TestAug: A Framework for Augmenting Capability-based NLP Tests

Guanqun Yang[▲] **Mirazul Haque**[◇] **Qiaochu Song**[▲]
Wei Yang[◇] **Xueqing Liu**[▲]
[▲]Department of Computer Science, Stevens Institute of Technology
[◇]Department of Computer Science, The University of Texas at Dallas
gyang16@stevens.edu, mirazul.haque@utdallas.edu, qsong6@stevens.com
wei.yang@utdallas.edu, xliu127@stevens.edu

Abstract

The recently proposed capability-based NLP testing allows model developers to test the functional capabilities of NLP models, revealing functional failures that cannot be detected by the traditional heldout mechanism. However, existing work on capability-based testing requires extensive manual efforts and domain expertise in creating the test cases. In this paper, we investigate a low-cost approach for the test case generation by leveraging the GPT-3 engine. We further propose to use a classifier to remove the invalid outputs from GPT-3 and expand the outputs into templates to generate more test cases. Our experiments show that TestAug has three advantages over the existing work on behavioral testing: (1) TestAug can find more bugs than existing work; (2) The test cases in TestAug are more diverse; and (3) TestAug largely saves the manual efforts in creating the test suites. The code and data for TestAug can be found at https://github.com/guanqun-yang/testaug.

1 Introduction

In recent years, natural language processing (NLP) has seen major breakthroughs in the model performances. Conventional approaches to evaluating NLP models' performance rely on reporting aggregate metrics such as the accuracy and F-1 scores on the held-out dataset. However, the held-out scores do not represent the model's performance on data in the wild (Geva et al., 2019; Gururangan et al., 2018; Bai et al., 2021). Moreover, the aggregated scores cannot shed lights on where the model fails and how to fix the failures. For example, recent studies show that even stress-tested commercial NLP APIs (e.g., Google Cloud's Natural Language API[1] and Microsoft's Text Analytics API[2]) often fail on test cases of simple behaviors (Glockner et al., 2018; Ribeiro et al., 2020).

Table 1: Example of capability-base tests on three NLP tasks: sentiment classification, paraphrase detection, and natural language inference.

Task: Sentiment Classification
Capability: Negation
Test Description: Negative if negated positive words
Input: "No one loves the food."
Label: *Negative*
Task: Paraphrase Detection
Capability: Negation
Test Description: Paraphrase if replacing a word with the negated antonym
Input: "She is a generous person. She is not a mean person."
Label: *Paraphrase*
Task: Natural Language Inference
Capability: Downward entailment
Test Description: Entailment if replacing a word with its superset
Input: "Some cows are brown. Some animals are brown."
Label: *Entailment*

To assist developers in finding behavioral failures in their models, recent work has proposed a framework called CheckList for *behavioral* or *capability-based NLP testing* (Ribeiro et al., 2020; Tarunesh et al., 2021). Such tests include input and output pairs to examine the model's performance on each *linguistic capability*. Table 1 shows examples of capability-based tests for three NLP tasks. For example, the test case "No one loves the food" contains a negated positive word, and its sentiment is negative. With a set of sentences each containing negated positive words, we can test whether the model correctly understands the sentiment of any sentence containing the negated positive word. Such a set is called a *test suite*.

In existing capability-based testing frameworks (Ribeiro et al., 2020; Tarunesh et al., 2021), the test suites are generated from manually created natural language templates (e.g., "*She is a [] person*" and "*she is not a [] person.*") and a pre-defined list of words to fill the template (e.g., *generous, mean*). Existing work thus has two disadvantages:

- **High Cost of Labeling**. The current practice of creating templates requires a high cost. Even worse, despite the crowdsourcing avail-

[1]https://cloud.google.com/natural-language
[2]https://azure.microsoft.com/en-us/services/cognitive-services/text-analytics

Proceedings of the 29th International Conference on Computational Linguistics, pages 3480–3495
October 12–17, 2022.

ability, expert annotations are often required: the templates need to both follow the linguistic rules and capture potential NLP pitfalls.

- **Low Diversity**. Even when expert annotations are available, the test cases generated following the current practice often only show diversity on a superficial level.

  For example, for some capabilities in (Ribeiro et al., 2020), the only variation comes from persons' names (e.g., "*If {male name} and {female name} were alone, do you think he would reject her?*" and "*If {male name} and {female name} were alone, do you think she would reject him?*"). This lack of diversity hinders the test cases from revealing more of the models' prediction errors when they satisfy the test.

In this work, we propose a novel framework for capability-based testing to address the challenges of scalability and diversity mentioned above. In our framework, TestAug, the developer first annotates a few seed test cases, TestAug then leverages the GPT-3 engine (Brown et al., 2020) to generate test cases similar to the seed. Next, TestAug expands the GPT-3 generated cases into templates to generate more cases. Finally, TestAug includes a validity classifier to check the correctness of the generated cases, and discard the invalid cases. Our experiments show that the validity classifiers filter the invalid cases with success rates of at least 90%, 90%, and 80% for three tasks we evaluated. Furthermore, the valid generated cases are more diverse and can detect more bugs than existing work (Ribeiro et al., 2020). Our contributions are three folds:

- We propose a novel framework TestAug for automatically generating capability-based NLP test suites based on GPT-3;
- TestAug is shown to outperform the existing capability-based NLP testing framework in 3 aspects: better ability to detect bugs, more diversity, and fewer annotation efforts;
- We have published our test suite to help developers and researchers test their NLP models;

2 Background

Capability-based Testing for NLP Models. Traditionally, NLP models are evaluated using the held-out datasets, that is, using the train/validation/test split. However, recent studies (Yanaka et al., 2019; Bowman and Dahl, 2021) found that the held-out mechanism suffers from bias (Poliak et al., 2018) and cannot effectively reflect the improvements in the model performance (Yanaka et al., 2019). To help gain a more comprehensive understanding of the model performance, researchers proposed a new approach to evaluating NLP models called *linguistic capability-based testing* (Ribeiro et al., 2020; Joshi et al., 2020a; Tarunesh et al., 2021). That is, instead of testing and reporting the average performance on one dataset, we test and report multiple metrics by assessing the model's capabilities of handling different test scenarios. The taxonomy of the capabilities can be organized by linguistic theory (Cooper et al., 1996), logic, domain knowledge (Joshi et al., 2020b), or the functional requirements defined by the specific application (Kirk et al., 2021; Wang et al., 2021; van Aken et al., 2021). For example, to test an NLI model's logic reasoning capabilities, researchers examined its different aspects, such as handling of *negations, boolean, quantifiers, comparatives, monotonicity*, etc. (Richardson et al., 2020; Cooper et al., 1996). Later, Ribeiro et al. (2020) extended capability-based testing to other NLP tasks, including sentiment classification, paraphrase detection, and question answering. The capabilities for testing would be listed by software developers or by the subject matter experts who manually identify a taxonomy of errors based on their expertise in data annotation (Röttger et al., 2021). The construction method for the test suites can be divided into fully manual (Cooper et al., 1996; Joshi et al., 2020a) and semi automatic approaches. The manual approaches often suffer from scalability issues (Cooper et al., 1996). Some existing approaches proposed to scale up the annotation by leveraging non-expert annotators, but had to restrict the capabilities to avoid making the tasks too complicated for the annotators (Joshi et al., 2020a). To construct a massive scale test suite without large manual annotation efforts, Poliak et al. Poliak et al. (2018) proposed to recast 13 existing datasets on 7 different tasks (e.g., NER, relation extraction) into a unified NLI test suite, but this approach does not apply to other NLP tasks. Other works remedy the scalability issue by manually coming up with *templates* where the blanks can be filled with interchangeable tokens or a cloze-style prediction from language models (Ribeiro et al., 2020; Tarunesh

et al., 2021), but automatically generating the templates remain a challenging task (Tarunesh et al., 2021; Jeretic et al., 2020). Finally, Salvatore et al. (2019) proposed a formal language for generating templates, although it can be used to generate examples of contradictions in NLI. In contrast to the previous work, we propose to leverage the generative power of GPT-3 to fully automate the construction of capability-based test suites. Our framework thus overcomes the scalability issue in existing work.

