Smart Lexical Search for Label Flipping Adversial Attack

Alberto J. Gutiérrez-Megías, Salud María Jiménez-Zafra
L. Alfonso Ureña-López, Eugenio Martínez-Cámara
Computer Science Department, SINAI, CEATIC, Universidad de Jaén

Abstract

Language models are susceptible to vulnerability through adversarial attacks, using manipulations of the input data to disrupt their performance. Accordingly, it represents a cybersecurity leak. Data manipulations are intended to be unidentifiable by the learning model and by humans, small changes can disturb the final label of a classification task. Hence, we propose a novel attack built upon explainability methods to identify the salient lexical units to alter in order to flip the classification label. We assess our proposal on a disinformation dataset, and we show that our attack reaches a high balance among stealthiness and efficiency.

1 Introduction

Adversarial attacks exploit the weaknesses of victim models through modifications in the model architecture or input data to change their efficiency. These attacks are more dangerous if both the model and the human eye are not able to identify them, using techniques of imperceptible characters or small modifications (Boucher et al., 2022).

Adversarial attacks may focus on decreasing the effectiveness or performance of the victim model depending on their approach. There can be targeted attacks, focused on label flipping, or untargeted attacks that base their strategy on decreasing its performance. In this work, we have performed label flipping inference attacks, using different attacks focusing on the use of different search spaces and stealthy modifications adapted to a real environment.

We study two search strategies, one based on an iterative search and the other focused on finding, through using the post-hoc explainability method SHAP (Lundberg and Lee, 2017), the most important words to modify and change the model label using a few search resources. As a result of this study, we propose an attack that combine the two search algorithms to increase the efficiency of the attack, making it stealthier in more realistic environments. We call this joint attack Hybrid KeyToken Attack.

To evaluate the robustness of the victim model, and the stealthiness of the attack, we need metrics adapted to these cases. The BODEGA framework (Przybyła et al., 2023) provides semantic similarity, Levenshtein distance, and the success of changing the target text label. We use a disinformation dataset with which the model victim RoBERTa-base (Liu et al., 2019) is trained for this task. This language model will be a victim of character, word, and word embedding attacks.

This work performs a system study that uses explainability and an iterative search to flip the label of a victim model. Using a different search space to find the best perturbation, a search space is a series of modifications and constraints to achieve a goal.

The Hybrid KeyToken Attack achieves a similar result than the brute force but in much less time, being more efficient and harder to detect in a real case. Furthermore, we compare the Hybrid KeyTokens Attack with state-of-the-art baseline algorithms in the context of adversarial attacks.

The rest of the paper is organized as follows: section 2 presents the context and the works that support our proposal. Section 3 details the targeting of adversary attacks and the types of attacks carried out in this work. It also explains the search spaces used as well as our novel hybrid attack. Section 4 presents the experimental framework. Section 5 analyse the results obtained and finally section 6 determines conclusions and discusses the future works.

2 Background and Related Works

Advances in machine learning (ML) have resulted in a variety of applications, such as data analysis, autonomous systems, and security methods. Machine learning is being applied in many possible areas of our lives, with easy deployment of new systems and active communication with private data. It is increasingly recognized that ML exposes new vulnerabilities in these systems, but addressing them is still a difficult task to tackle. Part of the solutions found is to identify attacks on these systems and build defenses for them, exploring the opposing relationship between model accuracy and resistance to adversarial manipulation (Papernot et al., 2016).

These security issues have also been transferred to the domain of Natural Language Processing (NLP). Common security vulnerabilities often have defined structures and patterns. These can be detected in real time when bad actors have already exploited them. Techniques exist to detect these problems, but they are not robust (Mahmood and Mahmoud, 2018; Yang et al., 2020). That is why it is important to know the attacks well to create a solution in a specific domain, as in NLP (Ziems and Wu, 2021).

