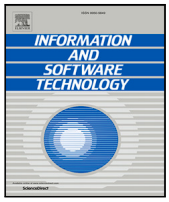
Information and Software Technology 160 (2023) 107228

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof



Test case classification via few-shot learning

Yuan Zhao, Sining Liu, Quanjun Zhang, Xiuting Ge, Jia Liu *

State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, 210093, China

ARTICLE INFO

Keywords:
Test case classification
Few-shot learning
Attention mechanism

ABSTRACT

Context: Crowdsourced testing can reduce testing costs and improve testing efficiency. However, crowdsourced testing generates massive test cases, requiring testers to select high-quality test cases for execution. Consequently, crowdsourced test cases require much effort to perform labeling due to the costly manual labor and domain knowledge.

Objective: Existing methods usually fail to consider the crowdsourced testing scenario's inadequate and imbalanced data issues. We aim to effectively and efficiently classify many crowdsourced test cases for developers to alleviate manual efforts.

Method: In this paper, we propose a test case classification approach based on few-shot learning and test case augmentation to address the limitations mentioned above. The proposed approach generates new test cases by the large pre-trained masked language model and extracts embedding representation by training word embedding models. Then a Bidirectional Long Short-Term Memory (BiLSTM)-based classifier is designed to perform test case classification by extracting the in-depth features. Besides, we also apply the attention mechanism to assign high weights to words that represent the test case category by lexicon matching.

Results: To verify the effectiveness of the classification framework, we select 1659 test cases from three crowdsourced testing projects to conduct in-usability evaluation experiments. The experimental results show that the proposed approach has a higher accuracy and precision rate than existing classification methods.

Conclusion: It can be concluded that (1) the proposed approach is an effective test case classification technique for crowdsourced testing; (2) the proposed approach is practical to help developers select high-quality test cases quickly and effectively.

1. Introduction

Crowdsourced testing is freely outsourced test tasks to crowdworkers, who are recruited from mass potential and undefined online users [1]. Due to the large-scale and real usage scenarios from users, crowdsourced testing has attracted considerable attention and been widely used in various communities, such as Quality of Experience (QoE) testing [2,3], GUI testing [4,5], and usability testing [6,7].

Crowdsourced testing mainly contains crowdworkers, the testing platform, and the test task publishers. In particular, when the test task publishers distribute test tasks to crowdworkers, crowdworkers perform test tasks via test cases. Test cases are the core part of software testing, and excellent test cases can guarantee software testing quality. Crowdsourcing testing is an emerging trend in software testing. Different from traditional testing [8–10], crowdsourcing testing makes full use of the advantages of crowdsourcing and cloud platforms and has the characteristics of fast testing speed. The industry highly favors low testing costs. It is very suitable for companies and individuals who lack professional testers and standardized testing. The crowdsourced testing process generates many test cases. Due to the different levels of crowdworkers, testers often need to screen the test case set and select suitable cases from them for inspection and later use.

Test cases for crowdsourced tests usually have many characteristics, including sparse category labels, uneven distribution of category labels, and incorrect category labels. It is sometimes complicated for testers to distinguish the test cases and judge the accuracy of the use case classification, resulting in poor reuse of test cases for crowdsourcing testing [11]. Because of the special complexity of crowdsourced test cases, the traditional test case optimization techniques [12–14] cannot be applied to crowdsourced testing. Although it is accurate to relabel or change these labels manually, it is inefficient and difficult to maintain a high accuracy rate when the set of test cases is extensive. To address these problems, we propose to refine test case content and identify test case categories by generating reasonable labels to assist testers in better selecting and utilizing existing test case sets.

* Corresponding author.
E-mail addresses: zhaoyuan@smail.nju.edu.cn (Y. Zhao), 181250093@smail.nju.edu.cn (S. Liu), quanjun.zhang@smail.nju.edu.cn (Q. Zhang), dg20320002@smail.nju.edu.cn (X. Ge), jialiu@nju.edu.cn (J. Liu).

https://doi.org/10.1016/j.infsof.2023.107228
Received 23 September 2022; Received in revised form 8 March 2023; Accepted 7 April 2023
Available online 13 April 2023

Table 1
Example of crowdsourced test cases.

id	TC7
name	Review use case tag tests
product_version_module	Test case management classification system - Review test case tags
premise_and_constraint	Users are authorized and authenticated
test_process	1. User login for reviewer account 2. Click the "Review Test Case Tags" button in the menu bar 3. Click the "View" button of a test case in the test case list 4. Click the "Edit Tags" button 5. Click the "Finish" button after modifying or not modifying the label
test_requirement	1. Display menu bar specific to reviewer permissions 2. Show the list of test cases to be reviewed for tags 3. Display the test case details page of the pending review tab 4. The test case details page enters edit mode; at this time you can edit the tag information, but you cannot edit other test case information 5. The system pops up the "audit completed" prompt 6. The status of the test case is changed to "Approved" or "Tag failed," and disappears from the list of tags to be reviewed
...	...

In crowdsourced testing, the crowdworker will manually create test cases for a given test target, and a test case set is a collection of all test cases under the same test target [15,16]. Each test case contains several attributes such as test case name, test process, test case description, label, and test target category [17]. In particular, the label attribute indicates the keyword set of the test case, and the test target category attribute refers to the category of the proposed approach module tested by the test case, such as functional integrity, user experience, page layout, abnormal exit, etc. The generalized labels in the proposed approach usually include the two specific attributes of labels and test target categories.

In this paper, we propose a novel test case classification technique to address the lack of inaccurate category labels for test cases in crowdsourced testing scenarios. We perform test case label extraction and category classification based on test case text analysis and few-shot learning. In particular, the proposed approach first augment existing test case based on SimBERT [18] and transform them into vector representation by embedding techniques. Then we design a BiLSTM-based classifier to extract the deep features of test cases and get classification results. We also apply the attention mechanism to the important words that can represent the corresponding test case categories based on the design lexicon matching method.

The experiment results demonstrate that the proposed approach can achieve better performance than existing methods in terms of accuracy, precision, recall, and F1-measure. Moreover, the ablation experiment proves the effectiveness of the designed test case augmentation. Besides, we also conduct a human study to prove the usefulness of the proposed approach for helping developers classify test cases quickly and effectively.

In summary, we make the following contributions.

- **Novel Technique.** We propose a novel test case classification technique that adopts few-shop learning and test case augmentation.
- **Extensive Study.** We conduct a large-scale experiment to evaluate the effusiveness and usefulness of the proposed approach. The results show that the proposed approach can classify the test cases effectively and efficiently.
- **Available artifacts.** We release the relevant materials (including source code, patches, and results) used in the experiments for replication and future research.[1]

[1] All artifacts relevant to this work can be found at https://git.mooctest.com/Sjim/exam-question-classification accessed February 2023.

The remainder of this paper is organized as follows. Section 2 discusses basic concepts involved in this work. Section 3 introduces the design of our approach in detail. Section 5 presents the evaluation results of the proposed approach. Section 6 presents threats to validity. Section 7 discusses the related work. We summarize the paper and give possible directions for future work in Section 8.

2. Basic concepts

2.1. Crowdsourced test cases

Crowdsourced test cases usually consist of multiple attributes. Common test case attributes include id, name, method, design time, designer, end condition, product_version_module, premise and constraint, test process, test requirement, adoption of the guideline, etc. Table 1 shows a brief test case. In professional testing scenarios, testers often manually record used test cases in text form for later reuse. As the style of test cases recorded by different organizations varies, we have selected three attributes, product_version_module, test process, and test requirements which are also called expected results, that will be included in all test cases for further analysis and classification. The product_version_module helps to locate the function and location of the test case application, and the test process and test requirements have a large number of test-related vocabulary inside, which can help find the relevant types of tests.

2.2. Attention mechanism

The attention mechanism refers to focusing on the information that is more critical to the task among many inputs to the model and reducing attention to other information. We combine a Bi-directional Long Short Term-Memory with an attention mechanism and a lexicon for helping to solve the problem of sparse annotation data. Important words can be given higher attention weights during text classification due to the attention mechanism. Table 2 is an example of a test case for login and logout functions. Based on the attention mechanism, we can select some strongly related words to the category based on high weights to build a lexicon to help generate pseudo-labels.

