Test Case Classification via Few-Shot Learning

Yuan Zhao, Haojie Tang, Quanjun Zhang, Sining Liu, Jin Wang, and Jia Liu
State Key Laboratory for Novel Software Technology Nanjing University, Nanjing, China
allenzcrazy@gmail.com, richardhjtang@gmail.com, quanjun.zhang@smail.nju.edu.cn,
181250093@smail.nju.edu.cn, 13572494149@163.com, jialiu@nju.edu.cn

***Abstract*—Crowdsourcing testing can reduce testing costs and improve testing efficiency. However, crowdsourcing testing generates many test cases, from which testers need to select task-related test cases for execution. Furthermore, it is difficult to get the labeled test cases, and it is also costly to mark the test cases manually. Even in the set of labeled test cases, there is still an uneven distribution of labels.**

To address the difficulties mentioned above, this paper proposes a test case classification framework based on Few-shot Learning. The framework augments test cases and extracts relevant information for vectorization, collects the category lexicon of test cases using the BiLSTM model, and finally classifies the test cases. To verify the effectiveness of the classification framework, we selected thousands of test cases from three crowd testing projects to conduct in-usability evaluation experiments.

The experimental results show that our classification method has a higher accuracy rate than existing classification methods.

Keywords—Test Case, Few-Shot Learning, Classification Algorithm, Category Label

I. Introduction

With the change in the software development model and the development and standard application of new software technologies such as artificial intelligence, big data, cloud computing, and the Internet of Things, software testing nowadays faces various challenges. For example, in the testing of AI software, testing is often black-box: from the viewpoint of the test sample, the sample usually has a considerable limitation; from the viewpoint of the testing process, the whole process has an extended period, and it is challenging to realize automated testing; from the viewpoint of the test results, the results are also uninterpretable and low robustness. The testing for IoT is faced with strict test environment complexity, high reliability, real-time, and compatibility requirements. In the future, software testing will progress in the direction of agile, highly automated, cloud-based, service-oriented, and intelligent , and how to provide rapid feedback on software quality, improve testing efficiency, improve resource usage efficiency, and reduce costs is the eternal theme of software testing on the way forward.

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. It makes full use of the advantages of crowdsourcing and cloud platforms, has the characteristics of fast testing speed, and the industry highly favors low testing costs. It is very suitable for companies and individuals who lack professional testers and standardized testing. The crowd testing process generates many test cases, and due to the different levels of crowd testing workers, testers often need to screen the test case set and select suitable cases from them for inspection and later use.

Test cases for crowdsourcing 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 use cases and judge the accuracy of the use case classification, resulting in poor reuse of test cases for crowdsourcing testing. Although it is accurate to re-label 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.

In a crowdsourced test, the crowdsourced worker 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. Each test case contains several attributes such as test case name, test process, test case description, label, and test target category, where the label attribute indicates the keyword set of the test case and the test target category attribute refers to the category of the framework module tested by the test case such as functional integrity, user experience, page layout, abnormal exit, etc. The generalized labels in this framework usually include the two specific attributes of labels and test target categories.

We design an automatic test case classification framework to address the lack of inaccurate category labels for test cases in crowdsourcing testing. We can complete test case label extraction and category classification based on test case text analysis and small sample learning. The framework can be based on accurate data such as test cases of crowdsourcing tests and their associated test reports, and use methods such as text feature extraction and deep learning to deeply mine them and generate representative labels indicating the categories and scenarios of the test cases, helping testers manage test cases and reduce testing costs.

II. Related work and Background

A. Data Augmentation

Data augmentation is the process of generating the data required by the user from the original data by means of artificially set variation methods. With appropriate data augmentation methods, the problem of insufficient training data in machine learning is significantly alleviated. The benefit of using data augmentation is not only that there is more of it, but also 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, which is 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 colour of images based on modifications of colour 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 imgaug. Natural language is represented by symbols in computers, and it is difficult to generate or otherwise 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 difficult than the other two types of augmentation [1].

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 is able to retain the basic semantic information of the text.J Ma [2] has 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 the generation of noise by replacing words, deleting words or characters in the text data, and generating new data that is similar to the original text data. Jason W. Wei [3] 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.

