Programming Language Representation with Semantic-level Structure

Anonymous Author(s)

ABSTRACT

Natural language processing (NLP) technique becomes one of the core techniques for developing text analytics applications. For developing the NLP applications, the applications are required to achieve high reliability before it goes to market. 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 does with held-out accuracy is limited to show its quality because the held-out datasets are often not comprehensive. While the behavioral testing over multiple general linguistic capabilities are employed, it relies on manually created test cases, and is still limited to measure its comprehensive performance for each linguistic capability. In this work, we introduce Auto-CHECKLIST, an NLP model testing methodology. Given a linguistic capability, The Auto-CHECKLIST finds relevant testcases to test the linguistic capability from existing datasets as seed inputs, generates sufficient number of new test cases by fuzzing the seed inputs based on their context-free grammar (Context-free grammar). We illustrate the usefulness of the Auto-CHECKLIST by showing input diversity and identifying critical failures in state-of-the-art models for NLP task. In our experiment, we show that the Auto-CHECKLIST generates more test cases with higher diversity, and finds more bugs.

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, 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Software testing is the essential component of software development process. It evaluates an attribute or capability of the software and determines correctness of the software by examining the behavior of the software and comparing it with the expected outcome. Software testing in the early stage of the development identifies errors, faults, defects, and fixing them saves amount of costs. Therefore, reliable software testing methodology assures quality in software, and it meets needs of the end user. Meanwhile, the trustworthiness in quality of natural language processing (NLP) application also becomes important component for its practical use in real world. Traditionally, the prevalent models of NLP are evaluated via train-validation-test splits. train and validation set is used to train the NLP model and the hold-out set is used for testing by measuring accuracy. In this method, the accuracy is a indicator of the performance of the models, and model with higher accuracy becomes better model.

Despite its simplilcity and usefulness, the testing paradigm often overestimates the performances from the hold-out set. Generalization does not hold in the hold-out dataset and it is likely to introduce specific biases. The biases increase the discrepancy of distribution between the dataset and real world [9]. Therefore, the hold-out set does not represent the real world and the accuracy from the dataset fails to measure comprehensive performrance of the NLP model. In addition, the aggregated accuracy metric does not validate certain behaviors of the model, and, as a consequence, it causes costly to locate where the errors comes from [18]. Therefore, on the subject of the limitation of traditional testing paradigm, a number of methods have been proposed for evaluating multiple perspectives on the NLP model such as robustness on adversarial examples, model coverage and fairness. Especially, Ribeiro et al. intoduced the CHECKLIST, a behavioral testing framework for evaluating NLP model on multiple linguistic capabilities [13]. The CHECKLIST provides predefined task-relevant linguistic capabilities and generate testcases relevant to each linguistic capability. However, the approach only relies on manually generated input templates, thus the template generation becomes expensive and time consuming. In addition, the generated templates are selective and often too simple, and it is limited to provide restricted evaluation of linguistic capabilities. Thus, it does not garauntee the comprehensive evaluation.

Notwithstanding, the evaluating the NLP model on the linguistic capabilities enables comprehensive testing of the model. Each linguistic capability explain the functionality of input and outpus behavior for the NLP model under test. the expected bahavior is determined by combining the input and output scopes specified on the linguistic capability. 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. Such evaluation on the specified functionalities avoids the overestimation of the model performance as it equivalently measures the model performance on each functionality, and the seperate model performance explain distribution of the performaces over the linguistic capabilities. In the end, it provides not only the overall model performance, but also the malfunction facets of the model. However, the prior work currently generated model.

In this paper, we present S^{2}LCT, an automated NLP model evaluation method for comprehensive behavioral testing of NLP models on sentiment analysis task. For each behavior of linguistic capability, S^{2}LCT does not rely on the manual input generation. Instead, it establishes input requirement for evaluating a linguistic capability and finds suitable inputs that meet the requirement from existing public dataset. Therefore, S^{2}LCT increases input diversity and generality. Further, S^{2}LCT applies the fuzzing testing principle to generate inputs by mutating the selected inputs as seed inputs.

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

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.

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

Shiyi: We should motivate the testing of linguistic capabilities. Give background of the work that has been done in CheckList: what capabilities they support and their limitations. Maybe find a motivating example?

