TurQUaz at CheckThat! 2024: Creating Adversarial Examples using Genetic Algorithm

Basak Demirok[1,*,†], Selin Mergen[1,†], Bugra Oz[1] and Mucahid Kutlu[2,*]

[1]TOBB University of Economics and Technology, Ankara, Türkiye
[2]Qatar University, Doha, Qatar

Abstract

As we increasingly integrate artificial intelligence into our daily tasks, it is crucial to ensure that these systems are reliable and robust against adversarial attacks. In this paper, we present our participation in Task 6 of CLEF CheckThat! 2024 lab. In our work, we explore several methods, which can be grouped into two categories. The first group focuses on using a genetic algorithm to detect words and changing them via several methods such as adding/deleting words and using homoglyphs. In the second group of methods, we use large language models to generate adversarial attacks. Based on our comprehensive experiments, we pick the genetic algorithm-based model which utilizes a combination of splitting words and homoglyphs as a text manipulation method, as our primary model. We are ranked third based on both BODEGA metric and manual evaluation.

Keywords

Adversarial Examples, Credibility Assessment, Natural Language Processing, Robustness

1. Introduction

With the impressive developments in artificial intelligence (AI) technologies, we started to use AI tools in various daily tasks. Therefore, any failure of these models might negatively affect our lives. While it is already challenging to develop robust models that can work correctly in real-world, people might also intentionally attempt to deceive these models with adversarial examples [1]. Thus, it is crucial to develop robust models which are not vulnerable to such attacks.

In this paper, we present our participation in Task 6 [2] of CLEF-2024 CheckThat! Lab [3]. We explore several approaches to create adversarial examples. Our proposed methods can be grouped into two groups: i) genetic algorithm-based methods and ii) large language model (LLM) based methods. In genetic algorithm [4, 5] based methods, we first identify the words that need to be manipulated. Next, we apply various text manipulation methods including adding/removing a letter, shuffling the order of the letters, using homoglyphs of letters, and splitting words by inserting space character. Regarding LLM-based approaches, we propose three different approaches: i) paraphrasing the text via LLAMA3, ii) utilizing LLMs to identify words to be manipulated, and iii) using LLMs to directly create adversarial examples.

In our experiments, we observe that genetic algorithm-based methods outperform all LLM-based approaches. Among genetic algorithm-based methods, the one that uses a combination of splitting words and using homoglyphs outperforms others. Thus, we use this model as our primary model. In the official ranking, we are ranked third based on both BODEGA score and manual evaluation.

CLEF 2024: Conference and Labs of the Evaluation Forum, September 09–12, 2024, Grenoble, France
*Corresponding author.
†These authors contributed equally.
g.demirok@etu.edu.tr (B. Demirok); s.mergen@etu.edu.tr (S. Mergen); ioz@etu.edu.tr (B. Oz); mucahidkutlu@qu.edu.qa (M. Kutlu)
0000-0003-1620-2013 (B. Demirok); 0009-0002-3284-9490 (S. Mergen); 0009-0003-7145-6726 (B. Oz); 0000-0002-5660-4992 (M. Kutlu)

2. Related Work

Researchers have investigated various adversarial attacks and defense mechanisms for NLP tasks [6]. Chen et al. [7] examined backdoor attacks where manipulated training data causes models to fail with specific triggers but perform normally otherwise. Yang et al. [8] demonstrated that altering a single word embedding can fool sentiment analysis models without affecting clean data results. Kurita et al. [9] compared various backdoor attacks for different NLP tasks, finding that attack success varies across tasks. Dai et al. [10] showed that inserting trigger sentences into LSTM-based models is highly effective. In this study, we target already trained models.

Researchers have also examined the vulnerabilities of NLP models in a black-box setting. The methods explored in prior work can be categorized into three types: 1) character-level changes, where words are modified with different spelling errors [11], 2) word-level changes, involving the replacement, removal, or addition of words [12], and 3) sentence-level changes, where new sentences or phrases are added, or existing ones are removed or paraphrased [13]. While our text manipulation techniques are similar to the ones in the literature, we use a genetic algorithm to decide the words to be changed and how to change them. In addition, we explore the utilization of LLMs for adversarial example generation.

