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 an NLP application, the application is 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 hold-out dataset. However, evaluating NLP on testset does with hold-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, the testing 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 (CFG). 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, 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Software testing is the cruicial process when developing software. It evaluates an attribute or capability of the software and determines that it meets the requirements by examining the behavior of the software under test. Software testing in the early stage of the development finds bugs, and fixing them saves amount of costs. In addition, reliable software testing methodology ensures software quality to users in that the software meets requirements by verification and validation. Regarding that, NLP application is a branch of artifical intelligence software, and testing NLP application also becomes important process as well.

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. The accuracy is a indicator of the performance of the models. Despite its usefulness, the main limitation of the testing paradigm is that the hold-out set often overestimates the performances. Each dataset comes with specific biases, and the biases increase the discrepancy of distribution between dataset and real-world [?]. The aforementioned accuracy on hold-out set does not consider the discrepancy and it is limited to achieve comprehensive performrance of the NLP model. As a consequence, it is difficult to analyze weekness of the model [?].

On the subject of the limitation of traditional testing paradaigm, a number of methods have been proposed. First, multiple diagnostic datasets for evaluating NLP model were introduced for obtaning generalized evaluation of the NLP model [?]. Not only that, model is evaluated on different aspects such as robustness of the model on adversarial sets [? ? ? ?], fairness [? ?], logical consistancy [?], prediction interpretations [?] and interactive error analysis [?]. Especially, CHECKLIST implements behavioral testing methodolgy for evaluating multiple linguistic capabilities of NLP model [?]. CHECKLIST introduces input-output behaviors of linguistic capabilities and generates behavior-guided inputs for validating the behaviors. It provides comprehensive behavioral testing of NLP models through a number of generated inputs. 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.

In this paper, we present Auto-CHECKLIST, an automated NLP model evaluation method for comprehensive behavioral testing of NLP models on sentiment analysis task. For each behavior of linguistic capability, Auto-CHECKLIST 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, Auto-CHECKLIST increases input diversity and generality. Further, Auto-CHECKLIST applies the fuzzing testing principle to generate inputs by mutating the selected inputs as seed inputs. Fuzzer in Auto-CHECKLIST 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 [?], RoBERTa-base [?] and DistilBERT-base [?]. We show that

ISSTA 2022, 18-22 July, 2022, Daejeon, South Korea
© 2022 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 BACKGROUND

3 RELATED WORK

4 TECHNIQUE AND IMPLEMENTATION

Auto-CHECKLIST generates input sentences with the following phases illustrated in 1: 1. search phase searches seed sentences according to its requirement of linguistic capability, 2. seed parsing phase parses the found seed sentences and extract their context-free grammar, 3. reference parsing phase collects probabilistic context-free grammar from large corpus, 4. production differentiation phase identifies structural expansion candidates for input expansion, and 5.sentence selection phase generates natural expanded sentence. In this section, we provide more details on each phase.

4.1 Search phase

The search phase in Auto-CHECKLIST searches inputs in dataset and selects subset of input sentences in the dataset that meets the linguistic capability requirement. The idea behind this phase is that input distribution of linguistic capability is important to generate inputs relevant to linguistic capability. Linguistic capability explains expected behaviors of NLP model on specific types of input and output. The NLP model is evaluated on how much it performs on the input and output. Thus, linguistic capability introduces the constraints of the input data. Input data from the constrained distribution are only qualified to be used for evaluating the NLP model on the linguistic capability. In addition, diversity in inputs is important to evaluate NLP models on the linguistic capability. Inputs that differ are more likely to cover the NLP model behavior, and more coverage increases trustworthiness of the evaluation. To generate inputs from same distribution on linguistic capability and high diversity of inputs, we estabilish requirements of input and output for each linguistic capability, and find inputs that fulfil the requirements. Given a linguistic capability, a requirement consists of search requirement, transform requirement and expansion requirement. The search requirement describes features and functionalities that we seek to have in inputs. Auto-CHECKLIST check each input if it satisfy the requirement.

