DialTest: Automated Testing for Recurrent-Neural-Network-Driven Dialogue Systems

Zixi Liu
zxliu@smail.nju.edu.cn
State Key Laboratory for Novel Software Technology
Nanjing University
Nanjing 210023, China

Yang Feng*
fengyang@nju.edu.cn
State Key Laboratory for Novel Software Technology
Nanjing University
Nanjing 210023, China

Zhenyu Chen
zychen@nju.edu.cn
State Key Laboratory for Novel Software Technology
Nanjing University
Nanjing 210023, China

ABSTRACT

With the tremendous advancement of recurrent neural networks (RNN), dialogue systems have achieved significant development. Many RNN-driven dialogue systems, such as Siri, Google Home, and Alexa, have been deployed to assist various tasks. However, accompanying this outstanding performance, RNN-driven dialogue systems, which are essentially a kind of software, could also produce erroneous behaviors and result in massive losses. Meanwhile, the complexity and intractability of RNN models that power the dialogue systems make their testing challenging.

In this paper, we design and implement `DialTest`, the first RNN-driven dialogue system testing tool. `DialTest` employs a series of transformation operators to make realistic changes on seed data while preserving their oracle information properly. To improve the efficiency of detecting faults, `DialTest` further adopts Gini impurity to guide the test generation process. We conduct extensive experiments to validate `DialTest`. We first experiment it on two fundamental tasks, i.e., intent detection and slot filling, of natural language understanding. The experiment results show that `DialTest` can effectively detect hundreds of erroneous behaviors for different RNN-driven natural language understanding (NLU) modules of dialogue systems and improve their accuracy via retraining with the generated data. Further, we conduct a case study on an industrial dialogue system to investigate the performance of `DialTest` under the real usage scenario. The study shows `DialTest` can detect errors and improve the robustness of RNN-driven dialogue systems effectively.

CCS CONCEPTS

• **Software and its engineering** → **Software testing and debugging**.

KEYWORDS

dialog system testing, automated testing, deep learning testing

*Yang Feng is the corresponding author.

ACM Reference Format:
Zixi Liu, Yang Feng, and Zhenyu Chen. 2021. DialTest: Automated Testing for Recurrent-Neural-Network-Driven Dialogue Systems. In *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA '21), July 11–17, 2021, Virtual, Denmark.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3460319.3464829

1 INTRODUCTION

The development of Recurrent Neural Networks (RNN) models has made significant progress in processing the serialized data [8], which arises advancements of many software applications, such as machine translation [1], speech recognition [34], intelligent dialogue systems [6] and so on. Especially, many companies have been vigorously developing RNN-driven dialogue systems, such as Apple's Siri [1], Google Assistant [2] and Amazon Alexa [3]. These systems have been widely deployed in customer service, voice control, home kit, and other fields [22], which has brought convenience to people's daily lives.

However, the RNN-driven dialogue systems are not as reliable as many may believe. Recently, several erroneous behaviors of RNN-driven dialogue systems have aroused wide attention. For example, Amazon's smart speaker Alexa could be fooled by some corner input cases as to makes creepy laughs, which scares the elderly and children and affects their lives [4]. Microsoft's intelligence chatbot Tay was misled to post anti-Semitic, sexist, and racist after chatting with users online less than 24 hours [5]. These erroneous behaviors could lead to misunderstanding, threats to personal safety, or even political conflicts [11]. Thus, the testing of RNN-driven dialogue systems becomes an important yet challenging task.

Compared with the testing of conventional software systems, the testing of RNN-driven dialogue systems is much more difficult due to multiple reasons. First, different from conventional software applications, which depend on engineers to construct the business logic manually, the development of RNN models adopts a data-driven programming paradigm that forms the logic by learning from massive data [25]. This feature makes it difficult to apply conventional software testing techniques to test RNN-driven dialogue systems. Moreover, RNN models often consist of multiple layers and millions of parameters [37]. This structure naturally hinders the analysis of their behaviors (i.e., the outputs of the hidden neurons) and further makes conventional testing criteria ineffective. The last

[1]https://www.apple.com/siri/
[2]https://assistant.google.com/
[3]https://developer.amazon.com/en-US/alexa
[4]https://www.bbc.com/news/technology-43325230
[5]https://www.bbc.com/news/technology-35902104

but not the least challenge lies in the construction of testing oracles, which describe the correct output for a given input. For a given input sentence, the correct response of a dialogue system may not be deterministic. Thus, it is difficult to define a strict oracle for this kind of system.

Only a few research [3, 4] are conducted for detecting erroneous behaviors of RNN-driven dialogue systems [11]. All of them are designed to conduct black-box testing on dialogue systems. They generate test sentences by filling out pre-defined templates with various parameters, i.e., actions, objects, and numbers. Because all parameters are selected from a given set, the diversity of generated data is limited, even though the number of combinations could be large. Further, these testing approaches lack proper guidance in generating test cases, which may limit their efficiency in detecting erroneous behaviors.

To overcome the aforementioned challenges and shortages, in this paper, we design and implement an automated testing tool, namely `DialTest`, for detecting erroneous behaviors of RNN-driven dialogue systems and further guide to improve them. `DialTest` can generate testing sentences from the seed testing data, i.e., labeled sentences, with transformation operators. `DialTest` employs transformation-specific metamorphic relations between the transformed sentences and seed data to detect the faults automatically. Further, to improve the efficiency of fault detection, `DialTest` employs Gini impurity [41], which measures the likelihood of detecting fault for a test, to guide the sentence generation process.

We first experiment `DialTest` with four RNN-driven NLU models on three datasets. On the two critical tasks, i.e., intent detection and slot filling, we found hundreds of erroneous behaviors in these models. Further, the experimental results show that the generated test data can significantly increase the accuracy of NLU models after retraining. Moreover, we conduct a case study by applying `DialTest` on an industrial RNN-driven dialogue system to investigate its performance under the real usage scenario. The results show that `DialTest` can detect many erroneous behaviors and increase the robustness of the RNN-driven dialogue system by retraining the model with generated data.

The contributions of this paper can be summarized as follows:

- We present a systematic technique to automatically test the RNN-driven systems like dialogue systems by sentence transformation with Gini impurity guidance. We empirically demonstrate that the transformed sentences with Gini impurity guidance can effectively generate test cases for discovering potential erroneous behaviors of the tested model.
- We leverage transformation-specific metamorphic relations to automatically test the NLU models, which are the first component of a dialogue system. We demonstrate that different realistic textual transformations, i.e., synonym replacement, back translation, and word insertion, can generate test cases effectively. Our experiments also show that the generated data can be used for retraining and improving the robustness of RNNs.
- We implement `DialTest`, the first automated testing tool for RNN-driven dialogue systems. We employ `DialTest` to systematically test four NLU models on three datasets. Also, we conduct a case study on an open-sourced industrial dialogue system to empirically validate the effect of `DialTest`.
- We have made the source of `DialTest` available [6]. We have also released the generated transformed sentences and testing results.

2 BACKGROUND

2.1 The Workflow of Dialogue Systems

The general workflow of dialogue systems is shown in Fig. 1. The user enters text or voice into the system, and the system outputs text or voice information as feedback after a series of processes inside the system. Compared with the text-based dialogue systems, the audio-based dialogue systems employ voice information for interaction and convert the voice messages into text through an automatic speech recognition (ASR) module. Similarly, after the system generates the output text, the audio-based dialogue systems convert text messages into voice and played them out through the Text-to-Speech (TTS) module. The text processing workflows are completed in three primary modules, i.e., natural language understanding (NLU), dialogue management (DM), and natural language generation (NLG).

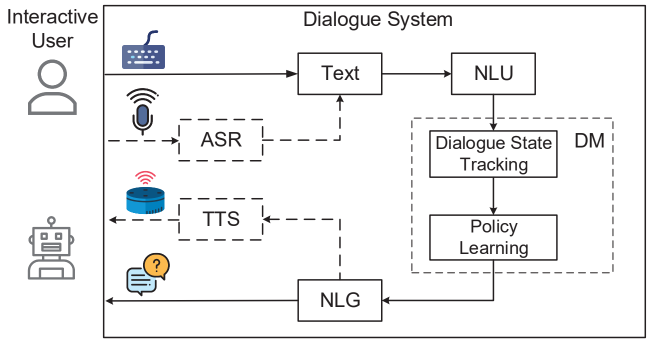

Figure 1: The general dialogue system workflow from the most popular development classes, such as Rasa [2], XiaoIce [55], and Topical-Chat [15].