Zhang Xiang [4] wanted to generate new data by replacing words with synonyms using an existing synonym lexicon because of insufficient experimental data. The experimental results showed that this method could improve the performance of their text classification model.

S Kobayashi [5] proposed a new method of data augmentation for labelled 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 Y [6] 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 wide applicability needs to be further explored [7].

SimBERT [8] is an integrated retrieval and generation BERT model based on UniLM, which was open sourced by Jianlin Su of Zhuiyi Technology in 2021. 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-labelled 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 generalisation 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.

B. Text Classification

Text classification [9] is the process of automatically classifying a text set using a machine according to a specific classification system or rules. And is 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 [10].

According to whether labeled data is required or not, text classification methods can be classified as unsupervised text classification [11], semi-supervised text classification [12], and supervised text classification [13]. 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.

Compared with 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 [14], ensemble Learning [15], transfer learning [16], reinforcement learning [17], 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.

Bi-LSTM [21] is a popular model in the field of deep learning and a variant of long and short-term memory network. LSTM is able to 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 bidirectional LSTM adds a backward layer to the forward layer of the LSTM, preserving the important information of several inputs after, 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 [18]. In sentiment analysis [19], 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 [20], the classification of questions raised by users enables intelligent customer service to provide answers to users' questions instead of human customer service, effectively reducing operating costs and improving user experience.

C. Few-Shot Learning

The concept of Few-Shot learning was formally introduced by Li in 2003 [22], who argued that old categories already 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 many 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 [23]. 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 [24], flipping [25], cropping [26], and scaling [27] 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 [28]. There are also three more advanced types of data augmentation methods. (1) augmentation of Few-Shot datasets using unlabeled data [29]. 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 [30] [31]. 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 [32] 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. Depending on their methods, they can be divided into metric learning based, meta-learning based, and graph neural network based methods.

Among the metric learning based methods, Koch were the first to propose using twin neural networks for one shot image recognition in 2015 [33], learning metrics from data and using the learned metrics to compare and match samples of unknown sample categories. Snell proposed a prototype network in 2017 [34], 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, and Sun added an attention mechanism to the prototype network in their subsequent work [35] [36], 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 [37] and subsequent strategies for fine-tuning the optimization of that parameter [38]. When the large 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.

III. Motivation Example

A. Test Cases

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 guideline, etc. Table I 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 organisations varies, we have selected three attributes, product_version_module, test process, and test requirements which is also called expected result, 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 requirement have a large number of test-related vocabulary inside, which can help find the relevant types of tests.

TABLE I
Example of 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.Showing 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 related information, 5.The system pops up ”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.
. . .	. . .

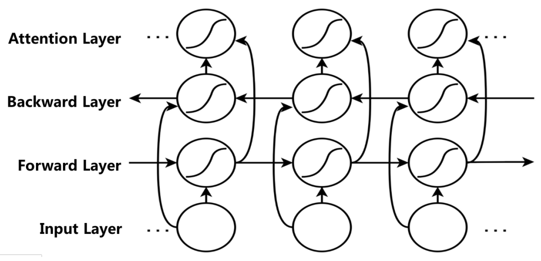

Fig. 1. Bi-LSTM with Attention’s Schematic Diagram

TABLE II
Example of attention words

Words	Attention Score
Click	**0.035**
Quit	**0.37**
Login	**0.43**
. . .	. . .
Open	**0.004**

B. 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 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 attention mechanism. TableII 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. BiLSTM with attention’s schematic diagram is shown as Fig1.

C. Lexicon

We use an attention-based LSTM to construct a set of critical words for text classification. Based on the attention weights of each category, the attention-based LSTM 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 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. TableIII shows an example of the set of keywords in each lexicon obtained from the initially trained classifier.

TABLE III
EXAMPLE OF LEXICONS

Type	Lexicon
Login and Logout	Login, 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......

D. Pseudo Labeled Test Cases

Pseudo labeled test cases are test cases that were originally unlabeled and, after 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 [39] have shown that expanding the training set with pseudo labeled test cases results in better classification of the trained classifier. TableIV 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, assume 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 and the matching numbers of words is 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 is equal to k1, so the classifier prediction category is assigned to TC_2 as a pseudo-label.