Following a review of (Qiu et al., 2019), vulnerabilities of learning models can be attacked in the training and testing stage. In training, they can be divided into data injection, data modification, and logical corruption. These attacks in the training stage are carried out in three ways:

- **Modify Training Dataset**: The original distribution of the training data is changed by modifying or buffering the training data to make the learning algorithm change.
- **Label Manipulation**: Randomly perturbing labels by selecting a label from the random distribution as the label of the training data, changing 40% of the training data is sufficient to reduce the performance of classifiers using SVM (Biggio et al., 2011).
- **Input Feature Manipulation**: This scenario assumes that the adversaries know the learning algorithm. The following papers (Mei and Zhu, 2015; Biggio et al., 2012) show that injecting data can carefully change the distribution of the training dataset, causing the accuracy of the model to decrease and predicting misclassification labels.

In the test stage, they can access the victim model to obtain specific information. With this information they can attack the model by a white-box and black-box attack approaches.

In recent years, vulnerabilities have also been discovered in language models, creating adversarial attacks designed specifically for NLP tasks. Adversarial attacks are manipulations applied to any input data supplied to a model. These attacks are designed to be imperceptible to a human review and to a trained model, when processing the data already modified by the attack, it causes the model to make an error in its classification (Huang et al., 2017).

These attacks can take various forms, such as substitution, insertion, deletion, and exchange of words/characters in a sentence or in the neighboring words of a target word to introduce disruption. There are two types of adversarial attacks, black-box or white-box, based on the attacker's access to the model parameters (Zhang et al., 2020).

The manipulations mentioned above, such as character-level attacks, can result in misspelled words that spellcheckers can easily detect. Due to the superiority of the word-level attack (Dey et al., 2024), (e.g. BERT-ATTACK or PWW) a comparison of character-level manipulations with these word-level attacks be made in this work.

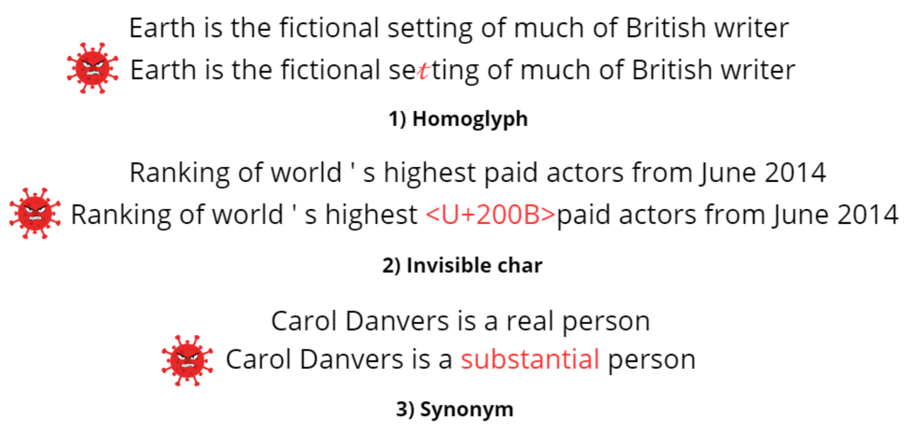
3 KeyToken Adversarial Attacks

We introduce in this section the types of search algorithms that are the backbone of the attacks assessed in this work, explaining how they work and which heuristics of each of them are used in the experiments. In the following subsections, we define how the text parts of a sentence to be modified are selected, explaining a method that focuses its search on explainability using the SHAP method (see section 3.1.1), and another one that uses an iterative method that does not take into account the execution time and its only objective is to change the label of the victim model. Finally, we propose a hybrid system able to use the advantages of the defined heuristics to obtain good performance and to be difficult to identify.

Perturbing a text input with linguistic modifications, such as substitutions or misspellings to damage an NLP model, while respecting certain restrictions, such as semantic similarity, is defined as search space (Morris et al., 2020).

We used the malicious modifications proposed in (Roth et al., 2024), for our targeted attacks focused on flipping the classification label. In particular, we use the following search algorithms:

- **Character-level**: This token modification at-


Earth is the fictional setting of much of British writer
Earth is the fictional se𝑡ting of much of British writer
1) Homoglyph

Ranking of world ' s highest paid actors from June 2014
Ranking of world ' s highest <U+200B>paid actors from June 2014
2) Invisible char

Carol Danvers is a real person
Carol Danvers is a substantial person
3) Synonym

Figure 1: Types of attacks carried out in this paper for each search method.

tack focuses on changing a character of a word into an imperceptible change. There are several well-known techniques, such as Unicode-based replacements, common misspellings, or leetspeak.

