TestAug: A Framework for Augmenting Capability-based NLP Tests

Anonymous ACL submission

Abstract

The recently proposed capability-based NLP tests go beyond the traditional held-out evaluation paradigm, allowing model developers to test the different *linguistic capabilities* of a model. However, existing work on capability-based testing requires the (semi-)manual creation of the test suites (templates); such approach thus heavily relies on the linguistic expertise and domain expertise of the developers. In this paper, we investigate an automatic approach for generating and augmenting the test suites by prompting the GPT-3 engine. Our experiments show that our approach can generate *diverse* test suites which has a better coverage than the existing approaches using templates. The augmented test suites can also be used to detect more errors compared to existing work. Our test suites can be downloaded at `https://anonymous-researcher-nlp.github.io/testaug/`.

1 Introduction

In the recent years, natural language processing has seen tremendous advancement in the model performances. Conventional approaches of evaluating NLP models' performance rely on reporting aggregate metrics such as accuracy and F-1 score on the held-out dataset. However, such performance estimations may fail to provide the complete information: high metric scores could be a result of less representative data than the data in the wild (for example, models are exploiting annotation bias or other types of shortcuts in the experiment data (Geva et al., 2019; Gururangan et al., 2018; Bai et al., 2021)), while low metric scores do not tell what exact shortcomings the model has. Furthermore, recent studies show that even stress-tested industrial models may not be truly linguistically capable: they fail on simple and non-adversarial test cases (Glockner et al., 2018; Ribeiro et al., 2020).

Table 1: Example test cases for three NLP tasks: sentiment analysis, paraphrase detection, and natural language inference.

Task: Sentiment Analysis
Description: Negated positive word **Input**: "No one loves the food." **Label**: *Negative*
Task: Paraphrase Detection
Description: Negation of antonym **Input**: "She is a generous person. She is not a mean person." **Label**: *Paraphrase*
Task: Natural Language Inference
Description: Downward entailment **Input**: "Some cows are brown. Some animals are brown." **Label**: *Entailment*

The capability-based testing checks whether an NLP model picks up a linguistic capability such as co-reference, negation, and temporal changes. It starts from a test case description specifying a linguistic capability the NLP model being investigated is expected to have. Then a test set of concrete examples satisfying this test case description is created either manually through crowd-sourcing (Bowman et al., 2015) or semi-automatically through templates (Tarunesh et al., 2021). For example, the test case description "a neutral sentence with neutral words" and the corresponding test cases such as "the company is Australian" are used to test whether a classifier could leverage neutral words for sentiment classification. Multiple such test case descriptions and test sets are aggregated together as a test suite to test an NLP model's overall linguistic capabilities.

The NLP models' capability-based testing have already been addressed for tasks such sentiment classification, paraphrase detection, and natural language understanding (Ribeiro et al., 2020; Tarunesh et al., 2021). However, current approaches of capability-based testing rely on domain experts' efforts and therefore suffer from both scalability and diversity. Specifically, the size of test set depends on the human efforts invested into writing test cases or curating templates, scaling down the number of available test cases. Moreover, test case descriptions do not provide direct instructions for crowd workers to create diverse test cases. The test cases generated in this way often only show diversity in the superficial level. For example, when testing a paraphrase detection model's co-reference capability, the only variation of sentence pair 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?") (Ribeiro et al., 2020).

In this work, we revisit the problem of generating test cases given test case descriptions such as "a negative sentiment sentence with negated positive word". We propose to leverage GPT-3 to address previous approaches' limitations in scalability and diversity. GPT-3 has demonstrated its potential for creative text generation in applications that require diverse texts of in huge quantities, such as composing stories and conversing with humans (Floridi and Chiriatti, 2020).

More specifically, we instructed the GPT-3's variants – `instruct-series` engines [1] – to generate test cases that satisfy the test case descriptions through carefully designed natural language inputs (i.e., prompts); these `instruct-series` engines have been augmented on top of the base GPT-3 to develop an ability to to better follow natural language instructions.

We demonstrate the effectiveness of our approach in testing NLP models of sentiment classification, paraphrase detection, and natural language inference tasks. Specifically, the test suite generated following our approach could better reveal models' erroneous behaviors than the counterparts generated through templates given the same test suite size. Moreover, our test suite has a substantially higher linguistic diversity than the test suite from templates. Further, our test suite is extensible to a larger scale as it is no more constrained by manual templates and lexicons; nonetheless, it could complement the template approach to generate new and diverse templates at scale.

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 out 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 gaining a more comprehensive understanding of the model performance, researchers proposed a new approach of evaluating NLP models, which is 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 approaches (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 is not applicable 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

[1] https://openai.com/

the templates remain a challenging task (Tarunesh et al., 2021; Jeretic et al., 2020). Finally, the CLCD dataset (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 and Generation 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., 2021b) 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. (Mishra et al., 2021) proposed a dataset for teaching GPT-3 and BART (Lewis et al., 2020) to follow instructions. Reynolds and McDonell (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 for generating the capability-based test suites (Section 4).

