Automatic Labeling of SDN Controller Defect Text based on Neural Topic Model

Bing Zheng[1,2] and Hua Li[1,*]
[1] Department of Computer Science Inner Mongolia University, Hohhot, Inner Mongolia, China
[2] Inner Mongolia Technical College of Construction, Hohhot, Inner Mongolia, China
zhengbing@mail.imu.edu.cn, cslihua@imu.edu.cn
*corresponding author

***Abstract*—Software defined networking controller defects may cause unexpected failures and reduce network reliability. The historical defect texts provide information on defecting symptoms and fixing strategies, but they are not sufficiently labeled. An approach based on a neural topic model is proposed to automatically assign phrase labels to the defect text. First, five types of information that can be extracted from the defect text are provided to guide the generation of candidate phrases. Second, contextualized embedding representations are extracted by the pre-training BERTOverflow model and input to the contextualized topic model to extract a more coherent topic distribution. Third, the accuracy of the unsupervised assignment of labels is improved by filtering candidate phrases through two-level label filtering. Finally, FastText is applied to train a multi-label classifier to assign labels to the new defect text. The experimental results demonstrate that the proposed approach can effectively assign interpretable labels to the defect text.**

Keywords- SDN controller; defect; text mining; neural topic model; phrase label

I. INTRODUCTION

Software defined networking (SDN) is a popular technology in cloud data centers. The SDN controller is the core component responsible for managing the network and is crucial to operating an SDN. Currently, many open-source controllers such as OpenDaylight (ODL) and Open Network Operating System (ONOS) play an important role in the production environment of SDN networks[1]. If there is a defect in the SDN controller, the reliability of the network is reduced, and the operation of the network is unexpected. Therefore, it is necessary to fully analyze SDN controller defects using data mining technology.

Software repositories such as issue tracking system (ITS) help researchers in studying and examining the different factors that influence and impact the quality of a software system[2]. In the information stored in the historical ITS repositories, the problems reported are not necessarily failures, errors, or defects but anomalies that include any deviation from the desired behavior of the system[3]. It can detect the defect types in the SDN controller through the guidance of historical information.

Research Motivation–A real historical defect report text (Issue ID ONOS-705) from SDN controller project ONOS is chosen as an example of motivation, where the ITS tool of ONOS is Jira[1]. The defect text consists of three parts: title, description and comments, the information contained in it is presented in Table 1.

Table 1. Examples of text and information contained in defect report of SDN controller

	Title	Description	Comments
Text	Exception while running ONOS (master) against Pica8 switch …	After ONOS starts, I'm not able to see the switch connected. … The exception is triggered with ONOS master and with the switch either in OF 1.0 or 1.3 mode. …	This issue seems to be a race condition with the order events are written to the device store. Sometimes device added events are added by the startup thread because they can be triggered by …
Infor-mation	• observed behavior	• detailed observed behavior • steps to reproduce • root cause	• root cause • fix strategy and link.

Table 1 shows that the historical defect text contains ample useful information, and the underlined text represents the entities and activities associated with the defect. The defect text records observed behavior, steps to reproduce, root cause, and fix strategies. Existing ITS tools provide information to users and categorize issues through labels; however, the labels provided are based on general purposes and are often missing. A survey on ITS GitHub in 2018 found that 72% of the 1,656,937 issue reports had no labels[4]. Our pilot study found a similar situation in Jira. Therefore, the defect text could provide useful historical defect information and the labels provided by the existing ITS tools are insufficient to meet the requirements of in-depth SDN controller defect analysis.

Effective mining and reuse of knowledge gained during the reporting and fixing of defect issues can reduce production costs and increases the efficiency of defect fixes[5]. Therefore, we aim to mine SDN controller defect texts to make sense of historical defect text. A defect text generally describes events with specific topics[6], and an intuitive idea is to classify defects with similar topics to guide defect detection and fix. In addition, more hints of defect information are provided to developers and users in the form of labels. With these considerations in mind, we propose a method to automatically provide users with interpretable labels using text-mining technology, which

[1] https://jira.onosproject.org/

2693-9371/22/$31.00 ©2022 IEEE
DOI 10.1109/QRS-C57518.2022.00065

help users effectively identify entities and activities related to defects.

According to our investigation, there are insufficient labeled defect texts, so unsupervised and supervised text-mining technologies were combined in this study. We assign the same labels to texts with similar latent topic semantics through neural topic model clustering. Furthermore, the problem that should be addressed for the labels of the clustered topics is how to make the labels contain the expected information and ensure that they can be assigned as accurately as possible to the defect texts. Finally, a supervised multi-label text classification technique is applied to automatically label new defect text.

The contributions of this study are listed as follows:

- The characteristics of SDN controller defect texts are analyzed. Four phrase extraction methods are combined to ensure that the provided labels contain defect-related entities and activities.
- An automatic labeling approach for SDN controller defects is proposed. The unsupervised neural contextualized topic model (CTM) is applied for topic clustering; the clustered documents are labeled by two-level label filtering, and the new defect text is automatically assigned multiple labels by the supervised classifier FastText.

The remainder of this study is organized as follows. Section II provides related work by discussing the current state of research on defect analysis of SDN controllers and topic-modeling techniques. Section III focuses on the methodology for designing and constructing the proposed approach. Section IV discusses the results obtained from the experiments with the proposed framework on an SDN controller. Finally, Section V provides concluding remarks.

II. Related Work

The SDN controller is the core component responsible for managing the network. Several studies have focused on SDN controller defects [7]. In the work of development, operation, and maintenance, SDN controller defects are often described in natural language text. By mining and analyzing SDN controller defects text in defect reports, information can be provided to support for defect analysis. Vizareta et al. presented the change law of the learning curve mode and error detection rate of the number of reported defects through a statistical analysis method[8]. Bhardwaj et al. focused on the "critical" defects caused by the SDN controller and presented classification and root cause analysis of "critical" defects[9]. The defect text could provide useful historical defect information. However, existing studies have not conducted in-depth semantic mining of text in an SDN controller historical defect report.