- **Word-level**: This method focuses on changing whole words for other words, trying not to change the meaning. Some commonly used methods are synonyms, word embedding-based, or phonetic replacements.

- **Insert word**: Another less stealthy method than the above would be to insert a word into the sentence. This could be words such as adding "bb", adding invisible Unicode characters, or using predefined parse template filling.

Search Space Our modifications to the search space are each of a type of heuristic named above, at the character, word, and insertion level. The search space perturbations are focused on obtaining a high semantic similarity and a minimum Levenstein distance of the target sentence. The perturbations performed are indicated bellow:

- **Homoglyphs**: A character-level attack that modifies a random letter of a target. It uses unique characters that render the same or visually very similar to disrupt model input.

- **Synonyms**: A word-level attack whose function is to replace a token with a synonym while maintaining the same meaning. In our case, we get a similarity vector of the possible synonyms of a word using cosine similarity. The synonym selected to replace the target word will be the one with the farthest similarity, to try to change the label objective, but trying not to change the meaning of the sentence.

- **Invisible character**: We insert an invisible character before the target word. These characters by design are not rendered and are imperceptible to the human eye, but they can change the output of a model at inference time (e.g. zero width space U+200B or Hangul Choseong Filler U+115F).

In any case as a restriction, if the selected word is a single-character word, it will be deleted. As well as if a word cannot be substituted effectively (e.g. it cannot find a synonym for the target). Figure 1 shows an example of each type of attack. In the case of the insertion of an invisible character, the character shown would not be visible, however we have added it in figure 1 for the sake of clarity.

In the following subsections, we define the search methods and the algorithms used. Smart KeyTokens Attacks uses a search algorithm focused on finding the most important words in the input text using SHAP (Lundberg and Lee, 2017). In contrast, a brute force search will also be performed using the same modifications as the other search space.

3.1 KeyTokens Identification

The search method is determined by a transformation and a number of constraints. Heuristic search algorithms cannot guarantee an optimal solution, they can be used to efficiently search a space for a valid adversarial example (Yoo et al., 2020). In our case, we use two types of heuristics to perform the alterations to the texts. This search ends when it succeeds or after a certain number of queries or a set of constraints are met.

The restrictions are adapted depending on the transformation selected in the attack. For synonyms, the synonym that has the most difference in cosine distance to the original word is selected, in order not to change the syntax of the sentence but to be capable of perturbing the output of the victim model. In the case of homoglyphs, only one letter of the text is selected, aiming to reduce the Levenshtein distance as much as possible in addition to achieving a disturbance that is difficult to perceive by both machine and human.

3.1.1 Smart KeyTokens Attacks

The label flipping attacks aim to modify the most influential tokens in the decisions of the models at inference time, obtaining few queries to the victim model and short execution time. Smart KeyToken Attacks use the SHAP algorithm to select the most salient tokens that determine the value of the label in order to modify them with the aim of boosting a label flipping. Attacking only these salient words is more efficient and faster than attacking the whole sentence, and they would even be more difficult to identify than more aggressive attacks, even if they are less effective.

SHAP It is a game theoretic approach to explain the output of any machine learning model. It connects optimal credit allocation with local explanations using the classic Shapley values from game theory and their related extensions. SHAP (SHapley Additive exPlanations) values are a way of explaining the output of any machine learning model. It uses a game theory approach that measures each player's contribution to the final outcome. In machine learning, each feature is assigned an importance value that represents its contribution to the model's outcome.

SHAP values show how each feature affects each final prediction, the importance of each feature compared to the others, and the dependence of the model on the interaction between features. SHAP values are a common way to obtain a consistent and objective explanation of how each feature influences the model prediction.

We use the Hugging Face[1] distilbert-base-uncased-finetuned-sst-2-english (Sanh et al., 2019) model for the weights used by SHAP. It explains the prediction of an instance by calculating the contribution of each feature to the prediction.

SHAP, once it has parsed the sentence, returns a vector of values for each of the tokens and a label associated with a prediction. In our work, SHAP will select between 2 to 5 tokens depending on the length of the target sentence and the most relevant label that has been selected.

