Programming Language Representation with Semantic-level Structure

Anonymous Author(s)

ABSTRACT

Natural language processing (NLP) technique has become one of the core techniques for developing text analytic applications. These applications are required to achieve high reliability to be useful in practice. The trustworthiness of the prevalent NLP applications is obtained by measuring the accuracy of the applications on held-out dataset. However, evaluating NLP on testset with the held-out accuracy is limited in validating its overall quality because the held-out datasets are often not comprehensive. Along with this, evaluating an NLP model on task-specific behaviors defined on empirical linguistic capabilities has been introduced. However, such evaluation relies on manually created test cases, and is still limited to measure the model performance on biased dataset. In this work, we introduce S^{2}LCT, an NLP model testing infrastructure. Given a linguistic capability that users want to evaluate for a NLP model, S^{2}LCT finds suitable seed inputs from existing datasets, generates sufficient number of new test inputs by fuzzing the seed inputs based on their context-free grammar (CFG). We evaluate S^{2}LCT by showing its reliability on generated inputs and its generalization ability. In our experiments, we also show that S^{2}LCT facilitates identification of critical failures and its origins in the NLP models for sentiment analysis task.

ACM Reference Format:
Anonymous Author(s). 2022. Programming Language Representation with Semantic-level Structure. In *Proceedings of ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA 2022).* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

In the early stage of the software development process, automated software testing and debugging can identify and fix defects. Therefore, effective software testing assures software quality, and it meets needs of the end user. Nowdays, software at work has elements of machine learning (ML). Especially, Natural language processing (NLP) applications in software are growing exponentially. Therefore, as quality assurance of software is an essential process in the software development processes, trustworthiness in quality of NLP application has become an important component for its practical use in real world. Traditionally, the prevalent models of NLP are evaluated via train-validation-test splits. Training and validation set are used to train the NLP models, and the test set, also named as hold-out set, is used for testing the trained model. Meanwhile, the model performance is estimated into numbers used as performance metric. Especially, accuracy, the fraction of model output that the model correctly predicts, is the most widely the performance metric for classification models. Accordingly, a model is considered better if it has higher accuracy than another model.

Despite its simplicity and usefulness, there are several limitations for the prevalent testing paradigm: First, the testing paradigm mostly requires manual work for collecting and generating the data. The manual work is costly with respect to time consumption and its impact on market price. Therefore, it necessitates automated data generation for improving model testing approach. Second, this testing paradigm often overestimates the model performances from the hold-out set [7, 9, 13]. The overestimation comes from the discrepancy between distribution of data collected and actual distribution in real world. Oftentimes, the hold-out dataset is not representative and it is likely to introduce specific biases. In addition, the inconsistency between test input and its oracle causes biases. Such biases increase the discrepancy of distribution between the dataset and real-world data. Consequently, representiveness of the test data is required for testing approach. Not only that, forced aggregation static into a single number such as average makes user to difficult to localize and fix the bug found in hold-out set. Therefore, making this testing method fails to validate model behaviors resulting in localization of the causes of the inaccuracy at a high. In cost [18]. Therefore behavioral testing approach provides the better ability to examine the inaccuracy and find the bugs in the model.

To address these challenges, several approaches have been proposed to evaluate different aspects of the NLP models, such as robustness on adversarial examples [1, 3, 12, 15], model coverage [14], and fairness [8, 14]. In addition, Ribeiro et al. introduced CHECKLIST, a behavioral testing framework for evaluating NLP model on multiple linguistic capabilities [13]. CHECKLIST defines task-relevant linguistic capabilities and generates testcases for each linguistic capability. In spite of their effectiveness, they have limitations with respective to model evaluation. For example, Testing model coverage only shows model behaviors, but not the model performance on existing testing set. The fairness and robustness testing only focus on only one model capability resulting in its limited comprehensiveness. In addition, CHECKLIST relies on manually generated input templates; thus, the template generation requires manual work, and it needs to be preset before test data generation. Consequently, the templates becomes distributed in limited range of their structure. It restricts its ability to comprehensively test the linguistic capabilities.

