Programming Language Representation with Semantic-level Structure

Anonymous Author(s)

ABSTRACT

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

ACM Reference Format:

Anonymous Author(s). 2022. Programming Language Representation with Semantic-level Structure. In *Proceedings of ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA 2022).* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Natural language processing (NLP) applications are growing exponentially. As a result, trustworthiness in the quality of NLP applications has become critical for its practical use in the real world. Therefore, quality assurance of NLP applications is an essential process in the software development processes. Researchers aim to improve the current practices of testing NLP models from three perspectives: (i) *Test input generation*, (ii) *Automated test oracle*, and (iii) *Meaningful quality metrics.*

Test input generation. Currently, most testing of an NLP model reuses existing large textual corpus as the testing dataset to evaluate the model. This practice will often overestimates the model performances from the hold-out set [16, 18, 23]. The overestimation comes from the discrepancy between the distribution of the used dataset and the actual data distribution in real world. Oftentimes, the hold-out dataset is not representative and is likely to introduce specific biases, leading to the decreased robustness of NLP models. In this regard, prior works have proposed techniques for testing the robustness of NLP models by crafting adversarial examples and attacking a model with them intentionally [2, 10, 22, 25].

Automated test oracle. The current testing practice requires manual work for labelling the test oracles of the hold-out data. The manual work is costly in terms of time consumption and its impact on market price. Therefore, it necessitates automated test oracle generation for improving the testing process of NLP models. However, automatically generated test oracles may not always be feasible; predicting the correct test oracles remains one of main challenges. Along with this, software metamorphic testing approach has been introduced [27] to alleviate the test oracle discrepancy between expected and observed test oracles.

Meaningful quality metrics. Traditionally, the quality of NLP models are represented by numbers in quality metrics. Especially, accuracy (i.e., the fraction of outputs that the model correctly predicts) is the most widely used metric for assessing the quality of classification models. Generally, higher accuracy number suggests better quality of a model. However, all NLP models have their strength and weakness and forcing aggregation statistics into a single number makes the users difficult to assess the capabilities of NLP models. Not to mention localizing and fixing the bugs found from the hold-out set (i.e., testing dataset, as opposed to training dataset). Therefore, this forced aggregation method not only fails to validate the linguistic capability of the model, but it also makes the localization of the causes of the inaccuracy more costly [31]. To address this limitation, Ribeiro et al. introduced CHECKLIST, a behavioral testing framework for evaluating NLP model on multiple linguistic capabilities [23]. CHECKLIST defines task-relevant linguistic capabilities and generates test cases for each linguistic capability.

However, none of the above approaches satisfy all three requirements at the same time. First, adversarial testing approaches merely focus on evaluating model robustness. They measure how sensitive the models are to input perturbations while do not evaluate linguistic functionalities. Second, the metamorphic testing approach is required to understand the characteristics of metamorphic relations between inputs and outputs. However, finding these remains one of the most challenging problem in metamorphic testing [27]. Along with this, such relation in textual data in NLP domain has not been evaluated despite its importance. Third, CHECKLIST relies on manually generated input templates, which need to be preset before test input generation. Consequently, CHECKLIST templates are distributed in a limited range of their structures. This restricts CHECKLIST's ability to comprehensively test the linguistic capabilities.

Despite CHECKLIST's limitations, assessing the quality of NLP models through the linguistic capabilities is a promising direction.

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

Each linguistic capability explains the functionality of the input and output behavior for the NLP model under test. Typically, it describes certain type of inputs and outputs observed in real world for the target NLP task ranging from simple to complicated behaviors, so the model developers can better understand the capabilities and potential issues of the NLP models. For example, a linguistic capability of "Negated neutral should still be neutral" measures how accurately the sentiment analysis model understands the negative neutral input as an neutral sentiment [23]. Therefore, it requires the sentiment analysis model to output neutral sentiment on the negated neutral input. Such methodology of evaluation on the specified functionalities avoids the overestimation of the model performance as it equivalently measures the model performance on each functionality. In the end, testing through linguistic capabilities provides not only the overall model performance, but also the malfunction facets of the model.

To satisfy all three requirements mentioned above, we present S^{2}LCT, an automated NLP model evaluation method for comprehensive behavioral testing of NLP models on sentiment analysis task. There are three main challenges that S^{2}LCT overcomes to satisfy all three aforementioned requirements.

C1 The test suite should cover diverse syntactic structures;
C2 Each test case should be categorized into a linguistic capability;
C3 The label of each test case should be automatically and accurately defined.