3 Problem Definition

The capability-based NLP testing starts from a testing subject $\mathcal{M}$ that is already trained and evaluated on respective datasets $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{val}}$; the aggregate metrics such as accuracy and F1-score are reported to indicate that the models' performances are acceptable [2]. The users therefore expect that the model $\mathcal{M}$ has picked up the *linguistic capabilities*, such as properly handling negation and co-reference, to perform well on a different test set.

Following each linguistic capability, a set of *test case descriptions* are created by the users to operationalize the testing of individual capability. A test case description is a natural language description of the test cases that help the crowd workers to manually curate test cases or templates with associated lexicons to fill in. For example, in Table 1, when testing a text classifier's capability to handle negation within sentences, several test case descriptions, focusing on different aspects of negation, are provided by the users, where each helps users generate templates such as "{it} {benot} {a:pos_adj} {air_noun}."; with lexicons ready for each slot, this template may end up as test cases like "That is not a perfect seat.".

The test cases generated following each test case description and the overarching linguistic capability constitute the test suite $\mathcal{T}$. The test suite provides evaluations of $\mathcal{M}$'s linguistic capabilities through test cases specializing in them. Therefore, each $\mathcal{M}$'s prediction error on $\mathcal{T}$ is considered as a *bug*.

Given a list of linguistic capabilities and their test case descriptions, previous approaches heavily rely on manual labor for creating specific test cases or templates and associated lexicons. Despite their preliminary success in revealing model bugs, they suffer from both limited diversity and scalability. We strive to addressing both issues with a preserved and even improved ability of revealing bugs of an NLP model.

4 The TestAug Framework

Starting from the test case descriptions and a few associated seed test cases, we first devise prompts suitable for the given NLP task and for eliciting valid GPT-3 generation. Then we manually check the generated test cases and select valid ones to augment the template-based test suite; these test cases could also be converted into new templates to enrich template-based test suite. 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 Designing Prompts to Instruct GPT-3 to Generate Test Cases

A prompt is a natural language sentence that describes the context of a text generation session using GPT-3; it is set to the test case description in this work. A prompt could work by itself or could be augmented with additional in-context examples (i.e., demonstrations). For example, when generating sentences under the test case description "A negative sentiment sentence with negated

[2] We focus on the text classification task in this work. But this definition could be easily extended to other models from supervised NLP tasks.

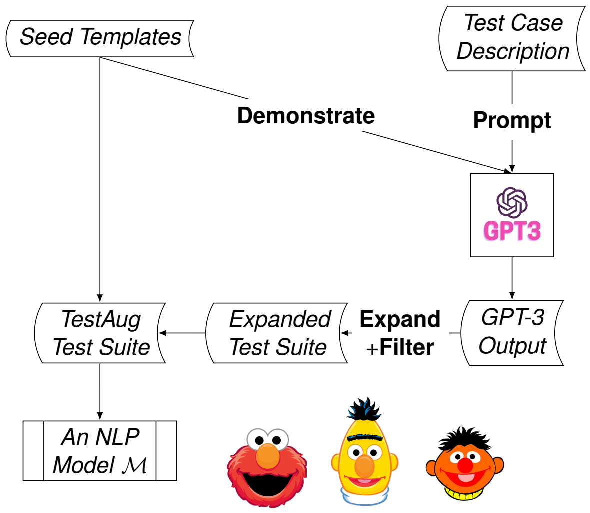

Figure 1: The control-flow graph of TestAug framework.

