S^2LCT: Specification- and Syntax-based Automated Testing Linguistic Capabilities of NLP Models

Anonymous Author(s)

ABSTRACT

Natural language processing (NLP) technique has become one of the core techniques for developing text analytic applications. These applications need to achieve high reliability to be useful in practice. The trustworthiness of the NLP applications is often obtained by measuring the accuracy of the applications on holdout datasets. However, evaluating NLP on the accuracy of the holdout datasets is limited in validating its overall quality because these datasets are often not comprehensive. To address this limitation, evaluating an NLP model on task-specific behaviors defined on linguistic capabilities has been introduced. However, such evaluation relies on manually created test cases, and is still limited to measure the model performance on limited datasets. In this work, we introduce S^2LCT, an NLP model testing infrastructure. Given a linguistic capability that users want to evaluate for a NLP model, S^2LCT finds suitable seed inputs from existing datasets, generates sufficient number of new test inputs by expanding the seed inputs based on their context-free grammar (CFG). We evaluate S^2LCT by showing that it can generate test cases that are more diverse than an existing approach. We also show that S^2LCT facilitates identification of critical failures and its origins in the NLP models for the sentiment analysis task.

ACM Reference Format:
Anonymous Author(s). 2022. S^2LCT: Specification- and Syntax-based Automated Testing Linguistic Capabilities of NLP Models. In *Proceedings of IEEE/ACM International Conference on Automated Software Engineering (ASE 2022).* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Natural language processing (NLP) applications are growing exponentially. As a result, trustworthiness in the quality of NLP applications has become critical for its practical use in the real world. Therefore, quality assurance of NLP applications is essential in the software development process. Researchers aim to improve the current practices of testing NLP models from three perspectives: (i) *test input generation*, (ii) *automated test oracle*, and (iii) *meaningful quality metrics*.

Test input generation. Currently, most testing of an NLP model uses existing, large textual corpus as the testing dataset to evaluate the model. This practice often overestimates the model performances from the holdout dataset [18, 20, 25]. The overestimation comes from the discrepancy between the distribution of the used dataset and the actual data distribution in real world. Oftentimes, the holdout dataset is not representative and is likely to introduce specific biases, leading to the decreased robustness of NLP models. In this regard, prior works have proposed techniques for testing the robustness of NLP models by crafting adversarial examples and attacking a model with them intentionally (i.e., adversarial testing) [2, 11, 24, 27].

Automated test oracle. The current testing practice requires manual efforts to label the test oracles of the holdout data. The manual work is costly in terms of time consumption and its impact on the market price. Therefore, it necessitates automated test oracle generation for improving the testing process of NLP models. However, automatically generating test oracles may not always be feasible; predicting the correct test oracles remains one of main challenges. Along with this, motivated by software metamorphic testing approach [29], we address the challenge by specifying linguistic rules determined by input-output relations, and we only accept validated test oracles by the rules.

Meaningful quality metrics. Traditionally, the quality of NLP models are represented by numbers in quality metrics. Especially, accuracy (i.e., the fraction of outputs that the model correctly predicts) is the most widely used metric for assessing the quality of classification models. Generally, higher accuracy number suggests better quality of a model. However, all NLP models have their strengths and weaknesses, and forcing aggregation statistics into a single number (i.e., accuracy) makes it difficult for the users to assess the capabilities of NLP models. Not to mention localizing and fixing the bugs found from the holdout set (i.e., testing dataset, as opposed to training dataset). Therefore, this forced aggregation method not only fails to validate the linguistic capability of the model, but it also makes the localization of the causes of the inaccuracy more costly [34]. To address this limitation, Ribeiro et al. introduced CHECKLIST, a behavioral testing framework for evaluating NLP model on multiple linguistic capabilities [25]. CHECKLIST defines task-relevant linguistic capabilities and generates test cases for each linguistic capability.

However, none of the above approaches satisfy all three requirements at the same time. First, the adversarial testing approaches merely focus on evaluating model robustness. They measure how sensitive the models are to input perturbations while do not evaluate linguistic functionalities. Second, the metamorphic testing approach requires understanding the characteristics of metamorphic relations between inputs and outputs. However, finding these remains one of the most challenging problems [29]. In addition, metamorphic relation in textual data in the NLP domain has not been explored despite its importance. Third, CHECKLIST relies on manually generated input templates, which need to be preset before test input generation. Consequently, CHECKLIST templates are distributed in a limited range of their structures. This restricts

ASE 2022, 10-14 October, 2022, Ann Arbor, Michigan, United States
© 2022 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

CHECKLIST's ability to comprehensively test the linguistic capabilities.

Despite CHECKLIST's limitations, assessing the quality of NLP models through the linguistic capabilities is a promising direction. Each linguistic capability explains the functionality of the input and output behavior for the NLP model under test. Typically, it describes certain type of inputs and outputs observed in real world for the target NLP task, ranging from simple to complicated behaviors. Testing the linguistic capabilities allows the model developers to better understand the capabilities and potential issues of the NLP models. For example, a linguistic capability of "Negated neutral should still be neutral" measures how accurately the sentiment analysis model understands that the negated neutral input has neutral sentiment [25], requiring the sentiment analysis model to output neutral sentiment on the negated neutral inputs. Such evaluation methodology on the specified functionalities avoids the overestimation of the model performance as it measures the model performance on each functionality. In the end, testing through linguistic capabilities provides not only the overall model performance, but also the malfunction facets of the model.

To satisfy all three requirements mentioned above, we present S^{2}LCT, an automated evaluation method for comprehensive behavioral testing of NLP models on sentiment analysis task. There are three main challenges that S^{2}LCT overcomes to satisfy all three aforementioned requirements.

C1 The test suite should cover diverse syntactic structures;
C2 Each test case should be automatically categorized into a linguistic capability;
C3 The label of each test case should be automatically and accurately defined.

C1. The first challenge comes with the second and third challenges of maintaining the linguistic capability and label of the test sentence while making the syntactic structure of the sentence diverse. To address this challenge, S^{2}LCT establishes specifications for evaluating a linguistic capability and searches suitable sentences that satisfy the specification from existing public dataset. In this process, S^{2}LCT generates new inputs by mutating the searched sentences, used as seed inputs. S^{2}LCT expands the grammar structure of a seed input and determines its available part-of-speech to maintain structural naturalness.
C2. Suitability of test sentences for evaluating NLP model on a linguistic capability is determined by its relevancy to the linguistic capability. Relevancy between a test sentence and its linguistic capability is challenging to be maintained when transforming and expanding the test sentence. This is because the linguistic capability is defined on a specific mixture of syntax and semantics of the sentence, and there exists no automatic way to check the consistency of each sentence with the semantics specified in the corresponding linguistic capability due to the inherent ambiguity of natural language sentences. To address this difficulty, S^{2}LCT implements search rules and the transformation templates of linguistic capabilities. In addition, it analyzes the parse tree of the seed sentence to identify possible expansion.
C3. Last challenge is on automated test oracle. For NLP tasks, test oracle is determined by understanding the meaning of texts. This requires domain knowledge of different NLP tasks. In this work, as a first step, we consider sentiment analysis as the NLP task for the models under test. S^{2}LCT obtains the appropriate test oracle by implementing domain-specific knowledge on the word sentiment dataset and word suggestion model for validation of generated sentence and its oracle.

We demonstrate the generality of S^{2}LCT and its utility as a NLP model evaluation tool by evaluating three well-known sentiment analysis models: BERT-base [5], RoBERTa-base [15] and DistilBERT-base [28].