C1. The first challenge comes with the second and third challenges of maintaining the label and linguistic capability of the test sentence while making the syntactic structure of the sentence more diverse. In addition, text have a higher order of structures and structures are obscured by word usage []. To address the challenge, S^{2}LCT establishes specifications for evaluating a linguistic capability and searches suitable sentences that satisfy the specification from existing public dataset. In this process, S^{2}LCT generates new inputs by mutating the searched sentences, used as seed inputs. S^{2}LCT expands seed input grammar structures and determines its available part-of-speech to maintain structural naturalness.
C2. Suitability of test sentences for evaluating NLP model on a linguistic capability is obtained from its high relevancy to the linguistic capability. Relevancy between a test sentence and its linguistic capability is challenging to be maintained during the transformation and expansion of the test sentence. This is because the linguistic capability is defined on a specific mixture of syntaxe and semantics of the sentence, and there exists no automatic way to check the consistency between each sentence with the semantics specified in corresponding linguistic capability due to the inherent ambiguity of natural language sentences. To address the difficulty, S^{2}LCT implements search rules and the transformation templates of linguistic capabilities. In addition, anlyzing parse tree of seed sentence is used for its expansion identification.
C3. Last challenge is on automated test oracle. For NLP tasks, test oracle is determined by understanding meaning from texts. This requires domain knowledge for different NLP tasks In this work, as a first step, we consider sentiment analysis as the NLP task for the models under test. S^{2}LCT obtain the appropriateness of test oracle by implementing domain-specific knowledge on word sentiment dataset and word suggestion model pre-trained on large corpus for validation of generated sentence and its oracle.

```
air_noun = [
    'flight', 'seat', 'pilot', 'staff',
    'service', 'customer service', 'aircraft', 'plane',
    'food', 'cabin crew', 'company', 'airline', 'crew'
]
pos_adj = [
    'good', 'great', 'excellent', 'amazing',
    'extraordinary', 'beautiful', 'fantastic', 'nice',
    'incredible', 'exceptional', 'awesome', 'perfect',
    'fun', 'happy', 'adorable', 'brilliant', 'exciting',
    'sweet', 'wonderful'
]
neg_adj = [
    'awful', 'bad', 'horrible', 'weird',
    'rough', 'lousy', 'unhappy', 'average',
    'difficult', 'poor', 'sad', 'frustrating',
    'hard', 'lame', 'nasty', 'annoying', 'boring',
    'creepy', 'dreadful', 'ridiculous', 'terrible',
    'ugly', 'unpleasant'
]

t = editor.template('{it} {air_noun} {be} {pos_adj}.',
                    it=['The', 'This', 'That'], be=['is', 'was'],
                    labels=2, save=True)
t += editor.template('{it} {be} {a:pos_adj} {air_noun}.',
                     it=['It', 'This', 'That'], be=['is', 'was'],
                     labels=2, save=True)
t += editor.template('{i} {pos_verb} {the} {air_noun}.',
                     i=['I', 'We'], the=['this', 'that', 'the'],
                     labels=2, save=True)
t += editor.template('{it} {air_noun} {be} {neg_adj}.',
                     it=['That', 'This', 'The'], be=['is', 'was'],
                     labels=0, save=True)
t += editor.template('{it} {be} {a:neg_adj} {air_noun}.',
                     it=['It', 'This', 'That'], be=['is', 'was'],
                     labels=0, save=True)
t += editor.template('{i} {neg_verb} {the} {air_noun}.',
                     i=['I', 'We'], the=['this', 'that', 'the'],
                     labels=0, save=True)
```

Figure 1: Example of CHECKLIST templates on linguistic capability of "Short sentences with sentiment-laden adjectives".

We demonstrate its generality and utility as a NLP model evaluation tool by evaluating well-known sentiment analysis models: BERT-base [5], RoBERTa-base [13] and DistilBERT-base [26]. We show that ...

2 BACKGROUND

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

3 SPECIFICATION- AND SYNTAX-BASED LINGUISTIC CAPABILITY TESTING

We design and implement a new NLP model testing method, *Specification- and Syntax-based Linguistic Capability Testing (S^{2}LCT)*, that automatically generate test cases with oracles to test the robustness of sentiment analysis models. S^{2}LCT addresses all three challenges discussed above.

Figure 2 shows the overview of S^{2}LCT, which consists of two phases. The *specification-based seed generation* phase performs rule-based searches from a real-world dataset 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., negating a sentence) are defined in the *linguistic capability specifications*, which guarantee that each resulting seed conforms to a specific linguistic capability (**C2**) and is labelled correctly (**C3**).

The *syntax-based sentence expansion* phase expands the seed sentences with additional syntactic elements (i.e., words) to cover many real-world syntactic structures (**C1**). 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), **C2** and **C3** are still satisfied. Last, because some validated sentences may include unacceptable suggested words given the context, 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.

3.1 Specification-based Seed Generation

The seed generation phase of S^{2}LCT starts by searching sentences in a real-world dataset that match the rules defined in the linguistic capability specification, and then transforming the matched sentences using templates to generate seed sentences that conform to individual linguistic capabilities. The reasons for this design choice are twofold. First, while generally judging which linguistic capability any sentence falls into and which label it should have is infeasible, there exist simple rules and templates to allow classifying the resulting sentences into individual linguistic capabilities and with the correct labels, with high confidence. This enables us to test each linguistic capability individually. Second, searching from a real-world dataset ensures that the sentences used as test cases for testing linguistic capabilities are realistic and diverse. The diverse test cases are more likely to achieve a high coverage of the target model's functionality in each linguistic capability, thus detecting more errors. In this phase, S^{2}LCT first search and selects sentences applicable to the linguistic capability in a given real-world dataset with search rules. In case that the search rules only fulfill portion of the linguistic capability specifications, the selected sentences are not yet appropriate to become seed, we transform the selected sentences into seed sentences using the heuristic templates. Table 1 shows the search rules and the transformation templates of all 11 linguistic capabilities we implemented in S^{2}LCT. The first column shows the linguistic capability type and its description, and the second column shows the search rule and transformation template used in each linguistic capability. For LC1 and LC2, the NLP models are evaluated in the scope of short sentences with selective sentiment words. It does not require any transformation because the search rule alone is sufficient to conform to the linguistic capabilities. On the other hand, search rules of LC3 to LC11 are not enough to match their linguistic capability specification, thus S^{2}LCT uses heuristic templates to conform the searched sentences to the linguistic capability. For example, in LC3's transformation template, the searched sentences become seeds by preturbing them with the defined templates. In LC4's transformation template, the searched demonstrative sentences are negated.