With the development of SDN technology, defect texts gradually accumulate, and it becomes difficult to mine and reuse knowledge in defect texts. We aim to mine historical defect information by clustering and automatically labeling defect text. The topic modeling technique is widely used in the field of defect text mining [4, 6, 10, 11]. Schopf et al. found topic modeling to be a promising approach for defect report classification and explored automated classification of defect reports into a predefined set of categories[12]. Focusing on the security issues posted in GitHub repositories, Althar et al. applied topic modeling to identify and understand common security issues[13]. Xia et al. improved automated bug triaging with specialized topic model[5]. Treude et al. applies topic models to corpora from GitHub and Stack Overflow to make sense of this textual data[14].Therefore, the topic model is adopted for clustering latent semantic defect texts in our paper.

Topic models can divide the words in a corpus according to semantics to obtain a topic generated by a fixed number of semantically related words. The corpus in this paper consists of all defect texts from issue reports. Documents (i.e. issue reports in the corpus) can be clustered based on these topics. Topic models can be divided into non-LDA(latent Dirichlet allocation)-based topic models and LDA-based topic models[15]. Non-negative Matrix Factorization(NMF) is a representative non-LDA-based topic model, which is simple but unable to deal with complex topic learning problems[16]. LDA-based techniques are the most frequently used topic-modeling techniques for identifying topics and clustering documents with similar topics[15]. There are two types of LDA-based topic models: topic models based on probabilistic statistical methods and topic models based on neural networks.

Topic models based on probabilistic statistical methods, such as LDA, employ word frequencies and co-occurrence of words in the documents in a corpus to build a model of related words[17]. Such methods extract meaningful word distributions from unstructured text, for example, bag-of-words (BoW) representations, and the order of words in the document is ignored. However, unsupervised topic models do not often correlate well with human judgment, and its semantic coherence is not ideal [18].The topic model can use the pre-trained word vector to enrich the text feature space expression, such that texts and words with similar semantics can be assigned to the same topic, thus improving the semantic coherence of the topic words and the accuracy of text clustering [19]. Gaussian LDA uses Word2Vec pre-training to improve semantic coherence [15].

Topic models based on a neural network can be applied to reconstruct the generation process of the topic model. The deep neural network helps to learn topics with better interpretability[15]. Owing to the flexibility of deep neural networks, inference networks can learn complicated non-linear distributions and process structured inputs such as word sequences [10]. The ProdLDA model mainly uses the variational autoencoder (VAE) framework to reconstruct the text generation process of a topic model and adds sparse constraints of topic vocabulary to generate more expressive topic words[20]. The feature extraction ability of the ProdLDA topic model is stronger than that of the LDA. The CTM introduces a bidirectional encoder representation from a transformer (BERT) pre-trained model in the embedding layer representation of ProdLDA to improve the semantic coherence of topics [20]. Continuous space word embeddings learned from large unstructured corpora can effectively capture semantic regularities.

Topic models based on probabilistic statistical methods cannot obtain features related to word order and higher-level semantics. Various extensions of topic models incorporate several types of information that are convenient for defect analysis and fix. The neural topic model CTM extends VAE framework, and the pre-trained representations are introduced in the embedding layer. It can capture semantic relationships and help advance feature extraction and cluster tasks [20]. Therefore, in this study, an approach for automatic labeling SDN defect text was established based on the CTM neural topic model.

III. APPROACH

A. Approach Overview

We propose a hybrid approach, as shown in Figure 1. The overall workflow is divided into four steps:

Step 1. Data preparation and text preprocessing. The dataset is extracted from Jira and GitHub. A detailed analysis of the defect information that can be obtained is provided, and preprocessing for the characteristics of the SDN controller defect text is also implemented.

Step 2. Topic modeling. This is an important step. To obtain a more coherent representation of labels, contextualized embedding semantic features are extracted through the in-domain pre-training model, BERTOverflow. CTM is applied to topic modeling. The contextualized document representation and BoW representation are concatenated as the input of the CTM, and the document-topic distribution and word topic distribution with better semantic coherence are extracted through the VAE framework. Documents are assigned to two topics based on the maximum posteriori principle.

Step 3. Two-level label filtering. Candidate phrases are extracted using named entities, bigrams and trigrams, phrase chunks, and custom in-domain terms. ①At the topic level, the top-3 phrases are selected as topic labels for each topic. Through the document-topic and topic-word distributions outputted by the CTM model, topics and candidate phrases are compared for similarity by KL (Kullback-Leibler) divergence and sorted based on the result calculated by the similarity score function. ②At the document level, cosine similarity calculation is performed using the context embedding representation of the document and topic label. Labels are reordered and top-3 topic labels are selected as the labels for each document. After two-level label filtering, the remaining labels not assigned to any document are eliminated, and a label set is built for the multiple-label classifiers. To ensure the accuracy of the results, validation was performed using manual sampling.

Step 4. Multi-label classification. We use the labeled dataset obtained in Step 3 to train a multi-label classifier. To automatically assign interpretable labels to new defect text, we train a classification model with FastText on an already labeled multi-label dataset.