IV. FRAMEWORK

The framework 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 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 Fig2.

TABLE IV
TESTER CLASSIFICATION QUALITY REVIEW SCORE SHEET

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)**

A. 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 original test case data obtained is difficult to be understood by the computer, so it is necessary to extract the test case features after filtering the dirty data by various conditions for use 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 LTP tools, the Word2Vec method, and the Word Embedding method for training the classification model.

The whole data preparation stage is processed as shown in Algorithm 1. Firstly, this framework will filter the test case data, such as test cases with test categories or program exceptions, test cases with empty test requirements, and remove test cases with pending review status. (Line1) After that, the data of test cases will be augmented. This framework 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. (Line2-4) Immediately after that, this framework will use LTP to segment the test case text data into words, remove the deactivated words, filter the words that are not verbs or nouns, and unify the synonyms for feature extraction. (Line5-7) After that, the framework 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 framework will Generate dictionary D2, where the Key is the word, and the Value is the word's index. (Line8-11) After generating a dictionary, the framework trains the Word2Vec model on historical test case data, builds word packages, and saves them. Finally, the word embedding vectors

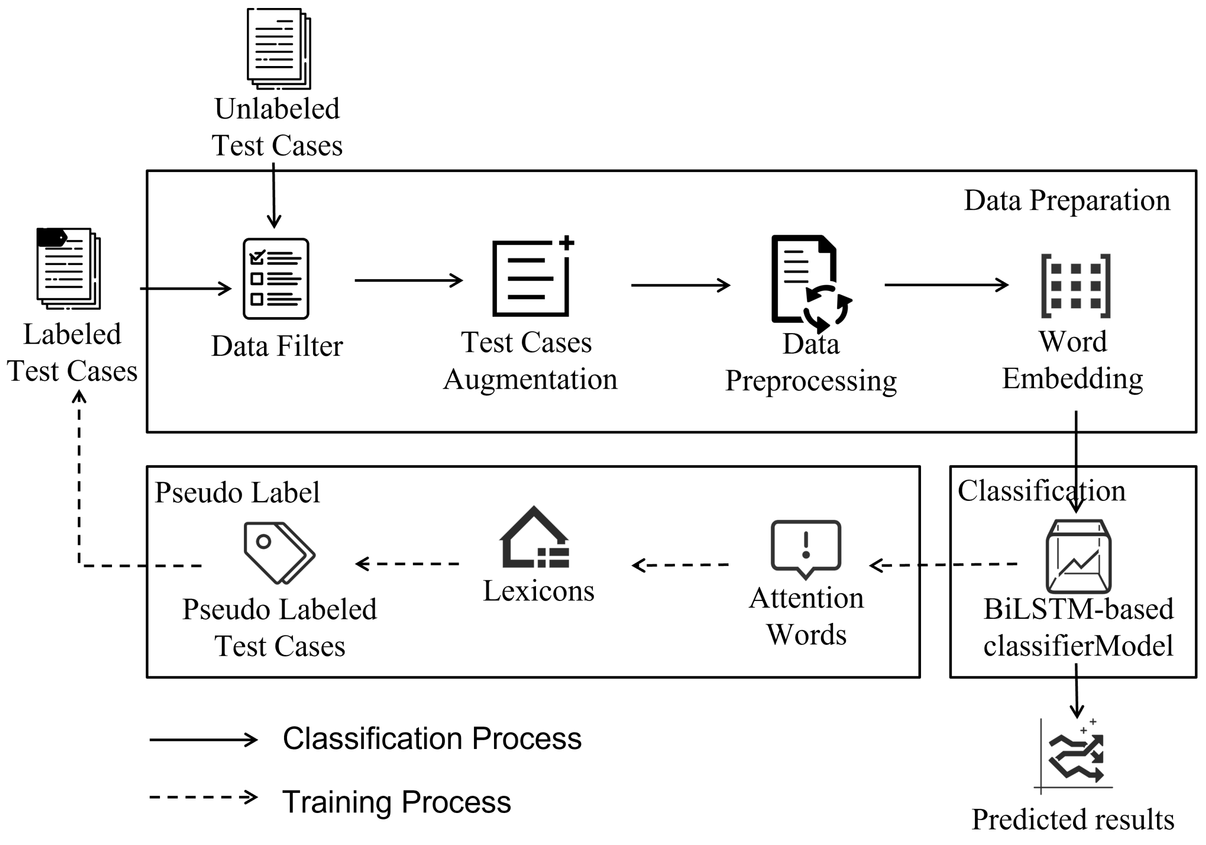

Fig. 2. The Framework of Few-Shot-Based Test Case Classification

```
Algorithm 1 Data Preparation
Input: Test case set $S$, length of Dictionary word2index $l1$,
    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: $D2 \Leftarrow$ wordindex($D1$)
12: for each test case $t2 \in S2$ do
13:     $M1 \Leftarrow$ word2index($t2$,$D1$)
14: end for
15: $M2 \Leftarrow$ embedding($M1$,$l1$,$l2$)
16: return $M2$;
```

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. (Line12-16)

B. Test Case Classification

The primary function of the test case classification stage is to train the bidirectional LSTM neural network model as the initial classification model for test cases using the historical test case word vectors obtained from the test case feature extraction phase.

First, build the bidirectional long and short-term memory neural network, define the model's input layer, hidden layer and forward computation, add attention layer into network.Then, obtain the word vector dataset of historical test case data in the document framework and divide the dataset into a training set consisting of labeled test case word vectors, a test set and a development set consisting of unlabeled test case word vectors, where the category lexicon collection module uses the development set. Finally, The initial classification model with the highest classification accuracy on the test set is obtained after several rounds of training using the training set.

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 LSTM layer is denoted as

$H : [h_1, h_2, \cdots, h_T]$. The weight matrix r obtained from its Attention layer is given by

$$M = tanh(H) \quad (1)$$