3. Proposed Methods

In our work, we propose two different methods including a genetic algorithm-based approach and LLM-based approach. Now we explain these methods.

3.1. Genetic Algorithm Based Approach

Some words are more important than others in several NLP tasks. For instance, let us consider the following statement for the fact-checking task: *The capital city of Turkiye is Ankara.* If we change the word **Ankara** to any other city name, the statement will be false. Therefore, if we can make the word *Ankara* unreadable for the models, it is likely that the model will be confused in the prediction. Once an important word is selected, then the next question is how to modify it to fool the models. Therefore, our approach can be considered in two steps: i) detecting the words to be modified and ii) applying the modification method. Now we explain these two steps in detail.

3.1.1. Selecting Words to be Modified

We develop a genetic algorithm for selecting the words to be modified. **Algorithm 1** describes our genetic algorithm. As the first step, we tokenize the input text and create potential mutations to form an initial population [**Line 1**]. Based on our mutation strategy, we apply mutations to each word with a probability defined by the *mutation_rate* (0.1) Each candidate's fitness is evaluated based on its ability to deceive the target model [**Line 3**]. If a candidate changes the label of the input, it receives a fitness score which is reflective of the modifications made. Otherwise, the candidate gets a 1000-point penalty additional to the modifications. Through the selection phase, the most promising candidates are retained, and through genetic operations like crossover and mutation, a new generation of text variants is created [**Lines 4-5**]. The crossover operation is executed by selecting a random, appropriate point in the token list of the chosen parents, ensuring that the structural integrity of words is maintained. Offspring are then produced by merging segments from each parent up to and beyond this point [**Line 5**]. As for the mutation, it alters each token based on the chosen mutation strategy/strategies, such as homoglyph replacement or various strategic word splits with a probability defined by the *mutation_rate* [**Line 5**]. We explain the mutation strategies in detail in Section 3.1.2. This iterative process continues until a successful adversarial text is generated or the maximum number of generations is reached [**Lines 2-7**]. If no successful manipulation is achieved, the original text is returned as a fallback [**Line 8**]. We set the maximum number of generations to 10 and the population size to 20 in all our experiments.

Algorithm 1 : Genetic Algorithm Structure

Initialize Population: generate initial $population\_size$ many mutations.
for $generation = 1$ **to** $max\_generations$ **do**
    Evaluate Fitness: calculate fitness for each candidate.
    Selection: select the top half of the population by fitness.
    Crossover and Mutation: apply crossover and mutation on selected parents to create new population.
    if any candidate change the label **then**
        return adversarial text
return original_text *as fallback*

3.1.2. Mutation Methods

In this section, we explain the word modification techniques used in our study. **Figure2** shows an example modified sentence for each method.

Homoglyph Replacement (HomoglyphRep). Some letters are visually similar, i.e., homoglyphs, but their encodings are different. So in this approach, we replace the characters with their visually similar ones. **Figure 1** shows the letters we replaced and which letters were used for the replacement. In case there are multiple homoglyphs for a letter, we randomly select one of them.

Character	Homoglyphs
a	а (Cyrillic 'a'), ɑ (Latin small letter alpha)
e	е (Cyrillic 'e)
o	о (Cyrillic 'o'), ο (small omicron), ｏ (Fullwidth Latin small 'o')
c	с (Cyrillic 'c'), ϲ (Greek lunate sigma symbol)
p	р (Cyrillic 'p')
x	х (Cyrillic 'c')
y	у (Cyrillic 'u')
i	і (Cyrillic 'i')
l	ⅼ (Small Roman numeral for fifty)

Figure 1: The letters with their homoglyphs used in our study. While there are other homoglyph character options available, we select only those that are indistinguishable to the human eye.

Splitting Words Randomly ($\textbf{Split}_\textbf{R}$). If a word is split into two, we can still easily understand the meaning of the corresponding sentence/phrase[1]. However, models like BERT can be highly affected by this kind of typo because it will lead to incorrect tokenization and it might cause getting out-of-vocabulary representations for these words. Therefore, in this approach, we split words by adding a space character to a randomly selected index of the word.