2.3. Lexicon

We use an attention-based BiLSTM to construct a set of critical words for text classification. Based on the attention weights of each category, the attention-based BiLSTM extracts a set of relevant words. We select the top n unlabeled data with the highest confidence level from each category. Then, we select the m words with the highest

Table 2
Example of attention words.

Words	Attention score
Click	0.035
Quit	0.37
Login	0.43
...	...
Open	0.004

attention weights from each of the selected n data and produce a set of n words consisting of m words. We consider the collected word sets as the lexicon of the corresponding categories. Table 3 shows an example of the set of keywords in each lexicon obtained from the initially trained classifier.

2.4. Pseudo-labeled test cases

Pseudo-labeled test cases are test cases that are originally unlabeled and, after the model iteration, have labels predicted by the classifier or lexicon. Pseudo-labeled test cases can be added to the training set and used as the original labeled cases to train the classifier. Experiments [19] have shown that expanding the training set with pseudo-labeled test cases results in better classification of the trained classifier. Table 4 shows the categories predicted by the classifier and the lexicon for the test case, respectively, and the confidence of the categories predicted by the classifier. According to the pseudo-label assignment rules specified in Part 4, it assumes the confidence threshold $q = 0.9$ and matching numbers of words threshold $k1 = 3$ and $k2 = 4$. For TC_1, the classifier prediction confidence is less than 0.9. The matching numbers of words are equal to $k2$, so the lexicon prediction category is assigned to TC_1 as a pseudo-label. For TC_2, the classifier prediction confidence is greater than 0.9, and the matching numbers of words are equal to $k1$, so the classifier prediction category is assigned to TC_2 as a pseudo-label.

3. Approach

The proposed approach can be divided into three stages, including the data preparation stage, test case classification stage, and lexicons collection stage. The data preparation stage performs data augmentation and enhancement on the test case data and then pre-processes the augmented data set with word separation, lexical filtering, deactivation, synonym conversion, etc. The pre-processed keyword text is trained with word vector and bag-of-words models. The primary function of the test case classification stage of the test case is to train the classification model and provide classification results. The primary function of the lexicons collection stage is to collect the words that can represent a category noticed during the classifier's training, form a lexicon, and assist the classifier in determining the test case category. The overall framework is shown in Fig. 1.

3.1. Data preparation

The primary function of the data preparation stage is to process the test case data acquired by the test case management module and save the processed intermediate data in the file frame. The raw test case data obtained is difficult to be understood by the computer, so after filtering out the dirty data, features are extracted from the test cases and used in the classification model. For the stock test cases, this module first uses them for a round of data augmentation to enrich the dataset and then transforms them into word vectors using Language Technology Platform (LTP) [20], the Word2Vec method, and the Word Embedding method for training the classification model. LTP is a natural language processing platform developed by Harbin Institute of Technology, which provides a series of Chinese natural language processing tools that can be used by users to perform word separation, lexical annotation, syntactic analysis, and other tasks on Chinese text. Since we need to perform a word splitting operation on the text in the test case, i.e., splitting sentences into words, LTP is a suitable choice.

Algorithm 1 Data Preparation

Input: case set S, length of Dictionary $l1$, embedding size $l2$
Output: The vectors of test cases $M2$
1: $S \Leftarrow$ Clean(S) {Function Clean(used to filter test case set}
2: **for** each test case $t \in S$ **do**
3: $S1 \Leftarrow$ SimBERT(t)
4: **end for**
5: **for** each test case $t1 \in S1$ **do**
6: $S2 \Leftarrow$ LTP($t1$)
7: **end for**
8: **for** each test case $t2 \in S2$ **do**
9: $D1 \Leftarrow$ wordcount($t2$)
10: **end for**
11: $D2 \Leftarrow$ convert_word_to_index.($D1$)
12: **for** each test case $t2 \in S2$ **do**
13: $M1 \Leftarrow$ Word2Vec($t2, D1$)
14: **end for**
15: $M2 \Leftarrow$ embedding($M1, l1, l2$)
16: **return** $M2$

The whole data preparation stage is processed as shown in Algorithm 1. Firstly, the proposed approach filters the test case data, such as test cases with test categories or program exceptions, and test cases with empty test requirements, and removes test cases with pending review status (Line 1). After that, the data of test cases will be augmented. In the process of augmenting test cases, we need to generate text cases that are similar to existing test cases. The SimBERT [18] model based on the BERT model can generate semantically similar text based on the existing text, so we think SimBERT can help us to perform test case augmentation. The proposed approach uses the SimBERT model to generate text with semantic similarity to the test cases, expand the number of test cases for subsequent neural network training, and save the expanded test case text to the file box (Lines 2–4). Immediately after that, the proposed approach uses LTP to segment the test case text data into words, removes the deactivated words, filters the words that are not verbs or nouns, and unifies the synonyms for feature extraction (Lines 5–7). After that, the proposed approach will count the number of word occurrences in S2 and generate a sorted dictionary D1, where Key is the word and Value is the word's occurrence number. Then the proposed approach generates dictionary D2, where the Key is the word, and the Value is the word's index (Lines 8–11). After generating a dictionary, the proposed approach trains the Word2Vec model on historical test case data, builds word packages, and saves them. Finally, the word embedding vectors of the test cases are obtained on the Word2Vec model, and the word embedding vectors corresponding to the test cases are saved to the file box to complete the process (Lines 12–16).

3.2. Test case classification

In the test case classification phase, the proposed approach takes the test case word embedding as input and trains a BiLSTM model with an attention mechanism as the test case classification model. The model can provide the classification results of the test cases and the corresponding attention values of each word in them.

We implement the BiLSTM model with an attention mechanism for classifying test cases in our proposed approach. The model takes as input the word vectors obtained in the previous steps. We divide the dataset into a training set, a test set and an unlabeled development set. The training set and test set are used for training and testing

Table 3
Example of Lexicons.

Type	Lexicon
Crash	Logout, Failure, Account, Exception.....
Functional Requirements	Problem, Lack, Can't, Need, Function.....
User Experience	Error, Verification, Hint, Correct, Experience.....
Page Layout	Display, Below, Interface, Buttons, Layout.....
Performance	Slow, Network, Latency, Fluctuation.....
Security	Risks, Alarms, Triggers, Changes, Normal, Abnormal.....

Table 4
Tester classification quality review score.

Test case	Correct label	Classifier's prediction	Confidence of the classifier's prediction	Lexicon's prediction (matching numbers of words)
TC_1	1	1	0.85	1(4)
TC_2	3	3	0.96	3(3)
TC_3	2	2	0.74	4(5)
TC_4	5	5	0.91	3(1)
...	...	...	...	...
TC_n	1	2	0.88	1(3)