3.2 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

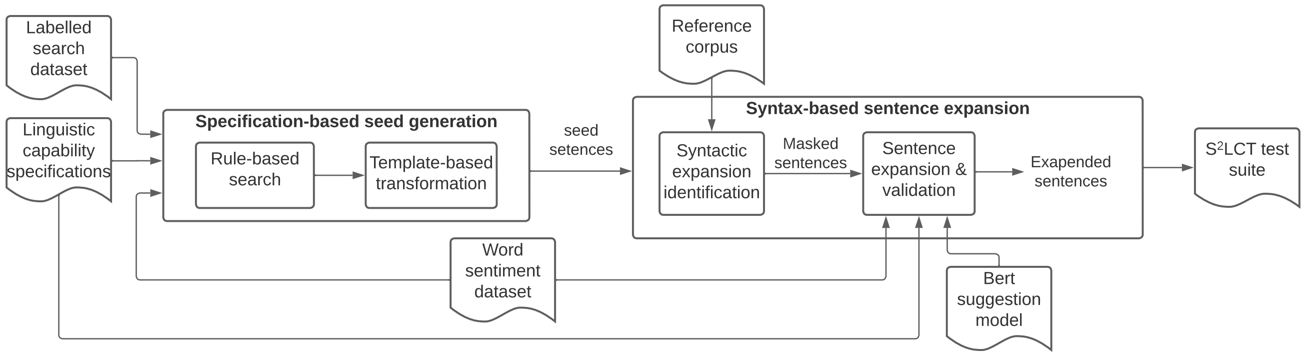

Figure 2: Overview of S^{2}LCT.

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.

3.2.1 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 [11, 12], and a reference context-free grammar (CFG) from the Penn Treebank corpus dataset [15] as inputs. The reference CFG is learned from a large dataset [28] 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.

Running example. Figure 3 shows an example using Algorithm 1 to generate a masked sentence. The sentence "Or both." is a seed of linguistic capability of "Short sentences with neutral adjectives and nouns". The tree on the left shows the parse tree of this seed; it consists of two productions: "FRAG->[CC, NP, .]" and "NP->[DT]". When matching the left-hand-side non-terminal of the second production (i.e., "NP") in the reference CFG, we found that it includes a production "NP->[DT, NNS]" which has an additional symbol "NNS" on the right-hand-side. The algorithm thus expands the parse tree with this symbol, shown in the second tree. The masked sentence "Or both {MASK}." is the result of the left-to-right traversal of this expanded parse tree.

3.2.2 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 [5]. 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. 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

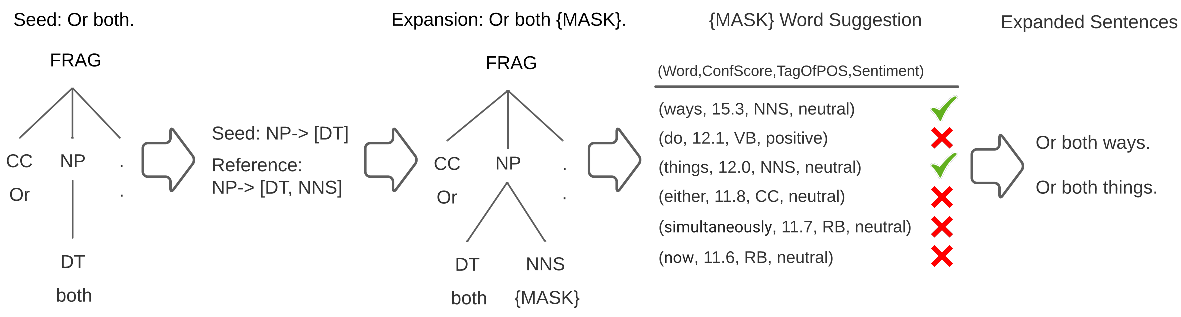

Figure 3: Example of masked sentence generation. Shiyi: Expansion: Or both {MASK}. -> Masked sentence: Or both MASK.

satisfy the linguistic capability specification. Therefore, we perform validation on the suggested words and only accept them if the following three criteria are met.

First, the PoS tag of the suggested word must match the PoS tag of the expanded symbol in the parse tree. For the example in Figure 3, the masked symbol is a “NNS" (i.e., plural noun); thus, the suggested word must also be a “NNS". In this work, we use SpaCy, a free open-source library for natural language processing, for extracting PoS tags for each suggested word.

