. Xia, D. Lo, and G. C. Murphy, "Automating intention mining," *IEEE Transactions on Software Engineering*, vol. 46, no. 10, pp. 1098–1119, 2018.

[8] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[9] A. Di Sorbo, S. Panichella, C. A. Visaggio, M. Di Penta, G. Canfora, and H. C. Gall, "Development emails content analyzer: Intention mining in developer discussions (t)," in *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2015, pp. 12–23.

[10] S. Panichella, A. Di Sorbo, E. Guzman, C. A. Visaggio, G. Canfora, and H. C. Gall, "Ardoc: App reviews development oriented classifier," in *Proceedings of the 2016 24th ACM SIGSOFT international symposium on foundations of software engineering*, 2016, pp. 1023–1027.

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

[12] F. Palomba, P. Salza, A. Ciurumelea, S. Panichella, H. Gall, F. Ferrucci, and A. De Lucia, "Recommending and localizing change requests for mobile apps based on user reviews," in *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*. IEEE, 2017, pp. 106–117.

[13] A. Di Sorbo, S. Panichella, C. A. Visaggio, M. D. Penta, G. Canfora, and H. C. Gall, "Exploiting natural language structures in software informal documentation," *IEEE Transactions on Software Engineering*, vol. 47, no. 8, pp. 1587–1604, 2021.

[14] C. Stanik, M. Haering, and W. Maalej, "Classifying multilingual user feedback using traditional machine learning and deep learning," in *2019 IEEE 27th International Requirements Engineering Conference Workshops (REW)*. IEEE, 2019, pp. 220–226.

[15] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[16] L. Shi, M. Xing, M. Li, Y. Wang, S. Li, and Q. Wang, "Detection of hidden feature requests from massive chat messages via deep siamese network," in *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*. IEEE, 2020, pp. 641–653.

[17] Y. Kim, "Convolutional neural networks for sentence classification," *Eprint Arxiv*, 2014.

[18] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[19] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.