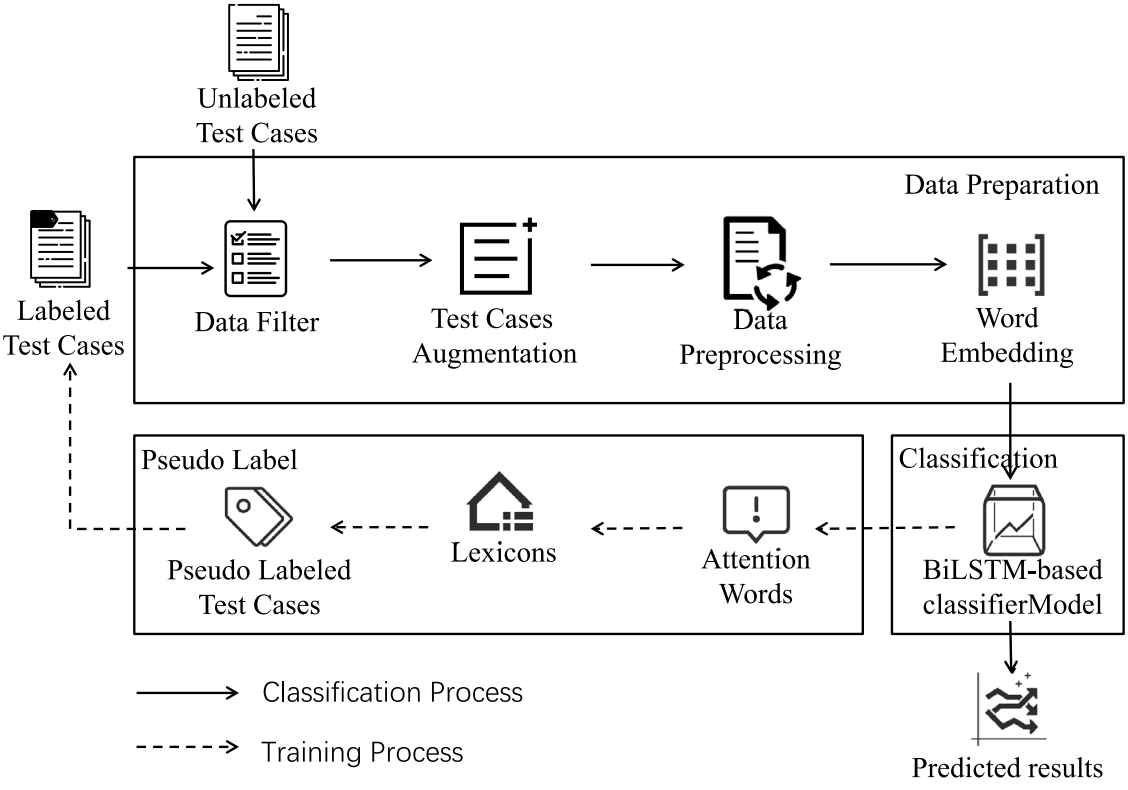


Fig. 1. The framework of few-shot-based test case classification.

the classification model, respectively, while the development set is used for generating the lexicon. In brief, since the development set is unlabeled, the proposed approach uses both classification models and lexicon matching to generate two pseudo-labels for the test cases in the development set. Then, based on several conditional judgments to determine which pseudo-label is closest to the true label, this label is given to that test case. Afterward, the word vectors that can be added to the lexicon are selected based on the prediction confidence and attention values. The test cases with the new labels are moved from the development set into the training set, resulting in a larger training set. This new training set is used to train the classification model to further improve the accuracy of the classification model. This process is repeated until no new test cases are added to the training set. This process is described in detail in Section 3.3. After several training rounds, the initial classification model with the highest classification accuracy on the final test set is obtained.

In the classification model of this paper, the attention mechanism assigns a weight to each word, representing the importance of the word in the test case. A softmax function is also used to normalize the attention weights of each word to represent the importance of each dimension (each word) in the word vector of the test case in this prediction category. The set of vectors input to the BiLSTM layer is denoted as $H : [h_1, h_2, \cdots, h_T]$, where T is the sentence length. The weight matrix r obtained from its Attention layer is given by

$$M = tanh(H) \tag{1}$$

$$\alpha = softmax(w^T M) \tag{2}$$

$$\gamma = H\alpha^T \tag{3}$$

where $H \in R^{d^\omega \times T}$, d^w is the dimension of the word vectors, ω^T is a transpose of the parameter vector obtained from training learning. The dimension of ω, α, γ is d^ω, T, d^ω separately.

The final sentence-pair representation used for classification is eventually obtained from:

$$h = \tanh(\gamma) \tag{4}$$

3.3. Lexicon collection

Important words tend to have a greater impact on the confidence of the test cases during the classification process, and the attention mechanism is similar in that it assigns higher weights to the inputs that are more capable of characterizing the data classes. Even if the performance of the initial classifier is poor, the classifier assigns higher weights to the essential words with high confidence in the predicted data. Therefore, this stage combines a deep learning classifier and lexicon matching, which can solve the problem of sparsely labeled data to some extent. By using the attention weights given by a trained classifier based on BiLSTM to obtain a broader and reliable training set, the unlabeled data in the development set are transformed into data with algorithmically assigned pseudo-labels and added to the training set.

With the attention-based BiLSTM model, the proposed approach obtains the attention weights of each word in the test cases. In the lexicon collection phase, the proposed approach will rank the words in each category according to the attention weights and collect the representative words for each category in the test cases. In the case of insufficient prediction confidence of the classification model, the proposed approach will determine the test case categories by the weights of the representative words, thus helping to improve the classification accuracy.

In this stage, the proposed approach first adds the attention layer to the BiLSTM neural network to obtain the attention weights of the word vectors. After that, the proposed method uses word packets obtained from Word2Vec in order to obtain the correspondence between the word vectors in the word packets and the actual words, which are obtained by feature extraction during the data preparation phase. The flow of the whole lexicon collection stage is shown in Algorithm 2.

Algorithm 2 has multiple inputs, including $V1$ (word vectors of the test cases in the development set), $V2$ (word vectors of the test cases in the training set), $q = 0.9$, $k1 = 3$, $k2 = 8$, where q, $k1$ and $k2$ are conditional parameters used in the lexicon matching process. Eventually, Algorithm 2 will result in a model M that has been retrained.

At the very beginning of Algorithm 2, the lexicon(l), the test case confidence dictionary $D2_i$ of class i, and the vector of transfer set($V3$) are initialized to empty. The proposed approach obtains the initial classification model(M) saved in the test case classification module, i.e., the BiLSTM model with attention mechanism, and trains the model using training set ($V2$) (lines 1–4).

After that, the development set($V1$) is used as the input to the classification model (M). The classification model (M) then performs classification prediction on the vectors in the development set ($V1$) and obtains the classification categories and prediction confidence (c) of each vector. The lexicon l is updated according to the confidence and attention given by the model M. Find words from the lexicon that match the test cases and count the number of the words (k) (lines 5–8).

When the number (k) of words that match the test cases than or equal to ($k2$), the word vector matches the word set according to the pattern matching rule. The category of the current word vector is predicted based on the number of matched words (k), and The word vector v of the predicted category is added to the transfer set V3 and is removed from the training set ($V3$). When the prediction confidence (c) is greater than or equal to q, the category of the test case is marked as the category predicted by the classification model (M) and added to the test case confidence dictionary ($D2_{t1}$). The word vector of the predicted category (v) is added to the transfer set V3 and is removed from the training set ($V3$). The word vector remains in the development set if the current word vector category is still not predicted according to the pattern matching rules (lines 9–21).

Algorithm 2 Lexicon Collection

Input: Development set of test cases' vectors $V1$, training set of test cases' vectors $V2$, confidence's threshold q, $k1$, $k2$
Output: classification model M

```
1: initialize Lexicon l
2: initialize dictionary of test case confidence D2_i for type i
3: initialize transfer set of test cases' vectors V3
4: train classification model M by V2 using BiLSTM with attention
5: for test case's vector v in V1 do
6:    t1, c ⇐ predict(M, v)
7:    update lexicon l
8:    t2, k ⇐ match(l, v) using lexicon l
9:    if k >= k2 then
10:      t2 ⇒ label v
11:      transfer(v, V1, V3)
12:   else
13:      if c >= q and k1 <= k < k2 then
14:         t1 ⇒ label v
15:         add v to D2_t1
16:         transfer(v, V1, V3)
17:      else
18:         continue
19:      end if
20:   end if
21: end for
22: The first n test cases with the highest confidence level are selected
    from D2_i as vector V4
23: for test case's vector v in V4 do
24:   add top m attention's word to Lexicon l
25: end for
26: if V3 ! = null then
27:   V2 ⇐ V2+V3
28:   back to Line 4
29: else
30:   save M
31:   return M
32: end if
```

After that, the top n test case word vectors ($V4$) with high confidence are selected in the test case-confidence dictionary ($D2_i$). The top m words with great attention in each word vector are selected with the word package. These words are added to lexicon l (lines 22–25).

If the transfer set $V3$ is not empty, the test case word vectors in the transfer set are transferred from the development set to the training set and removed from the development set. The classification model M is retrained based on the new getting training set, and then re-execute the steps above until the transfer set $V3$ is empty (lines 26–28).

If the transfer set is empty, the current initial classification model is saved as the final classification model, ending the process (lines 29–32).

The lexicon-based pattern matching rules mentioned in Algorithm 2 are as follows, where k is the number of words matched by the current test case in the lexicon and $k1 < k2$.

(1) If there are at least t2 matches in the lexicon, then the data are labeled according to the predictions of the lexicon.

(2) If the classifier predicts the unlabeled test cases with high confidence and at least k1 matching words in the lexicon, the test cases are labeled according to the classifier's prediction.

