The Journal of Systems & Software 195 (2023) 111514

Contents lists available at ScienceDirect

The Journal of Systems & Software

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

SCGRU: A general approach for identifying multiple classes of self-admitted technical debt with text generation oversampling ☆

Kuiyu Zhu [a,*,1], Ming Yin [a,1], Dan Zhu [b], Xiaogang Zhang [a], Cunzhi Gao [a], Jijiao Jiang [c]

[a] School of Software, Northwestern Polytechnology University, 710072 Xi'an, China
[b] Debbie and Jerry Ivy College of Business, Iowa State University, Ames 50011, USA
[c] School of Management, Northwestern Polytechnology University, 710072 Xi'an, China

ARTICLE INFO

Article history:
Received 17 July 2021
Received in revised form 18 July 2022
Accepted 12 September 2022
Available online 17 September 2022

Keywords:
Software quality
Self-admitted technical debt
Multi-classification
Text generation oversampling

ABSTRACT

Identifying self-admitted technical debt (SATD) plays an important role in maintaining software stability and improving software quality. Although existing methods can detect SATD and researchers have identified design debt and requirement debt, an approach to realize multiple classification of SATD, including defect, test, and documentation, is still lacking. In this paper, we combine text generation oversampling and the Convolutional Neural Networks-Gated Recurrent Unit (CNNGRU) model, and propose an approach called SCGRU to classify multiple debt, including defect, test, documentation, design, and requirement. First, SeqGAN-based text generation is employed to generate new samples by learning the original SATD data, thereby increasing the number of SATD samples such as defect debt and reducing data imbalance. Then, we apply the CNNGRU model to refine SATD into multiple classes. An experiment with cross-project identification of 10 projects shows that our approach is more effective than existing methods such as CNN and text mining. The proposed SCGRU approach has strong advantages especially in cases of flawed debt with very unbalanced data such as test debt and documention debt.

1. Introduction

Technical debt was proposed by Cunningham to achieve short-term goals and adopt temporary and immature solutions (Cunningham, 1993). Although it can temporarily achieve certain goals, the use of temporary solutions brings great hidden dangers to the software in the long run. Sometimes, developers intentionally introduce technical debt and mark it in a comment to submit code or solutions as soon as possible. This intentionally introduced debt is called self-admitted technical debt (SATD). Experienced programmers tend to introduce SATD (Potdar and Shihab, 2014). The debt, whether introduced intentionally or unintentionally, will have a great impact on software maintenance (Zazworka et al., 2011). Therefore, how to better identify these debt has become a current research hot spot (Sierra et al., 2019).

Previous work (Potdar and Shihab, 2014) conducted research on source code comments of four large open-source software projects. Researchers manually processed 101,762 source code comments and proposed 62 patterns to detect SATD. However, detecting SATD by summarizing patterns in large-scale projects is difficult. The pattern-based method needs a great deal of time and manpower. Fully detecting SATD by using the pattern-based method is difficult, and certain SATD in rare mode[2] can hardly be summarized manually. da S. Maldonado et al. (2017) proposed a natural language processing method to automatically detect design debt and requirement debt. However, this method requires sufficient data to train the maximum entropy classifier, and it cannot detect a small amount of test debt, defect debt and documentation debt. Huang et al. (2018) detected SATD by using text mining. Feature selection and compound classifiers were combined to improve the detection accuracy to a certain degree. However, some useful information may be filtered by feature selection. Ren et al. (2019) proposed five key issues that affect SATD detection and proposed a CNN method to identify SATD. More importantly, the computational structure of CNN was used to extract the key phrases that identify SATD through

☆ Editor: Neil Ernst.

* Corresponding author.
E-mail addresses: zhuky@mail.nwpu.edu.cn (K. Zhu), yming@nwpu.edu.cn (M. Yin), dzhu@iastate.edu (D. Zhu), xgznwpu@163.com (X. Zhang), gaocunzhi@mail.nwpu.edu.cn (C. Gao), jjj_leon@nwpu.edu.cn (J. Jiang).

[1] Kuiyu Zhu and Ming Yin contributed equally to this work.

[2] During software development, developers tend to use pattern words such as "todo", "hack", and "fixme" to represent SATD. In addition to these commonly used SATD patterns, there are many less obvious SATD patterns, such as "may", "unnecessary", "revisited", etc. Pattern-based method detection cannot enumerate all of these patterns (Ren et al., 2019).

https://doi.org/10.1016/j.jss.2022.111514
0164-1212/

backtracking, which improves the interpretability of the method. However, this method could detect only whether it is SATD or not.

Most existing research on SATD focuses on how to identify SATD and whether it is SATD or on classifying technical debt into design debt and requirement debt. One of the reasons SATD cannot be classified into other types of technical debt is that the number of minority samples is severely insufficient. In the absence of samples, training the classifier to achieve accurate classification is difficult (da S. Maldonado et al., 2017). The extreme imbalance of SATD data in comments also brings great difficulties to the classification of SATD. Currently, the text generation method has proven effective in addressing the imbalance problem caused by insufficient training samples (Goodfellow et al., 2014). Li et al. (2018b) used a generative adversarial network (GAN) to generate labeled sentences to expand the data. Wang and Wan (2018) proposed a SentiGAN structure with a multiclass discriminator and multiple generators, which can generate text data with different emotional labels. Xie and Zhang (2018) used GAN to simulate the distribution of original samples to generate new samples. Xu et al. (2018) proposed a new text generation model called diversity-promoting GAN to generate richer and more diverse texts. Although GAN has a significant effect on generating real-value data, it has limitations on text data (Yu et al., 2017). The samples generated using GAN do not have diversity and cannot effectively change the distribution characteristics of the data. Luo et al. (2019) used sequence GAN with policy gradient (SeqGAN) to generate stock comment samples, which solved the problem of fewer samples when predicting the sentiment of shareholders. This method obtains richer and more diverse results compared with the samples generated by GAN. In addition, deep learning has been widely used in text classification, achieving remarkable effects (Yilmaz and Toklu, 2020; Kalchbrenner et al., 2014). Kim (2014) proposed that convolutional neural network (CNN) can be used for sentence classification, showing that a simple CNN achieves excellent results on multiple text tasks with several hyperparameter adjustments. Combining the powerful local feature extraction ability of CNN and the accurate context-dependent extraction ability of LSTM, She and Zhang (2018) proposed a hybrid model of CNN and LSTM for text classification, which effectively improved the classification accuracy. GRU (Cho et al., 2014) is a variant of LSTM (Hochreiter and Schmidhuber, 1997) and its structure is simpler than that of LSTM while retaining the effect[3] of LSTM (Xianjing and Yong, 2018). The minority class samples are merged with the original training set to achieve a relatively balanced state with the majority class samples, thereby reducing the difficulty of classification.

In this paper, we propose an approach called SCGRU, which combines SeqGAN with the CNNGRU model to identify multiple classes of SATD. SeqGAN is used to oversample the SATD to solve data imbalance among five types of technical debt, and CNNGRU is used to identify multiple classes of SATD. SCGRU can not only effectively identify multiple classes of SATD, including design, requirement, defect, documentation, and test debt, but also has a significant performance. We successfully identified documentation debt and defect debt that cannot be identified using existing methods and proved the effectiveness of SCGRU.

Our contributions are as follows.

- The proposed SCGRU approach could successfully identify five SATDs, namely, design debt, requirement debt, defect debt, documentation debt, and test debt.
- To solve the problem of data imbalance in SATD comments, we propose a text generation oversampling method to expand the minority samples.
- Our approach could effectively identify SATD with a small sample size, such as test debt, defect debt, and documentation debt.

The rest of this paper is structured as follows. Section 2 reviews the related work. Section 3 describes our approach for identifying multiple classes of SATD and the experimental process. Section 4 reports and analyzes the experimental results. Section 5 presents some discussions related to our research. Section 6 analyzes threats to validity. Section 7 concludes the paper and shows future work.

2. Related work

2.1. Data imbalance problem using sample

Research on SATD identification faces extreme data imbalance, and the number of SATD samples in comments is far less than that of non-SATD samples. Data imbalance will bring great difficulties to classifiers. In most cases, the algorithm tends to be biased toward a large number of types, resulting in a decline in the accuracy of overall classification (Last et al., 2017; Lázaro et al., 2015; Mahmoud et al., 2020). Recent research has dealt with the problem of imbalance. Huang et al. (2018) used ensemble text mining to select important features to solve the problem of data imbalance. Ren et al. (2019) dealt with data imbalance in SATD at the algorithm level. They designed a weighted cross-entropy loss function by assigning corresponding weights for each class to punish the misclassification instances of minority samples. However, no real increase occurred in the number of SATD samples and the distribution of data did not change. Thus, the problem still could not be effectively solved.

Currently, resampling, synthesizing minority class oversampling technique (SMOTE), and text generation of oversampling are applied to solve the imbalance problem at the data level. Resampling is commonly used at the data level to deal with data imbalance (Yap et al., 2013). It includes oversampling and undersampling. Oversampling is a simple replication strategy to increase minority samples. However, it tends to lead to overfitting of the model (Lee et al., 2018), which makes the information learned by the model too specific and difficult to generalize. Li et al. (2018a) proposed a new oversampling technique to solve the imbalance problem. The method creates synthetic text from a word space, which can be used directly for text. Moreo et al. (2016) generated new random minority samples according to the distribution properties of terms in the set, which has achieved a certain effect on the text. Undersampling is a process in which the majority class samples are discarded to achieve balance with the minority class samples. However, the discarded samples may be key samples, and undersampling may lose a large amount of useful information (Ha and Lee, 2016; Batista et al., 2004). SMOTE is a data oversampling algorithm based on the k-nearest neighbor algorithm, which artificially synthesizes new samples through analyzing minority class samples (Chawla et al., 2002). However, SMOTE has some blindness when selecting nearby neighbors. In addition, it cannot overcome the problem of data distribution in unbalanced datasets and easily leads to distribution marginalization. More importantly, SMOTE synthesizes minority class samples on the basis of numerical feature space, which is not suitable for text data (Li et al., 2018a). Recently, the method of generating majority class samples has

[3] They can all be applied to the processing of time series data. And both GRU and LSTM have an internal mechanism of gates, which can regulate the flow of information, even though they have a different number of gates. These gates can understand which data in the sequence is important to retain or discard. For example, the update gate in the GRU can achieve a function similar to the forget gate and input gate in LSTM. It controls the extent to which the state information at the previous moment is retained to the current state. The larger the value, the more information is retained.

been proven to be effective for the imbalance problem. GAN has achieved excellent results in generating image samples (Lv et al., 2018; Wang et al., 2017). However, problems exist when generating text data through GAN. First, GAN is not suitable for generating non-real data (Huszar, 2015). Second, GAN can only give corresponding losses for the whole sequence in the process of generating samples. This method is not appropriate for text data where a partial generation sequence needs to be measured. SeqGAN can solve this problem well. Luo et al. (2019) proposed a SeqGAN-based preprocessing method to deal with data imbalance when studying the emotional tendency of stock investors. Douzas and Bação (2018) used conditional GANs to synthesize minority samples and achieved remarkable results.