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.

Third, we additionally verify that the expanded sentences satisfied the same search rules for the seed sentence in LC1 and LC2. This criteria cannot be applied to other linguistic capabilities because they have additional transformation templates.

In this work, out of the BERT suggested words, we randomly selects words that matches their PoS tag of the expanded symbol in the parse tree. The words that meets the second and third criteria are finally employed for the expanded sentences.

Running example. The third step in Figure 3 shows the words suggested by BERT. For this masked sentence, BERT suggested six words. Each word is associated with the confidence score provided by BERT, the PoS tag, and the sentiment. Among the six words, only “ways” and “things” are validated by S^{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.

4 EXPERIMENTAL SETUP

In this section, we present the setup of the experiments to evaluate the effectiveness of S^{2}LCT. We address the following research questions (RQs):

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

RQ1 : Can S^{2}LCT generate consistent test sentence and its oracle?
RQ2 : Are S^{2}LCT generated test cases relevant to be used for their linguistic capability evaluation?
RQ3 : Can S^{2}LCT generate more diverse test cases than CHECKLIST?
RQ4 : Can S^{2}LCT be useful to find root causes of bugs in the sentiment analysis models?

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

4.1 Experimental Subjects

NLP Models & Dataset. We evaluate our approach on three leaerning-based sentiment analysis models implemented in Transformer library from the Hugging Face centralized Model Hub.[1]: `BERT-base` (textattack/bert-base-uncased-SST-2), `RoBERTa-base` (textattack/roberta-base-SST-2), and `DistilBERT-base` (distilbert-base-uncased-finetuned-sst-2-english). They are pre-trained on English language using a masked language modeling (MLM) objective, and fine-tuned on sentiment analysis task. In this experiment, we use Stanford Sentiment Treebank version 1 dataset for searching seeds and expanding the seeds in S^{2}LCT. The Stanford Sentiment Treebank version 1 is a corpus of movie review, and it consists of 11,855 single sentences with its sentiment score. As original dataset suggests, we split the score range into [0, 0.4], (0.4, 0.6] and (0.6, 0.8] for assigning negative, neutral and positive labels respectively. In addition, we use SentiWordNet as word sentiment dataset [1]. SentiWordNet is a publicly available English sentiment lexicons. It provides lexical sentiment scores and the sentiment word labels are categorized by implementing the rules in [4].

Comparison Baselines. We compare our approach with CHECKLIST[2], which is a manual template-based approach to generate test cases. In this experiment, we used the CHECKLIST released sentiment analysis test cases which are generated from its publicly available jupyter notebook implementation.

[1]https://huggingface.co/models
[2]https://github.com/marcotcr/checklist

4.2 Experimental Process

RQ1 and RQ2. As described in Section 3, S^2LCT generates test cases in two steps: specification-based seed generation and syntax-based sentence expansion. These automated steps may generate seed/expanded sentences marked with incorrect sentiment labels or categorized into wrong linguistic capabilities. For example, the search rule and template defined in a linguistic capability may not always generate seed sentences in that capability or with the correct label. To answer RQ1 and RQ2, we perform a manual study to measure the correctness of the sentiment labels and linguistic capabilities associated with the seed/expanded sentences, produced by S^2LCT.

In the manual study, we randomly sample three sets of pairs of seed sentences and corresponding linguistic capability from seed test cases. For each set, we also select expanded sentences that S^2LCT generated from the formerly sampled seed sentences. In this experiment, each set has 100 sentences (50 from seed sentences and 50 from expanded sentences) and 200 sentences, in total, are used for the manual study. For each sampled set, two subjects are provided with the same set of sampled sentences. The subjects are asked for scoring the two following: **1. relevancy score between sentence and its associated linguistic capability**: this score measures the amount of appropriateness of the use of sentence for evaluating the model on its linguistic capability. The scores are discrete ranging from 1 to 5, and each represents “strongly not relevant” to “strongly relevant” respectively. **2. sentiment score of sentence**: this score measures the level of sentence sentiment. It is also discrete, and it ranges from 1 to 5 representing “strongly negative” to “strongly positive” respectively. In this work, we collect manual study scores from 3 subjects in total. In case of different scores with a same sentence from different subjects, we used their average score as a score of the sentence. From the collected scores, we measure the following metrics:

$$sentiment\_relecancy = \sum_i \delta(label_{S^2LCT} != label_{human}) \quad (1)$$

$$LC\_relevancy_{AVG} = \frac{1}{\#Data\cdot} \cdot \sum_i Norm(LC\_relevancy_i) \quad (2)$$

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

RQ3. Recall that a key limitation of CHECKLIST is that its template-based approach that relies on significant manual efforts may not generate test cases that comprehensively cover the sentences in a linguistic capability. S^2LCT, instead, automatically generates test cases based on a search dataset and the syntax in a large reference corpus. We expect S^2LCT can generate a more diverse test suite than CHECKLIST, and achieve more probable evaluation results. We first evaluate the three sentiment analysis models by testing them on the test cases that S^2LCT generates. Next, to measure diversity, we follow the approach presented by Ma et al. [14], where the authors measure the coverage of NLP model intermediate states as corner-case neurons. Because the matrix computation of intermediate states impacts NLP model decision-making, a test suite that covers a greater number of intermediate states can represent more NLP model decision-making, making it more diverse. Specifically, we used two coverage metrics in existing work [14], *boundary coverage* (BoundCov) and *strong activation coverage* (SActCov), as our metrics to evaluate the test suite diversity. TODO: It is worth noting that a test sample with a statistical distribution similar to the training data would rarely be found in the corner case region. As a result, covering a larger corner case region means that the test suite is more likely to be buggy.