Splitting Words Meaningfully ($\textbf{Split}_\textbf{M}$). In this method, instead of splitting the words from a random position, we focus on trying to create meaningful words after splitting, e.g., "langu age". By this approach, the model will get a completely irrelevant but valid word, causing huge changes in its representation. We use the NLTK word corpus to identify the valid words and split them accordingly. As an exceptional case, we avoid splitting the first and last characters unless the word starts or ends with 'a'.

Split Words Heuristically ($\textbf{Split}_\textbf{H}$). In this method, we split based on the first and the last character of the targeted word. In particular, if the targeted word starts or ends with characters 'a' or 'i', we divide them from the beginning or the end accordingly. Otherwise, we choose a random index to split.

$\textbf{Combine}_{\textbf{HomoglyphRep\&Split}}$: In this method, we combine *HomoglyphRep* and $Split_H$ methods. In particular, we randomly choose one of them and apply accordingly.

[1] e.g., "nat ural language processing"

Combine$_{\text{HADSSh}}$. In this method, we randomly select from one of the following methods: i) HomoglyphRep, ii) Split$_{\text{R}}$, iii) adding random characters into words, iv) deleting a randomly selected letter, and v) shuffling the order of the letters within the targeted word.

Mutation Method	Modified Text
HomoglyphRep	Emma Watson . Emma Charlotte Duerre Watson (born 15 April 1990) is a Frenᴄh - British actress , model , and activist . ~ Emma Watson is an Italiɑn actress .
SplitR	Emma Watson . Emma Cha rlotte Duerre Watson (born 15 Apr il 1990) is a French-British actress , model , and activist . ~ Emma Watson is an Ita lian actress .
SplitM	Emma Watson . Emma Charlotte Duerre Watson (born 15 April 1990) is a French-British actress , model , and activist . ~ Emma Watson is an Ita lian actress .
CombineHomoglyphRep&Split	Emma Watson . Emma Charlotte Duerre Watson (born 15 April 199 0) is a Frenᴄh - British actress , model , and activist . ~ Emma Watson is an Ita lian actress .
CombineHADSSh	Emma Watson . Emma Charlotte Duerre Watson (born 15 pril 1990) is a Fr enᴄh - British actr ess , modehl , and activist . g~ Emma Watson is an Italian actress .
SplitH	Emm a Watson . Emma Cha rlotte Duerre Watson (born 15 April 1990) is a French-British actress , model , and activist . ~ Emma Watson is an Itali an actress .

Figure 2: Example outputs of genetic algorithm-based attacks. The texts written in red show the modified parts in original sentence: *"Emma Watson. Emma Charlotte Duerre Watson (born 15 April 1990) is a French-British actress , model , and activist . Ẽmma Watson is an Italian actress."*

3.2. LLM Based Approach

As LLMs have impressive performance in text generation and semantic analysis of text, we investigate how they can be utilized to create adversarial examples. We propose three different methods based on LLMs, LLAMA 3[2] and Mistral[3]. The details of these methods are explained below. **Figure3** shows an example modified sentence for each LLM based method.

3.2.1. Paraphrasing with LLMs (LLM$_{\text{Paraphrase}}$)

In this method, we explore the impact of paraphrasing using LLMs. We use LLAMA3 to paraphrase the texts with the following prompt: *"Paraphrase the following sentence with similar length T: S"* where S is the input sentence and T represents the token count of the given text.