In this paper, we use SeqGAN to generate a text sequence similar to the minority class and increase the number of minority samples in the SATD to achieve the oversampling of SATD. The features that can be extracted are increased and the identification efficiency of SATDs is improved using this method. However, because the extreme imbalance between positive and negative samples of some types, sampling a certain number of SATDs to the same number as non-SATDs will produce a large number of redundant samples. For this reason, we recommend oversampling with a sampling rate less than 1.

2.2. Classified self-admitted technical debt

Technical debt brings great difficulties to the maintenance of software projects. Potdar and Shihab (2014) proposed the concept of SATD in source code comments. They identified 62 patterns for SATD detection by using srcML to extract comments in four large Java projects. da S. Maldonado et al. (2017) divide SATD into five types: design, defect, documentation, requirement, and test debt.

Current research on the identification of SATD can be divided into two aspects. The first is SATD detection. Potdar and Shihab (2014) proposed a pattern-based method for SATD detection. To detect SATD automatically and reduce manual participation, Huang et al. (2018) proposed a method based on text mining by extracting features through feature selection and constructing a compound classifier to detect SATD. Flisar and Podgorelec (2019) improved the ability to detect SATD by training a word embedding model when obtaining source code comments of open projects and then filtering useful features by using feature enhancement. Ren et al. (2019) adopted CNN to detect SATD after vectorizing the text, which improved the accuracy and interpretability of detection. Maipradit et al. (2020b) define a type of SATD called an "on-hold" type of technical debt that includes a condition to indicate that the developer is waiting for an event or an updated feature that has been implemented elsewhere. They designed a natural language processing-based classifier to automatically identify this type of SATD to detect the specific event developers are waiting for. Further, (Maipradit et al., 2020a) conducted an empirical study to explore the way to automatically detect on hold SATD. Based on auto-sklearning and n-gram, they built a classifier for automatic recognition of on hold SATD, and the average F1 value reached 0.73. Recently, some researchers tend to study SATD in specific systems. Xiao et al. (2021) extended the research of SATD beyond the source code and studied the SATD in the construction system. By analyzing 500 SATD comments extracted from 291 GitHub knowledge bases using Maven to build the system, they manually classified them according to their location, cause and purpose, and trained a SATD classifier to detect two of the most common causes and four of the most common purposes for detecting SATD in the comment content of the system. Muse et al. (2022) believed that technology debt is related to data-intensive systems, so they conducted large-scale empirical research on data-intensive systems to explore their universality and persistence. In addition, they identified 15 new SATD categories, 11 of which are unique to database access operations. The second is SATD classification. da S. Maldonado et al. (2017) identified design and requirement debt by using NLP. Wattanakriengkrai et al. (2018) identified design debt and requirement debt by combining n-gram and IDF as a feature extraction tool. After undersampling, random forest was used for classification. Santos et al. (2020) constructed a word vector model through word2vec and classified design debt and requirement debt by using LSTM. Our approach is conducted by oversampling minority samples to increase the amount of defect debt, document debt, and test debt that considerably lack data so that the learning algorithm can learn more features, thereby realizing the identification of multiple classes of SATD.

3. Approach

3.1. Framework of SCGRU

The SCGRU approach consists of three parts, as shown in Fig. 1. In the first part, we increase the number of the SATD samples in the comments of source code by using SeqGAN-based oversampling to bridge the extreme gap between SATD and non-SATD comment texts. When we perform cross-project prediction of each technical debt, we sample the minority samples in the training set 10-fold. After oversampling, the whole training set is formed together with non-SATD. The GloVe model is used to map words to a high-dimensional space to be expressed as a comment matrix after preprocessing. The second part trains the CNNGRU model, which is also called CGRU. The comment matrix is extracted through the filter of the CNN convolution layer to extract n-gram to obtain the local features of comments, then the pooling layer selects the most important features, and the GRU layer extracts the context features of comments to train the CGRU model. The third part is to identify multiple classes of SATD. We preprocess the test set and express it as a comment matrix through the GloVe model and then input it into the trained CGRU model to identify five types of SATD in the test set.

3.2. Oversampling SATD data

3.2.1. Oversampling

One of the reasons existing methods could not identify three types of technical debt such as defect debt, test debt, and documentation debt is the considerable lack of sample size, which leads to the serious imbalance of sample distribution and increases the difficulty of classification. The data used are text data. To realize the identification of multiple classes of SATD, the minority samples are oversampled by text generation to increase its number and reduce the imbalance of the dataset, especially for a small amount of document debt, test debt, and defect debt. To solve the problem of extreme data imbalance, we use SeqGAN to generate samples of SATD. SeqGAN is proposed to solve the problems of the generator having difficulty transferring gradient updates and the discriminator having difficulty evaluating incomplete sequences when GAN is processing discrete data. SeqGAN regards the whole GAN as a reinforcement learning system and updates the parameters of the generator with the policy gradient algorithm. In addition, SeqGAN draws on the idea of Monte Carlo tree search to evaluate the quality of the text generated at the current time to evaluate the incomplete sequence at any moment. This oversampling method has been proven effective in text (Luo et al., 2019). We employ SeqGAN to oversample the SATD comment of the training set and use the oversampled data obtained to train the model and perform the prediction on the test set.

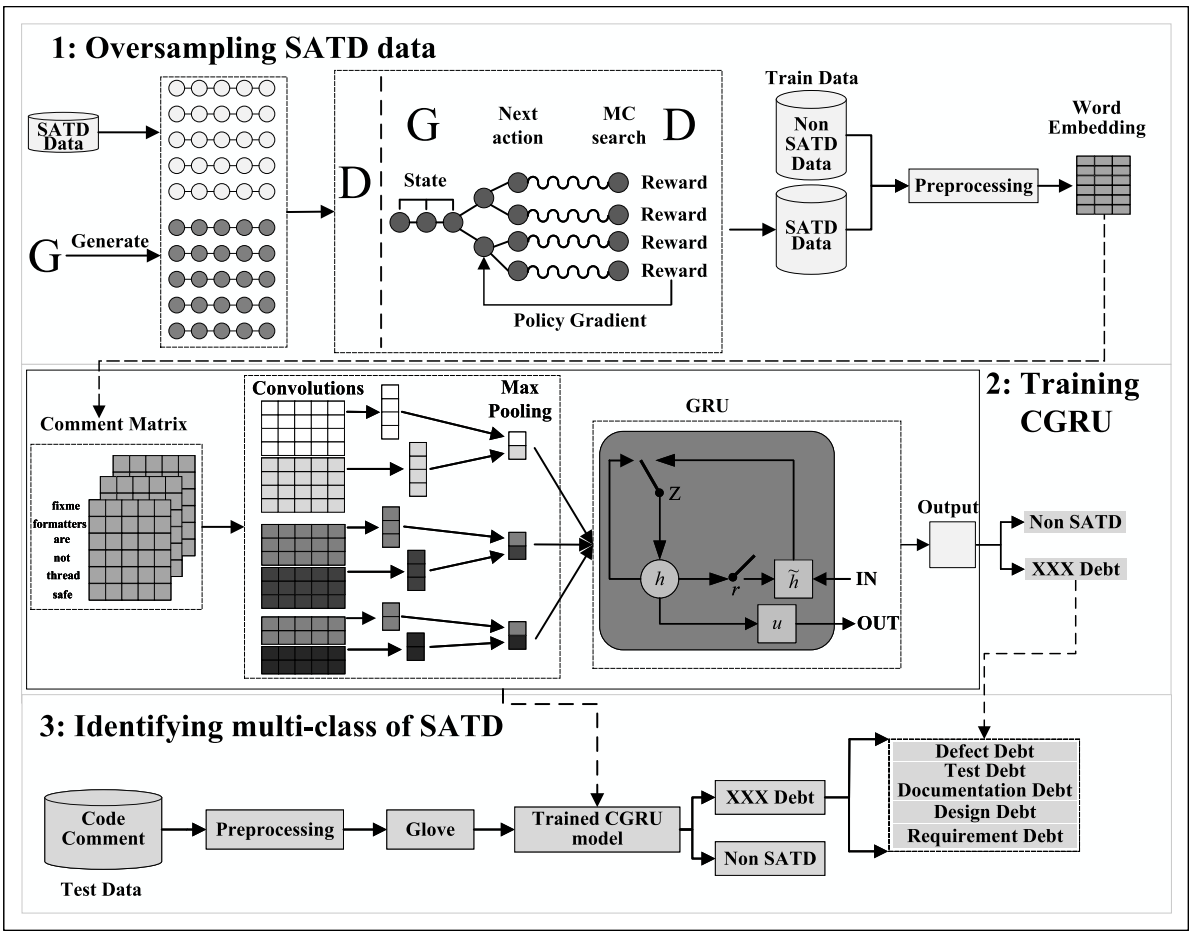


Fig. 1. This is a framework of SCGRU for identifying multiple classes of SATD. SCGRU consists of three parts, including oversampling SATD data, training CGRU model, and identifying multi-class of SATD. First, the SATD in the training set will be oversampled by SeqGAN to reduce the impact of data imbalance on the classification process. Then, the oversampled dataset is used to train a model composed of CNN and GRU. Finally, the trained model is used to predict SATD categories in the test set, such as defect debt, test debt, document debt, design debt, and demand debt.

3.2.2. Preprocessing

The preprocessing of comments includes build word index, texts to sequences, and padding.

Build word index: We use the tokenizer to spilt words to ensure that the words in each comment exist in the established dictionary. Before this work, necessary processing is performed, which includes converting text letters uniformly to lowercase and counting and sorting word frequency.

Texts to sequences: The vector sequences of each comment are output according to each original SATD data sample. At this time, the sequence is of indefinite length, and the length is the same as that of the original sample.

Padding: The length of each sample is unified to the same length. The length of the comment text in the dataset is mostly between 0 and 128, as shown in Table 1. If the maximum length is too small, a certain number of comments will be truncated. Although a large maximum length value can be used, it will increase the calculation overhead. According to statistics, the comment length within 128 accounts for 99.88% of all samples. And those greater than 128 only accounted for 0.12%. That is, 0.12% sequences are cropped due to the threshold of 128 words. Also, by convention maxlen is usually chosen to be an exponential multiple of 2 (Cunha et al., 2021). Therefore, in this paper, we have a uniform length of 128. We set the parameter maxlen to 128 to indicate the length of the comment sentence, which can cut the sentence with more than 128 words in the text comment and keep it at 128. For sentences with less than 128 words, we add them to 128. Use zero padding at the beginning until the comment length reaches the maximum. And Sentence length padding is to maintain consistency by adding zero tensor (zero tensor is a placeholder that does not represent any information) to avoid the existence of empty elements (Li et al., 2020). Adding a zero tensor at the beginning has the same effect as adding at the end. For consistency, this article chooses to padding at the beginning.

Table 1
Statistics of the sequence length in 10 projects.