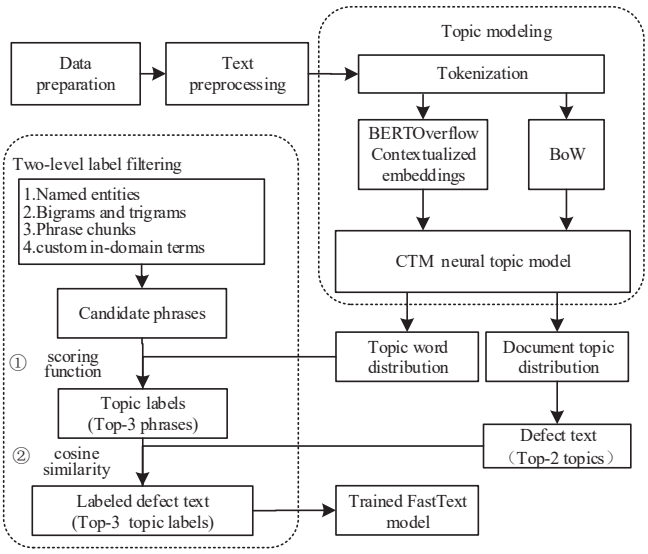

Figure 1. Overall workflow of the proposed approach.

B. Data Preparation and Preprocessing

Several studies have tested their methods on cross-projects or combined data from all projects, which would be better in the scope of their validity[4, 14]. Our dataset is extracted from two different ITSs, including Jira and GitHub, and contained 10 open-source SDN controller projects. From the perspective of natural language processing (NLP) technology research, we use a specific non-standard dataset. The crawler Beautiful Soup is used to obtain the raw dataset and perform specific preprocessing and feature extraction for input to the topic model to consist of various types of defect information.

For the collected raw dataset, it is necessary to determine the portion of defect text that is included. As listed in Table 1, the defect text consists of three parts: title, description and comments. Our pilot study shows that "title" and "description" contain more defect information. However, the reporter is likely not fully aware of the defect. It is often in the follow-up discussion with the developer that more observed behavior become clear, and a more reliable root cause analysis can be provided. Also, fix strategies are often given in the comments. Therefore, in order to provide more detailed defect information, the defect text in our dataset includes all the three parts.

Mining defect text requires prior knowledge of the information contained in the text. In reference[3], three types of information exist in defect reports, namely, observed behavior (OB), expected behavior (EB), and steps to reproduce (S2R). According to our additional analysis, five types of information can be obtained through historical defect text mining: OB, EB, S2R, root cause, and fixed strategy. It is also the most important in the process of defect analysis and fix, and therefore, is expected to be reflected as labels in this study. The entities and activities associated with the defect are contained in five types of information. Through the analysis, it is found that the Part of Speech (POS) of the words contained in this information is mainly

nouns and verbs. Therefore, in the two-level label filtering step, we mainly extract phrases containing nouns and verbs, thus trying to make the provided labels contain defect information

Defect text obtained from ITS is often noisy and greatly differs in its quality of information. Text preprocessing significantly affects the reliability of semantic modeling, and the poor quality of data leads to unexpected negative details in the classification results. In addition to the common NLP processing flow, each text-mining approach (including topic modeling) may require specific preprocessing steps. The following preprocessing techniques are applied:

1) Regular expression matching is used to eliminate irrelevant information, including URLs, IP addresses, and special characters. The predefined stop words are eliminated together with the stop words provided by NLTK. The digits are not simply removed; for example, the deployment environment of the SDN controller includes k8s, and the l2switch is an important component of the ODL, including layer 2 switch functions. These terms are added to the reserved list.

2) There is an OOV(out-of-vocabulary) problem; many unknown words are contractions; therefore, contractions should be expanded. For example, for distributed deployed controllers, the term "async" often appear and should be converted into "asynchronous." After processing, the proportion of OOV is reduced. Therefore, some terms specific to SDN controller items are added as reserved and rule lists based on studying the software engineering term list provided by[21].

3) Descriptions and comments sometimes include code snippets, trace entries, configurations, and operation commands. Structured information is typically a good indicator of the main functionality of an entity, and the words used in commands often reflect the purpose of the task being resolved. For example, in SDN controller Cord, “flowRuleCount” indicates the number of flow rules in the device, which should be divided into “flow,” “Rule,” and “Count.” Split tokens are based on several naming conventions, including SnakeCase, CamelCase, and underscores.

4) Elimination of infrequent and most frequent words (according to a specified frequency threshold) to limit vocabulary size. Less frequent words are typically special names or typos. Based on the experiments, using these tokens had less or no impact on the topic model.

We do not perform stemming and lemmatization filters because some important words become meaningless after processing. Moreover, some of our methods (e.g., BERTOverflow) have their preprocessing techniques. Because the pre-training model has specific requirements for the length of the input text, more dataset text information is presented in Table 2.

After preprocessing, the corpus C for automatic labeling is obtained, where C is composed of documents d, that is, $C = \{d_1, d_2, \ldots, d_m\}$. Document d is regarded as a simple collection of tokenized words, that is, $d = \{w_1, w_2, \ldots, w_n\}$. A unified dictionary, V, is established for subsequent feature extraction, topic labeling, and classification. The dictionary is a list of all unique words from corpus C, along with indices. Our corpus contained 17,414 documents (35,984 unique words in total) with a mean length of 312 words. The average number of input tokens per SDN controller ITS repository is 312. Although the title text has less noise, it can be seen from Table 2 that the title become a typical short text after preprocessing, and there will be a sparse problem after topic clustering. Therefore, our corpus contains all three parts of defect text, which can get better topic clustering effect and obtain labels containing more defect information.

Table 2. Defect text from SDN controller ITS*

SDN controller	ITS	#docs	#w/dt	#w/d
OpenDaylight	jira	10,117	7.42	333.22
ONOS	jira	2,241	7.18	570.53
TungSten	Jira	1,739	6.96	351.50
Open-Kilda	github	1,486	7.16	129.85
CORD	jira	796	6.22	90.15
Faucet	github	616	6.20	116.65
Pox	github	180	5.35	113.90
FloodLight	github	167	5.32	143.50
Ryu	github	46	4.72	133.43
Nox	github	26	4.73	117.77

*#docs: number of issue reports; #w/dt: average number of words per document title; #w/d: average number of words per document.