Algorithm 2 represents assigning weights to words, generating a lexicon, and acquiring pseudo-labels in the Lexicon Collection stage. After the test case from the development set acquires the pseudo-label, it can be used as new training data to help iterate the model for training and thus improve the model's classification accuracy.

4. Experiment setup

This section aims to conduct experiments to evaluate the effectiveness of our approach. First, we propose three Research Questions (RQs). Second, we introduce the experiment object. Third, we show the adopted evaluation metrics. Four, we show the experiment Settings.

4.1. Research questions

The empirical study is conducted to answer the following RQs.

RQ1: Can the proposed approach be effective for the test case classification?

RQ2: Is the test case augmentation effective for the proposed approach?

RQ3: Can the proposed approach further optimize the classification model using unlabeled data?

RQ4: Can the proposed approach help testers perform the test case classification?

RQ5: Can the lexicon generated by selecting word vectors based on the confidence and attention values accurately represent the data of each class?

RQ1 aims to evaluate the effectiveness of the proposed approach in the test case classification. **RQ2** aims to investigate whether the test case augmentation is effective for the proposed approach. **RQ3** aims to investigate whether the model can be further trained using unlabeled data using pseudo-labels. **RQ4** aims to evaluate the usefulness of the proposed approach. **RQ5** aims to evaluate the effectiveness of selecting word vectors based on confidence and attention values.

4.2. Experiment object

To explore the above questions, we evaluate the proposed approach on three datasets of test cases obtained from real crowdsourced testing competitions. These three datasets were obtained from a crowdsourced testing competition that we ran. The competition included three test projects named Aerospace Recognition, Autonomous and controllable management system, and Fun GIF.

Among them, Fun Gif is an open-source GIF browsing and sharing app for Android. Autonomous and controllable management system is an autonomous and controllable test of the MU test WEB side of the operation and maintenance management system. Aerospace recognition is a web application of a software evaluation company. In this experiment, the multiple classifications of tests are divided into a total of 6 categories including (1) abnormal exit, (2) incomplete functionality, (3) user experience, (4) page layout defects, (5) performance, and (6) security. The test cases to be classified are composed of the Chinese strings of "test_requirement", "test_process" and "product_version_module" in the crowdsourced test report.

The three test case datasets come from two fields, i.e., entertainment and tools. The number of test cases in the dataset is shown in Fig. 2.

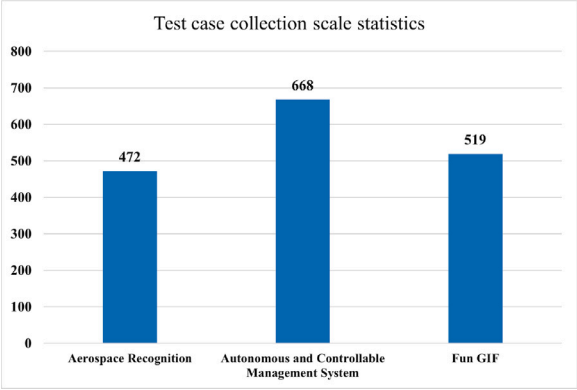


Fig. 2. Test case collection scale statistics of crowdsourced testing.

4.3. Evaluation metrics

To evaluate the performance of the proposed approach, we use the following evaluation metrics. These metrics are based on a confusion matrix, which includes True Positive (i.e., TP is the number of positive samples accurately predicted as positive samples), False Positive (i.e., FP is the number of negative samples falsely predicted as positive samples), True Negative (i.e., TN is the number of positive samples falsely predicted as negative samples), and False Negative (i.e., FN is the number of negative samples accurately predicted as negative samples).

Given a test case class, the definitions of the selected evaluation metrics are shown below.

- Accuracy is the ratio of correctly predicted test cases over all the test cases. It is calculated as follows.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5}$$

- Precision is the proportion of the test cases correctly predicted as the category over all test cases predicted as the category.

$$Precision = \frac{TP}{TP + FP} \tag{6}$$

- Recall is the ratio of the test cases correctly predicted as the category over all the test cases in the category.

$$Recall = \frac{TP}{TP + FN} \tag{7}$$

- F1 is the weighted harmonic mean of precision and recall, which is calculated as follows.

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} \tag{8}$$

In this work, the classified test cases come into multi-class cases, and the metrics should involve all the classes. Thus, following existing work [21], we use a multi-class measure of precision and recall to be inserted into the harmonic mean, i.e., Macro-F1-measure, Macro-Precision, and Macro-Recall.

Specifically, we calculate precision and recall by taking the average precision for each predicted class and the average recall for each actual class. When we switch from one class to another, we compute the quantities again, and the labels for the Confusion Matrix are changed accordingly. Each class has the same average weight, so there is no distinction between highly and poorly populated classes.

Given a class k, precision and recall for each class are computed using the same formulas of the binary setting and the labeling (as described above), which are defined by Eq. (9) and (10).

$$Precision_k = \frac{TP_k}{TP_k + FP_k} \tag{9}$$

$$Recall_k = \frac{TP_k}{TP_k + FN_k} \tag{10}$$

Then, considering there exist K classes in our experiment, macro average precision and recall can be computed as the arithmetic means of the metrics for single classes, and macro F1-Score is the harmonic mean of average macro precision and average macro recall, defined as follows.

$$Macro\text{-}Precision = \frac{\sum_{k=1}^{K} Precision_k}{K} \tag{11}$$

$$Macro\text{-}Recall = \frac{\sum_{k=1}^{K} Recall_k}{K} \tag{12}$$

$$Macro\text{-}F1 = 2 * \left(\frac{Macro\text{-}Precision * Macro\text{-}Recall}{Macro\text{-}Precision + Macro\text{-}Recall} \right) \tag{13}$$

Hence, the macro approach considers all the classes as fundamental elements of the calculation and is quite suitable for our experiment.

4.4. Tester recruitment

We invited 12 participants to our experiment. All of them are graduate students majoring in software engineering or computer science. Besides, as a part of their coursework, they are familiar with crowdsourced testing and test cases. We are confident that the tester recruitment can provide reliable results in our experiment.

4.5. Experiment settings

We implemented BiLSTM using the Long Short-Term Memory (LSTM) model in PyTorch, which contains a one-layer BiLSTM model with parameters embedding_size=64, hidden_size=16. Adam is used as the optimizer with a learning rate of 0.001, and Binary Cross Entropy is adopted as the loss function. The number of test cases in the dataset is shown in Fig. 2.

We conducted experiments on macOS 10.13, running on an Intel Core i5 CPU @2.7 GHz with two cores and with a RAM size of 8 GB.

5. Experiment evaluation

This section shows the experiment design and results. As for each RQ, we give the corresponding answer.

5.1. RQ1: can the proposed approach be effective for the test case classification?

Design. To verify whether the class labels generated by the classifier used in this system are reasonable and accurate enough in the small sample scenario, we set up comparative experiments using the test case classification algorithm based on small sample learning, support vector machine algorithm, k nearest neighbor algorithm and naive bayes algorithm build classifiers on three datasets. These classifiers use 60% of the data set as the training set for training, 20% of the data as a development set for extracting pseudo-labels to improve model efficiency, and 20% of the data set as the test set to simulate the fact that the number of classified samples in the test case set accounts for a small number of total samples in the actual scene, Count the accuracy of each classifier to see if our algorithm is superior.

Results. Table 5 shows that on the same dataset, the proposed approach achieves the best results in accuracy, macro-precision, macro-recall, and macro-f1 compared to other classical classification methods. The maximum advantages achieved in each metric compared to other methods are 60%, 45%, 45%, and 50%, respectively (77% vs 17% for accuracy, 62% vs 17% for macro-precision, 64% vs 19% for macro-recall, and 66% vs 16% for F1- score). This shows that the proposed approach can effectively improve the efficiency of test case classification compared to other classification algorithms.

Answer to RQ1: Compared with the existing classification algorithms, the proposed approach significantly improves accuracy, macro-precision, macro-recall, and macro-F1. Therefore, the proposed approach is practical for test case classification.