We made the following contributions in this work:

- We design and implement an automated testing tool, S^{2}LCT, and compared it with CHECKLIST. We find that the test cases generated by S^{2}LCT achieves higher coverage than CHECKLIST when used for testing sentiment analysis models.
- We perform a manual study to measure the correctness of the sentiment labels and their linguistic capabilities produced by S^{2}LCT. We find that S^{2}LCT generates test cases that consistently label their sentiment correctly as human understanding.
- We analyze the root causes of the misclassification in the sentiment analysis models, guided by S^{2}LCT testing result. We find that seed and expanded test cases, produced by S^{2}LCT, are usefulness of S^{2}LCT for understanding bugs in the models.

2 BACKGROUND

In this section, we provide a brief background of CHECKLIST's approach of test case generation via an example. CHECKLIST introduces task-dependent linguistic capabilities for monitoring model performance on each linguistic capability. It assumes that the linguistic phenomena can be represented in the behavior of the model under test, because each linguistic capability specifies the desired behavior between inputs and outputs. For each linguistic capability, CHECKLIST creates test case templates and generates sentences by filling in the values of each placeholder.

We show an example of CHECKLIST templates in Figure 1. These templates are used for evaluating a sentiment analysis model on the linguistic capability of "Sentiment change over time, present should prevail". For the templates defined from lines 22 to 33, they have placeholders such as *it*, *air_noun*, *pos_adj*. Values for the placeholders are defined at line 1, 6 and 23, . For each template, CHECKLIST fills in all the combinations of the values of placeholders, and the sentences such as "The flight is good", "That airline was happy" are generated. These sentences are used for the evaluation of the linguistic capability.

Despite its simplicity of test case generation, CHECKLIST has limitations. First, it relies on manual work for defining template structure and its placeholder values making the test case generation a costly process. Moreover, such manual work generates the limited number of templates and produces high similarity between test cases, and induces bias in the test cases. These limitations motivated the design of our approach.

3 SPECIFICATION- AND SYNTAX-BASED LINGUISTIC CAPABILITY TESTING

We design and implement a new NLP model testing method, *Specification- and Syntax-based Linguistic Capability Testing (S^2LCT)*, that automatically generates test cases with oracles to assess the quality of sentiment analysis models. S^2LCT addresses all three challenges discussed in Section 1.

Figure 2 shows the overview of S^2LCT, which consists of two phases. The *specification-based seed generation* phase performs rule-based searches from a real-world dataset and template-based transformation to obtain the initial seed sentences. The search rules (e.g., search for neutral sentences that do not include any positive or negative words) and transformation templates (e.g., negating a sentence) are defined in the *linguistic capability specifications* (Table 1), which result in seed sentences that are most likely to conform to a specific linguistic capability (**C2**) and is labelled correctly (**C3**).

```
air_noun = [
    'flight', 'seat', 'pilot', 'staff',
    'service', 'customer service', 'aircraft', 'plane',
    'food', 'cabin crew', 'company', 'airline', 'crew'
]
pos_adj = [
    'good', 'great', 'excellent', 'amazing',
    'extraordinary', 'beautiful', 'fantastic', 'nice',
    'incredible', 'exceptional', 'awesome', 'perfect',
    'fun', 'happy', 'adorable', 'brilliant', 'exciting',
    'sweet', 'wonderful'
]
neg_adj = [
    'awful', 'bad', 'horrible', 'weird',
    'rough', 'lousy', 'unhappy', 'average',
    'difficult', 'poor', 'sad', 'frustrating',
    'hard', 'lame', 'nasty', 'annoying', 'boring',
    'creepy', 'dreadful', 'ridiculous', 'terrible',
    'ugly', 'unpleasant'
]

t = editor.template('{it} {air_noun} {be} {pos_adj}.',
                    it=['The', 'This', 'That'], be=['is', 'was'], labels
                        =2, save=True)
t += editor.template('{it} {be} {a:pos_adj} {air_noun}.',
                     it=['It', 'This', 'That'], be=['is', 'was'], labels=2,
                         save=True)
t += editor.template('{i} {pos_verb} {the} {air_noun}.',
                     i=['I', 'We'], the=['this', 'that', 'the'], labels=2,
                         save=True)
t += editor.template('{it} {air_noun} {be} {neg_adj}.',
                     it=['That', 'This', 'The'], be=['is', 'was'], labels
                         =0, save=True)
t += editor.template('{it} {be} {a:neg_adj} {air_noun}.',
                     it=['It', 'This', 'That'], be=['is', 'was'], labels=0,
                         save=True)
t += editor.template('{i} {neg_verb} {the} {air_noun}.',
                     i=['I', 'We'], the=['this', 'that', 'the'], labels=0,
                         save=True)
```

Figure 1: Example of CHECKLIST templates on the linguistic capability of "Sentiment change over time, present should prevail".

The *syntax-based sentence expansion* phase expands the seed sentences with additional syntactic elements (i.e., words) to cover many real-world syntactic structures (**C1**). It first performs a syntax analysis to identify the part-of-speech (PoS) tags that can be inserted to each seed, by comparing the PoS parse trees between the seed sentence and many other sentences from a large reference dataset. Each identified tag is inserted into the seed as a *mask*. It then uses an NLP recommendation model (i.e., BERT [5]) to suggest possible words. If a resulting sentence is validated to be consistent with the specification which additionally defines the rules for expansion (e.g., the expanded word should be neutral), **C2** and **C3** are still satisfied.

We now describe each phase of S^2LCT in detail.

3.1 Specification-based Seed Generation

The seed generation phase of S^2LCT starts by searching sentences in a real-world dataset that match the rules defined in the linguistic capability specification, and then transforming the matched sentences using templates to generate seed sentences that conform to individual linguistic capabilities. The reasons for this design choice are twofold. First, while generally judging which linguistic capability any sentence falls into and which label it should have is infeasible, there exist simple rules and templates to allow classifying the resulting sentences into individual linguistic capabilities and with the correct labels, with high confidence. This enables us to test each linguistic capability individually. Second, searching from a real-world dataset ensures that the sentences used as test cases for testing linguistic capabilities are realistic and diverse. The diverse test cases are more likely to achieve a high coverage of the target model's functionality in each linguistic capability, thus detecting more errors. In this phase, S^2LCT first searches and selects sentences applicable to the linguistic capability in a given real-world dataset with search rules. In case that the search rules only fulfill portion of the linguistic capability specifications, the selected sentences are not yet ready to be used as seeds, we transform the selected sentences into seed sentences using the heuristic templates. Table 1 shows the search rules and the transformation templates of all 11 linguistic capabilities we implemented in S^2LCT. The first column shows the linguistic capability type and its description, and the second column shows the search rule and transformation template used in each linguistic capability. For LC1 and LC2, the NLP models are evaluated in the scope of short sentences with selective sentiment words. It does not require any transformation because the search rules alone are sufficient to find sentences that conform to the linguistic capabilities. On the other hand, search rules of LC3 to LC11 are not enough to match their linguistic capability specification; thus, S^2LCT additionally uses templates to transform the searched sentences to match the corresponding linguistic capability. For example, in LC3's transformation template, the searched sentences are appended as part of a sentence to generate a seed. In LC4's transformation template, the searched demonstrative sentences are negated.

Table 1: Search rules and transformation templates for linguistic capabilities.

