TextTrojaners at CheckThat! 2024: Robustness of Credibility Assessment with Adversarial Examples through BeamAttack*

Notebook for the CheckThat! Lab at CLEF 2024

David Guzman Piedrahita[1,*,†], Arnisa Fazla[1,†] and Lucas Krauter[1,†]

[1]University of Zurich, Rämistrasse 71, CH-8006 Zürich, Switzerland

Abstract

This paper describes the submission of TextTrojaners for the CheckThat! 2024 lab task 6: Robustness of Credibility Assessment with Adversarial Examples [1]. We achieved the second best results on the BODEGA evaluation metric. Our approach BeamAttack is a novel algorithm for generating adversarial examples in natural language processing through the application of beam search. To further improve the search process, we integrate a semantic filter that prioritizes examples with the highest semantic similarity to the original sample, enabling early termination of the search. Additionally, we leverage a model interpretability technique, LIME, to determine the priority of word replacements, along with existing methods such as that determine word importance through the model's logits. Our approach also allows for skipping and removing words, enabling the discovery of minimal modifications that flip the label. Furthermore, we utilize a masked language model to predict contextually plausible alternatives to the words to be replaced, enhancing the coherence of the generated adversarial examples. BeamAttack demonstrates state-of-the-art performance, outperforming existing methods with scores of up to 0.90 on the BiLSTM, 0.84 on BERT, and 0.82 on the RoBERTa classifier [1].

Keywords

Model Robustness, Adversarial Attack, Beam search, Masked Language Models

1. Introduction

Social media platforms increasingly rely on machine learning algorithms for content filtering to identify misleading, harmful, or illegal content. Despite advancements, these models remain vulnerable to adversarial attacks, wherein the input text is manipulated to deceive the model, even after undergoing adversarial training [2]. As adversaries continue to develop new attack techniques, assessing the robustness of text classification approaches becomes imperative.

We participated in "Task 6: Robustness of Credibility Assessment with Adversarial Examples (InCrediblAE)" at the 'CheckThat! lab' during CLEF 2024, which was designed to assess the robustness of popular text classification methods used in credibility assessment tasks [1]. The task involves conducting adversarial attacks on multiple text classification systems trained on various credibility assessment datasets. The goal is to modify the input text in a way that flips the classification of the models while minimizing semantic and syntactic modifications. These attacks are categorized as grey-box attacks, where we have access to the output logits of the victim models, and we can query the victim models as many times as we want.

Previous research on grey-box adversarial attacks on text classification systems has explored modifications at the character [3], word [4, 5, 6], and sentence levels of the input text. These methods encompassed a range of techniques, including random character and word replacements [7, 8, 9, 10, 11, 5], substitution with similar words [12, 13, 6], and controlled sentence paraphrasing to generate semanti-

[1]Code is available at https://github.com/LucK1Y/BeamAttack

CLEF 2024: Conference and Labs of the Evaluation Forum, September 09–12, 2024, Grenoble, France

*Corresponding author.

†These authors contributed equally.

david.guzmanpiedrahita@uzh.ch (D. G. Piedrahita); arnisa.fazla@uzh.ch (A. Fazla); lucassteffen.krauter@uzh.ch (L. Krauter)

cally similar text [14]. Our submission follows the line of research in the field that focuses on sequential word replacement using Masked Language Models (MLMs) combined with a search algorithm.

We introduce BeamAttack, our word-level adversarial attack and submission to the shared task. At its core, BeamAttack employs a beam search algorithm to identify optimal word substitutions, thereby enabling the generation of multiple alternative adversarial text hypotheses capable of flipping the model's classification. Leveraging techniques from existing literature and innovative approaches, BeamAttack integrates a Masked Language Model (MLM), specifically RoBERTa [15], to produce contextually appropriate word substitutions. BeamAttack demonstrated superior performance compared to all baseline methods across all datasets, securing the second place in the competition.

Our contributions include the following:

- We employ a beam search algorithm to explore combinations of multiple word substitutions, optimizing for the output logits of classifier models. This approach significantly expands the search space compared to the greedy search commonly used in the literature and allows for the generation of multiple alternative adversarial samples in a single run. While beam search has been utilized in white-box attacks by Ebrahimi et al. [16], to our knowledge, it has not been applied in a grey-box scenario.
- In line with existing methodologies [6], we generate contextual word substitution alternatives using a Masked Language Model. Although Li et al. [6] utilized a BERT model [17], we employ a RoBERTa model [15] to enhance the effectiveness of our approach.
- We adapt the beam search algorithm to allow for the option of not modifying certain words or removing them instead of substituting them at each step. This modification enables a more flexible and comprehensive exploration of possible word replacement combinations.
- In addition to the logit-based word importance ranking method commonly used in the literature [6], we experiment with identifying vulnerable words in the input text using the explainable AI framework LIME [18], offering a novel perspective on word vulnerability in adversarial text generation.

In the following sections, we will present the details of our approach addressing the challenge of robustness in credibility assessment. Section 2 provides an overview of related work in the field of NLP adversarial attacks, highlighting the fundamental principles of various attack methods and their applications. Section 3 describes the specific task we participate in, namely "Task 6: Robustness of Credibility Assessment with Adversarial Examples (InCrediblAE)", and outlines the datasets and victim models used in our experiments. In Section 4, we present our algorithm and methodology for generating adversarial examples. The results of our experiments are presented in Section 5, where we evaluate the effectiveness of our algorithm in attacking the victim models. In Section 6, we conduct an ablation study to analyze the impact of hyper-parameters on our algorithm's performance. Finally, we conclude our paper in Section 7 and outline limitations and potential avenues for future work in Section 8.

2. Related Work

Despite their powerful capabilities, neural networks suffer from a lack of interpretability, making it difficult to predict their behavior, which can raise security concerns. Szegedy et al. [19] highlighted that while neural networks exhibit impressive generalization capabilities, they are also vulnerable to adversarial examples.

An adversarial sample is generated from the original sample by applying the smallest possible change that leads to a misclassification by the target model, known as the victim classifier. The goal is to alter the sample just enough to trick the classifier into making an incorrect prediction. A formal definition for adversarial samples or examples can be found in [2].

While much of the existing research focuses on adversarial attacks in image classification [20], similar vulnerabilities exist in text classification tasks [21, 22, 4].

2.1. Categories of Adversarial Attacks

Adversarial attacks in NLP can be classified in three categories: white-box attacks, black-box attacks, and grey-box attacks. White-box attacks are also called gradient-based attacks, since the adversary (the attacker) has access to the weights and thus the gradients of the victim model [23, 24, 16, 5].

Black-box attacks are the more realistic scenario, where the adversary only has access to the prediction of the victim for any given input samples. Most algorithms assume an infinite number of queries to find the best perturbation [25, 14], or make use of surrogate models [26].

Przybyła et al. [27] proposes a grey-box scenario, where the adversary has access to not only the predictions, but also to the probabilities given to each class by the victim. This scenario enables adversaries to optimize their attacks based on the output probability distributions [28, 29, 6, 13, 30, 31, 32, 8].

We can also distinguish between targeted and untargeted attacks. In a targeted attack, the attacker aims to mislead the model into a specific incorrect class or label. In an untargeted attack, the attacker aims to mislead the model into any incorrect class or label, without a specific target in mind. The shared task at hand, InCrediblAE, focuses solely on untargeted attacks.

2.2. Granularity of Modifications in Adversarial Attacks

Adversarial attacks in NLP differ fundamentally from those in computer vision due to the symbolic and discrete nature of text versus the continuous nature of images. This discreteness complicates the creation of adversarial examples in NLP, as modifications must preserve the grammatical and semantic integrity of the text while still deceiving the model. Consequently, NLP adversarial attacks often focus on specific granularities of modification, such as characters, words, subwords, or entire sentences.

Character-level methods involve changing individual characters in the adversarial text [5, 16, 8, 3].