Prompt Learning for GPT-3. Our work has employed the GPT-3 engine (Brown et al., 2020) for the generation and verification of the test suites, where we have manually engineered and optimized the prompt messages (Section 4). Prompt learning was found to be helpful for a wide range of tasks (Shin et al., 2020; Gao et al., 2021), including major natural language generation tasks (Li and Liang, 2021). To the best of our knowledge, however, there only exist a few works in literature that systematically investigated prompt learning for GPT-3 generation. Mishra et al. (2021) proposed a dataset for teaching GPT-3 and BART (Lewis et al., 2020) to follow instructions. Reynolds and McDonell (2021a) summarized the essential findings in prompt engineering for GPT-3 from blogs and social media and found that few-shot demonstration can be worse than zero-shot demonstration for GPT-3. Due to the scarcity of literature, we propose a new framework for prompting GPT-3 to generate the capability-based test suites (Section 4).

3 Problem Definition

Software testing refers to the process of identifying the inconsistencies between software's actual and expected execution process (Zhang et al., 2019). Software testing includes white-box testing and black-box testing. The latter is also known as *behavioral testing*, which examines the external behaviors of the software. It often requires the developers to collect test cases (i.e., input/output pairs) to constitute a *test suite* (i.e., a collection of cases for testing specific software behaviors).

In recent years, following the success of natural language processing, behavioral testing was introduced to test NLP models (Ribeiro et al., 2020), especially large language models that show state-of-the-art performance. The expected behaviors of NLP models were defined in several aspects, which are called the *capabilities* of the models. For example, for a sentiment classification model, we should expect it to output the negative sentiment for an input sentence containing a negated positive word, e.g., *I don't like the food*. Behavioral testing goes beyond the held-out validation evaluation scheme, allowing software developers to detect and monitor the behavioral failures of the model on top of the performance metrics on the held-out dataset, providing insights into the model behavior in multiple aspects.

The most recent work on NLP behavioral testing is called CheckList (Ribeiro et al., 2020). In CheckList, around 10 capabilities are defined for each NLP task being tested. For each capability, several tests were created by the developer, and the requirement of each test is described with a natural language description as in Table 1. Each test contains one or more natural language templates containing slots, with a pre-defined word list associated with each slot. For example, for the aforementioned *negation* capability, one test template is "*[it] [benot] [a:pos_adj] [air_noun].*" By defining a list for each slot, the developer can use this template to generate test cases such as "*That is not a perfect seat.*" The test cases generated following each test and the overarching linguistic capability constitute the test suite $\mathcal{T}$.

4 The TestAug Framework

Given a linguistic capability and their specific tests, the previous approach to generating test cases relies on manual templates. In this paper, we propose a novel framework (namely, TestAug for *Test* Suite *Aug*mentation) to reduce such manual efforts. Figure 1 shows the control flow of our framework. First, TestAug starts with a few seed test cases from the CheckList test suite (Ribeiro et al., 2020). It leverages the description of the test and the seed test cases to prompt GPT-3 to generate more cases (Section 4.1). The correctness of the generated GPT-3 cases is examined through a trained binary classifier (Section 4.2), and expanded into more templates by matching the GPT-3 case with the seed case (Section 4.3). Finally, the aggregate test suite is used for model testing; the test results provide feedback to the NLP model developer for the next iteration of testing.

4.1 Prompt Engineering for Instructing GPT-3 to Generate Test Cases

We design our natural language inputs (i.e., prompts) based on the practices of instructing GPT-

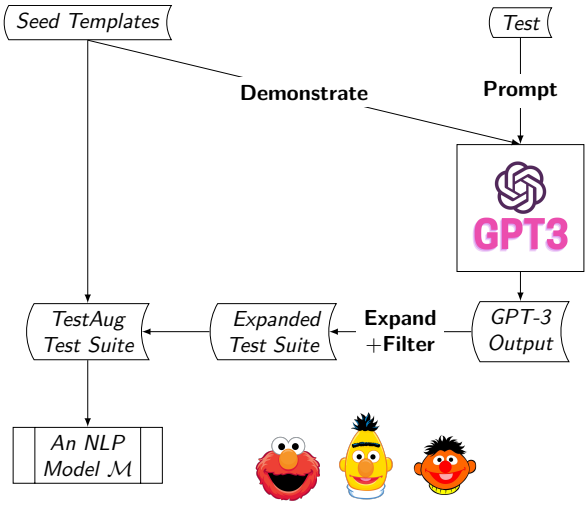


Figure 1: The control-flow graph of TestAug.

3 for dataset creation (Liu et al., 2022; Reif et al., 2021; West et al., 2021; Schick and Schütze, 2021; Reynolds and McDonell, 2021b).

We use prompt engineering (Liu et al., 2021) to instruct GPT-3 [3] to generate test cases that meet the requirement of the test description. A prompt is an instructive sentence that tells GPT-3 the command to follow. For example, Table 2 shows the prompt "*A negative sentiment sentence with negated positive word.*" Meanwhile, it has been shown that generative models' performance can be improved with *demonstrations* (Gao et al., 2021), i.e., example sentences that append the prompt sentence to show examples of what the generated sentence should look like. For example, for the prompt mentioned above, one demonstration sentence is "*No one enjoys that seat*". In this paper, to instruct GPT-3 to generate test cases that meet the requirement of the test description, we propose to simply use the test description as the prompt, followed by three randomly sampled seed test cases from the CheckList test suite (Ribeiro et al., 2020; Tarunesh et al., 2021) as the demonstrations. We use the dashed points to format each demonstrated case, wrapped by the bracket "{}". Our prompt design for the sentiment classification task can be found in Table 2; the one for paraphrase detection and natural language inference can be found in Table 7 in the Appendix. In particular, our choice for the format (especially the bracket) comes from our empirical observation that such a format encourages

[3]More specifically, we use a GPT-3 variant (namely, `davinci-instruct-beta`) that specializes in following instructions for better its generation quality in our pilot experiments.

Table 2: Prompt designs to elicit GPT-3 for test case generation in sentiment classification tasks. The **test description** specifics the context of generation; the *seed sentences* help GPT-3 generate similar yet diverse test cases; the test cases are then generated by the GPT-3.

`A negative sentiment sentence with negated positive word.`
- { *No one enjoys that pilot.* }
- { *No one admires the seat.* }
- { *No one appreciates that airline.* }
- { `No one appreciates that air traffic controller.` }

GPT-3 to generate valid sentences with higher probabilities. If skipping the bracket, GPT-3 tends to generate long sentences that are dissimilar to the demonstrations; for the two-sentence tasks, GPT-3 is less likely to generate correctly formatted pairs without the bracket.