C. Topic Modeling

The tokenized word list obtained after preprocessing constitutes a high-dimensional word space that is not interpretable. Our goal is to automatically assign labels to the SDN controller defect text, that is, to label each document. A good set of labels should be understandable to user text[22]. The keyword-extraction algorithm can be directly applied to a preprocessed corpus. However, the granularity of the labels extracted in this manner is more meaningful than the title.

In the topic model, a topic is essentially a cluster of words of a given vocabulary ranked by the probability of belonging to such a cluster. The topic model captures the topic distributions of each document. Since typically $k \ll n$, topic model is mapping the document from the space of words (n) into a smaller space of topics (k). Therefore, the topic model is trained to cluster documents such that we can assign interpretable topic labels to documents.

To cluster documents with similar topics, it is necessary to choose topic modeling methods with better topic consistency. The first step is to extract the vector representation. Documents and words can be represented as vectors. A richer representation that captures semantics has a significant effect on improving topic modeling coherence and classification accuracy and facilitates semantic similarity calculation for labeling. In our approach, two types of vector representations are extracted: BoW representation and contextualized embedding representation.

The BoW model uses a set of unordered words to express sentences or documents and regards the document as a simple collection of words. The vector representation does not consider the order in which the words appear in the text;

only the frequency of each word in the dictionary appearing in the text. The information that can be provided by the BoW is mainly word frequency information, which can be considered as words with a multinomial distribution. A few additional steps are applied here to generate bigrams and trigrams and to eliminate all words except nouns, adjectives, verbs, and adverbs. Bigrams and trigrams are used in the candidate phrase extraction process. The BoW represents the input in an inherently incoherent manner. Since grammar and word order are ignored, the consistency of the latent topic distributions obtained by the BoW is often unsatisfactory.

Contextualized embedding representation is a type of word-embedding method[20, 23]. Word embedding is the representation of words in the form of a dense vector of floating-point values that encode the meaning of the word such that, the words that are closer in the vector space are expected to be similar in meaning. This characteristic can also be used to compare the semantic similarities between phrases and documents. Classical embedding methods such as Word2Vec, FastText, and GloVe generate fixed vectors for polysemous words irrespective of the context in which they occur[4]. Moreover, many words in the SDN controller defect text have different meanings from those in the general corpus, and it is necessary to solve the problem of polysemy in embedding representations to more accurately represent documents. For example, SDN manages network traffic based on the flow. "Flow" has important and specific meanings in the corpus. Contextualized embeddings capture the context of word usage and, hence, produce different vector representations for the same word depending on the context. Therefore, the problem of polysemy can be resolved. The state-of-the-art model that can extract contextual embedding representations is BERT.

BERT is based on the encoder module in the transformer architecture and uses an attention mechanism [24]. This type of embedding contains position, sequence, and context information, which are token, segment, and position embeddings. BERT employs learnable position-encoding information. The attention mechanism helps encode a word using other positions in the input sequence that would lead to a better representation of the word. To incorporate the rich information from very large domain datasets, pre-trained word (document) embedding vectors trained on external corpora are utilized. BERT is trained in the general domain, which has a different data distribution from that of the target domain. Therefore, it is necessary to further pre-train BERT with the target domain data. There are two existing strategies for applying pre-trained representations: feature-based and fine-tuning [24]. In our approach, we adopt the pre-training model of fine-tuning based on Stack Overflow, because this Q&A website mainly reports and solves problems encountered in software development and use, and our defect text mining task has the same goal. Embedding trained on a domain corpus performs better than embedding on the general corpus. The pre-trained BERTOverflow model from Huggingface is fine-tuned on the Stack Overflow dataset[25].

After extracting the vector representation of the defect text, it is necessary to establish a topic model to cluster the text that needs to be labeled. A crucial issue with topic modeling is the quality of the topics they discover. The neural topic model CTM produces more meaningful and coherent topics than traditional LDA[20]. In the CTM, the probability vector of each document topic provides the latent topic semantic distribution characteristics at the document level. CTM combines contextualized representations with neural topic models and implements black-box variational inference using VAE. The VAE framework explicitly approximates a Dirichlet prior to using Gaussian distributions. The latent semantic space is constructed after matrix dimensionality reduction, and the word document matrix is reconstructed. In our approach, we introduce the in-domain pre-training model. The topic generation process of the neural topic model CTM that introduces the in-domain pre-training model is shown in Figure 2.

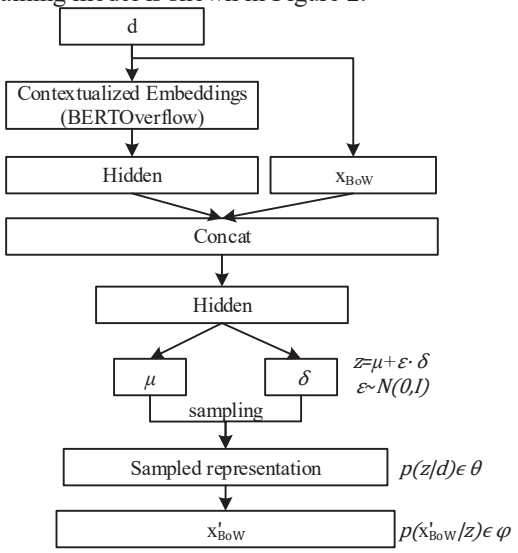

Figure 2. Neural topic model CTM with in-domain pretraining model.