As the sentence input into the dialogue system, the NLU module first identifies its intent and recognizes whether it contains keywords related to the predefined tasks. These two tasks are represented as slot filling and intent detection. Intent detection is to classify the input sentences into corresponding intention types through classification. After determining the sentence's intent, the next few modules' processing scopes can be effectively reduced. Slot filling is to search for suitable words in the input sentence and fills them into a predefined semantic slot. As shown in Table 1, the user requests to add a track into the specified playlist. The intent of this sentence is to add the playlist. And the semantic slot can be filled with corresponding words according to the sentence's semantics, such as the name of the playlist can be filled as classical relaxations.

Then, the DM module can be subdivided into two sub-modules, i.e., the dialogue state tracking module and the policy learning

[6]https://github.com/cicilzx/DialTest

Table 1: An example of slot labels in the sentence with the intent "add to playlist".

Sentence	I'd	like	to	have	this	track	onto	my	classical	relaxations	playlist.
Slot Lable	O	O	O	O	O	B-music_item	O	B-playlist_owner	B-playlist	I-playlist	O

module. The function of the dialogue state tracking module is to manage each round of input and the dialogue history, then output the current dialogue state. And the dialogue policy learning module is often designed to learn the next action according to the current dialogue state. This module usually generates corresponding actions based on rules. For example, if the dialogue status is "recommended" in an online shopping scenario, then the "recommended" action will be triggered, and the system will retrieve the product from the product database. Finally, the function of the NLG module is to transform abstract dialogue actions into expressions in natural language for users to understand.

The intent detection and slot filling tasks lay the foundation of the dialogue systems' workflow and significantly impact the final behaviors. In this paper, we employ these two tasks to evaluate the performance of `DialTest`.

2.2 Testing for RNN Models

2.2.1 RNN Architecture. Understanding the dialogue's contextual information is the key to constructing a dialogue system, keeping the dialogue coherent, and enhancing the user experience. Because of the outstanding ability of RNN in processing data with sequence changes, many dialogue systems employ RNN models to drive various tasks, especially in NLU modules. The NLU module generally applies a hierarchical RNN model to capture the meaning of individual words and then integrate them into a complete sentence.

Fig. 2 illustrates a general structure of RNN. The RNN unit iteratively makes a prediction output y based on the input x and the intermediate hidden state s, referred to as the hidden state. Different from the fully connected networks, these hidden states output s_t at time step t is decided by current input x_t as well as s_{t-1} from the previous time step, and then passed forward to compute the softmax final output y_t [53]. This feature makes RNN particularly useful when a sequence of data is being processed to make a classification decision or regression estimate [32].

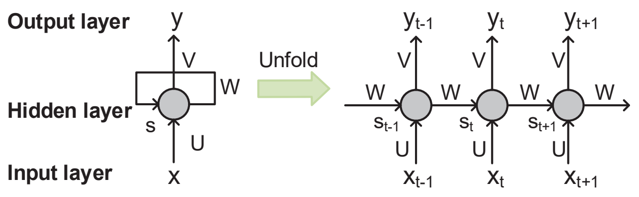

Figure 2: RNN structure.

Nevertheless, the disadvantage of RNN is that the logical structure is relatively rigid, regarding the later input has greater impacts than the earlier input. Therefore, the most widely used and most successful optimization models of RNN is the LSTM (Long Short-Term Memory) model [19], which adds a forget gate to the hidden layer to discard unimportant information, so the model can better express the long-term and short-term dependence locally [31].

2.2.2 Testing Criteria of RNN Models. Inspired by the success of employing code coverage as guidance in generating tests for conventional software systems, several recent research has proposed techniques that leverage the neuron coverage to guide the generation of adversarial inputs. Existing deep learning model testing approaches [29, 38, 45] are primarily designed for generating test cases with the guidance of neuron coverage. These approaches measure the testing adequacy based on the proportion of neurons activated by the test cases. As discussed in existing studies [13, 16], these criteria are specifically designed for the Feed-forward Neural Networks (FNN) [46], e.g., Convolution Neural Networks (CNN) [36] and fully connected neural networks. However, since the structure and function of the hidden neurons of RNN are essentially different from FNN and the length of input data is inconsistent, these coverage criteria are not applicable for RNN models.

Similar to the insight of these coverage criteria, a few techniques have been proposed for the RNN models. DeepStellar [13] specifically aims at the hidden state's structure of RNN, constructs RNN into an abstract state transition system, and designs a series of coverage indicators by supervising the internal state transition of the model. The state coverage is calculated as the proportion of the set of abstract states visited by the training inputs and the test inputs. RNN-Test [16] proposed three neuron coverage metrics to measure the testing completeness and guide the generation of adversarial inputs. These criteria represent the activation states of the hidden cells and gates in the RNN model.

Meanwhile, even though the authors of these works have demonstrated the effectiveness of coverage-guided adversarial input generation, several recent studies have presented a counter view with evidence suggesting that coverage criteria could be ineffective and even be misleading for generating tests for DNNs [14, 17, 26, 50]. Considering these controversial points and `DialTest` is designed for generating tests with transformation operators, we explore another solution, i.e., Gini impurity, to guide the generation process. Different from neuron coverage criteria, Gini impurity measures the likelihood of detecting fault for test data as the guidance to generate the transformed test sentences.

3 APPROACH

In this section, we introduce the design and implementation of `DialTest`, which presents a method to test the RNN-driven dialogue system automatically. As shown in Fig. 3, `DialTest` first applies a series of transformation approaches with corresponding operators, specially adapted to the slot filling and intent detection tasks of the NLU model. The transformation can be applied to generate new test sentences without changing the semantics of the

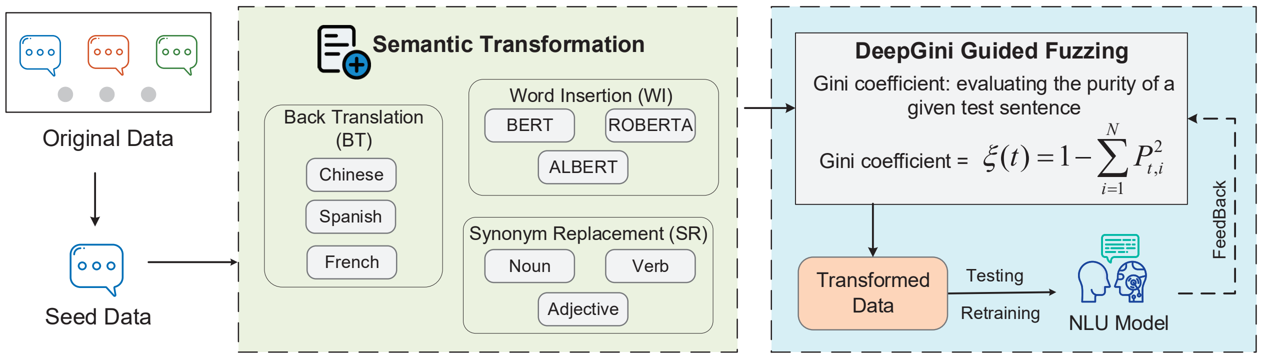

Figure 3: Overview of DialTest.

original sentence. While generating the test cases, DialTest calculates Gini impurity and then measures the uncertainty of the tested model for the test data. With Gini impurity guidance, DialTest can effectively select the test data that is most likely to trigger potential defects in the model. Finally, the generated transformed test data can be used to evaluate the robustness of the model. Besides, DialTest can improve the model's accuracy by retraining with the transformed training data.

3.1 Metamorphic Relation for RNN-Driven Dialogue Systems

Metamorphic Testing [42] is widely employed to generate tests to detect errors of many complex software systems, such as autonomous driving systems [47], machine translation [54], and so on. The most significant advantage of metamorphic testing lies in its capability of identifying oracle information for the tests via metamorphic relations (MRs) [56]. MRs describes the necessary properties of multiple executions of target programs. The violation of MRs often indicates potential errors.