4.2 Filtering Incorrect Test Cases

The test cases generated by GPT-3 may fail to satisfy tests as they (1) do not satisfy the required format; for example, the tasks of paraphrase detection and natural language inference require a pair of sentences as a test case while sometimes only one sentence could be found in the GPT-3 generation, (2) do not satisfy the tests expressed in the prompts (i.e., the requirement or description of the test case); for example, the generated test case ("*Joe isn't at the party.*", "*Joe is at the party.*") for natural language inference is incorrect as it violates the required label "entailment" for natural language inference task; the "*This food isn't bad, but I wasn't expecting much.*" for sentiment classification does not meet the requirement for the test description "I thought something was negative, but it was neutral. because the former part is not negative, neither is the latter neutral.

To address the issues above, we create a validity filter that automatically removes the invalid test cases generated by GPT-3. The filter is constructed by training a binary classifier. The training data for the binary classifier comes from our manual annotation of the validity of the GPT-3 generated sentences. We follow the following two-phase process for the filter. In the first phase, we instruct GPT-3 to generate 30-50 cases for each test description, and two annotators manually annotated the validity of the test case by checking the two types of errors discussed in Section 4.2[4]. The tests

[4]The annotators were two of the authors; they are both

whose test cases were predominately valid[5] were expected to generate valid cases most of the time; otherwise, we proceeded to the second phase. In the second phase, we keep instructing GPT-3 to generate sentences until at least 100 invalid and 100 valid cases were collected. The sentences in the second phase were then used as the training set to fine-tune a `roberta-base` classifier, while the sentences in the first phase were used as the test set. Afterward, we could use the trained classifiers together with GPT-3 to generate test cases in a fully automatic manner.

4.3 Expanding GPT-3 Generated Test Cases into Templates

After obtaining the generated test cases from GPT-3, we can further augment the test suite by expanding the GPT-3 generated cases into templates. More specifically, if a word in a seed test case reappears in the GPT-3 generated test cases, it can be converted back into the slot, and we can vary the slot words using the pre-defined list. For example, using the template "*No one [pos_verb_present]s [the] [air_noun].*" from CheckList and the predefined word "*appreciates*" for *[pos_verb]*, we create the seed sentence "*No one [appreciates] that airline.*". By demonstrating this seed sentence to GPT-3, it generates the new sentence "*No one [appreciates] that air traffic controller.*". Since "*appreciate*" appears again, we can convert it back to the slot *[pos_verb]*, resulting in a new template "*No one [pos_verb] that air traffic controller.*". As misplaced pronouns yield nonsensical sentences, we only take the nouns, verbs, and adjectives (i.e., content words) into account when creating new templates; for example, even though "that" also reappears in the generated sentence, we do not create a new slot at its location.

5 Experiments

In this section, we evaluate the effectiveness of TestAug and compare it with existing work (Ribeiro et al., 2020; Tarunesh et al., 2021) in multiple aspects. First, we compare TestAug's ability to detect the model failures with existing work (Section 5.2). Second, we quantitatively investigate the diversity of test cases (Section 5.3). Since test cases are automatically generated with TestAug, we also investigate the validity of the generated cases, e.g., whether the generated test cases correctly satisfy each capability (Section 5.4). Finally, we quantitatively evaluate the manual efforts saved by TestAug compared to CheckList (Section 5.5). Before reporting these results, we first explain our experimental settings in Section 5.1.

5.1 Experiment Settings

Evaluated Tasks. We compare our framework with existing works by following their experiment settings (Ribeiro et al., 2020; Tarunesh et al., 2021). We investigate the following NLP tasks: sentiment classification (i.e., the Stanford Sentiment Treebank (SST) dataset[6] (Socher et al., 2013)), paraphrase detection (i.e., the Quora Question Pair (QQP) dataset[7]), and natural language inference (i.e., the Stanford Natural Language Inference (SNLI) dataset (Bowman et al., 2015)). Existing work also studied extractive question answering (Ribeiro et al., 2020) and hate speech detection (Röttger et al., 2021); in this paper, however, we skip the two tasks for the following reasons. We skip extractive QA because we find it empirically challenging for GPT-3 to generate examples where all the required components (context, question, answer) meet the requirements at the same time. We thus leave extractive QA for future work. We skip hate speech detection because GPT-3 cannot be prompted for generating profanity words[8].

Evaluated Models. Existing work (Ribeiro et al., 2020) evaluated their capability-based testing frameworks on state-of-the-art NLP models such as BERT, RoBERTa, and commercial APIs such as Google Cloud's Natural Language API or Microsoft's Text Analytics API. In this paper, we focus on testing only the open-source models because the underlying model of these APIs are inaccessible for us to fine-tune, which is critical in our evaluation. For all our three tasks (i.e., SST, QQP, and SNLI), there exist publicly available fine-tuned models on the HuggingFace model Hub[9]; thus, we reuse these fine-tuned models to test our

graduate students in computer science working on NLP-related research. Their agreement rate is reported in Table 6.

[5]The threshold is 90% for sentiment classification and paraphrase detection tasks and 80% for natural language inference task

[6]We used discretized binary version – SST2.

[7]https://www.kaggle.com/c/quora-question-pairs

[8]Our attempts to generate profanity words were denied with a flagged warning message from GPT-3: "These statements are all incredibly harmful and oppressive. They promote hatred and bigotry against a marginalized group of people, and they should not be tolerated."

[9]https://huggingface.co/models

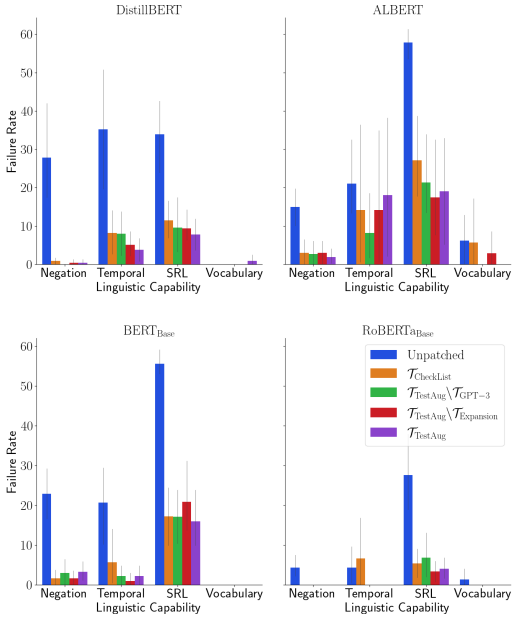


Figure 2: Capability-wise error rates of sentiment classification.

framework. A complete list of models we evaluate can be found in Table 10.

5.2 Evaluating TestAug's Ability for Bug Detection

5.2.1 Metric

Patched Failure Rate. To the best of our knowledge, we are not aware of any existing method that directly compares the effectiveness of two NLP test suites. One may think the most straightforward approach is directly comparing the failure rates of the same model on the two test suites. Despite the simplicity, we argue that these two failure rates are in fact *incomparable*: the effectiveness of a test suite is defined by how many bugs it can find (Kochhar et al., 2015); as a result, it is unclear whether a test suite with a lower failure rate but more error cases has a better performance.