Sentence-level methods modify longer passages of a text, including paraphrasing entire sentences in a constrained manner to create adversarial samples [14].

Word-level methods modify the original text by removing, adding, or substituting words, often combining these techniques to create adversarial examples [10, 11]. This process typically involves identifying vulnerable words in the input text and then modifying these selected words to generate adversarial samples [4, 33, 28, 29, 6, 13, 23].

Most of the recent work in adversarial text is based on search algorithms that first rank the words in an input sequence by their importance scores and then search the space of possible replacements for each word according to this pre-computed ranking. This common approach requires a method for calculating word importance scores to determine the optimal order for word replacement, a strategy for selecting suitable word replacements, and a search strategy for iterative word replacement to guide the process of finding the most effective substitutions.

The replacement strategies involve replacing selected words with out-of-vocabulary words [7, 8, 9]; introducing common typos [10, 11, 5]; replacing words with their synonyms or using rule-based systems like dictionaries or WordNet [12]; leveraging embedding spaces to identify semantically similar words [5]; and using masked language models (MLM) to find a word that fits the given context [13, 6].

The methods for calculating the word importance scores involve grey-box methods utilizing the change in the output probabilities of the victim model, when some words are masked or removed [4, 5, 6] and black-box attacks such as calculating the gradient of a surrogate model's output regarding the input words [26]. For example, BERT-Attack [6] iteratively masks each word, measuring the resulting change in prediction, and thereby assessing the degree to which the outcome depends on the deleted word.

One of the most common search strategies is greedy search where each word is replaced sequentially in the order of the word importance [33, 4, 26, 6]. For example, BERT-Attack tries replacing each word with alternatives iteratively until it either manages to flip the classification or moves on to the next word to be replaced. This is a relatively query-efficient strategy, however the search space is severely limited, and it is highly dependent on the performance of the word importance scoring method.

Other search strategies with a larger search space include evolutionary algorithms [28, 29]; and beam

search [16]. For example, HotFlip [16], a white-box adversarial attack for text classification, uses a beam search-based approach that leverages gradient information to guide the search process.

3. Task 6: Robustness of Credibility Assessment with Adversarial Examples (InCrediblAE)

The CheckThat! lab at CLEF 2024 [34] introduced a novel task designed to assess the robustness of popular text classification approaches. The task 6 [1] simulates the content filtering processes commonly employed in social media, spanning multiple domains. For each domain, we were provided with three pre-trained classifiers and a corresponding attack dataset. Our objective was to craft adversarial examples for each sample in the dataset, which would flip the binary classification outcome while preserving the original meaning.

3.1. Dataset Description

The basic statistics of the provided datasets are summarized in Table 1. Each dataset is divided into three subsets: training, development (dev), and attack. The training and dev subsets are used to train the classifier, while the attack subset serves as the evaluation dataset, where the effectiveness of our attack is measured. Each dataset is designed for a binary classification task, and therefore, the positive rate corresponds to the portion of samples belonging to class 1, and is therefore also a measure of class imbalance. Notably, we observed significant variations in the length of samples across the datasets, measured in terms of the number of characters, which are also reported in the table.

Table 1
Datasets used in BODEGA, described by the task, number of instances in training, attack and development subsets, an overall percentage of positive (non-credible) class, and maximum length of samples in characters.

Task	Training	Attack	Dev	Positive	Max Sample Length
RD	8,683	415	2,070	34.24%	3,8350
FC	172,763	405	19,010	51.27%	6,983
HN	60,234	400	3,600	50.00%	142,492
PR2	11,546	407	3,186	71.03%	11,626
C19	1,130	595	-	34.96%	349

Next, we briefly outline the different text domains. All except C19 are already published and described in the CheckThat! lab at CLEF 2023 [27]:

- **Style-based news bias assessment (HN)**: Categorizing news articles as either credible or non-credible based on stylistic cues.
- **Propaganda detection (PR)**: Text passages from news articles that employ propaganda techniques to influence readers.
- **Fact checking (FC)**: Evaluating the accuracy of news articles by considering contextual information from given related Wikipedia snippets.
- **Rumor detection (RD)**: Identifying Twitter threads that disseminate information without a reliable source.
- **COVID-19 misinformation detection (C19)**: Comprises social media messages that convey either factual information or misinformation about the COVID-19 pandemic. The classifier must rely on subtle cues, such as writing styles reminiscent of those found in high-quality news sources (HN), to make its assessments.

3.2. Target Classifier Models

For each domain, the organizers provided three classifiers, each trained in a binary classification setup on the training split of the corresponding dataset.

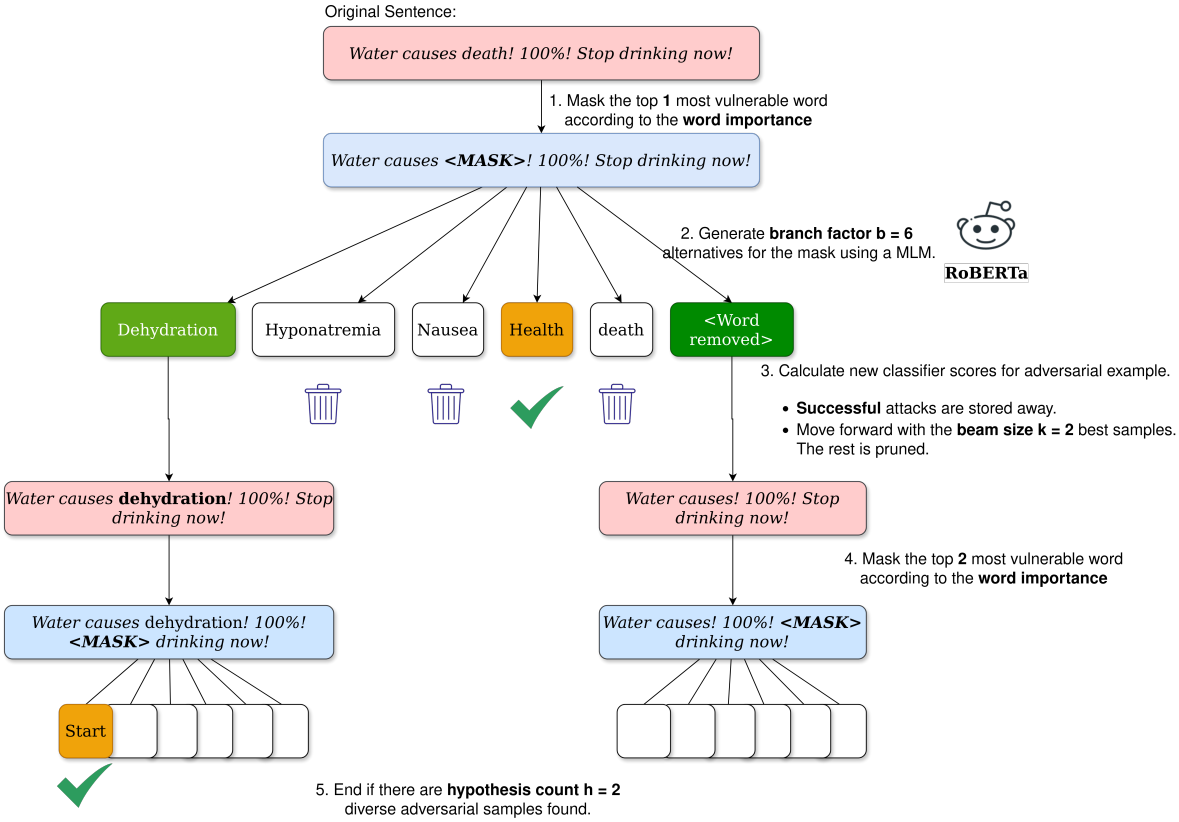

Figure 1: Diagram depicting the series of actions in our BeamAttack algorithm, where beam size k = 2, branching factor b = 6, hypothesis count h = 2.