Table 2: Prompt designs to elicit GPT-3 for test case generation in sentiment analysis tasks. The **test case description** specifics the context of generation; the *in-context examples* 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. }
```

positive word", three in-context examples meeting this test case description are provided to make GPT-3 better understand the desired outcomes. It has been shown that such augmentation is conducive for generating more complex texts without violating users' expectation specified in the prompt (Liu et al., 2022). In our example, a new *valid* sentence "No one appreciates that air traffic controller." is generated by GPT-3 (Table 2).

Despite its powerful text generation ability, the outputs of the GPT-3 heavily depend on the structure and contents of prompts: it has been observed that how users write the description, the number of in-context examples, and their structures significantly influence the validity of the output sentences with regard to the test case description (Liu et al., 2021). This observation motivates the study of prompt engineering, whose goal is to elicit the GPT-3 to generate texts that satisfy the test case descriptions.

In this work, we designed our prompts (Table 2 and Table 9) following previous practices of eliciting GPT-3 for dataset creation (Liu et al., 2022; Reif et al., 2021; West et al., 2021; Schick and Schütze, 2021; Reynolds and McDonell, 2021b). Specifically, starting from seed test cases sampled from template-based test suite $\mathcal{T}_{\text{Template}}$, we formatted the prompt and the in-context examples following the guidelines below:

Natural Language Description. The natural language description describes the context of a generation session with GPT-3. For the natural language inference task, we used the fixed description "Write a pair of sentences that have the same relationship as the previous examples. Examples:" following previous work of natural language inference dataset creation (Liu et al., 2022).

In-context Examples. The in-context examples augment the natural language description to inform GPT-3 about the scope and format of the desired sentences. We sample in-context examples (i.e., seed sentences) from the existing template-based test suites.

Formatting. The in-context examples have been formatted as an unordered list, which drives GPT-3 working on the completion of the list. The paired brackets are used to indicate sentence (or sentence pair) boundaries between consecutive examples; GPT-3 could hence better distinguish different examples and constrain its possible continuation by terminating generation on the brackets; at the same time, users could leverage brackets to fetch returned results without confounding different sentences.

4.2 Augmenting Template-based Test Suite with GPT-3 Generated Test Cases

The test cases generated by GPT-3 may fail to satisfy test case descriptions as they 1) may repeat in-context examples, 2) does 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, 3) does not fulfill the test case descriptions expressed in the prompts; for example, the generated test case ("Joe isn't at the party.", "Joe is at the party.") 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." is also incorrect as it does not convey the expected sentiment change defined in the test case description "I thought something was negative, but it was neutral." for sentiment classification task.

Previous works used dastset cartography or a separate classifier to automatically filter out texts failing the expectation (West et al., 2021; Liu et al., 2022). However, dataset cartography requires multiple checkpoints saved during training to estimate a sample's uncertainty while training a text classifier requires a large number of negative samples to build a balanced training set; therefore, neither of the approaches are applicable to our settings: we do not assume access to the checkpoints saved during training and the negative samples are scarce (Table 5). In this work, rather than using the end-to-end automatic filtering, we resorted to human-in-the-loop filtering. Specifically, we trained a ranker that is designed to rank invalid cases before valid ones. With the help of the ranker, the human annotators only need to manually investigate top $k\%$ of the data to exclude the invalid test cases; the bottom $(1-k)\%$ of the test cases are valid with high probability. We fine-tuned an ensemble of language models to rank the test cases (Gao et al., 2021a).

4.3 Expanding Templates in Template-based Test Suite

The slots in the templates that generate $\mathcal{T}_{\text{Template}}$ capture the key linguistic capabilities; for example, the slots {pos_verb_present} and {pos_adj} correspond to the positive words specified in the test case description (Table 1). Furthermore, the GPT-3 generation follows the provided in-context examples, making some of the words reappear in the new test cases; therefore, test cases in the test suite $\mathcal{T}_{\text{GPT-3}}$ could be converted to new templates based on these repeated words.

Specifically, we compared each of the generated words with each word in the in-context examples, if a slot word in a in-context example reappears in the generated test case, we converted the generated word as a new slot, leading to a new template. For example, "No one appreciates that air traffic controller." is generated following the prompt shown in sentiment classification task of Table 2; as "appreciates" repeats the one in the in-context example "No one appreciates that airline.", a new template "No one {pos_verb}s that air traffic controller." is generated following the template "No one {pos_verb_present}s {the} {air_noun}." As misplaced pronouns yield insensible 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.

By converting GPT-3 generated test cases into new templates, we enrich the number of templates available for $\mathcal{T}_{\text{Template}}$.

5 Experiment

