TextDefendR: Detecting Adversarial Attacks on French Text Classifiers

Gabriel Watkinson
ENSAE
gabriel.watkinson@ensae.fr

Baptiste Pasquier
ENSAE
baptiste.pasquier@ensae.fr

Abstract

Adversarial attacks on natural language processing (NLP) models are becoming increasingly prevalent, with considerable consequences in real-world applications. In this paper, we review recent research on detecting and defending against such attacks, focusing on techniques for identifying adversarial examples and improving model robustness. We examine various methods, including outlier detection and classification techniques, and discuss their strengths and limitations. To illustrate our point, we analysed the French Allociné dataset used for sentiment analysis and applied various adversarial examples and defences to it. Our review highlights the need for continued research in this area to develop more effective and robust defences against adversarial attacks in NLP. [1]

1 Introduction

Natural Language Processing's state of the art has been steadily improving on a wide range of benchmarks such as Language Translation, Question Answering, Sentiment Analysis, and Text Generation. Recent breakthroughs rely on Large Language Models (LLMs), which are themselves based on the self-attention mechanism. As these LLMs get even larger, they can mimic human interactions in a more convincing way. What used to be research topics are now widely integrated into our daily lives.

However, as these models become more ubiquitous, there is a growing need to ensure their fairness and robustness to adversarial attacks. Fairness has become a crucial topic in the NLP community (Pichler et al., 2022; Colombo et al., 2021, 2022b), with many studies investigating ways to detect and mitigate biases in language models. Out-of-Distribution (OOD) detection is another important issue that has emerged with the increased use of LLMs in real-world applications (Darrin et al., 2023; Gomes et al., 2023; Colombo et al., 2022a). OOD detection aims to identify when the input data is outside the range of the training data distribution, which is critical for ensuring the reliability and safety of NLP models. Overall, while LLMs have brought tremendous progress to the field of NLP, it is important to also address these emerging challenges to ensure their responsible use in practice.

Attack	Text	Pred
Original	C'est un sous Alien qui a très mal vieilli car comme tout film de plus de 20 ans, les effets spéciaux sont dépassés.	0%
BAE	C'est un sous Alien qui a très mal vieilli car comme tout film de plus de 20 ans, les effets spéciaux sont mis.	80%
DeepWordBug	C'est un sous Alien qui a très mal vieilli car comme tout film de plus de 20 ans, les effets spéciaux sont dépassvs.	71%
Input Reduction	C'est un a mal tout de, effets sont.	99%
TextBugger	C'est un sous Alien qui a très mal vieilli car comme tout film de plus de 20 ans, les effets spéciaux comme dpassés.	52%

Table 1: Examples of attacks on the Allociné dataset against CamemBERT using different types of attack methods.

In addition to fairness and OOD detection, the detection and defense against adversarial attacks have also gained significant attention (Picot et al., 2022, 2023). Adversarial examples can be generated by introducing imperceptible perturbations to the input data, which can cause the model to produce incorrect outputs. With the release of new libraries making attacks (Morris et al., 2020) simple this problem is of growing interest as the consequences of adversarial attacks on NLP models can be particularly severe in areas such as sentiment analysis and text classification. For example, a malicious actor could manipulate a product review to falsely influence consumer opinion or deceive a

[1] All our code and experiences are available on Github: https://github.com/baptiste-pasquier/TextDefendR

spam filter to infiltrate a network. Spammers can continuously bombard email servers, subtly misspelling words in efforts to evade spam detection while preserving the emails' intended meaning (Lee and Ng, 2005; Fumera et al., 2006). Therefore, it is essential to develop effective techniques for detecting and defending against adversarial attacks in NLP models.

2 Related Work

In recent years, considerable effort has been devoted to defending NLP models against adversarial attacks. We will first define in more details some common attacks, then talk about defense mechanisms.

2.1 Adversarial Attacks

Adversarial attacks in the field of NLP can manifest at different levels of granularity, often in combination, and can be generated in numerous ways.

Character-level attacks are among the easiest to detect, where a letter is replaced with another, typically visually similar (such as replacing "a" with "α" or a "O" with a "0"), or close on a keyboard, to simulate a typing error. Attackers can also randomly remove characters, white spaces, or punctuation marks, which can cause tokenizers to completely misunderstand a sentence.

On the other hand, other attacks may be more difficult for humans to identify. Word and sentence-level attacks fall into this category. In word-level attacks, entire words are replaced with semantically similar ones. In sentence-level attacks, the entire sentence is replaced, often using back-translation (Ribeiro et al., 2018) or paraphrasing models (Iyyer et al., 2018). This makes the attack indistinguishable to humans but can affect the model's prediction.

Combining different types of attacks can often improve their effectiveness. For instance, HotFlip (Ebrahimi et al., 2018) is a popular combination-based attack using both character and word level perturbations.

Most perturbations are context-aware, meaning that word or character-level attacks still take into account the entire sentence, often leveraging the power of pre-trained language models to easily compare input similarity. The BAE attack (Garg and Ramakrishnan, 2020) is an example of a context-aware attack.

For all types of attacks, the result must be semantically similar to the original input, i.e., have the same meaning to a human observer. Therefore, for a given input $x_0 \in \mathcal{X}$, there exists a space $\mathcal{X}_{in}(x_0)$ defined by a set of constraints, such that all elements $z_0 \in \mathcal{X}_{in}(x_0)$ have the "same" meaning as x_0. TextAttack (Morris et al., 2020) provide a framework to easily define such constraints and generate perturbed inputs with the most common methods.

