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

1 INTRODUCTION

Software testing is the crucial 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 [13]. 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 where the errors comes from [23].

On the subject of the limitation of traditional testing paradigm, a number of methods have been proposed. First, multiple diagnostic datasets for evaluating NLP model were introduced for obtaining generalized evaluation of the NLP model [22]. Not only that, model is evaluated on different aspects such as robustness of the model on adversarial sets [2, 5, 16, 19], fairness [12, 18], logical consistancy [14], prediction interpretations [15] and interactive error analysis [23]. Especially, CHECKLIST implements behavioral testing methodolgy for evaluating multiple linguistic capabilities of NLP model [17]. 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 [4], RoBERTa-base [10] and DistilBERT-base [20]. We show that

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

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.* [22] 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 [2, 5, 16, 19], fairness evaluation [12, 18], logical consistency evaluation [14], prediction interpretations [15] and interactive error analysis [23] 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. [17] 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^2LCT, 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^2LCT, 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) 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?) 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 unrealistic suggested words or use a rare syntactic structure Shiyi: is this the right motivation to do selection?, we use a heuristic (i.e., combination of the confidence score from the NLP recommendation model and the frequency of the syntactic productions) to select the more realistic and frequent expanded sentences into S^2LCT's test suite.

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

4.1 Specification-based Seed Generation

The seed generation phase of S^2LCT starts by searching sentences in a real-world dataset that match the rules defined in the linguistic capability specification, and then transforming the matched sentences using templates to generate seed sentences that conform to individual linguistic capabilities. The reasons for this design choice are twofold. First, while generally judging which linguistic capability any sentence falls into and which label it should have is infeasible, there exist simple rules and templates to allow classifying the resulting sentences into individual linguistic capabilities and with the correct labels, with high confidence. This enables us to test each linguistic capability individually. Second, searching from a real-world dataset ensures that the sentences used as test cases for testing linguistic capabilities are realistic and diverse. The diverse test cases are more likely to achieve a high coverage of the target model's functionality in each linguistic capability, thus detecting more errors.

Table 1 shows the search rules and the transformation templates of all 11 linguistic capabilities we implemented in S^2LCT. Shiyi: will revise the rest of 3.1 after talking to Jaeseong... I think we can show all the specifications.

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. In this work, the sentiment of the words for the search are classified based on the sentiment scores from SentiWordNet [1], a publicly available English sentiment lexicons. It provides lexical sentiment scores and the sentiment word labels are categorized by implementing the rules in [3]. 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