DialTest is designed to realize the metamorphic testing for the RNN-driven dialogue systems. Specifically, we denote the RNN-Driven dialogue system as $\mathbb{R}$ that continuously detects the intent and fills the slots for an input sentence. The MR can be defined as given a seed sentence set $\mathbb{S}$, and transformation operators can derive a set of sentences from each sentence $s \in \mathbb{S}$ without influencing the intent and slot labels of s. This MR to test $\mathbb{R}$ with additional transformed sentences can be formalized as follows:

$$\forall s \in \mathbb{S} \wedge \forall t \in \mathbb{T}, \quad R(s) = R(t(s))$$

Given this MR, we can simply obtain the oracle information by verifying whether it is satisfied in the testing process of $\mathbb{R}$. Under this setting, the design and implementation of transformation operators become critical for the performance of DialTest.

3.2 Transformation Operators

We investigate three families of realistic sentence transformation operators, i.e., synonym replacement, back translation, and word insertion. Table 2 shows an example of the transformed sentence generated after different transformations and corresponding operators. For each of these transformation families, we implement several transformation operators, which are defined as follows:

(1) Synonym Replacement (SR): The operators of this family transform the sentence by replacing an individual word with its synonyms, which keeps the meaning of this sentence not changed. DialTest employs WordNet [33], which is a large-scale lexical database for English, to find and replace the synonyms. In WordNet, nouns, verbs, adjectives are grouped into sets of cognitive synonyms, and each set of words expresses a distinct concept [33].

(2) Back Translation (BT): The operators of this family translate the target sentence into an intermediate language and then translates it back to the original language. Because machine translation generally reorganizes the sentence structure, the transformed data not only keeps the sentence meaning unchanged but also improves the richness of sentence structures [40]. Regarding intermediate languages, we choose Chinese, French, and Spanish because they are of completely different alphabets and grammar.

(3) Word Insertion (WI): The operators of this family transform the sentence by inserting words. When the NLU model performs intent detection and slot filling, it is actually trying to understand the input sentence. Therefore, if a few words that are not related to the intent are added to the sentence while keeping the grammar correct, it may cause potential interference of the model. The language representation models, such as BERT [12], ROBERTA [28], and ALBERT [24] have been proposed for generating natural languages automatically. The Masked Language Model (MLM) task predicts the masked word based on the semantics of the context, which can be used for sentence transformation. As for the transformation operators, we chose three pre-trained language representation models with different parameter configurations to insert new words on the original sentence. Meanwhile, we control the number of words from one to three. Thus both a single word and a phrase can be inserted.

Note that the tested NLU model processes both intent detection and slot filling tasks. Particularly, detecting intents for a sentence can be considered as a sentence classification task, i.e., labeling the input sentence with a predefined intent category. Slot filling can be regarded as a sequence labeling task [43], which is to tag each element in the linear input sequence with a certain label in the label set. Table 1 illustrates an example of the labeled data, which is taken from the Snips dataset. The model outputs the labeled results for each word in the sentence, the labeling format is BIO format, i.e., each element is labeled as "B-X", "I-X" or "O". Among them, "B-X" represents that this word belongs to type X and is located at the beginning of this segment. "I-X" represents that the corresponding word belongs to type X and is located in the middle of the fragment, and "O" means that the word does not belong to any type.

When applying SR transformation, `DialTest` randomly substitutes synonyms for the non-entity words, i.e., the slot label is "O". Since the replaced word originally has no entity label, the slot and intent labels keep the same as the original data. When applying BT transformation, as the sentence is translated, there may be word substitutions and order changes, so the slot label is difficult to keep consistent with the original slot label. Therefore, `DialTest` only keeps the intent label of the transformed sentence consistent with the original label. When applying WI transformation, `DialTest` labels the inserted word's slot label as "O" and keeps the intent label as its original intent.

Table 2: Example sentences with metamorphic relations augmentation.

Transformations		Sentence
Original		I'd like to have this track onto my classical relaxations playlist.
SR	Noun	I'd like to have this track onto my classical entertainment playlist.
	Verb	I'd care to have this track onto my classical relaxations playlist.
	Adjective	I'd like to have this track onto my traditional relaxations playlist.
BT	Chinese	I want to put this song on my classic relaxation list.
	French	Add this song to classic relaxation list.
	Spanish	I want to add this song on my classic relaxation list.
WI	BERT	I'd like to have added this track onto my classical relaxations playlist.
	ROBERTA	I'd like to have this track be added onto my classical relaxations playlist.
	ALBERT	I'd like to be able to have this track onto my classical relaxations playlist.

3.3 Gini-Guided Sentence Transformation

As introduced in Section 2.2.2, we explore an alternative solution, Gini impurity, to guide the sentence transformation process. We choose DeepGini [14], which is a DNN-specific Gini impurity, to implement `DialTest`. DeepGini can be used to evaluate the quality of test cases in the classification model, where the last layer calculates the probability of the label through the softmax function and then outputs the classification result. Because the calculation of the DeepGini coefficient can only be applied to a one-dimensional softmax matrix, the slot filling task can not directly calculate the DeepGini coefficient because it outputs a two-dimensional softmax matrix. Therefore, we calculate the DeepGini coefficient for the model processing intent detection task.

DeepGini coefficient is defined as:

$$\xi(t) = 1 - \sum_{i=1}^{N} P_{t,i}^2 \tag{1}$$

where $P_{t,i}$ represents the probability that the test case t belongs to the class i.

For the softmax layer of the tested model for intent classification, the test cases leading to a high DeepGini coefficient indicate that the model is uncertain about the prediction result, so the potential defects of the model are more likely to be found.

Similar to the testing approach of traditional software-based on code coverage, `DialTest` attempts to generate a test set that can increase the DeepGini coefficient. Since our test goal is to generate more data that may cause erroneous behaviors in the model's prediction results, we employ DeepGini to guide the sentence transformation instead of neuron coverage.

3.4 Combining Transformations to Increase DeepGini Coefficient

Since a single transformation of a sentence can increase the richness of data samples, an obvious question is whether they can be combined to further improve the DeepGini coefficient of the test sample. When generating test sets, `DialTest` applies a combination of multiple transformations to increase the DeepGini coefficient of the tested model, so as to detect more erroneous behaviors of the model. However, if the combinations of all transformation approaches are applied to original test sets, a large number of test samples will be generated, resulting in low test efficiency and high storage space costs. Moreover, if the number of combinations is not limited, theoretically, the combinations are endless, leading to the uncertainty of data quality. `DialTest` provides a Gini-guided technique to generate the transformed data, which can effectively find the combination of transformations that tend to enrich the semantics of text data.

Algorithm 1 presents the process of combining the transformations to increase the DeepGini coefficient. When generating transformed data, `DialTest` applies the DeepGini function as a guide to ensure that the generated test data is more likely to find potential bugs in the system than the original test. The algorithm takes the tested model $\mathbb{R}$, and a list of sentence transformations $\mathbb{T}$ with the corresponding parameters as input, and a set of seed sentences $\mathbb{S}$. The main implementation process of this algorithm is to first transform the seed sentence by a random transformation and then calculate the DeepGini coefficient of the tested model. If the DeepGini coefficient increases to a specified threshold τ, then the generated sentence is added to the seed sentence list. On the contrary, the generated sentence is directly discarded because the DeepGini coefficient of it is small, which indicates that $\mathbb{R}$ has a high degree of confidence in this input sentence and is not prone to erroneous behaviors.