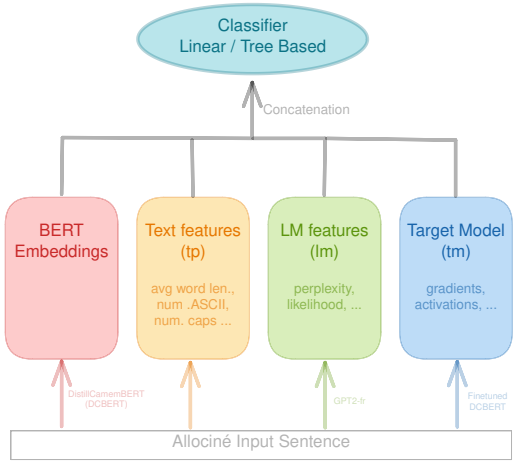


Figure 1: Pipeline for supervised attack detection.

2.2 Defending Against Attacks

Current efforts in defending NLP models against adversarial attacks have focused on developing robust models by design. One approach is to modify the architecture or the training process of the model. Adversarial training (Goodfellow et al., 2014) and data augmentation (Li et al., 2017; Belinkov and Bisk, 2017) are two common techniques used for this purpose.

Data augmentation involves adding perturbed examples to the training data that satisfy the set of constraints, hopping to have a classifier that is more confident on attacks. On the other hand, in adversarial training the model is trained to minimize a loss that is attacked by perturbed inputs. While these methods can improve the model's robustness, they are computationally expensive and require high computational power, as entire models need to be retrained from scratch. This limits the utilization of the vast amount of pre-trained models available.

As a result, a class of defenses emerged that focus on adjusting or correcting already pre-trained models, which is often less computationally intense and interesting in a multitude of cases.

Since most recent models rely on tokenizers and embeddings to process textual data, some researchers have proposed methods that place a correction model in front of the pre-trained downstream classifier, correcting the input tokens, or even imitating the embedding layer (Pruthi et al., 2019; Jezabek and Singh, 2022). However, those solutions are often model dependent, meaning that a correction needs to be learned for each model, and can therefore not be practical in real use cases where we often want to compare multiple models.

Jezabek and Singh (2022) mention tokenizer-free models, such as CharacterBERT (Boukkouri et al., 2020) CANINE (Clark et al., 2022), saying that they could be more resilient to misspelling by default. However, they didn't mention additional ways to improve their robustness outside of adversarial training, as they are probably not natively resistant to more complex transformations (word or sentence level).

Lastly, there are methods that aim to identify out-of-distribution inputs and exclude them. This is particularly useful when we want to filter out attacks, such as in the case of spam filtering. Some methods are unsupervised and require no training, while others are supervised and require adversarial examples. Both types of methods are relatively easy and cost-effective to implement.

One supervised method that we found interesting is the work of Xie et al. (2022), which improves attack detection performance by using additional text-related features in addition to embeddings. These features include text features, for instance the number of letters, non-ASCII characters, and the number of misplaced capitals, etc. They also include language model features, such as input complexity and likelihood. And targeted model features, such as gradient values and activations. Linear and tree-based classifiers trained on these features achieve excellent performance. The Figure 1 presents the pipeline of the model. The authors also describe extensively how to generate the attack examples and how to build the dataset.

We also explored non-supervised methods based on Out-Of-Distribution (OOD) approaches. We focused on the method proposed by Sun et al. (2022), which uses KNN on the embeddings of the inputs. It is appealing because is a light weight, extremely fast to calculate and a non-parametric approach that can be used in most context regardless of the complexity of the embedding space. This method looks at the distance between new inputs and their nearest neighbors in the embedding space. Then if the distance is larger than a precomputed threshold, it is classified as OOD. This simple approach can perform well, and pretraining the embeddings using a contrastive loss further improves performance.

Appendix B contains more explanations on all the methods evoked in this section.

Attack	Access level	Perturbation
BAE	Black box	Word
DeepWordBug	Gray box	Character
Input Reduction		Word
PWWS	Black box	Word
TextBugger	Black box	Character
TextFooler	Black box	Word

Table 2: Characteristics of the different attack methods. Note that Input Reduction works on dropping words all together.

3 Experiments Protocol

Dataset As most paper focus exclusively on the English language, we decided to experiment on a French dataset. We chose a 20000 subset of the Allociné dataset (Blard, 2020), which contains reviews of films that are labelled as positive or negative.

Models To perform this classification task, we tried using both CamemBERT (Martin et al., 2020), a French implementation of RoBERTa (Liu et al., 2019), and a distilled version DistilCamemBERT (Delestre and Amar, 2022). We ended up choosing this distilled version, that we fine tuned for 3 epochs on the Allociné dataset, achieving an accuracy of 97%. [2]

Attack methods In order to evaluate our models, we had to generate some perturbed outputs to add to the dataset. To do so, we followed the recommendation of Xie et al. (2022) using their GitHub repository. We considered the attacks described in Table 2. We focused mainly on word and character level attacks, as they are the cheapest to process. The results and details of the attacks are presented in Table 5. We built a bit more than 9000 attacks with this method.

[2]The model is available on HuggingFace: https://huggingface.co/baptiste-pasquier/distilcamembert-allocine

Embeddings extraction We first include representations corresponding to the BERT (Devlin et al., 2018) embeddings for each sequence (`bert` features). We then add textual properties (`tp` features) such as the number of alphanumeric characters, the number of punctuation, and several features on word case. To further characterize our samples, a next step is to use a GPT2 (Radford et al., 2019) causal language model to encode the perplexity and a RoBERTa masked language model (MLM) to encode the probabilities and ranks of each token. These features are called `lm`. Finally, we encode features from the model we attacked (DistilCamemBERT) by extracting activations and gradients from the internal layers. This configuration is called `default`. We also used equivalent French models, refered as `fr+small`.

Experiments We build a supervised classifier using features extracted from the aforementioned models. We compared the performance of linear classifier with tree based models, on different set of features.

For OOD detection, we finetuned a CamemBERT model using supervised contrastive loss (Khosla et al., 2020). We then used KNN method to define Out-Of-Distribution samples. We applied this method to the same test set in order to compare it to the supervised learning objective. We tested evaluated the method for different values of k ranging from 1 to 200.