In this section, we evaluate the effectiveness of TestAug. TestAug is a capability-based testing framework that can generate a large number of test cases satisfying a description by the developer, with only a small amount of expert annotations as the demonstration. To examine the effectiveness, we first investigate TestAug's ability in detecting the model failures, and compare its performance with existing work (Ribeiro et al., 2020; Tarunesh et al., 2021) (Section 5.2). Second, since TestAug is based on GPT-3, it is expected to show better linguistic diversity than existing work where all test cases are from templates (Ribeiro et al., 2020; Tarunesh et al., 2021). We thus also quantitatively investigate the diversity of test cases (Section 5.3). Finally, since TestAug cases are automatically generated, we also need to investigate the validity of the generated cases, e.g., how many percentages of TestAug cases do not satisfy the description? (Section 5.4). Before reporting these investigation results, we first explain our experimental settings in Section 5.1.

5.1 Experiment Settings

Evaluated Tasks. Existing work on capability-based testing has studied the following three tasks: sentiment classification (Ribeiro et al., 2020), paraphrase detection (Ribeiro et al., 2020), and natural language inference (Tarunesh et al., 2021). To compare TestAug's performance with existing work, we study the same three tasks. We skip question answering (Ribeiro et al., 2020) as we find it difficult for TestAug to generate valid test cases for question answering, potentially because QA cases involve more components. We also skip hate speech detection (Röttger et al., 2021) because in practice, we found GPT-3 cannot be used for generating cases containing profanity words[3]. We leave QA and

[3] In particular, when we tried to augment HATECHECK's (Röttger et al., 2021) test cases from template "[IDENTITY_P] can all go [redacted] themselves", where "[redacted]" is a profane word that is offensive in nature. The request to generate test cases is

hate speech detection for future work.

Evaluated Models. Following previous work, we test models that have leading healdout performances (Ribeiro et al., 2020). To evaluate more models, we leverage the publicly available fine-tuned models from the HuggingFace model hub [4]. For each task, we first selected the most downloaded fine-tuned models on the model hub. Within them, we evaluated each model's validation error rate, and keep the best models while also balancing models of different sizes. A complete list of the selected models can be found in Table 11; their heldout validation error rates can be found in the column ERR% of Table 3.

5.2 Evaluating TestAug's Ability for Bug Detection

In this section, we evaluate TestAug's ability for detecting bugs, and compare its performance with existing work (Ribeiro et al., 2020; Tarunesh et al., 2021).

5.2.1 Evaluation Method

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 simplest approach is to directly comparing the error rates of the same model on the two test suites. Despite the simplicity, we argue that these two error rates are in fact *incomparable*. The reason is below: the effectiveness of a test suite is defined by how many bugs it can find (Kochhar et al., 2015). As a result, if a test suite has a higher error rate but fewer error cases, it is uncertain whether it has a better performance.

To make the two metrics comparable, we propose to evaluate a test suite by leveraging its fine-tuned model's error rate. More specifically: (1) first, we merge the two test suites $\mathcal{T}_{\mathrm{A}}$ and $\mathcal{T}_{\mathrm{B}}$ into a large suite $\mathcal{T}$; (2) second, we randomly partition $\mathcal{T}$ into a training suite $\mathcal{T}_{\mathrm{train}}$ and a testing suite $\mathcal{T}_{\mathrm{test}}$; (3) third, we fine-tuning the model using $\mathcal{T}_{\mathrm{train}} \cap \mathcal{T}_{\mathrm{A}}$, testing its performances on $\mathcal{T}_{\mathrm{test}}$, and compare with when fine-tuned with $\mathcal{T}_{\mathrm{train}} \cap \mathcal{T}_{\mathrm{B}}$. The advantage of our proposed metric is that the two scores are both tested on the same testing data, thus a lower testing error indicates the fine-tuning process has successfully patched more errors, and as a result, more errors have been found by that test suite. We also report the error rate before the find-tuning and the reduction in the error rate.

5.2.2 Evaluation Results

We tested the models that had already shown acceptable accuracy on the original held-out dataset. The results in Table 3[5, 6] show that our test suites $\mathcal{T}_{\mathrm{TestAug}}$ **consistently** augmented template-base test suites to reduce error rates. When looking at error rates per linguistic capability (Table 10), we could see that the augmented test suites $\mathcal{T}_{\mathrm{TestAug}}$ are effective in enhancing capability-based testing in most of the cases: the $\mathcal{T}_{\mathrm{TestAug}}$'s error rates $\mathrm{ERR\%}_{\mathrm{Patched}}$ are smaller than other test suites in all linguistic capabilities except the "negation" in the paraphrase detection task, leading to a higher error rate reduction $\Delta_{\mathrm{ERR\%}}$.

We investigated the curious case mentioned above. In the "negation" capability of the paraphrase detection task, the error rates for patched models remain same (12.3%) across four different test suites; we found that the specific error cases were also same regardless which test suite was used to patch the model (Figure 3). This shows that, despite overall strength to reveal more bugs for a given *task* (Table 3), the TestAug is not guaranteed to generate competitive test cases in all *linguistic capabilities*.

Following the approach described in Section 4, we created new templates to enrich the original pool of templates available for $\mathcal{T}_{\mathrm{Template}}$. When generating new templates, restricting new slots on only reappeared words in the original templates decrease the possibility of generating invalid templates (Table 12). We manually sampled and annotated 100 generated templates per task and found that the valid templates constitute 91%, 89%, and 91% of all templates for sentiment classification, paraphrase detection, and natural language inference tasks. The invalid templates mostly come from invalid modifiers such as "{ADJECTIVE_OF_PERSON}" in "Some of the creams are {ADJECTIVE_OF_PERSON} in colour.", where

denied with a flagged warning message: "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.".

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

[5] For the sentiment classification task, as our experiments require further fine-tuning on top of the already fine-tuned models; while the number of output classes do not match, we replaced the final 2-class classification layer with a newly initialized 3-class classification layer and fine-tuned the model for additional 3 epochs. We used the discretized 3-class SST dataset for this further fine-tuning.