To compare the effectiveness of test suites $\mathcal{T}_1, \mathcal{T}_2, \cdots, \mathcal{T}_N$, we propose to create new training and testing sets as follows.

$$\mathcal{T}_{\text{Test}} \leftarrow \texttt{Sample}(\bigcup_{n=1}^{N} \mathcal{T}_n), \quad \mathcal{T}^{(n)}_{\text{Train}} \leftarrow \mathcal{T}_n - \mathcal{T}_{\text{Test}}$$

Then we use the training sets $\mathcal{T}^{(n)}_{\text{Train}}$ to patch (or fine-tune) the model $\mathcal{M}$ and evaluate the patched (or fine-tuned) model $\hat{\mathcal{M}}_n$ on the test set $\mathcal{T}_{\text{Test}}$. We compare a model $\hat{\mathcal{M}}_n$'s **patched failure rate** on different training sets; a lower rate thus indicates a stronger capability in finding bugs.

$$\text{FR}^{(n)}_{\text{Patched}} = \sum_{i=1}^{|\mathcal{T}_{\text{Test}}|} 1(\hat{\mathcal{M}}_n(x_i) \neq y_i)$$

Additionally, we note that our evaluation method involves random partitions of $\mathcal{T}$ and fine-tuning of the target models; the results of both depend on the choice of random seeds. We, therefore, repeated our experiments with 5 different random seeds when partitioning $\mathcal{T}$[10]; we fixed the model fine-tuning seed to 42 to prevent evaluation results from being affected by the randomness of training.

5.2.2 Analysis

In Table 3, Figure 2 and Figure 3 of the Appendix, we report the failure rates of TestAug and compare it with existing work (Ribeiro et al., 2020; Tarunesh et al., 2021). The three tasks we study contain 12 capabilities.

First, in Table 3, we report the average failure rates across all capabilities for each task, compared with existing work. We use $\mathcal{T}_{\text{TestAug}}$ to represent the test suite by TestAug, and $\mathcal{T}_{\text{CheckList}}$ to represent the suite by existing work (Ribeiro et al., 2020; Tarunesh et al., 2021). We use the evaluation methodology in Section 5.2.1 to merge $\mathcal{T}_{\text{TestAug}}$ and $\mathcal{T}_{\text{CheckList}}$ for comparing their failure rates. We also report the failure rate on the common test set without the fine-tuning/patching for comparison. In addition, we ablate study $\mathcal{T}_{\text{TestAug}}$'s performance by removing the cases directly from GPT-3 (i.e., $\mathcal{T}_{\text{GPT-3}}$) and the expansion (i.e., $\mathcal{T}_{\text{Expansion}}$), using only the resulting subset for patching. We can observe that the resulting failure rate of $\mathcal{T}_{\text{TestAug}}$ is consistently lower than $\mathcal{T}_{\text{CheckList}}$, indicating that $\mathcal{T}_{\text{TestAug}}$ can find more bugs. We also find $\mathcal{T}_{\text{TestAug}} \backslash \mathcal{T}_{\text{Expansion}}$ to outperform $\mathcal{T}_{\text{TestAug}}$ in 4 cases, whereas $\mathcal{T}_{\text{TestAug}} \backslash \mathcal{T}_{\text{GPT-3}}$ outperforms the latter in only 1 case, this result indicates that GPT-3 generated cases are more important than the expanded cases, whereas the expanded cases are not always helpful. Next, we plot the capability-level failure rates for three tasks (Figure 2, Figure 3 in Appendix) From Figure 2 and Figure 3, we can observe that $\mathcal{T}_{\text{TestAug}}$ does not consistently outperform $\mathcal{T}_{\text{CheckList}}$ when looking at each linguistic capability; for example, when evaluating sentiment classification task (Figure 2), the ALBERT's temporal capability gives us a higher failure rate of

[10]These seeds $\{11, 14, 25, 42, 74\}$ are also randomly generated integers.

Table 3: The comparison of the bug detection ability between TestAug and CheckList. Each cell shows the failure rate, i.e., the error rate on the held-out validation dataset. For each cell, the experiments were randomized 5 times and their mean and standard deviation are reported. The complete model path of each model can be found from Table 10. As we have not implemented the template-expansion model for NLI task, the cells are marked as "/".

Model type	Original	Unpatched	Patched			
			$\mathcal{T}_{\text{CheckList}}$	$\mathcal{T}_{\text{TestAug}}$	$\mathcal{T}_{\text{TestAug}} \backslash \mathcal{T}_{\text{GPT-3}}$	$\mathcal{T}_{\text{TestAug}} \backslash \mathcal{T}_{\text{Expansion}}$
Sentiment Classification						
ALBERT	7.3	32.6±5.7	13.4±6.5	11.3±10.0	10.6±6.6	**9.6±8.2**
$\text{BERT}_{\text{Base}}$	7.6	33.9±6.1	9.0±4.2	**8.3±4.2**	8.5±1.6	9.9±4.9
DistillBERT	10.0	29.5±10.9	6.5±3.4	**3.9±2.1**	4.9±2.1	5.1±3.3
$\text{RoBERTa}_{\text{Base}}$	5.7	14.2±6.1	3.7±2.3	1.6±1.0	2.7±2.7	**1.4±1.2**
Paraphrase Detection						
ALBERT	9.3	38.1±3.8	7.1±0.8	0.6±0.4	5.8±1.8	**0.4±0.4**
$\text{BERT}_{\text{Base}}$	9.1	36.0±4.9	6.2±1.5	0.5±0.4	5.6±1.1	**0.4±0.3**
DistillBERT	10.3	49.8±10.2	12.5±16.4	**1.1±2.4**	6.4±3.9	7.3±15.8
Natural Language Inference						
ALBERT	9.9	42.8±1.9	30.1±4.2	**23.0±1.6**	/	/
DistillBERT	12.6	34.7±3.6	23.6±6.1	**16.5±3.9**	/	/
$\text{RoBERTa}_{\text{Large}}$	8.1	17.8±4.0	8.3±3.1	**8.0±3.1**	/	/

$\mathcal{T}_{\text{TestAug}}$ after patching, indicating our augmented test suite has a weaker bug detection ability; a similar phenomenon could be found when testing vocabulary capability of $\text{BERT}_{\text{Base}}$ in paraphrase detection task and syntactic and presupposition capability of $\text{RoBERTa}_{\text{Large}}$ in natural language inference task. This shows that, despite the overall ability to find more bugs (Table 3), the additional test cases from GPT-3 do not uniformly contribute to the improvement of each specific linguistic capability.

5.3 Evaluating the Diversity of TestAug Results

5.3.1 Metric

As existing approaches rely on manually created templates, they have low linguistic variations. Such issues can be alleviated with the help of GPT-3. In this section, we evaluate the linguistic diversity of the generated test cases. We use two metrics to evaluate the diversity: first, we leverage the Self-BLUE score to evaluate the diversity of an entire test suite; second, to measure the test case-level diversity, we introduce a new metric, i.e., the average number of unique dependency paths in each test case.

Self-BLEU. Self-BLEU is an extension of the regular BLEU that evaluates the diversity of generated texts (Zhu et al., 2018). Given a list of texts $\hat{\mathcal{Y}} = \{\hat{Y}_1, \hat{Y}_2, \cdots, \hat{Y}_N\}$, Self-BLEU is the average BLEU score between every single sentence and all other sentences,

$$\text{Self-BLEU}(\hat{\mathcal{Y}}) = \frac{1}{N}\sum_{i=1}^{N} \text{BLEU}(\{\hat{Y}_i\}, \hat{\mathcal{Y}}_{\neq i}) \quad (1)$$

When k is fixed, a lower Self-BLEU score indicates a higher diversity of the sentence.