4 Results

4.1 Supervised Attack Detection

The results of this experiment were mostly as expected. Using more features globally lead to better results, as we can see in the Table 3 that presents the results for the logistic regressions. We indeed expected for the `fr+small` to perform better, as the used models are trained on French data, however, with all features, the `default` configuration works best. We do not have a clear explanation as to why that is. It may be due to the high number of feature (more than 5000), that add noise in the `fr+small` setting. When looking at the full table of metrics in Table 7, the best model is a LightGBM on all features. It achieves 96.4% accuracy, which is 0.3 more than the `fr+small`.

Embeddings	Features	Model	Accuracy
default	all	LR	0.951
fr+small	all	LR	0.934
fr+small	bert+tp+lm	LR	0.862
fr+small	bert+tp	LR	0.850
fr+small	bert	LR	0.848
default	bert+tp+lm	LR	0.767
default	bert+tp	LR	0.728
default	bert	LR	0.699

Table 3: Results of the logistic regressions in the different settings for the binary experiment.

4.2 Unsupervised OOD using KNN

Using UMAP, the Figure 5 shows the embeddings of the validation set, that contains both clean and attacked text. The contrastive finetuning was performed on the train set, but we see that the classes are still well separated on the validation set. More interestingly, we see that a lot of the attacks are in between both blobs, meaning that the classifier is struggling to classify them. Using KNN in this context should separate the attacks from the clean dataset.

Figure 4 shows the distribution of the KNN-distance. We see that both classes are somewhat separated. From there, we calculated the 90th percentile of the train distribution, to use as the threshold, then proceeded to classify the observations to obtain the metrics. Using $k = 20$, we obtain a weighted accuracy of 0.78, a precision of 0.76, a recall of 0.78 and a f1-score of 0.77. Surprisingly, the number of neighbors does not have a huge impact, the accuracy is equal to $78\% \pm 0.3$ in all settings. This is comparable, if not better, with some of the supervised model, which is quite impressive.

5 Conclusion and Discussion

In this report, we have discussed the challenges associated with adversarial attacks in Natural Language Processing. We have presented an extensive review of literature that covers various types of attacks and the solutions proposed to mitigate them. Our focus has been on attack detection in low-resource settings, and we have evaluated supervised and unsupervised methods on a chosen dataset.

However, there are still several areas that require further development. For example, we have not evaluated how the supervised classification performs on new types of attacks that were not present in the training set, and it is important to

test the model's ability to transfer features to other contexts. Reproducing this experiment on other datasets is also necessary to ensure the validity of the conclusions drawn. Furthermore, the lack of a proper baseline makes it challenging to claim the obtained results as definitive.

Regarding out-of-distribution (OOD) detection, we have only evaluated the performance of supervised contrastive learning on embeddings of CamemBERT in one setting. There is a need to compare the performance of such embeddings with those obtained through regular fine-tuning and explore other optimization techniques that could potentially improve the model. Additionally, choosing the appropriate hyperparameters, including the threshold and the number of neighbors, is crucial. Finally, we could test the effectiveness of character-level models such as CANINE-C to see if they can improve the model's performance significantly.

References

Yonatan Belinkov and Yonatan Bisk. 2017. Synthetic and natural noise both break neural machine translation. *arXiv preprint arXiv:1711.02173*. 2, 7

T Blard. 2020. French sentiment analysis with bert. *GitHub repository, https://github.com/TheophileBlard/french-sentiment-analysis-with-bert.(Last access: March 2021)*. 3

Hicham El Boukkouri, Olivier Ferret, Thomas Lavergne, Hiroshi Noji, Pierre Zweigenbaum, and Junichi Tsujii. 2020. Characterbert: Reconciling elmo and bert for word-level open-vocabulary representations from characters. *arXiv preprint arXiv:2010.10392*. 3, 8

Jonathan H Clark, Dan Garrette, Iulia Turc, and John Wieting. 2022. Canine: Pre-training an efficient tokenization-free encoder for language representation. *Transactions of the Association for Computational Linguistics*, 10:73–91. 3, 8

Pierre Colombo, Chloe Clavel, and Pablo Piantanida. 2021. A novel estimator of mutual information for learning to disentangle textual representations. *arXiv preprint arXiv:2105.02685*. 1

Pierre Colombo, Eduardo Dadalto Câmara Gomes, Guillaume Staerman, Nathan Noiry, and Pablo Piantanida. 2022a. Beyond mahalanobis distance for textual ood detection. In *NeurIPS 2022*. 1

Pierre Colombo, Guillaume Staerman, Nathan Noiry, and Pablo Piantanida. 2022b. Learning disentangled textual representations via statistical measures of similarity. *arXiv preprint arXiv:2205.03589*. 1

Maxime Darrin, Guillaume Staerman, Eduardo Dadalto Câmara Gomes, Jackie CK Cheung, Pablo Piantanida, and Pierre Colombo. 2023. Unsupervised layer-wise score aggregation for textual ood detection. *arXiv preprint arXiv:2302.09852*. 1

Cyrile Delestre and Abibatou Amar. 2022. DistilCamemBERT : une distillation du modèle français CamemBERT. In *CAp (Conférence sur l'Apprentissage automatique)*, Vannes, France. 3

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*. 4

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018. HotFlip: White-box adversarial examples for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 31–36, Melbourne, Australia. Association for Computational Linguistics. 2, 7

Shi Feng, Eric Wallace, Alvin Grissom II, Mohit Iyyer, Pedro Rodriguez, and Jordan Boyd-Graber. 2018. Pathologies of neural models make interpretations difficult. *arXiv preprint arXiv:1804.07781*. 8

Giorgio Fumera, Ignazio Pillai, and Fabio Roli. 2006. Spam filtering based on the analysis of text information embedded into images. *Journal of Machine Learning Research*, 7(12). 2