5.2. RQ2: Is the test case augmentation effective for the proposed approach?

Design. To answer this RQ, we conduct the experiments with and without the test case augmentation. Specifically, during the experiments, the standard procedure increased the number of test cases from an unequal state of 50 to 100 per classification to about 500. The dataset was text processed and vectorized. The comparison procedure did not augment the test cases but performed text processing and subsequent operations on the test cases directly. Subsequently, we use accuracy, macro-precision, macro-recall, and macro-F1 to evaluate this RQ.

Results. Table 6 shows that on the same data set, the proposed approach achieves better results in terms of accuracy, macro-precision, macro-recall, and macro-F1 compared to the approach without augmentation of the training set. The maximum advantages achieved in each metric are 34%, 39%, 33%, and 50%, respectively (86% vs 52% for accuracy, 62% vs 23% for macro-precision, 64% vs 31% for macro-recall, and 66% vs 16% for F1- score). This shows that the classification effectiveness of the proposed approach decreases significantly when the test case augmentation module is removed. Therefore, test case augmentation is indispensable for the proposed approach.

Answer to RQ2: It can be found from the experiments that the classification effect of the algorithm decreases significantly after the test case augmentation module is removed. Test case augmentation can significantly improve the classification effect of the proposed approach.

5.3. RQ3: Can the proposed approach further optimize the classification model using unlabeled data?

Design. The proposed approach iterates the model by labeling the unlabeled data with pseudo-labels. We aim to conduct experiments in both cases with and without pseudo-labels to answer this research question. Specifically, during the execution of the proposed approach, a classification model is first trained using a training set containing labeled data. After that, the proposed approach classifies the unlabeled data with pseudo-labels, and then iterative training is performed to improve the model's classification. The compared technique does not generate pseudo-labels and does not perform iterative training. Finally, we evaluate this RQ in terms of accuracy, macro-accuracy, macro-recall, and macro-F1.

Results. Table 7 shows the comparison results for the proposed approach with or without pseudo-label phrase based on three datasets. We can find that applying pseudo-labels to unlabeled data can achieve better accuracy, macro-precision, macro-recall, and macro-F1, respectively. The maximum improvement for each metric are 30%, 37%, 38%, and 38%, respectively (86% vs. 56% for accuracy, 85% vs. 48% for macro-precision, 87% vs. 49% for macro-recall, and 88% vs. 50% for F1-score). Although the improvement degree is slight, it shows that with the help of the pseudo-label, the model can be further trained using unlabeled data to improve the model classification performance. As such, the improvement degree is still valuable in practice.

Answer to RQ3: With the help of the pseudo-label, the proposed approach further improves the classification performance by training unlabeled data. This shows that after training the initial model using the proposed approach, we can directly use the unlabeled data to optimize the model further, which can significantly reduce the workload of manually labeling the data.

Table 5
Comparison of indicators by classification approaches.

Dataset	Classification algorithm	Accuracy	Macro-precision	Macro-recall	Macro-F1
Fun GIF	Our approach	**0.86**	**0.85**	**0.87**	**0.88**
	KNN	0.43	0.21	0.24	0.22
	SVM	0.49	0.13	0.2	0.14
	Naive Bayes	0.13	0.2	0.15	0.1
Autonomous and controllable management system	Our approach	**0.77**	**0.62**	**0.64**	**0.66**
	KNN	0.45	0.17	0.18	0.16
	SVM	0.54	0.14	0.18	0.13
	Naive Bayes	0.17	0.17	0.19	0.11
Aerospace Recognition	Our approach	**0.71**	**0.43**	**0.43**	**0.54**
	KNN	0.45	0.23	0.27	0.22
	SVM	0.56	0.3	0.33	0.26
	Naive Bayes	0.31	0.26	0.26	0.21

Table 6
Impact analysis of data augmentation in our approach.

Dataset	Method	Accuracy	Macro-precision	Macro-recall	Macro-F1
Fun GIF	Our approach	**0.86**	**0.85**	**0.87**	**0.88**
	Without augmentation	0.52	0.37	0.37	0.43
Autonomous and controllable management system	Our approach	**0.77**	**0.62**	**0.64**	**0.66**
	Without augmentation	0.57	0.23	0.31	0.16
Aerospace Recognition	Our approach	**0.71**	**0.43**	**0.43**	**0.54**
	Without augmentation	0.6	0.22	0.31	0.18

Table 7
Impact analysis of pseudo-label in our approach.

Dataset	Method	Accuracy	Macro-precision	Macro-recall	Macro-F1
Fun GIF	Our approach	**0.86**	**0.85**	**0.87**	**0.88**
	Without pseudo label	0.56	**0.48**	**0.49**	0.5
Autonomous and controllable management system	Our approach	**0.77**	**0.62**	**0.64**	**0.66**
	Without pseudo label	0.75	0.59	0.62	0.62
Aerospace Recognition	Our approach	**0.71**	**0.43**	**0.43**	**0.54**
	Without pseudo label	0.6	0.37	0.42	0.4

5.4. RQ4: Can the proposed approach help testers perform the test case classification?

Design. To answer this RQ, we recruit 12 testers to conduct a comparative experiment (i.e., the control and experiment groups). The control group performs the test case classification without the assistance of the proposed approach. The experiment group performs the test case classification with the assistance of the proposed approach. Subsequently, we divide 12 testers into two groups, where six testers conduct the control group and the remaining six testers conduct the experimental group. After that, we compare the cost that testers classifying test cases in the control and experiment groups.

Further, we investigate whether the proposed approach can help improve the classification quality of test cases. Specifically, after these 12 testers completed the test case classification, we invited three experienced testing experts to score the accuracy of the test case classification 12 testers. Specifically, each expert scores each test case classification results from 12 testers from 1 to 10 points. In the end, we calculate the average point for each dataset in the experiment and control groups.

Results. Table 8 shows the average cost of the control and experiment groups when testers are asked to classify test cases in three datasets. As shown in Table 8, it can be seen that with the assistance of the proposed approach, the average cost of test case classification in the experiment group is shorter than that of the control group. Specifically, the experiment group is consistently better than the control group on each dataset in terms of cost. Overall, the experiment group is six minutes faster than the control group. It indicates that with the assistance of the proposed approach, the experiment group improves the control group by about 26%.

Table 8
The average cost of test case classification on 12 testers.

Dataset	Experiment group	Control group
Aerospace Recognition	19.3 min	25.1 min
Autonomous and controllable management system	25.2 min	34.7 min
Fun GIF crowdsourcing test	20.2 min	27.9 min
Average	23.2 min	29.3 min

Table 9 shows the score results of three testing experts. As shown in Table 9, the score of the experiment group is consistently better than the control group. For example, the experiment group is 1.8 points better than the control group in the aerospace recognition dataset. The above results indicate that the proposed approach can help testers perform the test case classification.

Answer to RQ4: By conducting the comparative experiments on 12 testers in terms of the cost and accuracy of test case classification, the experimental results show that the proposed approach can help testers perform the test case classification.

5.5. RQ5:can the lexicon generated by selecting word vectors based on the confidence and attention values accurately represent the data of each class?

Design. To answer this RQ, we applied a distance-based prioritization method for the comparison. The distance-based approach is

Table 9
The Score results of three testing experts. It is noted that "E" is the experiment group and "C" is the control group. "1" is the aerospace recognition dataset, "2" is the autonomous and controllable management system dataset, "3" is the fun gif crowdsouring test dataset.

Expert	1-E	1-C	2-E	2-C	3-E	3-C
Expert1	8.2	7.4	8.3	7.7	8.1	7.2
Expert2	8.0	7.2	8.5	8.1	7.9	7.1
Expert3	8.5	8.3	9.1	8.6	8.3	7.7

Table 10
Impact Analysis of word vectors prioritization in our approach.

Dataset	Method	Accuracy	Macro-precision	Macro-recall	Macro-F1
Fun GIF	Our approach	**0.86**	**0.85**	**0.87**	**0.88**
	Distance-based approach	0.85	0.85	0.86	0.87
Autonomous and controllable management system	Our approach	**0.77**	**0.62**	**0.64**	**0.66**
	Distance-based approach	0.76	0.61	0.62	0.65
Aerospace Recognition	Our approach	**0.71**	**0.43**	0.43	**0.54**
	Distance-based approach	0.64	0.4	**0.44**	0.4

one of the most commonly used methods in test case prioritization. Specifically, the proposed method selects the top m word vectors based on confidence and attention. In the distance-based comparison method, we will select the top m word vectors based on the cosine distance between word vectors. For the first selection, we choose a word vector with the highest confidence and then find the m closest word vectors to it and add them to the lexicon. The other parts are the same as the proposed method. Then we evaluate the two methods on three datasets separately using accuracy, macro-accuracy, macro-recall, and macro-F1.

