

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, 5 pages. <https://doi.org/10.1145/nnnnnnnn.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 artificial 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 performance of the NLP model. As a consequence, it is difficult to analyze weakness of the model [23].

On the subject of the limitation of traditional testing paradigma, 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 consistency [14], prediction interpretations [15] and interactive error analysis [23]. Especially, CHECKLIST implements behavioral testing methodology 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 guarantee 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

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

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/nnnnnnnn.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 establish 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. 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 transformed. 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 [6, 8].

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 output 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

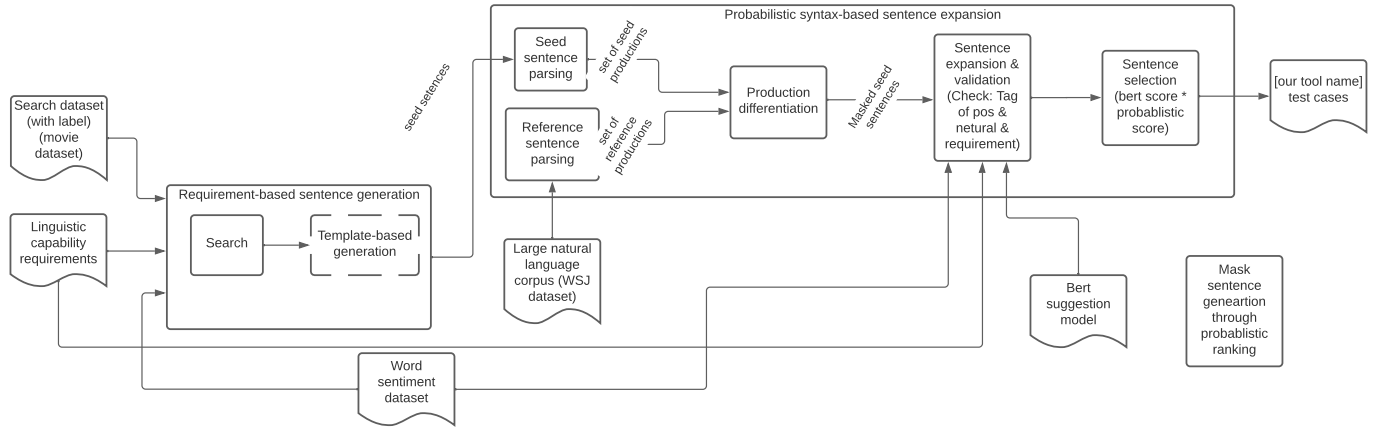


Figure 1: Overall diagram of Auto-CHECKLIST.

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  $R[parent.lhs]$ 
13:            do
14:              if  $parent.rhs == par\_rhs$  then
15:                 $prob = prob \cdot par\_prob$ 
16:                 $parent = parent.parent$ 
17:                break
18:              end if
19:            end for
20:          end while
21:           $P.add([seed\_prod, ref\_prod, prob])$ 
22:        end if
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  $P$ 
27:        do
28:          replace  $seed\_prod$  with  $ref\_prod$  in  $seed$ 
29:          replace expanded component in  $ref\_prod$  with
30:            MASK token in  $seed$  sentence
31:           $S_{exp}.add([ref\_prod, expandedsentence])$ 
32:        end for
33:    end if
34:  end for

```

```

{
  "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".

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

```

349 {
350   "description": "Negated positive with
351     neutral content in the middle",
352   "search": [
353     {
354       "length": "<20",
355       "label": "positive"
356     },
357     {
358       "length": "<20",
359       "label": "neutral"
360     }
361   ],
362   "transform": "negate positive",
363   "transform_req": [
364     {
365       "label": "negative"
366     }
367   ]
368 }
369
370
371
372 (a) Transform requirement
373
374 SRL_PHASE_TEMPLATE = {
375   "prefix": [
376     "Some people think that",
377     "Many people agree with that",
378     "They think that",
379     "You agree with that"
380   ],
381   "sent1": [],
382   "middlefix": ["but I think that"],
383   "sent2": [],
384 }
385
386 (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”.

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. Lastly, the expanded component is replaced with the mask token for the next phase. The example of the expansion is illustrated in Figure 4. The “NP->[DT]” is queried into reference, and “NP->[DT,NNS]” is identified as its expansion candidate since the right-hand side of “NP->[DT,NNS]” is superset of that of “NP->[DT]”. the component of NNS is replaced with mask token in sentence-level. Therefore, the “Or both {MASK}” is suggested for the next phase.

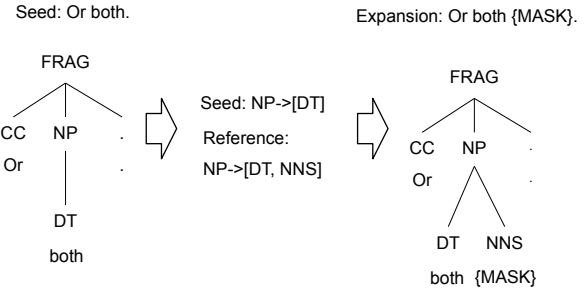


Figure 4: Expansion of the seed sentence “Or both.”. For a production rule in seed “NP->[DT]” on the left, the production rule of “NP->[DT, NNS]” is found in reference. Thus, the component NNS can be expanded in the seed and “NP->[DT]” is replaced with “NP->[DT, NNS]” and it generates new expanded sentence on the right.

Next, the masked tokens in the expanded masked sentences are estimated in this phase. For the mask token prediction, we use the BERT pretrained model. The BERT model suggests word for the mask token according to its context around in sentence. The suggested words are validated by three criteria. First, The tag of POS must be matched with that suggested from the production differentiation phase. In the example in the Figure 4, the mask token comes from the structural component of NNS, plural noun. Therefore, the BERT suggested words must be also tagged with the NNS. Accordingly, every words of the NNS are only available. Second, Auto-CHECKLIST focuses on the sentiment analysis task, and it assume that the suggested words must not change sentiment of its input and must preserve its consistency of original sentiment label. Therefore, we only accept the neutral words for the expansion. Third, the expanded sentences with the BERT suggested words must be appropriate for evaluating NLP models on the target linguistic capability, and the sentences must pass the requirement of the linguistic capability. In this work, the BERT suggestions are validated by the three criteria.

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 analyzing model’s performance on the generated test cases. We also measure the diversity among the generated test cases using similarities 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, Adtefan VLAČduĂescu, 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 Tulio 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](http://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 Minh 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>