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. 8

Siddhant Garg and Goutham Ramakrishnan. 2020. Bae: Bert-based adversarial examples for text classification. *arXiv preprint arXiv:2004.01970*. 2, 7, 8

Eduardo Dadalto Câmara Gomes, Pierre Colombo, Guillaume Staerman, Nathan Noiry, and Pablo Piantanida. 2023. A functional perspective on multilayer out-of-distribution detection. 1

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*. 2, 7

Po-Sen Huang, Robert Stanforth, Johannes Welbl, Chris Dyer, Dani Yogatama, Sven Gowal, Krishnamurthy Dvijotham, and Pushmeet Kohli. 2019. Achieving verified robustness to symbol substitutions via interval bound propagation. *arXiv preprint arXiv:1909.01492*. 7

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. *arXiv preprint arXiv:1804.06059*. 2, 7

Jan Jezabek and Akash Singh. 2022. Mockingbert: A method for retroactively adding resilience to nlp models. *arXiv preprint arXiv:2208.09915*. 3, 8

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. 8

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. 2020. Supervised contrastive learning. *Advances in neural information processing systems*, 33:18661–18673. 4

Honglak Lee and Andrew Y Ng. 2005. Spam deobfuscation using a hidden markov model. In *CEAS*. 2

Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. 2018. Textbugger: Generating adversarial text against real-world applications. *arXiv preprint arXiv:1812.05271*. 8

Yitong Li, Trevor Cohn, and Timothy Baldwin. 2017. Robust training under linguistic adversity. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 21–27. 2, 7

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. *arXiv preprint arXiv:1907.11692*. 3

Louis Martin, Benjamin Muller, Pedro Javier Ortiz Suárez, Yoann Dupont, Laurent Romary, Éric Villemonte de la Clergerie, Djamé Seddah, and Benoît Sagot. 2020. Camembert: a tasty french language model. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 3

John X Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. 2020. Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp. *arXiv preprint arXiv:2005.05909*. 1, 2, 7

Georg Pichler, Pierre Jean A Colombo, Malik Boudiaf, Günther Koliander, and Pablo Piantanida. 2022. A differential entropy estimator for training neural networks. In *International Conference on Machine Learning*, pages 17691–17715. PMLR. 1

Marine Picot, Francisco Messina, Malik Boudiaf, Fabrice Labeau, Ismail Ben Ayed, and Pablo Piantanida. 2022. Adversarial robustness via fisher-rao regularization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 1

Marine Picot, Guillaume Staerman, Federica Granese, Nathan Noiry, Francisco Messina, Pablo Piantanida, and Pierre Colombo. 2023. A simple unsupervised data depth-based method to detect adversarial images. 1

Danish Pruthi, Bhuwan Dhingra, and Zachary C Lipton. 2019. Combating adversarial misspellings with robust word recognition. *arXiv preprint arXiv:1905.11268*. 3, 7

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9. 4

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. 8

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Semantically equivalent adversarial rules for debugging nlp models. In *Proceedings of the 56th annual meeting of the association for computational linguistics (volume 1: long papers)*, pages 856–865. 2, 7

Yiyou Sun, Yifei Ming, Xiaojin Zhu, and Yixuan Li. 2022. Out-of-distribution detection with deep nearest neighbors. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 20827–20840. PMLR. 3, 8

Zhouhang Xie, Jonathan Brophy, Adam Noack, Wencong You, Kalyani Asthana, Carter Perkins, Sabrina Reis, Sameer Singh, and Daniel Lowd. 2022. Identifying adversarial attacks on text classifiers. *arXiv preprint arXiv:2201.08555*. 3, 8

Appendices

A Detailed Related Work

In recent years, considerable effort has been devoted to defending NLP models against adversarial attacks. We will first define some common attacks, then talk about defence mechanisms.

A.1 Adversarial Attacks

Adversarial attacks in the field of natural language processing can manifest at different levels of granularity, often in combination, and can be generated in numerous ways.

Character-level attacks are among the easiest to detect, where a letter is replaced with another, typically visually similar (such as replacing "a" with

"α" or a "O" with a "0"), or close on a keyboard, to simulate a typing error. Attackers can also randomly remove characters, white spaces, or punctuation marks, which can cause tokenizers to completely misunderstand a sentence.

On the other hand, other attacks may be more difficult for humans to identify. Word and sentence-level attacks fall into this category. In word-level attacks, entire words are replaced with semantically similar ones. In sentence-level attacks, the entire sentence is replaced, often using back-translation (Ribeiro et al., 2018) or paraphrasing models (Iyyer et al., 2018). This makes the attack indistinguishable to humans but can affect the model's prediction.

Combining diverse types of attacks can often improve their effectiveness. For instance, HotFlip (Ebrahimi et al., 2018) is a popular combination-based attack using both character and word level perturbations.

Most perturbations are context-aware, meaning that word or character-level attacks still consider the entire sentence, often leveraging the power of pre-trained language models to easily compare input similarity. The BAE attack (Garg and Ramakrishnan, 2020) is an example of a context-aware attack.

For all types of attacks, the result must be semantically similar to the original input, i.e., have the same meaning to a human observer. Therefore, for a given input $x_0 \in \mathcal{X}$, there exists a space $\mathcal{X}_{in}(x_0)$ defined by a set of constraints, such that all elements $z_0 \in \mathcal{X}_{in}(x_0)$ have the "same" meaning as x_0. TextAttack (Morris et al., 2020) provide a framework to easily define such constraints and generate perturbed inputs with the most common methods.

A.2 Robust Models by Design

Current efforts in defending NLP models against adversarial attacks have focused on developing robust models by design. One approach is to modify the architecture or the training process of the model. Adversarial training (Goodfellow et al., 2014) and data augmentation (Li et al., 2017; Belinkov and Bisk, 2017) are two common techniques used for this purpose.