Algorithm 1: Gini-guided fuzzing

```
Input: The tested model $\mathbb{R}$, Transformations $\mathbb{T}$, Seed
       Sentences $\mathbb{S}$
Output: Synthetically generated test sentences
numAug = 0;
while $\mathbb{S}$ is not empty do
    s = $\mathbb{S}$.pop();
    originalGini = CalculateGini(s);
    numFailedTries = 0;
    $\tau$ = setThreshold();
    while numFailedTries $\leq$ maxFailedTries do
        foreach $t \in \mathbb{T}$ do
            newSentence = ApplyTransforms(s, t);
            gini = CalculateGini(newSentence);
            if gini $-$ originalGini $> \tau$ then
                S.add(newSentence);
                genTests.add(newSentence);
                numAug = numAug + 1;
            else
                numFailedTries = numFailedTries + 1;
            end
        end
    end
end
return genTests, numAug
```

4 EXPERIMENTAL DESIGN

In this section, we introduce the experimental setup, including the datasets and NLU models under-tested, approaches of sentence transformations, the industrial dialogue system, and the research questions we study in the experiments. To conduct the experiments, we implement `DialTest` upon Pytorch 1.6.0 [7] with TensorFlow 1.14.0 [8]. All experiments are performed on a Ubuntu 18.04.3 LTS server with Tesla V100-SXM2, one 10-core processor with 2.50GHz and 32GB physical memory.

`DialTest` is designed to systematically test the dialogue system, especially to verify its robustness for the NLU module. To this end, we empirically explore the following four research questions (RQ).

- RQ1: Can `DialTest` generate testing data effectively with the guidance of Gini impurity?
- RQ2: How effective are different transformation methods for detecting the erroneous behaviors of intent detection & slot filling tasks?
- RQ3: Can `DialTest` guide the retraining of an NLU module to improve its accuracy?
- RQ4: (Case Study) Can `DialTest` be applied to improve the industrial dialogue system effectively?

[7]https://pytorch.org/
[8]https://github.com/tensorflow/tensorflow

4.1 Datasets and NLU Models

Datasets. As shown in Table 3, we evaluated three popular publicly available datasets, i.e., ATIS [48], Snips [10], Facebook's multilingual dialogue corpus [9].

The ATIS dataset stands for the airline travel information system, which is a corpus in the air travel field. It is a standard benchmark dataset widely used for intent detection and slot filling tasks. ATIS dataset contains over 5,000 input data in total, of which over 4,400 are training data, 500 valid data, and over 800 test data.

The Snips dataset is collected from personal voice assistants. Compared with the single-domain ATIS data set, Snips is more complex because of its diverse intent and a larger vocabulary. There are over 13,000 training data, 700 valid data, and 700 test data.

Facebook's multilingual dialogue corpus contains labeled dialogue data in multiple languages (English/Spanish/Thai). This corpus comes from intelligent assistants and involves multiple fields, such as weather query, playing music, etc. In this paper, we only consider the English corpus and call this dataset Facebook for short. There are over 30,000 training data, over 4,000 valid data, and over 8,000 test data.

Table 3: Descriptions and statistics of datasets.

Dataset	# Intents	# Slots	# Avg. words	# Test sets
ATIS	18	130	13.13	893
Snips	7	72	9.17	700
Facebook	12	28	7.26	8621

NLU Models. To evaluate the performance of `DialTest` effectively, we choose four state-of-the-art models as the testing subjects in the experiment. We briefly introduce them as follows:

$LSTM_{ELMo}$ [44] is a LSTM model pre-trained with ELMo [39] for word embedding. The size of word embedding generated by ELMo is 1,024, and the number of LSTM hidden units is 200.

$BLSTM_{ELMo}$ [44, 51] is a BLSTM model pre-trained with ELMo for word embedding. The parameter configuration of ELMo and the hidden units is exactly the same as $LSTM_{ELMo}$.

$BLSTM_{BERT}$ [7] is a BLSTM model pre-trained with the BERT language model [12] for embedding. The pre-trained BERT model applies the configuration of bert-base-uncased [12].

$BLSTM_{CRF}$ [7] is pre-trained on the BERT language model and based on BLSTM with a conditional random field (CRF) [21] layer, which has been wildly applied in NLU model building [21, 23, 27]. The output of all BLSTM can be used as the input of the CRF layer, and the final prediction result is be obtained by learning the order dependency information between tags.

The Industrial Dialogue System. To verify whether `DialTest` is effective in the dialogue system under real usage scenario, we select an open-sourced dialogue system DeepPavlov [5] to empirically investigate the test tool proposed in this paper. The project of DeepPavlov has 4.9k stars on GitHub [10]. According to the introduction of its official documentation, DeepPavlov is designed for (1) the development of production-ready chat-bots and complex

[9]https://fb.me/multilingual_task_oriented_data
[10]https://demo.deeppavlov.ai/#/en/chat

conversational systems, and (2) research in the area of NLP and, particularly, of dialog systems. We consider DeepPavlov is consistent with our goals, and since it open-sourced the code, we can test the NLU module individually.

DeepPavlov implements a goal-oriented intelligent dialogue system containing an NLU module, processing word embedding, intent detection, and slot filling task processing. In this dialogue system, the NLU module is followed by the dialogue policy management module. In this module, an action with the highest probability is selected based on the intent and the judgment result of the entity, such as *welcome(), reques_place()*, etc. Then the system makes the final response based on the responses stored in the data template. The DeepPavlov Go-Bot Framework is trained on Dialog State Tracking Challenge 2 dataset (DSTC2) [11]. DeepPavlov evaluates the dialogue system using turn accuracy [18, 49]. For each turn in each dialogue, the turn accuracy is calculated by comparing the history output of the user and the action of dialogue systems. The turn is correct if the output of the dialogue system matches the reference answer, and it is incorrect if not.

4.2 Sentence Transformation

`DialTest` focuses on improving the robustness of the NLU model by making realistic transformations in textual expression. As described in Section 3.2, we mainly apply three transformations, i.e., synonym replacement, back transition, and word insertion.

The synonym replacement approach is implemented upon the WordNet library, which groups words according to their meanings, and each group with the same meaning is a synset. For a given sentence, `DialTest` randomly substitutes synonyms for words whose slot label is "O". Since the replaced word originally has no entity label, the slot and intent labels keep the same as the original data.

The back transition approach is to first translate English into another language and then translate back to English. `DialTest` translates the given sentence into Chinese, French, and Spanish based on Baidu translation API [12], which provides machine translation services in over 200 languages. Since some new words may be introduced during the translation process, the entity words in original sentences may also be replaced with synonyms. We only keep the intent label and discard the slot label. The back transition approach is mainly used for the verification of intent detection tasks.

The word insertion approach is implemented based on the BERT language model. For a given sentence, we first dig out space at a random location as the mask and then apply the pre-trained language models to fill in words. These language models can ensure that the inserted words or phrases conform to the grammatical rules and have smooth semantics. We label the slot label of the inserted word as "O", the other words keep their original labels, and the intent of the sentence also keeps the original labels unchanged.

4.3 Evaluation Metric

We employ the accuracy score to measure the testing performance of `DialTest` and answer research questions. Intent detection is one kind of classification tasks [9]. For each input sentence, the model outputs a classification result, which represents the intent of the sentence. Thus, we employ the widely-used metric, accuracy, to measure the performance of `DialTest`, which is defined as follows:

$$Intent\ Acc = \frac{|Correct\ Predict|}{|Total\ Sentence|}$$

Slot filling is a sequence labeling task [35]. After inputting the entire sentence containing few words $(x_1, x_2, \ldots, x_n)$, the model outputs the labeled result for each word $(O_1^S, O_2^S, \ldots, O_n^S)$. Since an entity may correspond to multiple consecutive words, only when the predicted label has correctly marked each word in this entity can it be recorded as a correct prediction. Therefore, we do not directly compare the predicted label and the actual label of each word to calculate the accuracy. The accuracy is represented as the ratio of the number of correct entities to the total number of entities:

$$Slot\ Acc = \frac{|Correct\ Entity|}{|Total\ Entity|}$$

5 DISCUSSION

5.1 Result Analysis

5.1.1 Answer to RQ1. To verify the test sentences generated by `DialTest` can efficiently and effectively detect erroneous behaviors, we compare three test data generation approaches, i.e., random transformation, neuron coverage guided generation, and Gini impurity guided generation. The neuron coverage-guided generation has been used in many existing testing approaches, which is based on the theory that the test sets with high coverage cover a large range of deep learning models. We apply the basic static coverage proposed in DeepStellar [13] to calculate the neuron coverage. To be consistent with `DialTest`'s process of generating test cases, we employ the Coverage-Total Method (CTM) [52], i.e., it keeps the test case with the highest coverage rate and discards the test cases with low coverage rate. Meanwhile, when we apply three transformations to generate test cases, we ensure that the generated test set size keeps the same as the original size.