$$\alpha = softmax(w^T M) \quad (2)$$

$$r = H^T \quad (3)$$

$H \in R^{d^w \times T}$,d^ws the dimension of the word vector,w^T is a transpose of the parameter vector obtained from training learning.

C. Lexicon Collection

Important words tend to have higher attention weights in classification, and the attention mechanism also assigns higher weights to words that represent the input data categories. 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 algorithm 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 in the LSTM-based trained classifier to obtain a more extensive reliable training set, the unlabeled samples are transformed into samples with algorithm-assigned pseudo-labels to be used as training sets.

The primary function of the lexicon collection stage is to add the Attention mechanism to the bidirectional LSTM neural network, visualize the attention layer, focus on the attention weight of the words in the specific test cases in the calculation of the classification model, collect representative words for each category of the test cases, determine the test case categories by pattern matching methods in the case of low confidence of the initial classification model prediction, and improve the classification Accuracy.

The flow of the whole lexicon collection stage is shown in Algorithm 2. In this stage, we first add the attention layer to the bidirectional long- and short-term memory neural network to obtain the attention weights of the word vectors. After that, we obtain the word packets of the Word2Vec model saved in the test case feature extraction module from the data preparation stage and obtain the correspondence between the word vectors in the word packets and the actual words. After that, we initialize the category lexicon and initialize the word set of each category as empty. Initialize the test case confidence lexicon to empty and then sort the lexicon from highest confidence to lowest confidence. Obtain the initial classification model saved in the test case classification module, and obtain the test case word vector development set assigned to the test case classification module. (Line1-4)

After that, we use the development set as the input to the initial classification model. The classification model performs classification prediction on the test cases in the development set and obtains the classification categories and prediction confidence levels. (Line5-8) When the prediction confidence is less than q, the word vector is matched in 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, and the word vector of the predicted category is added to the transfer set. (Line9-11) When the prediction confidence is greater than or equal to q, the test case category is marked as the category predicted by the initial classification model and added to the test case confidence dictionary. The word vectors of the predicted category are added to the transfer set. (Line12-16) If the current word vector category is still not predicted according to the pattern matching rules, the word vector remains in the development set. (Line17-18)

After that, the top n test case word vectors with high confidence are selected in the test case-confidence dictionary. The top m words with great attention in each word vector are selected with the word package. These words are added to the word set of the category corresponding to that word vector. (Line22-24)