Despite the limitations, CHECKLIST approach of testing 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 input and outputs observed in real world for the target NLP task ranging from simple to complicated behaviors so that they are able to evaluate the NLP model comprehensively. For exmaple,

ISSTA 2022, 18-22 July, 2022, Daejeon, South Korea

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

a linguistic capability of "Negated neutral should still be neutral." measures how accurately the sentiment analysis model understands the negative neutral input as an neutral sentiment. Therefore, it requires the sentiment analysis model to output neural sentiment on the negated neutral input. Such methodology of evaluation on the specified functionalities avoids the overestimation of the model performance as it equivalently measures the model performance on each functionality, and the separate model performance explain distribution of the performance over the linguistic capabilities. In the end, it provides not only the overall model performance, but also the malfunction facets of the model.

To address the limitations of existing works, we present S^{2}LCT, an automated NLP model evaluation method for comprehensive behavioral testing of NLP models on sentiment analysis task. We first address increasing input representiveness. Compared with templates used in CHECKLIST, S^{2}LCT instead establishes input requirement for evaluating a linguistic capability and finds suitable inputs that meet the requirement from existing public dataset. In this process, S^{2}LCT applies the fuzzing testing principle to generate inputs by mutating the selected inputs as seed inputs. Fuzzer in S^{2}LCT first expands seed input grammar structures and determines its available part-of-speech to maintain structural naturalness. After that, to hold contextual naturalness of the mutated inputs, the fuzzer completes the expanded new structures via data-driven context-aware word suggestion. Additionally, sentiment-independent words in the inputs are replaced with rule-based word suggestion. Further, we address the manual input generation process by automating the process mentioned above. Lastly, we adopts behavioral model testing method by introducing multiple behavior of linguistic capabilities on sentiment analysis task and generating inputs relevant to each linguistic capability.

We demonstrate its generality and utility as a NLP model evaluation tool by evaluating well-known sentiment analysis models: BERT-base [2], RoBERTa-base [6] and DistilBERT-base [16]. We show that ...