A.2.1 Data Augmentation

One method for robust model training is data augmentation, which involves adding perturbed examples to the training data that satisfy the set of constraints, hopping to have a classifier that is more confident on attacks. However, generating enough samples to cover all constraints is computationally expensive, making this approach less effective than others.

A.2.2 Adversarial Training

Another method is adversarial training (Goodfellow et al., 2014), where the model is trained to optimize the saddle point problem:

$$\min_{\theta} \mathop{\mathbb{E}}_{(x_0,y)} \left[\max_{z_0 \in X_{in}(x_0)} \ell_\theta(z_0, y) \right]$$

Here, the inner maximization problem finds an adversarial perturbation $z_0 \in X_{in}(x_0)$ that maximizes the loss. The outer minimization problem updates the model parameters such that the adversarial risk of (A.2.2) is minimized. An interpolation weight of 0.5 can be used to balance between adversarial robustness and nominal accuracy.

A.2.3 Verified Robustness

Huang et al. (2019) go even further, modelling the space of constraints around an input as a simplex, and using Interval Bound Propagation to train a model robust to this entire space of constraints. This means that every transformation within this set is classified in the same way. This adds some strong theoretical guaranties which are strongly appreciated.

While these methods can improve the model's robustness, they are computationally expensive and require high computational power, as entire models need to be retrained from scratch. This limits the utilization of the vast amount of pre-trained models available.

A.3 Adjusting Pretrained Models

As a result, a class of defences emerged that focus on adjusting or correcting already pre-trained models, which is often less computationally intense and interesting in a multitude of cases.

Since most recent models rely on tokenizers and embeddings to process textual data, some researchers have proposed methods that place a correction model in front of the pre-trained downstream classifier, correcting the input tokens, or even imitating the embedding layer. For instance, Pruthi et al. (2019) add a word recognition model trained to recognize words corrupted by random adds, drops, swaps, and keyboard mistakes, correcting the inputs before feeding them to the original classifier. This method is quite elegant, as the

amount of training is quite but it significantly improves the performance of those model.

Another approach proposed by Jezabek and Singh (2022) is MockingBERT. This model aims to add a layer that imitates the classic embeddings of a given model but is trained to be robust using character-level information. They demonstrated that this model performs better on corrupted data without a significant loss of performance on in-distribution inputs. However, those solutions are often model dependent, meaning that a correction needs to be learn for each model, and can therefore not be practical in real use cases where we often want to compare multiple models.

In the paper for MockingBERT, the authors mention tokenizer-free models, such as CharacterBERT (Boukkouri et al., 2020), which adds a robust embedding method for unknown words (similar to ELMo), and CANINE (Clark et al., 2022), which propose a similar method. Saying that they could be more resilient to misspelling by default. However, they didn't mention additional ways to improve their robustness outside of adversarial training, as they are probably not resistant to more complex transformations (word or sentence level).

A.4 Attack Detection

In this section, the focus shifts to attack detection methods that aim to identify out-of-distribution inputs and exclude them. This is particularly useful when we want to filter out attacks, such as in the case of spam filtering. Some methods are unsupervised and require no training, while others are supervised and require adversarial examples. Both types of methods are relatively easy and cost-effective to implement.

One supervised method that we found interesting is the work of Xie et al. (2022), which improves attack detection performance by using additional text-related features in addition to Transformer embeddings. These features include the number of letters, non-ASCII characters, and the number of misplaced capitals. They also include language model features, such as input complexity and likelihood, and targeted model features, such as gradient values and activations. Linear and tree-based classifiers trained on these features achieve excellent performance. They also describe extensively how to generate the attack examples and how to build the dataset.

We also explored non-supervised methods based on Out-Of-Distribution (OOD) approaches. In particular, we focused on the method proposed by Sun et al. (2022), which uses KNN on the embeddings of the inputs is appealing, as it is a non-parametric approach that can be used in most context. This method looks at the distance between a new input and its nearest neighbors in the embedding space, and considers it out of distribution if the distance is larger than a precomputed threshold, using only clean data. This simple approach can perform well, and pretraining the embeddings using a contrastive loss further improves performance.

B Description of the Attacks Used

BAE (Garg and Ramakrishnan, 2020) (Backward translation with Adversarial Example) This black box attack generates adversarial examples for text classification models by replacing and inserting tokens in the original text using a BERT masked language model (MLM).

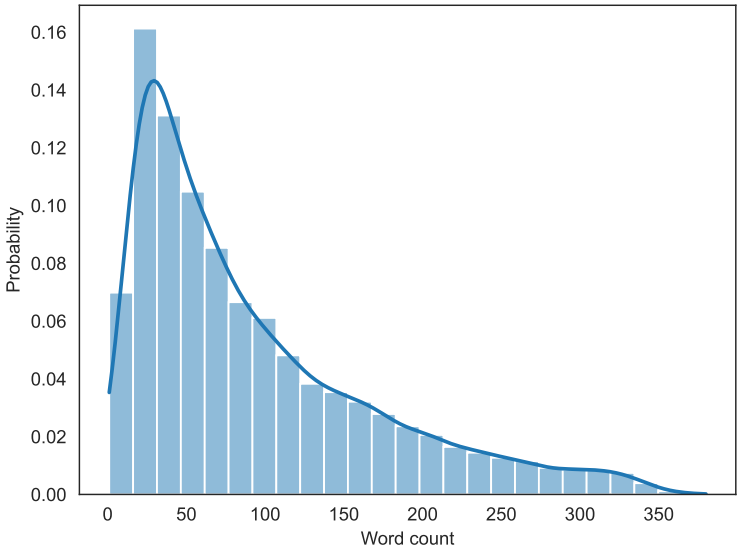
DeepWordBug (Gao et al., 2018) The algorithm uses some scoring strategies to identify the most important tokens that, if modified, cause the classifier to make an incorrect prediction. Character-level transformations are then applied to these tokens to change the original classification.