If the transfer set is not empty, the use case word vectors

Algorithm 2 Lexicon Collection

```
Input: Dictionary of word2index $D1$, development set of test
    cases' vectors $V1$, training set of test cases' vectors $V2$,
    confidence's threshold $q$, $k1$, $k2$
Output: classfication model $M$
 1: initialize Lexicon $l$
 2: initialize Dictionary of testcase-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:     $t1$, $c \Leftarrow$ predict($M$, $v$)
 7:     $t2$, $k \Leftarrow$ match($l$, $v$)
 8:     using Lexicon $l$
 9:     if $c < q$ and $k >= k2$ then
10:         $t2 \Leftarrow$ label $v$
11:         transfer($v$, $V1$, $V3$)
12:     else
13:         if $c >= q$ and $k >= k2$ then
14:             $t1 \Leftarrow$ 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: select top n confidence's test cases vectors $V4$ in $D2_i$
23: for test case's vector $v$ $in$ $V1$ do
24:     add top m attention's word to Lexicon $l$
25: end for
26: if $V3$ ! $= null$ then
27:     $V2 \Leftarrow V2$+$V3$
28:     back to Line 4
29: else
30:     save $M$
31:     return $M$;
32: end if
```

in the transfer set are transferred from the development set to the training set, and the initial classification model is retrained based on the training set. (Line25-27) If the transfer set is empty, the current initial classification model is saved as the final classification model, ending the process. (Line29-30)

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

(1) 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.

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

Algorithm 2 represents assigning weights to words, generating a lexicon, and acquiring pseudo-labels in the Lexicon Collection stage. After the test case 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.

V. Experiment

After verifying the availability and reliability of the test case management and classification system, we also need to evaluate the classification effect of the test case classification service, checking the effect of the classifier generated by the test case classification technique based on few-shot learning.

A. Research Questions

The empirical study is conducted to answer the following research questions.

RQ1: For the BiLSTM model with an added attention mechanism, can augmentation of the test cases improve the classification effect?

RQ2: Whether the category labels generated by the classifier in the small sample scenario are reasonable and accurate?

RQ3: Whether the generated category labels can help experts review test case categories?

B. Experiment Object

To explore the above questions, we evaluate our method on three datasets of test cases obtained from real crowdsourced testing competitions. The three test case datasets come from two different fields: entertainment and tools. The number of test cases in the dataset is shown in Figure 3.

C. Experiment Environment

We implemented our experiment on macOS 10.13, running on an Intel Core i5 CPU @2.7GHz with 2 cores and with RAM size of 8GB.

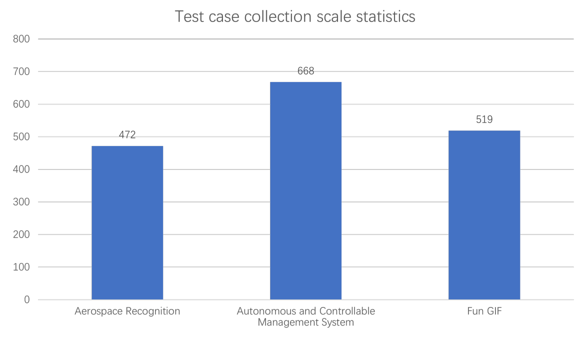

Fig. 3. Test case collection scale statistics of Crowdsourced Testing

D. Experiment Design

To verify whether augmentation of the test cases can improve the classification effect of the BiLSTM model with an added attention mechanism, we designed comparison experiments. We chose the same test case dataset. In the experimental procedure, the standard procedure augmented the test cases and then performed text processing and vectorization of the data; the comparison procedure did not augment the test cases but performed text processing and subsequent operations on the test cases directly.

In order 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 The Bayesian algorithm builds classifiers on three datasets. These classifiers use 20% of the data set as the training set for training and 80% 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 real scene, Count the accuracy of each classifier to see if our algorithm is superior.

Whether the class labels generated by the validating classifier can help experts review test case classes and testers manually classify test cases. We set up a comparative experiment and divided 12 testers into two groups, A and B. Group A used the test case classification service to generate category labels as a reference for manually classifying test cases, and group B did not use the classification service to manually classify test cases. We asked testers to manually fill in the categories for the test cases in the three sets of test cases and counted the time they spent manually completing the classification to explore whether the classifier could help reduce the cost of classifying test cases. At the same time, we invited three experienced testers to act as review experts to score the accuracy of the testers' manual classification of test cases, and to explore whether the classifier can help improve the classification quality of test cases.