Table 4: Testing results on different models and datasets with the testing cases generated by three approaches, i.e., random, coverage-guided, and Gini-impurity-guided.

Model	Dataset	Intetn Acc				Slot Acc			
		Ori.	Ran.	Cov	Gini	Ori.	Ran.	Cov	Gini
$LSTM_{ELMo}$	ATIS	0.96	0.84	0.89	**0.66**	0.93	0.60	0.69	**0.51**
	SNIPS	0.98	0.85	0.95	**0.61**	0.84	0.46	0.43	**0.36**
	Facebook	0.98	0.89	0.92	**0.87**	0.83	0.55	0.54	**0.51**
$BLSTM_{ELMo}$	ATIS	0.98	0.84	0.89	**0.64**	0.96	0.66	0.73	**0.61**
	SNIPS	0.98	0.89	0.97	**0.70**	0.94	0.51	0.49	**0.44**
	Facebook	0.98	0.91	0.94	**0.89**	0.92	0.61	0.60	**0.56**
$BLSTM_{BERT}$	ATIS	0.98	0.88	0.91	**0.75**	0.92	0.75	0.74	**0.71**
	SNIPS	0.98	0.91	0.99	**0.51**	0.93	**0.58**	**0.58**	0.76
	Facebook	0.99	**0.86**	0.95	0.91	0.91	0.60	0.58	**0.55**
$BLSTM_{CRF}$	ATIS	0.97	0.87	0.91	**0.74**	0.91	0.74	0.74	**0.70**
	SNIPS	0.98	0.91	0.99	**0.77**	0.94	0.58	0.58	**0.53**
	Facebook	0.99	0.93	0.95	**0.92**	0.91	0.57	0.56	**0.54**

Results. Table 4 presents testing the results on different models and datasets with the testing cases generated by three generation

[11] http://docs.deeppavlov.ai/en/master/features/skills/go_bot.html#datasets
[12] https://api.fanyi.baidu.com/

Table 5: The sample sentences showing the erroneous behaviors detected by DialTest using transformed test dataset.

Transform		Sentence	Intent
WI	original	Delay my alarm by 1 hour tomorrow.	modify alarm ✓
	transformed	Delay my alarm time by 1 hour tomorrow.	cancel reminder ✗
	original	Please remind me all the events this week.	show reminders ✓
	transformed	Please remind me of all the events this week.	set reminder ✗
BT	original	Delete all my set alarms for saturday and sundays.	cancel alarm ✓
	transformed	Delete all my clocks for Saturdays and Sundays.	cancel reminder ✗
	original	I want to book a restaurant close-by in Inman for five people.	book restaurant ✓
	transformed	I'd like to reserve a restaurant for five people near Inman.	search creative work ✗
SR	original	Set reminder for birthday party.	set reminder ✓
	transformed	Mark reminder for birthday party.	cancel reminder ✗
	original	I want to see Outcast.	search screening event ✓
	transformed	I desire to see Outcast.	get weather ✗

approaches. We separately record the *intent acc* and *slot acc* corresponding to each test set. Compared with the test results on original test sets, the accuracy of both tasks in the transformed test sets has decreased, which shows that these three generation methods can detect potential erroneous behaviors more effectively than original test sets.

Compared with the random method and the coverage-guided method, `DialTest`'s test effect on each model and data set is more obvious, which makes both *intent acc* and *slot acc* have a significant decrease, comparing with the original test set. This indicates that `DialTest` employs Gini impurity as guidance to generate test data is particularly effective. Since the Gini impurity represents the model's uncertainty to the input sentence, it is effective to generate a test set that makes the model make an erroneous behavior.

For both intent detection and slot filling tasks, the random method and the coverage-guided method lead to similar accuracy results. In some cases, the random method even outperforms the coverage-guided method in detecting bugs. When generating test cases based on coverage guidance, the algorithm opts to select the tests with high coverage. Thus, some similar test cases are likely to be selected simultaneously, resulting in relatively low bug detection capabilities. Although we retain test cases with high coverage, it is difficult to guarantee that all neurons are covered, and the coverage of the model's internal state does not directly correlate with error detection ability. Even though some recent research work [13, 16, 20] has shown that structural neuron coverage may be effective in discriminating adversarial samples of images and audios, the experimental results show that the guidance of Gini impurity is more effective in detecting incorrect behaviors in dialogue tasks.

5.1.2 *Answer to RQ2.*

For each combination of the tested model and dataset, `DialTest` applies three transformations introduced in Section 3.2 on the original test set T_O to generate test sentences. To avoid the inconsistency between the size of the generated test dataset T_A and the original test set T_O, `DialTest` generates the same size of datasets as T_O with Gini impurity guidance. Then, the transformed dataset $T_A = \{T_{A1}, T_{A2}, T_{A3}\}$ generated by three transformations $\mathbb{T} = \{\mathbb{T}_{SR}, \mathbb{T}_{BT}, \mathbb{T}_{WI}\}$ are input into the trained model M to make the model predict and calculate the intent accuracy and slot accuracy, respectively. Note that the intent accuracy of most models on the original data set exceeds 95%, and most slot accuracy exceeds 90%, i.e., only a few errors can be found on the original test set. For the test set $T_A = \{T_{A1}, T_{A2}, T_{A3}\}$ generated by `DialTest`, the drop of accuracy score indicates that `DialTest` can effectively detect the potential defects of the tested model.

Table 6: Testing results on different models and datasets with three transformations generated by DialTest.

Model	Dataset	Original		SR		WI		BT
		Intent Acc	Slot Acc	Intent Acc	Slot Acc	Intent Acc	Slot Acc	Intent Acc
$LSTM_{ELMo}$	ATIS	0.96	0.93	0.36	0.70	0.61	0.54	0.66
	SNIPS	0.98	0.84	0.50	0.46	0.65	0.25	0.38
	Facebook	0.98	0.83	0.26	0.51	0.70	0.35	0.54
$BLSTM_{ELMo}$	ATIS	0.98	0.96	0.37	0.79	0.65	0.63	0.57
	SNIPS	0.98	0.94	0.69	0.64	0.75	0.29	0.44
	Facebook	0.98	0.92	0.45	0.71	0.71	0.44	0.60
$BLSTM_{BERT}$	ATIS	0.98	0.92	0.70	0.82	0.73	0.57	0.76
	SNIPS	0.98	0.93	0.88	0.74	0.88	0.45	0.54
	Facebook	0.99	0.91	0.53	0.74	0.76	0.43	0.83
$BLSTM_{CRF}$	ATIS	0.97	0.91	0.60	0.82	0.73	0.57	0.74
	SNIPS	0.98	0.94	0.90	0.73	0.87	0.48	0.53
	Facebook	0.99	0.91	0.55	0.75	0.77	0.42	0.83

Results. Table 6 presents the results of RQ2, which shows the *Intent Acc*, and *Slot Acc* of all models for the test set T_O and $T_A = \{T_{A1}, T_{A2}, T_{A3}\}$. For the first two models in the table, i.e., the LSTM and BLSTM models that use word embedding for pre-training, the accuracy on three transformed test sets has decreased significantly. Especially on the SR test set, the intent accuracy of most models on both tasks is even lower than 50%. This is mainly because the model is not familiar with the new words after synonym replacement, which leads to erroneous behaviors. For the WI and BT test sets, new words may also be introduced, but usually, some common words are inserted or replaced in most cases, so the accuracy decrease of the models on the WI and BT test sets is slightly lighter than it on the SR test set. For the latter two models in the table, the accuracy

Table 7: The test results comparison before and after retraining the model with the transformed data generated by DialTest.