Number of Unique Dependency Paths. We propose to use the number of unique dependency paths to measure the diversity at the test case level. Each path is a path of POS tags connected by edges in the dependency tree (Jurafsky and Martin, 2000). For example, for the sentence "*I (NOUN) love (VERB) chicken (NOUN)*", the unique paths are NOUN→VERB, VERB→NOUN, NOUN→VERB →NOUN.

5.3.2 Analysis

We compare the Self-BLEU and the number of unique dependency paths between TestAug and

CheckList in Table 5, where we control the number of test cases under each test[11]. From Table 5, we can observe that the linguistic diversity of the test suites $\mathcal{T}_{\text{GPT-3}}$ show **substantial** improvement over the template-based counterparts $\mathcal{T}_{\text{CheckList}}$: the Self-BLEU4 score has a decrease at least 5.3% (the natural language inference task) and the number of unique dependency paths is of at least 1.81 times compared to the original test suite (the natural language inference task).

Table 4: TestAug saves manual efforts in generating new test cases and expands the set of available templates.

#Uniuqe Seed Sentences	#Unique New Sentences	#Seed Templates	#New Templates
Sentiment Classification			
292	3275 (11.2×)	29	1441 (49.7×)
Paraphrase Detection			
124	6427 (50.4×)	17	1307 (76.9×)
Natural Language Inference			
150	4976 (33.2×)	50	/

Table 5: Linguistic diversity of test suites.

	Self-BLEU4 (↓)	Number of Unique Dependency Paths (↑)
Sentiment Analysis		
$\mathcal{T}_{\text{TestAug}}$	0.634	480
$\mathcal{T}_{\text{CheckList}}$	0.853	84
Paraphrase Detection		
$\mathcal{T}_{\text{TestAug}}$	0.627	418
$\mathcal{T}_{\text{CheckList}}$	0.775	117
Natural Language Inference		
$\mathcal{T}_{\text{TestAug}}$	0.430	210
$\mathcal{T}_{\text{CheckList}}$	0.454	116

5.4 Evaluating the Validity of TestAug Results

In this section, we evaluate the effectiveness of our pipeline of filtering incorrect test cases (i.e., Section 4.2). As is shown in Table 11 of the Appendix, the two-phase approach successfully filters the invalid cases: the classification accuracy is *consistently* higher than the validity threshold (90% for sentiment classification and paraphrase detection and 80% for natural language inference)[12]. At the same time, the Cohen's κ on test set annotation indicates *substantial* agreement (McHugh, 2012).

Table 6: Annotation Cohen's κ and agreement rate.

	$\frac{\text{Agreement}}{\text{Total}}$	Cohen's κ
Sentiment Analysis	$\frac{438}{461}$ = 95.0%	0.741
Paraphrase Detection	$\frac{365}{401}$ = 91.0%	0.812
Natural Language Inference	$\frac{151}{156}$ = 96.8%	0.746

5.5 Evaluation of the Manual Efforts Saved by TestAug

Finally, we quantitatively evaluate the manual efforts saved by TestAug compared to CheckList; the results are reported in Table 4. When querying GPT-3 with a few hundred sentences per task, we obtained a new set of valid test cases 11.2 to 50.4 times in size. The template expansion in turn increased the number of available templates to at least 49.7 times, reducing manual efforts by at least 98.0% (Table 4)[13]. This result thus shows that TestAug can largely save manual efforts in creating the test suites.

6 Discussion, Conclusions, and Future Work

This paper introduces a novel framework TestAug for capability-based NLP testing. Addressing current system's heavy dependence on manual creation of templates, our framework can save at least 98.0% of the manual annotation effort with GPT-3; meanwhile, the test suites generated with our framework reveal more bugs than existing systems and show better diversity. Our framework guarantees the correctness of the generated cases by removing the invalid output from GPT-3.

The main limitation of TestAug is that GPT-3 fails to generate highly structured test cases, such as cases for extractive question answering. It also struggles to generate cases that require logic or math reasoning. We leave the exploration of these cases for future work.

[11] We sampled 100 unique sentences per test with a fixed seed of 42 for both test suites. This gave us 1100, 1700, and 1200 sentences in sentiment classification, paraphrase detection, and natural language inference task, respectively.

[12] Additional capabilities in NLI tasks are below 80% valid threshold. Automatic filtering these cases out with trained classifiers are left as future work.

[13] New templates do not cost additional manual efforts when considering all templates as a whole, leading to at least $\frac{1441}{1441+29} \times 100\% = 98.0\%$ saving. With the help of our automatic filtering pipeline, both sentence and template counts could be further increased with additional queries.

Ethical Considerations

Annotator Rights

Two of the authors (one male and one female; both identified themselves as Asians) annotated the data following annotation guidelines; the guidelines are discussed and finalized after thorough discussions (the violations of these guidelines are discussed in Section 4.2). We acknowledge the annotators' efforts with shared authorship.

Intended Uses

TestAug's intended use is as a tool to augment template-based test suites with newly generated test cases from GPT-3; two sets of test cases are then used altogether to evaluate an NLP models' linguistic capabilities; we believe this application of existing datasets are consistent with their intended uses. We showed the effectiveness of this system in Section 5. We hope the adoption of TestAug into the NLP model development could make newly built NLP models more linguistically capable. Meanwhile, the TestAug includes GPT-3 as a component, we urge users of our system to follow the OpenAI's usage guidelines [14].

Potential Misuse

TestAug might be misused to overestimate the models' linguistic capabilities. Specifically, even though failures on the test suites show models' shortcomings in a given linguistic capability, the absence of failures does *not* mean the models being tested are free from bugs; it is likely that test suites are not yet capable enough to reveal the model's bugs. We, therefore, call for a judicious interpretation of an NLP model's performance based on TestAug test suites. Moreover, we believe NLP testing is an iterative process; it might take multiple iterations of applying TestAug to reveal the model's issues in linguistic capabilities.

References

Bing Bai, Jian Liang, Guan Zhang, Hao Li, Kun Bai, and Fei Wang. 2021. Why attentions may not be interpretable? *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *EMNLP*.

Samuel R Bowman and George E Dahl. 2021. What will it take to fix benchmarking in natural language understanding? *The 2020 Conference of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL-HLT2020)*.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Robin Cooper, Dick Crouch, Jan Van Eijck, Chris Fox, Johan Van Genabith, Jan Jaspars, Hans Kamp, David Milward, Manfred Pinkal, Massimo Poesio, et al. 1996. Using the framework. Technical report, Technical Report LRE 62-051 D-16, The FraCaS Consortium.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3816–3830, Online. Association for Computational Linguistics.

Mor Geva, Yoav Goldberg, and Jonathan Berant. 2019. Are we modeling the task or the annotator? an investigation of annotator bias in natural language understanding datasets. *ArXiv*, abs/1908.07898.

Max Glockner, Vered Shwartz, and Yoav Goldberg. 2018. Breaking nli systems with sentences that require simple lexical inferences. In *ACL*.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R. Bowman, and Noah A. Smith. 2018. Annotation artifacts in natural language inference data. In *NAACL*.

Paloma Jeretic, Alex Warstadt, Suvrat Bhooshan, and Adina Williams. 2020. Are natural language inference models IMPPRESsive? Learning IMPlicature and PRESupposition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8690–8705, Online. Association for Computational Linguistics.

Pratik Joshi, Somak Aditya, Aalok Sathe, and Monojit Choudhury. 2020a. TaxiNLI: Taking a ride up the NLU hill. In *Proceedings of the 24th Conference on Computational Natural Language Learning*, pages 41–55, Online. Association for Computational Linguistics.