Results. Table 10 shows the experimental results of comparing our proposed method with the distance-based method. As shown in Table 10, the performance of the two methods is closer, but in general, our proposed method has a slight advantage in the performance of the three datasets and a more pronounced advantage in the Aerospace Recognition data.

Answer to RQ5: From the results of the comparison experiments, it can be found that the top m word vectors selected according to the confidence and attention values can better represent the corresponding categories.

6. Threats to validity

Although test case classification techniques based on small sample learning have been used in real test case management platforms with some success, it supports a more reasonable classification of incremental test cases belonging to the set of historical test cases with a small number of already classified test cases. However, there are still many areas for improvement and continued research in implementing the proposed approach and experiments.

First, due to practical factors, the data size of the test case set used for experiments at this stage is small and more suitable for processing by machine learning. The small sample-based test case classification technique belongs to the deep learning and neural network category, which is more suitable for larger data sets. We cannot observe the performance of this technique on large test case sets for the time being. In the future, we need to expand the test case set size to provide credible large-scale datasets for similar research work in test case classification.

Second, although this technique outperforms the rest of the models, such as SVM, DBM, etc., in the case of a small number of classified data in the dataset, the classifier accuracy still needs to be improved. This paper introduces a category label review mechanism at the proposed approach level to provide positive input feedback for the classifier's training. However, there is still a need to improve the algorithm for collecting category lexicons at the algorithmic level. Based on experimental observations, some exact keywords are often collected from different categories in different lexicons. These are given higher attention weights when the test case classification model predicts different categories and thus belongs to the public cross-words between categories. In future work, we hope to avoid the misleading role of these words in the lexicon matching mechanism by building a public thesaurus to collect keywords that frequently occur in each category and excluding words from the public lexicon in the lexicon matching process.

Finally, the proposed approach needs to train separate classifiers for each test case set of different topics. We still have not found a way to classify all test cases using a single classifier accurately. A follow-up idea could be to cluster or classify the test case sets, and a standard classifier could be trained for similar case sets.

7. Related work

7.1. Data augmentation

Data augmentation is the process of generating the data required by the user from the original data using artificially set variation methods. The problem of insufficient training data in machine learning is significantly alleviated with appropriate data augmentation methods. The benefit of using data augmentation is that there is more of it and that the data is evenly distributed. Data augmentation can be divided into image data augmentation methods, text data augmentation methods, and other data augmentation methods. Text data, made up of a combination of characters, cannot be amplified using the widely used image amplification methods. Because image augmentation is relatively easy compared to text augmentation, data augmentation methods have been used extensively in the CV domain first, where researchers have been able to implement image blurring based on various matrix operators, modifying the color of images based on modifications of color channel values, and geometric transformation of images through lossless geometric transformation methods derived from mathematical methods, which are common in the traditional CV domain. Batch augmentation of images can be achieved quickly with packaged Python libraries such as imaging. Symbols in computers represent natural language, and it is difficult to generate or modify these symbols by simple mathematical equations; any change in the symbols can drastically alter the meaning of the text. Therefore, text augmentation is more complicated than the other two types of augmentation [22].

The back-translation method is derived from linguistics, where back-translation methods are used to learn other languages and verify the accuracy of language translations. In the field of data augmentation, back translation augmentation is the process of translating keyword text data into another language using a neural network translation

system and then translating the text back into the original language using that neural network translation system. The back-translation method can generate a large amount of similar text data because of the inaccurate translation of the neural network translation system. The back-translation augmentation method can retain the basic semantic information of the text. Ma et al. [23] have validated back-translation augmentation, and the experimental results show that back-translation augmentation can improve the performance of the text classification model.

Noise augmentation refers to generating noise by replacing words, deleting words or characters in the text data, and generating new data similar to the original text data. Wei et al. [24] proposed an EDA augmentation method, which includes inserting random words into the text, using synonyms for substitution, randomly swapping the order of words in the text, and randomly deleting characters or words in the text. The method achieves good results with a small amount of text training data.

Because of insufficient experimental data, Zhang et al. [25] wanted to generate new data by replacing words with synonyms using an existing synonym lexicon. The experimental results showed that this method could improve the performance of their text classification model.

Kobayashi et al. [26] proposed a new data augmentation method for labeled sentences called contextual augmentation. Words are randomly replaced with other words predicted by a bidirectional language model at word positions. The language model is adapted with a label conditional architecture, which allows the model to augment sentences without destroying the labels. This can be used to augment data for text classification.

Hou et al. [27] proposed a sequence-based data augmentation framework that uses the same semantic alternatives to a word in the training data to augment a word, regardless of its relationship to other words. For the first time, diversity levels are incorporated into the corpus representation, allowing the model to produce a diverse corpus. These diversity-enhanced corpora help to improve the language comprehension module, but whether the method is somewhat restrictive and has broad applicability needs to be further explored [28].

SimBERT [18] is an integrated retrieval and generation BERT model based on UniLM. SimBERT can be used to generate similar text from existing classified text data or to build a corpus of test cases to retrieve similar text from the corpus to obtain pseudo-labeled test case text data with good semantic relevance, which can effectively improve the effectiveness of the classifier.

The two text data augmentation routes, back translation, and noise addition can help deep learning models improve performance and generalization by producing the required data. However, there is currently no text augmentation method for keyword text data, and generic text augmentation methods may not be able to generate the required augmented keyword text data when a keyword text is augmented.

7.2. Text classification

Text classification [29] is the process of automatically classifying a text set using a machine according to a specific classification system or rules. And is it a fundamental task in natural language processing tasks, which aims to organize and categorize text resources, and is also a crucial part of solving the text information overload problem? The text classification process generally includes text preprocessing, feature extraction, model construction, and classifier training processes [30].

According to whether labeled data is required or not, text classification methods can be classified as unsupervised text classification [31], semi-supervised text classification [32], and supervised text classification [33]. Unsupervised text classification does not require training data with category labeling. Supervised machine learning methods require large amounts of labeled training data, but the classification results depend highly on manual labeling, which is costly. Semi-supervised text classification algorithms require only a small amount of labeled data and build classification models by learning the potential features of a small amount of labeled data and a large amount of unlabeled data and making predictions for new data. Traditional classification models include plain Bayesian, nearest neighbor, decision tree, support vector machine, etc.

Unlike traditional classification models, non-traditional text classification methods can access higher-level and more abstract semantic features in text, compensating for the shortcomings of human-designed features in traditional models.

Non-traditional text classification methods include deep learning [34], ensemble Learning [35], transfer learning [36], reinforcement learning [37], etc. Integration learning improves the accuracy of text classification by fusing the classification results of multiple text classifiers, which preserves the variability among the base classifiers and improves the robustness of the model; migration learning trains language models on a large-scale general corpus and fine-tunes them on specific datasets, which can reduce the training difficulty of the models and save the computational resources for training; reinforcement learning models the text classification problem by the core ideas of trial-and-error and delayed payoff. Reinforcement learning models the text classification problem as a discrete and ordered decision process with high interpretability through the core ideas of trial-and-error and delayed payoff.

BiLSTM [38] is a popular model in deep learning and a variant of long and short-term memory networks. LSTM can perform better on longer sequences than a normal RNN. However, in some cases, the prediction may need to be determined by a combination of several inputs before and several inputs after. The Bi-directional LSTM adds a backward layer to the forward layer of the LSTM, preserving the critical information of several inputs so that the prediction result of the network will be more accurate.

Text classification can be applied to several fields. The accurate classification labels generated by text classification methods provide powerful support for resource retrieval and personalized recommendation in search and recommendation [39]. In sentiment analysis [40], text classification can analyze the sentiment tendency of user comments, help analyze hot topics, understand user habits, and monitor crisis public opinion. In the field of dialogue system [41], the classification of questions raised by users enables intelligent customer service to answer users' questions instead of human customer service, effectively reducing operating costs and improving user experience.