Projects	16	32	64	128	More than 128
Ant	3,484	436	156	19	3
ArgoUML	7,637	1,248	436	111	20
Columba	5,889	477	84	14	4
EMF	3,562	567	216	40	5
Hibernate	2,337	453	139	33	6
JEdit	9,847	343	99	27	6
JFreeChart	4,115	247	36	5	5
JMeter	7,232	619	177	22	7
JRuby	4,897	4,242	485	134	30
SQuirrel	5,930	864	324	82	15
Total	54,275	5,739	1,801	383	77
% of different length	87.14%	9.21%	2.89%	0.61%	0.12%

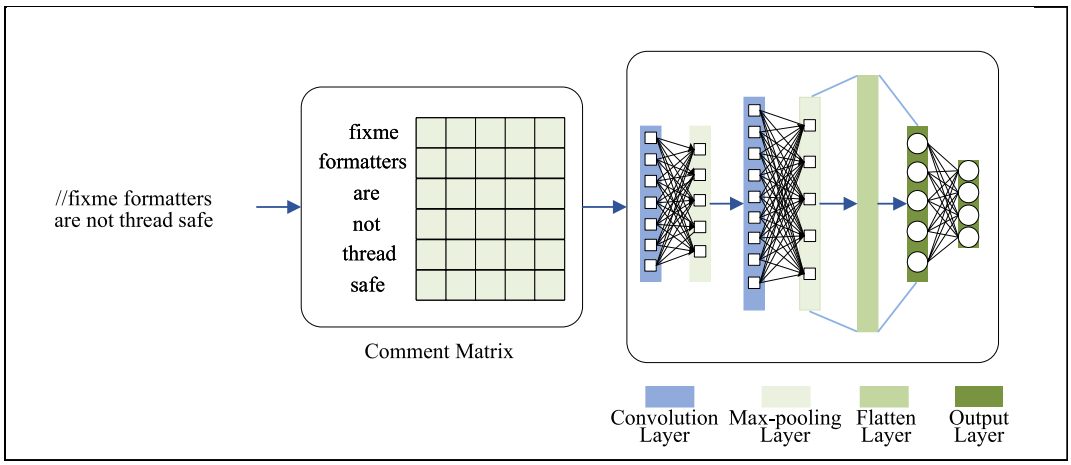


Fig. 2. An example of a CNN Architecture for comment text classification. In this case, a source code comment in the form of a sequence of words is represented as a comment matrix by word embedding. Then, the comment matrix will be extracted and concatenated by the convolution layer, pooling layer, fully connected layer and output layer of CNN, and finally classified as SATD or non-SATD.

3.2.3. Word embedding

We use the GloVe[4] model to train word vectors. The GloVe model is employed to obtain word representation by decomposing the "word–word" co-occurrence matrix. The calculation of the GloVe model is simple, which can accelerate the training speed of the model. In addition, unlike in word2vec, the weight function is introduced to the GloVe model, which can control the relative weight of words and will not lead to excessive weight of words with high exposure.

3.3. Training CNNGRU

CNN (Kim, 2014) can be used to extract multidimensional features by setting weights and obtain local key information through the pooling layer. Compared with CNN, GRU could more accurately grasp the overall information of the sentence and will not pay much attention to the local information and cause information loss. Our model is composed of CNN and GRU to accurately obtain the local features of SATD comments and effectively grasp the comment context.

A. CNN model

CNN is used to obtain the key features of comments. The convolution and maximum pooling process involves a one-dimensional filter traversing a matrix of SATD annotated word vectors to detect SATD features at various locations and obtain maximum features. Fig. 2 shows an example for comment text classification. CNN architecture is usually composed of convolution layer, pooling layer and full connection output layer.

We need to represent the input text as a comment matrix for convolution and pooling. Specifically, If $W_i \in R^d$ is the word vector corresponding to the i th word in a comment and d is the dimension of the word embedding, then $W \in R^{n\times d}$ represents the entire comment of the input sequence, where n represents the length of the comment. Then, the word vector comment matrix can be expressed as

$$Comment_w = w_1 \oplus w_2 \oplus \cdots \oplus w_n \tag{1}$$

In Eq. (1), where w_n is the nth word vector of $Comment_w$, and $\oplus$ represents the connection operator. In convolution layer, we use the window size of the filter is m and the width is d to extract n-gram features. Then obtained feature map $Conv_i$ as follows

$$Conv_i = f(Comment_w \cdot x_{i:i+d-1} + b) \tag{2}$$

In Eq. (2), $b \in R$ is the bias, $Comment_w$ is the comment matrix and f is the nonlinear activation function, we use *ReLU* as a activation function. Although Leaky ReLU is theoretically better than ReLU, according to a large number of studies (Adem, 2022; Liang and Xu, 2021; Maguolo et al., 2021; Tanaka, 2020), there is insufficient evidence in practice that Leaky ReLU is always better than ReLU, and ReLU is better than Leaky ReLU in some cases[5]. In addition, the ELU can output the negative values for the negative inputs, it is useful to reduce the mean of the outputs to zero but as the side effect the inputs of the ELU have unnecessary negative values. This is not good for the generalization. Dubey and Jain (2019) conducted a comparative study on the ReLU and Leaky-ReLU activation functions of convolutional neural networks and found that the effect of the activation function was related to the evaluation index. Specifically, to extract more features, multiple convolution cores are selected to obtain feature graphs that represent different feature information, which are arranged in depth. In addition, dropout is added after the convolutional layer to prevent overfitting.

Pooling also becomes downsampling or subsampling, including mean pooling and maximum pooling. In our example, we use the maximum pooling layer to obtain the maximum characteristic value after convolution, which can suppress the phenomenon of mean deviation caused by network parameter error and better extract information.

B. GRU model

The local information features extracted by the CNN layer are used as the input of GRU to learn the sequence relationship of comments. GRU (Cho et al., 2014) uses a gating mechanism to control input, memory, and other information, and make predictions at the current time step. GRU contains only two gates: update gate and reset gate. The update gate z_t is shown in Eq. (3), which x_t represents the input vector of the current time t and h_{t-1} represents the input vector of the previous time $t-1$. Its function is similar to that of the forget gate and input gate in LSTM (Hochreiter and Schmidhuber, 1997), which determines

[4] BERT is better for sentences with complex structures and more ambiguous words, and GloVe is better for sentences with many unregistered words (Arora et al., 2020). Relatively speaking, compared with standard text, we also found that the semantic requirements in annotations are not very high.

[5] In our scenario, ReLU outperforms LeakReLU. (Ide and Kurita, 2017) study showing that ReLU can intrude the problem of unnecessary growth of the inputs, but using regularized ReLU solves this problem. In the future, we will try to use regularized ReLU as the activation function.

Table 2
Amount of technical debt and their imbalance rate of 10 projects.

Projects	Defect	Test	Documentation	Design	Requirement	No-Technical debt
Ant	13	10	0	95	13	3,967
ArgoUML	127	44	30	801	411	8,039
Columba	13	6	16	126	43	6,264
EMF	8	2	0	78	16	4,286
Hibernate	52	0	1	355	64	2,496
JEdit	43	3	0	196	14	10,066
JFreeChart	9	1	0	184	15	4,199
JMeter	22	12	3	316	21	7,683
JRuby	161	6	2	343	110	4,275
SQuirrel	24	1	2	209	50	6,929
Total	472	85	54	2,703	757	58,204
Imbalance rate	123.3	684.8	1077.9	21.5	76.9	–

what information needs to be forgotten and which information needs to be added. Especially, σ is the activation function as shown in Eq. (4). The reset gate r_t is shown in Eq. (5), which determines how to combine the newly input comment information with the previous memory. The reset gate also controls how much information from the previous state is written to the current candidate set $\widetilde{h}_t$, as shown in Eq. (6), and the hidden state h_t is shown in Eq. (7).

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t]) \tag{3}$$

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{4}$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t]) \tag{5}$$

$$\widetilde{h}_t = \tanh(W \cdot [r_t \circ h_{t-1}, x_t]) \tag{6}$$

$$h_t = (1 - z_t) \circ h_{t-1} + z_t \circ \widetilde{h}_t \tag{7}$$

3.4. Identifying multiple classes of SATD

The classification process is as follows. After a comment to be classified is preprocessed, the GloVe model is used to vectorize it, and then the CGRU model is trained to extract local information features through CNN as the input of GRU to learn text words. Finally, the classification results are given. Five types of SATD are identified, namely, defect debt, test debt, design debt, requirement debt, and documentation debt, and cross-project identification is performed. For each identification of a certain debt, one project is taken as the test project in turn, and the other nine projects are taken as the training projects.

4. Experiment setup

4.1. Research questions

In this section, a series of experiments are carried out, and we aim to answer the following research questions:

RQ1: What is the effect of sampling rates when identifying SATD?

Different data sampling rates will have different effects. What sampling rate should be set in our experiment, this RQ aims to explore this issue.

RQ2: Performance Comparison with the state-of-art approach in identifying the five SATDs?

CNN has a significant effect on text classification, and researchers have taken CNN as the baseline in SATD detection. This RQ aims to investigate how much improvement is the proposed SCGRU method in identifying five SATDs compared to the state-of-the-art methods?

RQ3: How effective are sampling methods for identifying SATD?

Sampling methods are divided into oversampling and undersampling. By comparing a variety of oversampling methods and undersampling methods. This RQ is designed to verify the validity of the oversampling method with SeqGAN.

RQ4: How effective is SCGRU for identifying SATD?

The process of our SCGRU approach is divided into two parts. The first part is SeqGAN-based oversampling SATD data. The second part uses the sampled data to train a CGRU model for identifying SATD. In addition to CNN, we compare SCGRU with four other methods, namely GRU, CGRU, SeqGAN-based oversampling with CNN (S-CNN), and SeqGAN-based oversampling with GRU (S-GRU). Therefore, this RQ is designed to verify the effectiveness of SCGRU more comprehensively.

4.2. Dataset

The dataset is derived from the comments of 10 projects sorted by da S. Maldonado et al. (2017), including Ant, ArgoUML, Columba, EMF, Hibernate, JEdit, JFreeChart, JMeter, JRuby, and Squirrel. The data has been made public and available to researchers. Table 2 shows the amount of various technical debt and non-technical debt, and the imbalance rate of each type of technical debt in 10 projects. An extreme imbalance in the data is observed. With defect debt taken as an example, there are 472 defect debt in 10 projects, while the number of non-SATD is 58,204, which is about 123 times that of defect debt. Such extreme data imbalance will bring great difficulty to the classification. In addition, they divided the self-admitted technical debt into five categories, including design debt, defect debt, document debt, demand debt and test debt. Table 3 shows some examples of projects corresponding to the five SATD types.

4.3. Metrics

In this paper, we use four commonly used metrics namely Precision, Recall, F-Measure to measure the performance of the method.

It is the correct classification result when the predicted category matches the true category, such as true positive (*TP*) and true negative (*TN*). Similarly, it is a misclassification result when it does not match, such as false positive (*FP*) and false negative (*FN*). *TP* indicates that the predicted result belongs to SATD, and the real result also belongs to SATD. *TN* indicates that the predicted result does not belong to SATD, and the final real result does not belong to SATD. *FP* indicates that the predicted result does not belong to SATD, but the final real result belongs to SATD.