Figure 2 shows linguistic capability of "Short sentences with neutral and nouns". To evaluate this linguistic capability, the input is required to be short and have only neutral adjectives, neutral nouns. In addition, the label needs to be neutral. Therefore, all short natural sentences with only neutral adjectives and neutral nouns are available to evaluate NLP models. Next, transform requirement explains how the input and output needs to be tranfromed. Some linguistic capability only accepts heavily limited input distribution, and it is unlikely to be included in searching dataset because of its high structural diversity, thus, finding such sentences is costly. Therefore, our approach is to find inputs by relaxing search requirement and transform the input to match the target requirement of the linguistic capability. In this work, the inputs are transformed by word addition or perturbing the found inputs with linguistic capability dependent templates. The figure 3 shows the example of use of the template requirement. The linguistic capability of "Negated positive with neutral content in the middle" in the figure 3a requires inputs to be the negated positive sentences and the neutral expression in the middle. Rather than searching sentences that match the input distribution of the linguistic capability, the Auto-CHECKLIST search positive and neutral inputs and combine them into negated positive sentences. Figure 3b illustrates template for the linguistic capability. According to the linguistic capability, The value of "sent1" and "sent2" become each searched neutral and positive inputs respectively, and the template completion generates new inputs that matches the target linguistic capability. In addition, the transformation of inputs also produce high diversity in the inputs because of that from initially found inputs. In this paper, we will denote the searched inputs in this phase as seed inputs.

4.2 Seed parsing phase

To expands seed sentence and generate fluent and faithful sentence used for evaluation, Auto-CHECKLIST studies structure of each seed input for its expansion. To extract the structure, this phase takes each seed as input, and the parse tree of the seed input is output. In this phase, Auto-CHECKLIST implements the Berkeley Neural Parser [**? ?**].

4.3 Reference parsing phase

We take a large scale corpus as reference for seed expansion. It is motivated by that large corpus represents data distribution of real world, and the discrepancy between seed and reference leads to determine how the seed can be expanded within real world data distribution. This phase builds probabilistic context-free grammar from the reference corpus. context-free grammar is constructed by parsing sentences in the corpus and extracting production rules. In addition, the probability of context-free grammar is estimated by its frequency over corpus. The outpus of this phase is the constructed probabilistic context-free grammar, and it is compared with seed parse trees. For our implementation, we build the probabilistic context-free grammar from the Penn Treebank corpus dataset.

4.4 production differentiation phase

Given the seed parse tree and reference probabilistic context-free grammar, production differentiation phase suggests structural expansion candidates on the seed input. this phase aims to analyze which structural components and where they can be added into the seed structure for its expansion. To do so, we explore reference production rules comparing it with each production rule used in seed input. This results in the phase described in Algorithm 1. For each production rule in seed inputs ($seed\_prod$), it searches production rules in reference (ref_prod) which it has same non-terminal on the left-hand side ($seed\_prod.lhs == ref\_prod.lhs$) and superset of right-hand side of the seed production rule ($seed\_prod.rhs \subset ref\_prod.rhs$). As we assume that the reference context-free grammar is built from real world data distribution, the elements in the complement set ($ref\_prod.rhs - seed\_prod.rhs$) become an expansion candidate which can be expanded from the $seed\_prod.rhs$ found in real world. In addition, the measure of how consistent the production rule is with the given seed structure is given in its probability of the reference production rule (ref_prob) multiplied by that of parents of $seed\_prod$. The expansion candidate consists of terminal or nonterminal symbols. When there is a phrase-level or clause-level nonterminal symbol, e.g. noun phrase, it needs to