7.3. Few-shot learning

Li et al. formally introduced the concept of Few-Shot learning in 2003 [42], who argued that old categories learned can help predict new categories when the new category has only one or a few labeled samples. Few-Shot learning is a new machine learning paradigm proposed to learn from a limited number of examples with supervised information, learning classifiers when only a few labeled samples are given for each category. In addition to wanting to make artificial intelligence learn like humans, Few-Shot learning can learn models for these rare cases when it is difficult or impossible to obtain enough samples, such as in a drug discovery task where Few-Shot learning can predict whether a new ingredient is toxic. Few-Shot learning can also help alleviate the burden of collecting samples with supervised information and reduce the cost of data collection and computation, as labeling unlabeled data can be time- and labor-intensive.

Existing work on few-shot learning can be divided into data, models, and algorithms [43]. From the data perspective, Few-Shot learning approaches mainly use prior knowledge to augment the training data and enhance the sample set to enrich the supervised information for training. Data augmentation can be performed by artificially defined rules, such as panning [44], flipping [45], cropping [46], and scaling [47] in the image domain. However, these augmentation rules depend heavily on domain knowledge and require expensive labor.

Firstly, they are challenging to produce effects in other datasets and, secondly, to enumerate all possible augmentation rules, so manual data augmentation cannot fully solve the Few-Shot problem [48]. There are also three more advanced types of data augmentation methods. (1) Augmentation of Few-Shot datasets using unlabeled data [49]. When many weakly supervised or unlabeled samples exist for the target task or category, some of them can be selected to give pseudo-label to enhance the training set. (2) Augmenting the training set by augmenting sample features [50,51]. Feature augmentation of samples using auxiliary datasets or additional attributes improves the low feature diversity due to the sample size in Few-Shot learning. (3) Enhancing the training set by aggregating and adapting samples from other datasets. Using generative adversarial networks [52] to determine whether the newly generated samples belong to the sample classes in the training set, the newly generated samples are synthesized into the original dataset by corrupting and generating samples from auxiliary datasets.

From the modeling perspective, Few-Shot learning mainly focuses on using old knowledge to learn new knowledge by transferring the already learned source domain knowledge to a new target domain with particular relevance to help train the classification model. Their methods can be divided into metric learning-based, meta-learning-based, and graph neural network-based methods.

Among the metric learning-based methods, Koch was the first to propose using twin neural networks for one-shot image recognition in 2015 [53], learning metrics from data and using the learned metrics to compare and match samples of unknown sample categories. Snell et al. proposed a prototype network in 2017 [54], where the average of sample vectors for samples belonging to the same category is obtained as the prototype for that category. The model is continuously trained by Gao et al. and Sun added an attention mechanism to the prototype network in their subsequent work [55,56], demonstrating that different samples and features are essential for the classification task.

From an algorithmic perspective, Few-Shot learning focuses on providing the best initialization parameter for the model [57] and subsequent strategies for fine-tuning the optimization of that parameter [58]. When the extensive data set and the target few-shot data set are similarly distributed, it is common to pre-train the model on the large-scale data and fine-tune the parameters of the connectivity layer of the neural network model on the Few-Shot data set to obtain the fine-tuned model finally. However, in real-world scenarios, the data distributions of the target and source datasets are often not similar, and using fine-tuning methods can lead to the overfitting of the model on the target dataset.

8. Conclusions and future work

This paper proposes a test case classification approach based on few-shot learning. The proposed approach filters test cases, extracts test case-related information, augments them, and performs text processing and vectorization. After that, the proposed approach introduces an attention mechanism on the BiLSTM model to construct a category lexicon of test cases and obtain the classification of test cases. Subsequently, the proposed approach constructs the category lexicon of test cases to help iteratively train the model and obtain higher accuracy. We construct a dataset consisting of three crowdsourced test projects with thousands of test cases and use this dataset for comparison experiments. The experimental results show that the proposed approach has a 5%–25% improvement in accuracy compared to existing algorithms. Besides, with a control group, the classification framework's category labels in the proposed approach can help testers improve their testing efficiency.

In the future, we need to expand the test case set size to provide credible large-scale datasets for similar research work in test case classification since the size of the dataset we use in this paper is small. Second, during our experiments, we found that certain keywords would be classified as different types in different thesauri. These keywords are given higher attention values in different categories when the classification model classifies the data in the corresponding classes. In future work, we hope to avoid the misleading role of these words in the thesaurus lexicon matching mechanism by building a public thesaurus to collect the frequently occurring keywords in each category and excluding words from the public thesaurus during the word-matching process.

Declaration of competing interest

No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.infsof.2023.107228.

Acknowledgments

The authors would like to thank the anonymous reviewers for their insightful comments. This work is supported partially by National Natural Science Foundation of China (62272220, 62141215), Innovative Research Foundation of Ship General Performance, China (25422207), and Science, Technology, and Innovation Commission of Shenzhen Municipality, China (CJGJZD20200617103001003).

References

[1] W.-T. Tsai, W. Wu, M.N. Huhns, Cloud-based software crowdsourcing, IEEE Internet Comput. 18 (3) (2014) 78–83.
[2] T. Hoßfeld, C. Keimel, M. Hirth, B. Gardlo, J. Habigt, K. Diepold, P. Tran-Gia, Best practices for QoE crowdtesting: QoE assessment with crowdsourcing, IEEE Trans. Multimed. 16 (2) (2014) 541–558.
[3] B. Gardlo, S. Egger, M. Seufert, R. Schatz, Crowdsourcing 2.0: Enhancing execution speed and reliability of web-based QoE testing, in: Proceedings of 2014 IEEE International Conference on Communications, 2014, pp. 1070–1075.
[4] E. Dolstra, R. Vliegendhart, J. Pouwelse, Crowdsourcing GUI tests, in: Proceedings of 2013 IEEE 6th International Conference on Software Testing, Verification and Validation, 2013, pp. 332–341.
[5] S. Komarov, K. Reinecke, K.Z. Gajos, Crowdsourcing performance evaluations of user interfaces, in: Proceedings of 2013 the ACM Conference on Human Factors in Computing Systems, 2013, pp. 207–216.
[6] A.I. Khan, Z. Al-khanjari, M. Sarrab, Crowd sourced testing through end users for Mobile Learning application in the context of Bring Your Own Device, in: Proceedings of 2016 the 7th IEEE Annual Information Technology, Electronics and Mobile Communication Conference, 2016, pp. 1–6.
[7] T. Zhang, J. Gao, J. Cheng, Crowdsourced testing services for mobile apps, in: Proceedings of 2017 the 13th IEEE Symposium on Service-Oriented System Engineering, 2017, pp. 75–80.
[8] F. Belli, C.J. Budnik, W.E. Wong, Basic operations for generating behavioral mutants, in: Second Workshop on Mutation Analysis, Mutation 2006-ISSRE Workshops 2006, 2006, pp. 9–18.
[9] Q. Zhang, C. Fang, W. Sun, S. Yu, Y. Xu, Y. Liu, Test case prioritization using partial attention, J. Syst. Softw. 192 (2022) 111419.
[10] C. Fang, Z. Chen, B. Xu, Comparing logic coverage criteria on test case prioritization, Sci. China Inf. Sci. 55 (12) (2012) 2826–2840.
[11] D. Liu, Y. Feng, X. Zhang, J. Jones, Z. Chen, Clustering crowdsourced test reports of mobile applications using image understanding, IEEE Trans. Softw. Eng. (2020).
[12] C. Fang, Z. Chen, K. Wu, Z. Zhao, Similarity-based test case prioritization using ordered sequences of program entities, Softw. Qual. J. 22 (2) (2014) 335–361.
[13] W.E. Wong, J.R. Horgan, S. London, A.P. Mathur, Effect of test set minimization on fault detection effectiveness, in: Proceedings of the 17th International Conference on Software Engineering, 1995, pp. 41–50.
[14] W.E. Wong, J.R. Horgan, S. London, A.P. Mathur, Effect of test set size and block coverage on the fault detection effectiveness, in: Proceedings of 1994 IEEE International Symposium on Software Reliability Engineering, IEEE, 1994, pp. 230–238.
[15] P. Wang, M. Varvello, A. Kuzmanovic, Kaleidoscope: A crowdsourcing testing tool for web quality of experience, in: Proceedings of 2019 the 39th IEEE International Conference on Distributed Computing Systems, IEEE, 2019, pp. 1971–1982.
[16] A. Swearngin, Y. Li, Modeling mobile interface tappability using crowdsourcing and deep learning, in: Proceedings of 2019 the ACM Conference on Human Factors in Computing Systems, 2019, pp. 1–11.