3.1.2 Non-Smart KeyTokens Attacks

The objective of these attacks is to change the label regardless of the cost and time to achieve it. The target is the entire sentence, attacking the sentence in different ways until a constraint is matched or the label is flipped.

We follow two straightforward heuristics for Non-Smart KeyTokens Attacks:

- We go through the target sentence word by word, attacking each one and checking if the attack was successful. If it is not successful, we move to the next word leaving the previous one unaltered. If we reach the end of the sentence and the label has not changed, the attack heuristic is unsuccessful.
- The other heuristic uses the same process as the previous one, but when the sentence has finished without success, it will start again, attacking word by word, but leaving the previous one altered. This will be done until all the words of the sentence are modified and the label has not changed, otherwise, it has been successful.

In this work we refer to the latest definition of Non-Smart KeyTokens Attacks as a brute force attack, to differentiate it from the the Smart KeyToken attack.

3.2 Hybrid KeyTokens Attack

Smart KeyToken Attack performs a search for a series of important words to be modified in a sentence, without the need to attack the whole sentence and obtaining good computation time. Non-Smart KeyToken Attack, on the other hand, has two ways of working. The first one is a pass through the phrase modifying each word one by one, without keeping the previous changes. The second is to use brute force by storing each modification made to the previous words, this does not take into account the execution time, although it is more effective. We propose a hybrid system that mixes the two methods, taking advantage of the benefits of both.

The functionality of the hybrid system is to make a first step using the iterative modifications of the

[1]https://huggingface.co/

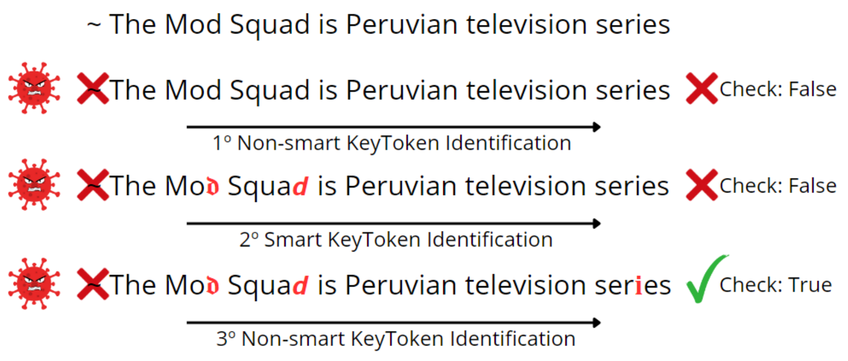

~ The Mod Squad is Peruvian television series
The Mod Squad is Peruvian television series
Check: False
1° Non-smart KeyToken Identification
The Moɒ Squad is Peruvian television series
Check: False
2° Smart KeyToken Identification
The Moɒ Squad is Peruvian television series
Check: True
3° Non-smart KeyToken Identification

Figure 2: Three-step operation of the Hybrid KeyToken Attack. In the (1) step the modifications of each word of the phrase did not modify the output of the model, but the special character has been removed. In the (2) step, SHAP selects the words 'Mod' and 'Squad', after modifying them the output of the victim model is not disturbed, and these changes are stored. In the (3) step, after attacking each word of the sentence, the final result is changed by modifying the word 'series'.

Non-Smart KeyToken Attack, as explained above, each word of the sentence is modified one by one, checking if the modification is successful, without saving the changes made to the previous word. If unsuccessful, the Smart KeyToken attack based on the SHAP algorithm is performed returning the most important words of the phrase to be attacked. If after attacking these words the result is not satisfactory either, we perform a brute force attack by iterating through each word storing the modifications made by SHAP and the current iterative search. This heuristic is shown in Figure 2. This approach eliminates the need to attack every token in the sentence until the target is reached or a constraint is satisfied.

With this system we have get results almost as good as brute force algorithms in much less time, making them more stealthy in a real detection environment.

4 Experimental Framework and Results

We use the BODEGA framework to evaluate our attacks according to a specific evaluation measure that takes into account the effectiveness of the attack and its stealthiness. Likewise, the evaluation measure of BODEGA is defined to evaluate the robustness of classifiers against adversarial attacks by measuring the eligibility of changes made to perturb the output of the victim model. Additionally, the BODEGA framework provides the victim model, RoBERTa-base, trained with the disinformation dataset selected for this task.