Model	Dataset	Synonym Replacement (SR)				Word Insertation (WI)				Back Translation (BT)	
		Intent Acc		Slot Acc		Intent Acc		Slot Acc		Intent Acc	
		Before	After	Before	After	Before	After	Before	After	Before	After
$LSTM_{ELMo}$	ATIS	0.36	**0.91**	0.70	**0.84**	0.61	**0.96**	0.54	**0.82**	0.66	**0.86**
	SNIPS	0.50	**0.93**	0.46	**0.73**	0.65	**0.97**	0.25	**0.67**	0.38	**0.86**
	Facebook	0.26	**0.84**	0.51	**0.72**	0.70	**0.98**	0.35	**0.7**	0.54	**0.92**
$BLSTM_{ELMo}$	ATIS	0.37	**0.91**	0.79	**0.86**	0.65	**0.96**	0.63	**0.86**	0.57	**0.87**
	SNIPS	0.69	**0.95**	0.64	**0.85**	0.75	**0.97**	0.29	**0.74**	0.44	**0.86**
	Facebook	0.45	**0.89**	0.71	**0.83**	0.71	**0.98**	0.44	**0.78**	0.60	**0.94**
$BLSTM_{BERT}$	ATIS	0.70	**0.95**	0.82	**0.89**	0.73	**0.98**	0.57	**0.87**	0.76	**0.87**
	SNIPS	0.88	**0.97**	0.74	**0.87**	0.88	**0.98**	0.45	**0.8**	0.54	**0.87**
	Facebook	0.53	**0.9**	0.74	**0.86**	0.76	**0.97**	0.43	**0.81**	0.83	**0.92**
$BLSTM_{CRF}$	ATIS	0.60	**0.94**	0.82	**0.89**	0.73	**0.97**	0.57	**0.86**	0.74	**0.88**
	SNIPS	0.90	**0.97**	0.73	**0.88**	0.87	**0.97**	0.48	**0.80**	0.53	**0.89**
	Facebook	0.55	**0.89**	0.75	**0.86**	0.77	**0.98**	0.42	**0.81**	0.83	**0.92**

of the model does not decrease as much as the first two models. This is mainly because the pre-trained language models are more robust than the word embedding pre-trained models.

To prove the effectiveness of `DialTest` in detecting erroneous behaviors, we intercepted some errors found by `DialTest`, as shown in Table 5. For each transformation method, we list two real examples of erroneous predictions in the tested model. As shown in the table, the transformed sentence has the same semantics as the original sentence, and the transformed sentence keeps the same label as the original sentence. Although only a little modification is added to the original sentence, the intent of the sentence predicted by the model is completely changed.

5.1.3 *Answer to RQ3.*

For each combination of the tested models and datasets, three transformations $\mathbb{T} = \{\mathbb{T}_{SR}, \mathbb{T}_{BT}, \mathbb{T}_{WI}\}$ are applied on original training dataset to generate the transformed data for model retraining. `DialTest` applies the Gini impurity-based transformation on the tested model with original training data as seed data to generate new data for retraining. Since the models already have very high accuracy on the original testing set (> 90% or even > 95%), we cannot clearly show the accuracy improvement of these NLU models. Thus, we leverage the transformed testing data to answer RQ3. Specifically, we use the three transformation methods of `DialTest` to generate a transformed test set with the same size as the original test set.

Results. Table 7 presents the *Intent Acc* and *Slot Acc* of all models after retraining with the transformed data generated by `DialTest`. We keep the experimental configuration of retraining consistent with the original training model process, and the retraining epoch is set as 30. The test result shown in the table is that we generated 50% size of the original training set as transformed data for retraining. If the size of the retraining data set is too large, the time cost will be extremely high, while a small amount of data used for retraining may effectively improve the performance of the model. Thus, the quality of the retraining data is particularly important. Our experiments show that using 10% size of the original data set as transformed data for retraining can significantly improve the accuracy of the model on the intent detection task. When the retraining data size reaches 50%, the accuracy on both tasks has been greatly improved. Compared with the intent accuracy after training, i.e., basically over 90%, the accuracy of the slot filling task is not so high, which mainly concentrated in 85%. This is because, on the one hand, the slot filling task needs to label each word in the sentence, and its complexity is greater than intent detection. Therefore, the models generally perform better on intent detection than on slot filling tasks before or after retraining. On the other hand, the accuracy of slot accuracy before retraining is relatively low. Thus its accuracy is significantly increased.

5.1.4 *Answer to RQ4.*

In the case study, we apply `DialTest` on DeepPavlov to evaluate its performance in the industrial dialogue system. In the evaluation, we calculate the accuracy of intent detection and slot filling, respectively. When evaluating DeepPavlov's goal-oriented dialogue system, we calculate the accuracy of intent detection, slot filling, and turn accuracy, respectively. Meanwhile, we also generate the transformed training data for retraining the deep learning model in DeepPavlov.

Table 8: The testing turn accuracy of the original model and the retrained model on the original test set and the transformed test set, respectively.

	Original model	Retrained model
Original Test Set	0.5226	0.6142
Transformed Test Set	0.399	0.6495

Table 8 shows the accuracy of the original model and the retrained model on the original test set and the transformed test set, respectively. The experimental results indicate that `DialTest` can detect more erroneous behaviors than the original test set, and the accuracy of the dialogue system can be improved by retraining with the transformed data.

To show that the retrained model can improve the dialogue system's performance, we intercepted the chat records with the dialogue system, and the test input sentence is completely consistent,

Table 9: Two dialogue examples using the model before and after retraining. The incorrect outputs of the dialogue system are displayed in blue font.

Before retraining	After retraining
User:: I'm looking for an expensive restaurant in the east part of town. Bot:: Royal standard is a nice place in the east of town and the prices are expensive. User:: Do you have anything else? Bot:: Royal standard is a nice place in the east of town and the prices are expensive.	User:: I'm looking for an expensive restaurant in the east part of town. Bot:: Royal standard is a nice place in the east of town and the prices are expensive. User:: Do you have anything else? Bot:: Yu garden is a nice place in the east of town serving tasty Chinese food.
Bot:: Royal spice is a nice place in the north of town serving tasty Indian food. User:: Where is it? Bot:: Royal spice is a nice place in the north of town serving tasty Indian food.	Bot:: Royal spice is a nice place in the north of town serving tasty Indian food. User:: Where is it? Bot:: Sure, royal spice is on Victoria avenue Chesterton.

as shown in Table 9. In the dialogue, the statement at the beginning with "User::" is the sentence entered by the end-user, i.e., the test input, and the statement at the beginning with "Bot::" is the feedback of the dialogue system. To clarify the differences between before- and after- retraining the dialogue system, we highlight the different responses with blue (before retraining) and red (after retraining) colors. We can clearly identify that the response with blue color failed to answer users' requests correctly, i.e., erroneous behavior of the dialogue system. Specifically, for the first example, the user enters a demand first and wants to find a restaurant with a specific requirement. Both the dialogue system before and after retrained gives a corresponding response. Then, when the user asks if there is anything else, the original dialogue system repeats the previous answer, which is not useful for the user, while the retrained system gives a new restaurant as the answer.

As for the second example, after the dialogue system recommends a restaurant, the user asks where the restaurant is, and the original dialogue system repeats the restaurant's description. However, the trained system gives the correct address information, which performs more intelligently than the original system. The reason for this phenomenon in the example is mainly caused by the sentences in the training data being monotonous. For example, in the original training set, the sentence asking for an address is usually represented as "Give me the address.". Therefore, the model is unfamiliar with other sentences expressing this kind of intent. Since we have transformed the original training set sentences during retraining, there are more sentence pattern changes or the substitution of synonyms for the same intent. Therefore, after retraining by using the data generated by `DialTest`, the performance of the dialogue system is improved.

5.2 Threats to Validity

Dataset Selection. The dataset selection is one of the primary threats to validity. `DialTest` generates realistic transformed sentences by applying different textual transformations to the seed data of the original dataset. While we have adopted a variety of contextual transformation operators and greatly enriched the dataset, the original dataset could fundamentally influence the quality of the transformed data. Therefore, for input sentences with semantics that do not exist in the training set, the retrained model may not predict correctly. However, the datasets employed in our experiment are widely used in the training of industrial dialogue systems. This may reduce the threat to some extent.

Natural Language Selection. One threat to validity lays on the complexity of natural languages. All datasets employed in our experiments are English corpora, which could threaten the generalizability to other natural languages. Different natural languages are of various grammar rules and have their own inherent characteristics. This paper focuses on testing dialogue systems for English language environments. However, both the transformation operators as well as the Gini-impurity-based guidance are not limited to be applied in English. We believe only some minor fine-tuning or adjustment is required for applying `DialTest` on the dialogue systems of other languages.

