Appendix

I. Approach Details

The procedure GET_COV() is shown in Algorithm 1, which contains five parameters: ① The model $model$ refers to the model under test. ② $metrics$ is an array containing the coverage metrics used for the attack. As this paper uses three coverage metrics, so $metrics$ contains these three metrics, i.e., $metrics = \{metric_1, metric_2, metric_3\}$ for Equations of three coverage metrics correspondingly. ③ $text$ represents the current input text that needs to be mutated. ④ $states$ contains the statistics of neuron activation for each coverage metric on the training dataset. Since three coverage metrics are used in this paper, $states = \{state_1, state_2, state_3\}$, i.e., $state_i$ denotes the neuron activation for $metric_i$. ⑤ θ is the threshold for neuron activation. In this procedure, we obtain the neuron activation state (the covered neurons) under the input text by each coverage metric and the coverage value from the neuron activation state (see Lines 3 to 5 in Algorithm 1). After obtaining the set of neurons covered, we take the union[1] of the coverage set of the single input with the total coverage set (neuron activation statistics on the training set before the attack begins) to accumulate all coverage metrics. The cumulative coverage of all metrics is the final coverage used to sort adversarial samples. In this procedure, GET_COVERED() is used to traverse the layers of the model, determining whether the neuron is activated. GET_VALUE() calculates the neuron coverage based on their activation.

Algorithm 1 Compute Coverage for a Given Transformation

Input:
$model$: the language model under test
$metrics$: an array of metrics used to compute neuron coverage
$text$: the input text to be transformed
$states$: an array of neuron activations for different metrics
θ: the threshold for neuron activation
Output:
cov_sum: the sum of coverage values across all three metrics

```
1: procedure GET_COV
2:     $cov\_sum \leftarrow 0$ // coverage sum across all metrics
3:     for $m \leftarrow 1$ to 3 do // iterate through metrics
4:         $state \leftarrow$ GET_COVERED$(model, text, metric_m, \theta)$
               // activation state of the current metric
5:         $cov\_sum \leftarrow cov\_sum +$ GET_VALUE$(states_m \cup state)$
               // accumulate coverage value across three metrics
6:     return $cov\_sum$
```

[1]In the implementation, each state is a dictionary mapping from strings to boolean tensors. The key represents the name of the weight matrix, and the value represents the activation of the neurons. The union operation of two states means taking the union of the value matrices corresponding to the same key. Specifically, if $state_1 = \{k_1 : tensor_1, k_2 : tensor_2\}$, $state_2 = \{k_1 : tensor_3, k_2 : tensor_4\}$, then $state_1 \cup state_2 = \{k_1 : tensor_1 \cup tensor_3, k_2 : tensor_2 \cup tensor_4\}$

II. Experiment Results

A. Original Accuracy

Motivation&Approach. This research question evaluates three representative 7B-parameter language models (Llama-2-7B, Mistral-7B, and Internlm2-7B) to establish baseline accuracy metrics for subsequent adversarial testing, where attack success rates will be measured against these original performance benchmarks. The evaluation covers factual reasoning (AI2_ARC), commonsense reasoning (Winogrande), and specialized knowledge (MMLU) to assess multidimensional competencies.

Results. Table I presents models' original accuracy (before attack) on three datasets. For comparing these three models, Llama-2-7B has the highest error rate, and Internlm2-7B achieves the highest accuracy across all datasets. From the dataset view, MMLU is the most difficult dataset for all language models.

TABLE I: The original accuracy of three models.

Dataset	Model	Error	Accuracy (%)
Ai2_ARC	Llama-2-7B	295	70.50
Ai2_ARC	Mistral-7B	155	84.50
Ai2_ARC	Internlm2-7B	99	90.10
Winogrande	Llama-2-7B	489	51.10
Winogrande	Mistral-7B	367	63.30
Winogrande	Internlm2-7B	166	83.40
MMLU	Llama-2-7B	529	47.10
MMLU	Mistral-7B	408	59.20
MMLU	Internlm2-7B	391	60.90