4.1 Fact Checking Dataset

We use the dataset[2] FEVER: a large-scale dataset for Fact Extraction and VERification (Thorne et al., 2018) used in the FEVER shared task. The experiments use a disinformation dataset based on Fact Checking (FC). FC is the most advanced way human experts can verify the credibility of a given text: by assessing the veracity of the claims it includes concerning a knowledge base. It deals with Natural Language Inference (NLI) in the field of encyclopedic knowledge and newsworthy events.

The dataset has 51.27% positive labels. The dataset is classified as positive if the assertions are supported. The dataset is balanced, and the victim model has been trained with 172,763 instances of this dataset, leaving 405 instances for adversarial attacks.

4.2 Attack Scenario

In black-box scenarios, there's an assumption that we lack any information regarding the inner mechanisms of the model we're targeting. We can only observe the outputs of the system for a given input. On the other hand, white-box scenarios involve full accessibility to the model, enabling precise tuning of methods for generating adversarial examples based on the model's weights, primarily through gradient-based techniques. The BODEGA framework uses the grey-box scenario approach:

- A "hidden" classifier returns 0, 1 for any input and a probability score that an example is

[2]https://fever.ai/dataset/fever.html

	Con	Sem	Char	**BOD**	Run time (s)	Time/Exe (s)	Queries
Smart KeyTokens Attack							
SHAP Homoglyph	0.254	0.925	0.989	0.232	5495.201	13.568	4.730
SHAP Invisible Char	0.074	0.813	0.985	0.059	5867.365	14.042	4.972
SHAP Synonym	0.241	0.745	0.976	0.176	5748.580	14.194	86.459
Non-Smart KeyTokens Attack							
Homoglyph	0.496	0.894	0.994	0.441	1246.501	3.077	43.143
Invisible Char	0.170	0.794	0.992	0.129	903.621	2.231	46.683
Synonym	0.182	0.753	0.984	0.135	685.911	1.693	46.281
Homoglyph Brute Force	0.916	0.864	0.966	**0.768**	38490.413	95.038	665.496
Invisible Char Brute Force	0.451	0.699	0.936	0.297	57723.456	142.527	1476.503
Synonym Brute Force	0.301	0.722	0.945	0.206	28856.284	71.250	1931.908
Hybrid KeyTokens Attack							
SHAP + Homoglyph	0.708	0.880	0.989	**0.617**	4710.286	11.630	64.254

Table 1: Results obtained after attacking the test set of the Fact-Checking dataset. Results measures include confusion score (Con), semantic score (Sem), character score (Char), BODEGA score (BOD), the total run time of execution (Run time), average time per execution (Time/Exe), and the average of the queries to the attack model (Queries).

assigned a positive class.

- The classifier architecture is a RoBERTa encoder followed by a dense layer and a softmax normalization.
- Training, evaluation, and test data are provided to the attacker.

This configuration allows to discover of classifier vulnerabilities without needing full access to the internal workings of the model while preserving a semblance of real-world applicability.

4.3 Attack Evaluation Measure

The changes between the original text and the altered text are considered to evaluate the experiments. We use the BODEGA score (Przybyła et al., 2023) to evaluate the effectiveness of our adversarial attacks. The subsequent equation defines the BODEGA score:

$$BODEGA\_score(x_i, x_i^*) = Con\_score(x_i, x_i^*) \times Sem\_score(x_i, x_i^*) \times Char\_score(x_i, x_i^*),$$

The semantic score (Sem_score) is based on BLEURT (Sellam et al., 2020). It is designed to compute the similarity between a text and its modified referent, returning a value, being 1 (identical text) and 0 (no similarity). The character score (Char_score) uses the Levenshtein distance to express the difference in the text, returning 1 if there is a high similarity between the target texts and 0 if there is no similarity. To measure the hit ratio, use the measure of confusion (Con_score). This measures when the target text label is successfully changed.

For our experiments, we will also take into account other measures in order to be able to analyze the attacks in more detail. We will evaluate the execution time of the entire test dataset, the average time per query, and the queries made to the victim model during the inference time.