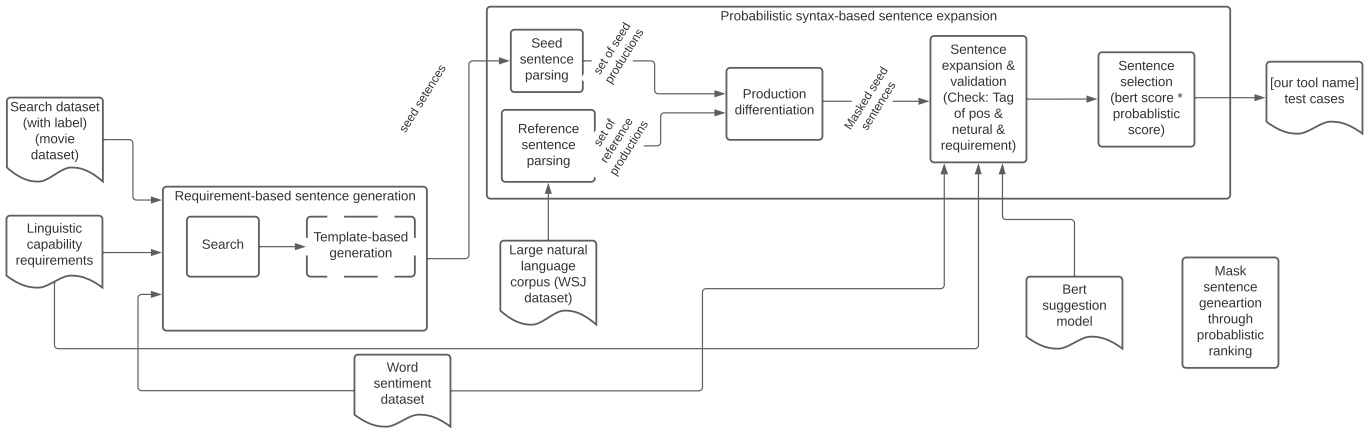
Probabilistic syntax-based sentence expansion
Seed sentence parsing
set of seed productions
Reference sentence parsing
set of reference productions
Production differentiation
Masked seed sentences
Sentence expansion & validation (Check: Tag of pos & netural & requirement)
Sentence selection (bert score * 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 generartion through problablistic ranking
Word sentiment dataset

Figure 1: Overall diagram of Auto-CHECKLIST.

```
{
    "capability": "Vocab_POS",
    "description": "Short sentences with
        neutral adjectives and nouns",
    "search": [
        {
            "length": "<10",
            "include": {
                "POS": [
                    "neutral adjs",
                    "neutral nouns"
                ],
                "word": null
            },
            "exclude": {
                "POS": [
                    "positive adjs",
                    "negative adjs",
                    "positive nouns",
                    "negative nouns"
                ],
                "word": null
            },
            "label": "neutral"
        }
    ]
    ...
}
```

Figure 2: Search requirement on the linguistic capability of "Short sentences with neutral and nouns".

Algorithm 1 Pseudocode of Production differentiation

```
1:  Input: Parse trees of seed sentences S, reference probabilistic context-free grammar R
2:  Output: Set of expanded production rule P
3:  for seed from $S$ do
4:      for each production rule $seed\_prod$ from seed do
5:          $seed\_lhs = seed\_prod.lhs$
6:          $seed\_rhs = seed\_prod.rhs$
7:          for $ref\_rhs$, $ref\_prob$ from $R[seed\_lhs]$ do
8:              if $seed\_rhs$ is superset of $ref\_rhs$ then
9:                  $parent = seed\_prod.parent$
10:                 $prob = ref\_prob$
11:                 while $parent$ is not empty do
12:                     for $par\_rhs$, $par\_prob$ from $R[parent.lhs]$ do
13:                         if $parent.rhs == par\_rhs$ then
14:                             $prob = prob \cdot par\_prob$
15:                             $parent = parent.parent$
16:                             break
17:                         end if
18:                     end for
19:                 end while
20:                 $P.add([ref\_prod, prob])$
21:             end if
22:         end for
23:     end for
24:     Select Top-k $ref\_prod$ along with probabilities in P
25: end for
```