2 BACKGROUND

In this section, we provide a brief background on CHECKLIST testcase generation via an example. Quality of software is verified by ensuring the proper working of all functionalities without knowing the internal workings of the software. Knowing performances of model on the multiple functionalities provides users with better understanding and debugging the software. The same principle applies to the NLP model. traditional NLP model evaluation relying on a test set is lack of specification of model functionality. However, In NLP domain, there are many phenomena on linguistic input such as negation, questionization. Given a NLP task, the phenomena determine task-relevant output. Traditional evaluation method neglects them, thus, it becomes less efficient to detect and analyze which aspect the model yields unexpected outcome. To tackle this limitation, CHECKLIST introduces task-dependent linguistic capabilities for monitoring model performance on each linguistic capability. It assumes that the linguistic phenomena can be represented into the model behaviors as they provide what input and output are desired and how the model works with them. For each linguistic capability, CHECKLIST makes testcase templates and generate sentences by filling-in the value for each placeholder. We show an example of the templates in Figure 1. The templates in the figure is used for evaluating a sentiment analysis model on "Short sentences with sentiment-laden adjectives". For the templates defined from line 22 to 33 have placeholders such as it, air_noun, pos_adj. Values for the placeholders are defined at line 23, 1 and 6. Ater all, all combinations of the values of placeholders in a template are filled-in the template, and the senteneces are generated such as "The flight is good", "That airline was happy" and so on. Finally, these are used for evaluation of linguistic capability. Despite its simplicity of testcase generation,

```
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 linguistic capability of "Short sentences with sentiment-laden adjectives".

it still has limitations: it first relies on manual work for defining template structure and its values. Therefore, the manual work for testcase generation keeps the process costly. Extended from it, Second, such manual work produces the imitative forms between test cases, and it is likely to introduce bias on the testcases. We will show that these observations contribute to the performance of our models.

3 RELATED WORK

NLP Testing. With the increasing use of NLP models, evaluation of NLP models is becoming more important. Wang *et al.* [17] propose multiple diagnostic datasets to evaluate NLP models. Few recent works have also considered model robustness as an aspect for model evaluation. Different methods like adversarial set generation [1, 3, 12, 15], fairness evaluation [8, 14], logical consistency evaluation [10], prediction interpretations [11] and interactive error analysis [18] have been proposed to evaluate model robustness. More recently, CHECKLIST introduces input-output behaviors of linguistic capabilities and generates behavior-guided inputs for validating the behaviors. [13] However, the approach only relies on manually generated input templates, thus the template generation becomes expensive and time consuming. Also, it does not guarantee the comprehensive evaluation.

4 SPECIFICATION- AND SYNTAX-BASED LINGUISTIC CAPABILITY TESTING

Shiyi: Some paragraphs in this overview part should go earlier in the paper. I am just writing them here for now as I don't think we have them in the intro/background yet. We design and implement *Specification- and Syntax-based Linguistic Capability Testing (S^2LCT)* to automatically generate test cases to test the robustness of sentiment analysis models. We identify four goals for a large and effective test suite:

G1 the test suite should contain realistic sentences;
G2 the test suite should cover diverse syntactic structures;
G3 each test case should be categorized into a linguistic capability;
G4 the label of each test case should be automatically and accurately defined.

CHECKLIST's templates generate complete and realistic sentences, and each template maps to a linguistic capability, satisfying **G1** and **G3**. But CHECKLIST only uses Shiyi: X manually created templates to generate its test suite; all test cases generated by the same template share the same syntactic structure, thus violating **G2**. In addition, the label of each CHECKLIST test case has to be decided manually, associated with each template, violating **G4**.

We present S^{2}LCT, a new linguistic capability test case generation tool, that satisfies all of these criteria. Shiyi: I could not summarize a cohesive idea that drives our design. Leaving it here to fill in. Also, I felt my writing below still lacks justification for some design choices (e.g., why we do the differentiation). Figure 2 shows the overview of S^{2}LCT, which consists of two phases. The *specification-based seed generation* phase performs rule-based searches from a real-world dataset (**G1**) 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., Shiyi: add an example JL: sentence negation) are defined in the *linguistic capability specifications*, which guarantee that each resulting seed conforms to a specific linguistic capability (**G3**) and is labelled correctly (**G4**).

The *syntax-based sentence expansion* phase expands the seed sentences with additional syntactic elements (i.e., words Shiyi: more?JL: and production ruls in context-free grammar) to cover many real-world syntactic structures (**G2**). 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 []) 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), **G3** and **G4** are still satisfied. Last, because some validated sentences may include unacceptable suggested words given the context Shiyi: is this the right motivation to do selection? JL: I edited the sentence. please let me know if it is clear, we use a heuristic (i.e., the confidence score from the NLP recommendation model) to select the more realistic context-aware expanded sentences into S^{2}LCT's test suite.

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

4.1 Specification-based Seed Generation

The seed generation phase of S^{2}LCT 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^{2}LCT first search 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 appropriate to become seed, 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^{2}LCT. 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 guidance for its transformation because the search rule alone is sufficient to conform to the linguistic capabilities. On the other hand, search rules of LC3 and LC4 are not is not enough to match their linguistic

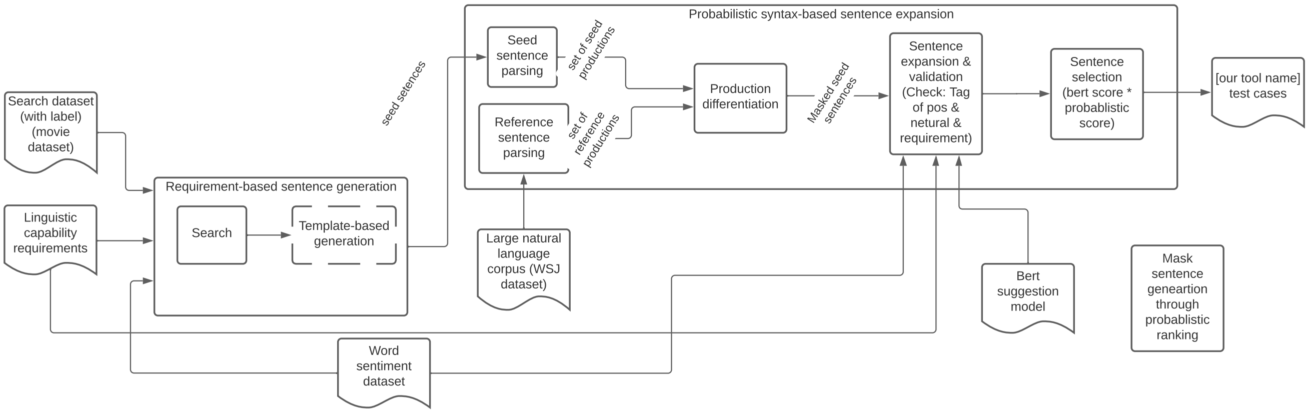
Probabilistic syntax-based sentence expansion
Seed sentence parsing
set of seed productions
Reference sentence parsing
set of reference productions
Production differentiation
Masked seed sentences
Sentence expansion & validation (Check: Tag of pos & netural & requirement)
Sentence selection (bert score * probablistic score)
[our tool name] test cases
seed setences
Search dataset (with label) (movie dataset)
Linguistic capability requirements
Requirement-based sentence generation
Search
Template-based generation
Large natural language corpus (WSJ dataset)
Bert suggestion model
Mask sentence geneartion through probablistic ranking
Word sentiment dataset

Figure 2: Overview of S^{2}LCT.

capability specification, thus S^{2}LCT uses heuristic templates to conform the found sentences to the linguistic capability. For example, the selected sentences becomes seeds by preturbing them with the templates and by negating the selected demonstrative sentences to conform to LC3 and LC4 respectively (see the third and forth rows in Table 1). Shiyi: @Jaeseong: revise the rest of 3.1 based on Table 1. I commented out the old text but it is still in the tex file. JL: I revised and added the explanation

4.2 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 S^{2}LCT's test suite. Shiyi: May have some redundancy and inconsistency with the overview part.

4.2.1 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 [4, 5], and a reference context-free grammar (CFG) from the Penn Treebank corpus dataset [] as inputs. The reference CFG is learned from a large dataset [] 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 match 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 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. Lastly, due to the inefficient cost of accessing full list of the masked sentences, we randomly select k masked sentences for the next sentence expansion and validation phase when the masked sentences are more than maximum number of masked sentences. The random sampling is unbiased approach since it gives same chance to be chosen. Thus, the random sample becomes representative of the population of the masked sentences, and it efficiently shows the usefulness of the S^{2}LCT. Shiyi: Add justification: why we need to select k masked sentences (performance?) and why random makes sense. JL: I added the statement for it

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:                  $M = M \cup insertMask(\text{r\_rhs-s\_rhs}, s)$
10:             end if
11:         end for
12:     end for
13: end for
14: return $random(M, k)$
```