Linguistic capability	Search rule and transformation template
LC1: Short sentences with neutral adjectives and nouns	**Search** seed={length: <10; include: neutral adjs & neutral nouns; exlcude: pos adjs & neg adjs & pos nouns & neg nouns; label: neutral} **Transform** N/A
LC2: Short sentences with sentiment-laden adjectives	**Search** seed={length: <10; include: pos adjs; exlcude: neg adjs & neg verbs & neg nouns; label: pos} | {length: <10; include: neg adjs;exclude: pos adjs & pos verbs & pos nouns & neg verbs & neg nouns; label: neg} **Transform** N/A
LC3: Sentiment change over time, present should prevail	**Search** pos_sent={label: pos}, neg_sent={label: neg} **Transform** seed={[‘Previously, I used to like it saying that’,‘Last time, I agreed with saying that’,‘I liked it much as to say that’]+[pos_sent | neg_sent]+[‘but’, ‘although’, ‘on the other hand’]+[‘now I don’t like it.’, ‘now I hate it.’]} | {[‘I used to disagree with saying that’,‘Last time, I didn’t like it saying that’,‘I hated it much as to say that’]+[neg_sent, pos_sent]+[‘but’, ‘although’, ‘on the other hand’]+[‘now I like it.’]}
LC4: Negated negative should be positive or neutral	**Search** demonstrative_sent={start: [This, That, These, Those] + [is, are]; label: neg} **Transform** seed=negation of demonstrative_sent ([‘is’] -> [‘is not’, ‘isn’t’], [‘are’] -> [‘are not’, ‘aren’t’])
LC5: Negated neutral should still be neutral	**Search** demonstrative_sent={start: [This, That, These, Those] + [is, are]; label: neutral} **Transform** negation of demonstrative_sent

LC6: Negation of negative at the end, should be positive or neutral	**Search** neg_sent={label: neg}
	Transform seed={[‘I agreed that’, ‘I thought that’]+[neg_sent]+[‘but it wasn’t’, ‘but I didn’t’]}

LC7: Negated positive with neutral content in the middle	**Search** pos_sent={length: <20; label: pos}, neutral_sent={length: <20; label: neutral}
	Transform seed={[‘I wouldn’t say,’, ‘I do not think,’, ‘I don’t agree with,’]+[neutral_sent]+[‘,’]+[pos_sent]}

LC8: Author sentiment is more important than others	**Search** pos_sent={label:pos}, neg_sent={label: neg} **Transform** seed={[temp1]+[pos_sent]+[temp2]+[neg_sent]} | {[temp1]+[neg_sent]+[temp2]+[pos_sent]} where temp1={[‘Some people think that’, ‘Many people agree with that’, ‘They think that’, ‘You agree with that’], temp2=[‘but I think that’]}

LC9: Parsing sentiment in (question, yes) form	**Search** pos_sent={label: pos}, neg_sent={label: neg}
	Transform seed={[‘Do I think that’, ‘Do I agree that’]+[pos_sent | neg_sent]+[‘? yes’]}

LC10: Parsing positive sentiment in (question, no) form	**Search** pos_sent={label: pos}
	Transform seed={[‘Do I think that’, ‘Do I agree that’]+[pos_sent]+[‘? no’]}

LC11: Parsing negative sentiment in (question, no) form	**Search** neg_sent={label: neg}
	Transform seed={[‘Do I think that’, ‘Do I agree that’]+[neg_sent]+[‘? no’]}

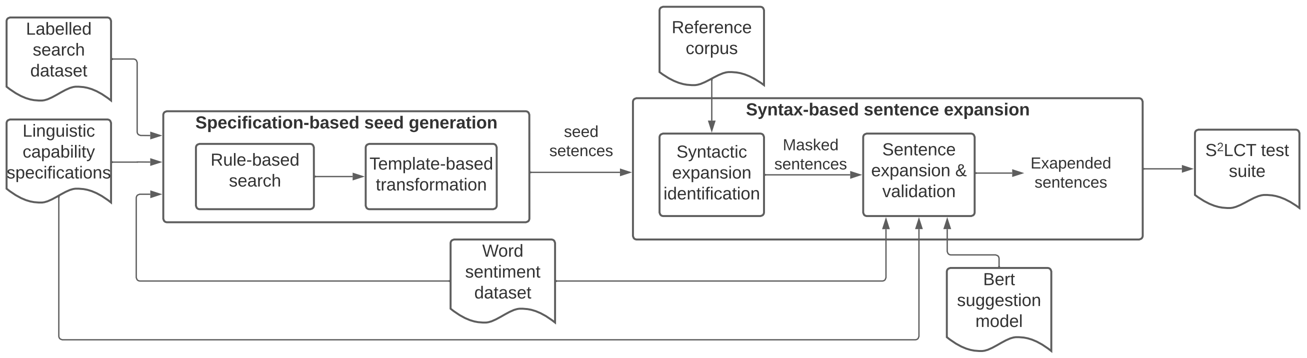

Labelled search dataset
Linguistic capability specifications
Specification-based seed generation
Rule-based search
Template-based transformation
seed setences
Reference corpus
Syntax-based sentence expansion
Syntactic expansion identification
Masked sentences
Sentence expansion & validation
Expanded sentences
S²LCT test suite
Word sentiment dataset
Bert suggestion model

Figure 2: Overview of S²LCT.

3.2 Syntax-based Sentence Expansion

The simple search rules and transformation templates used to generate the seed sentences may limit the syntactic structures these seeds may cover. To address this limitation, the syntax-based sentence expansion phase extends the seed sentences to cover syntactic structures commonly used in real-life sentences. Our idea is to differentiate the parse trees between the seed sentences and the reference sentences from a large real-world dataset. The extra PoS tags in the reference parse trees are identified as potential syntactic elements for expansion and inserted into the seed sentences as masks. We then use masked language model to suggest the fill-ins. If the resulting sentences still conform to the linguistic capability specification, they are added to S²LCT's test suite.

3.2.1 Syntax Expansion Identification. Algorithm 1 shows how masks are identified for each seed sentence. It takes the parse trees of the seeds, generated by the Berkeley Neural Parser [12, 13], and a reference context-free grammar (CFG) from the Penn Treebank corpus dataset [17] as inputs. This reference CFG was learned from a large dataset [31] that is representative of the distribution of real-world language usage. The algorithm identifies the discrepancy between the seed syntax and the reference grammar to decide how a seed can be expanded.

For each production of in each seed's parse tree (lines 3 and 4), we extract its non-terminal at the left-hand-side (line 5), *s_lhs*, and the grammar symbols at the right-hand-side (line 6), *s_rhs*. In line 7, the algorithm iterates through all productions in the reference context-free grammar and matches these that have the same non-terminal at the left-hand-side as *s_lhs*. The right-hand-side of each matched production is called *r_rhs*. If *s_rhs* consists of a subset of the grammar symbols in *r_rhs* (line 8), the additional symbols in the *r_rhs* are inserted as masks in the parse tree of the seed sentence, in their respective positions in the expanded production. The left to right traversal of the leaves of an expanded parse tree forms a masked sentence. Due to a potential large number of masked sentences created by this algorithm, we randomly select k masked sentences for the next sentence expansion and validation phase (line 14).

Running example. Figure 3 shows an example using Algorithm 1 to generate a masked sentence. The sentence "Or both." is a seed of the linguistic capability of "Short sentences with neutral adjectives and nouns". The tree on the left shows the parse tree of this seed; it consists of two productions: "FRAG->[CC, NP, .]" and "NP->[DT]". When matching the left-hand-side non-terminal of the second production (i.e., "NP") in the reference CFG, we found that it includes a production "NP->[DT, NNS]" which has an additional symbol "NNS" on the right-hand-side. The algorithm thus expands the parse tree with this symbol, shown in the second tree. The masked sentence "Or both {MASK}." is the result of the left-to-right traversal of this expanded parse tree.