VI. Experiment results

TableV shows the classification results of different algorithms on the same test case dataset as the training set, where the row data represent the accuracy of the classifier trained by each classification algorithm on different datasets, and the column data represent the classification accuracy of each dataset on the classifier trained by different classification The classification accuracy of each data set on the classifier trained by each classification algorithm is represented in the column data. We can see that without applying the test case augmentation method, here is a significant decrease in the classification results of this algorithm. The performance was particularly pronounced on the third item, where the accuracy dropped by 11% , and the average accuracy dropped by almost 10%.

Answer to RQ1: After removing the test case augmentation module, there is a significant decrease in the algorithm's accuracy. The test case augmentation can improve the classification effect for the BiLSTM model with an added attention mechanism.

TABLE V
LIST OF CLASSIFICATION ACCURACY RESULTS

Classification algorithm	**Aerospace Recognition Practice Competition**	**Autonomous and controllable management system for the final**	**Fun GIF crowd-sourced testing**
Our Algorithm	**0.51**	**0.57**	**0.48**
Algorithm without Simbert	0.44	0.50	0.37
SVM	0.47	0.51	0.44
KNN	0.33	0.42	0.32
Naive Bayes	0.20	0.24	0.17

TableV also shows that our algorithm has a classification accuracy of 57% on the best performing items and 48% on the worst items. Meanwhile, the best accuracies of other general classification methods are 51% and 44%, respectively. Our classification algorithm can effectively improve the accuracy of test case classification.

Answer to RQ2: Compared with existing algorithms, our algorithm has a significant improvement in inaccuracy. Therefore the category labels generated by the classifier in the small sample scenario are reasonable and accurate.

Table VI shows the average time statistics of the two groups of A and B testers when they are asked to classify unclassified test cases in different test case sets. It can be seen that with the help of the class labels generated by the classifier, the average statistical time of testers in group A is shorter than that of testers in group B by more than 6 minutes, which greatly improves the efficiency of manual classification of test cases.

Table VII shows the evaluation status of the three review experts on the classification quality of each test case set by each tester. The method of scoring and calculating the average score is used to reflect the classification. Quality, the score

TABLE VI
STATISTICS OF TESTER CLASSIFICATION TIME

Dataset	**Average classification time of group A**	**Average classification time of group B**
Aerospace Recognition Practice Competition	19.3min	25.1min
Autonomous and controllable management system for the final	25.2min	34.7min
Fun GIF crowdsourcing test	20.2min	27.9min

TABLE VII
TESTER CLASSIFICATION QUALITY REVIEW SCORE SHEET

Reviewer number	**1-A**	**1-B**	**2-A**	**2-B**	**3-A**	**3-B**
E1	8.2	7.4	8.3	7.7	8.1	7.2
E2	8.0	7.2	8.5	8.1	7.9	7.1
E3	8.5	8.3	9.1	8.6	8.3	7.7

can be from 1 to 10 points. It can be seen that the average score of testers in group A on the same dataset is higher than that of testers in group B, indicating that the class labels generated by the classifier can help improve the quality of manual classification test cases.

Answer to RQ3: By comparing the two control groups in terms of both test completion time and scores, we can confirm that the category labels generated using our framework is relatively accurate and can help experts review test case categories.

VII. 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 framework 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. In this paper, a category label review mechanism is introduced at the framework level to provide positive input feedback for the classifier's training. However, there is still a need to improve the algorithm for the collection of category lexicons at the algorithmic level. Based on experimental observations, there are often some of the same keywords collected from different categories in different types of lexicons, which are given higher attention weights when the test case classification model predicts different categories, and thus belong to the public cross-words between categories. In future work, we hope to avoid the misleading role of these words in the lexicon word matching mechanism by building a public thesaurus to collect keywords that occur frequently in each category and excluding words from the public lexicon in the word matching process.

Finally, the framework needs to train separate classifiers for each test case set of different topics. We still have not found a way to accurately classify all test cases using a single classifier. 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.