Table 3
Some examples of five types of self-admitted technical debt.

Types of SATD	Comments

Defect	// the generated classes must not be added in the generic JAR!
	// is that buggy on old JOnAS(2.4)??
	// TODO is not including line comments at the end of annotations a bug?
	——from EMF

Test	// For some reason ListSet.equals() allocates a lot of memory, well // some memory at least. Lets try to avoid that when not needed by // invoking this test only when the two previous tests are not decisive. ——from ArgoUML
Documentation	// FIXME: Document difference between warn and warning (or rename one better) ——from JRuby

Design	// XXX //All this to get package according to weblogic standards
	// Can be written better…this is too hack!
	// Careful…similar code in scanDir, but slightly different!! ——from Ant

Requirement	// TODO is it possible to use more than one variable? ——from ArgoUML

FN indicates that the predicted result belongs to SATD, but the actual result does not belong to SATD. Different classifier has different experimental results.

The precision indicates the proportion of samples whose true results are also positive among the samples predicted by the model to be positive, as shown in Eq. (8).

$$Precision = \frac{TP}{TP + FP} \tag{8}$$

The recall indicates the ratio of samples with positive prediction results of the model to samples with positive prediction results, as shown in Eq. (9).

$$Recall = \frac{TP}{TP + FN} \tag{9}$$

F-Measure is introduced supplementally for the reason that recall and precision cannot fully evaluate performance. As shown in Eq. (10), it is that the harmonic mean of recall and precision tends to be close to a smaller value, so a high F-Measure value could show that both precision and recall are relatively high.

$$F\text{-}Measure = \frac{2 * Precision * Recall}{Precision + Recall} \tag{10}$$

4.4. Experimental setting

The hyperparameter settings of our model are shown in Table 4. The key parameters (Zhang and Wallace, 2017) that affect the deep learning model include the dimension of word embedding, the number of filters, and filtering windows. A word embedding dimension of 300 is used to better express the comment text. GloVe is used for pre-training word embedding and vectorization so that semantic and grammatical information could be contained among vectors as much as possible. In order to find a suitable hyperparameter for subsequent experiments. We randomly sampled 10% of the 62275 annotations as the training set, which is 6227 annotations. 10% was chosen because it is approximately equal to the average number of comments for 10 projects, it can be seen from Table 2 below. In addition, we also selected the remaining 1% as the test set, i.e., 622 annotations, to evaluate models trained with different sampling rates. After our pre-experiment,[6] we set the number of filters is set to 128 and the window size is (3, 4, 5). In addition, to unify the SATD text, the comments are truncated and supplemented to a fixed length of 128.

[6] This process can be found in supplementary materials at https://github.com/beyondnpu/Multi-class_SATD_Identification.

Table 4
The hyper-parameter settings.

Word embedding dimension	300
The number of filters	128
The window size of filters	(3, 4, 5)
Length of self-admitted technical debt comments	128
Dropout	0.5

Table 5
Statistics of the comments in the 10 projects.

Projects	#Comment	#SATD	% of SATD
Ant	4,098	131	0.60
ArgoUML	9,452	1,413	2.08
Columba	6,478	204	0.60
EMF	4,390	104	0.41
Hibernate	2,968	472	4.05
JEdit	10,322	256	1.50
JFreeChart	4,408	209	0.89
JMeter	8,057	374	1.86
JRuby	4,897	622	5.57
SQuirrel	7,215	286	1.04
Avg.	6,228	407	0.65
Total	62,275	4,071	6.54

5. Results analysis

5.1. RQ1: What is the effect of sampling rates when identifying SATD?

Sampling rate (SR) may affect the results of the experiment. In order to solve this question, we introduced the Sampling Rate to illustrate the impact of sampling on the experiment. Please note that SR refers to the ratio of the number of samples in minority categories to that in most categories in the training set. When they are equal, the value is 1 and the data balance is reached. Therefore, to analyze the influence of sampling rate on the experiment, we performed a comparison of different sampling rate from 0.1 to 1.0.

In order to find a suitable sampling rate for subsequent experiments. We randomly sampled 10% of the 62,275 annotations as the training set, which is 6227 annotations. 10% was chosen because it is approximately equal to the average number of comments for 10 projects, it can be seen from Table 5. In addition, we also selected the remaining 1% as the test set, i.e., 622 annotations, to evaluate models trained with different sampling rates.

Fig. 3 shows the F-measure for five technical debt at different sampling rates. We vary the SR between 0.1 and 1, increasing by 0.1 each time, and here the F1 score is the average of the 10-item predictions. The experiment found that the overall change

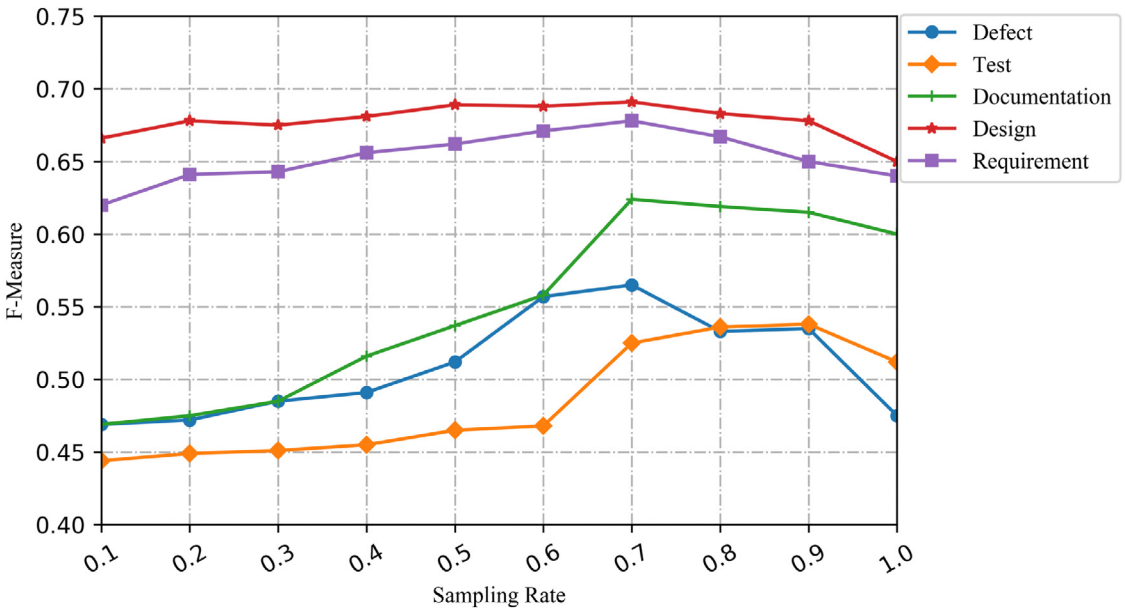


Fig. 3. Classification performance at different sampling rates.

trend of the five types of technical debt performance measures is relatively consistent. At the beginning, due to the small value of SR and the insufficient number of minority class sample expansion, the performance index did not improve significantly, but when the SR was set between 0.6 and 0.8, it was found that the performance improved rapidly, but after continuing to increase the sampling rate, for example, when the sampling rate is adjusted to more than 0.9, it is found that the performance is greatly reduced compared to the former. According to the results of this experiment, it is verified that the sampling rate cannot be too large. For example, setting SR to 1 is not the optimal parameter of the model with a high probability, which may be related to the generation of too many redundant samples.

Summary for RQ1:
According to our research, setting the sampling rate to be too high or too low is not optimal, and it is recommended to be between 0.6–0.8. So we used the sampling rate of 0.7 for all subsequent experiments involving sampling.

5.2. RQ2: Performance comparison with the state-of-art approach in identifying the five SATDs?

CNN has a significant effect on text classification, and researchers have taken CNN as the baseline in SATD detection. To show the effect of our approach, it is compared with CNN in identifying multiple classes of SATD. Table 6 shows the precision of our approach compared with CNN on 10 projects. It is worth noting that '-' represents non-existence. For example, the precision of documentation debt in the test project Ant is marked as '-', indicating that no such technical debt exists, so we cannot calculate the evaluation indicators of this technical debt. For defect debt, our approach achieves the best precision (1.0) on EMF, while CNN achieves the best precision (0.878) on Hibernate. On average, the precision of our approach and CNN are 0.692 and 0.563, respectively, which represents an improvement of 22.91%. For test debt, CNN has a precision of 0 on multiple projects, while our approach can successfully identify this kind of debt except on JFreeChart and SQuirrel, and the precision on multiple projects reached the highest value of 1.0. In addition, our approach has an average precision of 0.648 on nine projects for test debt, which is an increase of 2492% compared with CNN. For documentation debt, the average precision of our approach and CNN on 10 projects is 0.25 and 0.588, respectively, and our approach reflects an improvement of 135.2%. The precision of our approach increased by 4.42% and 5.19% for the design debt and requirement debt, respectively, which has relative data balance with a small increase. Therefore, our approach is more suitable for identifying SATD on projects with extremely imbalanced data.

Table 7 shows the recall of our approach compared with that of CNN. For defect debt, the maximum recall of CNN is 0.638 on ArgoUML, and the minimum is 0.056 on JRuby. The maximum recall of our approach is 0.692 on Columba, and the minimum is 0.163 on JEdit. The average recall of our approach and CNN on 10 projects is 0.348 and 0.508, respectively, and our approach improves by 45.97%. Similarly, for test debt, the recall of CNN on multiple projects is 0, while the average recall of our approach on nine projects is 0.454, which is 1127% higher than that of CNN. In addition, our approach improves by 3.54% and 2.74% for design debt and requirement debt, respectively.

Table 8 shows the F-measure of the two methods on 10 projects. Our approach improved significantly in identifying five types of technical debt. Specifically, compared with CNN, our approach achieved an average F-measure increase on 10 projects increase by 40.66%, 1650%, 250.56%, 250.56%, 3.44%, and 5.32% for defect debt, test debt, documentation debt, design debt, and requirement debt, respectively. For defect debt, the maximum F-measure of our approach is 0.753 on Hibernate, which is a 10.41% increase compared with 0.682 of CNN. The minimum F-measure of our approach for defect debt is 0.255 on JEdit, while the minimum of CNN is 0.102 on JRuby. CNN has difficulty identifying test debt, identifying it successfully on only two projects, while our approach could identify it on seven out of nine projects and the average F-measure reaches 0.525, which is significantly higher than that of CNN. Our approach could identify documentation debt as well, and its F-measure reaches 100% on two projects. For design debt and requirement debt, the F-measure of our approach improved to some extent. The average F-measure on 10 projects reaches 0.691 and 0.693, respectively, which are higher than CNN's 0.668 and 0.658.

Table 6
Precision of our approach and CNN for five technical debt classifications.