The Testing Functionality Selection. Another threat to validity comes from the testing functionality of dialogue systems. While a general dialogue system often includes not only NLU modules but also NLG and other processing modules, this paper mainly evaluated the NLU module. Even though the experimental results show that transformed data is effective in detecting the incorrect behaviors and improve the performance of intent detection and slot filling tasks, it is difficult to guarantee `DialTest`'s capability of improving the effectiveness of the whole dialogue system because we have not investigated the performance of other modules. To alleviate this threat, we tested and retrained an industrial dialogue system in the case study, and the results showed that our proposed tool is promising in improving the application of the dialogue system. In our future work, we plan to propose testing methods for other modules of the dialogue system.

6 RELATED WORK

In this section, we compare our work with other testing approaches for dialogue systems and recurrent neural network models. We primarily focus on two perspectives: 1. the testing of dialogue systems; and 2. the testing of recurrent neural networks.

6.1 The Testing of Dialogue Systems

Bottester [3] designs and implements a testing tool for chatbots. It monitors multi-dimensional indicators such as the mean size of answers, response time per question and then generates corresponding chart displays. Bozic et al. [3] propose a testing approach based on AI planning to verify the communication capabilities of chatbots. A case study on a tourism chatbot for booking hotels is conducted to evaluate the proposed approach. Another black-box testing approach with metamorphic relations [4] generates test sentences by filling out pre-defined templates with parameters, such as actions, objects, and numbers.

The key differences between the related work mentioned above and DialTest are the following aspects: (i) These related works are the black-box testing approach for dialogue systems, depending on the application scenarios of the system under test, while our gray-box testing approach can analyze the neural networks and generate test data by systematically maximizes the Gini impurity; (ii) The transformation approach in DialTest for generating test samples is specifically aimed at slot filling and intent detection tasks in the dialogue system, while the data generated in other related words lack the pertinence of task characteristics, and (iii) DialTest can generate massive labeled data for retraining and improving the RNN-driven dialogue systems, while aforementioned approaches can generate limited data to test these systems.

6.2 The Testing of Recurrent Neural Networks

For the software systems based on deep learning models, coverage-based testing approaches [29, 30, 38] have become a trend. DeepXplore [38] first introduces neuron coverage, which is defined over CNN neurons with pre-defined thresholds. Then, DeepGauge [29] defines a set of coverage metrics with fine-grained granularity, where neuron value ranges are split as thousands of sections according to training data. As illustrated in Section 2.2, these neuron-based coverage metrics can not be applied to RNN states directly.

DeepStellar [13] adapts five coverage criteria of DeepGauge [29] to test and analyze RNN models, which is first transformed into a Discrete-Time Markov Chain (DTMC) as an abstraction. As described in Section 5.1.1, we apply the basic coverage criterion proposed in DeepStellar to guide the transformed sentences' selection for comparison with DialTest. TestRNN [20] proposes a series of neuron coverage metrics of the LSTM network and develops a coverage-guided fuzzing approach for deep learning models with LSTM network structure. Then, some random mutation enhanced with the coverage is designed to generate test cases. Different from TestRNN, DialTest conducts systematic tests based on RNN models for natural language understanding tasks, while the coverage measurement of TestRNN only applies to LSTM models.

As we discussed in Section 2.2.2, plenty of recent researches [14, 17, 50] have discussed limitations and problems of employing neuron coverage criteria to guide the generation of adversarial examples. Different from the above-mentioned works that leverage neuron coverage as the guidance to generate the adversarial examples, DialTest explores another solution to guide the test data generation process.

7 CONCLUSION

In this paper, we proposed and evaluated DialTest, the first automated testing tool for RNN-driven dialogue systems. To enhance the abundance of test samples and detect more erroneous behaviors of the tested models, DialTest applies three realistic transformation approaches, i.e., synonym replacement, back translation, and word insertion. DialTest generates the transformed sentences with Gini impurity guidance, which improves the efficiency of detecting erroneous behaviors.

DialTest has been evaluated on four state-of-art NLU models and three widely-used datasets. The experiment results show that the transformed sentences generated by DialTest can detect erroneous behaviors efficiently. Besides, the results of a case study conducted on an industrial dialogue system show that DialTest can effectively detect erroneous behaviors of the dialogue system and improve its effectiveness through retraining. We believe DialTest moves the first yet critical step towards assuring the quality of RNN-driven dialogue systems.

ACKNOWLEDGEMENT

We would like to thank anonymous reviewers for their insightful comments. This project was partially funded by the National Natural Science Foundation of China under Grant Nos. 62002158, 61832009, and 61932012.

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).

[2] Tom Bocklisch, Joey Faulkner, Nick Pawlowski, and Alan Nichol. 2017. Rasa: Open Source Language Understanding and Dialogue Management. arXiv:1712.05181 [cs.CL]

[3] Josip Bozic, Oliver A Tazl, and Franz Wotawa. 2019. Chatbot testing using AI planning. In *2019 IEEE International Conference On Artificial Intelligence Testing (AITest)*. IEEE, 37–44. https://doi.org/10.1109/AITest.2019.00-10

[4] Josip Bozic and Franz Wotawa. 2019. Testing Chatbots Using Metamorphic Relations. In *IFIP International Conference on Testing Software and Systems*. Springer, 41–55. https://doi.org/10.1007/978-3-030-31280-0_3

[5] Mikhail Burtsev, Alexander Seliverstov, Rafael Airapetyan, Mikhail Arkhipov, Dilyara Baymurzina, Nickolay Bushkov, Olga Gureenkova, Taras Khakhulin, Yurii Kuratov, Denis Kuznetsov, et al. 2018. Deeppavlov: Open-source library for dialogue systems. In *Proceedings of ACL 2018, System Demonstrations*. 122–127. https://doi.org/10.18653/v1/P18-4021

[6] Hongshen Chen, Xiaorui Liu, Dawei Yin, and Jiliang Tang. 2017. A survey on dialogue systems: Recent advances and new frontiers. *Acm Sigkdd Explorations Newsletter* 19, 2 (2017), 25–35. https://doi.org/10.1145/3166054.3166058

[7] Qian Chen, Zhu Zhuo, and Wen Wang. 2019. Bert for joint intent classification and slot filling. *arXiv preprint arXiv:1902.10909* (2019).

[8] KR Chowdhary. 2020. Natural language processing. In *Fundamentals of Artificial Intelligence*. Springer, 603–649. https://doi.org/10.1002/aris.1440370103

[9] Gobinda G Chowdhury. 2003. Natural language processing. *Annual review of information science and technology* 37, 1 (2003), 51–89. https://doi.org/10.1002/aris.1440370103

[10] Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, et al. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *arXiv preprint arXiv:1805.10190* (2018).

[11] Jan Deriu, Alvaro Rodrigo, Arantxa Otegi, Guillermo Echegoyen, Sophie Rosset, Eneko Agirre, and Mark Cieliebak. 2020. Survey on evaluation methods for dialogue systems. *Artificial Intelligence Review* (2020), 1–56. https://doi.org/10.1007/s10462-020-09866-x

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[13] Xiaoning Du, Xiaofei Xie, Yi Li, Lei Ma, Yang Liu, and Jianjun Zhao. 2019. Deepstellar: Model-based quantitative analysis of stateful deep learning systems. In

Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. 477–487. https://doi.org/10.1145/3338906.3338954

[14] Yang Feng, Qingkai Shi, Xinyu Gao, Jun Wan, Chunrong Fang, and Zhenyu Chen. 2020. DeepGini: prioritizing massive tests to enhance the robustness of deep neural networks. In *Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis*. 177–188. https://doi.org/10.1145/3395363.3397357

[15] Karthik Gopalakrishnan, Behnam Hedayatnia, Qinglang Chen, Anna Gottardi, Sanjeev Kwatra, Anu Venkatesh, Raefer Gabriel, Dilek Hakkani-Tür, and Amazon Alexa AI. 2019. Topical-Chat: Towards Knowledge-Grounded Open-Domain Conversations.. In *INTERSPEECH*. 1891–1895. https://doi.org/10.21437/Interspeech.2019-3079

[16] Jianmin Guo, Yue Zhao, Xueying Han, Yu Jiang, and Jiaguang Sun. 2019. Rnn-test: Adversarial testing framework for recurrent neural network systems. *arXiv preprint arXiv:1911.06155* (2019).