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^{2}LCT)* 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 1 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

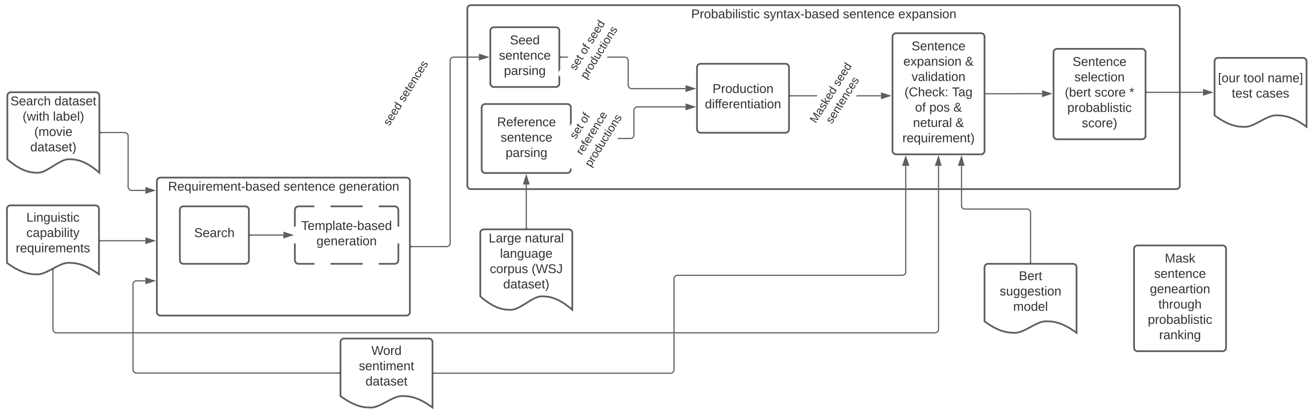
Probabilistic syntax-based sentence expansion
Seed sentence parsing
set of seed productions
Reference sentence parsing
set of reference productions
Production differentiation
Masked seed sentences
Sentence expansion & validation (Check: Tag of pos & netural & requirement)
Sentence selection (bert score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

Figure 1: Overview of S^{2}LCT.

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 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.