[6] The expansion of test cases in NLI task requires alternative approaches and we leave it as future work.

Table 3: Model accuracy on held-out validation set and their overall error rate reduction using different test suites. The accuracy ACC% is computed over the original held-out dataset. The error rate reduction $\Delta_{\text{ERR}\%} = \text{ERR}\%_{\text{Unpatched}} - \text{ERR}\%_{\text{Patched}}$ follows the evaluation metrics introduced in Section 5.2. The exact identifiers of model checkpoints we used in experiments are listed in Table 11. Some cells are marked with "/" as we leave template expansion of the NLI task as future work. We used a small subset of original template-based test suite as demonstrations and the percentage is shown beside the task name.

	ERR%	ERR%$_{\text{Unpatched}}$	$\mathcal{T}_{\text{TestAug}}$		$\mathcal{T}_{\text{TestAug}} \setminus \mathcal{T}_{\text{Template}}$		$\mathcal{T}_{\text{TestAug}} \setminus \mathcal{T}_{\text{Expansion}}$		$\mathcal{T}_{\text{Template}}$	
			ERR%$_{\text{Patched}}$	$\Delta_{\text{ERR}\%}$	ERR%$_{\text{Patched}}$	$\Delta_{\text{ERR}\%}$	ERR%$_{\text{Patched}}$	$\Delta_{\text{ERR}\%}$	ERR%$_{\text{Patched}}$	$\Delta_{\text{ERR}\%}$
Sentiment Analysis 183 / 28921 = 0.6%										
DistillBERT	10.0	42.4	**8.5**	**33.9**	15.0	27.4	15.4	27.0	31.0	11.4
ALBERT	7.3	41.6	**6.5**	**35.1**	12.7	28.9	17.4	24.2	29.0	12.6
BERT$_{\text{Base}}$	7.6	40.9	**4.1**	**36.8**	7.5	33.4	6.2	34.7	16.5	24.4
RoBERTa$_{\text{Base}}$	5.7	36.6	**4.4**	**32.1**	6.3	30.2	6.0	30.6	10.2	26.3
Paraphrase Detection 54 / 11126 = 0.5%										
DistillBERT	10.3	45.4	**3.7**	**41.8**	8.8	36.6	6.5	38.9	11.9	33.5
ALBERTA	9.3	45.3	**9.2**	**36.0**	14.2	31.1	12.1	33.1	15.8	29.5
BERT$_{\text{Base}}$	9.1	51.6	**2.7**	**48.9**	5.2	46.4	4.2	47.3	8.0	43.6
Natural Language Inference 240 / 347531 = 0.1%										
DistillBERT	12.6	49.5	/	/	/	/	27.4	22.1	36.3	13.2
ALBERT	9.9	45.0	/	/	/	/	21.0	24.0	28.7	16.4
RoBERTa$_{\text{Large}}$	8.1	32.2	/	/	/	/	10.5	21.7	15.8	16.5

adjectives for people are misused for creams since slots are created oblivious of the contexts. Despite some invalid templates, template expansion leverages the improved scale and diversity of test cases and scales up the creation of template-based test suite $\mathcal{T}_{\text{Template}}$.

5.3 Evaluating the Diversity of TestAug Results

5.3.1 Evaluation Method

The linguistic diversity of an NLP test suite could be measured either from the test case level or the test suite level. We introduce the number of unique dependency paths as a proxy for linguistic diversity for each individual test case; while in test suite level, we use the metric for diversity in natural language generation – Self-BLEU.

Number of Unique Dependency Paths. Dependency parsing of a sentence returns a directed tree where there is a unique path from the root and every vertex. The arcs in the dependency tree are attributed with a fixed set of grammatical relations. The dependency tree approximates the semantic relations between predicates and their arguments (Jurafsky and Martin, 2000). We therefore propose to use the number of unique dependency paths to measure the richness of semantic relations.

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, lower Self-BLEU score indicates a higher diversity of the sentence.

5.3.2 Evaluation Results

The test cases the annotators unanimously deemed consistent with the given test case description constitute test suites for respective tasks. After controlling for the number of test cases under each test case description, the linguistic diversity (Table 4) of the test suites $\mathcal{T}_{\text{GPT}-3}$ show substantial improvement over the template-based counterparts $\mathcal{T}_{\text{Template}}$: the Self-BLEU4 score has an decrease of at least 9.4% (the paraphrase detection task) and the number of unique dependency paths is of at least 2.18 times compared to the original test suite (the natural language inference task).

5.4 Evaluating the Validity of TestAug Results

5.4.1 Evaluation Method

Our experiments require test cases that have been verified consistent with the given test case description. Rather than creating templates or test cases from scratch, the human annotators in our system take a more efficient and effective role to correct mistakes made by the GPT-3. Specifically, we worked with human annotators to annotate each

Table 4: Linguistic diversity of test suites.

	Self-BLEU4 (↓)	Number of Unique Dependency Paths (↑)
Sentiment Analysis		
$\mathcal{T}_{\text{GPT}-3}$	0.558	548
$\mathcal{T}_{\text{Template}}$	0.778	88
Paraphrase Detection		
$\mathcal{T}_{\text{GPT}-3}$	0.587	957
$\mathcal{T}_{\text{Template}}$	0.645	113
Natural Language Inference		
$\mathcal{T}_{\text{GPT}-3}$	0.412	692
$\mathcal{T}_{\text{Template}}$	0.514	317

generated test case and decided whether it would be used for testing the NLP model.

5.4.2 Evaluation Results

A test case that satisfies the given test case description expresses the linguistic capability without grammatical errors. Two of the authors manually labeled each test case by checking whether it satisfied the given description; the test case they did not unanimously agree upon were considered ambiguous and therefore discarded. We used Cohen's κ to measure the agreement of annotation. The annotation interface is shown in Figure 4.

We instructed GPT-3 to generate test cases with linguistic capabilities and seed sentences from CheckList and LoNLI dataset [7] (Ribeiro et al., 2020; Tarunesh et al., 2021). The annotators manually checked whether each test case satisfied the test case description; the Cohen's κ ranges between 0.434 and 0.450 for three tasks (Table 5), indicating moderate agreement (McHugh, 2012). The samples the annotators did not agree upon were discarded, leading to a test case description satisfiability from 74.5% to 84.5%; this shows that a significant portion of test cases generated following our approach satisfy the test case description.

Table 5: Annotation statistics on the test case description satisfiability.

	Cohen's κ	Satisfiability (%)
Sentiment Analysis	0.434	82.2
Paraphrase Detection	0.450	84.5
Natural Language Inference	0.437	74.5