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

Eq. 3 shows the formal definition of the corner-case neuron of the NLP model $f(\cdot)$, where $\mathcal{X}$ is the given test suite, N is the number of neurons in model $f(\cdot)$, $f_n(\cdot)$ is the n^{th} neuron's output, and $high_n, low_n$ are the n^{th} neurons' output bounds on the model training dataset. Eq. 3 can be interpreted as the collection of neurons that emit outputs beyond the model's numerical boundary.

$$\begin{aligned} BoundCov(\mathcal{X}) &= \frac{|UpperCorner(\mathcal{X})| + |LowerCorner(\mathcal{X})|}{2 \times |N|} \\ SActCov(\mathcal{X}) &= \frac{|UpperCorner(\mathcal{X})|}{|N|} \end{aligned} \quad (4)$$

The formal definition of our coverage metrics are shown in Eq.4, where BoundCov measures the coverage of neurons that produce outputs that exceed the upper or lower bounds, and SActCov measures the coverage of neurons that create outputs that exceed the lower bound. Higher coverage indicates the test suite is better for triggering the corner-case neurons, thus better test suite diversity.

To answer **RQ3**, for each NLP model under test, we first feed its training dataset to compute each neuron's lower and upper bounds. After that, we select the same number of test cases from S^2LCT and CHECKLIST as the test suite and compute the corresponding coverage metrics.

RQ4. To answer RQ4, we conduct experiments to demonstrate that S^2LCT can help developers understand the bugs in the NLP models. Recall that S^2LCT generates test cases by mutating seed sentences (e.g. by expanding one token in the seed input). Still, it is unclear why mutating one token will cause the model to produce misclassified results. We seek to help developers to understand why such mutation will result in the misclassification. Existing work [3, 6, 20] has demonstrated that the ML model prediction is dominated by a minimal set of input features (i.e. tokens in input sentences). Motivated by such intuition, we seek to identify a minimal set of input tokens that dominate the model prediction.

Formally, given a input sentence $x = [tk_1, tk_2, \cdots, tk_n]$, and the NLP model under test $f(\cdot)$, our goal is to find a masking template $T = [t_1, t_2, \cdots, t_n]$, where t_i is 0 or 1, representing masking the i^{th} token in x or not. The template T can mask some tokens in x with attribute tokens, and the masked input has a high probability of retaining the original prediction x, denoted as

$$P(f(T(x)) = f(x)) \geq P_{thresh} \tag{5}$$

To create such a template T, we first compute the contribution score of each input token using an existing explainable ML technique []. Following that, we begin with the full mask template (i.e., all tokens are masked); such full mask template definitely does not satisfy Eq. 5. We then iteratively shift one position from mask to non-mask based on the order of each token's contribution score, until the template T satisfies Eq. 5 Because we iterate the size of our mask, the generated template T will keep the minimum number of tokens in x. Moreover, since the input x is an incorrect prediction, the generated template T is likely to produce misclassification (i.e., the probability to be misclassified is larger than P_{thresh}).

Implementation Details.

Shiyi: Missing: environment running these experiments.

5 EXPERIMENTAL RESULTS

This section presents experiment results and answer the RQs by studying the results quantatively and qualitatively.

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

Table **??** shows results of our manual study. The first column represents type of test case. The number of test cases used for the study is represented in second column. The label consistency score defined in equation 1 is shown in column 3.

We observe that *S^{2}LCT generates test cases that consistently label their sentiment correctly.* Column 3 shows that the label consistency scores are 0.83 and 0.84 for the seed and expanded sentences, respectively. We can observe that the scores are not same as maximum scores meaning that there are inconsistency on labels between S^{2}LCT and subjects. The cuases of inconsistency in the scores are two following: First, complicated sentence leads subjects to misunderstand its meaning. Second, phrase in a sentence introduces its multiple interpretations to understand its sentiment. For example, the word "easy" could be interpreted as both compliment and back-handed insult. This means that S^{2}LCT generates test oracles consistent with human understanding most of the time. It is observed that there is little difference of the scores between the seed and expanded sentences. This implies that the syntax-based sentence expansion in S^{2}LCT preserves the sentiment as its seed.

5.2 RQ2: Correctness of Linguistic Capability Categorization

The linguistic capability relevancy score defined in 2 is shown in column 4 of Table **??**. The result shows that *S^{2}LCT generates test cases that are correctly categorized to the corresponding linguistic capabilities most of the time.* The linguistic capability relevancy scores for the seed and expanded sentences are both 0.9, achieving high order of agreement with human assessment. The fact that the expanded sentences generated by S^{2}LCT also have same level of linguistic capability relevancy as the seed sentences shows that the syntax-based sentence expansion retains the linguistic capabilities.