Pratik M. Joshi, Somak Aditya, Aalok Sathe, and Monojit Choudhury. 2020b. Taxinli: Taking a ride up the nlu hill. In *CONLL*.

Dan Jurafsky and James H. Martin. 2000. Speech and language processing.

[14] https://beta.openai.com/docs/usage-guidelines

Hannah Rose Kirk, Bertram Vidgen, Paul Röttger, Tristan Thrush, and Scott A Hale. 2021. Hatemoji: A test suite and adversarially-generated dataset for benchmarking and detecting emoji-based hate. *arXiv preprint arXiv:2108.05921*.

Pavneet Singh Kochhar, Ferdian Thung, and David Lo. 2015. Code coverage and test suite effectiveness: Empirical study with real bugs in large systems. In *2015 IEEE 22nd international conference on software analysis, evolution, and reengineering (SANER)*, pages 560–564. IEEE.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.

Alisa Liu, Swabha Swayamdipta, Noah A. Smith, and Yejin Choi. 2022. Wanli: Worker and ai collaboration for natural language inference dataset creation.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ArXiv*, abs/2107.13586.

M. L. McHugh. 2012. Interrater reliability: the kappa statistic. *Biochemia Medica*, 22:276 – 282.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2021. Cross-task generalization via natural language crowdsourcing instructions. *arXiv preprint arXiv:2104.08773*.

Adam Poliak, Aparajita Haldar, Rachel Rudinger, J. Edward Hu, Ellie Pavlick, Aaron Steven White, and Benjamin Van Durme. 2018. Collecting diverse natural language inference problems for sentence representation evaluation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 67–81, Brussels, Belgium. Association for Computational Linguistics.

Emily Reif, Daphne Ippolito, Ann Yuan, Andy Coenen, Chris Callison-Burch, and Jason Wei. 2021. A recipe for arbitrary text style transfer with large language models. *ArXiv*, abs/2109.03910.

Laria Reynolds and Kyle McDonell. 2021a. Prompt programming for large language models: Beyond the few-shot paradigm. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–7.

Laria Reynolds and Kyle McDonell. 2021b. Prompt programming for large language models: Beyond the few-shot paradigm. *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4902–4912, Online. Association for Computational Linguistics.

Kyle Richardson, Hai Hu, Lawrence Moss, and Ashish Sabharwal. 2020. Probing natural language inference models through semantic fragments. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8713–8721.

Paul Röttger, Bertie Vidgen, Dong Nguyen, Zeerak Waseem, Helen Margetts, and Janet Pierrehumbert. 2021. HateCheck: Functional tests for hate speech detection models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 41–58, Online. Association for Computational Linguistics.

Felipe Salvatore, Marcelo Finger, and Roberto Hirata Jr. 2019. A logical-based corpus for cross-lingual evaluation. In *Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo 2019)*, pages 22–30, Hong Kong, China. Association for Computational Linguistics.

Timo Schick and Hinrich Schütze. 2021. Generating datasets with pretrained language models. In *EMNLP*.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235, Online. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Ishan Tarunesh, Somak Aditya, and Monojit Choudhury. 2021. Lonli: An extensible framework for testing diverse logical reasoning capabilities for nli. In *Thirty-Sixth AAAI Conference on Artificial Intelligence (AAAI-22)*, volume abs/2112.02333.

Betty van Aken, Sebastian Herrmann, and Alexander Löser. 2021. What do you see in this patient? behavioral testing of clinical nlp models. *NeurIPS 2021 Research2Clinics Workshop, Bridging the Gap: From Machine Learning Research to Clinical Practice*.

Jun Wang, Chang Xu, Francisco Guzmán, Ahmed El-Kishky, Benjamin Rubinstein, and Trevor Cohn. 2021. As easy as 1, 2, 3: Behavioural testing of NMT systems for numerical translation. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4711–4717, Online. Association for Computational Linguistics.

Peter West, Chandrasekhar Bhagavatula, Jack Hessel, Jena D. Hwang, Liwei Jiang, Ronan Le Bras, Ximing Lu, Sean Welleck, and Yejin Choi. 2021. Symbolic knowledge distillation: from general language models to commonsense models. *ArXiv*, abs/2110.07178.

Hitomi Yanaka, Koji Mineshima, Daisuke Bekki, Kentaro Inui, Satoshi Sekine, Lasha Abzianidze, and Johan Bos. 2019. HELP: A dataset for identifying shortcomings of neural models in monotonicity reasoning. In *Proceedings of the Eighth Joint Conference on Lexical and Computational Semantics (*SEM 2019)*, pages 250–255, Minneapolis, Minnesota. Association for Computational Linguistics.

Jie M. Zhang, Mark Harman, Lei Ma, and Yang Liu. 2019. Machine learning testing: Survey, landscapes and horizons. *ArXiv*, abs/1906.10742.

Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. 2018. Texygen: A benchmarking platform for text generation models. *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*.

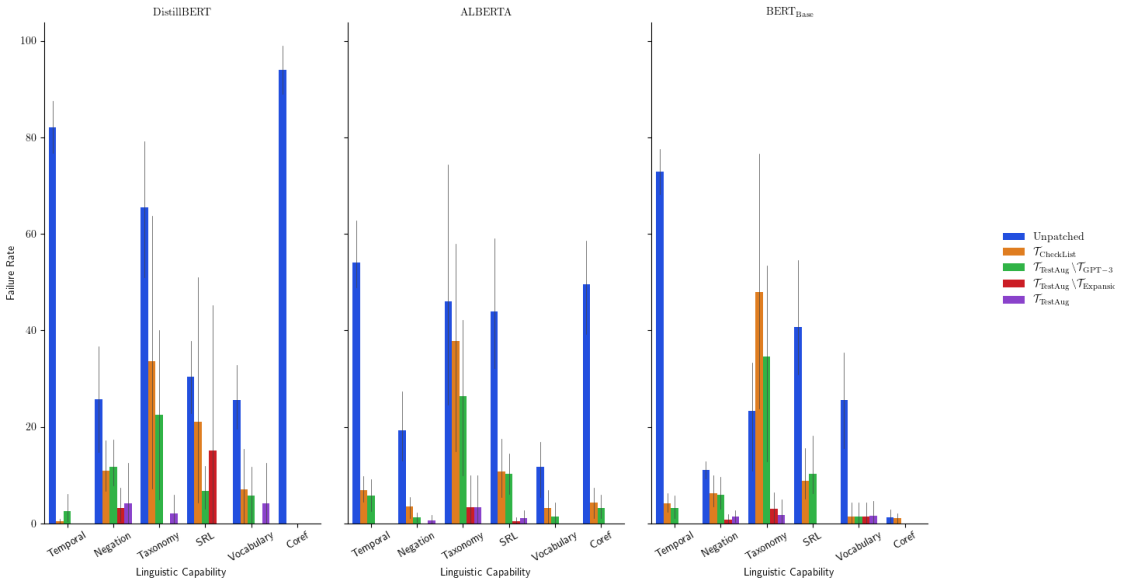
A Appendix

Table 7: Prompt designs for paraphrase detection and natural language inference tasks.