4.4 Results

We performed a series of experiments using Smart and Non-Smart KeyTokens Attacks, which can be shown in Table 1, performed on a test set of 405 instances of the FC dataset. KeyTokens Attacks finds the most important tokens of the target sentence and these are modified by a given technique (homoglyphs, insert invisible character, or synonyms). On the other hand, Non-Smart KeyTokens Attacks go through each token of the target text modifying each one and checking on the change of the final tag, if the result is negative it will go to the next token of the phrase. Finally, this type of attack will also use brute force, when going through the whole sentence it results in no successful change of the tag, another pass will be made but saving the changes made to the tokens one by one.

It is assumed that Smart KeyTokens Attacks are less successful, as not all attack possibilities are explored, although, in a real scenario, they would be the most effective and hardest to find. They are less time-consuming and costly to query the victim

Method	Con	Sem	Char	**BOD**	Run Time (s)	Time/Exe (s)	Queries
BAE	0.518	0.687	0.957	0.342	2823.303	6.971	78.654
BERT-ATTACK	**0.795**	0.732	0.955	0.559	3778.886	168.856	168.856
DeepWordBug	0.456	0.835	0.983	0.375	1237.915	3.056	54.244
Genetic	0.782	0.684	0.938	0.506	119.982	0.296	1293.851
PWWS	0.686	0.709	0.958	0.468	1811.994	4.474	225.474
SCPN	0.679	0.301	0.341	0.074	4074.523	10.060	11.679
Text Fooler	0.676	0.693	0.936	0.442	748.28	1.847	108.703
Hybrid KeyTokens Attack	0.708	**0.880**	**0.989**	**0.617**	4710.286	11.630	64.254

Table 2: Comparison of Hybrid KeyToken Attack models with BODEGA solutions. Results measures include confusion score (Con), semantic score (Sem), character score (Char), BODEGA score (BOD), total run time of execution (Run time), average time per execution (Time/Exe), and the average of the queries to the attack model (Queries).

model. On the other hand, Non-Smart KeyTokens Attacks tend to attack more successfully, although the computation time and queries to the victim model are very high, so they could be easier to detect in a real scenario.

A test with Hybrid KeyToken Attacks has also been carried out to compare the performance with the other experiments. This hybrid system uses the search space that has previously produced the best results, in this case, the homoglyphs. This method uses a first run with the Non-Smart KeyAttacks search method using the technique most successful in the experiments. If this first pass does not achieve a successful result we use Smart KeyToken to find tokens to attack. Finally, we attack the stored tokens and make another pass through the phrase using Non-Smart KeyToken Attacks again, to try to obtain a success without spending so much computation time and to attack the phrase with more performance.

4.5 Baseline Comparison

BODEGA provides the possibility to test different solutions to evaluate the robustness of the models, in this case, there is no data on the use of these solutions in RoBERTa-Large. We evaluate each scenario towards the victim classifier and analyze the differences obtained with the Hybrid KeyTokens Attack. The methods we compare our attack with are as follows:

- **BAE** uses BERT (Devlin et al., 2018) to generate likely candidate words in a given context by inserting existing tokens as new ones (Garg and Ramakrishnan, 2020).

- **BERT-ATTACK** finds a vulnerable word by checking the victim's response, then those words are replaced by BERT candidates (Li et al., 2020).

- **DeepWordBug** searches for an important word and modifies at the character level to create an unknown word with modifications like substitution, insertion, deletion, or reordering (Gao et al., 2018).

- **Genetic** uses a genetic algorithm substituting words, using GloVe (Global Vectors for Word Representation) for the conservation of meaning (Alzantot et al., 2018).

- **PWWS** uses greedy substitution using WordNet to obtain synonym candidates (Ren et al., 2019).

- **SCPN** paraphrases the entire text using a trained model through back-translation from English into Czech (Iyyer et al., 2018).

- **Text Fooler** performs a greedy word search taking into account the syntax of the text and that it makes sense in the sentence (Jin et al., 2020).

Table 2 shows the results obtained after attacking RoBERTa-Large with the solutions provided by BODEGA with the same 405 test instances used in our experiments shown in Table 1. The same evaluation metrics have been taken into account as in our previous experiments to be able to analyze the results later (see section 5).