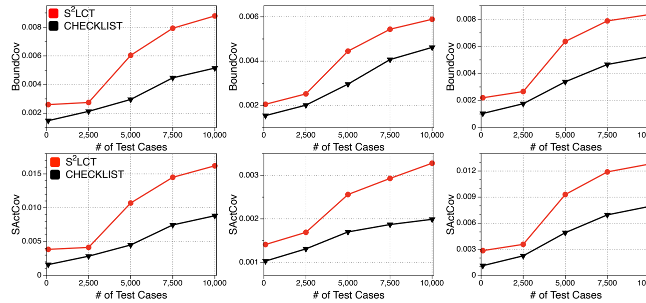

Figure 4: The coverage results of the generated test samples. Shiyi: Make the figures larger.

5.3 RQ3: Test Suite Diversity

We first show evaluation results of three sentiment analysis models via table 3. In the table, first column lists linguistic capability for sentiment analysis task, and Columns 2-3 show the number of seed and expanded test cases respectively. Columns 4-5 show the failure rate in percentage by evaluating the sentiment analysis models on the seed and expanded test cases respectively. Column 6 shows the number of expanded test cases failed, but their seed test cases passed. From the table, we can observe that LC3 has no expanded sentences from its seeds. It means that the 50 seeds selected in this experiment does not provide any validated expansions. It is because the randomness of selection of 50 seeds determines the number of validated expansions. we expect that more selection of seeds increases the probability of availability of their expansion. In addition, LC1 and LC5 obtains 19 and 26 seeds respectively. It represents that S^{2}LCT finds and generates their seeds lower than the target number of seeds, which is 50 in this experiment. In this case, use of larger search dataset increases the amount of seeds for the linguistic capabilities. From the table, it is found that all three models achieves better performance on LC9 and LC2 while they obatains high failure rates on other linguistic capabilities. Finally, we find that there are number of a number of failed expanded test cases succeeded before the expansion (pass on seed, but fail on its expanded test cases). This shows that phase of syntax-based sentence expansion in S^{2}LCT captures failure-inducing expansion given a seed. It finally shows usefulness of S^{2}LCT since the different test results between seed and expanded cases provide accurate guidance for debugging model.

Next, Fig. 4 shows the coverage results of the generated test samples and reliability of the former evaluation results. In the figure, the red line represents S^{2}LCT and the black line represents CHECKLIST. Each column in Fig. 4 represent the results for one NLP model, the first row is the *BoundCov* results and the second row is the *SActCov* results. From the results, we make three observations observations. TODO: First, as the number of test cases increases, the test suite can achieve better coverage. Such observation is intuitive. However, generating a more extensive test suite is not easy, particularly for CHECKLIST, which is a manually template-based approach. Second, for *all* experimental settings (e.g. NLP model, coverage metric), S^{2}LCT achieves high coverage than CHECKLIST. Recall that a higher coverage implies the test case in the test suite is more diverse and rarely to have a statistical distribution similar to the model training data. As a result, a test suite with greater coverage complements the model training data distribution (i.e. hold-out testing data) better. TODO: For example, for the first NLP model under test, S^{2}LCT can achieve a higher coverage than CHECKLIST with only half test cases. Such experimental results confirm that S^{2}LCT can generate more diverse test cases to complement the hold-out testing data for testing NLP models. Shiyi: What does the growth trend in each figure indicate? Shiyi: Does the absolute numbers or relative difference of the two lines on y axis mean anything concrete? E.g., how significant is the improvement in diversity?

Another interesting finding is that for each NLP model, there is no fixed relationship between *BoundCov* and *SActCov*. In other words, while a test suite may produce higher *BoundCov* for some models, the same test suite may get higher *SActCov* for other NLP models. Recall that *BoundCov* measures both the upper and lower corner neurons and *SActCov* measures only the upper corner neurons. Such observation implies that the upper and lower corner neurons are distributed unevenly, and measuring only one of them is not enough.

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

6 THREATS TO VALIDITY

Use of publicly available datasets poses an external validity threat with respect to the generalizability of our results. We implemented domain-spefic knowledge on publically available word sentiments dataset [1]. We also constructed reference CFG from publically available corpus dataset [15, 28]. The dataset used in our study might not be completely representative of all English grammatical structures and English word sentiments. We mitigate this threat by observing that they are widely used in NLP domain [9] and reviewed each script and the output logs.

7 RELATED WORK

NLP Testing.. With the increasing use of NLP models, evaluation of NLP models is becoming more important. Apart from the task accuracy based testing scheme, recent works have also considered model robustness as an aspect for model evaluation. Belinkov *et al.* [2] aims to fail neural machine translation model by intentionally introducing noise in the input text.Pinjia *et al.* [7, 8] measures the robustness by assuming syntatic and semantic relation between input and output of neural machine translation model. Ribeiro *et al.* [22] proposed an approach to generalize semantically equivalent adversarial rules. In addition, Rychalska *et al.* [25] measures drops in BLEU scores by corruption operation, and compare model robustness based on the amount of the drops. In addition, Iyyer *et al.* [10] introduced learning based model for adversarial data augmentation. Not only evaluating the robustness on adversarial set, but various aspets on the NLP model are considered for the robustness evaluation. Prabhakaran *et al.* [17] developed an evaluation framework to detect unintended societal bias in NLP model.*et al.* [24] introduced a functional test suite for hate speech detection in the NLP model. In addition Ribeiro *et al.* [19, 21] measures logical consistency of NLP model. These techniques evaluate the robustness of the NLP model. However, we focused on evaluation of model capability over multiple perspectives.