Projects	Precision									
	Defect		Test		Documentation		Design		Requirement	
	CNN	SCGRU	CNN	SCGRU	CNN	SCGRU	CNN	SCGRU	CNN	SCGRU
Ant	0.571	0.429	0	0.286	–	–	0.542	0.588	0.5	0.615
ArgoUML	0.42	0.457	0	0.882	0	0.526	0.8	0.771	0.657	0.657
Columba	0.7	0.75	0	1.0	0	0	0.793	0.77	0.833	0.833
EMF	0.4	**1.0**	0	1.0	–	–	0.81	0.822	**1.0**	**1.0**
Hibernate	**0.878**	0.854	–	–	0	0	0.826	0.848	0.912	0.862
JEdit	0.571	0.583	0	1.0	–	–	0.643	0.707	0.444	0.417
JFreeChart	0.375	0.75	0	0	–	–	0.66	0.775	0.9	0.9
JMeter	0.611	0.667	0	1.0	0.5	1.0	**0.931**	**0.865**	0.314	0.571
JRuby	0.6	0.904	**0.222**	0.667	0	1.0	0.706	0.802	0.747	0.769
SQuirrel	0.5	0.529	0	0	1.0	1.0	0.762	0.851	0.824	0.872
Avg.	0.563	**0.692**	0.025	**0.648**	0.25	**0.588**	0.747	**0.78**	0.713	**0.75**
Imp.	**22.91%**		**2492%**		**135.2%**		**4.42%**		**5.19%**	

Table 7
Recall of our approach and CNN for five technical debt classifications.

Projects	Recall									
	Defect		Test		Documentation		Design		Requirement	
	CNN	SCGRU	CNN	SCGRU	CNN	SCGRU	CNN	SCGRU	CNN	SCGRU
Ant	0.308	0.462	0	0.4	–	–	0.611	0.632	0.462	0.615
ArgoUML	**0.638**	0.669	0	0.682	0	0.333	**0.874**	**0.936**	0.822	0.905
Columba	0.538	**0.692**	0	0.667	0	0	0.762	0.825	**0.930**	**0.930**
EMF	0.25	0.25	0	0.5	–	–	0.436	0.474	0.438	0.438
Hibernate	0.558	0.673	–	–	0	0	0.792	0.789	0.813	0.875
JEdit	0.093	0.163	0	0.667	–	–	0.505	0.531	0.286	0.375
JFreeChart	0.333	0.667	0	0	–	–	0.380	0.337	0.6	0.6
JMeter	0.5	0.545	0	**0.833**	0.333	**1.0**	0.687	0.75	0.762	0.571
JRuby	0.056	0.584	**0.333**	0.333	0	0.5	0.609	0.615	0.618	0.636
SQuirrel	0.208	0.375	0	0	**0.5**	0.5	0.550	0.545	0.840	0.820
Avg.	0.348	**0.508**	0.037	**0.454**	0.139	**0.389**	0.621	**0.643**	0.657	**0.675**
Imp.	**45.97%**		**1127%**		**179.86%**		**3.54%**		**2.74%**	

Table 8
F-Measure of our approach and CNN for five technical debt classifications.

Projects	F-Measure									
	Defect		Test		Documentation		Design		Requirement	
	CNN	SCGRU	CNN	SCGRU	CNN	SCGRU	CNN	SCGRU	CNN	SCGRU
Ant	0.4	0.444	0	0.333	–	–	0.574	0.609	0.48	0.615
ArgoUML	0.506	0.543	0	0.769	0	0.408	**0.835**	**0.846**	0.731	0.762
Columba	0.609	0.72	0.002	0.8	0	0	0.777	0.797	**0.879**	**0.879**
EMF	0.308	0.4	0	0.667	–	–	0.567	0.602	0.609	0.609
Hibernate	**0.682**	**0.753**	–	–	0	**1.0**	0.809	0.818	0.86	0.868
JEdit	0.160	0.255	0	0.8	–	–	0.566	0.606	0.348	0.385
JFreeChart	0.353	0.706	0	0	–	–	0.465	0.47	0.72	0.72
JMeter	0.55	0.6	0	**0.909**	0.4	1.0	0.791	0.803	0.444	0.571
JRuby	0.102	0.709	**0.267**	0.444	0	0.667	0.654	0.696	0.677	0.697
SQuirrel	0.294	0.439	0	0	**0.667**	0.667	0.639	0.665	0.832	0.845
Avg.	0.396	**0.557**	0.03	**0.525**	0.178	**0.624**	0.668	**0.691**	0.658	**0.693**
Imp.	**40.66%**		**1650%**		**250.56%**		**3.44%**		**5.32%**	

Summary for RQ2:
The CNN method is slightly lower than the SCGRU method in identifying a relatively large number of design debt and requirement debt. However, for a small number of defective debt, it is difficult to function without the CNN method.

5.3. RQ3: How effective are sampling methods for identifying SATD?

Sampling methods are divided into oversampling and undersampling. Oversampling achieves balance by increasing the number of minority samples (Rivera and Xanthopoulos, 2016), while undersampling achieves balance by reducing the number of majority samples (Akkasi et al., 2018). Undersampling inevitably loses some features, especially when these features are critical to classification. Therefore, in this paper, we focus on the effectiveness of the oversampling method for identifying SATD. To verify the validity of the oversampling method with SeqGAN, we compare the results of five technical debt with random oversampling (ROS), SMOTE, and ADASYN (He et al., 2008) on 10 projects. We use the text oversampling method with SeqGAN. We compared it with other methods such as ROS, SMOTE, and ADASYN. CGRU models are trained separately with the sampled data obtained by different methods. Four methods are applied: SCGRU, ROS-CGRU, SMOTE-CGRU, and ADASYN-CGRU. Table 9 shows some examples of the five SATD types generated using the SeqGAN-based text generation method.

Table 9
Some examples of the five SATD types generated using the SeqGAN-based text generation method.

Types of SATD	Comments

Defect	// todo this does n't work but should collection elements model.
	// todo this is an freopen in mri this is close but not quite the same plugins.
	// is this an error defined null put to look.

Test	// test the constructor used for loading a profile from a jar file.
	// throws exception if something goes wrong.
	// tests if the list with namespaces defined in getvalidnamespaceclasses contains only valid namespaces.

Documentation	// fixme this function needs documentation return sstring name umlfactory file.

Design	// this should probably be an exception since it would not change current 3 arraycopy.
	// todo this should not be moved out of the implementation class.
	// we ca n't use systemclassloader here but i think this should be done in done.

Requirement	// todo not implemented obvious transaction using clause generate false.

Figs. 4(a), 4(b), 4(c) and respectively show the F-measures of SCGRU, ROS-CGRU, SMOTE-CGRU, and ADASYN-CGRU for identifying defect debt, test debt, and document debt. SCGRU achieves a significant improvement for three technical debt and outperforms the other three methods on 10 projects. Specifically, as shown in Fig. 4(a), the F-measure of our approach is above 0.5 on multiple projects, with the highest F-measure reaching 0.753. The average F-measure on 10 projects is 0.557, which is significantly higher than 0.453 of ROS-CGRU, 0.230 of SMOTE-CGRU, and 0.222 of ADASYN-CGRU. For test debt, the improvement of our approach is obvious, as shown in Fig. 4(b). The F-measure of our approach on multiple projects is significantly higher than that of other methods. The highest F-measure is 0.909 on JMeter, the average F-measure on nine projects is 0.525, and the average F-measure of the other three methods is 0.401, 0.161, and 0.146, respectively. We found that none of the four methods could identify test debt on SQuirrel, a project with a very small amount of test debt. Fig. 4(c) shows the F-measure of four methods for documentation debt on six projects. Our approach achieved the highest value of 1 on two projects, while most of the other methods were below 0.5, which indicates that they could not effectively identify documentation debt. Fig. 4(d) shows the results of the four methods for design debt. The average F-measure of our approach on 10 projects is 0.691, while the average F-measure of the other three methods is 0.666, 0.415, and 0.42 respectively, indicating that our method achieved increases of 3.75%, 66.51%, and 64.52%. Similarly, as shown in Fig. 4(e), the F-measure of our proposed SCGRU approach is slightly higher than that of ROS-CGRU and significantly better than that of SMOTE-CGRU and ADASYN-CGRU for requirement debt. The highest F-measure is 0.879 on Columba, and the average F-measure on 10 projects is 0.693, indicating a 2.51% increase compared with 0.676 of ROS-CGRU. Moreover, it is significantly higher than 0.384 of SMOTE-CGRU and 0.287 of ADASYN-CGRU, reflecting an increase of 80.47% and 141.46%, respectively.

Table 10 shows the average F-measure on 10 projects for five technical debt identified by four sampling methods using the CGRU model. The average F-measure of our approach in identifying defect debt, test debt, documentation debt, design debt, and requirement debt are 0.557, 0.525, 0.624, 0.691, and 0.693, respectively, which are 22.96%, 30.92%, 93.79%, 3.75%, and 2.51% higher than that of ROS-CGRU, respectively. This result shows that our approach improves significantly on the imbalanced types of defect debt, test debt, and documentation debt, and indicates the effectiveness of the oversampling method.

Table 11 shows the p-value of our approach compared with that of other methods. P-value[7] is used to illustrate the significant differences between methods. Results show that our approach exhibits a significant improvement for defect debt, test debt, documentation debt, design debt, and requirement debt compared with SMOTE-CGRU and ADASYN-CGRU.

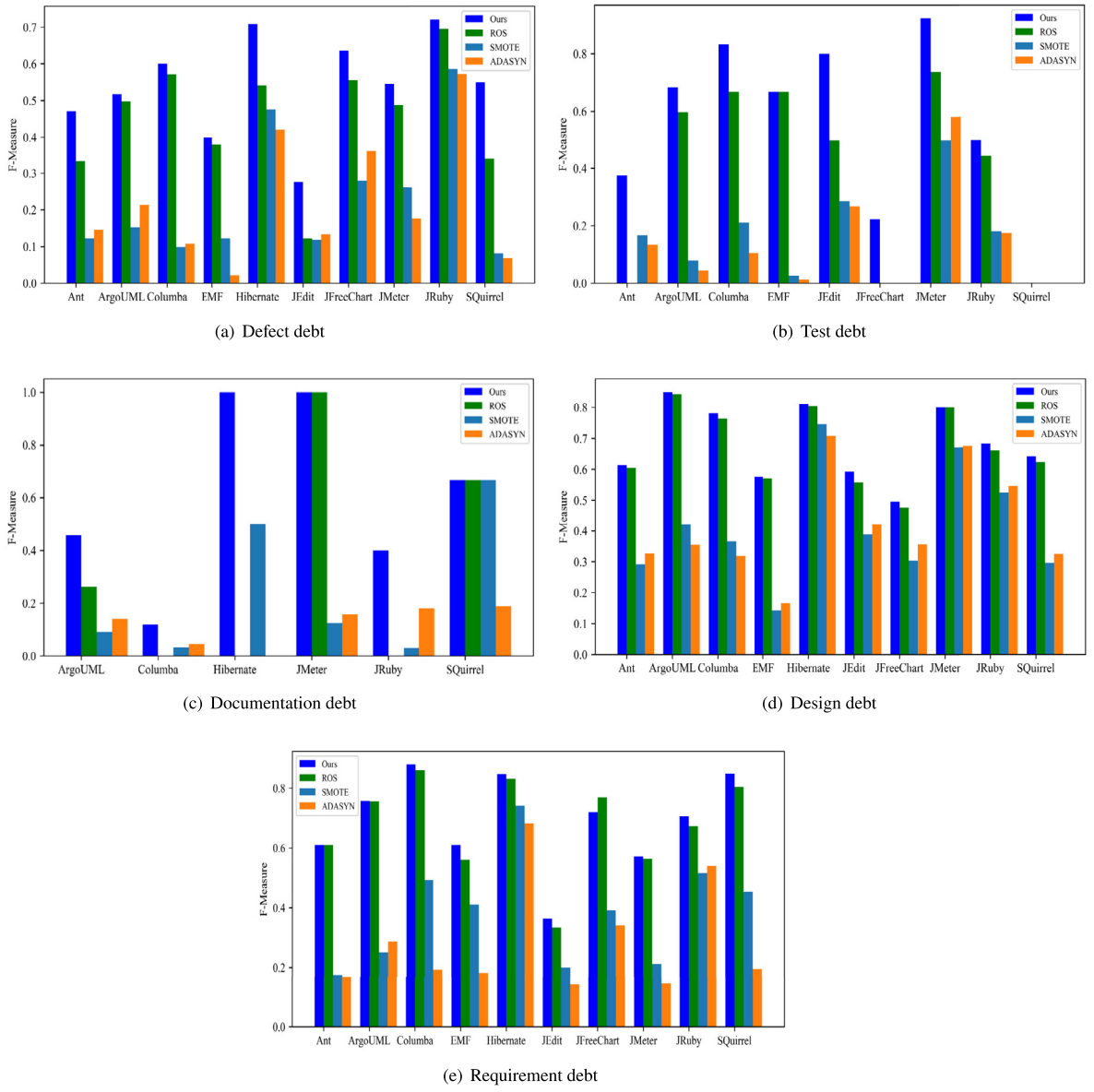
Summary for RQ3:
SeqGAN-based text generation oversampling method can improve the effectiveness of SATD identification, especially in defect debt, document debt and test debt. The under-sampling method failed to improve the performance of recognition, which may be related to the missing information.