Most of the heuristics used obtain a similar level of success by changing the final label of the victim model, although both BERT-ATTACK and Genetic do not take into account the semantic score as much, whereas our hybrid model performs much better.

Our system makes half as many calls to the victim model as the second one, BERT-ATTACK, and it is also slower. Here we do not take into account the Genetic algorithm, as it would be very easy to identify in a real context, by the number of queries to the model.

According to the results, our hybrid model has a higher success rate than the BODEGA literature, even when measuring metrics such as execution time or queries performed on the models, constraints that are usually taken into account.

5 Result Analysis

The KeyTokens Attacks performed do not succeed very well in flipping the label of the victim model RoBERTa-base. Although, it can be noted that the homoglyphs obtain a high semantic and character score. All these types of attacks take an average of 14 seconds to execute but taking into account the number of queries made to the victim model, the homoglyphs are the best performers in these experiments using the KeyTokens search method.

Analysing the Non-KeyTokens Attacks it can be differentiated that techniques using brute force are always more successful. On the other hand, homoglyphs are more successful in all metrics, whether they use brute force or not. These attacks, especially brute force attacks, have a very high computational and query time, as they use much more computationally expensive search methods.

In our Proposal Hybrid Attack, we used the technique that has given the best results and SHAP. We obtained results similar to the best results obtained with brute force attacks but using much less computation time and queries to the model. This means, we sacrifice success for efficiency, and in a real case, it would be much harder to detect.

Using techniques such as adding words as invisible characters or changing words into synonyms has obtained very similar results. Our proposed system obtains the best results in a real attack environment. It obtains a success rate of 70% and is almost imperceptible to the human eye and the victim model, in this case, RoBERTa-base.

After obtaining a system with a lower success rate but with a significant improvement in the time and number of queries to the victim model, a comparison is performed with the algorithms provided by BODEGA to attack with adversarial attacks using the same dataset under the same conditions and metrics.

Our hybrid system obtains a higher BODEGA score than all the algorithms provided by the framework for use as adversarial attacks, in this case, the model that had not been evaluated in BODEGA, RoBERTA-Large. The Hybrid KeyToken Attacks stand out for being stealthy and making few queries to the victim model, moreover, except for out-layer cases, the average time per execution is much lower than the second-best result BERT-ATTACK.

6 Conclusion and Future Work

We perform 3 different heuristics to make the adversarial attacks of this work, homoglyphs, invisible characters, and the use of synonyms. We observed that homoglyphs have a better ratio in both success and stealth. This is because it only changes one letter of the attacked text and that it does not remove almost any semantic meaning from the sentence.

Non-Smart KeyToken Attacks using brute force give better results. The brute-force homoglyphs obtain remarkable results compared to the other experiments, but in contrast, it uses a lot of time and queries to the victim model, which makes it easy to detect in a real context. Hybrid KeyToken Attacks reach a balance between disrupting model output, stealth, and queries to the victim model.

It also performs better than the algorithms offered by the BODEGA framework in its literature. Therefore, we can conclude that our hybrid model that uses SHAP-based search spaces to find the most important tokens of a sentence, using as support a greedy system that stores in memory the modifications previously made to the sentence, obtains good, stealthy and difficult to detect results in a real environment, where the execution time and the requests to the victim model are essential.

Our proposal needs several attacking shots to be successful. Hence, a system can be defend attending to the number of consecutive shots regarding a similar input text. An additional defense method may be built upon a machine translation method, since this methods are less vulnerable to character-level modifications like homoglyphs.

As future work, it would also be interesting to create a real-time defense capable of identifying these disturbances, either by execution time or by prior analysis of the input data, and in case of detection, to return the input data to its original state.

Acknowledgments

This work has been partially supported by projects CONSENSO (PID2021-122263OB-C21), MODERATES (TED2021-130145B-I00), SocialTOX (PDC2022-133146- C21) and FedDAP (PID2020-116118GA-I00) funded by MCIN/AEI/ 10.13039/501100011033 and by the "European Union NextGenerationEU/PRTR". The research work conducted by Salud María Jiménez-Zafra has been supported by Action 7 from Universidad de Jaén under the Operational Plan for Research Support 2023-2024.

References

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2890–2896, Brussels, Belgium. Association for Computational Linguistics.