3.2.2. Identifying Words to be Changed Using LLMs (LLM$_{\text{Identify}}$)

In this method, we use LLMs to identify the words that convey the most important information for the general meaning of the given text. We directly ask LLAMA 3 to identify two important words and then apply *HomoglyphRep* method for those methods. We use following prompt for this task: *"You are an information extractor and your task is to extract and return the two most important words that convey the meaning of the sentence. You should output the extracted words in the 'word1, word2' format. Sentence: {sentence}"*

3.2.3. Creating Adversarial Examples (LLM$_{\text{Adversarial}}$)

In this method, we utilize LLMs to create adversarial examples and pre-evaluate its validity by another LLM. **Figure 4** shows the process flow of our method. In particular, firstly, we do not ask only to paraphrase a given text but ask LLAMA3 to create an adversarial example for a given text. Next, we

[2]https://ollama.com/library/llama3:8b
[3]https://ollama.com/library/mistral v02

Mutation Method	Original Text	Modified Text
LLMParaphrase	Spider-Man 2. Spider-Man 2 was released in both conventional and IMAX theaters on June 30 , 2004 . ~ Spider-Man 2 was released in 2004.	Spider-Man's sequel debuted in regular cinemas and IMAX screens on June 30, 2004.
LLMIdentify	Dan Brown. His books have been translated into 52 languages , and as of 2012 , sold over 200 million copies . ~ Dan Brown is an author.	Dan Brown. His books have been translated into 52 languages , and as of 2012 , sold over 200 million copies . ~ Dan Brown is an aut**h**or.
LLMAdverserial	Kendall Jenner. As of April 2017 , she is one of the top 15 most followed celebrities on Instagram . ~ Kendall Jenner is unfamous.	Kendall Jenner, a renowned artist, has only 1.5 million followers on Instagram as of 2023, which is unusually low for someone of her stature.

Figure 3: Example outputs of LLM-based attacks. The red letter shows the modified letter by HomoglyphRep method.

ask Mistral to check if the generated text is an adversarial attack for the corresponding task. If Mistral verifies it, we use that generated text. Otherwise, we generate another sample using LLAMA3. This generation and verification process continues at most three iterations. After three attempts, we use LLAMA3's output although Mistral does not verify that.

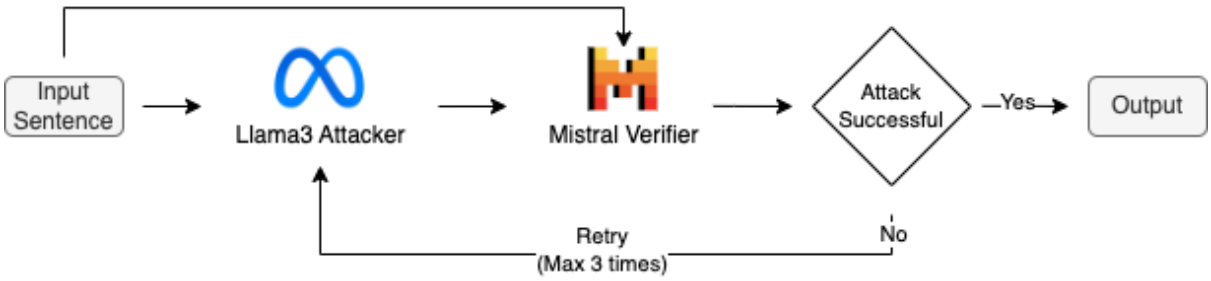

Figure 4: The process flow of our method $\text{LLM}_{\text{Adversarial}}$.

4. Experiments

In this section, we present the experimental setup and the results.

4.1. Experimental Setup

Datasets. The dataset shared by the organizers of the lab covers five different binary classification tasks: Style-based news bias assessment (HN), propaganda detection (PR2), fact-checking (FC), rumor detection (RD), and COVID-19 misinformation detection (C19). **Table 1** provides statistics about the datasets.

Table 1
Dataset size for each task.

Task	Train	Development	Test
Style-based news bias assessment	60,234	3,600	400
Propaganda Detection	11,546	3.186	407
Fact Checking	172,763	10,010	405
Rumor Detection	8,683	2,070	415
COVID-19 Misinformation Detection	1,130	-	595

Evaluation Metrics: This task uses the Bodega score [14] to evaluate the systems, which is basically

the multiplication of confusion score (or success rate), semantic score, and character score. This score takes values between 0 and 1. A high score indicates that the model is deceived by preserving the meaning and appearance, while a low score indicates a weak deception by changing the meaning and/or appearance.