[17] Fabrice Harel-Canada, Lingxiao Wang, Muhammad Ali Gulzar, Quanquan Gu, and Miryung Kim. 2020. Is neuron coverage a meaningful measure for testing deep neural networks?. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 851–862. https://doi.org/10.1145/3368089.3409754

[18] Matthew Henderson, Blaise Thomson, and Jason D Williams. 2014. The third dialog state tracking challenge. In *2014 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 324–329. https://doi.org/10.1109/SLT.2014.7078595

[19] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[20] Wei Huang, Youcheng Sun, James Sharp, Wenjie Ruan, Jie Meng, and Xiaowei Huang. 2019. Coverage Guided Testing for Recurrent Neural Networks. *arXiv preprint arXiv:1911.01952* (2019). https://doi.org/10.1109/tr.2021.3080664

[21] Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991* (2015).

[22] Veton Kepuska and Gamal Bohouta. 2018. Next-generation of virtual personal assistants (microsoft cortana, apple siri, amazon alexa and google home). In *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE, 99–103. https://doi.org/10.1109/CCWC.2018.8301638

[23] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360* (2016).

[24] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942* (2019).

[25] Hugo Larochelle, Yoshua Bengio, Jérôme Louradour, and Pascal Lamblin. 2009. Exploring strategies for training deep neural networks. *Journal of machine learning research* 10, 1 (2009).

[26] Zenan Li, Xiaoxing Ma, Chang Xu, and Chun Cao. 2019. Structural coverage criteria for neural networks could be misleading. In *2019 IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*. IEEE, 89–92. https://doi.org/10.1109/ICSE-NIER.2019.00031

[27] Jie Liu, Shaowei Chen, Zhicheng He, and Huipeng Chen. 2018. Learning BLSTM-CRF with Multi-channel Attribute Embedding for Medical Information Extraction. In *CCF International Conference on Natural Language Processing and Chinese Computing*. Springer, 196–208. https://doi.org/10.1007/978-3-319-99495-6_17

[28] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* (2019).

[29] Lei Ma, Felix Juefei-Xu, Fuyuan Zhang, Jiyuan Sun, Minhui Xue, Bo Li, Chunyang Chen, Ting Su, Li Li, Yang Liu, et al. 2018. Deepgauge: Multi-granularity testing criteria for deep learning systems. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*. 120–131. https://doi.org/10.1145/3238147.3238202

[30] Lei Ma, Fuyuan Zhang, Jiyuan Sun, Minhui Xue, Bo Li, Felix Juefei-Xu, Chao Xie, Li Li, Yang Liu, Jianjun Zhao, et al. 2018. Deepmutation: Mutation testing of deep learning systems. In *2018 IEEE 29th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 100–111. https://doi.org/10.1109/ISSRE.2018.00021

[31] Pankaj Malhotra, Lovekesh Vig, Gautam Shroff, and Puneet Agarwal. 2015. Long short term memory networks for anomaly detection in time series. In *Proceedings*, Vol. 89. Presses universitaires de Louvain, 89–94.

[32] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*.

[33] George A Miller. 1995. WordNet: a lexical database for English. *Commun. ACM* 38, 11 (1995), 39–41. https://doi.org/10.1145/219717.219748

[34] Ali Bou Nassif, Ismail Shahin, Imtinan Attili, Mohammad Azzeh, and Khaled Shaalan. 2019. Speech Recognition Using Deep Neural Networks: A Systematic Review. *IEEE Access* 7 (2019), 19143–19165. https://doi.org/10.1109/ACCESS.2019.2896880

[35] Nam Nguyen and Yunsong Guo. 2007. Comparisons of sequence labeling algorithms and extensions. In *Proceedings of the 24th international conference on Machine learning*. 681–688. https://doi.org/10.1145/1273496.1273582

[36] Keiron O'Shea and Ryan Nash. 2015. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458* (2015).

[37] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *International conference on machine learning*. PMLR, 1310–1318.

[38] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. 2017. Deepxplore: Automated whitebox testing of deep learning systems. In *proceedings of the 26th Symposium on Operating Systems Principles*. 1–18. https://doi.org/10.1145/3361566

[39] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365* (2018).

[40] Shrimai Prabhumoye, Yulia Tsvetkov, Ruslan Salakhutdinov, and Alan W Black. 2018. Style transfer through back-translation. *arXiv preprint arXiv:1804.09000* (2018).

[41] J. Ross Quinlan. 1986. Induction of decision trees. *Machine learning* 1, 1 (1986), 81–106. https://doi.org/10.1007/BF00116251

[42] Sergio Segura, Gordon Fraser, Ana B Sanchez, and Antonio Ruiz-Cortés. 2016. A survey on metamorphic testing. *IEEE Transactions on software engineering* 42, 9 (2016), 805–824. https://doi.org/10.1109/TSE.2016.2532875

[43] Burr Settles and Mark Craven. 2008. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*. 1070–1079.

[44] Aditya Siddhant, Anuj Goyal, and Angeliki Metallinou. 2019. Unsupervised transfer learning for spoken language understanding in intelligent agents. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 4959–4966. https://doi.org/10.1609/aaai.v33i01.33014959

[45] Youcheng Sun, Xiaowei Huang, Daniel Kroening, James Sharp, Matthew Hill, and Rob Ashmore. 2018. Testing deep neural networks. *arXiv preprint arXiv:1803.04792* (2018).

[46] Daniel Svozil, Vladimir Kvasnicka, and Jiri Pospichal. 1997. Introduction to multi-layer feed-forward neural networks. *Chemometrics and intelligent laboratory systems* 39, 1 (1997), 43–62. https://doi.org/10.1016/S0169-7439(97)00061-0

[47] Yuchi Tian, Kexin Pei, Suman Jana, and Baishakhi Ray. 2018. Deeptest: Automated testing of deep-neural-network-driven autonomous cars. In *Proceedings of the 40th international conference on software engineering*. 303–314. https://doi.org/10.1145/3180155.3180220

[48] Gokhan Tur, Dilek Hakkani-Tür, and Larry Heck. 2010. What is left to be understood in ATIS?. In *2010 IEEE Spoken Language Technology Workshop*. IEEE, 19–24. https://doi.org/10.1109/SLT.2010.5700816

[49] Jason D Williams, Kavosh Asadi, and Geoffrey Zweig. 2017. Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning. *arXiv preprint arXiv:1702.03274* (2017).

[50] Shenao Yan, Guanhong Tao, Xuwei Liu, Juan Zhai, Shiqing Ma, Lei Xu, and Xiangyu Zhang. 2020. Correlations between deep neural network model coverage criteria and model quality. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 775–787. https://doi.org/10.1145/3368089.3409671

[51] Jie Yang and Yue Zhang. 2018. NCRF++: An Open-source Neural Sequence Labeling Toolkit. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*. https://doi.org/10.18653/v1/P18-4013

[52] Shin Yoo and Mark Harman. 2012. Regression testing minimization, selection and prioritization: a survey. *Software testing, verification and reliability* 22, 2 (2012), 67–120. https://doi.org/10.1002/stv.430

[53] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329* (2014).

[54] Wujie Zheng, Wenyu Wang, Dian Liu, Changrong Zhang, Qinsong Zeng, Yuetang Deng, Wei Yang, Pinjia He, and Tao Xie. 2019. Testing untestable neural machine translation: An industrial case. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*. IEEE, 314–315. https://doi.org/10.1109/ICSE-Companion.2019.00131

[55] Li Zhou, Jianfeng Gao, Di Li, and Heung-Yeung Shum. 2020. The design and implementation of xiaoice, an empathetic social chatbot. *Computational Linguistics* 46, 1 (2020), 53–93. https://doi.org/10.1162/coli_a_00368

[56] Zhi Quan Zhou, DH Huang, TH Tse, Zongyuan Yang, Haitao Huang, and TY Chen. 2004. Metamorphic testing and its applications. In *Proceedings of the 8th International Symposium on Future Software Technology (ISFST 2004)*. Software Engineers Association Xian, China, 346–351.