- Fine-tuned BERT
- Trained BiLSTM,
- Fine-tuned RoBERTa (Surprise classifier): Trained to improve robustness, only available in the test phase.

3.3. Evaluation

The evaluation process is designed to assess the effectiveness of generated adversarial examples in altering the victim's decision while preserving the original meaning. Driven by the formulation of the shared task itself, our approach employs an untargeted attack strategy (see Section 2.1). This means that, for every sample, we aim to alter the victim's decision by flipping it to the opposite of its original prediction.

To quantify the success of an attack, InCrediblAE uses the BODEGA score [27], a comprehensive metric that evaluates the adversarial examples from multiple angles. The BODEGA score is calculated by multiplying three components. The confusion score, which indicates whether the victim classifier's decision has changed and the attack was successful; the semantic score, which measures the similarity between the original and adversarial examples using BLEURT; and the character score, which calculates the Levenshtein distance as a similarity score. All three metrics range in values from 0 to 1.

4. Methodology

Our approach focuses on word-level attacks, a well-regarded technique in recent literature for its balance between edit distance and semantic similarity. We aim to improve upon this by evaluating and

addressing the shortcomings of previous techniques.[1]

We redefine previous word-level attacks, such as HotFlip [16] and BERT-Attack [6], as search problems. In this perspective, the total search space is determined by the number of words to be replaced and the number of valid replacements for each word. This redefinition highlights the potential of search algorithms in optimizing word replacements.

4.1. BeamAttack

BeamAttack combines three components: a method for calculating a ranking of words to determine their replacement order, a word replacement strategy, and the beam search algorithm to explore the search space of possible word replacement combinations. The algorithm overview is given in Figure 1.

4.1.1. Beam Search

Most existing techniques employ greedy search algorithms, selecting the first word(s) that effectively alters the model's prediction. This greedy strategy can lead to sub-optimal adversarial samples, requiring more word changes than necessary.

Our approach adopts beam search as a replacement for greedy search, leveraging its efficacy as a decoding strategy for language models. Beam search explores multiple potential solutions concurrently, structured in a tree-like fashion, where each node represents a partial solution and branches signify possible extensions. At each step, beam search maintains a predefined number of the most promising partial solutions, known as the beam size k. This parameter balances exploration and computational efficiency: larger k values increase the likelihood of finding optimal or near-optimal solutions but elevate computational complexity, while smaller k values reduce demands at the cost of potentially overlooking superior solutions.

In our approach, beam search operates by iteratively expanding the most promising nodes until a final solution is reached. The root node represents the input text, and subsequent nodes denote candidate adversarial texts. At each depth, one word is replaced, with branching occurring when alternative words are considered for replacement. To manage computational complexity, we decouple the number of alternatives evaluated at each depth (branching factor b) from the number of branches retained after pruning. At each depth beyond the first, $k \times b$ nodes are evaluated, with b replacements per alternative from the previous depth and k nodes from the previous depth, since we prune down to the k nodes retained from the previous depth post-pruning.

Selection of promising nodes is based on their effectiveness in reducing the predicted probability of the original class while increasing the probability of the target class. This objective guides branch pruning during the search process. The search continues until cumulative changes are sufficient to alter the model's classification for a designated number of hypotheses. To preserve semantic similarity to the original sample, we introduce the hypothesis count h: the search extends until at least h branches flip the model classification. The final adversarial sample is chosen as the node with the highest semantic similarity to the original sample, as measured by Bleurt scores [35], among the selected hypotheses.

4.1.2. Word Replacement Strategy

For each individual replacement, we enable the option to either keep the word intact or remove it altogether at any depth of the beam search. This added flexibility allows us to preserve the original word if replacements have no impact on changing the probability, thus reducing similarity and edit distances. Conversely, removing the word entirely may be the best strategy to confuse the model in more challenging cases.

To maximize text fluency, we use a masked language model to suggest replacements for any given word. This approach leverages the model's pretraining to ensure that the replacements are likely to be coherent with the rest of the text. We primarily use RoBERTa-Large for this purpose.

[1]The pseudocode for the algorithm can be found in the Appendix 1.

For each depth of beam search, we evaluate the top b (branching factor) highest likelihood word replacements as suggested by the masked language model. This strategy ensures that the potential replacements not only fit well within the context of the sentence but also maintain the overall semantic coherence of the text.

4.1.3. Word Importance Scoring Method

To determine the order in which words should be replaced, we use two different techniques to identify the most important words for the model's decision: LIME [18] and logit-based importance scores, as proposed by Li et al. [6].

LIME is an explainability algorithm that provides local explanations for model predictions. It works by perturbing the input data and observing the changes in the output. Specifically, for each word in the input sentence, LIME generates a set of perturbed versions of the sentence with that word replaced by various alternatives. It then fits a simple, interpretable model to these perturbed instances to estimate the importance of each word based on how much the model's output changes.

Logit-based importance scores, on the other hand, measure the influence of each word on the model's output logits directly. Let $S = [w_0, \ldots, w_i, \ldots]$ denote the input sentence, and $o_y(S)$ represent the logit output by the target model for the correct label y. The importance score I_{w_i} for each word w_i is defined as the difference in logits when the word is masked:

$$I_{w_i} = o_y(S) - o_y(S \setminus w_i),$$

where $S \setminus w_i$ is the sentence with w_i replaced by a [MASK] token. This score quantifies how much the presence of w_i contributes to the model's prediction.

Our approach is specifically designed to maximize the risk of incorrect model predictions by exploiting the vulnerabilities in the model's decision-making process. We achieve this by identifying the most susceptible words in the input text and strategically replacing them with semantically consistent alternatives. This targeted approach enables us to effectively manipulate the model's decision while preserving the overall coherence of the text. Moreover, by focusing on the most critical words, we increase the likelihood of achieving our goal with a minimal number of word replacements, thereby maintaining the semantic integrity of the original text.

4.2. Hyper-parameters

For each dataset and victim model combination, we determined a different set of hyperparameters for our BeamAttack approach. To achieve this, we employed an informed random search on subsets of 10-50 samples for each task and victim model. Our search strategy was guided by intuition and initial observations, allowing us to effectively fine-tune the hyperparameters. We considered the following ranges for the hyperparameter values:

1. **Beam size** k: We tested values between 10 to 100, increasing in steps of 5 (i.e., 10, 15, 20, ...). We increased the beam size until the improvement in the BODEGA score became negligible (less than 1 point). Figure 2 illustrates how this approach led to convergence.
2. **Branching factor** b: We tested a range of values similar to the chosen k values, within a range of -20 to +20 around the selected k values.
3. **Hypothesis count** h: Based on initial experiments, we decided to test hypothesis counts of 5, 10, and 20. This decision was made considering the difficulty of the task at hand and the BODEGA scores obtained during the hyper-parameter search. We observed that further increasing h did not significantly improve the scores, as explained in Section 6.4.
4. **Word importance scoring method**: We compared the results of both word importance scoring methods, LIME and logit-based, in the initial experiments of each dataset and victim model combination.

We report the selected parameters for each scenario in Table 2.

As mentioned, we employed RoBERTa-large[2] to replace the masks, with one exception: for the Covid-19 (C19) task on the BERT classifier, we utilized the "vinai/bertweet-large"[3] model instead. This decision was motivated by the intuition that, given the Twitter domain of the COVID-19 dataset, this model might perform better.

Table 2
Hyper-parameters for each combination of victim model and dataset.