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 $random(M, k)$
```

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

Running example. Figure 2 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 2, 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 2 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. 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 EXPERIMENT

In this section, we present experiments to evaluate the effectiveness of our proposed evaluation methodology. In particular, we address the following research questions (RQs):

- **RQ1** : Can S^{2}LCT generate sentences consistent to their sentiment label and their target linguistic capability?
- **RQ2** : Can the S^{2}LCT generated testcases have stronger generalization ability to test sentiment analysis models?
- **RQ3** : Can S^{2}LCT be useful to find root of bug in the sentiment analysis models?

To answer the RQs, we need to show: (I) correctness of testcase to evaluate its target linguistic capability. (II) effect of testcase distribution on finding bugs; (III) degree of execution of a sentiment analysis model on testcases; and (IV) ability to guide to find cause of bug in a sentiment analysis model.

First, S^{2}LCT generates sentences and its label by empirical reasoning derived from linguistic capability statements. Therefore, we meaure acceptibility of the testcases by verifying the relevancy between sentence and label, and between sentence and its linguistic capability. The RQ **RQ1** thereby is answered by verifying the quality of generated testcases. In this work, we conduct manual study and measure the correlation of human labeled testcases and S^{2}LCT labeled testcases. Second, the amount of sentiment analysis model component executed during testing is a critical measurement for assessing quality of software testing. A high software coverage results in higher chances of unidentified bugs in the sentiment analysis model. On the other hand, limited distribution only represents

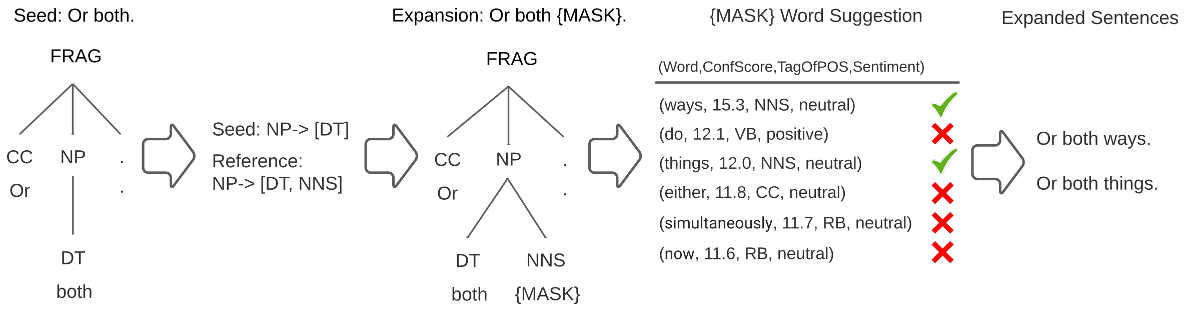
Seed: Or both.
FRAG
CC
NP
.
Or
DT
both
.
Seed: NP-> [DT]
Reference:
NP-> [DT, NNS]
Expansion: Or both {MASK}.
FRAG
CC
NP
.
Or
DT
NNS
both
{MASK}
.
{MASK} Word Suggestion
(Word,ConfScore,TagOfPOS,Sentiment)
(ways, 15.3, NNS, neutral) ✓
(do, 12.1, VB, positive) ✗
(things, 12.0, NNS, neutral) ✓
(either,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

Figure 2: Example of masked sentence generation. Shiyi: Expansion: Or both {MASK}. -> Masked sentence: Or both MASK.

narrow portion of real world covering limited execution behaviors in a sentiment analysis model. It leads to detect bugs within the restricted execution behaviors. Therefore, testcases more equivalent to real world represents more generalized distribution, and it results in high coverage of the sentiment analysis model. Accordingly, the ability of testcases to produce high portion of model coverage needs to be measured. We answer the RQ **RQ3** measuring neural coverage of the neural sentiment analysis model according to the measurements. In addition to the detection of bugs in the model, explaination of the bugs is also important stage of debugging and repairing the model. Therefore, we anwer RQ **RQ3** by showing usefulness of S^{2}LCT by analyzing the sentiment analysis model based on S^{2}LCT and finding root of the bug.

6.1 Experiment Setup

We conduct practical experiments and show the effectiveness of S^{2}LCT.

***Manual Study*.** We sample set of test cases and corresponding linguistic capability randomly sampled from seed sentences and expanded sentences from S^{2}LCT. Subjects are also provided with the sampled testcases. Each subject is asked for scoring relevancy between the linguistic capability of the sentences. The score is discrete and it ranges from 1 to 5 representing "strongly not relevant" to "strongly relevant" respectively. Additionally, the subject scores the relevancy between sentences and their sentiment labels. The score is also discreate and ranges from 1 to 5. It represents "strongly negative" to "strongly positive" respectively. In this work, we collect manual study scores from 8 subjects attended. From the collected scores, we measure the following metrics:

$$Reported\ bugs = \sum_i \delta(label_{S^2LCT} != pred_{model}) \quad (1)$$

$$Incorrect\ inputs = \sum_i \delta(pred_{model} != label_{human}) \quad (2)$$

$$(3)$$

The metrics represents the number of data instance that its label assigned from are different between S^{2}LCT and its model prediction (eq 1) and between the model prediction and human (eq 2). We show the correspondence between S^{2}LCT and human by computing the precision and recall from the reported bugs and incorrect inputs.

