Comparative Study of NLP Adversarial Attack Frameworks Against a BERT-Based Textual Entailment Model

Ernests Lavrinovics
University of Copenhagen
lfc161@alumni.ku.dk

Abstract

Deep learning models can provide incredible results although when impaired through adversarial attacks, they can quickly crumble. This work shows the effects of two fundamentally different adversarial attack frameworks namely *Universal Adversarial Triggers*(Wallace et al., 2019) and *TextFooler*(Jin et al., 2019), and their effects on DestilBERT model fine-tuned on Fever dataset for sequence classification problem. The result shows that both attacks are efficient and can make the model virtually unusable as the predictions are heavily skewed. The results do not quite correlate with previous research done within the field as they do not give much insight into how the model works, although this is attributed to potential flaws in the methodology when obtaining the results. Overall this work demonstrates the motivation for adversarial training to defend against adversarial attacks and shows also that both attack frameworks themselves can be further improved.

1 Introduction

Deep learning can yield impressive results and in certain cases even outperform humans[1]. Nevertheless, deep learning models are still vulnerable to perturbations that manipulate a model's output. These perturbations are called adversarial attacks (Kurakin et al., 2017) and earlier works originate from the image-processing domain where they have proven to be effective by introducing noise to an image (Szegedy et al., 2013), or even changing a single pixel (Su et al., 2019). Contrary to images, text tokens are a discrete media format although work has still been done to also perform adversarial attacks on NLP models. Notably by using gradient-guided searches for trigger tokens (Ebrahimi et al., 2018), and expanding on that work by using universal, input-agnostic tokens (Wallace et al., 2019), or by using search-based approaches (Jin et al., 2019). Comprehensive software frameworks have also been developed for adversarial attacks (Morris et al., 2020; Zeng et al., 2021a) that deliver multiple attack methodologies, model training and data augmentation in a production-grade software package. The development of such attack frameworks democratizes and improves the quality of the trained models as they allow research groups to perform analysis using different attack methodologies that each may uncover otherwise unnoticed vulnerabilities of their NLP models, this inspires a new model training paradigm called adversarial training (Kurakin et al., 2017).

It is important to also consider the security implications of adversarial attacks and, generally, the incorrect output of machine learning models. There are cases of bypassing spam filters (Biggio et al., 2013), manipulating fake news detection (Zhou et al., 2019) or even people being taken to jail for incorrect machine translation (He et al., 2020).

Natural language inference (NLI) is a task in natural language processing (NLP) that involves determining whether a given hypothesis h can be inferred from a given premise p, as depicted in the following example (Chen et al., 2017), the premise entails the hypothesis.

p: *Several airlines polled saw costs grow more than expected, even after adjusting for inflation.*

h: *Some of the companies in the poll reported cost increases.*

Many well-established NLI datasets (Thorne et al., 2018; Bowman et al., 2015; Williams et al., 2018; Aly et al., 2021) allow for data-driven approaches for performing model training for NLI. Similarly, leveraging language models (Devlin et al., 2018a) and modern ML-Ops frameworks (Wolf et al., 2019), allows the possibility to fine-tune them on downstream tasks (Wolf et al., 2019;

[1]https://rajpurkar.github.io/SQuAD-explorer/

Jiang and de Marneffe, 2019; Liu et al., 2021; Atanasova et al., 2020; Lee and Hsiang, 2020).

The goal of this project is to perform an analysis of a monolingual *language inference* model by using adversarial attacks and exploring the model's vulnerabilities and weaknesses.

2 Background

2.1 Language Inference

The idea of natural language inference or *textual entailment* is to reason about the directional relationship between a piece of text and a hypothesis. For humans, this may come intuitively as we make conclusions based on our world-knowledge and underlying reasoning process often called as *common sense* (Storks et al., 2019). For machines, commonsense reasoning for a long time has been considered a non-trivial problem, although recent advances in language modelling have shown remarkable results for logical reasoning by closed-source models (Susnjak, 2022) (Qiao et al., 2022).

Normally textual entailment is framed as a classification problem where a model predicts one of the multiple labels that describe the directional relationship. All modern language inference training pipelines use deep learning and their usual components are shown in Figure 1.