VIII. Conclusion

This paper proposes a test case classification framework based on Few-shot Learning. We use this algorithm to filter test cases, extract test case-related information, augment them, and then perform text processing and vectorization. After that, we introduced an attention mechanism on the BiLSTM model to construct a category lexicon of test cases and obtain the classification of test cases. Constructing the category lexicon of test cases can help iteratively train the model and obtain higher accuracy. We constructed a dataset consisting of three crowdsourced test projects with thousands of test cases. We also used this dataset for comparison experiments, which proved that our algorithm has a 5%-25% improvement in accuracy compared to existing algorithms. Also, with a control group, we verified that the category labels generated using the classification framework could help testers effectively improve their testing efficiency.Using our framework can better solve the problem of classifying the many test cases obtained from crowdsourcing tests and reduce the manual workload.

Acknowledgements

The authors would like to thank the anonymous reviewers for insightful comments. This work is supported partially by National Key R&D Program of China (2018YFB1403400) and Science, Technology and Innovation Commission of Shenzhen Municipality (CJGJZD20200617103001003).

References

[1] LU X, ZHENG B, VELIVELLI A, et al. Research Paper: Enhancing Text Categorization with Semantic-enriched Representation and Training Data Augmentation[J]. J. Am. Medical Informatics Assoc., 2006, 13(5) : 526 – 535.

[2] MA J, LI L. Data Augmentation For Chinese Text Classification Using Back-Translation[J]. Journal of Physics: Conference Series, 2020, 1651(1) : 012039.

[3] WEI J W, ZOU K. EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks[C] // INUI K, JIANG J, NG V, et al. 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, Hong Kong, China, November 3-7, 2019. [S.l.] : Association for Computational Linguistics, 2019 : 6381 – 6387.

[4] ZHANG X, ZHAO J J, LECUN Y. Character-level Convolutional Networks for Text Classification[J]. CoRR, 2015, abs/1509.01626.

[5] KOBAYASHI S. Contextual Augmentation: Data Augmentation by Words with Paradigmatic Relations[C] // WALKER M A, JI H, STENT A. Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 2 (Short Papers). [S.l.] : Association for Computational Linguistics, 2018 : 452 – 457.

[6] HOUY,LIUY,CHEW,etal.Sequence-to-Sequence Data Augmentation for Dialogue Language Understanding[C] // Proceedings of the 27th International Conference on Computational Linguistics. Santa Fe, New Mexico, USA : Association for Computational Linguistics, 2018 : 1234 – 1245.

[7] WEI Xiaona, LI Yinghao, WANG Zhenyu, LI Haozun, WANG Hongzhi. Methods of training data augmentation for medical image artificial intelligence aided diagnosis[J]. Journal of Computer Applications, 2019, 39(9): 2558-2567.

[8] SU J. SimBERT: Integrating Retrieval and Generation into BERT[R/OL]. 2020. https://github.com/ZhuiyiTechnology/simbert.

[9] Aggarwal, Charu C., and ChengXiang Zhai. "A survey of text classification algorithms." Mining text data. Springer, Boston, MA, 2012. 163-222.

[10] Kowsari, Kamran, et al. "Text classification algorithms: A survey." Information 10.4 (2019): 150.

[11] Shafiabady, Niusha, et al. "Using unsupervised clustering approach to train the Support Vector Machine for text classification." Neurocomputing 211 (2016): 4-10.

[12] Miyato, Takeru, Andrew M. Dai, and Ian Goodfellow. "Adversarial training methods for semi-supervised text classification." arXiv preprint arXiv:1605.07725 (2016).

[13] Kadhim, Ammar Ismael. "Survey on supervised machine learning techniques for automatic text classification." Artificial Intelligence Review 52.1 (2019): 273-292.

[14] Minaee, Shervin, et al. "Deep learning–based text classification: a comprehensive review." ACM Computing Surveys (CSUR) 54.3 (2021): 1-40.

[15] Wang, Gang, et al. "Sentiment classification: The contribution of ensemble learning." Decision support systems 57 (2014): 77-93.

[16] Do, Chuong B., and Andrew Y. Ng. "Transfer learning for text classification." Advances in neural information processing systems 18 (2005).