4.2. Experimental Results

In this section, we present results for our proposed methods on the test set. Firstly, we present the result for the genetic algorithm-based approaches against victim models by taking the average of all problem domains (Section 4.2.1). Next, we report the results for both LLM-based and genetic algorithm-based approaches in the fact-checking task (Section 4.2.2). Lastly, we present our official results (Section 4.3).

4.2.1. Results for Genetic Algorithm Based Approach

We report genetic algorithm-based results when the target model is BiLSTM, BERT, and Surprise. The results are shown in **Table 2.** Our observations based on the results are as follows. Firstly, among split models, Split_M has significantly lower success rate and BODEGA score, but slightly higher semantic and character scores. While Split_R and Split_H have highly similar scores, Split_R outperforms Split_R slightly in terms of BODEGA. Secondly, HomoglyphRep achieves the highest semantic score in all cases and the highest character score in two cases, but its BODEGA and success rate are lower than our combination-based methods. Thirdly, $\text{Combine}_{\text{HomoglyphRep\&Split}}$ outperforms $\text{Combine}_{\text{HADSSh}}$ in all cases in terms of BODEGA, suggesting that we can focus on only a few text manipulation methods instead of covering all. Finally, among all models, $\text{Combine}_{\text{HomoglyphRep\&Split}}$ and Split_R has the same average BODEGA score, but $\text{Combine}_{\text{HomoglyphRep\&Split}}$ slightly outperforms Split_R in terms of success rate and semantic score.

Table 2
Average performance of approaches on all problem domains on the test set. Evaluation measures include BODEGA score (B.), success rate (SR), Semantic Score (SemSc), character score (CharSc) and number of queries to the attacked model (Q.)[14]. The best performing score for each case is written in **bold**.

Target Model	Method	BODEGA	SR	SemSc	CharSc	Q.
BiLSTM	Split_R	0.61	0.89	0.70	0.97	97.98
	Split_H	0.61	0.89	0.70	0.97	103.30
	Split_M	0.50	0.69	0.73	**0.98**	172.11
	HomoglyphRep	0.59	0.83	**0.74**	0.95	149.41
	$\text{Combine}_{\text{HomoglyphRep\&Split}}$	**0.62**	0.91	0.71	0.95	116.80
	$\text{Combine}_{\text{HADSSh}}$	0.59	**0.94**	0.66	0.95	118.86
BERT	Split_R	**0.50**	0.75	0.68	0.96	157.77
	Split_H	**0.50**	0.75	0.68	0.97	162.17
	Split_M	0.44	0.63	**0.71**	**0.98**	203.51
	HomoglyphRep	0.49	0.73	**0.71**	0.93	229.69
	$\text{Combine}_{\text{HomoglyphRep\&Split}}$	**0.50**	**0.77**	0.68	0.94	213.93
	$\text{Combine}_{\text{HADSSh}}$	0.47	**0.77**	0.64	0.94	227.60
Surprise	Split_R	**0.37**	0.56	0.68	0.96	236.94
	Split_H	0.36	0.55	0.67	0.96	241.56
	Split_M	0.31	0.44	**0.70**	**0.98**	276.73
	HomoglyphRep	0.36	0.55	**0.70**	0.92	341.65
	$\text{Combine}_{\text{HomoglyphRep\&Split}}$	0.36	0.55	0.68	0.94	331.73
	$\text{Combine}_{\text{HADSSh}}$	0.34	**0.57**	0.64	0.94	336.89

4.2.2. Results for LLM Based Approach

As getting results for LLM based approaches require much more computation power and time, we could get results only for the fact checking task and for BERT and BiLSTM models. **Table3** shows the results

for LLM based approaches and also corresponding genetic algorithm based approaches for comparison.

All LLM based approaches resulted in lower BODEGA scores compared to all genetic algorithm based methods. Furthermore, genetic algorithm based methods achieve very high success rates and character scores. $\text{Combine}_{\text{HomoglyphRep\&Split}}$ achieves a perfect success rate in both cases but HomoglyphRep slightly outperforms $\text{Combine}_{\text{HomoglyphRep\&Split}}$ in terms of average BODEGA score.