Linguistic capability Evaluation. Wang *et al.* [29, 30] propose multiple diagnostic datasets to evaluate NLP models. This dataset evaluates NLP model's ability to understand input sentence via natural language inference problems. More recently, CHECKLIST proposes evaluation method of input-output behavior defined as linguistic capabilities. CHECKLIST generates behavior-guided inputs for validating the behaviors. [23]. Unlike prior work on manual data generation method, we used structural information in text to generate data which enables automated data generation.

8 CONCLUSIONS

In this paper, we present S^{2}LCT automatically that finds and generates suitable seed test cases for each linguistic capability. S^{2}LCT fuzz the seeds into a sufficient number of test cases by expanding context-free grammar of the seed inputs. We evaluated sentiment analysis models on the generated test cases and anlyzed its impact on testing the models. Further, we examined failure-inducing cases for seeking the root of the bug. We also studied the quality of S^{2}LCT test cases by measuring their consistencies with their test oracles and linguistic capability. These led to show generalization ability of S^{2}LCT for model evaluation and its reliability in the end. Our work stands to provide automated and comprehensive NLP model testing tool.

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] Simin Chen, Soroush Bateni, Sampath Grandhi, Xiaodi Li, Cong Liu, and Wei Yang. 2020. *DENAS: Automated Rule Generation by Knowledge Extraction from Neural Networks*. Association for Computing Machinery, New York, NY, USA, 813âĂŞ825. https://doi.org/10.1145/3368089.3409733

[4] Mihaela Colhon, Åđtefan VlĂČduÅčescu, 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

[5] 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

[6] Wenbo Guo, Dongliang Mu, Jun Xu, Purui Su, Gang Wang, and Xinyu Xing. 2018. Lemna: Explaining deep learning based security applications. In *proceedings of the 2018 ACM SIGSAC conference on computer and communications security*. 364–379.

[7] Pinjia He, Clara Meister, and Zhendong Su. 2019. Structure-Invariant Testing for Machine Translation. *CoRR* abs/1907.08710 (2019). arXiv:1907.08710 http://arxiv.org/abs/1907.08710

[8] Pinjia He, Clara Meister, and Zhendong Su. 2020. Testing Machine Translation via Referential Transparency. *CoRR* abs/2004.10361 (2020). arXiv:2004.10361 https://arxiv.org/abs/2004.10361

[9] Mujtaba Husnain, Malik Muhammad Saad Missen, Nadeem Akhtar, Mickaël Coustaty, Shahzad Mumtaz, and VB Prasath. 2021. A systematic study on the role of SentiWordNet in opinion mining. *Frontiers of Computer Science* 15, 4 (2021), 1–19.

[10] 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

[11] 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

[12] 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

[13] 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

[14] Lei Ma, Felix Juefei-Xu, Fuyuan Zhang, Jiyuan Sun, Minhui Xue, Bo Li, Chunyang Chen, Ting Su, Li Li, Yang Liu, et al. 2018. Deepgauge: Multi-granularity testing criteria for deep learning systems. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*. 120–131.

[15] 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), 313âĂŞ330.

[16] Kayur Patel, James Fogarty, James A. Landay, and Beverly Harrison. 2008. Investigating Statistical Machine Learning as a Tool for Software Development. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Florence, Italy) *(CHI '08)*. Association for Computing Machinery, New York, NY, USA, 667âĂŞ676. https://doi.org/10.1145/1357054.1357160

[17] 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

[18] 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

[19] 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

[20] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. " Why should i trust you?" Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 1135–1144.

[21] 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

[22] 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

[23] 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)*.

[24] 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

[25] 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.

[26] 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).

[27] Sergio Segura, Gordon Fraser, Ana B. Sanchez, and Antonio Ruiz-CortÃľs. 2016. A Survey on Metamorphic Testing. *IEEE Transactions on Software Engineering* 42, 9 (2016), 805–824. https://doi.org/10.1109/TSE.2016.2532875

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

[29] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems. *CoRR* abs/1905.00537 (2019). arXiv:1905.00537 http://arxiv.org/abs/1905.00537

[30] 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

[31] 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..