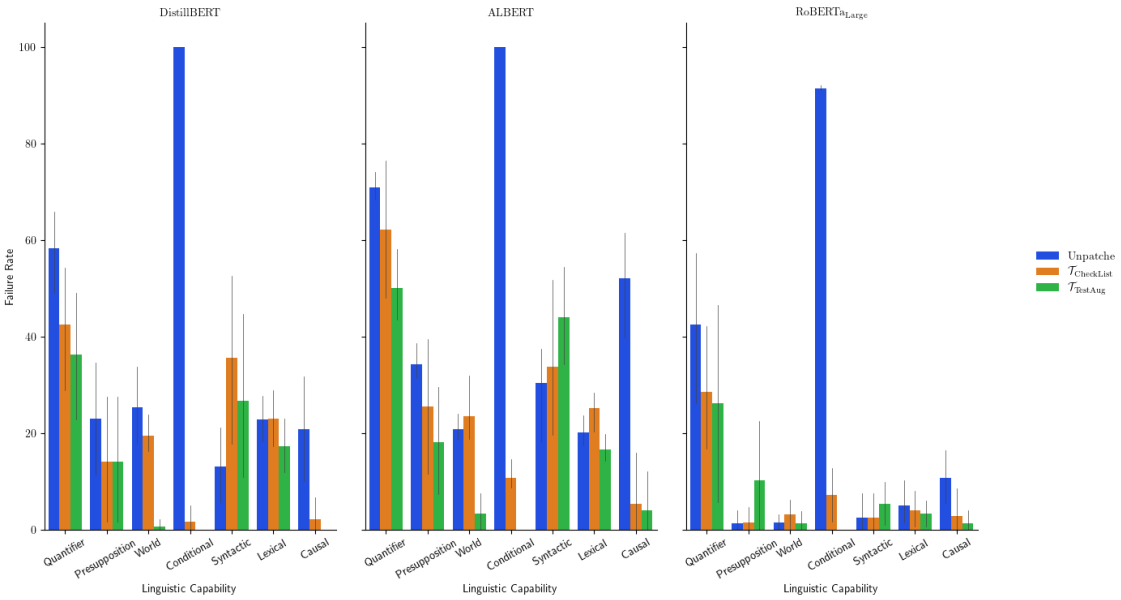
Paraphrase Detection
`Two sentences are equivalent when using according to.` *- {{ Who do analysts think is the smartest footballer in the world? }* *- { Who is the smartest footballer in the world according to analysts? }}* *- {{ Who do students think is the top woman in the world? }* *- { Who is the top woman in the world according to students? }}* *- {{ Who do readers think is the worst gamer in the world? }* *- { Who is the worst gamer in the world according to readers? }}* *- {{* `What does the data say about the most popular baby names? }` `- { What are the most popular baby names according to the data? }`
Natural Language Inference
`Write a pair of sentences that have the same relationship as the previous examples. Examples:` *- {{ Philip, Charles and Colin are the only children of Henry. }* *- { Henry has exactly 3 children. }}* *- {{ Grace, Thomas and Helen are the only children of Andrea. }* *- { Andrea has exactly 3 children. }}* *- {{ Don has 2 dollars. He received 8 more dollars. }* *- { Don now has 10 dollars. }}* *- {{* `Mary has a cat. She also has a dog. }` `- { Mary has two pets. }}`

Table 8: Hyperperparamer choice for model fine-tuning

Hyperparameter	Value
Learning rate	$5e-6$
Batch size	16
Number of training epochs	10
Max. sequence length	128
Seed	42



(a) Paraphrase Detection



(b) Natural Language Inference

Figure 3: Capability-wise error rates on paraphrase detection and natural language inference tasks.

Table 9: Linguistic capabilities that appeared in the experiments and their explanations with corresponding examples. The description of the tested linguistic capabilities follow the taxonomy and definition provided in previous work (Ribeiro et al., 2020; Tarunesh et al., 2021; Joshi et al., 2020b).

Linguistic Capability	Explanation	Template and Example
		Sentiment Analysis
Negation	Resolving "not"	{neg} {pos_verb_present} {the} {air_noun}. I didn't admire that service.
SRL	Resolving subjects and objects	Do I think {it} {be} {a:pos_adj} {air_noun}? No Do I think that is an amazing aircraft? No
Temporal	Resolving temporal changes	I {neg_verb_present} this airline, {change} in the past I would {pos_verb_present} it. I regret this airline, although in the past I would appreciate it.
Vocabulary	Resolving word choice variations	{it} {air_noun} {be} {neg_adj}. That food was ugly.
		Paraphrase Detection
Coref	Resolving male and female names	S1: If {female} and {male} were alone, do you think he would reject her? S2: If {female} and {male} were alone, do you think she would reject him? S1: If Julie and Roy were alone, do you think he would reject her? S2: If Julie and Roy were alone, do you think she would reject him?
Negation	Resolving "not"	S1: What are things {a:noun} should worry about? S2: What are things {a:noun} should not worry about? S1: What are things a friend should worry about? S2: What are things a friend should not worry about?
SRL	Resolving subjects and objects	S1: Did {first_name} {verb[0]} the {obj}? S2: Was {first_name} {verb[1]} by the {obj}? S1: Did Sam remember the factory? S2: Was Sam remembered by the factory?
Taxonomy	Resolving external taxonomic hierarchy	S1: How can I become {a:x[1]} person? S2: How can I become {a:x[0]} person? S1: How can I become a frightened person? S2: How can I become a scared person?
Temporal	Resolving temporal changes	S1: Is {first_name} {last_name} {a:noun}? S2: Did {first_name} {last_name} use to be {a:noun}? S1: Is Dorothy Clarke an agent? S2: Did Dorothy Clarke use to be an agent?
Vocabulary	Resolving word choice variations	S1: How can I become less {x[1]}? S2: How can I become more {x[1]}? S1: How can I become less active? S2: How can I become more active?
		Natural Language Inference
Causal	Resolving actions between one object and two entities	P: {NAME} moved the {OBJECT_TABLE} to {LOCATION_HOUSE1} from {LOCATION_HOUSE2}. H: The {OBJECT_TABLE} is now in {LOCATION_HOUSE1}. P: George moved the notebook to study room from bedroom. H: The notebook is now in study room.
Conditional	Resolving reasoning over conditions	P: If {NAME1} comes to the {LOCATION}, {NAME2} won't come. {NAME1} did not come to the {LOCATION}. H: {NAME2} didn't come to the {LOCATION}. P: If Kim comes to the park, William won't come. Kim did not come to the park. H: William didn't come to the park.
Lexical	Resolving word choice variations	P: {NAME} was born in {COUNTRY1}. H: {NAME} is born in {COUNTRY2}. P: Emily was born in Germany. H: Emily is born in Malaysia.
Presupposition	Resolving implications	P: {NAME}'s {T12_RELATION} is {ADJECTIVE_OF_PERSON}. H: {NAME} has {A} {T12_RELATION}. P: Florence's brother is intolerant. H: Florence has a brother.
Quantifier	Resolving "all" (universal quantification) and "some", "none" (existential quantification)	P: None of the {OBJECTS} are {COLOUR1} in colour. H: Some of the {OBJECTS} are {COLOUR2} in colour. P: None of the cars are maroon in colour. H: Some of the cars are pink in colour.
Syntactic	Resolving ellipsis	P: {NAME} tried but wasn't able to {VERB}. H: {NAME} didn't try to {VERB}. P: Alan tried but wasn't able to give. H: Alan didn't try to give.
World	Resolving world knowledge such as geography	P: {NAME} lives in {T4_CAPITAL1}. H: {NAME} lives in {T4_COUNTRY2}. P: Ken lives in Kathmandu. H: Ken lives in North Korea.