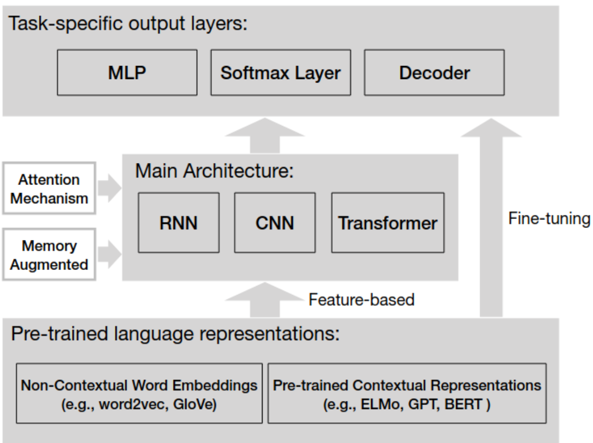

Figure 1: Common components in neural-based model training pipelines (Storks et al., 2019)

Word representation via numerical vectors is a core part of performing machine learning. Embeddings such as Word2Vec(Mikolov et al., 2013) and GloVe(Pennington et al., 2014) are context-independent meaning that they have fixed values for certain words. This is important to note because languages are non-linear and have ambiguities, therefore context-dependent representations have been developed to address this problem (Devlin et al., 2018b). These representations can be used as features or fine-tuned for downstream tasks (Storks et al., 2019).

Neural network architectures are designed for different downstream applications. For language inference, it is important to capture long-term dependencies which can be done with transformers (Vaswani et al., 2017) and LSTMs (Hochreiter and Schmidhuber, 1997) (Storks et al., 2019).

Textual entailment in itself can also be used for downstream tasks, for example fact-checking by predicting whether a claim can be supported or denied given a list of sources. Research suggests that the average time of fact-checking a typical article is about a day (Hassan et al., 2015) and fake news poses threats to the integrity of journalism, creates turmoils in the political world (Wang, 2017) and even is attributed to mass shootings[2]. So all things considered, the amount of information and the speed at which it can spread through the internet creates a need for robust language inference. A more detailed overview of fact-checking is available in two recent surveys (Zeng et al., 2021b; Guo et al., 2022).

2.2 Datasets

A well-established, highly used and studied dataset SNLI (Bowman et al., 2015) contains about 500k sentence pairs as datapoints that are crowd-sourced. The dataset was generated by asking Amazon Mechanical Turk workers to write a hypothesis for *contradiction*, *entailing* or *neutral* labels. During the generation of the dataset, the authors also ensured a validation step in which four other workers had to agree on the correct labels for a given hypothesis.

Another popular, crowdsourced dataset MultiNLI (Williams et al., 2018) contains about 430k sentence pairs of multiple genres (fiction, travel, government, others).

Similarly, dataset Fever (Thorne et al., 2018) is also a collection of premise-hypothesis-label triplets that have been crowd-sourced by human workers augmenting sentences derived from Wikipedia and subsequently verified.

All three of the aforementioned datasets have been critiqued for containing spurious correlations between bi-grams in the hypothesis and their cor-

[2]http://www.nytimes.com/2016/12/05/business/media/comet-ping-pong-pizza-shooting-fake-news-consequences.html

responding labels and suggest that models trained on these datasets do not perform genuine reasoning based on provided evidence (Schuster et al., 2019; Gururangan et al., 2018; Tan et al., 2019; Wallace et al., 2019). This suggests a general pattern of crowd-sourcing creating annotation artifacts.

2.3 Adversarial Attacks

Adversarial attacks are an active research field within the NLP community with two recent surveys providing a detailed overview of the field (Roth et al., 2021; Qiu et al., 2022).

Gradient-based adversarial attacks have been popular, with earlier work (Ebrahimi et al., 2018) using an approach to estimate how much a result changes, when certain triggers are appended to a sequence, this is called the *Hotflip* algorithm. Further work *Universal Adversarial Triggers (UAT)*(Wallace et al., 2019) builds upon this knowledge to generate input-agnostic triggers that are *one-solution fits all* and manipulate the output irrespective of the input sequence. The UAT framework also assumes white-box access to the victim model for performing the gradient-based search. The task of a universal attack is summarized as follows:

$$\arg\min_{t_{adv}} E_{t\sim T}\left[L(\tilde{y}, f(t_{adv}; t))\right]$$

Where a task-dependant loss-function L is minimized against a target-label $\tilde{y}$ and output of a model f, given a trigger and input $(t_{adv}; t)$ over the whole dataset $E_{t\sim T}$.

While the UAT framework is powerful from the perspective of its universal setting, it may not be as subtle when processed input is reviewed by humans. Meaning that semantics break by adding non-logical or grammatically questionable triggers to the input. The problem of retaining semantics is approached in the *TextFooler* publications (Jin et al., 2019) by estimating the most important words within the input and substituting them with context-independent synonyms until the model's output changes. The word importance is determined by the amount of change in the prediction that it makes when it is deleted during the search-stage. Contrary to the UAT framework, TextFooler does not assume white-box access to the victim model. Listing 1 depicts an example of *TextFooler* modifications.