Algorithm 1 Syntax expansion identification algorithm.

```
1:  Input: Parse trees of seed sentences $S$, reference context-free grammar $R$
2:  Output: Set of masked sentences $M$
3:  for each part tree $s$ from $S$ do
4:      for each production $s\_prod$ from $s$ do
5:          $s\_lhs = s\_prod.lhs$
6:          $s\_rhs = s\_prod.rhs$
7:          for each $r\_rhs$ from $R[s\_lhs]$ do
8:              if $s\_rhs \subset r\_rhs$ then
9:                  $M = M \cup insertMask($r_rhs-s_rhs$, s)$
10:             end if
11:         end for
12:     end for
13: end for
14: return $random(M, k)$
```

3.2.2 Sentence Expansion and Validation. In this phase, the words to fill in the masks in the masked sentences are suggested by the BERT model [5]. BERT is a transformer-based natural language model that is pre-trained on two tasks: masked token prediction and next sentence prediction. BERT model is capable of suggesting words for the masked token according to its surrounding context in a sentence. For each masked token, multiple words are suggested, ranked by their confidence scores. Because BERT model is not aware of the linguistic capability specification and the grammar symbol in the expanded parse tree, an expanded sentence using the suggested words may no longer satisfy the linguistic capability

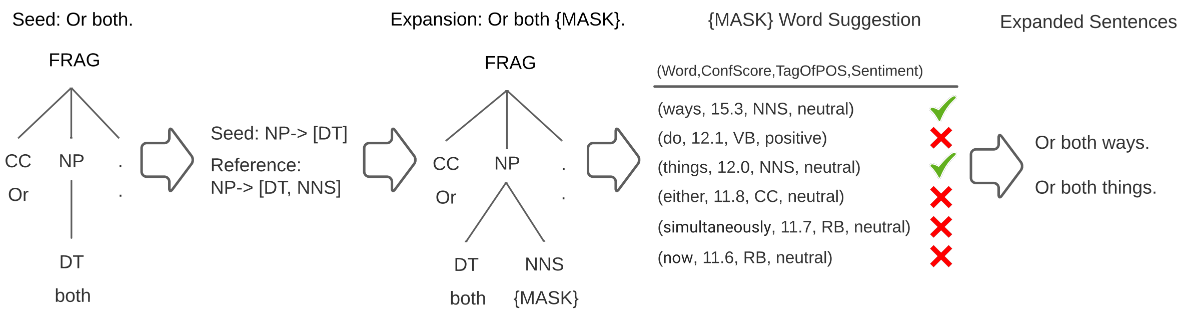

Seed: Or both.
FRAG
CC
NP
.
Or
DT
both
Seed: NP-> [DT]
Reference:
NP-> [DT, NNS]
Expansion: Or both {MASK}.
FRAG
CC
NP
.
Or
DT
NNS
both
{MASK}
{MASK} Word Suggestion
(Word,ConfScore,TagOfPOS,Sentiment)
(ways, 15.3, NNS, neutral) ✓
(do, 12.1, VB, positive) ✗
(things, 12.0, NNS, neutral) ✓
(either, 11.8, CC, neutral) ✗
(simultaneously, 11.7, RB, neutral) ✗
(now, 11.6, RB, neutral) ✗
Expanded Sentences
Or both ways.
Or both things.

Figure 3: Example of sentence expansion.

specification. Therefore, we perform validation on the suggested words and only accept them if the following three criteria are met.

First, the PoS tag of the suggested word must match the PoS tag of the expanded symbol in the parse tree. For the example in Figure 3, the masked symbol is a “NNS" (i.e., plural noun); thus, the suggested word must also be a “NNS". We use SpaCy [9], a free open-source library for natural language processing, to extract the PoS tag for each suggested word.

Second, it is required that the sentiment of the expanded sentence is the same as the seed sentence. To ensure this, the suggested words must be neutral.

Third, we additionally verify that the expanded sentences satisfy the same search rules for the seed sentence in LC1 and LC2. This criteria cannot be applied to other linguistic capabilities because they have additional transformation templates.

Running example. The third step in Figure 3 shows the words suggested by BERT. For this masked sentence, BERT suggested six words. Each word is associated with the confidence score provided by BERT, the PoS tag, and the sentiment. Among the six words, only “ways” and “things” are validated by S^{2}LCT because they have the Pos tag “NNS” and are neutral. In addition, it is found that both sentences meets the search rule of the associated linguistic capability of “Short sentences with neutral adjectives and nouns”. In the end, two sentences of “Or both ways” and “Or both things” are generated.

4 EXPERIMENTAL SETUP

In this section, we present the setup of the experiments to evaluate the effectiveness of S^{2}LCT. We address the following research questions (RQs):

RQ1 : Can S^{2}LCT generate more diverse test cases than CHECKLIST?
RQ2 : Can S^{2}LCT generate test sentences with correct sentiment labels?
RQ3 : Are S^{2}LCT generated test sentences correctly categorized into a linguistic capability?
RQ4 : Can S^{2}LCT be useful to find root causes of bugs in the sentiment analysis models?

4.1 Experimental Subjects

NLP Models & Dataset. We evaluate our approach on three learning-based sentiment analysis models implemented in Transformer library from the Hugging Face centralized Model Hub:[1] `BERT-base` (textattack/bert-base-uncased-SST-2), `RoBERTa-base` (textattack/bert-base-uncased-SST-2), and `DistilBERT-base` (distilbert-base-uncased). These models were pre-trained on English language using a masked language modeling (MLM) objective, and fine-tuned on the sentiment analysis task. In this experiment, we use SST-2 [30] dataset for searching the seeds in S^{2}LCT. SST-2 is a corpus of movie review, and consists of 11,855 sentences with sentiment scores. We split the scores into ranges [0, 0.4], (0.4, 0.6] and (0.6, 1.0] to assign negative, neutral and positive labels, respectively. In addition, we use SentiWordNet as the word sentiment dataset [1]. SentiWordNet is a publicly available English sentiment lexicons. It provides lexical sentiment scores and the sentiment word labels are categorized by implementing the rules in [4].

Comparison Baseline. We compare S^{2}LCT with CHECKLIST,[2] a manual template-based approach to generate test cases. In this experiment, we used the CHECKLIST released sentiment analysis test cases which are generated from its publicly available jupyter notebook implementation.

4.2 Experimental Process

RQ1. Recall that a key limitation of CHECKLIST is that its template-based approach relies on significant manual efforts and may not generate test cases that comprehensively cover the sentences in a linguistic capability. S^{2}LCT, instead, automatically generates test cases based on a search dataset and the syntax in a large reference corpus. We expect S^{2}LCT can generate a more diverse test suite than CHECKLIST, and achieve more probable evaluation results.

We first evaluate the three sentiment analysis models by testing them on the S^{2}LCT generated test cases, reporting number of test cases and each model’s failure rate in each linguistic capability.