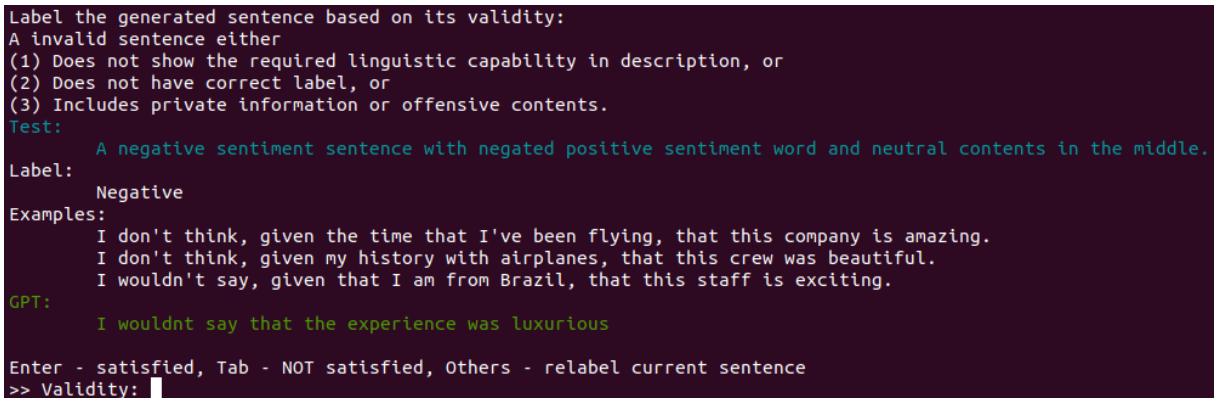


Figure 4: The command line interface for data annotation. Annotators are given a test and three associated test cases from the template-based test suite; they are asked to the annotate the validity of the GPT-3-generated sentences. Annotators are reminded of the guidelines for filtering invalid samples when labeling each sentence (shown at the top of the interface). We communicated explicitly for the intended uses of the annotated datasets before the annotation.

Table 10: The fine-tuned models we evaluated in this paper.

Model name	Task	Size	Checkpoint Identifier
DistillBERT	Sentiment Analysis	Small	`textattack/distilbert-base-cased-SST-2`
ALBERT	Sentiment Analysis	Small	`textattack/albert-base-v2-SST-2`
$\text{BERT}_{\text{Base}}$	Sentiment Analysis	Base	`textattack/bert-base-uncased-SST-2`
$\text{RoBERTa}_{\text{Base}}$	Sentiment Analysis	Base	`textattack/roberta-base-SST-2`
DistillBERT	Paraphrase Detection	Small	`textattack/distilbert-base-cased-QQP`
ALBERTA	Paraphrase Detection	Small	`textattack/albert-base-v2-QQP`
$\text{BERT}_{\text{Base}}$	Paraphrase Detection	Base	`textattack/bert-base-uncased-QQP`
DistillBERT	Natural Language Inference	Small	`textattack/distilbert-base-cased-snli`
ALBERT	Natural Language Inference	Small	`textattack/albert-base-v2-snli`
$\text{RoBERTa}_{\text{Large}}$	Natural Language Inference	Large	`ynie/roberta-large-snli_mnli_fever_anli_R1_R2_R3-nli`

Table 11: Proportion of valid sentences and performance of trained classifiers for automatic filtering.

Linguistic Capability	Test	Accuracy	F1	Proportion of Valid Test Cases
	Sentiment Analysis			
SRL	A negative sentiment sentence with negative sentiment question and word yes as the answer.	/	/	1.000
SRL	A positive sentiment sentence with positive sentiment question and word yes as the answer.	/	/	0.994
SRL	My opinion is more important than others' when expressing positive sentiment.	/	/	0.948
SRL	A negative sentiment sentence with positive sentiment question and word no as the answer.	/	/	0.928
Temporal	I used to have negative sentiment to something, but now I have positive sentiment to it.	/	/	0.922
Negation	A negative sentiment sentence with negated positive word.	/	/	0.910
SRL	My opinion is more important than others' when expressing negative sentiment.	/	/	0.900
Temporal	I used to have positive sentiment to something, but now I have negative sentiment to it.	0.942	0.967	0.873
Negation	I thought something was positive, but it was negative.	0.934	0.961	0.860
Vocabulary	A negative sentiment sentence with negative words.	0.897	0.932	0.804
Negation	A negative sentiment sentence with negated positive sentiment word and neutral contents in the middle.	0.915	0.935	0.653
	Paraphrase Detection			
Temporal	Two sentences are different when talking about a person's current job and his or her previous job.	/	/	1.000
Negation	Two sentences are different when talking about someone should and should not do something.	/	/	0.973
Temporal	Two sentences are different when talking about a person's current job and his or her future job.	/	/	0.964
Vocabulary	Two sentences are different when adjectives are modified by more and less.	/	/	0.912
Taxonomy	Two sentences are equivalent when the nouns are modified by synonymous adjectives.	0.889	0.933	0.885
Coref	Two sentences are different when swapping the subjects and objects.	0.980	0.989	0.843
Temporal	Two sentences are different when talking about doing something before and after another thing.	0.982	0.989	0.800
Temporal	Two sentences are different when describing doing something before and after some specific time.	0.982	0.988	0.782
Negation	Two sentences are different when talking about the properties of doing or not doing something.	0.965	0.977	0.754
Vocabulary	A sentence with the noun modified by an adjective is equivalent to the sentence without adjective.	0.920	0.946	0.740
SRL	Two sentences are equivalent when using according to.	0.958	0.971	0.708
Negation	Two sentences are different when describing a person with adjective and a clause including the negation of the same adjective.	0.871	0.913	0.690
Taxonomy	Two sentences are equivalent when one has an adjective modified by more and the other one has an antonym modified by less.	0.920	0.909	0.511
Coref	Two sentences are different when referring someone's family using different pronouns.	0.940	0.940	0.500
SRL	Two sentences are different when swapping active and passive action.	0.930	0.933	0.471
	Natural Language Inference			
Lexical	/	/	/	0.975
Syntactic	/	/	/	0.936
Presupposition	/	/	/	0.903
World	/	/	/	0.902
Quantifier	/	/	/	0.895
Causal	/	/	/	0.857
Conditional	/	/	/	0.834

Table 12: Creating new templates based on test cases generated by GPT-3.

Original Template and Test Case	Generated Test Case	New Template
	Sentiment Analysis	
No one {pos_verb_present}s {the} {air_noun}. No one enjoys that seat.	This is not an easy service to appreciate. That customer service was not fun. I don't think your customer service is admired.	This is not an easy {air_noun} to appreciate. That customer {air_noun} was not fun. I don't think your customer {air_noun} is admired.
	Paraphrase Detection	
Is it {mid} to {activity} before {hour}{ampm}? Is it {mid} to {activity} after {hour}{ampm}? Is it healthy to drink before 10am? Is it healthy to drink after 10am?	Is it bad to drink before 8pm Is it bad to drink after 8pm Is it acceptable to drink before 2pm Is it acceptable to drink after 2pm Is it advisable to eat before 8pm Is it advisable to eat after 8pm	Is it bad to {activity} before 8pm Is it bad to {activity} after 8pm Is it {mid} to {activity} before 2pm Is it {mid} to {activity} after 2pm Is it advisable to {activity} before 8pm Is it advisable to {activity} after 8pm