Input Reduction (Feng et al., 2018) This attack involves removing words from the original text to change the prediction.

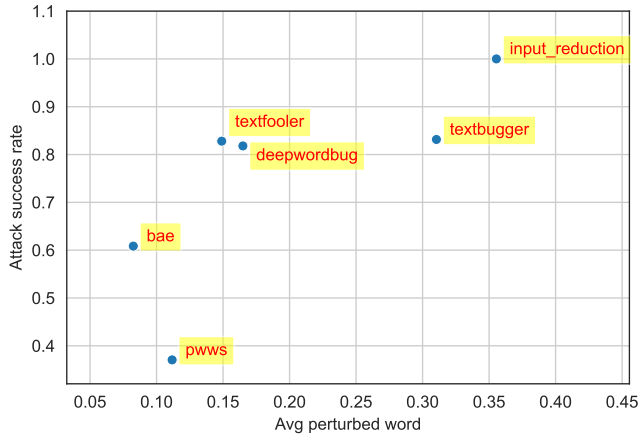
PWWS (Ren et al., 2019) (Probabilistic Word Weighting Scheme): This attack involves replacing words in the text with synonyms that are similar based on word saliency.

TextBugger (Li et al., 2018) In the white-box scenario, the attack finds important words by computing the Jacobian matrix of the classifier and selects an optimal perturbation from different kinds of perturbations. In the black-box scenario, the algorithm first finds important sentences and then uses a scoring function to identify important words to manipulate.

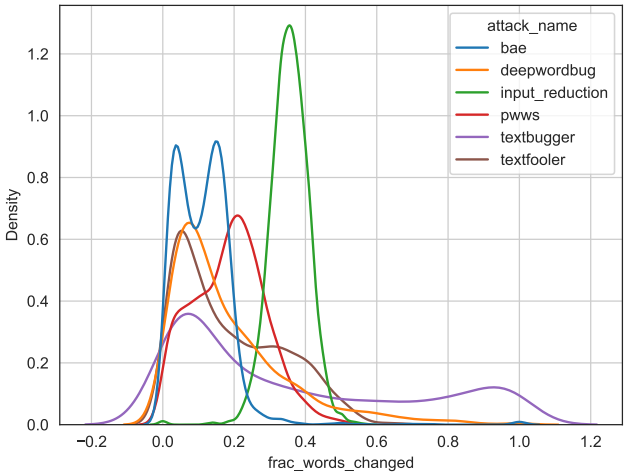
TextFooler (Jin et al., 2020) This attack involves replacing words in the text by first ranking word by their importance and then finding the best transformation in a pool of candidates.



Appendix 1: Histogram of the number of words in the Allociné corpus. We notice that most comments have less than 100 words, which is a good thing for transformer based language model, as they only have a context window of 512 in our case.



Appendix 2: Scatter plot of the number of average perturbed words by comment in relation to the attack success rate, by type of attacks. Attacks that have a low attack success rate need to perform more tries, resulting in more calculation. On the other hand, perturbing fewer words in less words is cheaper. We can see a positive correlation between those two features, which is logical.



Appendix 3: Fraction of words changed by attack type

Attack	Text	Label	Confidence	Status	Changed words
Original	Rien de bien intéressant dans ce film. C'est un sous Alien qui a très mal vieilli car comme tout film de plus de 20 ans, les effets spéciaux sont dépassés. Les acteurs sont pas mal (notons la présence de Robert Englund) ainsi que le scénario qui essaye quand même d'éviter le copier/coller d'Alien. Chose marrante, c'est James Cameron qui est le réal de la deuxième équipe.	Negative	100%		
BAE	coup de bien intéressant dans ce film. C'est un sous Alien qui a très mal vieilli car comme tout film de plus de 20 ans, les effets spéciaux sont mis. Les acteurs sont pas mal (notons la présence de Robert Englund) ainsi que le scénario qui essaye quand même d'éviter le copier/coller d'Alien. Chose marrante, c'est James Cameron qui est le réal de la deuxième équipe.	Positive	80%	Success	2.99%
DeepWordBug	gien qe bien intéressant dans ce film. C'est un sous Alien qui a très mal vieilli car comme tout film de plus de 20 ans, les effets spéciaux sont dépassvs. Les acteurs sont pas mal (notons la présence de Robert Englund) ainsi que le scénario qui essaye quand même d'éviter le copier/coller d'Alien. Chosse marrante, c'est James Cameron qui est le réal de la deuxième équipe.	Positive	71%	Success	5.97 %
Input Reduction	bien dans. C'est un a mal tout de, effets sont. acteurs (présence) ainsi que le scénario essaye quand même/coller d'Alien. marrante, Cameron qui est de deuxième.	Positive	99%	Success	40.30%
PWWS	RRien Delaware bien intéressant dans ce celluloid. C'est UN sous exotic qui a très mal vieilli car comme gasconade celluloid de plus de XX ANS, LE effets spéciaux sont dépassés. Les acteurs sont pas mal (notons La présence DE Robert Englund) ainsi que LE scénario qui essaye quand même d'éviter LE copier/coller d'Alien. choose marrante, c'est jam Cameron qui est le réal DE lah deuxième équipe.	Negative	100%	Failed	25.37 %
TextBugger	Rien de bien intéressant dans ce film. C'est un sous Alien qui a très mal vieilli car comme tout film de plus de 20 ans, les effets spéciaux comme dpassés. Les acteurs sont pas mal (notons la présence de Robert Englund) ainsi que le scénario comme essaye quand même d'éviter le copier/coller d'Alien. Chose marrante, c'est James Cameron qui est le réal de la deuxième équipe.	Positive	52%	Success	11.94%
TextFooler	Beaucoup de bien intéressant dans ce film. C'est un sous Alien qui a très mal vieilli car comme tout film de plus de 20 ans, les effets spéciaux sont dépassés. Les acteurs sont pas mal (notons la présence de Robert Englund) ainsi que le scénario comme essaye quand même d'éviter le copier/coller d'Alien. Chose marrante, c'est James Cameron qui est le réal de la deuxième équipe.	Positive	55%	Success	2.99%

Table 4: Examples of attacks on the Allociné dataset against CamemBERT using different types of attack methods. For the TextBugger attack, some of the changes appear identical to the original text. These are actually characters converted to the Cyrillic alphabet.