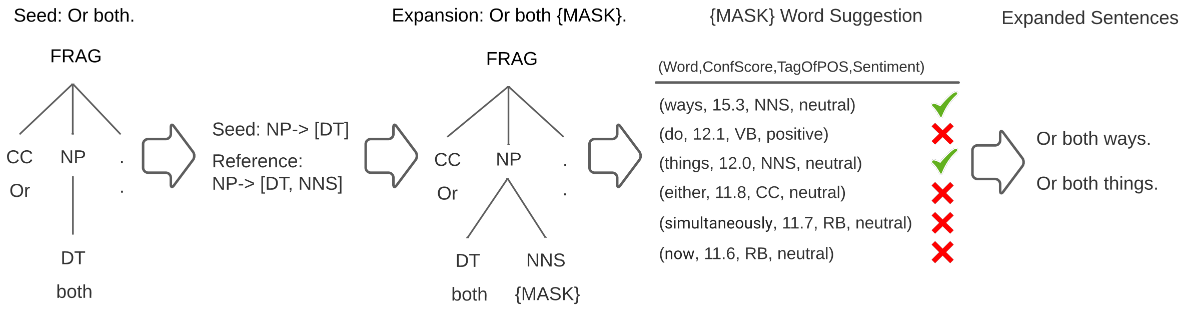
Seed: Or both.
FRAG
CC
NP
.
Or
.
DT
both
Seed: NP-> [DT]
Reference:
NP-> [DT, NNS]
Expansion: Or both {MASK}.
FRAG
CC
NP
.
Or
.
DT
NNS
both
{MASK}
{MASK} Word Suggestion
(Word,ConfScore,TagOfPOS,Sentiment)
(ways, 15.3, NNS, neutral) ✓
(do, 12.1, VB, positive) ✗
(things, 12.0, NNS, neutral) ✓
(either, 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 masked sentence generation. Shiyi: Expansion: Or both {MASK}. -> Masked sentence: Or both MASK.