Victim	Task	beam size k	hypothesis count h	branching factor b	importances
BiLSTM	Pr2	25	30	50	logit-based
BiLSTM	FC	10	10	30	logit-based
BiLSTM	RD	10	10	30	logit-based
BiLSTM	HN	20	10	20	logit-based
BiLSTM	C19	15	10	30	logit-based
BERT	PR2	60	10	80	logit-based
BERT	FC	10	10	30	LIME
BERT	RD	10	10	10	logit-based
BERT	HN	20	5	20	LIME
BERT	C19	10	10	30	logit-based
RoBERTa	PR2	60	10	80	logit-based
RoBERTa	FC	60	10	80	logit-based
RoBERTa	RD	20	20	40	logit-based
RoBERTa	HN	10	10	10	LIME
RoBERTa	C19	40	10	60	logit-based

4.3. Computational Resources

This chapter outlines the computational resources utilized and the hyper-parameter tuning strategies employed to optimize our BeamAttack approach, which demanded substantial computational power. To overcome the limitations imposed by computational and time constraints, we resorted to a random search strategy to determine the optimal parameter set for each scenario, recognizing that this approach may not necessarily result in the identification of the globally optimal combination. We primarily used Kaggle's free GPU infrastructure, which offers NVIDIA Tesla P100s and T4x [4]. We report the running time statistics for the final runs of each model and dataset combination in Table 3.

5. Results

The outcomes of this shared task and a comparison with those reported by Przybyła et al. [27] are presented in Tables 4, 5, and 6. Notably, we do not report scores for the Covid (C19) classification subtask besides our own, as no official published scores are available. The same thing applies to the RoBERTa victim in Table 6.

Our algorithm outperforms all other algorithms on BERT and BiLSTM victims, demonstrating the superior effectiveness of BeamAttack. While other algorithms may generate adversarial samples with better character or semantic scores in specific tasks, they fail to achieve this consistently across the entire dataset. Moreover, our confusion scores are always superior to those of other techniques.

Nevertheless, our algorithm needs many victim queries to achieve this goal. For example, in the task RD on the BERT classifier (see Table 4), our BeamAttack takes roughly double the number of queries

[2]FacebookAI/roberta-large model from Hugging Face: https://huggingface.co/FacebookAI/roberta-large
[3]vinai/bertweet-large model from Hugging Face: https://huggingface.co/vinai/bertweet-large
[4]For the submission phase, we used an NVIDIA Tesla T4 GPU.

Table 3
Running time statistics (in seconds) for each combination of victim model and dataset, measured on the entire dataset.

Victim	Task	Time per Example (s)	Total Attack Time (s)	Total Evaluation Time (s)
BiLSTM	Pr2	1.513	629.279	15.843
BiLSTM	FC	2.111	854.859	54.206
BiLSTM	RD	13.248	5497.992	560.019
BiLSTM	HN	9.578	3831.309	615.956
BiLSTM	C19	2.26	1344.562	95.838
BERT	PR2	12.586	5235.587	15.986
BERT	FC	7.988	3235.195	51.033
BERT	RD	70.347	29193.831	488.945
BERT	HN	86.049	34419.532	660.498
BERT	C19	12.453	7409.627	103.194
RoBERTa	PR2	72.192	30031.747	20.433
RoBERTa	FC	3.467	1403.967	44.0466
RoBERTa	RD	72.192	30031.747	20.433
RoBERTa	HN	144.188	57675.011	544.444
RoBERTa	C19	54.731	32564.714	117.173

compared to the Genetic algorithm. This highlights a trade-off between the quality of the generated adversarial samples and the computational resources required to achieve them.

In summary, our BeamAttack algorithm has shown exceptional capability in producing high-quality adversarial samples that are highly effective in deceiving victims, thereby surpassing the performance of other algorithms when tested on BERT and BiLSTM victims.

6. Ablation Experiments

We conducted a small-scale ablation study to investigate the impact of various hyper-parameters on the performance of our BeamAttack algorithm, shedding some light on the relationship between these parameters and the algorithm's effectiveness. Specifically, we investigate the impact of logit-based versus LIME, beam size k, branching factor b and the hypothesis count h on the algorithm's performance, exploring each of these parameters in the context of specific scenarios of task and victim model. The full results of the ablation studies can be found in the Appendix B.

6.1. Word Importance

Besides the logit-based approach from BERT-Attack [6], we utilized LIME, a state-of-the-art interpretability framework, to identify crucial words in a sample that influence the victim's decision-making process. By modifying these words, we increased the victim's susceptibility to our adversarial attack.

We report some comparisons of LIME in the appendix in Table 7. From this table, we can gather multiple insights. For instance, for the BERT victim the choice of importance method depends on the task dataset, with RD favoring the logit-based method and FC preferring LIME. For the BiLSTM, the logit-based method is superior on all datasets. For RoBERTa, which is architecturally similar to BERT, the choice of importance method also depends on the dataset, with FC favoring LIME.

Furthermore, we observed that for the RoBERTa victim and C19 task, the choice between LIME and logit-based depends on other hyper-parameters such as beam size, branching factors, and hypothesis count. More specifically, our results suggest transformer-based classifiers, namely classifiers BERT and RoBERTa are generally more vulnerable to LIME over logit-based for the HN and FC datasets. Both these datasets classify whether a statement contains facts, with HN using subtle clues in writing styles and FC using external knowledge from Wikipedia. We leave the investigation behind this observation

Table 4
Results of adversarial attacks on the **BERT** classifier in five misinformation detection tasks in an untargeted scenario, including previously reported results by Przybyła et al. [27] and our findings. Evaluation measures include BODEGA score (B.), confusion score (con), semantic score (sem), character score (char) and number of queries to the attacked model (Q.). The best score in each task is in boldface.

Task	Method	B.	con	sem	char	Q.
HN	BAE	0.34	0.60	0.58	0.96	606.83
	BERT-ATTACK	0.60	0.96	0.64	0.97	648.41
	DeepWordBug	0.22	0.29	0.78	**1.00**	395.94
	Genetic	0.40	0.86	0.47	0.98	2713.80
	SememePSO	0.16	0.34	0.50	0.99	341.70
	PWWS	0.38	0.82	0.47	0.98	2070.78
	SCPN	0.00	0.92	0.08	0.02	11.84
	TextFooler	0.39	0.92	0.44	0.94	660.52
	BeamAttack (Ours)	**0.84**	**1.00**	**0.86**	0.97	4327.67
PR	BAE	0.11	0.18	0.69	0.94	33.96
	BERT-ATTACK	0.43	0.70	0.68	0.90	80.16
	DeepWordBug	0.28	0.36	**0.79**	**0.96**	27.43
	Genetic	0.50	0.84	0.65	0.89	962.40
	SememePSO	0.41	0.68	0.66	0.90	96.17
	PWWS	0.47	0.75	0.68	0.91	131.92
	SCPN	0.09	0.47	0.36	0.46	11.47
	TextFooler	0.43	0.77	0.64	0.87	57.94
	BeamAttack (Ours)	**0.69**	**0.98**	0.77	0.89	1373.41
FC	BAE	0.34	0.51	0.70	0.96	80.69
	BERT-ATTACK	0.53	0.77	0.73	0.95	146.73
	DeepWordBug	0.44	0.53	**0.84**	**0.98**	54.32
	Genetic	0.52	0.79	0.70	0.95	1215.19
	SememePSO	0.44	0.64	0.71	0.96	148.20
	PWWS	0.48	0.69	0.72	0.96	225.27
	SCPN	0.09	0.90	0.29	0.31	11.90
	TextFooler	0.46	0.70	0.70	0.93	106.13
	BeamAttack (Ours)	**0.79**	**1.00**	0.83	0.96	1390.83
RD	BAE	0.07	0.18	0.41	0.98	313.01
	BERT-ATTACK	0.18	0.44	0.43	0.96	774.31
	DeepWordBug	0.16	0.23	0.70	**0.99**	232.74
	Genetic	0.20	0.46	0.45	0.96	4425.11
	SememePSO	0.10	0.21	0.46	0.97	345.89
	PWWS	0.16	0.38	0.45	0.95	1105.99
	SCPN	0.01	0.38	0.16	0.10	11.35
	TextFooler	0.16	0.41	0.43	0.91	657.15
	BeamAttack (Ours)	**0.59**	**0.80**	**0.80**	0.91	10618.93
C19	BeamAttack (Ours)	0.71	0.98	0.78	0.92	2628.90

to future work.