5.4. RQ4: How effective is SCGRU for identifying SATD?

The process of our SCGRU approach is divided into two parts. The first part is SeqGAN-based oversampling SATD data. The second part uses the sampled data to train a CGRU model for identifying SATD. Then, to verify the effectiveness of SCGRU, we compare SCGRU with five other methods, namely, CNN, GRU, CGRU, SeqGAN-based oversampling with CNN (S-CNN), and SeqGAN-based oversampling with GRU (S-GRU).

Fig. 5 shows the F-measure of SCGRU, S-GRU, S-CNN, CGRU, CNN, and GRU for five types of technical debt, namely, defect, test, documentation, design, and requirement debt. Fig. 5(a) shows the F-measure of six methods for identifying defect debt. The F-measure of our SCGRU approach is better than that of the other five methods on 10 projects and is significantly improved on Columba. Compared with S-GRU, S-CNN, CGRU, CNN, and GRU, the improvement is 26.09%, 39.53%, 20%, 18.23%, and 13.21%, respectively, and the highest F-measure on 10 projects is 0.753 on Hibernate. Fig. 5(b) shows the F-measure of various methods for identifying test debt; our approach outperforms the other methods on multiple projects. We can see that CNN has difficulty identifying test debt. Most importantly, due to the lack of training data, the six methods all fail to perform cross-project identification on JFreeChart and Squirrel. However, our approach can identify it successfully. The F-measures of various methods for identifying documentation debt are shown in Fig. 5(c). Only six projects contain documentation debt; thus, their F-measure for cross-project prediction is shown in Fig. 5(c). For documentation debt, the F-measure of our approach is significantly better than that of other methods on ArgoUML and Hibernate, and the results on JMeter, JRuby, and Squirrel are not much different from those of GRU and CGRU. Figs. 5(d) and 5(e) respectively show a variety of methods for identifying design debt and requirement debt. Enough training data are available for these two types of debt; thus, the classifier can learn more features. Therefore, little difference in F-measures exists between these methods. In general,

[7] Specifically, we use Wilcoxon signed-rank test show that the improvements in performance are all statistically significant at the p-value < 0.05.



(a) Defect debt

(b) Test debt

(c) Documentation debt

(d) Design debt

(e) Requirement debt

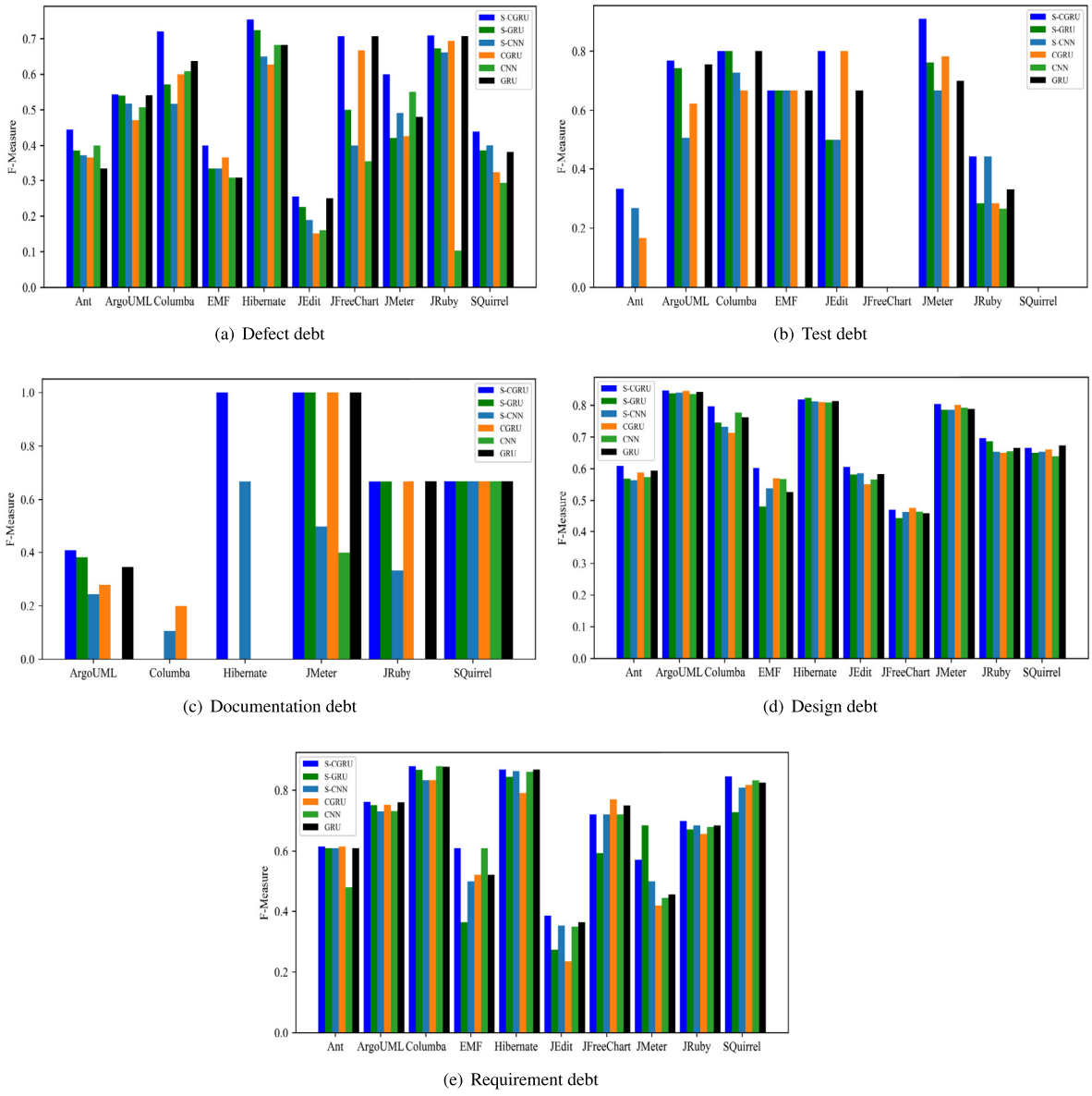
Fig. 4. F-measure of four methods for five technical debt.

Table 10
Average F-Measure of four methods for identifying five technical debt.

	Classification	Methods			
		SCGRU	ROS-CGRU	SMOTE-CGRU	ADASYN-CGRU
F-Measure	Defect	**0.557**	0.453	0.23	0.222
	Test	**0.525**	0.401	0.161	0.146
	Documentation	**0.624**	0.322	0.241	0.119
	Design	**0.691**	0.666	0.415	0.42
	Requirement	**0.693**	0.676	0.384	0.287

the average F-measure of our approach is 0.693, which is slightly better than 0.686 of GRU. This finding shows that our approach is suitable for extremely imbalanced data and achieves only a small improvement when the data are relatively balanced.

Table 12 shows the average F-measure of six methods for identifying defect debt, test debt, documentation debt, design debt, and requirement debt on different projects. Only six projects have documentation debt, as shown in Table 2, which is why we only calculate the average F-measure on six projects for documentation debt. Similarly, the F-measure for test debt is average on nine projects. Overall, our approach ensures significant improvements for defect debt, test debt, and documentation debt, as well as small improvement for design debt and requirement debt. Specifically, compared with S-GRU, S-CNN, CGRU, CNN, and GRU, our approach achieves a significant improvement for defect debt and the average F-measure increased by 12.53%, 18.01%, 19.02%, 40.66%, and 16.04%, respectively. For test debt, the average F-measure of our approach reached 0.525, which is an increase of 15.13% compared with GRU. For documentation debt, the average F-measure of our approach is 0.624, which is 1.63% higher than 0.614 of S-GRU. As for the average F-measure of design debt,



(a) Defect debt

(b) Test debt

(c) Documentation debt

(d) Design debt

(e) Requirement debt

Fig. 5. F-measure of six methods for identifying five technical debt.

Table 11
P-value of three methods for identifying five technical debt.

	Classification	Methods		
		ROS-CGRU	SMOTE-CGRU	ADASYN-CGRU
p-value	Defect	0.008	0.005	0.005
	Test	0.042	0.018	0.018
	Documentation	0.05	0.05	0.046
	Design	0.017	0.005	0.005
	Requirement	0.059	0.005	0.005

our approach increased by 4.54%, 5.34%, 5.18%, 3.44%, and 0.73% compared with S-GRU, S-CNN, CGRU, CNN, and GRU, respectively. For requirement debt, the increase is 1.46% to 5.80% compared with the other five methods. Table 13 shows the *p*-value of our approach compared with the other five methods of five technical debt. The method significantly improved for defect debt, test debt, and design debt.

Summary for RQ4:
When identifying SATD, the effectiveness of the method is not only related to the oversampling method used, but also affected by the model, and the former has a greater impact. Therefore, it is worth paying attention to the data level to solve the problem of data imbalance faced when identifying SATD.

6. Threats to validity

Construct validity may be derived from evaluation indexes. The evaluation indexes we use are precision, recall, and F-measure. These indicators are widely used to evaluate the

Table 12
Average F-Measure of six methods for five types of technical debt.