Running example. Figure 3 shows an example using Algorithm 1 to generate a masked sentence. The sentence "Or both." is a seed of 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.

4.2.2 Sentence Expansion and Validation. In this phase, the words to fill in the masks in the masked sentences are suggested by the BERT pretrained model [2]. The BERT is a transformer-based natural language model. It is pretrained on two tasks of masked token prediction and next sentence prediction. As a result of the training process, the BERT model suggests word for the mask token according to its surrounding context in sentence. For each masked token, multiple words are suggested ranked by their confidence scores. Shiyi: Algorithm 1 does not require we only have one mask in the masked sentence. Can the model suggest multiple words at the same time? JL: yes. it can predict multiple masked tokens at the same time. Shiyi: Say a bit more about the BERT model suggestion. E.g., it may suggest multiple words for the same mask but they are ranked? JL: I added the explanation of BERT 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 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". In this work, we use SpaCy, a free open-source library for natural language processing, for extracting PoS tags for each suggested word.Shiyi: Say how we obtain the PoS tag of a suggested word.JL: I added it Second, it is required that the sentiment of the expanded sentence becomes the same as the seed sentence. To ensure this, the suggested words must be neutral. Shiyi: Should we present this as part of specification, called expansion rule? JL: I did it because I think that there is no issue on mentioning it and that is how we assumed and did Third, we additionally verify that the expanded sentences satisfied the same search rules for the seed sentence. Our goal for generating the expanded sentences is to use them for evaluating the sentiment analysis models on the associated linguistic capability in addition to the seed sentence. It is only achieved when the expanded sentences are also met with the same search rules for the linguistic capability. For LC1 as an example, the expanded sentence must still conform to the specification of the seed's linguistic capability specification. Therefore, the expanded sentences are required to be short and to only have neutral adjectives and nouns. Shiyi: Say why we use the third criteria only for LC1 and LC2. JL: I added the explanation

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^2LCT 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. Shiyi: Do we also check if the expanded sentence meets the specification?JL: Yes. I added the explanation of validation of linguistic capability requirement

4.2.3 Sentence Selection. Shiyi: @Jaeseong: Add how we select expanded sentences and motivate why. After

Running example. Shiyi: Refer to the example to say how we use the BERT score to select. JL: I dont think we need this subsection of sentence selection it is already explained at the previous stage with running example.

5 RESEARCH QUESTIONS FOR EVALUATION

6 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):

Shiyi: We may miss a RQ for the test results using S^{2}LCT. In the setup, we have not said which sentiment analysis models we tested, and how we measure the results (e.g., number of misclassified test cases).

RQ1 : Can S^{2}LCT generate consistent test sentence and its oracle?

RQ2 : Is S^{2}LCT generated testcases relevant to be used for their linguistic capability evaluation?

RQ3 : Can S^{2}LCT generate more diverse test cases than CHECKLIST?

RQ4 : Can S^{2}LCT be useful to find root causes of bugs in the sentiment analysis models?

Shiyi: Make clear (earlier in the paper): a test case is a sentence in a linguistic capability with a sentiment label.

RQ1 and RQ2. As described in Section 4, 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 RQ1 and RQ2, we perform 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 three sets of pairs of seed sentences and corresponding linguistic capability from seed test cases. For each set, we also select expanded sentences that S^{2}LCT generated from the formerly sampled seed sentences. In this experiment, each set has 100 sentences (50 from seed sentences and 50 from expanded sentences) and 300 sentences, in total, are used for the manual study. For each sampled set, two subjects are provided with the same set of sampled sentences. The subjects are asked for scoring the two following: **1. relevancy score between sentence and its associated linguistic capability**: this score measures the amount of appropriateness of the use of sentence for evaluating the model on its linguistic capability. The scores are discrete ranging from 1 to 5, and each represents "strongly not relevant" to "strongly relevant" respectively. **2. sentiment score of sentence**: this score measures the level of sentence sentiment. It is also discrete, and it ranges from 1 to 5 representing "strongly negative" to "strongly positive" respectively. In this work, we collect manual study scores from 6 subjects in total. From the collected scores, we measure the following metrics:

$$\begin{aligned} sentiment\_relecancy &= \textstyle\sum_i \delta(label_{S^2LCT} != label_{human}) \qquad (1)\\ LC\_relevancy_{AVG} &= \tfrac{1}{\#data} \cdot \textstyle\sum_i LC\_relevancy_i \qquad (2)\end{aligned}$$