In Figure 2, document *d* is represented by contextualized embedding and BoW(x_{BoW}). The two representations are used as the input of the topic model in a concatenated manner to obtain more consistent topics. VAE aims to model the true posterior distribution $p(z|d)$ of the latent variable *z*, which is the probability distribution of the topic of document *d* and is a *k*-dimensional vector. The neural variational framework trains a neural inference network to directly map the concatenation of contextualized document representations and BoW into a continuous latent representation $p(z|d)$. The latent document representation is sampled from a Gaussian distribution parameterized by μ and σ, which are parts of the variational inference framework, where $z\sim\mathcal{N}(\mu,\sigma)$. A reparameterization trick is used in this process[26]. A decoder network is then used to reconstruct the representation by generating its words using the latent document representation $p(x'_{BoW}|z)$, where x'_{BoW} is the reconstructed representation. The latent topics reconstruct the words in their respective documents to the maximum extent possible. This means that the words in a document lie roughly on the plane formed by their

corresponding topic vectors. Text similarity comparison can be performed with the help of topic distribution.

The output of CTM training is a topic model that contains the following information:

- k topics, where each topic is a distribution of words;
- Document-topic matrix θ, probability of topic to occur in documents;
- Topic-word matrix φ, topic assigned to word w in a document.

In the CTM topic model, a document is a probability distribution over topics and a topic is a probability distribution over words. Therefore, a single document contains one or more topics. We estimate the dominant topic members for documents in each cluster using the maximum posteriori principle:

$$z_{d,i} = arg\ max_{z_{d,i}=0,\dots,k}\ p(z_{d,i}|d), and\ p(z_{d,i}|d) \in \theta \quad (1)$$

Particularly, we use the top two topics for each document, as they usually provide sufficient detail to convey the information of a document and distinguish one document from another[27]. Therefore, document d belongs to the i_{th} cluster if and only if t_i is among the top two topics of d.

D. Two-level Label Filtering

After topic clustering, labels with better interpretability and defect-related information are assigned to the clustered documents. The motivation behind document labeling is that topics are time-consuming to interpret and are faster to reason about if they are labeled, particularly in the activities of detecting and fixing defects. One problem with using topic model clustering documents is that the generated representative labels are often insufficiently interpretable [20]. Topics can be simply represented by their top-*N* terms, that is, by words with the highest probability in a topic distribution[6]. A good set of labels should capture as much semantic information as the text. Unfortunately, such labels in words may not be helpful in accurately capturing the semantics of topics. Therefore, we use phrases as labels, and propose a two-level label filtering method to make the labels of defect text more representative.

To assign labels, the method is as follows: first, the candidate phrase label set is selected by combining four phrase extraction methods; then, the two-level label filtering method is adopted, that is, the topic label is selected from the candidate phrase label set, the filtered topic labels are reordered by similarity calculation, and document labels are assigned to documents.

1) Entities can be extracted as named entities from the title by NLTK NER (named entity recognition) tools. The extracted named entities are often words that distinguish topics and have specific marking meanings, such as proper nouns. Named entities are extracted from the title text because the title contains more entity information, whereas the description and comments contain a lot of noise.

2) Subject-predicate and verb-object phrases should be selected to determine the activities performed by or on entities that may exist in the corpus. We construct a syntax tree and extract phrases with specific regular expression rules <NN. *>{1,2}<VB.*>, <VB.*><NN.*>{1,2}. This task is implemented by applying the NLTK POS tools. This tagging technique analyzes the grammatical role of words in the text and helps eliminate undesired material.

3) The most significant bigrams and trigrams are extracted using NLTK. Because of noise, it is necessary to filter the phrases generated by the bigram and trigram.

4) SDN related terms (e.g., related protocol OpenFlow) are merged into a phrase set. The terms are crawled from websites such as ODL official documentation. Introducing in-domain terms significantly improves label accuracy [28].

For phrases obtained using methods 3), PMI (Pointwise Mutual Information) should be calculated to ensure that meaningful phrases are obtained. Higher PMI values indicate that the combination of two words is more likely to be a meaningful phrase. We extract meaningful phrases by experimentally setting a threshold for the PMI. Phrases with PMIs greater than the threshold are extracted.

$$PMI(w_i, w_j) = log\frac{p(w_i,w_j)}{p(w_i)p(w_j)} \quad (2)$$

where $p(w_i, w_j)$ indicates the co-occurrence probability of the phrase, $p(w_i)$ and $p(w_j)$ represents the probability of the words w_i and w_j in the whole corpus C. Higher PMI values indicate that the combination of the two words is more likely to be a meaningful phrase.

We combined the results of the four methods to extract a set of candidate phrases. To ensure the accuracy of the results, we completed the manual validation process independently. Finally, the candidate phrase sets are obtained.

Based on the candidate phrase sets, a two-level filtering method is applied to assign labels to documents. At the topic level, the similarity between candidate phrases and topics is calculated, and the topic labels are obtained by sorting and filtering using the scoring function. At the document level, the similarity between the topic label embedding representation and the document embedding representation is calculated, and the document labels are obtained by sorting and filtering.

At the topic level, a good set of labels should have high semantic relevance to the target topic , and low relevance to other topics. Topic labeling is the process of determining or generating appropriate labels for topics derived from topic distributions over words inferred from the topic model CTM. We rank the candidate phrases using a scoring function. The topic-word matrix φ is extracted from the output of the CTM in Section III-C. KL(Kullback-Leibler) divergence is employed, as in [27] to measure the semantic similarity between one candidate phrase α and the target topic φ_i, defined as:

$$sim(\alpha, \varphi_i) = -D_{KL}(\alpha||\varphi_i) \approx \sum_{w\in C} p(w|\varphi_i) log\frac{p(a,w|C)}{p(a|C)p(w|C)} \quad (3)$$

where w is the word in the entire corpus C. $p(w|\varphi_i)$ is the probability of w in the topic distribution φ_i. $p(a|C)$, and $p(w|C)$ denote the percentages of words w and a in corpus C. $p(a,w|C)$ indicates the probability of co-occurrence of a and w in C.