Next, to measure diversity, we follow the approach presented by Ma et al. [16], where the authors measure the coverage of NLP model

[1]https://huggingface.co/models
[2]https://github.com/marcotcr/checklist

intermediate states as corner-case neurons. Because the matrix computation of intermediate states impacts NLP model decision-making, a test suite that covers a greater number of intermediate states can represent more NLP model decision-making, making it more diverse. Specifically, we used two coverage metrics by Ma et al. [16], *boundary coverage* (BoundCov) and *strong activation coverage* (SActCov), to evaluate the test suite diversity. It is worth noting that a test sample with a statistical distribution similar to the training data would rarely be found in the corner case region. As a result, covering a larger corner case region means that the test suite is more likely to be buggy.

$$\begin{aligned} \text{UpperCorner}(\mathcal{X}) &= \{n \in N \mid \exists x \in \mathcal{X} : f_n(x) \in (high_n, +\infty)\}; \\ \text{LowerCorner}(\mathcal{X}) &= \{n \in N \mid \exists x \in \mathcal{X} : f_n(x) \in (-\infty, low_n)\}; \end{aligned} \tag{1}$$

Equation 1 shows the formal definition of the corner-case neuron of the NLP model $f(\cdot)$, where $\mathcal{X}$ is the given test suite, N is the number of neurons in model $f(\cdot)$, $f_n(\cdot)$ is the n^{th} neuron's output, and $high_n$ and low_n are the n^{th} neurons' output bounds on the model training dataset. Eq. 1 can be interpreted as the collection of neurons that emit outputs beyond the model's numerical boundary.

$$\begin{aligned} BoundCov(\mathcal{X}) &= \frac{|UpperCorner(\mathcal{X})| + |LowerCorner(\mathcal{X})|}{2 \times |N|} \\ SActCov(\mathcal{X}) &= \frac{|UpperCorner(\mathcal{X})|}{|N|} \end{aligned} \tag{2}$$

The formal definition of our coverage metrics are shown in Equation 2, where BoundCov measures the coverage of neurons that produce outputs that exceed the upper or lower bounds, and SActCov measures the coverage of neurons that create outputs that exceed the lower bound. Higher coverage indicates the test suite is better for triggering the corner-case neurons, thus better test suite diversity.

To answer RQ1, for each NLP model under test, we first feed its training dataset to compute each neuron's lower and upper bounds. After that, we select the same number of test cases from S^{2}LCT and CHECKLIST as the test suite and compute the corresponding coverage metrics.

***RQ2 and RQ3*.** As described in Section 3, S^{2}LCT generates test cases in two steps: specification-based seed generation and syntax-based sentence expansion. These automated steps may generate seed/expanded sentences marked with incorrect sentiment labels or categorized into wrong linguistic capabilities. For example, the search rule and template defined in a linguistic capability may not always generate seed sentences in that capability or with the correct label. To answer RQ2 and RQ3, we performed a manual study to measure the correctness of the sentiment labels and linguistic capabilities associated with the seed/expanded sentences, produced by S^{2}LCT.

In the manual study, we randomly sample 100 seed sentences, each of which has at least one expanded sentence, and divide these seeds to two sets (i.e., 50 in each set). For each sampled seed sentence, we randomly obtain one of its expanded sentences. This forms the two sets of sentences (200 sentences in total) we use for this study, each with 50 seeds and 50 corresponding expanded sentences. We recruited three participants of this study; all of them are graduate students with no knowledge about this work. 2 of them were assigned one set of sentences and the third was assigned the other set. Each participant was asked to provide two scores for each sentence. *(1) Relevancy score between sentence and its associated linguistic capability*: this score measures the correctness of S^{2}LCT linguistic capability categorization. The scores are discrete, ranging from 1 ("strongly not relevant") to 5 ("strongly relevant"). *(2) sentiment score of the sentence*: this score measures the sentiment level of the sentence . It is also discrete, ranging from 1 ("strongly negative") to 5 ("strongly positive"). We measure the following:

$$sent\_consistency = \frac{1}{\#Sample \cdot} \cdot \sum_i \delta(label_{S^2LCT} = label_{human}) \tag{3}$$

$$LC\_relevancy_{AVG} = \frac{1}{\#Sample \cdot} \cdot \sum_i Norm(LC\_relevancy_i) \tag{4}$$

Equation 3 represents the percentage of the test cases that S^{2}LCT and the participants produce the same sentiment labels. High value of this metric indicates S^{2}LCT generates test cases with correct labels. Equation 4 represents the average of the normalized relevancy score between a sentence and its associated linguistic capability. Higher average score means indicates the linguistic capability categorization by S^{2}LCT is correct. We answer RQ2 and RQ3 using the metrics defined by Equation 3 and Equation 4, respectively.

***RQ4*.** To answer RQ4, we conduct experiments to demonstrate that S^{2}LCT can help developers understand the bugs in the NLP models. Recall that S^{2}LCT generates test cases by mutating seed sentences (e.g. by expanding one token in the seed input). Still, it is unclear why mutating one token will cause the model to produce misclassified results. We seek to help developers understand why such mutation will result in the misclassification. Existing work [3, 6, 22] has demonstrated that the ML model prediction is dominated by a minimal set of input features (i.e. tokens in input sentences). Motivated by such intuition, we identify a minimal set of input tokens that dominate the model prediction.

Formally, given a input sentence $x = [tk_1, tk_2, \cdots, tk_n]$, and the NLP model under test $f(\cdot)$, our goal is to find a masking template $T = [t_1, t_2, \cdots, t_n]$, where t_i is 0 or 1, representing masking the i^{th} token (i.e. tk_i) in x or not. The template T can mask some tokens in x with attribute tokens, and the masked input has a high probability of retaining the original prediction x, denoted as

$$P(f(T(x)) = f(x)) \geq P_{thresh} \tag{5}$$

To create such a template T, we first compute the contribution score of each input token using an existing explainable ML technique [6]. We then begin with the full mask template (i.e., all tokens are masked); such full mask template definitely does not satisfy Equation 5. We then iteratively shift one position from mask to non-mask based on the order of each token's contribution score, until the template T satisfies Equation 5. Because we iterate the size of the mask, the generated template T will keep the minimum number of tokens in x. Moreover, since the input x is an incorrect prediction, the generated template T is likely to produce misclassification (i.e., the probability to be misclassified is larger than P_{thresh}).

5 EXPERIMENTAL RESULTS

This section presents experiment results and answer the RQs by studying the results quantitatively and qualitatively.

Table 2: Results of BERT-base, RoBERTa-base and DistilBERT-base sentiment analysis models on S^{2}LCT test cases. BERT-base, RoBERTa-base and DistilBERT-base models are denoted as BERT, RoBERTa and dstBERT, respectively.