	Classification	Methods					
		SCGRU	S-GRU	S-CNN	CGRU	CNN	GRU
F-Measure	Defect	**0.557**	0.495	0.472	0.468	0.396	0.48
	Test	**0.525**	0.408	0.4	0.424	0.03	0.456
	Documentation	**0.624**	0.614	0.434	0.322	0.178	0.445
	Design	**0.691**	0.661	0.656	0.657	0.668	0.686
	Requirement	**0.693**	0.664	0.655	0.656	0.658	0.683

Table 13
P-value of six methods for five types of technical debt.

	Classification	Methods				
		S-GRU	S-CNN	CGRU	CNN	GRU
p-value	Defect	0.005	0.005	0.005	0.005	0.008
	Test	0.043	0.043	0.043	0.018	0.043
	Documentation	0.18	0.041	0.035	0.068	0.005
	Design	0.007	0.005	0.017	0.005	0.013
	Requirement	0.037	0.008	0.038	0.018	0.066

performance of classifiers and proved to be effective. In addition, *p*-value, which is widely used to verify the significance of methods, is introduced to prove the significant difference of methods.

Internal validity may be related to errors in the implementation process, such as hyperparameter settings and the authority of the dataset. We checked the implementation process several times. The hyperparameters are determined according to certain key parameters that affect the deep learning model provided by researchers. Different data may lead to differences in results and cause internal threats. For this reason, authoritative and widely used datasets of SATD provided by Maldonado are used.

External validity include the quality and quantity of our experimental datasets, as well as the factors that restrict us from extending the experimental results to other research populations or research environments. In order to ensure the quality and quantity of the dataset, we used open source projects from 10 different fields, different number of comments and characteristics, with a total of 62,275 comments. In addition, since the datasets we use are all open source projects, they may not be able to be extended to non-open source commercial projects. And because of the high transparency of the open source community, developers will communicate with each other in the form of annotations, so SATD will be introduced. However, in commercial projects, developers may not be inclined to introduce SATD for some reasons (evaluation of the quality of their work, etc.)

Across-project prediction is conducted to ensure conclusion validity. Each time, we turn one of 10 projects into a test set and the other nine into a training set, which can reduce the illusion that an approach is suitable for a certain project. These projects all have certain differences that can help better illustrate the effectiveness of the approach.

7. Conclusion and future work

In this paper, we propose an approach called SCGRU to realize the identification of multiple classes of SATD. We dealt with extreme data imbalance by using SeqGAN-based text generation oversampling and identified multiple classes of SATD by using the CGRU model. Five types of technical debt were identified, namely, defect debt, test debt, documentation debt, design debt, and requirement debt. Cross-project experiments show that our approach is significantly better than existing approaches, especially when the data are extremely imbalanced.

Our proposed SCGRU approach provides a new idea for the practice of SATD identification. When conducting SATD predictions on software projects, we effectively addressed some types of problems that SATD cannot identify because of extreme data imbalance or lack of sufficient data. Moreover, the proposed approach successfully identified the multiple classes of SATD and helped achieve accurate positioning of debt. In addition, identifying multiple classes of SATD could help with other studies on SATD, such as removal and management of SATD.

If the amount of a certain technical debt is extremely lacking on projects, such as fewer than five training samples, even if we perform text generation sampling on some projects, then the generated samples may be insufficient for a classifier to learn features due to a low amount of technical debt data. In the future, we hope to acquire more data as a supplement for learning and generating diverse and valuable samples of technical debt and provide more possibilities to verify the effectiveness of our approach.

CRediT authorship contribution statement

Kuiyu Zhu: Methodology, Software, Writing – original draft, Writing – review & editing. **Ming Yin:** Conceptualization, Supervision, Writing – review & editing. **Dan Zhu:** Data curation, Investigation, Validation Writing – review & editing. **Xiaogang Zhang:** Methodology, Investigation, Visualization. **Cunzhi Gao:** Methodology, Investigation, Validation. **Jijiao Jiang:** Conceptualization, Writing – review & editing.

Acknowledgments

The work reported herein was supported by Shaanxi Provincial Social Science Fund Research Project (2018S28). The authors thank all the participants in the experiment and the professors for their advice on feather functions and inference mechanisms. The authors would like to thank the anonymous reviewers for their valuable suggestion and constructive comments.

References

Adem, K., 2022. Impact of activation functions and number of layers on detection of exudates using circular hough transform and convolutional neural networks. Expert Systems with Applications 203, 117583. http://dx.doi.org/10.1016/j.eswa.2022.117583.

Akkasi, A., Varoglu, E., Dimililer, N., 2018. Balanced undersampling: A novel sentence-based undersampling method to improve recognition of named entities in chemical and biomedical text. Appl. Intell. 48, 1965–1978. http://dx.doi.org/10.1007/s10489-017-0920-5.

Arora, S., May, A., Zhang, J., Ré, C., 2020. Contextual embeddings: when are they worth it?. In: Jurafsky, D., Chai, J., Schluter, N., Tetreault, J.R. (Eds.), Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020. Association for Computational Linguistics, pp. 2650–2663. http://dx.doi.org/10.18653/v1/2020.acl-main.236.

Batista, G.E.A.P.A., Prati, R.C., Monard, M.C., 2004. A study of the behavior of several methods for balancing machine learning training data. SIGKDD Explor. 6, 20–29. http://dx.doi.org/10.1145/1007730.1007735.

Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P., 2002. SMOTE: Synthetic minority over-sampling technique. J. Artificial Intelligence Res. 16, 321–357. http://dx.doi.org/10.1613/jair.953.

Cho, K., van Merrienboer, B., Gülçehre, Ç., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y., 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In: Moschitti, A., Pang, B., Daelemans, W. (Eds.), Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25–29, 2014, Doha, Qatar, a Meeting of SIGDAT, a Special Interest Group of the ACL. ACL, pp. 1724–1734. http://dx.doi.org/10.3115/v1/d14-1179.

Cunha, W., Mangaravite, V., Gomes, C., Canuto, S., Resende, E., Nascimento, C., Viegas, F., França, C., Martins, W.S., Almeida, J.M., Rosa, T., Rocha, L., Gonçalves, M.A., 2021. On the cost-effectiveness of neural and non-neural approaches and representations for text classification: A comprehensive comparative study. Inf. Process. Manage. 58, 102481. http://dx.doi.org/10.1016/j.ipm.2020.102481, URL: https://www.sciencedirect.com/science/article/pii/S0306457320309705.

Cunningham, W., 1993. The WyCash portfolio management system. OOPS Messenger 4, 29–30. http://dx.doi.org/10.1145/157710.157715.

Douzas, G., Bação, F., 2018. Effective data generation for imbalanced learning using conditional generative adversarial networks. Expert Syst. Appl. 91, 464–471. http://dx.doi.org/10.1016/j.eswa.2017.09.030.

Dubey, A.K., Jain, V., 2019. Comparative study of convolution neural network's Relu and Leaky-Relu activation functions. In: Mishra, S., Sood, Y.R., Tomar, A. (Eds.), Applications of Computing, Automation and Wireless Systems in Electrical Engineering. Springer, Singapore, pp. 873–880.

Flisar, J., Podgorelec, V., 2019. Identification of self-admitted technical debt using enhanced feature selection based on word embedding. IEEE Access 7, 106475–106494. http://dx.doi.org/10.1109/ACCESS.2019.2933318.

Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A.C., Bengio, Y., 2014. Generative adversarial nets. In: Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D., Weinberger, K.Q. (Eds.), Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8–13 2014, Montreal, Quebec, Canada. pp. 2672–2680.

Ha, J., Lee, J., 2016. A new under-sampling method using genetic algorithm for imbalanced data classification. In: Proceedings of the 10th International Conference on Ubiquitous Information Management and Communication, IMCOM 2016, Danang, Vietnam, January 4–6, 2016. ACM, pp. 95:1–95:6. http://dx.doi.org/10.1145/2857546.2857643.

He, H., Bai, Y., Garcia, E.A., Li, S., 2008. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In: Proceedings of the International Joint Conference on Neural Networks, IJCNN 2008, Part of the IEEE World Congress on Computational Intelligence, WCCI 2008, Hong Kong, China, June 1–6, 2008. IEEE, pp. 1322–1328. http://dx.doi.org/10.1109/IJCNN.2008.4633969.

Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural Comput. 9, 1735–1780. http://dx.doi.org/10.1162/neco.1997.9.8.1735.

Huang, Q., Shihab, E., Xia, X., Lo, D., Li, S., 2018. Identifying self-admitted technical debt in open source projects using text mining. Empir. Softw. Eng. 23, 418–451. http://dx.doi.org/10.1007/s10664-017-9522-4.

Huszar, F., 2015. How (not) to train your generative model: Scheduled sampling, likelihood, adversary? CoRR, arXiv:1511.05101.

Ide, H., Kurita, T., 2017. Improvement of learning for CNN with ReLU activation by sparse regularization. In: 2017 International Joint Conference on Neural Networks, IJCNN 2017, Anchorage, AK, USA, May 14–19, 2017. IEEE, pp. 2684–2691. http://dx.doi.org/10.1109/IJCNN.2017.7966185.

Kalchbrenner, N., Grefenstette, E., Blunsom, P., 2014. A convolutional neural network for modelling sentences. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22–27, 2014, Baltimore, MD, USA. In: Long Papers, vol. 1, The Association for Computer Linguistics, pp. 655–665. http://dx.doi.org/10.3115/v1/p14-1062.

Kim, Y., 2014. Convolutional neural networks for sentence classification. In: Moschitti, A., Pang, B., Daelemans, W. (Eds.), Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25–29, 2014, Doha, Qatar, a Meeting of SIGDAT, a Special Interest Group of the ACL. ACL, pp. 1746–1751. http://dx.doi.org/10.3115/v1/d14-1181.

Last, F., Douzas, G., Bação, F., 2017. Oversampling for imbalanced learning based on K-means and SMOTE. CoRR, arXiv:1711.00837.

Lázaro, M., Herrera, F., Figueiras-Vidal, A.R., 2015. Classification of binary imbalanced data using a Bayesian ensemble of Bayesian neural networks. In: Iliadis, L.S., Jayne, C. (Eds.), Engineering Applications of Neural Networks - 16th International Conference, EANN 2015, Rhodes, Greece, September 25–28, 2015, Proceedings. In: Communications in Computer and Information Science, vol. 517, Springer, pp. 304–314. http://dx.doi.org/10.1007/978-3-319-23983-5_28.

Lee, S., Hong, S., Yang, S., 2018. Oversampling for imbalanced data classification using adversarial network. In: International Conference on Information and Communication Technology Convergence, ICTC 2018, Jeju Island, Korea, South, October 17–19, 2018. IEEE, pp. 1255–1257. http://dx.doi.org/10.1109/ICTC.2018.8539543.

Li, Y., Guo, H., Zhang, Q., Mingyun, G., Yang, J., 2018a. Imbalanced text sentiment classification using universal and domain-specific knowledge. Knowl.-Based Syst. 160, 1–15. http://dx.doi.org/10.1016/j.knosys.2018.06.019.