Battista Biggio, Blaine Nelson, and Pavel Laskov. 2011. Support vector machines under adversarial label noise. In *Asian conference on machine learning*, pages 97–112. PMLR.

Battista Biggio, Blaine Nelson, and Pavel Laskov. 2012. Poisoning attacks against support vector machines. *arXiv preprint arXiv:1206.6389*.

Nicholas Boucher, Ilia Shumailov, Ross Anderson, and Nicolas Papernot. 2022. Bad characters: Imperceptible nlp attacks. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1987–2004. IEEE.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Roopkatha Dey, Aivy Debnath, Sayak Kumar Dutta, Kaustav Ghosh, Arijit Mitra, Arghya Roy Chowdhury, and Jaydip Sen. 2024. Semantic stealth: Adversarial text attacks on nlp using several methods. *arXiv preprint arXiv:2404.05159*.

Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. 2018. Black-box generation of adversarial text sequences to evade deep learning classifiers. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 50–56. IEEE.

Siddhant Garg and Goutham Ramakrishnan. 2020. BAE: BERT-based adversarial examples for text classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6174–6181, Online. Association for Computational Linguistics.

Sandy Huang, Nicolas Papernot, Ian Goodfellow, Yan Duan, and Pieter Abbeel. 2017. Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284*.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1875–1885, New Orleans, Louisiana. Association for Computational Linguistics.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. 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, pages 8018–8025.

Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. 2020. BERT-ATTACK: Adversarial attack against BERT using BERT. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6193–6202, Online. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc.

Rahma Mahmood and Qusay H Mahmoud. 2018. Evaluation of static analysis tools for finding vulnerabilities in java and c/c++ source code. *arXiv preprint arXiv:1805.09040*.

Shike Mei and Xiaojin Zhu. 2015. Using machine teaching to identify optimal training-set attacks on machine learners. In *Proceedings of the aaai conference on artificial intelligence*, volume 29.

John X Morris, Eli Lifland, Jin Yong Yoo, and Yanjun Qi. 2020. Textattack: A framework for adversarial attacks in natural language processing. *Proceedings of the 2020 EMNLP, Arvix*.

Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael Wellman. 2016. Towards the science of security and privacy in machine learning. *arXiv preprint arXiv:1611.03814*.

Piotr Przybyła, Alexander Shvets, and Horacio Saggion. 2023. Bodega: Benchmark for adversarial example generation in credibility assessment. *arXiv preprint arXiv:2303.08032*.

Shilin Qiu, Qihe Liu, Shijie Zhou, and Chunjiang Wu. 2019. Review of artificial intelligence adversarial attack and defense technologies. *Applied Sciences*, 9(5):909.

Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. 2019. Generating natural language adversarial examples through probability weighted word saliency. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1085–1097, Florence, Italy. Association for Computational Linguistics.

Tom Roth, Yansong Gao, Alsharif Abuadbba, Surya Nepal, and Wei Liu. 2024. Token-modification adversarial attacks for natural language processing: A survey. *arXiv preprint arXiv:2103.00676*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Thibault Sellam, Dipanjan Das, and Ankur P Parikh. 2020. Bleurt: Learning robust metrics for text generation. *arXiv preprint arXiv:2004.04696*.

James Thorne, Andreas Vlachos, Oana Cocarascu, Christos Christodoulopoulos, and Arpit Mittal. 2018. The fact extraction and verification (fever) shared task. *arXiv preprint arXiv:1811.10971*.

Xueqi Yang, Jianfeng Chen, Rahul Yedida, Zhe Yu, and Tim Menzies. 2020. How to recognize actionable static code warnings (using linear svms). *ArXiv*.

Jin Yong Yoo, John X Morris, Eli Lifland, and Yanjun Qi. 2020. Searching for a search method: Benchmarking search algorithms for generating nlp adversarial examples. *arXiv preprint arXiv:2009.06368*.

Wei Emma Zhang, Quan Z Sheng, Ahoud Alhazmi, and Chenliang Li. 2020. Adversarial attacks on deep-learning models in natural language processing: A survey. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 11(3):1–41.

Noah Ziems and Shaoen Wu. 2021. Security vulnerability detection using deep learning natural language processing. In *IEEE INFOCOM 2021-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 1–6. IEEE.