The scoring function is then defined by combining $sim(\alpha, \varphi_i)$ with similarity scores to other topics,

$sim(\alpha, \varphi_{1,\ldots,i-1,i+1,\ldots,k})$ is the sum of similarity scores of other topics. The combined similarity scores mean that α should be semantically close to the topic distribution φ_i and discriminate from other topics.

$$Score(\alpha, \varphi_i) = sim(\alpha, \varphi_i) - \frac{\lambda}{k-1} * sim(\alpha, \varphi_{1,\ldots,i-1,i+1,\ldots,k}) \quad (4)$$

where the parameter λ is used to adjust the penalty for semantic similarities to other topics. A larger λ signifies that the candidates are more different from the other topics.

For a topic, candidate phrases are ranked using a scoring function[27]. Based on the survey, three labels are moderate choices for users to comprehend the topics. Therefore, for each topic, we chose the top 3 phrases as topic labels. Since each document is assigned two topics, each topic has three topic labels, and each document has six topic labels.

At the document level, it is necessary to compare the semantic similarity of topic labels and documents and ensure the accuracy of document labels through the filtering of document-level semantic similarity. Since our topic label was a 2-3-word phrase, its semantic similarity comparison with the document is a typical comparison of short and long texts. We use the contextualized embeddings obtained from the fine-tuned pre-trained BERTOverflow model for both phrases and documents to perform label filtering. The embedding of a phrase is approximately the sum of the embeddings of its component words. Since all learned embeddings share the same feature space, their distances can be considered their semantic similarity. The cosine similarity of the contextualized embeddings of phrases and documents is calculated, and the phrases are sorted in descending order. We choose the top 3 phrases for each document. After two-level label filtering, the remaining labels that are not assigned to any document are eliminated.

Unsupervised clustering is adopted to select labels for documents, and manual validation is applied through sampling to ensure that more accurate labels are assigned to documents as possible. Domain experts are required to accurately label documents. We rely on the domain knowledge of the two authors and a graduate student, and randomly select a statistically representative sample of 4,354 labeled documents from the dataset. This sample allows us to generalize the conclusions with a confidence level of 95% and a confidence interval of 1%. Through two-level label filtering, the number of labels is significantly reduced, a small number of key phrases could be used to label texts, and defect texts could be classified. An SDN controller defect-text dataset with interpretable labels was obtained.

E. Multi-label Classification

The final step of our approach is to build a classifier that assigns multiple labels to new defect text. The accurate classification of new defects and more understandable labels is a multi-label classification problem. The model should be trained to label new defect text as accurately as possible. The CTM is a full generative model, and the inference is not sufficiently fast [10]. Generally, supervised methods are a good approach when time is limited. Furthermore, from the perspective of accuracy, supervised methods are preferred. FastText is a lightweight supervised learning method that can be used for text classification, particularly in the case of rare words, by exploiting character-level information. FastText proposes a method of subword embedding that extracts n-gram features for each word, where n-gram is the character level. FastText uses character-level n-grams to represent words. We use FastText to train the supervised text classifier.

The input of the FastText model is the sequence of words and the output is the probability that the sequence of words belongs to each category. During model training, we use titles, comments, and descriptions from the corpus as input, which could cover the content that may be missing in the title. For labels, we transform these phrase labels into multi-hot-encoded vectors and use them in multi-label classifiers. When using the classifier for multi-label classification of new defect text, the title can only be used as an input. The FastText model is illustrated in Figure 3.

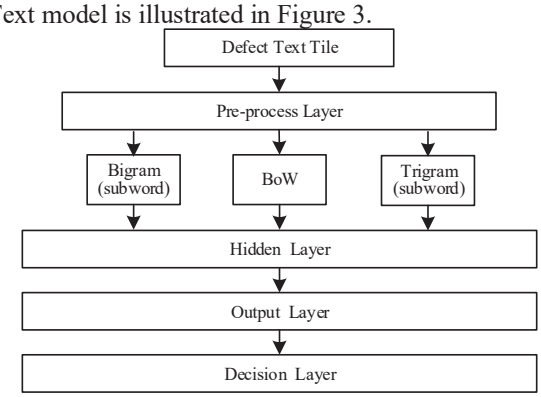

Figure 3. FastText model .

IV. EVALUATION

In order to verify the effectiveness of our proposed approach, this section has carried out various confirmatory experiments. The operating system in this paper is Ubuntu 18.04 LTS, the hardware configuration is Intel(R) Xeon(R) CPU @ 2.20GHz, 16G RAM, NVIDIA-SMI 460.32.03, CUDA 11.2, development language is python 3.7.13, development framework includes scikit-learn 0.24.2, pytorch 1.12.1. All evaluations will be performed on our specific SDN controller defect text dataset (Section III-B).

In order to ensure that the neural topic model with the contextual embedding representation as input has good consistency and can reflect the content of the defect text, the semantic coherence and the topic diversity are evaluated. The evaluation metrics are from the reference[16]. To ensure the accuracy of automatic classification, the classification accuracy and loss values are presented as evaluation content. The evaluation metrics are from the reference[21].

A. Optimal Evaluation of Topic Models

The word embedding layer of the neural topic model adopts the BERT pre-trained language model. A strategy to address the maximum length limitation needs to be given. BERTOverflow model takes an input of a sequence of no more than 512 tokens [24]. The part of the document that

exceeds the maximum length is often the information about root causes and fix strategies in descriptions and comments. After preprocessing, we have 13.5% (2347) documents with more than 510 characters length (L) and adopt hierarchical methods in order not to lose information as much as possible [21]. The input text is divided into S=L/510 fractions, which is fed into BERT. The representation of each fraction is the hidden state of the [CLS] tokens of the last layer. The mean pooling is used to combine the representations of all the fractions. By fine tuning BERTOverflow, we get the word contextualized representation and document contextualized representation.