Our results suggest that exploring explainable AI frameworks for adversarial attacks is a promising direction of research. However, our experiments show that LIME only improved upon the logit-based approach on a few datasets. Additionally, LIME requires more queries, and when the improvement was marginal, we opted for the logit-based approach due to resource constraints.

Table 5
Results of adversarial attacks on the **BiLSTM** classifier in five misinformation detection tasks in an untargeted scenario, including previously reported results by Przybyła et al. [27] and our findings. Evaluation measures include BODEGA score (B.), confusion score (con), semantic score (sem), character score (char) and number of queries to the attacked model (Q.). The best score in each task is in boldface.

Task	Method	B.	Con	Sem	Char	Q.
HN	BAE	0.48	0.77	0.64	0.98	489.27
	BERT-ATTACK	0.64	0.98	0.66	0.99	487.85
	DeepWordBug	0.41	0.53	0.77	**1.00**	396.18
	Genetic	0.44	0.94	0.48	0.98	2029.31
	SememePSO	0.21	0.42	0.50	0.99	313.51
	PWWS	0.44	0.93	0.48	0.99	2044.96
	SCPN	0.00	0.94	0.08	0.02	11.86
	TextFooler	0.43	0.94	0.47	0.97	543.68
	BeamAttack (Ours)	**0.90**	**1.00**	**0.91**	0.99	936.96
PR	BAE	0.15	0.23	0.72	0.94	32.94
	BERT-ATTACK	0.53	0.80	0.72	**0.91**	61.41
	DeepWordBug	0.29	0.38	0.79	0.96	27.45
	Genetic	0.54	0.88	0.67	0.89	782.15
	SememePSO	0.47	0.76	0.68	0.89	85.34
	PWWS	0.53	0.84	0.69	0.90	130.85
	SCPN	0.12	0.55	0.39	0.50	11.55
	TextFooler	0.51	0.85	0.67	0.88	52.59
	BeamAttack (Ours)	**0.70**	**0.97**	**0.80**	0.90	593.38
FC	BAE	0.36	0.55	0.69	0.96	77.76
	BERT-ATTACK	0.60	0.86	0.73	0.95	132.80
	DeepWordBug	0.48	0.58	**0.85**	**0.98**	54.36
	Genetic	0.61	0.90	0.71	0.95	840.99
	SememePSO	0.53	0.76	0.72	0.96	112.84
	PWWS	0.57	0.82	0.73	0.96	221.60
	SCPN	0.08	0.75	0.29	0.32	11.75
	TextFooler	0.55	0.82	0.71	0.94	98.31
	BeamAttack (Ours)	**0.76**	**1.00**	0.81	0.94	1549.12
RD	BAE	0.09	0.21	0.43	0.98	312.77
	BERT-ATTACK	0.29	0.79	0.41	0.89	985.52
	DeepWordBug	0.16	0.24	0.68	**0.99**	232.75
	Genetic	0.32	0.71	0.47	0.96	3150.24
	SememePSO	0.15	0.31	0.48	0.97	314.63
	PWWS	0.29	0.64	0.47	0.97	1059.07
	SCPN	0.01	0.55	0.17	0.09	11.53
	TextFooler	0.24	0.64	0.41	0.87	639.97
	BeamAttack (Ours)	**0.83**	**1.00**	**0.87**	0.96	3831.72
C19	BeamAttack (Ours)	**0.72**	**0.99**	**0.78**	**0.92**	837.35

6.2. Beam Size k

We also conducted an experiment to investigate the effect of the beam size k on our algorithm's performance. Specifically, we experimented on a subset of 30 samples from the **PR2** dataset using the **RoBERTa** victim model. The results are presented in Figure 2, with the exact scores provided in Appendix Table 8. Our analysis reveals a consistent improvement in all scores as the beam size increases.

However, we encountered a significant limitation. As shown in the table, a beam size of 100 requires 16,500 queries per sample, which is impractical for limited GPU setup. Therefore, we sought an optimal

Table 6
Results of our adversarial attack on the **RoBERTa** classifier in five misinformation detection tasks in an untargeted scenario. Evaluation measures include BODEGA score (B.), confusion score (con), semantic score (sem), character score (char) and number of queries to the attacked model (Q.).

Task	Method	B.	Con	Sem	Char	Q.
HN	BeamAttack (Ours)	0.67	1.00	0.72	0.91	4596.62
PR	BeamAttack (Ours)	0.45	0.97	0.55	0.79	10286.82
FC	BeamAttack (Ours)	0.82	1.00	0.843	0.97	498.93
RD	BeamAttack (Ours)	0.54	0.87	0.69	0.83	15458.12
C19	BeamAttack (Ours)	0.66	1.00	0.72	0.91	6491.39

trade-off where the scores would reach a plateau. In the case of PR2 and RoBERTa, we determined that a beam size of 60 strikes a reasonable balance between performance and computational feasibility.

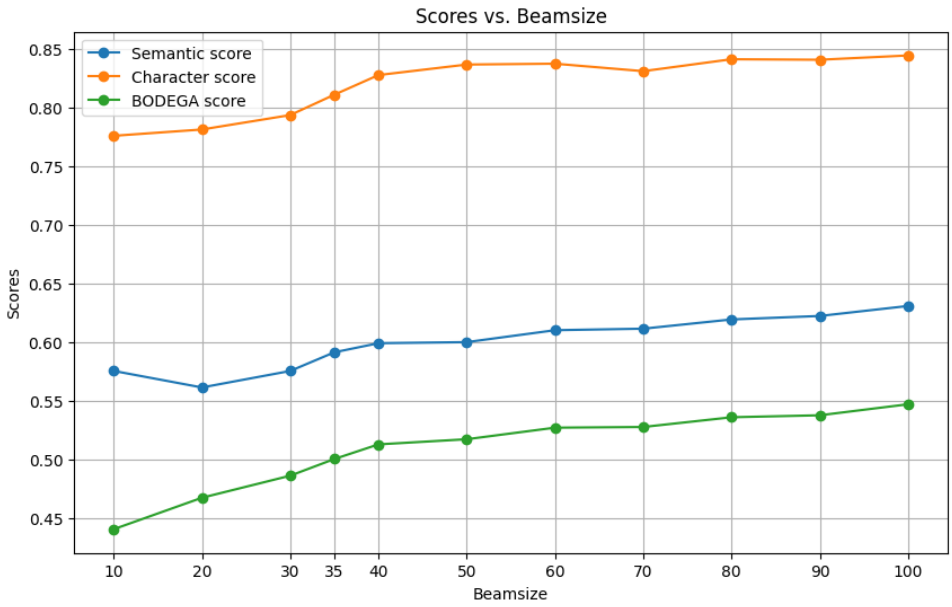

Figure 2: Comparison of the scores across different settings for the beam size

6.3. branching factor b

We further explored the impact of the branching factor b on our algorithm's performance, using a fixed beam size of 40. We conducted an experiment on a subset of 30 samples from the **PR2** dataset, employing the **RoBERTa** victim model. The results are illustrated in Figure 3, which displays the scores against varying branching factors. The exact scores can be found in Appendix Table 9. Our analysis reveals that increasing the branching factor b leads to a slight improvement in the BODEGA score.

6.4. Hypothesis Count h

Our algorithm's flexibility in replacing, skipping, or removing words, combined with the use of a masked language model, ensures that the resulting sentence remains semantically correct. The beam search approach enables us to efficiently query for multiple adversarial samples and use the one that is most semantically close to our original sentence.