Li, P., Li, X., Pan, H., Khyam, M.O., Noor-A-Rahim, M., 2020. Text-based indoor place recognition with deep neural network. Neurocomputing 390, 239–247. http://dx.doi.org/10.1016/j.neucom.2019.02.065, URL: https://www.sciencedirect.com/science/article/pii/S0925231219314419.

Li, Y., Pan, Q., Wang, S., Yang, T., Cambria, E., 2018b. A generative model for category text generation. Inform. Sci. 450, 301–315. http://dx.doi.org/10.1016/j.ins.2018.03.050.

Liang, X., Xu, J., 2021. Biased relu neural networks. Neurocomputing 423, 71–79. http://dx.doi.org/10.1016/j.neucom.2020.09.050.

Luo, Y., Feng, H., Weng, X., Huang, K., Zheng, H., 2019. A novel oversampling method based on SeqGAN for imbalanced text classification. In: 2019 IEEE International Conference on Big Data, Big Data.

Lv, Y., Chen, Y., Li, L., Wang, F., 2018. Generative adversarial networks for parallel transportation systems. IEEE Intell. Transp. Syst. Mag. 10, 4–10. http://dx.doi.org/10.1109/MITS.2018.2842249.

Maguolo, G., Nanni, L., Ghidoni, S., 2021. Ensemble of convolutional neural networks trained with different activation functions. Expert Syst. Appl. 166, 114048. http://dx.doi.org/10.1016/J.ESWA.2020.114048.

Mahmoud, A., El-Kilany, A., Ali, F., Mazen, S., 2020. A novel oversampling technique to handle imbalanced datasets. In: Steglich, M., Mueller, C., Neumann, G., Walther, M. (Eds.), Proceedings of the 34th International ECMS Conference on Modelling and Simulation, ECMS 2020, Wildau, Germany, June 9–12, 2020. European Council for Modeling and Simulation, pp. 177–182. http://dx.doi.org/10.7148/2020-0177.

Maipradit, R., Lin, B., Nagy, C., Bavota, G., Lanza, M., Hata, H., Matsumoto, K., 2020a. Automated identification of on-hold self-admitted technical debt. In: 20th IEEE International Working Conference on Source Code Analysis and Manipulation, SCAM 2020, Adelaide, Australia, September 28 – October 2, 2020. IEEE, pp. 54–64. http://dx.doi.org/10.1109/SCAM51674.2020.00011.

Maipradit, R., Treude, C., Hata, H., Matsumoto, K., 2020b. Wait for it: Identifying "on-hold" self-admitted technical debt. Empir. Softw. Eng. 25, 3770–3798. http://dx.doi.org/10.1007/s10664-020-09854-3.

Moreo, A., Esuli, A., Sebastiani, F., 2016. Distributional random oversampling for imbalanced text classification. In: Perego, R., Sebastiani, F., Aslam, J.A., Ruthven, I., Zobel, J. (Eds.), Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2016, Pisa, Italy, July 17–21, 2016. ACM, pp. 805–808. http://dx.doi.org/10.1145/2911451.2914722.

Muse, B.A., Nagy, C., Cleve, A., Khomh, F., Antoniol, G., 2022. FIXME: Synchronize with database an empirical study of data access self-admitted technical debt. CoRR, arXiv:2201.02180.

Potdar, A., Shihab, E., 2014. An exploratory study on self-admitted technical debt. In: 30th IEEE International Conference on Software Maintenance and Evolution, Victoria, BC, Canada, September 29 – October 3, 2014. IEEE Computer Society, pp. 91–100. http://dx.doi.org/10.1109/ICSME.2014.31.

Ren, X., Xing, Z., Xia, X., Lo, D., Wang, X., Grundy, J., 2019. Neural network-based detection of self-admitted technical debt: From performance to explainability. ACM Trans. Softw. Eng. Methodol. 28, 15. http://dx.doi.org/10.1145/3324916.

Rivera, W.A., Xanthopoulos, P., 2016. A priori synthetic over-sampling methods for increasing classification sensitivity in imbalanced data sets. Expert Syst. Appl. 66, 124–135. http://dx.doi.org/10.1016/j.eswa.2016.09.010.

da S. Maldonado, E., Shihab, E., Tsantalis, N., 2017. Using natural language processing to automatically detect self-admitted technical debt. IEEE Trans. Softw. Eng. 43, 1044–1062. http://dx.doi.org/10.1109/TSE.2017.2654244.

Santos, R.M., Santos, I.M., Júnior, M.C.R., de Mendonça Neto, M.G., 2020. Long term-short memory neural networks and Word2vec for self-admitted technical debt detection. In: Filipe, J., Smialek, M., Brodsky, A., Hammoudi, S. (Eds.), Proceedings of the 22nd International Conference on Enterprise Information Systems, ICEIS 2020, Prague, Czech Republic, May 5–7, 2020, Volume 2. SCITEPRESS, pp. 157–165. http://dx.doi.org/10.5220/0009796001570165.

She, X., Zhang, D., 2018. Text classification based on hybrid CNN-LSTM hybrid model. In: 11th International Symposium on Computational Intelligence and Design, ISCID 2018, Hangzhou, China, December 8–9, 2018, Volume 2. IEEE, pp. 185–189. http://dx.doi.org/10.1109/ISCID.2018.10144.

Sierra, G., Shihab, E., Kamei, Y., 2019. A survey of self-admitted technical debt. J. Syst. Softw. 152, 70–82. http://dx.doi.org/10.1016/j.jss.2019.02.056.

Tanaka, M., 2020. Weighted sigmoid gate unit for an activation function of deep neural network. Pattern Recognit. Lett. 135, 354–359. http://dx.doi.org/10.1016/j.patrec.2020.05.017.

Wang, K., Gou, C., Duan, Y., Lin, Y., Zheng, X., Wang, F., 2017. Generative adversarial networks: Introduction and outlook. IEEE/CAA J. Autom. Sin. 4, 588–598. http://dx.doi.org/10.1109/JAS.2017.7510583.

Wang, K., Wan, X., 2018. SentiGAN: Generating sentimental texts via mixture adversarial networks. In: Lang, J. (Ed.), Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13–19, 2018, Stockholm, Sweden. ijcai.org, pp. 4446–4452. http://dx.doi.org/10.24963/ijcai.2018/618.

Wattanakriengkrai, S., Maipradit, R., Hata, H., Choetkiertikul, M., Sunetnanta, T., Matsumoto, K., 2018. Identifying design and requirement self-admitted technical debt using N-gram IDF. In: 9th International Workshop on Empirical Software Engineering in Practice, IWESEP 2018, Nara, Japan, December 4, 2018. IEEE, pp. 7–12. http://dx.doi.org/10.1109/IWESEP.2018.00010.

Xianjing, G., Yong, L., 2018. Text implication recognition learning with gated recurrent unit. In: Proceedings of the 2018 2nd International Conference on Computer Science and Artificial Intelligence, CSAI 2018, the 10th International Conference on Information and Multimedia Technology, ICIMT 2018, Shenzhen, China, December 08–10, 2018. ACM, pp. 282–285. http://dx.doi.org/10.1145/3297156.3297252.

Xiao, T., Wang, D., McIntosh, S., Hata, H., Kula, R.G., Ishio, T., Matsumoto, K., 2021. Characterizing and mitigating self-admitted build debt. CoRR, arXiv:2102.09775.

Xie, Y., Zhang, T., 2018. Imbalanced learning for fault diagnosis problem of rotating machinery based on generative adversarial networks. In: 2018 37th Chinese Control Conference. CCC, pp. 6017–6022. http://dx.doi.org/10.23919/ChiCC.2018.8483334.

Xu, J., Ren, X., Lin, J., Sun, X., 2018. Diversity-promoting GAN: A cross-entropy based generative adversarial network for diversified text generation. In: Riloff, E., Chiang, D., Hockenmaier, J., Tsujii, J. (Eds.), Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 – November 4, 2018. Association for Computational Linguistics, pp. 3940–3949. http://dx.doi.org/10.18653/v1/d18-1428.

Yap, B.W., Rani, K.A., Rahman, H.A.A., Fong, S., Khairudin, Z., Abdullah, N.N., 2013. An application of oversampling, undersampling, bagging and boosting in handling imbalanced datasets. In: Herawan, T., Deris, M.M., Abawajy, J.H. (Eds.), Proceedings of the First International Conference on Advanced Data and Information Engineering, DaEng 2013, Kuala Lumpur, Malaysia, December 16–18, 2013. In: Lecture Notes in Electrical Engineering, vol. 285, Springer, pp. 13–22. http://dx.doi.org/10.1007/978-981-4585-18-7_2.

Yilmaz, S., Toklu, S., 2020. A deep learning analysis on question classification task using Word2vec representations. Neural Comput. Appl. 32, 2909–2928. http://dx.doi.org/10.1007/s00521-020-04725-w.

Yu, L., Zhang, W., Wang, J., Yu, Y., 2017. SeqGAN: Sequence generative adversarial nets with policy gradient. In: Singh, S.P., Markovitch, S. (Eds.), Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4–9, 2017, San Francisco, California, USA. AAAI Press, pp. 2852–2858.

Zazworka, N., Shaw, M.A., Shull, F., Seaman, C.B., 2011. Investigating the impact of design debt on software quality. In: Ozkaya, I., Kruchten, P., Nord, R.L., Brown, N. (Eds.), Proceedings of the 2nd Workshop on Managing Technical Debt, MTD 2011, Waikiki, Honolulu, HI, USA, May 23, 2011. ACM, pp. 17–23. http://dx.doi.org/10.1145/1985362.1985366.

Zhang, Y., Wallace, B.C., 2017. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. In: Kondrak, G., Watanabe, T. (Eds.), Proceedings of the Eighth International Joint Conference on Natural Language Processing, IJCNLP 2017, Taipei, Taiwan, November 27 – December 1, 2017. In: Long Papers, vol. 1, Asian Federation of Natural Language Processing, pp. 253–263.

Kuiyu Zhu is a master student in software engineering at the Northwestern Polytechnical University, China. His research interests include software quality, mining software repositories, machine learning.

Ming Yin is an associate professor at the school of software of the Northwestern Polytechnical University, China. She received the Ph.D. in Software Engineering in June 2007 at the Northwestern Polytechnical University. Her research interests include software quality, analysis and forecast of smart tourism, software development demand forecast.

Dan Zhu is a professor of Information Systems and Computer Science at Iowa State University. She received her Ph.D. in Information Systems from Carnegie Mellon University. Her research emphasizes the development of computational and analytical models to support business decision makings.

Xiaogang Zhang is a master student in software engineering at Northwestern Polytechnical University, China. His research interests include software quality, deep learning, data mining.

Cunzhi Gao is a master student in software engineering at Northwestern Polytechnical University, China. His research interests include software quality, deep learning, data mining.

Jijiao Jiang is an associate professor of management school at Northwestern Polytechnical University, China. He holds a doctorate degree in management science and engineering from Northwestern Polytechnical University. His research interests focus on agile software development, social media, expertise coordination and etc. He also concerns about software development teams in agile software development and in requirements engineering.