***Model Coverage*.** We implement DeepXplore to measure the sentiment analysis model coverage [7]. DeepXploreis the first efficient whitebox testing framework for large-scale deep learning systems. It introduces neuron coverage of a set of test inputs as the ratio of the number of unique activated neurons and the total number of neurons in input deep learning system. In this experiment, we compute the neuron coverage of a test cases from S^{2}LCT and CHECKLIST on the fine-tuned sentiment analysis model of textattack/bert-base-uncased-SST-2 and compare the coverage between S^{2}LCT and CHECKLIST.

***Analysis of Root of bug*.** In this experiment, we analyze a sentiment analysis model based on the test results of the set of S^{2}LCT test inputs and find the root of bug in the model. It shows that S^{2}LCT generated testcases are useful to explain the sentiment analysis model test results and identify the rationale of the bug found. We implement DENASfor this experiment. Rules generated from DENASis interpretable functions mapping certain features of the input to the expected output of a deep neural network system, and the generated rules are considered as the behaviors of the deep neural network system. To anlayze the bug of a sentiment analysis model, we generates rules over the test inputs using DENASand identifies falty rules for the failed test inputs. We manually identify the root of the falty rules and find the root of bug in the end.

7 RESULT

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] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. 2017. DeepXplore. In *Proceedings of the 26th Symposium on Operating Systems Principles*. ACM. https://doi.org/10.1145/3132747.3132785

[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’]}

Table 2: Result of retraining on each linguistic capability and the evaluation of the retrained model.

Approach	RetrainLC	EvalLC	#Fail2Pass	#Fail(BeforeRetrain)	#Pass(BeforeRetrain)	#Fail(AfterRetrain)	#Pass(AfterRet
Retrain:Ours::Test:Checklist	parsing sentiment in (question yes) form	Negated positive with neutral content in the middle	0	860	140	1000	0
Retrain:Ours::Test:Checklist	parsing sentiment in (question yes) form	Short sentences with sentiment-laden adjectives .	0	26	8632	8658	0
Retrain:Ours::Test:Checklist	parsing sentiment in (question yes) form	parsing sentiment in (question, yes) form	1082	1793	7411	1203	8001
Retrain:Ours::Test:Checklist	parsing sentiment in (question yes) form	Author sentiment is more important than of others	0	3741	4787	8528	0
Retrain:Ours::Test:Checklist	parsing sentiment in (question yes) form	Short sentences with neutral adjectives and nouns	1330	1330	386	0	1716
Retrain:Ours::Test:Checklist	parsing sentiment in (question yes) form	Negated neutral should still be neutral	2427	2427	69	0	2496
Retrain:Ours::Test:Checklist	parsing sentiment in (question yes) form	Sentiment change over time, present should prevail	0	1680	6320	8000	0
Retrain:Ours::Test:Checklist	Short sentences with sentiment-laden adjectives	Negated positive with neutral content in the middle	0	860	140	1000	0
Retrain:Ours::Test:Checklist	Short sentences with sentiment-laden adjectives	Short sentences with sentiment-laden adjectives .	0	26	8632	8658	0
Retrain:Ours::Test:Checklist	Short sentences with sentiment-laden adjectives	parsing sentiment in (question, yes) form	1540	1793	7411	7644	1560
Retrain:Ours::Test:Checklist	Short sentences with sentiment-laden adjectives	Author sentiment is more important than of others	0	3741	4787	8528	0
Retrain:Ours::Test:Checklist	Short sentences with sentiment-laden adjectives	Short sentences with neutral adjectives and nouns	1330	1330	386	0	1716
Retrain:Ours::Test:Checklist	Short sentences with sentiment-laden adjectives	Negated neutral should still be neutral	2427	2427	69	0	2496
Retrain:Ours::Test:Checklist	Short sentences with sentiment-laden adjectives	Sentiment change over time, present should prevail	0	1680	6320	8000	0
Retrain:Ours::Test:Checklist	Short sentences with neutral adjectives and nouns	Negated positive with neutral content in the middle	0	860	140	1000	0
Retrain:Ours::Test:Checklist	Short sentences with neutral adjectives and nouns	Short sentences with sentiment-laden adjectives .	0	26	8632	8658	0
Retrain:Ours::Test:Checklist	Short sentences with neutral adjectives and nouns	parsing sentiment in (question, yes) form	1540	1793	7411	7644	1560
Retrain:Ours::Test:Checklist	Short sentences with neutral adjectives and nouns	Author sentiment is more important than of others	0	3741	4787	8528	0
Retrain:Ours::Test:Checklist	Short sentences with neutral adjectives and nouns	Short sentences with neutral adjectives and nouns	1330	1330	386	0	1716
Retrain:Ours::Test:Checklist	Short sentences with neutral adjectives and nouns	Negated neutral should still be neutral	2427	2427	69	0	2496
Retrain:Ours::Test:Checklist	Short sentences with neutral adjectives and nouns	Sentiment change over time, present should prevail	0	1680	6320	8000	0
Retrain:Ours::Test:Checklist	Sentiment change over time present should prevail	Negated positive with neutral content in the middle	0	860	140	1000	0
Retrain:Ours::Test:Checklist	Sentiment change over time present should prevail	Short sentences with sentiment-laden adjectives .	0	26	8632	8658	0
Retrain:Ours::Test:Checklist	Sentiment change over time present should prevail	parsing sentiment in (question, yes) form	1540	1793	7411	7644	1560
Retrain:Ours::Test:Checklist	Sentiment change over time present should prevail	Author sentiment is more important than of others	0	3741	4787	8528	0
Retrain:Ours::Test:Checklist	Sentiment change over time present should prevail	Short sentences with neutral adjectives and nouns	1330	1330	386	0	1716
Retrain:Ours::Test:Checklist	Sentiment change over time present should prevail	Negated neutral should still be neutral	2427	2427	69	0	2496
	Sentiment change	Sentiment change	0	1680	6320	7888	112