We conducted an ablation study on the hypothesis count h, which defines how many adversarial samples to use before selecting the closest. We explored different scenarios, incrementing the parameter from 10 to 20, but observed only minimal effects, resulting in a negligible improvement or deterioration

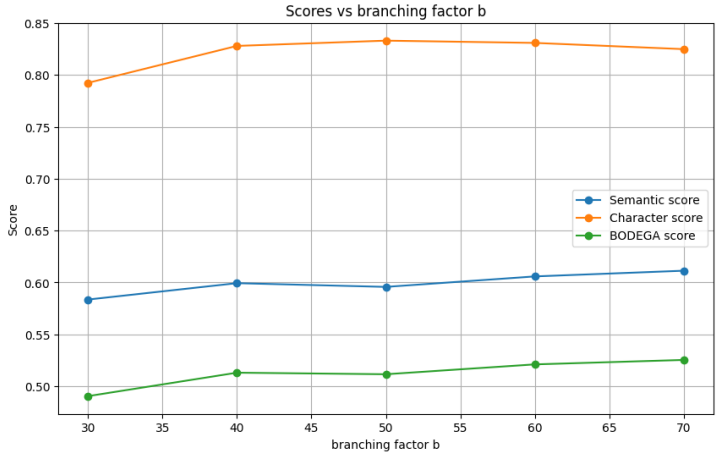

Figure 3: Comparison of the scores across different settings for the branching factor

of less than 1% in the BODEGA score. We provide a small-scale report of our findings for the **RoBERTa** victim and various tasks in the appendix (Table 10). Notably, in some cases, it came at the cost of requiring more queries. Therefore, we opted to maintain a constant value of 10 for the hypothesis count in most cases.

We acknowledge that a more comprehensive investigation of this parameter could have potentially yielded further improvements.

6.5. LIME Parameters

The LIME explainability method itself presents several hyper-parameters that can be optimized. Although tuning these parameters was not our primary focus, we recognized their potential impact on our approach. Due to time constraints, we initially chose to keep the parameters fixed, but we later realized that adjusting them could have a significant effect on our algorithm's performance. The complete documentation of parameters can be found here[5].

Specifically, we found that by reducing the number of samples used by LIME, we could substantially decrease the number of victim model queries required without compromising our algorithm's performance. Notably, by setting the number of samples to 500, a significant reduction of 4500 queries per explanation was achieved, compared to LIME's default setting of 5000. This adjustment may come at the cost of reduced performance for longer samples, highlighting the need for further exploration of these parameters in future work.

In future studies, researchers could explore matching the explainability model better to the victim model. This could involve trying out different kernel sizes or entirely different kernels. We think this adjustment could improve the efficency of generating an explanation.

7. Conclusion

In this shared task, we introduced BeamAttack, an innovative adversarial text generation method that combines multiple techniques from the literature with novel approaches. At the core of BeamAttack lies the beam search algorithm, which serves as a fundamental tool in optimizing adversarial text generation.

Our method leverages a Masked Language Model (MLM), specifically RoBERTa, to produce contextually appropriate word substitutions. Notably, RoBERTa's ability to generate alternative words with typos and controversial connotations significantly enhances BeamAttack's effectiveness.

[5]Code Documentation of LimeTextExplainer. https://lime-ml.readthedocs.io/en/latest/lime.html#lime.lime_text.LimeTextExplainer

Additionally, we integrate LIME, an interpretability method, to calculate the contribution of each word to the model's classification. Our experiments demonstrate significant improvements in attack effectiveness, particularly in specific task contexts.

The key feature of BeamAttack is its utilization of beam search, which enables the identification of optimal word-level modifications. Unlike previous methods relying on simplistic greedy search approaches, BeamAttack's beam search strategy allows for more nuanced modification decisions. Moreover, by incorporating the capability to skip or remove words, inspired by prior work, we enhance the algorithm's flexibility in exploring potential solutions.

Furthermore, BeamAttack's beam search produces multiple alternative adversarial text hypotheses capable of flipping the model's classification. By selecting the hypothesis with the highest semantic similarity to the input text, our method effectively filters out dissimilar or contradictory hypotheses.

In summary, BeamAttack's efficacy is significantly attributed to its beam search component, which plays a crucial role in optimizing adversarial text generation. The flexibility provided by tunable hyper-parameters allows users to prioritize semantic or character similarity based on their specific requirements.

8. Limitations

Our word replacement strategy in BeamAttack is currently limited to methods of substituting and removing words. Removing a word can sometimes result in syntactically incorrect and nonsensical sentences, especially since RoBERTa is not utilized for this operation. This decreases the coherence of the context after the change. To enhance the quality of our outputs, we recommend exploring the integration of word insertion techniques into the beam search process. This approach could potentially improve the syntactic and semantic integrity of the sentences.

The BeamAttack approach strives to maintain semantic similarity to the original sample via semantic filtering. However, there could be a benefit to scoring the beams at each step of the beam search, rather than solely relying on selecting the hypothesis with the highest semantic score.

Moreover, while RoBERTa provides context-based alternatives, these alternatives solely depend on the context rather than the word to be replaced itself. This limitation may impact the performance of the beam search, potentially resulting in alternatives that are opposite in meaning to the word to be replaced. To address this, potential solutions include prompting Language Models (LLMs) to produce word candidates at each step of the beam search or filtering out produced alternatives based on similarity to the word to be replaced. However, these approaches may introduce trade-offs and require careful consideration to maintain performance.

In our experiments, we limited ourselves to the use of the Explainable AI method LIME. However, we suggest that future research should explore the potential of diverse Explainable AI methods to uncover vulnerabilities in the victim's decision-making process. Our experience with LIME has demonstrated the value of these approaches, but we believe that Explainable AI frameworks could provide even more insightful explanations. By leveraging these techniques, researchers may be able to identify more effective strategies for generating adversarial examples, ultimately leading to more robust and reliable machine learning models.

Acknowledgments

We would like to express our gratitude to the Department of Computational Linguistics at the University of Zurich (UZH) for the valuable advice and guidance they offered throughout our research. Additionally, we appreciate their efforts in providing us with access to a GPU during the evaluation phase of our project. For more information about the department, please visit their website at https://www.cl.uzh.ch/en.html.

References

[1] P. Przybyła, B. Wu, A. Shvets, Y. Mu, K. C. Sheang, X. Song, H. Saggion, Overview of the CLEF-2024 CheckThat! lab task 6 on robustness of credibility assessment with adversarial examples (incrediblae), in: G. Faggioli, N. Ferro, P. Galuščáková, A. García Seco de Herrera (Eds.), Working Notes of CLEF 2024 - Conference and Labs of the Evaluation Forum, CLEF 2024, Grenoble, France, 2024.

[2] W. E. Zhang, Q. Z. Sheng, A. Alhazmi, C. Li, Adversarial attacks on deep-learning models in natural language processing: A survey, ACM Trans. Intell. Syst. Technol. 11 (2020). URL: https://doi.org/10.1145/3374217. doi:10.1145/3374217.

[3] B. Formento, C. S. Foo, L. A. Tuan, S. K. Ng, Using punctuation as an adversarial attack on deep learning-based NLP systems: An empirical study, in: A. Vlachos, I. Augenstein (Eds.), Findings of the Association for Computational Linguistics: EACL 2023, Association for Computational Linguistics, Dubrovnik, Croatia, 2023, pp. 1–34. URL: https://aclanthology.org/2023.findings-eacl.1. doi:10.18653/v1/2023.findings-eacl.1.

[4] D. Jin, Z. Jin, J. T. Zhou, P. Szolovits, Is bert really robust? natural language attack on text classification and entailment, 2019. URL: https://api.semanticscholar.org/CorpusID:198967870.

[5] J. Li, S. Ji, T. Du, B. Li, T. Wang, Textbugger: Generating adversarial text against real-world applications, ArXiv abs/1812.05271 (2018). URL: https://api.semanticscholar.org/CorpusID:54815878.