In order to reduce the required manual efforts while maintaining high annotation accuracy, we trained an ensemble of rankers with three large base language models and a ranking loss (Gao et al., 2021a); the rankers were trained to optimize the relative rankings of invalid and valid test cases, pushing the invalid ones up to the top (full details in Appendix A.2). With the help of this ranker ensemble, the annotators only need to check the test cases ranking at the top; the remaining test cases are all considered valid. To measure the annotation accuracy under this setting, we define ACC@k as in Equation 2: only the minority samples after rank k are assigned incorrect labels while all the other samples are annotated correctly.

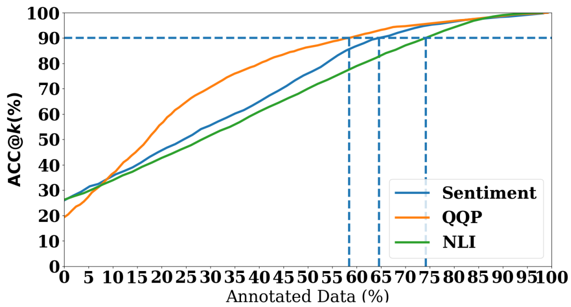

Figure 2: Ranker-assisting annotation accuracy versus annotation efforts.

$$\text{ACC@}k = 1 - \frac{\sum_{i=k+1}^{N} 1(\hat{y}_i = l)}{N} \qquad (2)$$

where N is total number of test cases, l is the majority label in the training set (in our case, the "valid" label), and $\hat{y}_i$ is the validity label of the test case ranked at i-th position based on the ranking score. The assistance of this ranker helps reduce required annotation to maintain a 90% accuracy by ~25% to ~40% (Figure 2).

6 Discussion, Conclusions and Future Work

We introduced the TestAug framework to augment capability-based test suites to better reveal NLP models' shortcomings in linguistic capabilities; empirical results have demonstrated the effectiveness of our framework. Looking forward, we plan to extend the set of NLP tasks supported by TestAug to more challenging tasks such as question answering (QA). We are also interested in further reducing manual efforts in TestAug by automating prompt design used for eliciting GPT-3.

[7] Both datasets are under MIT license.

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). The estimated time commitment for labeling is 20 hours per annotator for a total of 8172 sentences (or sentence pairs). We acknowledge the annotators' efforts with a 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 set of test cases are then used altogether to evaluate a 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 [8].

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 a 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.

L. Floridi and Massimo Chiriatti. 2020. Gpt-3: Its nature, scope, limits, and consequences. *Minds and Machines*, 30:681–694.

Luyu Gao, Zhuyun Dai, and Jamie Callan. 2021a. Rethink training of bert rerankers in multi-stage retrieval pipeline. In *ECIR*.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021b. 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.

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

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.

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.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2021. What makes good in-context examples for gpt-3? *ArXiv*, abs/2101.06804.

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.

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.

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

A.1 Details of Testing NLP Models

The error rate $\mathrm{ERR}\%$ in Table 3 was obtained on the original validation split for sentiment analysis and paraphrase detection tasks and test split of natural language inference task; the statistics of these datasets are reported in Table 7.

The models used for testing are reported in Table 11. The test cases used to evaluate these models were those both annotators considered valid with respect to the test case descriptions. The statistics are shown in Table 6 (i.e., the "valid" column). During model testing, the test suites were partitioned according Section 5.2 for $\mathcal{T}_{\text{TestAug}}$. We used one Nvidia Tesla V100 (with 32 GB graphical memory) throughout the experiments. Following Section 5.2, one testing session (one model of one task) involves inference on the test split twice and fine-tuning once on the training split, which takes fewer than 5 minutes even for the large models. The hyperparameters used for fine-tuning is fixed and reported in Table 8.

A.2 Details of Training Rankers

We used the models listed below as base models and trained an ensemble of rankers following the setting that achieves the best perofmrnace in (Gao et al., 2021a) [9].

- `bert-large-uncased`
- `google/electra-large-discriminator`
- `facebook/muppet-roberta-large`

We increased the number of training epochs from 2 to 10 but left other choices to the authors' defaults. Each trained ranker returned a ranking list with a potentially different ranking of the same document. We aggregated ranking lists returned by three rankers with the Borda count aggregator implemented in `pyrankagg` [10]. We used the same hardware as testing NLP models in Section A.1.

[9] Our implementation is based on the authors' code release: https://github.com/luyug/Reranker

[10] https://github.com/thelahunginjeet/pyrankagg

Table 6: Statistics of test suites generated with GPT-3.

Task	Valid	Invalid
Sentiment Analysis	1607	347
Paraphrase Detection	1916	352
Natural Language Inference	2942	1008

Table 7: Statistics of datasets used to evaluate models' performances on held-out dataset.

Task	Split	Size	Dataset Identifier
Sentiment Analysis	Validation	872	`sst2`
Paraphrase Detection	Validation	40430	`qqp`
Natural Language Inference	Test	10000	`snli`

Table 8: Hyperperparamer choice for model fine-tuning

Hyperparameter	Value
Learning rate	$5e-6$
Batch size	8
Number of training epochs	3
Max. sequence length	128
Seed	42

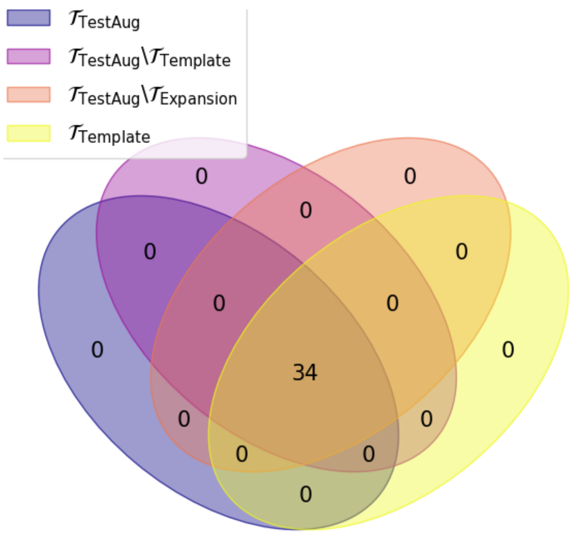

Figure 3: The error cases of the test sets for four test suites. The model made same mistakes across four test suites

Table 9: Prompt designs for paraphrase detection and natural language inference tasks. The prompt for natural language inference task follows (Liu et al., 2022), where "Implication", "Possibility", and "Contradiction" correspond to "entailment", "neutral", and "contradiction" label.

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. }
- Implication: { Henry has exactly 3 children. }

- { Grace, Thomas and Helen are the only children of Andrea. }
- Implication: { Andrea has exactly 3 children. }

- { Don has 2 dollars. He received 8 more dollars. }
- Implication: { Don now has 10 dollars. }

- { Mary has a cat. She also has a dog. }
- Implication: { Mary has two pets. }
```