Among LLM based approaches, our results are mixed. $\text{LLM}_{\text{Identify}}$ achieves the lowest BODEGA score when the target model is BERT. However, it yields the highest BODEGA score when the target model is BiLSTM. This suggests that BiLSTM models are highly affected by homoglyph attacks. Interestingly, $\text{LLM}_{\text{Paraphrase}}$ outperforms $\text{LLM}_{\text{Adversarial}}$ in both target models although the prompt we use in $\text{LLM}_{\text{Paraphrase}}$ just asks to paraphrase the text without any intention of creating an adversarial example while we ask to create adversarial example in $\text{LLM}_{\text{Adversarial}}$.

Table 3
Results of LLM-used and GA-used methods on **Fact-Checking Task's** attack dataset. Evaluation measures include BODEGA score, success rate (SR), semantic score (SemSc), character score (CharSC) and number of queries to the attacked model (Q.)[14]. **Bold** scores indicate the highest score of the corresponding target model.

Target Model	Method	BODEGA	SR	SemSc	CharSC	Q.
BERT	$\text{LLM}_{\text{Paraphrase}}$	0.066	0.380	0.429	0.397	2.380
	$\text{LLM}_{\text{Identify}}$	0.027	0.032	0.86	0.96	2.02
	$\text{LLM}_{\text{Adversarial}}$	0.056	0.496	0.312	0.326	2.496
	Split_{R}	0.73	0.98	0.76	0.97	69.20
	Split_{H}	0.72	0.97	0.77	0.97	75.41
	Split_{M}	0.63	0.82	0.78	**0.98**	132.55
	HomoglyphRep	**0.75**	0.97	0.82	0.94	79.85
	$\text{Combine}_{\text{HomoglyphRep\&Split}}$	0.74	**1.00**	0.78	0.95	70.70
	$\text{Combine}_{\text{HADSSh}}$	0.69	0.98	0.74	0.95	99.42
BiLSTM	$\text{LLM}_{\text{Paraphrase}}$	0.070	0.404	0.426	0.402	2.404
	$\text{LLM}_{\text{Identify}}$	0.110	0.130	**0.867**	0.969	2.128
	$\text{LLM}_{\text{Adversarial}}$	0.048	0.420	0.312	0.319	2.420
	Split_{R}	0.78	**1.00**	0.80	**0.98**	38.60
	Split_{H}	0.77	**1.00**	0.79	**0.98**	42.55
	Split_{M}	0.65	0.83	0.79	**0.98**	121.60
	HomoglyphRep	**0.79**	0.98	0.84	0.95	54.54
	$\text{Combine}_{\text{HomoglyphRep\&Split}}$	**0.79**	**1.00**	0.82	0.97	39.44
	$\text{Combine}_{\text{HADSSh}}$	0.75	**1.00**	0.77	0.97	45.67

4.3. Official Ranking

We submitted the results of $\text{Combine}_{\text{HomoglyphRep\&Split}}$ as our official run because of its superior performance on average. We achieved 0.4859 BODEGA score on average, ranking third among participants. Based on the results with manual annotations for preserving meaning, we achieved 0.62, ranking again third.

5. Conclusion

In this paper, we present our participation in Task 6 of the CLEF 2024 CheckThat! Lab. In our study, we explore two different approaches to create adversarial examples. In the first approach, we use a genetic algorithm to detect the words to be changed and to identify text manipulation methods. We investigate various text manipulation methods, such as adding/deleting a letter, using homoglyphs, and shuffling the order of letters within a text. In the second approach, we utilize large language models to create adversarial examples. This involves three different methods: asking LLMs to paraphrase a given text, using LLMs to directly generate adversarial examples, and employing LLMs to identify the words that need to be changed to create adversarial examples.

In our comprehensive set of experiments, which involve six different tasks, three different target models, and a total of nine methods, we have the following observations. Firstly, genetic algorithm-based methods outperform all LLM-based approaches. Secondly, among the genetic algorithm-based methods, using the combination of homoglyphs and splitting words as text manipulation outperforms the other methods. This suggests that we need to be more selective in text manipulation methods instead of using all possible methods. In the official ranking, our primary model is ranked third based on the BODEGA score and semantic preservation scores which are based on manual annotations.