The hyperparameter *k* has a significant impact on topic consistency. In the parameter optimization process of CTM, 2000 iterations are selected, and the optimal number of topics *k* is found by perplexity and coherence values, 5-200, 5 is the step size. For topic model optimization, two metrics are used for evaluation, including coherence value and the perplexity, as shown in Figure 4.

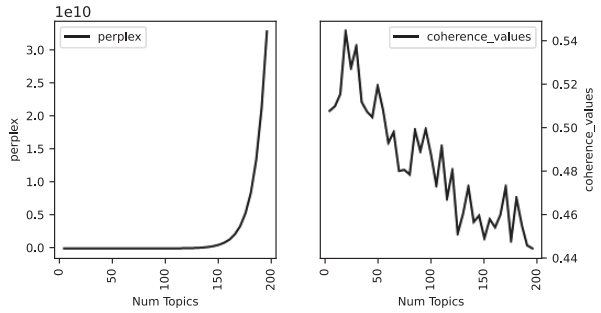

Figure 4. The perplexity and coherence values .

When there are 20 topics, the coherence value and perplexity are ideal, but for the consideration of label richness, we choose 60 as the topic number, which also has a good metric value. Because of the defect text we use, it is generally believed that the topics reflected in a defect text are concentrated, and the experimental results are also the same.

For comparison with other topic models, we choose the most commonly used benchmark model, LDA, and the non-LDA-based topic model NMF. Furthermore, to evaluate the effect of in-domain embedding representation, BERT-Base and BERTOverflow are used as the contextualized embedding representation layer of CTM respectively. The results are listed in Table 3.

Table 3. Topic coherence&diversity on different topic model

Model	Coherences		Diversity	
	T(20)	*T(60)*	*T(20)*	*T(60)*
BoW+LDA	0.1629	0.1161	0.6450	0.5983
BoW+NMF	0.1633	0.1270	0.5351	0.3767
BERT-Base+CTM	0.1632	0.1381	0.8220	0.7701
BERTOverflow +CTM	**0.1651**	**0.1464**	0.7552	0.6703

From the results listed in Table 3, it can be found that the coherence value of the neural topic model is better than that of the two baseline topic models. Although the in-domain embedding contextualized representation is slightly worse than general BERT in topic diversity, it can improve topic semantic coherence, which is the focus of our approach.

B. Effectiveness of Multi-label Classification

We divided our preprocessed dataset of GitHub repositories (Section III-D) to three subsets of training, validation, and testing datasets. We first split the data into train and test sets with ratios of 80%, and 20%, respectively. Then we split the train set to two subsets to have a validation set as well (with ratios 90% to 10%).

In the evaluation of multi-label classification effect, GLoVe+LSTM is selected as the model for comparison[13]. The learning rate is set to 3e-5, the number of epochs to 4 and the batch size to 32. We set the remaining parameters to default values. Precision and loss values are chosen as evaluation metrics. The training and evaluation loss values for 4 epochs are shown in Figure 5.

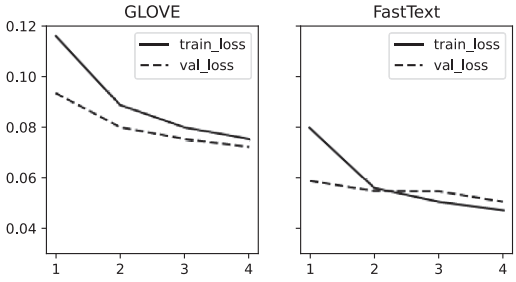

Figure 5. Comparison of FastText model and GloVe model .

As shown in Figure 5, on our dataset, the loss value of FastText model is less than GLoVe+LSTM, and the accuracy of the two after 4 epochs is 0.8817 and 0.8768, respectively. Such accuracy results are acceptable due to our custom dataset. FastText model is better than our chosen baseline model, proving that our method is effective.

V. CONCLUSION

Defects in the SDN controller can affect the reliability of the network. Historical defect text can guide a good understanding of the cause of triggering defects and develop a better fixing strategy. Giving meaningful labels to SDN controller defect text is important for controller defect comprehension. Further, it is very helpful for classifying, locating and repairing defects. In this paper, we present a method to automatically assign labels to SDN controller defect texts. The characteristics of the unlabeled text dataset are analyzed in detail. Based on neural topic modeling, the dataset is assigned phrase labels through a two-layer filtering method, and the assigned labels are related to entities and activities that contain defects. Based on the custom labeled dataset, the FastText model is applied as a classifier for automatic multi-label classification of new defect texts. The effectiveness of our method is verified by evaluation. The research can help developers understand the defects, better locate and repair defects in SDN controller software, and

also provide convenience for controller deployment and later maintenance.

Although with the help of neural topic models with contextual information, we obtain better consistent topics and better interpretable labels. Yet understanding defect text composed of human natural language remains difficult and subjective. Future studies should explore ways to apply different approaches to automatically label the defect text.

ACKNOWLEDGMENT

We do appreciate the great support of National Natural Science Foundation of China (No.61862047, 62066034), Inner Mongolia Science & Technology Plan (No. 2020GG0186), Inner Mongolia discipline inspection and supervision big data laboratory open project fund(No.IMDBD2020011), Research Program of science and technology at Universities of Inner Mongolia Autonomous Region(No. NJZY22425)

REFERENCES

[1] S. Ahmad and A. H. Mir, "Scalability, consistency, reliability and security in SDN controllers: a survey of diverse SDN controllers," *Journal of Network and Systems Management,* vol. 29, no. 1, pp. 1-59, 2021.