[6] L. Li, R. Ma, Q. Guo, X. Xue, X. Qiu, BERT-ATTACK: Adversarial attack against BERT using BERT, in: B. Webber, T. Cohn, Y. He, Y. Liu (Eds.), Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, Online, 2020, pp. 6193–6202. URL: https://aclanthology.org/2020.emnlp-main.500. doi:10.18653/v1/2020.emnlp-main.500.

[7] Y. Belinkov, Y. Bisk, Synthetic and natural noise both break neural machine translation, ArXiv abs/1711.02173 (2017). URL: https://api.semanticscholar.org/CorpusID:3513372.

[8] J. Gao, J. Lanchantin, M. L. Soffa, Y. Qi, Black-box generation of adversarial text sequences to evade deep learning classifiers, in: 2018 IEEE Security and Privacy Workshops (SPW), 2018, pp. 50–56. doi:10.1109/SPW.2018.00016.

[9] H. Hosseini, S. Kannan, B. Zhang, R. Poovendran, Deceiving google's perspective api built for detecting toxic comments, ArXiv abs/1702.08138 (2017). URL: https://api.semanticscholar.org/CorpusID:15418780.

[10] B. Liang, H. Li, M. Su, P. Bian, X. Li, W. Shi, Deep text classification can be fooled, in: Proceedings of the 27th International Joint Conference on Artificial Intelligence, IJCAI'18, AAAI Press, 2018, p. 4208–4215.

[11] S. Samanta, S. Mehta, Towards crafting text adversarial samples (2017).

[12] G. A. Miller, WordNet: A lexical database for English, in: Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994, 1994. URL: https://aclanthology.org/H94-1111.

[13] S. Garg, G. Ramakrishnan, Bae: Bert-based adversarial examples for text classification, ArXiv abs/2004.01970 (2020). URL: https://api.semanticscholar.org/CorpusID:214802269.

[14] M. T. Ribeiro, S. Singh, C. Guestrin, Semantically equivalent adversarial rules for debugging NLP models, in: I. Gurevych, Y. Miyao (Eds.), Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Melbourne, Australia, 2018, pp. 856–865. URL: https://aclanthology.org/P18-1079. doi:10.18653/v1/P18-1079.

[15] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, Roberta: A robustly optimized bert pretraining approach, 2019. arXiv:1907.11692.

[16] J. Ebrahimi, A. Rao, D. Lowd, D. Dou, HotFlip: White-box adversarial examples for text classification, in: I. Gurevych, Y. Miyao (Eds.), Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), Association for Computational Linguistics, Melbourne, Australia, 2018, pp. 31–36. URL: https://aclanthology.org/P18-2006.

doi:10.18653/v1/P18-2006.
[17] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. arXiv:1810.04805.
[18] M. T. Ribeiro, S. Singh, C. Guestrin, "why should I trust you?": Explaining the predictions of any classifier, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016, 2016, pp. 1135–1144.
[19] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, R. Fergus, Intriguing properties of neural networks, 2014. arXiv:1312.6199.
[20] D. Hendrycks, K. Zhao, S. Basart, J. Steinhardt, D. Song, Natural adversarial examples, 2021. arXiv:1907.07174.
[21] T. McCoy, E. Pavlick, T. Linzen, Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference, in: A. Korhonen, D. Traum, L. Màrquez (Eds.), Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Florence, Italy, 2019, pp. 3428–3448. URL: https://aclanthology.org/P19-1334. doi:10.18653/v1/P19-1334.
[22] R. Jia, P. Liang, Adversarial examples for evaluating reading comprehension systems, in: M. Palmer, R. Hwa, S. Riedel (Eds.), Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Copenhagen, Denmark, 2017, pp. 2021–2031. URL: https://aclanthology.org/D17-1215. doi:10.18653/v1/D17-1215.
[23] N. Papernot, P. McDaniel, A. Swami, R. Harang, Crafting adversarial input sequences for recurrent neural networks, in: MILCOM 2016 - 2016 IEEE Military Communications Conference, IEEE Press, 2016, p. 49–54. URL: https://doi.org/10.1109/MILCOM.2016.7795300. doi:10.1109/MILCOM.2016.7795300.
[24] P. Gaiński, K. Bałazy, Step by step loss goes very far: Multi-step quantization for adversarial text attacks, in: A. Vlachos, I. Augenstein (Eds.), Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics, Association for Computational Linguistics, Dubrovnik, Croatia, 2023, pp. 2038–2048. URL: https://aclanthology.org/2023.eacl-main.149. doi:10.18653/v1/2023.eacl-main.149.
[25] Z. Yu, X. Wang, W. Che, K. He, TextHacker: Learning based hybrid local search algorithm for text hard-label adversarial attack, in: Y. Goldberg, Z. Kozareva, Y. Zhang (Eds.), Findings of the Association for Computational Linguistics: EMNLP 2022, Association for Computational Linguistics, Abu Dhabi, United Arab Emirates, 2022, pp. 622–637. URL: https://aclanthology.org/2022.findings-emnlp.44. doi:10.18653/v1/2022.findings-emnlp.44.
[26] R. Maheshwary, S. Maheshwary, V. Pudi, A strong baseline for query efficient attacks in a black box setting, ArXiv abs/2109.04775 (2021). URL: https://api.semanticscholar.org/CorpusID:237485150.
[27] P. Przybyła, A. Shvets, H. Saggion, Verifying the robustness of automatic credibility assessment, 2023. arXiv:2303.08032.
[28] M. Alzantot, Y. Sharma, A. Elgohary, B.-J. Ho, M. Srivastava, K.-W. Chang, Generating natural language adversarial examples, in: E. Riloff, D. Chiang, J. Hockenmaier, J. Tsujii (Eds.), Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Brussels, Belgium, 2018, pp. 2890–2896. URL: https://aclanthology.org/D18-1316. doi:10.18653/v1/D18-1316.
[29] Y. Zang, F. Qi, C. Yang, Z. Liu, M. Zhang, Q. Liu, M. Sun, Word-level textual adversarial attacking as combinatorial optimization, in: D. Jurafsky, J. Chai, N. Schluter, J. Tetreault (Eds.), Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Online, 2020, pp. 6066–6080. URL: https://aclanthology.org/2020.acl-main.540. doi:10.18653/v1/2020.acl-main.540.
[30] X. Yang, Y. Gong, W. Liu, J. Bailey, D. Tao, W. Liu, Semantic-preserving adversarial text attacks, IEEE Transactions on Sustainable Computing 8 (2023) 583–595. doi:10.1109/TSUSC.2023.3263510.
[31] M. Ni, Z. Sun, W. Liu, Fraud's bargain attack: Generating adversarial text samples via word manipulation process, IEEE Transactions on Knowledge and Data Engineering (2024) 1–14. doi:10.1109/TKDE.2024.3349708.

[32] D. Lee, S. Moon, J. Lee, H. O. Song, Query-efficient and scalable black-box adversarial attacks on discrete sequential data via bayesian optimization, 2022. arXiv:2206.08575.

[33] S. Ren, Y. Deng, K. He, W. Che, Generating natural language adversarial examples through probability weighted word saliency, in: A. Korhonen, D. Traum, L. Màrquez (Eds.), Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Florence, Italy, 2019, pp. 1085–1097. URL: https://aclanthology.org/P19-1103. doi:10.18653/v1/P19-1103.

[34] A. Barrón-Cedeño, F. Alam, T. Chakraborty, T. Elsayed, P. Nakov, P. Przybyła, J. M. Struß, F. Haouari, M. Hasanain, F. Ruggeri, X. Song, R. Suwaileh, The clef-2024 checkthat! lab: Check-worthiness, subjectivity, persuasion, roles, authorities, and adversarial robustness, in: N. Goharian, N. Tonellotto, Y. He, A. Lipani, G. McDonald, C. Macdonald, I. Ounis (Eds.), Advances in Information Retrieval, Springer Nature Switzerland, Cham, 2024, pp. 449–458.