Attack	N samples	Success attacks	Failed attacks	Attack success rate	Accuracy under attack	Avg perturbed word	Avg num queries	Avg attack time
BAE	2000	1217	783	0.609	0.391	0.083	221.020	6.071
DeepWordBug	2000	1636	364	0.818	0.182	0.165	158.790	1.208
Input Reduction	2000	2000	0	1.000	0.000	0.355	154.220	0.889
PWWS	2000	741	1259	0.370	0.630	0.112	397.940	2.034
TextBugger	2000	1663	337	0.832	0.169	0.310	266.320	1.573
TextFooler	2000	1656	344	0.828	0.172	0.149	371.920	2.659

Table 5: Summary of the attacks. We ran 2000 tries for each of the 7 types of attack. Note that Input Reduction deletes words until the prediction switches, therefore, the resulting input is heavily modified.

C Supervised Classification Graphs and Tables

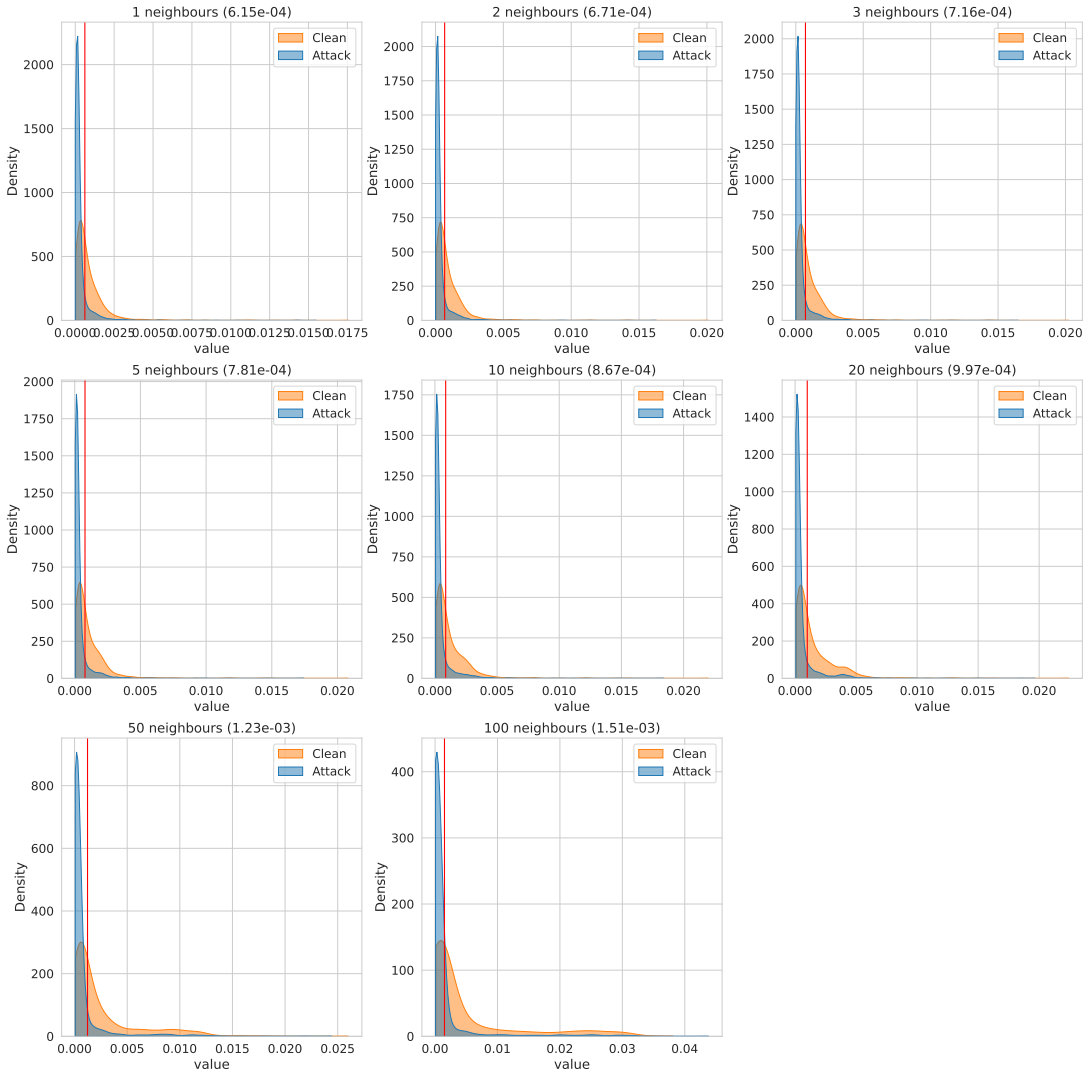
Embeddings	Features	Model	Accuracy	Recall	Precision	F1 score	ROC AUC
default	all	LR	0.951	0.938	0.964	0.951	0.951
fr+small	all	LR	0.934	0.921	0.945	0.933	0.934
fr+small	bert+tp+lm	LR	0.862	0.834	0.884	0.858	0.862
fr+small	bert+tp	LR	0.850	0.825	0.868	0.846	0.850
fr+small	bert	LR	0.848	0.821	0.868	0.844	0.848
default	bert+tp+lm	LR	0.767	0.699	0.810	0.750	0.767
default	bert+tp	LR	0.728	0.656	0.766	0.706	0.728
default	bert	LR	0.699	0.629	0.732	0.676	0.699

Table 6: Results of the logistic regressions in the different settings. `default` stands for the use of BERT and GPT2 (for lm), which are English models explaining the worst performance, however `all` also includes information on the target model, which is a distilCamemBERT. `fr+small` refers to the French distilled equivalent we used. We notice that `fr+small` performs better than `default` in most settings, except with all features.