Linguistic capability	S^{2}LCT #Seeds	S^{2}LCT #Exps	S^{2}LCT Seed Fails[%]	S^{2}LCT Exp Fails[%]	S^{2}LCT #PassToFail
LC1: Short sentences with neutral adjectives and nouns	19	210	BERT: 78.95 RoBERTa: 89.47 dstBERT: 100.00	BERT: 86.67 RoBERTa: 76.19 dstBERT: 94.29	BERT: 19 RoBERTa: 9 dstBERT: 0
LC2: Short sentences with sentiment-laden adjectives	50	394	BERT: 4.00 RoBERTa: 4.00 dstBERT: 2.00	BERT: 4.31 RoBERTa: 4.82 dstBERT: 2.79	BERT: 4 RoBERTa: 5 dstBERT: 8
LC3: Sentiment change over time, present should prevail	50	-	BERT: 24.00 RoBERTa: 56.00 dstBERT: 78.00	BERT: - RoBERTa: - dstBERT: -	BERT: - RoBERTa: - dstBERT: -
LC4: Negated negative should be positive or neutral	50	1784	BERT: 92.00 RoBERTa: 82.00 dstBERT: 88.00	BERT: 94.17 RoBERTa: 86.15 dstBERT: 88.51	BERT: 49 RoBERTa: 64 dstBERT: 26
LC5: Negated neutral should still be neutral	26	1009	BERT: 92.31 RoBERTa: 92.31 dstBERT: 92.31	BERT: 92.37 RoBERTa: 93.06 dstBERT: 95.24	BERT: 62 RoBERTa: 38 dstBERT: 74
LC6: Negation of negative at the end, should be positive or neutral	50	1486	BERT: 100.00 RoBERTa: 100.00 dstBERT: 100.00	BERT: 100.00 RoBERTa: 100.00 dstBERT: 100.00	BERT: 0 RoBERTa: 0 dstBERT: 0
LC7: Negated positive with neutral content in the middle	50	1634	BERT: 84.00 RoBERTa: 42.00 dstBERT: 76.00	BERT: 88.00 RoBERTa: 40.64 dstBERT: 78.34	BERT: 40 RoBERTa: 54 dstBERT: 9
LC8: Author sentiment is more important than of others	50	2323	BERT: 44.00 RoBERTa: 26.00 dstBERT: 40.00	BERT: 50.06 RoBERTa: 31.21 dstBERT: 44.77	BERT: 134 RoBERTa: 78 dstBERT: 31
LC9: Parsing sentiment in (question, yes) form	50	1373	BERT: 4.00 RoBERTa: 2.00 dstBERT: 6.00	BERT: 2.69 RoBERTa: 4.15 dstBERT: 5.54	BERT: 43 RoBERTa: 46 dstBERT: 14
LC10: Parsing positive sentiment in (question, no) form	50	1218	BERT: 42.00 RoBERTa: 58.00 dstBERT: 92.00	BERT: 46.14 RoBERTa: 64.53 dstBERT: 87.60	BERT: 40 RoBERTa: 18 dstBERT: 8
LC11: Parsing negative sentiment in (question, no) form	50	1161	BERT: 100.00 RoBERTa: 100.00 dstBERT: 94.00	BERT: 100.00 RoBERTa: 100.00 dstBERT: 98.97	BERT: 0 RoBERTa: 0 dstBERT: 4

5.1 RQ1: Test Suite Diversity

Our experimental results show that *S^{2}LCT generated many test cases that the sentiment analysis models failed to predict the correct labels, and it produced significantly more diverse test cases than CHECKLIST did.*

Table 2 shows the testing results of three sentiment analysis models. First column lists linguistic capabilities for the sentiment analysis task, and Columns 2-3 show the numbers of seed and expanded test cases, respectively. Columns 4-5 show the failure rate (i.e., percentage of test cases that a model predicts incorrect label) of each sentiment analysis models on the seed and expanded test cases respectively. Column 6 shows the number of expanded test cases that failed, but their corresponding seed test cases passed.

We observe that in all but one linguistic capabilities, S^{2}LCT produces hundreds of test cases, expanding at least an order of magnitude more test cases than the seeds. LC3 is an outlier; the 50 randomly selected seeds in this linguistic capability did not produce any validated expansions. LC1 and LC5 have 19 and 26 seeds, respectively. This means that S^{2}LCT's search rules and transformation templates of these two linguistic capabilities produced few seeds. Nevertheless, the syntax-base sentence expansion phase allows generation of 210 and 1009 test cases, respectively.

In terms of sentiment analysis model performance, all three models achieve low failure rates in LC9 and LC2 while the failure rates are high in all other linguistic capabilities (26%-100%). We also observe that there are significant number of expanded test cases that failed, but their corresponding seeds did not (last column in Table 2). This shows that the syntax-based sentence expansion phase indeed introduces more diverse sentence structures in the test cases that cause the models to fail.

Next, Figure 4 shows the coverage results of S^{2}LCT and CHECKLIST test cases. The red line represents S^{2}LCT coverage and the black line represents CHECKLIST coverage. Each column in Figure 4 represents the results for one NLP model. The first row is the *BoundCov* results and the second row is the *SActCov* results.

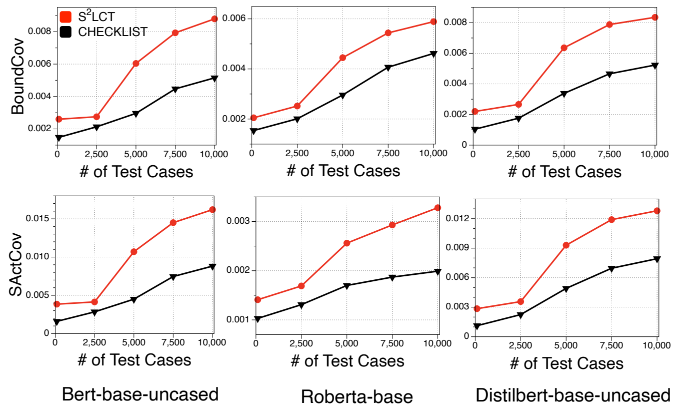


Figure 4: Coverage results of S^{2}LCT and CHECKLIST test cases.

Table 3: Manual study results of label consistency and linguistic capability relevancy of S^{2}LCT generated test cases compared to human annotations.

Type	#Test_Cases	Label_Consistency	Sent_Relevancy
Seed	100	0.83	0.90
Expanded	100	0.84	0.90

We made three observations. First, for *all* experimental settings (i.e., NLP model and coverage metric), S^{2}LCT achieves higher coverage than CHECKLIST. Recall that a higher coverage implies the test cases are more diverse and do not have a similar statistical distribution to the model training data. As a result, a test suite with greater coverage complements the model training data distribution (i.e. holdout testing data) better. For example, for the first NLP model under test, S^{2}LCT can achieve a higher coverage than CHECKLIST with only half the number of test cases. This result confirms that S^{2}LCT can generate more diverse test cases to complement the holdout dataset for testing NLP models.

Second, as the number of test cases increases, the test suite can achieve better coverage. Such observation is intuitive. However, generating a more extensive test suite is not easy, particularly for CHECKLIST, which is a manually template-based approach.

Third, for each NLP model, there is no fixed relationship between *BoundCov* and *SActCov*. In other words, while a test suite may produce higher *BoundCov* for some models, the same test suite may get higher *SActCov* for other NLP models. Recall that *BoundCov* measures both the upper and lower corner neurons and *SActCov* measures only the upper corner neurons. Such observation implies that the upper and lower corner neurons are distributed unevenly, and measuring only one of them is not enough.

5.2 RQ2: Correctness of Sentiment Labels

Table 3 shows results of our manual study. The first column distinguishes the seed and expanded sentences. The number of test cases used for the study is represented in the second column. The label consistency score defined in Equation 3 is shown in column 3.

We observe that *S^{2}LCT generates test cases that consistently label their sentiment correctly.* Column 3 shows that the label consistency scores are 0.83 and 0.84 for the seed and expanded sentences, respectively. This means that S^{2}LCT generates test oracles consistent with human understanding most of the time. Also, there is little difference of the scores between the seed and expanded sentences. This implies that the syntax-based sentence expansion in S^{2}LCT preserves the sentiment as its seed.