[17] Zhang, Tianyang, Minlie Huang, and Li Zhao. "Learning structured representation for text classification via reinforcement learning." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 32. No. 1. 2018.

[18] Lei, Kai, et al. "Tag recommendation by text classification with attention-based capsule network." Neurocomputing 391 (2020): 65-73.

[19] Melville, Prem, Wojciech Gryc, and Richard D. Lawrence. "Sentiment analysis of blogs by combining lexical knowledge with text classification." Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. 2009.

[20] Zhao, Ran, Oscar J. Romero, and Alex Rudnicky. "SOGO: a social intelligent negotiation dialogue system." Proceedings of the 18th International Conference on intelligent virtual agents. 2018.

[21] Liu, Gang, and Jiabao Guo. "Bidirectional LSTM with attention mechanism and convolutional layer for text classification." Neurocomputing 337 (2019): 325-338.

[22] FE-FEI L, OTHERS. A Bayesian approach to unsupervised one-shot learning of object categories[C] // proceedings ninth IEEE international conference on computer vision. 2003 : 1134 – 1141.

[23] WANG Y,YAO Q,KWOKJ T,etal. Generalizing from a few examples: A survey on few-shot learning[J]. ACM computing surveys (csur), 2020, 53(3) : 1 – 34.

[24] BENAIM S, WOLF L. One-shot unsupervised cross domain translation[J]. advances in neural information processing systems, 2018, 31.

[25] QI H, BROWN M, LOWE D G. Low-shot learning with imprinted weights[C] // Proceedings of the IEEE conference on computer vision and pattern recognition. 2018 : 5822 – 5830.

[26] SHYAM P, GUPTA S, DUKKIPATI A. Attentive recurrent comparators[C] // International Conference on Machine Learning. 2017 : 3173 – 3181.

[27] ZHANG Y, TANG H, JIA K. Fine-grained visual categorization using meta-learning optimization with sample selection of auxiliary data[C] // Proceedings of the european conference on computer vision (ECCV). 2018 : 233 – 248.

[28] KOZERAWSKI J, TURK M. Clear: Cumulative learning for one-shot one-class image recognition[C] // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018 : 3446 – 3455.

[29] PFISTER T, CHARLES J, ZISSERMAN A. Domain-adaptive discriminative one-shot learning of gestures[C] // European Conference on Computer Vision. 2014 : 814 – 829.

[30] MILLERE G,MATSAKISN E,VIOLAP A.Learning from one example through shared densities on transforms[C] // Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No. PR00662) : Vol 1. 2000 : 464 – 471.

[31] LIU B,WANG X,DIXIT M,etal.Feature space transfer for data augmentation[C] // Proceedings of the IEEE conference on computer vision and pattern recognition. 2018 : 9090 – 9098.

[32] GOODFELLOW I, POUGET-ABADIE J, MIRZA M, et al. Generative adversarial nets[J]. Advances in neural information processing systems, 2014, 27.

[33] KOCH G,ZEMEL R,SALAKHUTDINOV R,etal.Siamese neural networks for one-shot image recognition[C] // ICML deep learning workshop : Vol 2. 2015 : 0.

[34] SNELL J, SWERSKY K, ZEMEL R. Prototypical networks for few-shot learning[J]. Advances in neural information processing systems, 2017, 30.

[35] GAO T, HAN X, LIU Z, et al. Hybrid attention-based prototypical networks for noisy few-shot relation classification[C] // Proceedings of the AAAI Conference on Artificial Intelligence : Vol 33. 2019 : 6407 – 6414.

[36] SUN S, SUN Q, ZHOU K, et al. Hierarchical attention prototypical networks for few-shot text classification[C] // 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 : 476 – 485.

[37] YU M, GUO X, YI J, et al. Diverse few-shot text classification with multiple metrics[J]. arXiv preprint arXiv:1805.07513, 2018.

[38] HOFFMAN J,TZENG E,DONAHUE J,etal.One-shot adaptation of supervised deep convolutional models[J]. arXiv preprint arXiv:1312.6204, 2013.

[39] Lee, Ju-Hyoung, Sang-Ki Ko, and Yo-Sub Han. "SALNet: Semi-supervised Few-Shot Text Classification with Attention-based Lexicon Construction." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 35. No. 14. 2021.