Table 10: Capability-wise error rate reductions for $\text{RoBERTa}_{\text{Base}}$ (sentiment analysis), $\text{BERT}_{\text{Base}}$ (paraphrase detection), $\text{RoBERTa}_{\text{Large}}$ (natural language inference) (i.e., the most performant models in three tasks shown in Table 3). Some cells are marked with "/" as we leave template expansion of the NLI task as future work.

	$\text{ERR\%}_{\text{Unpatched}}$	$\mathcal{T}_{\text{TestAug}}$		$\mathcal{T}_{\text{TestAug}} \backslash \mathcal{T}_{\text{Template}}$		$\mathcal{T}_{\text{TestAug}} \backslash \mathcal{T}_{\text{Expansion}}$		$\mathcal{T}_{\text{Template}}$	
		$\text{ERR\%}_{\text{Patched}}$	$\Delta_{\text{ERR\%}}$	$\text{ERR\%}_{\text{Patched}}$	$\Delta_{\text{ERR\%}}$	$\text{ERR\%}_{\text{Patched}}$	$\Delta_{\text{ERR\%}}$	$\text{ERR\%}_{\text{Patched}}$	$\Delta_{\text{ERR\%}}$
Sentiment Analysis									
Negation	30.9	3.2	27.7	4.5	26.4	4.3	26.6	9.6	21.3
SRL	54.9	7.8	47.1	9.8	45.2	9.4	45.5	12.4	42.5
Temporal	34.2	0.0	34.2	0.9	33.3	0.0	34.2	3.9	30.3
Vocabulary	9.9	2.7	7.2	7.2	2.7	6.9	3.0	11.1	-1.2
Paraphrase Detection									
Negation	12.6	0.4	12.3	0.4	12.3	0.4	12.3	0.4	12.3
SRL	37.4	8.2	29.1	17.9	19.4	12.6	24.7	17.4	20.0
Temporal	83.4	0.9	82.5	1.0	82.3	1.8	81.6	6.4	77.0
Vocabulary	15.5	2.5	13.0	3.7	11.8	3.1	12.4	8.1	7.5
Natural Language Inference									
Boolean	43.3	/	/	/	/	13.9	29.4	16.5	26.8
Causal	14.3	/	/	/	/	3.6	10.7	15.2	-0.9
Comparative	40.1	/	/	/	/	25.3	14.8	28.0	12.1
Conditional	65.4	/	/	/	/	16.7	48.7	23.5	41.8
Coreference	17.5	/	/	/	/	9.5	7.9	11.6	5.8
Implicature	51.3	/	/	/	/	24.4	26.9	31.1	20.2
Lexical	18.1	/	/	/	/	5.3	12.8	19.1	-1.0
Negation	7.2	/	/	/	/	0.0	7.2	5.2	2.0
Numerical	30.0	/	/	/	/	15.0	15.0	26.0	4.0
Presupposition	3.8	/	/	/	/	0.0	3.8	3.8	0.0
Quantifier	31.3	/	/	/	/	7.2	24.1	9.2	22.1
Relational	34.0	/	/	/	/	6.9	27.0	11.3	22.6
Spatial	46.2	/	/	/	/	16.9	29.2	20.0	26.2
Syntactic	3.3	/	/	/	/	3.3	0.0	3.3	0.0
Taxonomic	74.3	/	/	/	/	7.6	66.7	12.3	62.0
Temporal	35.4	/	/	/	/	24.0	11.5	27.1	8.3
World	12.1	/	/	/	/	1.6	10.4	4.9	7.1