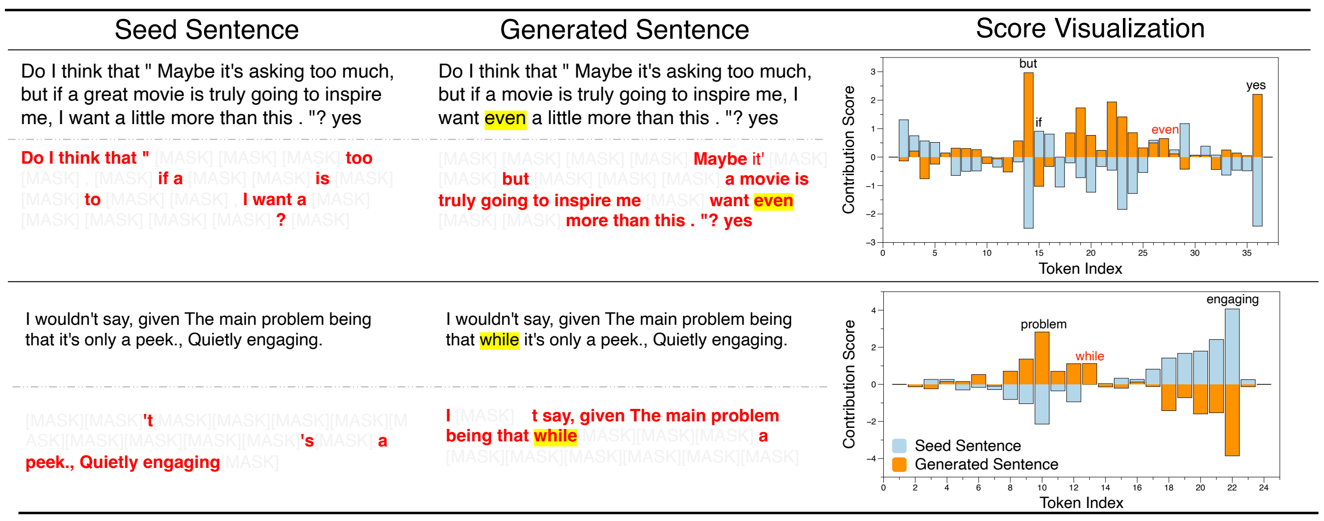
5.3 RQ3: Correctness of Linguistic Capability Categorization

The linguistic capability relevancy score defined in Equation 4 is shown in Column 4 of Table 3. The result shows that *S^{2}LCT generates test cases that are correctly categorized to the corresponding linguistic capabilities most of the time.* The linguistic capability relevancy scores for the seed and expanded sentences are both 0.9, achieving high order of agreement with human assessment. The fact that the expanded sentences generated by S^{2}LCT also have the same level of linguistic capability relevancy as the seed sentences shows that the syntax-based sentence expansion retains the linguistic capabilities.

5.4 RQ4: Use S^{2}LCT for Understanding Errors

Recall our RQ4 is to generate a template that will dominate the NLP model prediction to assist developers in understanding the false predictions. Figure 5 shows the generated templates for two randomly selected seed inputs and corresponding generated test inputs. The first example is to test “sentiments in (question, yes) form” (LC9), and the second example is to test “negative positive with neutral content in the middle” (LC7). The first column contains the seed sentence, the second contains the generated sentence, and the third contains thCT of each token’s contribution score. The blue bar indicates thxe score for seed inputs, whereas the orange bar reflects the score f CTor created sentences. We highlight the mutated token with yellow background and generated templates with red text.

The results show that after mutating the seed sentence with one token, the token set that dominates the NLP model prediction has changed. We can trace the root causes to the bias of training data on the linguistic capability under test; as a result, for the linguistic capability under test, the model has a bias towards positive/negative for certain token sequence patterns. For example, linguistic capability 9 has a bias toward the token sequence pattern that includes "maybe .... but ... even... yes". Thus, adding the token "even" to the seed sentence will match one of those biased sequence patterns. Sentences with s.8gmxuin the training dataset are dominantly positive, thus the models makes the wrong decision on the sentence with “even” as positive. The visualization of each token’s contribution score in the third column also confirms our observation. Once “even” is added, scores of other tokens such as “but” and “yes” all change from negative to positive. To fix the issue for LC9, we ne ed to add more training negative samples with the è£Ÿ of “maybe . .. but ... even ... yes”.



Seed Sentence	Generated Sentence
Do I think that " Maybe it's asking too much, but if a great movie is truly going to inspire me, I want a little more than this . "? yes	Do I think that " Maybe it's asking too much, but if a movie is truly going to inspire me, I want even a little more than this . "? yes
Do I think that " [MASK] [MASK] [MASK] too [MASK] , [MASK] if a [MASK] [MASK] is [MASK] [MASK] to [MASK] [MASK] , I want a [MASK] [MASK] [MASK] [MASK] [MASK] ? [MASK]	[MASK] [MASK] [MASK] [MASK] Maybe it' [MASK] [MASK] but [MASK] [MASK] [MASK] a movie is truly going to inspire me [MASK] want even [MASK] [MASK] more than this . "? yes
I wouldn't say, given The main problem being that it's only a peek., Quietly engaging.	I wouldn't say, given The main problem being that while it's only a peek., Quietly engaging.
[MASK][MASK]'t[MASK][MASK][MASK][MASK][MASK][MASK][MASK][MASK][MASK]'s[MASK] a peek., Quietly engaging[MASK]	I [MASK] t say, given The main problem being that while[MASK][MASK][MASK] a [MASK][MASK][MASK][MASK][MASK][MASK]

Figure 5: Visualization of the buggy reason of two S^{2}LCT generated test cases.

6 THREATS TO VALIDITY

There are two potential threats to validity. First, the dataset used in our study might not be representative of all English grammatical structures and word sentiments. We mitigate this threat by using widely used dataset in the NLP domain [10]. Second, using ranges of the sentiment scores in SST-2 to determine the sentiment labels of search dataset may introduce incorrect labels. We performed a manual study to confirm that this is rare. [17, 31].

7 RELATED WORK

NLP Testing. With the increasing use of NLP models, evaluation of NLP models is becoming a more important issue. Apart from the accuracy-based testing scheme, recent works have considered model robustness as an aspect for evaluation. Belinkov and Bisk *et al.* [2] aimed to fail neural machine translation model by intentionally introducing noise in the input text. Pinjia *et al.* [7, 8] measured the robustness by assuming syntactic and semantic relation between input and output of neural machine translation model. Ribeiro *et al.* [24] proposed an approach to generalize semantically equivalent adversarial rules. In addition, Rychalska *et al.* [27] measured drops in BLEU scores by corruption operation, and compared model robustness based on the amount of the drops. Iyyer *et al.* [11] introduced learning-based model for adversarial data augmentation.

In addition to the robustness on adversarial set, various other aspects of the NLP model have been considered for the robustness evaluation. Prabhakaran *et al.* [19] developed an evaluation framework to detect unintended societal bias in NLP models. Rottger *et al.* [26] introduced a functional test suite for hate speech detection in the NLP model. Ribeiro *et al.* [21] measured logical consistency of NLP model. These techniques evaluate the robustness of the NLP model. However, we focused on the evaluation of model capability over multiple perspectives and produce debugging information by comparing seed and expanded test cases.

Linguistic Capability Evaluation. Wang *et al.* [32, 33] proposed multiple diagnostic datasets to evaluate NLP models. These datasets evaluate NLP model's ability to understand input sentence via natural language inference problems. More recently, CHECKLIST proposed an evaluation method of input-output behavior defined as linguistic capabilities. CHECKLIST generates behavior-guided inputs for validating the behaviors. [25]. Unlike prior work that relies on manual test case generation, we used structural information in text to generate test cases automatically.

NLP Model Debugging. Researchers have been explaining NLP model prediction and analyzing it for debugging the model. Ribeiro *et al.* [23] evaluated model prediction guided by human feedback providing relevance scores for words on the model prediction. Interactive error analysis [34] also has been proposed to evaluate model robustness. Zylberajch [35] used influence functions to generate model explanation, and it enables interactive debugging incorporating humans feedback explanation. Lertvittaya [14] proposed an approach to understand behavior of text classifier model and improve the model by disabling irrelevant hidden features. In this work, S^{2}LCT is useful for identifying the sources of model failure as shown in RQ4.