[17] M. Almeida, M. Bilal, A. Finamore, I. Leontiadis, Y. Grunenberger, M. Varvello, J. Blackburn, CHIMP: Crowdsourcing human inputs for mobile phones, in: Proceedings of 2018 the 27th International Conference on World Wide Web, 2018, pp. 45–54.
[18] J. Su, SimBERT: Integrating Retrieval and Generation into BERT, Tech. rep., 2020, URL https://github.com/ZhuiyiTechnology/simbert.
[19] J.-H. Lee, S.-K. Ko, Y.-S. Han, Salnet: semi-supervised few-shot text classification with attention-based lexicon construction, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 35, No. 14, 2021, pp. 13189–13197.
[20] W. Che, Y. Feng, L. Qin, T. Liu, N-LTP: An open-source neural language technology platform for Chinese, in: Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, Association for Computational Linguistics, 2021, pp. 42–49, http://dx.doi.org/10.18653/v1/2021.emnlp-demo.6, Online and Punta Cana, Dominican Republic. URL https://aclanthology.org/2021.emnlp-demo.6.
[21] M. Grandini, E. Bagli, G. Visani, Metrics for multi-class classification: an overview, 2020, arXiv preprint arXiv:2008.05756.
[22] X. Lu, B. Zheng, A. Velivelli, C. Zhai, Enhancing text categorization with semantic-enriched representation and training data augmentation, J. Am. Med. Inform. Assoc. 13 (5) (2006) 526–535.
[23] J. Ma, L. Li, Data augmentation for chinese text classification using back-translation, J. Phys. Conf. Ser. 1651 (1) (2020) 012039.
[24] J. Wei, K. Zou, Eda: Easy data augmentation techniques for boosting performance on text classification tasks, 2019, arXiv preprint arXiv:1901.11196.
[25] X. Zhang, J. Zhao, Y. LeCun, Character-level convolutional networks for text classification, Adv. Neural Inf. Process. Syst. 28 (2015).
[26] S. Kobayashi, Contextual augmentation: Data augmentation by words with paradigmatic relations, 2018, arXiv preprint arXiv:1805.06201.
[27] Y. Hou, Y. Liu, W. Che, T. Liu, Sequence-to-sequence data augmentation for dialogue language understanding, 2018, arXiv preprint arXiv:1807.01554.
[28] X. WEI, Y. LI, Z. WANG, H. LI, H. WANG, Methods of training data augmentation for medical image artificial intelligence aided diagnosis, J. Comput. Appl. 39 (9) (2019) 2558.
[29] C.C. Aggarwal, C. Zhai, A survey of text classification algorithms, in: Mining Text Data, Springer, 2012, pp. 163–222.
[30] K. Kowsari, K. Jafari Meimandi, M. Heidarysafa, S. Mendu, L. Barnes, D. Brown, Text classification algorithms: A survey, Information 10 (4) (2019) 150.
[31] N. Shafiabady, L.H. Lee, R. Rajkumar, V. Kallimani, N.A. Akram, D. Isa, Using unsupervised clustering approach to train the Support Vector Machine for text classification, Neurocomputing 211 (2016) 4–10.
[32] T. Miyato, A.M. Dai, I. Goodfellow, Adversarial training methods for semi-supervised text classification, 2016, arXiv preprint arXiv:1605.07725.
[33] A.I. Kadhim, Survey on supervised machine learning techniques for automatic text classification, Artif. Intell. Rev. 52 (1) (2019) 273–292.
[34] S. Minaee, N. Kalchbrenner, E. Cambria, N. Nikzad, M. Chenaghlu, J. Gao, Deep learning–based text classification: a comprehensive review, ACM Comput. Surv. 54 (3) (2021) 1–40.
[35] G. Wang, J. Sun, J. Ma, K. Xu, J. Gu, Sentiment classification: The contribution of ensemble learning, Decis. Support Syst. 57 (2014) 77–93.
[36] C.B. Do, A.Y. Ng, Transfer learning for text classification, Adv. Neural Inf. Process. Syst. 18 (2005).
[37] T. Zhang, M. Huang, L. Zhao, Learning structured representation for text classification via reinforcement learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 32, No. 1, 2018.
[38] G. Liu, J. Guo, Bidirectional LSTM with attention mechanism and convolutional layer for text classification, Neurocomputing 337 (2019) 325–338.
[39] K. Lei, Q. Fu, M. Yang, Y. Liang, Tag recommendation by text classification with attention-based capsule network, Neurocomputing 391 (2020) 65–73.
[40] P. Melville, W. Gryc, R.D. Lawrence, Sentiment analysis of blogs by combining lexical knowledge with text classification, in: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2009, pp. 1275–1284.
[41] R. Zhao, O.J. Romero, A. Rudnicky, SOGO: a social intelligent negotiation dialogue system, in: Proceedings of the 18th International Conference on Intelligent Virtual Agents, 2018, pp. 239–246.
[42] L. Fe-Fei, et al., A Bayesian approach to unsupervised one-shot learning of object categories, in: Proceedings Ninth IEEE International Conference on Computer Vision, IEEE, 2003, pp. 1134–1141.
[43] Y. Wang, Q. Yao, J.T. Kwok, L.M. Ni, Generalizing from a few examples: A survey on few-shot learning, ACM Comput. Surv. 53 (3) (2020) 1–34.
[44] S. Benaim, L. Wolf, One-shot unsupervised cross domain translation, Adv. Neural Inf. Process. Syst. 31 (2018).
[45] H. Qi, M. Brown, D.G. Lowe, Low-shot learning with imprinted weights, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 5822–5830.
[46] P. Shyam, S. Gupta, A. Dukkipati, Attentive recurrent comparators, in: Proceedings of International Conference on Machine Learning, PMLR, 2017, pp. 3173–3181.
[47] Y. Zhang, H. Tang, K. Jia, Fine-grained visual categorization using meta-learning optimization with sample selection of auxiliary data, in: Proceedings of the European Conference on Computer Vision, 2018, pp. 233–248.
[48] J. Kozerawski, M. Turk, Clear: Cumulative learning for one-shot one-class image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 3446–3455.
[49] T. Pfister, J. Charles, A. Zisserman, Domain-adaptive discriminative one-shot learning of gestures, in: Proceedings of European Conference on Computer Vision, Springer, 2014, pp. 814–829.
[50] E.G. Miller, N.E. Matsakis, P.A. Viola, Learning from one example through shared densities on transforms, in: Proceedings IEEE Conference on Computer Vision and Pattern Recognition, Vol. 1, IEEE, 2000, pp. 464–471.
[51] B. Liu, X. Wang, M. Dixit, R. Kwitt, N. Vasconcelos, Feature space transfer for data augmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 9090–9098.
[52] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial networks, Commun. ACM 63 (11) (2020) 139–144.
[53] G. Koch, R. Zemel, R. Salakhutdinov, et al., Siamese neural networks for one-shot image recognition, in: ICML Deep Learning Workshop, Vol. 2, Lille, 2015.
[54] J. Snell, K. Swersky, R. Zemel, Prototypical networks for few-shot learning, Adv. Neural Inf. Process. Syst. 30 (2017).
[55] T. Gao, X. Han, Z. Liu, M. Sun, Hybrid attention-based prototypical networks for noisy few-shot relation classification, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, No. 01, 2019, pp. 6407–6414.
[56] S. Sun, Q. Sun, K. Zhou, T. Lv, Hierarchical attention prototypical networks for few-shot text classification, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP, 2019, pp. 476–485.
[57] M. Yu, X. Guo, J. Yi, S. Chang, S. Potdar, Y. Cheng, G. Tesauro, H. Wang, B. Zhou, Diverse few-shot text classification with multiple metrics, 2018, arXiv preprint arXiv:1805.07513.
[58] J. Hoffman, E. Tzeng, J. Donahue, Y. Jia, K. Saenko, T. Darrell, One-shot adaptation of supervised deep convolutional models, 2013, arXiv preprint arXiv:1312.6204.