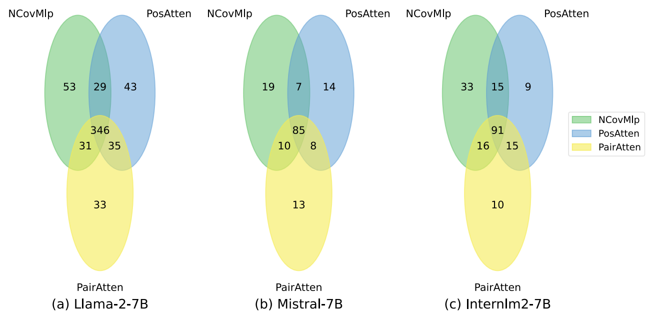
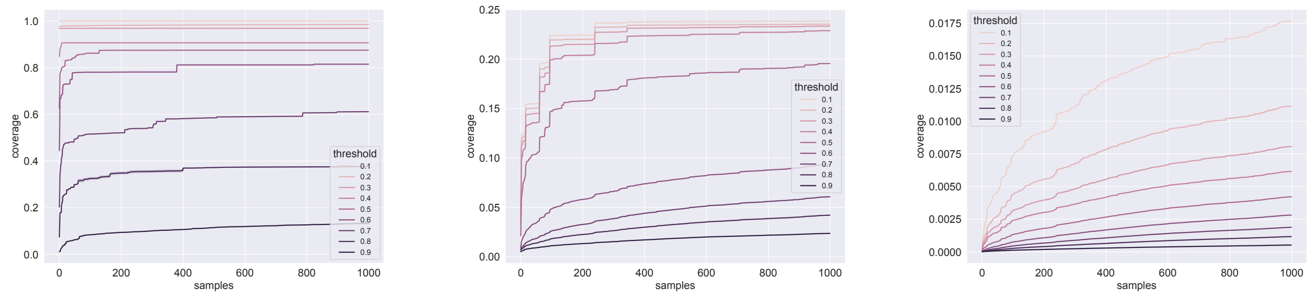


Fig. 1: The overlaps of three coverage metrics.

B. Metric Overlaps

Motivation&Approach. To analyze the effect of each coverage metric in our approach, we study the overlaps between successful samples targeted by these three metrics. In this RQ, we employ a single coverage metric to attack the three studied models in each dataset. We compare the number of success instances and calculate the overlaps of three coverage



(a) NCovMLP (b) NCovPosAtten (c) NCovPairAtten

Fig. 2: The trends of three coverage metrics as the number of input samples increases at different thresholds.

metrics: if different coverage metrics successfully attack the same sample, it is considered *overlapped*.

Results. Figure 1 illustrates the overlaps of successful attack samples on three neuron coverage metrics for the three studied models. From the figure, we observe that (a) for three language models, the majority of attack samples are shared among the three coverage metrics; (b) Each coverage metric contributed around 10% unique successful attack, while every combination of two metrics resulted in an approximately 10% increase in the number of successful attack. We conclude that the proposed adversarial testing approach incorporates three neuron coverage metrics that are *complementary* in different scenarios.

III. Discussion

In this section, we provide a detailed discussion of the results obtained from our method and analyze their implications.

A. Trend of Three Coverage Metrics

In this section, we illustrate the trends of neuron coverage metrics as the number of test samples increases. We randomly selected 1000 data samples to test on the Llama-2-7B model. Figure 2 shows the trends of NCovMLP, NCovPosAtten, and NCovPairAtten with increasing test samples at different thresholds. We increase the threshold from 0.1 to 0.9 with a step size of 0.1.

We have the following observations from Figure 2. (a) NCovMLP is the easiest of the three coverage metrics to achieve. NCovPosAtten is relatively more challenging, and NCovPairAtten has the lowest coverage. (b) As the threshold decreases, the coverage values increase. For NCovMLP, coverage can even reach 100% at low thresholds, such as 0.1. (c) For NCovMLP and NCovPosAtten, the coverage rapidly increases with expanding the number of samples and then tends to stabilize, while NCovPairAtten maintains an increasing trend even at 1000 samples. This indicates that the activation of neurons in the Self-Attention Layer is very sparse (i.e., *high NCovPairAtten coverage is the most difficult to achieve*), suggesting that certain neurons' heads and positions remain inactive across most inputs.

Our coverage-guided method can obtain more diverse samples from the neural network's feature perspective under sparse coverage, thereby improving the success rate of attacks and enhancing the model's robustness.

B. Time Efficiency

We aim to conduct a statistical analysis of the time efficiency of DTCover and baselines. Table II presents the average time (in seconds) required to generate each mutation test sample using different methods across three datasets. From the table, we observe the following:

TABLE II: The time efficiency of generating mutation samples using different methods for the Llama-2-7B model across three datasets.

Method	Ai2_ARC	Winogrande	MMLU
CharacterDeletion	2.14	1.66	1.71
CharacterInsertion	2.14	1.68	1.66
CharacterSwap	2.11	1.69	1.68
QWERTY	2.15	1.67	1.69
FGSM	3.59	2.78	2.73
HotFlip	0.95	0.66	0.80
leap	3.27	1.83	2.89
A2T	1.29	0.91	1.08
checklist	0.20	0.22	0.39
pruthi	2.93	1.78	2.57
DTCover	8.11	6.47	7.28

(a) The fastest method is Checklist, but it performs poorly on our three datasets and three models.

(b) Most white-box methods are slower regarding time efficiency than black-box methods, except for HotFlip.

(c) Our method, due to the need to check the activation state of each neuron and calculate neuron activation rates, is less time-efficient than existing black-box methods. However, we believe it remains within an acceptable range (equivalent to 225.9%-266.7% of FGSM's time).

References