<table>
<tr><th>Linguistic capability</th><th>Search rule and transformation template</th></tr>
<tr><td>LC1: Short sentences with neutral adjectives and nouns</td><td>Search seed={length: <10; include: neutral adjs & neutral nouns; exlcude: pos adjs & neg adjs & pos nouns & neg nouns; label: neutral}
Transform N/A</td></tr>
<tr><td>LC2: Short sentences with sentiment-laden adjectives</td><td>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</td></tr>
<tr><td>LC3: Sentiment change over time, present should prevail</td><td>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.’]}</td></tr>
<tr><td>LC4: Negated negative should be positive or neutral</td><td>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’])</td></tr>
<tr><td>LC5: Negated neutral should still be neutral</td><td>Search demonstrative_sent={start: [This, That, These, Those] + [is, are]; label: neutral}
Transform negation of demonstrative_sent</td></tr>
<tr><td>LC6: Negation of negative at the end, should be positive or neutral</td><td>Search neg_sent={label: neg}
Transform seed={[‘I agreed that’, ‘I thought that’]+[neg_sent]+[‘but it wasn’t’, ‘but I didn’t’]}</td></tr>
<tr><td>LC7: Negated positive with neutral content in the middle</td><td>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]}</td></tr>
<tr><td>LC8: Author sentiment is more important than others</td><td>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’]}</td></tr>
<tr><td>LC9: Parsing sentiment in (question, yes) form</td><td>Search pos_sent={label: pos}, neg_sent={label: neg}
Transform seed={[‘Do I think that’, ‘Do I agree that’]+[pos_sent | neg_sent]+[‘? yes’]}</td></tr>
<tr><td>LC10: Parsing positive sentiment in (question, no) form</td><td>Search pos_sent={label: pos}
Transform seed={[‘Do I think that’, ‘Do I agree that’]+[pos_sent]+[‘? no’]}</td></tr>
<tr><td>LC11: Parsing negative sentiment in (question, no) form</td><td>Search neg_sent={label: neg}
Transform seed={[‘Do I think that’, ‘Do I agree that’]+[neg_sent]+[‘? no’]}</td></tr>
</table>

Table 3: Comparison of evaluation of BERT-base, RoBERTa-base and DistilBERT-base sentiment analysis models on seed and expanded testcases (TCs). Due to the spatial constraints of table, BERT-base, RoBERTa-base and DistilBERT-base models are denoted as BERT, RoBERTa and dstBERT respectively.

Linguistic capability	S^2LCT #Seeds	S^2LCT #Exps	S^2LCT Seed Fails[%]	S^2LCT Exp Fails[%]	S^2LCT #PassToFail
LC1: Short sentences with neutral adjectives and nouns	19	210	BERT: 78.95 RoBERTa: 89.47 dstBERT: 100.00	BERT: 86.67 RoBERTa: 76.19 dstBERT: 94.29	BERT: 19 RoBERTa: 9 dstBERT: 0
LC2: Short sentences with sentiment-laden adjectives	50	394	BERT: 4.00 RoBERTa: 4.00 dstBERT: 2.00	BERT: 4.31 RoBERTa: 4.82 dstBERT: 2.79	BERT: 4 RoBERTa: 5 dstBERT: 8
LC3: Sentiment change over time, present should prevail	50	-	BERT: 24.00 RoBERTa: 56.00 dstBERT: 78.00	BERT: - RoBERTa: - dstBERT: -	BERT: - RoBERTa: - dstBERT: -
LC4: Negated negative should be positive or neutral	50	1784	BERT: 92.00 RoBERTa: 82.00 dstBERT: 88.00	BERT: 94.17 RoBERTa: 86.15 dstBERT: 88.51	BERT: 49 RoBERTa: 64 dstBERT: 26
LC5: Negated neutral should still be neutral	26	1009	BERT: 92.31 RoBERTa: 92.31 dstBERT: 92.31	BERT: 92.37 RoBERTa: 93.06 dstBERT: 95.24	BERT: 62 RoBERTa: 38 dstBERT: 74
LC6: Negation of negative at the end, should be positive or neutral	50	1486	BERT: 100.00 RoBERTa: 100.00 dstBERT: 100.00	BERT: 100.00 RoBERTa: 100.00 dstBERT: 100.00	BERT: 0 RoBERTa: 0 dstBERT: 0
LC7: Negated positive with neutral content in the middle	50	1634	BERT: 84.00 RoBERTa: 42.00 dstBERT: 76.00	BERT: 88.00 RoBERTa: 40.64 dstBERT: 78.34	BERT: 40 RoBERTa: 54 dstBERT: 9
LC8: Author sentiment is more important than of others	50	2323	BERT: 44.00 RoBERTa: 26.00 dstBERT: 40.00	BERT: 50.06 RoBERTa: 31.21 dstBERT: 44.77	BERT: 134 RoBERTa: 78 dstBERT: 31
LC9: Parsing sentiment in (question, yes) form	50	1373	BERT: 4.00 RoBERTa: 2.00 dstBERT: 6.00	BERT: 2.69 RoBERTa: 4.15 dstBERT: 5.54	BERT: 43 RoBERTa: 46 dstBERT: 14
LC10: Parsing positive sentiment in (question, no) form	50	1218	BERT: 42.00 RoBERTa: 58.00 dstBERT: 92.00	BERT: 46.14 RoBERTa: 64.53 dstBERT: 87.60	BERT: 40 RoBERTa: 18 dstBERT: 8
LC11: Parsing negative sentiment in (question, no) form	50	1161	BERT: 100.00 RoBERTa: 100.00 dstBERT: 94.00	BERT: 100.00 RoBERTa: 100.00 dstBERT: 98.97	BERT: 0 RoBERTa: 0 dstBERT: 4