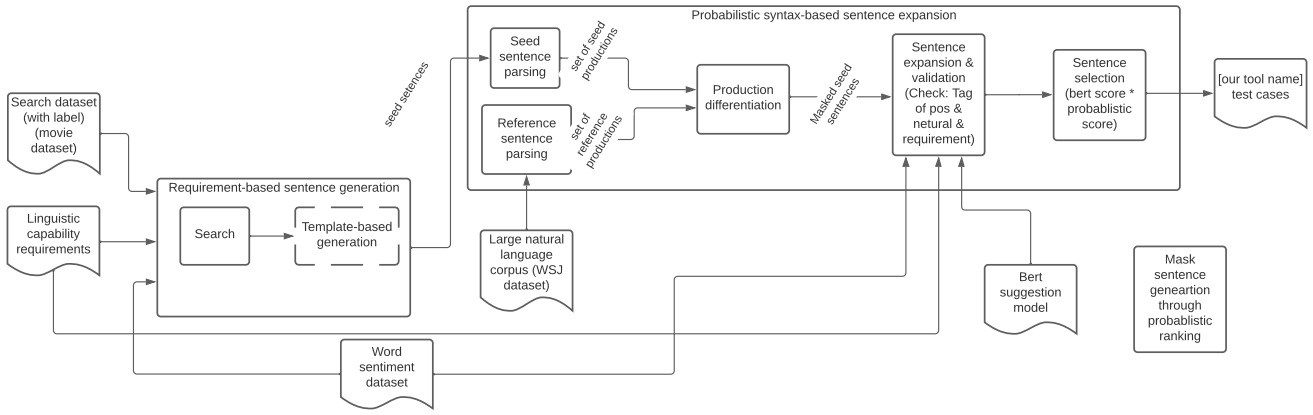

Figure 1: Overview of S^{2}LCT.

```
{
    "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".

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 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 **??** shows how masks are identified for each seed sentence. It takes the parse trees of the seeds, generated by the Berkeley Neural Parser [6, 8], and a reference context-free grammar (CFG) from the Penn Treebank corpus dataset [] as inputs. The reference CFG is learned from a

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: {length: <10; include: neutral adjs & neutral nouns; exclude: pos adjs & neg adjs & pos nouns & neg nouns; label: neutral} transform: -
LC2: Short sentences with sentiment-laden adjectives	search: {length: <10; include: pos adjs; exclude: 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: -
LC3: Sentiment change over time, present should prevail	search: {label: pos} | {label: neg} transform:{['Previously, I used to like it saying that','Last time, I agreed with saying that','I liked it much as to say that']+[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]+['but', 'although', 'on the other hand']+['now I like it.']}
LC4: Negated negative should be positive or neutral	search: {start: [This, That, These, Those] X [is, are]; label: neg} transform: negation of demonstratives
LC5: Negated neutral should still be neutral	search: {start: [This, That, These, Those] X [is, are]; label: neutral} transform: negation of demonstratives
LC6: Negation of negative at the end, should be positive or neutral	search: {label: neg} transform:{['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: {length: <20; label: pos} | {length: <20; label: neutral} transform:{['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: {label:pos} | {label: neg} transform:{['Some people think that', 'Many people agree with that', 'They think that', 'You agree with that']+[pos sent | neg sent]+['but I think that']+[neg sent | pos sent]}
LC9: Parsing sentiment in (question, yes) form	search: {label: pos} | {label: neg} transform:{['Do I think that', 'Do I agree that']+[pos sent | neg sent]+['? yes']}
LC10: Parsing positive sentiment in (question, no) form	search: {label: pos} transform:{['Do I think that', 'Do I agree that']+[pos sent]+['? no']}
LC11: Parsing negative sentiment in (question, no) form	search: {label: neg} transform:{['Do I think that', 'Do I agree that']+[neg sent]+['? no']}

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.

```
{
    "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".

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, we randomly select k masked sentences for the next sentence expansion and validation phase.

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 masked sentences **S_exp**
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* · *par_prob*
15: *parent* = *parent.parent*
16: break
17: **end if**
18: **end for**
19: **end while**
20: *P.add*([*seed_prod*, *ref_prod*, *prob*])
21: **end if**
22: **end for**
23: **end for**
24: **if** *P* is not empty **then**
25: Select Top-k *ref_prod* along with probabilities in P
26: **for** each *seed_prod*, *ref_prod* in the Top-k selected P **do**
27: replace *seed_prod* with *ref_prod* in *seed*
28: replace expanded component in *ref_prod* with *MASK* token in *seed* sentence
29: *S_exp.add*([*ref_prod*, *expandedsentence*])
30: **end for**
31: **end if**
32: **end for**

Shiyi: Add justification: why we need to select k masked sentences (performance?) and why random makes sense.

Figure 4 shows an example using Algorithm **??** to generate a masked sentence. The sentence "Or both." is a seed of Shiyi: which? linguistic capability. Shiyi: Not sure about the quality of the example. Maybe use a better seed for this example? 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 on the right. 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 []. The BERT model suggests word for the

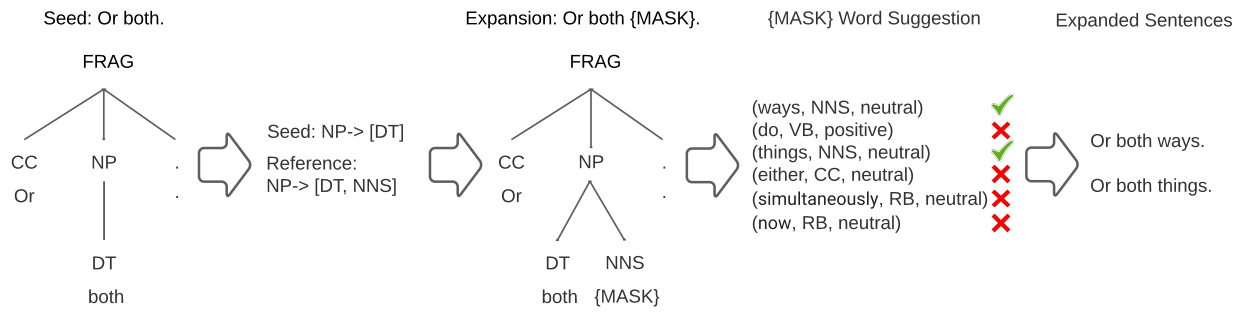

Figure 4: Example of masked sentence generation. Shiyi: Expansion: Or both MASK. -> Masked sentence: Or both MASK.

mask symbol according to its context Shiyi: what does context mean here? around in sentence. 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? 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?

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 4, the masked symbol is a "NNS" (i.e., plural noun); thus, the suggested word must also be a "NNS". Shiyi: Say how we obtain the PoS tag of a suggested word. Shiyi: Also, expand Figure 4 to actually include the example of the suggested words? That way, we can easily discuss each validation criteria along with the example. Second, we require that the sentiment of the expanded sentence is 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? Third, the expanded sentence must still conform to the specification of the seed's linguistic capability specification. Shiyi: It is not clear to me how we check with search rules and transformation templates.

4.2.3 Sentence Selection. Shiyi: Use BERT score as heuristic to select expanded sentences. Motivate why.

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 [1]. 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. [3].

Context-free grammar Expansion. We build a reference Context-free grammar of natural language from the English Penn Treebank corpora [11, 21]. 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 [7, 9], 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 [4], RoBERTa-base [10] and DistilBERT-base [20]. 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

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

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

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

[4] 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

[5] 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

[6] 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

[7] 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

[8] 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

[9] 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

[10] 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

[11] Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Comput. Linguist.* 19, 2 (jun 1993), 313–330.

[12] 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

[13] 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

[14] 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

[15] 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

[16] 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

[17] 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)*.

[18] 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

[19] 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.

[20] 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).

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

[22] 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

[23] 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