be expanded and replaced with word-level nonterminal or terminal symbols to generate the expansion candidate. The number of feasible replacement is unbounded because of its high degree of freedom. Therefore, in this work, we focus on the expansion candidate with only the word-level nonterminal or terminal symbols for the effectiveness of Auto-CHECKLIST.

```
{
    "description": "Negated positive with
        neutral content in the middle",
    "search": [
        {
            "length": "<20",
            "label": "positive"
        },
        {
            "length": "<20",
            "label": "neutral"
        }
    ],
    "transform": "negate positive",
    "transform_req": [
        {
            "label": "negative"
        }
    ]
}
```

(a) Transform requirement

```
SRL_PHASE_TEMPLATE = {
    "prefix": [
        "Some people think that",
        "Many people agree with that",
        "They think that",
        "You agree with that"
    ],
    "sent1": [],
    "middlefix": ["but I think that"],
    "sent2": [],
}
```

(b) Transform template

Figure 3: Transform requirement (3a) and its template (3b) on the linguistic capability of "Negated positive with neutral content in the middle".

4.5 Sentence selection phase

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:

RQ1 : How effective is our proposed evaluation model for finding failures given a linguistic capability?

RQ2 : How effective is our proposed model for generating diverse test cases?

RQ3 : How effective is test cases generated from our proposed model for detecting diverse type of errors? acc score

RQ4 : How effective is our new test case generation using context-free grammar expansion?

For answering **RQ1** and **RQ2**, we generate test cases and use them for evaluating model on linguistic capabilities. In this experiment, We assess the ability to find failures by anlyzing model's performance on the generated test cases. We also measure the diversity among the generated test cases using similarites among them. Next, we answer **RQ3** by retraining sentiment analysis model with generated test cases and measuring performances. The idea behind this is that more comprehensive inputs becomes closer to real-world distribution and addresses more type of errors. Therefore, it leads to improve the model performance. In this experiment, We retrain the model and compare performances of the retrained model. Not only that, we conduct ablation study of context-free grammar expansion to understand the its impact in our approach.

6.1 Experiment Setup

Seed Input Selection. For each linguistic capability, we first search all sentences that meet its requirement. Among found sentences, we randomly select 10 sentences due to memory constraint.
Word Sentiment. we extract sentiments of words using the SentiWordNet [?]. The SentiWordNet is a publicly available lexical resource of words on Wordnet with three numerical scores of objectivity, positivity and negativity. Sentiment word labels from the scores are classified from the algorithm from Mihaela et al. [?].
Context-free grammar Expansion. We build a reference Context-free grammar of natural language from the English Penn Treebank corpora [? ?]. The corpus is sampled from 2,499 stories from a tree year Wall Street Journal collection The Treebank provides a parsed text corpus with annotation of syntactic and semantic structure. In this experiment We implement the treebank corpora available through NLTK, which is a suite of libraries and programs for Natural language processing for English. In addition, we parse the seed input using into its CFG using the Berkeley Neural Parser [? ?], a high-accuracy parser with models for 11 languages. The input is a raw text in natural language and the output is the string representation of parse tree. Next after comparing CFGs between reference and seed input, we randomly select 10 expansions for generating templates due to memory constraint.
Synonyms. Auto-CHECKLIST searches synonyms of each token from synonym sets extracted from WordNet using Spacy open-source library for NLP.
Models. We evaluate the following sentiment analysis models via Auto-CHECKLIST: BERT-base [?], RoBERTa-base [?] and DistilBERT-base [?]. These models are fine-tuned on SST-2 and their accuracies are 92.43%, 94.04% and 91.3%.
Retraining. We retrain sentiment analysis models. we split Auto-CHECKLIST generated test cases into train/validation/test sets with the ratio of 8:1:1. The number of epochs and batch size for retraining are 1 and 16 respectively.

7 RESULT

REFERENCES