The equation 1 represent the number of test cases that their labels assigned from are different between S^{2}LCT and human. Higher number of this metric indicates worse correlation of test oracle that S^{2}LCT generated with human. In addition, the equation 2 represents the average score of the relevancy score between sentence and its associated linguistic capability. higer average score means that higher human-level agreement of the use of sentence for its linguistic capability, resulting in higher suitability of the use of the testcases for evaluating model on the linguistic capability. Given the metrics, we answer RQ1 and RQ2 by the metrics from the equation 1 and equation 2 respectively, thereby, show its ability of S^{2}LCT to understand human intelligence.

RQ3. Recall that a key limitation of CHECKLIST is that its template-based approach that relies on significant manual efforts 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. To measure diversity, we follow the approach presented by Ma et al. [**?**], where the authors measure the coverage of NLP model intermediate states as corner-case neurons. Shiyi: Why is this a good metric for diversity? Specifically, we use the Shiyi: are these metrics we define or are they from ma2018deepgauge? two coverage metrics in existing work [**?**], *boundary coverage* (BoundCov) and *strong activation coverage* (SActCov), as our metrics to evaluate the test suite diversity.

$$\begin{aligned} \text{UpperCornerNeuron}(\mathcal{X}) &= \{n \in N | \exists x \in \mathcal{X} : f_n(x) \in (high_n, +\infty)\};\\ \text{LowerCornerNeuron}(\mathcal{X}) &= \{n \in N | \exists x \in \mathcal{X} : f_n(x) \in (-\infty, low_n)\};\end{aligned} \qquad (3)$$

Eq. 3 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 $high_n, low_n$ are the n^{th} neurons' output bounds on the model training dataset. Eq. 3 can be interpreted as the collection of neurons that emit outputs beyond the model's numerical boundary.

$$\begin{aligned} BoundCov(\mathcal{X}) &= \frac{|UpperCornerNeuron(\mathcal{X})| + |LowerCornerNeuron|}{2 \times |N|}\\ SActCov(\mathcal{X}) &= \frac{|UpperCornerNeuron(\mathcal{X})|}{|N|}\end{aligned} \qquad (4)$$

The formal definition of our coverage metrics are shown in Eq.4, 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 **RQ3**, for each NLP model under test, we first feed its training dataset to compute each neuron's lower and upper bounds. After that, we randomly select 100 Shiyi: Why 100, not all? test cases from S^{2}LCT and CHECKLIST as the test suite and compute the corresponding coverage metrics. Simin: we repeat this process and record both the average and variance value of each coverage

RQ4. To demonstrate that S^{2}LCT can help developers to understand the bugs in the sentiment analysis modes, we conduct experiments to visualize

Shiyi: @Simin: add the setup of the bug explanation case study here.

Implementation Details.

Shiyi: Missing: environment running these experiments.

7 RESULT

8 EXPERIMENTAL RESULTS

8.1 RQ1: S^{2}LCT Sentiment Label Consistency

8.2 RQ2: Correctness of Linguistic Capability Categorization

8.3 RQ3: Test Suite Diversity

Table ?? shows that ....

8.4 RQ4: Use S^{2}LCT for Debugging

REFERENCES

[1] 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

[2] 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

[3] 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

[4] 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

[5] 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

[6] 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

[7] 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, 667–676. https://doi.org/10.1145/1357054.1357160

[8] 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

[9] 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

[10] 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

[11] 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

[12] 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

[13] 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)*.

[14] 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

[15] 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.

[16] 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).

[17] 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

[18] 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

Table 1: Search rules and transformation templates for linguistic capabilities. Shiyi: Add transformation templates. May need to find a better specification language..

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’]}