[35] T. Sellam, D. Das, A. Parikh, BLEURT: Learning robust metrics for text generation, in: D. Jurafsky, J. Chai, N. Schluter, J. Tetreault (Eds.), Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Online, 2020, pp. 7881–7892. URL: https://aclanthology.org/2020.acl-main.704. doi:10.18653/v1/2020.acl-main.704.

A. BeamAttack Algorithm

Input: $S = [w_0, w_1, \ldots, w_n]$ // input tokenized sentence, Y // gold-label, K // beam size, B // branching factor, H // hypothesis count, $model$ // the target model to attack
Output: S_{adv} // adversarial sample
Initialize the beam as a priority queue with the original sentence and its score $beam \leftarrow$ PriorityQueue() $beam.add((S, model.predict\_proba(S)[Y]))$

begin FCalculateWordImportance
  $importance\_scores \leftarrow []$ **foreach** *word* w_i *in* S **do**
    $score\_LIME \leftarrow$ `LIME_importance`$(w_i, S, model)$ $score\_logit \leftarrow$ `logit_importance`$(w_i, S, model)$
    $importance\_scores.append((w_i, score\_LIME + score\_logit))$
  end
  return $importance\_scores$
end

begin FGetTopReplacements
  Input: $word, S, B$
  $replacements \leftarrow$ masked_language_model.predict($word, S, B$) **return** $replacements$
end

$importance\_scores \leftarrow$ `CalculateWordImportance`$(S, model)$ Sort $importance\_scores$ by descending order

while *beam is not empty* **do**
  $current\_beam \leftarrow []$ **foreach** $(S_{current}, score)$ *in beam* **do**
    foreach *word* w_j *in* $importance\_scores$ **do**
      $replacements \leftarrow$ `GetTopReplacements`$(w_j, S_{current}, B)$ **foreach** *replacement in replacements* **do**
        $S_{new} \leftarrow S_{current}$ with w_j replaced by replacement $new\_score \leftarrow model.predict\_proba(S_{new})[Y]$
        $current\_beam.add((S_{new}, new\_score))$
      end
    end
  end
  Sort $current\_beam$ by score $beam \leftarrow$ top K items from $current\_beam$
  Check for success: **foreach** $(S_{candidate}, score)$ *in beam* **do**
    if *model.predict*$(S_{candidate}) \neq Y$ **then**
      $success\_candidates.add((S_{candidate}, score))$
    end
  end
  if *length of* $success\_candidates \geq H$ **then**
    break
  end
end

$S_{adv} \leftarrow$ `SelectBestCandidate`$(success\_candidates)$

return S_{adv} if $success\_candidates$ is not empty, otherwise **return** None

begin FSelectBestCandidate
  $best\_candidate \leftarrow$ None $best\_similarity \leftarrow -\infty$ **foreach** $(S_{candidate}, score)$ *in* $success\_candidates$ **do**
    $similarity \leftarrow$ Bleurt_score($S_{candidate}, S$) **if** $similarity > best\_similarity$ **then**
      $best\_candidate \leftarrow S_{candidate}$ $best\_similarity \leftarrow similarity$
    end
  end
  return $best\_candidate$
end

begin FLIME_importance
  Input: $word, S, model$
  return LIME_score
end

begin FLogit_importance
  $masked\_S \leftarrow S$ with $word$ replaced by [MASK] **return** $model.predict\_proba(S)[Y] - model.predict\_proba(masked\_S)[Y]$
end

Algorithm 1: BeamAttack

B. Ablation Studies

Task	Victim	**Importance Method**	k, h, b	Queries per example	Success	Semantic	Character	BODEGA
RD	BERT	LIME	10, 10, 10	16342.9	0.7	0.8160	0.9174	0.5380
RD	BERT	logit-based	10, 10, 10	5947.0	0.9	0.8189	0.9130	**0.6874**
FC	BERT	LIME	10, 10, 10	5520.44	1.0	0.8392	0.9546	**0.8044**
FC	BERT	logit-based	10, 10, 10	732.66	1.0	0.8188	0.9466	0.7787
FC	BiLSTM	LIME	10, 10, 30	2597.3	0.985	0.8111	0.9332	0.7511
FC	BiLSTM	logit-based	10, 10, 30	1549.1	1.0	0.8059	0.9410	**0.7642**
HN	BiLSTM	LIME	10, 10, 20	1529.04	1.0	0.9043	0.9837	0.8905
HN	BiLSTM	logit-based	10, 10, 20	665.72	1.0	0.9112	0.9893	**0.9031**
FC	RoBERTa	LIME	10, 10, 20	557.08	1.0	0.8398	0.9644	**0.8110**
FC	RoBERTa	logit-based	10, 10, 20	112.4	1.0	0.8392	0.9641	0.8105
C19	RoBERTa	LIME	40, 10, 40	9714.4	1.0	0.5625	0.8602	0.4860
C19	RoBERTa	logit-based	40, 10, 40	5505.0	1.0	0.6541	0.8939	**0.5853**
C19	RoBERTa	LIME	10, 10, 20	1858.0	1.0	0.6304	0.8669	**0.5500**
C19	RoBERTa	logit-based	10, 10, 20	972.7	1.0	0.6120	0.8677	0.5316

Table 7
Evaluation metrics for different importance methods on different tasks and victims. We always bold the better BODEGA score with the same settings for either LIME or BERT. k refers to the selected beam size, h to hypothesis counts and b to the branching factor.

beam size k branching factor b	Subset size	Queries per example	Success	Semantic	Character	BODEGA
10	30	**530.53**	0.93	0.58	0.78	0.44
20	30	1334.07	1.00	0.56	0.78	0.47
30	30	2451.77	1.00	0.58	0.79	0.49
35	30	3090.13	1.00	0.59	0.81	0.50
40	30	3656.93	1.00	0.60	0.83	0.51
50	30	5308.23	1.00	0.60	**0.84**	0.52
60	30	7151.87	1.00	0.61	**0.84**	0.53
70	30	9279.90	1.00	0.61	0.83	0.53
80	30	11374.77	1.00	0.62	**0.84**	0.54
90	30	14546.27	1.00	0.62	**0.84**	0.54
100	30	16499.97	1.00	**0.63**	**0.84**	**0.55**

Table 8
Scores for Different Beam Sizes k for task **PR2** and victim **RoBERTa**. Here hypothesis counts h is always 10 and $b = k$ for each case.

branching factor b	Beam size k	Subset size	Queries per example	Success	Semantic	Character	BODEGA
30	40	30	**3032.733**	1.0	0.5835	0.7924	0.4904
40	40	30	3656.933	1.0	0.5993	0.8280	0.5130
50	40	30	4476.300	1.0	0.5957	**0.8331**	0.5115
60	40	30	4836.333	1.0	0.6058	0.8309	0.5210
70	40	30	5799.533	1.0	**0.6113**	0.8249	**0.5253**

Table 9
Scores for Different branching factors b with a fixed Beam size k for task **PR2** and victim **RoBERTa** with hypothesis counts h = 10.

Task	**hypothesis count** h	beam size k	branching factor b	Subset size	Queries per example	Success	Semantic	Character	BODEGA
RD	20	20	40	20	37576.3	0.95	0.6645	0.7952	0.5501
RD	10	20	40	20	37764.7	0.95	0.6654	0.7951	0.5508
PR2	20	40	40	30	3744.8	1.0	0.5996	0.8235	0.5101
PR2	10	40	40	30	3656.9	1.0	0.5993	0.8280	0.5130
C19	20	40	40	10	5505.0	1.0	0.6541	0.8939	0.5853
C19	10	40	40	10	5505.0	1.0	0.6541	0.8939	0.5853

Table 10
Results of different hypothesis counts h with the **RoBERTa** victim and different tasks.