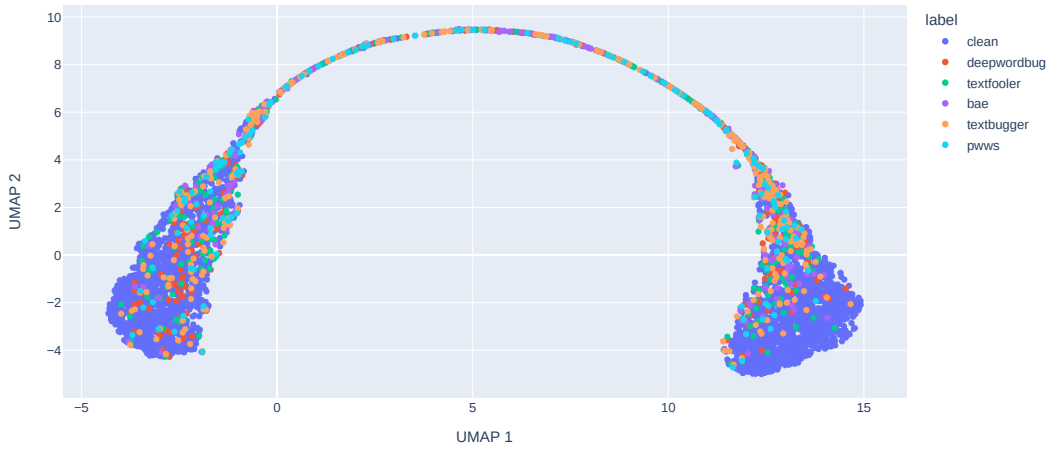
Embeddings	Features	Model	Accuracy	Recall	Precision	F1 score	ROC AUC
default	all	LGB	0.964	0.972	0.956	0.964	0.964
default	all	RF	0.954	0.970	0.940	0.955	0.954
default	all	LR	0.951	0.938	0.964	0.951	0.951
fr+small	all	LGB	0.946	0.959	0.934	0.946	0.946
fr+small	all	LR	0.934	0.921	0.945	0.933	0.934
fr+small	all	RF	0.926	0.962	0.897	0.929	0.926
fr+small	bert+tp+lm	LR	0.862	0.834	0.884	0.858	0.862
fr+small	bert+tp+lm	LGB	0.859	0.803	0.903	0.850	0.859
fr+small	bert+tp	LR	0.850	0.825	0.868	0.846	0.850
fr+small	bert	LR	0.848	0.821	0.868	0.844	0.848
fr+small	bert+tp	LGB	0.808	0.740	0.856	0.794	0.808
fr+small	bert+tp+lm	RF	0.806	0.729	0.862	0.790	0.806
fr+small	bert	LGB	0.798	0.727	0.847	0.782	0.798
default	bert+tp+lm	LGB	0.788	0.714	0.838	0.771	0.788
default	bert+tp+lm	RF	0.770	0.711	0.806	0.755	0.770
default	bert+tp+lm	LR	0.767	0.699	0.810	0.750	0.767
fr+small	bert	RF	0.756	0.724	0.773	0.748	0.756
fr+small	bert+tp	RF	0.751	0.715	0.770	0.742	0.751
default	bert+tp	LR	0.728	0.656	0.766	0.706	0.728
default	bert+tp	LGB	0.726	0.616	0.788	0.692	0.726
default	bert	LR	0.699	0.629	0.732	0.676	0.699
default	bert+tp	RF	0.669	0.563	0.714	0.630	0.669
default	bert	LGB	0.661	0.549	0.706	0.618	0.661
default	bert	RF	0.622	0.542	0.645	0.589	0.622

Table 7: Detailed results of our experiments, including logistic regression, Random Forest and LightGBM. We observe the same pattern as in the previous table.

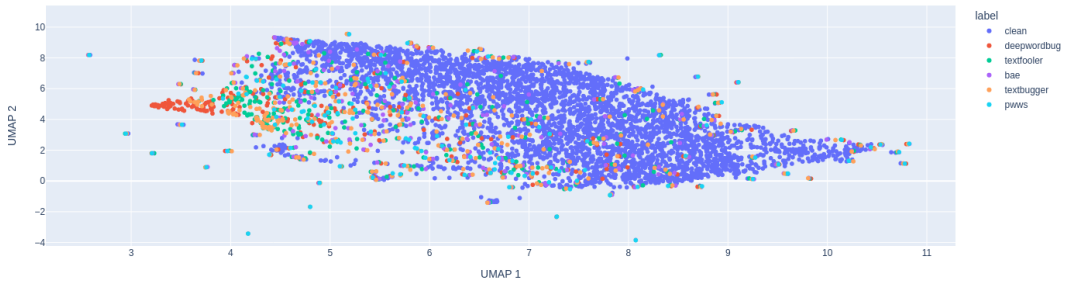
D OOD figures



Appendix 4: Distribution of the clean texts and attacks for different values of k. The blue kde plot represent the clean texts, the orange one the attacks, and the red lines represents the 90th percentile cutoff calculated on the train dataset.



Appendix 5: Scatter plot of the UMAP projection of the CamemBERT embeddings after finetuning with a contrastive loss, on the validation set, colored by attacks. We see that the attacks are on often on the border of the two classes, that are clearly separated.



Appendix 6: Scatter plot of the UMAP projection of the default CamemBERT embeddings, on the validation set, colored by attacks. We see that the attacks are also on the border, however classes are not separated in the embedding space.