[2] S. Davies and M. Roper, "What's in a bug report?," in *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measuremen*, Italy, 2014: ACM, pp. 1-10.

[3] O. Chaparro *et al.*, "Detecting missing information in bug descriptions," in *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, 2017, pp. 396-407.

[4] X. Xie, Y. Su, S. Chen, L. Chen, J. Xuan, and B. Xu, "MULA: A just-in-time multi-labeling system for issue reports," *IEEE Transactions on Reliability,* vol. 71, no. 1, pp. 250-263, 2021.

[5] X. Xia, D. Lo, Y. Ding, J. M. Al-Kofahi, T. N. Nguyen, and X. Wang, "Improving automated bug triaging with specialized topic model," *IEEE Transactions on Software Engineering,* vol. 43, no. 3, pp. 272-297, 2016.

[6] A. Panichella, B. Dit, R. Oliveto, M. Di Penta, D. Poshynanyk, and A. De Lucia, "How to effectively use topic models for software engineering tasks? an approach based on genetic algorithms," in *2013 35th International conference on software engineering (ICSE)*, 2013: IEEE, pp. 522-531.

[7] Y. Yu *et al.*, "Fault management in software-defined networking: A survey," *IEEE Communications Surveys & Tutorials,* vol. 21, no. 1, pp. 349-392, 2018.

[8] P. Vizarreta, K. Trivedi, V. Mendiratta, W. Kellerer, and C. Mas-Machuca, "Dason: Dependability assessment framework for imperfect distributed sdn implementations," *IEEE Transactions on Network and Service Management,* vol. 17, no. 2, pp. 652-667, 2020.

[9] A. Bhardwaj, Z. Zhou, and T. A. Benson, "A Comprehensive Study of Bugs in Software Defined Networks," in *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2021: IEEE pp. 101-115.

[10] X. Song, J. Petrak, Y. Jiang, I. Singh, D. Maynard, and K. Bontcheva, "Classification aware neural topic model for COVID-19 disinformation categorisation," *PloS one,* vol. 16, no. 2, p. e0247086, 2021.

[11] M. Pettinato, J. P. Gil, P. Galeas, and B. Russo, "Log mining to re-construct system behavior: An exploratory study on a large telescope system," *Information and Software Technology,* vol. 114, pp. 121-136, 2019.

[12] T. Schopf, D. Braun, and F. Matthes, "Lbl2Vec: An Embedding-based Approach for Unsupervised Document Retrieval on Predefined Topics," in *WEBIST*, 2021, pp. 124-132.

[13] R. R. Althar, D. Samanta, M. Kaur, A. A. Alnuaim, N. Aljaffan, and M. Aman Ullah, "Software systems security vulnerabilities management by exploring the capabilities of language models using NLP," *Computational Intelligence and Neuroscience,* pp. 1-19, 2021.

[14] C. Treude and M. Wagner, "Predicting good configurations for github and stack overflow topic models," in *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*, 2019: IEEE, pp. 84-95.

[15] U. Chauhan and A. Shah, "Topic modeling using latent Dirichlet allocation: A survey," *ACM Computing Surveys (CSUR),* vol. 54, no. 7, pp. 1-35, 2021.

[16] S. Terragni, E. Fersini, B. G. Galuzzi, P. Tropeano, and A. Candelieri, "Octis: comparing and optimizing topic models is simple!," in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, 2021, pp. 263-270.

[17] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research,* vol. 3, no. Jan, pp. 993-1022, 2003.

[18] D. Q. Nguyen, R. Billingsley, L. Du, and M. Johnson, "Improving topic models with latent feature word representations," *Transactions of the Association for Computational Linguistics,* vol. 3, pp. 299-313, 2015.

[19] S. Bhatia, J. H. Lau, and T. Baldwin, "Automatic labelling of topics with neural embeddings," *arXiv preprint arXiv:1612.05340,* 2016.

[20] F. Bianchi, S. Terragni, and D. Hovy, "Pre-training is a hot topic: Contextualized document embeddings improve topic coherence," *arXiv preprint arXiv:2004.03974,* 2020.

[21] M. Izadi, A. Heydarnoori, and G. Gousios, "Topic recommendation for software repositories using multi-label classification algorithms," *Empirical Software Engineering,* vol. 26, no. 5, pp. 1-33, 2021.

[22] Q. Mei, X. Shen, and C. Zhai, "Automatic labeling of multinomial topic models," in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2007, pp. 490-499.

[23] V. Venktesh, M. Mohania, and V. Goyal, "Topic Aware Contextualized Embeddings for High Quality Phrase Extraction," in *European Conference on Information Retrieval*, 2022: Springer, pp. 457-471.

[24] C. Sun, X. Qiu, Y. Xu, and X. Huang, "How to fine-tune bert for text classification?," in *China national conference on Chinese computational linguistics*, 2019: Springer, pp. 194-206.

[25] J. Tabassum, M. Maddela, W. Xu, and A. Ritter, "Code and named entity recognition in stackoverflow," *arXiv preprint arXiv:2005.01634,* 2020.

[26] A. Srivastava and C. Sutton, "Autoencoding variational inference for topic models," *arXiv preprint arXiv:1703.01488,* 2017.

[27] C. Gao, J. Zeng, M. R. Lyu, and I. King, "Online app review analysis for identifying emerging issues," in *Proceedings of the 40th International Conference on Software Engineering*, 2018, pp. 48-58.

[28] Z. Haj-Yahia, A. Sieg, and L. A. Deleris, "Towards unsupervised text classification leveraging experts and word embeddings," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 371-379.