8 CONCLUSIONS

In this paper, we present S^{2}LCT that automatically searches and generates seed test cases for each linguistic capability. S^{2}LCT expands the seeds into a sufficient number of test cases based on context-free grammar, covering diverse syntactic structures. We evaluated sentiment analysis models on the generated test cases. It is shown that S^{2}LCT ensures quality of test cases by showing consistency of test oracle with its input sentence and its linguistic capability. In addition, we showed that diversity in S^{2}LCT test cases leads to cover more region in sentiment analysis models, resulting in higher reliability on the testing. We also examined failure-inducing cases to understand the root causes of the bugs. These results demonstrate the correctness and usefulness S^{2}LCT for model evaluation.

REFERENCES

[1] Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. SentiWordNet 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*. European Language Resources Association (ELRA), Valletta, Malta. http://www.lrec-conf.org/proceedings/lrec2010/pdf/769_Paper.pdf

[2] Yonatan Belinkov and Yonatan Bisk. 2017. Synthetic and Natural Noise Both Break Neural Machine Translation. *CoRR* abs/1711.02173 (2017). arXiv:1711.02173 http://arxiv.org/abs/1711.02173

[3] Simin Chen, Soroush Bateni, Sampath Grandhi, Xiaodi Li, Cong Liu, and Wei Yang. 2020. *DENAS: Automated Rule Generation by Knowledge Extraction from Neural Networks*. Association for Computing Machinery, New York, NY, USA, 813âĂŞ825. https://doi.org/10.1145/3368089.3409733

[4] Mihaela Colhon, Åđtefan VlĂČduÅčescu, and Xenia Negrea. 2017. How Objective a Neutral Word Is? A Neutrosophic Approach for the Objectivity Degrees of Neutral Words. *Symmetry* 9, 11 (2017). https://doi.org/10.3390/sym9110280

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR* abs/1810.04805 (2018). arXiv:1810.04805 http://arxiv.org/abs/1810.04805

[6] Wenbo Guo, Dongliang Mu, Jun Xu, Purui Su, Gang Wang, and Xinyu Xing. 2018. Lemna: Explaining deep learning based security applications. In *proceedings of the 2018 ACM SIGSAC conference on computer and communications security*. 364–379.

[7] Pinjia He, Clara Meister, and Zhendong Su. 2019. Structure-Invariant Testing for Machine Translation. *CoRR* abs/1907.08710 (2019). arXiv:1907.08710 http://arxiv.org/abs/1907.08710

[8] Pinjia He, Clara Meister, and Zhendong Su. 2020. Testing Machine Translation via Referential Transparency. *CoRR* abs/2004.10361 (2020). arXiv:2004.10361 https://arxiv.org/abs/2004.10361

[9] Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. (2017). To appear.

[10] Mujtaba Husnain, Malik Muhammad Saad Missen, Nadeem Akhtar, Mickaël Coustaty, Shahzad Mumtaz, and VB Prasath. 2021. A systematic study on the role of SentiWordNet in opinion mining. *Frontiers of Computer Science* 15, 4 (2021), 1–19.

[11] Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial Example Generation with Syntactically Controlled Paraphrase Networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics, New Orleans, Louisiana, 1875–1885. https://doi.org/10.18653/v1/N18-1170

[12] Nikita Kitaev, Steven Cao, and Dan Klein. 2019. Multilingual Constituency Parsing with Self-Attention and Pre-Training. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Florence, Italy, 3499–3505. https://doi.org/10.18653/v1/P19-1340

[13] Nikita Kitaev and Dan Klein. 2018. Constituency Parsing with a Self-Attentive Encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Melbourne, Australia, 2676–2686. https://doi.org/10.18653/v1/P18-1249

[14] Piyawat Lertvittayakumjorn, Lucia Specia, and Francesca Toni. 2020. FIND: Human-in-the-Loop Debugging Deep Text Classifiers. https://doi.org/10.48550/ARXIV.2010.04987

[15] 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. *CoRR* abs/1907.11692 (2019). arXiv:1907.11692 http://arxiv.org/abs/1907.11692

[16] 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.

[17] Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Comput. Linguist.* 19, 2 (jun 1993), 313âĂŞ330.

[18] Kayur Patel, James Fogarty, James A. Landay, and Beverly Harrison. 2008. Investigating Statistical Machine Learning as a Tool for Software Development. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Florence, Italy) *(CHI '08)*. Association for Computing Machinery, New York, NY, USA, 667âĂŞ676. https://doi.org/10.1145/1357054.1357160

[19] Vinodkumar Prabhakaran, Ben Hutchinson, and Margaret Mitchell. 2019. Perturbation Sensitivity Analysis to Detect Unintended Model Biases. *CoRR* abs/1910.04210 (2019). arXiv:1910.04210 http://arxiv.org/abs/1910.04210

[20] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. 2019. Do ImageNet Classifiers Generalize to ImageNet?. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 5389–5400. https://proceedings.mlr.press/v97/recht19a.html

[21] Marco Tulio Ribeiro, Carlos Guestrin, and Sameer Singh. 2019. Are Red Roses Red? Evaluating Consistency of Question-Answering Models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Florence, Italy, 6174–6184. https://doi.org/10.18653/v1/P19-1621

[22] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. " Why should i trust you?" Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 1135–1144.

[23] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. *CoRR* abs/1602.04938 (2016). arXiv:1602.04938 http://arxiv.org/abs/1602.04938

[24] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Semantically Equivalent Adversarial Rules for Debugging NLP models. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Melbourne, Australia, 856–865. https://doi.org/10.18653/v1/P18-1079

[25] Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond Accuracy: Behavioral Testing of NLP models with CheckList. In *Association for Computational Linguistics (ACL)*.

[26] Paul Röttger, Bertram Vidgen, Dong Nguyen, Zeerak Waseem, Helen Z. Margetts, and Janet B. Pierrehumbert. 2020. HateCheck: Functional Tests for Hate Speech Detection Models. *CoRR* abs/2012.15606 (2020). arXiv:2012.15606 https://arxiv.org/abs/2012.15606

[27] Barbara Rychalska, Dominika Basaj, Alicja Gosiewska, and Przemysław Biecek. 2019. Models in the Wild: On Corruption Robustness of Neural NLP Systems. In *Neural Information Processing*, Tom Gedeon, Kok Wai Wong, and Minho Lee (Eds.). Springer International Publishing, Cham, 235–247.

[28] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *ArXiv* abs/1910.01108 (2019).

[29] Sergio Segura, Gordon Fraser, Ana B. Sanchez, and Antonio Ruiz-CortÃľs. 2016. A Survey on Metamorphic Testing. *IEEE Transactions on Software Engineering* 42, 9 (2016), 805–824. https://doi.org/10.1109/TSE.2016.2532875

[30] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Seattle, Washington, USA, 1631–1642. https://aclanthology.org/D13-1170

[31] Sphinx and NLTK Theme. 2021. *NLTK Documentation*. https://www.nltk.org/howto/corpus.html.

[32] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems. *CoRR* abs/1905.00537 (2019). arXiv:1905.00537 http://arxiv.org/abs/1905.00537

[33] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2018. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. *CoRR* abs/1804.07461 (2018). arXiv:1804.07461 http://arxiv.org/abs/1804.07461

[34] Tongshuang Wu, Marco Tulio Ribeiro, Jeffrey Heer, and Daniel Weld. 2019. Errudite: Scalable, Reproducible, and Testable Error Analysis. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Florence, Italy, 747–763. https://doi.org/10.18653/v1/P19-1073

[35] Hugo Zylberajch, Piyawat Lertvittayakumjorn, and Francesca Toni. 2021. HILDIF: Interactive Debugging of NLI Models Using Influence Functions. *Proceedings of the First Workshop on Interactive Learning for Natural Language Processing* (2021).