In the future, we plan to extend this work in two different directions. Firstly, although LLM models did not achieve high performance in this task, we believe that their effectiveness can be improved through several strategies, such as using different prompts and fine-tuning them specifically for this task. Secondly, regarding the genetic algorithm-based methods, we plan to explore other text manipulation methods to enhance this model further.

References

[1] W. E. Zhang, Q. Z. Sheng, A. Alhazmi, C. Li, Adversarial attacks on deep-learning models in natural language processing: A survey, ACM Transactions on Intelligent Systems and Technology (TIST) 11 (2020) 1–41.

[2] 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: Working Notes of CLEF 2024 - Conference and Labs of the Evaluation Forum, CLEF 2024, Grenoble, France, 2024.

[3] A. Barrón-Cedeño, F. Alam, J. M. Struß, P. Nakov, T. Chakraborty, T. Elsayed, P. Przybyła, T. Caselli, G. Da San Martino, F. Haouari, C. Li, J. Piskorski, F. Ruggeri, X. Song, R. Suwaileh, Overview of the CLEF-2024 CheckThat! Lab: Check-worthiness, subjectivity, persuasion, roles, authorities and adversarial robustness, in: L. Goeuriot, P. Mulhem, G. Quénot, D. Schwab, L. Soulier, G. M. Di Nunzio, P. Galuščáková, A. García Seco de Herrera, G. Faggioli, N. Ferro (Eds.), Experimental IR Meets Multilinguality, Multimodality, and Interaction. Proceedings of the Fifteenth International Conference of the CLEF Association (CLEF 2024), 2024.

[4] M. Alzantot, Y. Sharma, A. Elgohary, B.-J. Ho, M. Srivastava, K.-W. Chang, Generating natural language adversarial examples, in: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, 2018, pp. 2890–2896.

[5] N. Boucher, I. Shumailov, R. Anderson, N. Papernot, Bad characters: Imperceptible nlp attacks, in: 2022 IEEE Symposium on Security and Privacy (SP), IEEE, 2022, pp. 1987–2004.

[6] K. Ren, T. Zheng, Z. Qin, X. Liu, Adversarial attacks and defenses in deep learning, Engineering 6 (2020) 346–360.

[7] X. Chen, A. Salem, D. Chen, M. Backes, S. Ma, Q. Shen, Z. Wu, Y. Zhang, Badnl: Backdoor attacks against nlp models with semantic-preserving improvements, in: Proceedings of the 37th Annual Computer Security Applications Conference, 2021, pp. 554–569.

[8] W. Yang, L. Li, Z. Zhang, X. Ren, X. Sun, B. He, Be careful about poisoned word embeddings: Exploring the vulnerability of the embedding layers in nlp models, in: Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2021, pp. 2048–2058.

[9] K. Kurita, P. Michel, G. Neubig, Weight poisoning attacks on pretrained models, in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 2020, pp. 2793–2806.

[10] J. Dai, C. Chen, Y. Li, A backdoor attack against lstm-based text classification systems, IEEE Access 7 (2019) 138872–138878.

[11] Y. Belinkov, Y. Bisk, Synthetic and natural noise both break neural machine translation, in: International Conference on Learning Representations, 2018.

[12] D. Jin, Z. Jin, J. T. Zhou, P. Szolovits, Is bert really robust? a strong baseline for natural language

attack on text classification and entailment, in: Proceedings of the AAAI conference on artificial intelligence, volume 34, 2020, pp. 8018–8025.

[13] D. Dogan, B. Altun, M. S. Zengin, M. Kutlu, T. Elsayed, Catch me if you can: Deceiving stance detection and geotagging models to protect privacy of individuals on twitter, in: Proceedings of the International AAAI Conference on Web and Social Media, volume 17, 2023, pp. 173–184.

[14] P. Przybyła, A. Shvets, H. Saggion, Bodega: Benchmark for adversarial example generation in credibility assessment, arXiv preprint arXiv:2303.08032 (2023).