Table 11: 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 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

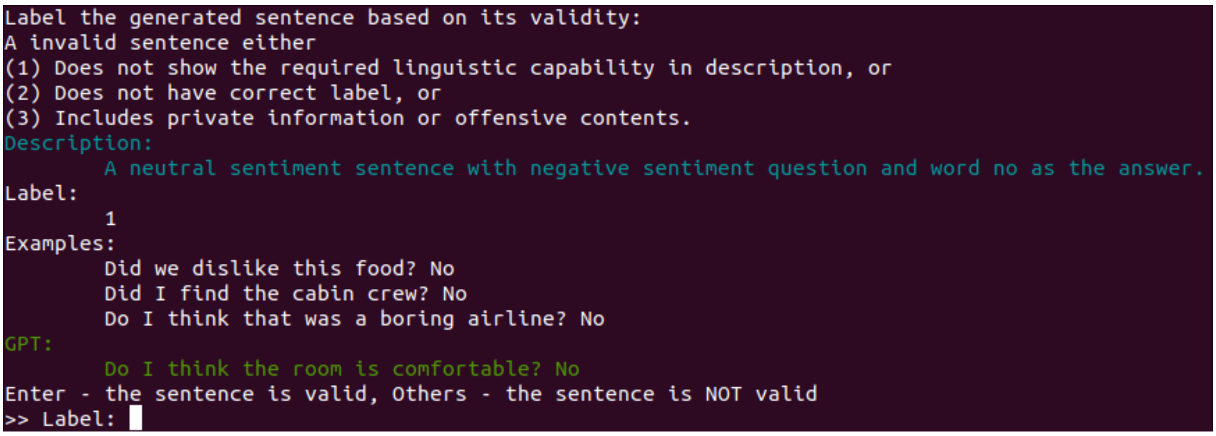

Figure 4: The command line interface for data annotation. Annotators are given a test case description and three examples from the template-based test suite; they are asked to the annotate the validity of the GPT-3-generated sentence (pair). Annotators are reminded of the guidelines for filtering invalid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