Before
P: Bermuda Triangle is in the western part of the **Himalayas**
H: The Bermuda triangle , also known as the Devil ' s Triangle , is a loosely - defined region in the western part of the north atlantic ocean , where a number of aircraft and ships are said to have disappeared under mysterious circumstances.
Res: **Refutes (63.13%)**

After
P: bermuda triangle is in the western part of the **himalaya**.
H: the bermuda triangle , also known as the devil ' s triangle , is a loosely - defined region in the western part of the north atlantic ocean , where a number of aircraft and ships are said to have disappeared under mysterious circumstances .
Res: **Not Enough Info (79.66%)**

Listing 1: Example of TextFooler manipulating the input. Underlined token represents the TextFooler modifications; *Premise(P), Hypothesis(H), Result(Res)*

2.4 Adversarial Training

The idea to employ adversarial perturbations as part of the training dataset is called *adversarial training* (Madry et al., 2018). In recent years, this technique has been employed in NLP, image processing and audio processing, and has proven to improve the robustness of the models (Jin et al., 2019; Morris et al., 2020). The task can be summarized as

$$\arg\min_{\theta} E_{(x,y)\sim D}\left[L(\theta, x, y) + \alpha L(\theta, A(\theta, x, y), y)\right]$$

Where $L(\theta, x, y)$ represents the loss-function for a model, text x and label y. $A(\theta, x, y)$ represents the adversarial attack that produces x_{adv} and α can be used to weigh the adversarial example (Yoo and Qi, 2021).

3 Methodology

3.1 Dataset

The dataset used in the experiment was Fever (Thorne et al., 2018) in a modified format[3] with hypothesis-premise-label triplets, thus making the

[3] https://huggingface.co/datasets/pietrolesci/nli_fever

Class	Unbal (Train)	Bal (Train)	Bal (Dev)
Supports	∼123k	7.5k	450
NEI	∼35k	7.5k	450
Refutes	∼49k	7.5k	450
Total	∼208k	22.5k	1.35k

Table 1: Overview of Fever dataset before and after balancing. *Note: NEI refers to "Not Enough Information"*

data retrieval very easy. The preference for Fever is motivated due to it being a well-established, simple and popular dataset for NLI tasks, with general-domain data points.

The training split contains over 200k data points and is not balanced, hence balancing was performed by down-sampling each class for both training and development. Table 1 summarizes the Fever datapoint counts before and after balancing, the development set was drastically downsampled to 1.35k datapoints across all labels, mainly due to the adversarial attacks requiring a lot of computational power and hence running them through a larger dataset becomes infeasible given the computational resources available. See Appendix A.1 for the training script which also includes the dataset balancing function.

3.2 Model Fine-Tuning

A subset of a BERT (Devlin et al., 2018b) model *DistilBERT*[4] was fine-tuned on the previously described dataset for a sequence classification task. The motivation for using the DistilBERT was due to its lightweight size and saving computational resources when compared to the base or large model. HuggingFace MLOps framework was used to perform the training with the hyperparameters shown in Table 2.

LearnRate	BatchSize	Epochs	DropoutProb
0.001	64	10	0.25

Table 2: DestilBERT fine-tuning hyperparameters

3.3 Adversarial Attack Framework

The OpenAttack (Zeng et al., 2021a) was used as it combines many different attack methodologies in a single package and it integrates seamlessly with the HuggingFace MLOps platform. The framework had issues with out-of-date SSL certificates and also did not provide means to control hyper-parameters from user-side code for the *Universal Adversarial Attack*, hence modifications were to the framework were made[5]. UAT attack was run using a beam-search of size **3** over 2 epochs.

Two attacks were used, primarily Universal Adversarial Triggers, and TextFooler (Jin et al., 2019) due to them both being distinctively different with chances of exposing different kinds of vulnerabilities.

4 Results

In order to provide a test-bed for the adversarial attacks and make the results more prominent, a model needs to perform better than a *random-guess* probability. Further sections show the baseline model results, as well as the results of successful impairments on the model by deploying the adversarial attacks.

4.1 Model Fine-Tuning

The baseline model was trained and a model card was published on HuggingFace[6]. After fine-tuning on the training split described in Table 1 over 10 epochs, the baseline results of the development split are summarized in Table 3 per each label. The confusion matrix is presented in Figure 2, it shows the majority of labels being correctly predicted, with a higher confusion between *Not Enough Info* label and *Refutes* or *Support* labels. Confusion between *Support* and *Refutes* is minimal.

Label	precision	recall	f1-score
Supports	0.77	0.79	0.78
NEI	0.60	0.73	0.66
Refutes	0.80	0.60	0.68

Table 3: Overview of baseline model results. *Note: NEI refers to "Not Enough Information"*

4.2 Attack: Universal Adversarial Triggers (UAT)

When running the *OpenAttack* framework over the development dataset, the UAT attack estimated a trigger ***emia nothing segments*** although after manually removing the *emia* token the results were even more impaired disregarding that there's one token less in the trigger. So the final trigger used for obtaining the results is ***nothing segments***.

[4] https://huggingface.co/distilbert-base-uncased

[5] https://github.com/ernlavr/OpenAttack

[6] https://huggingface.co/ernlavr/destilbert_uncased_fever_nli

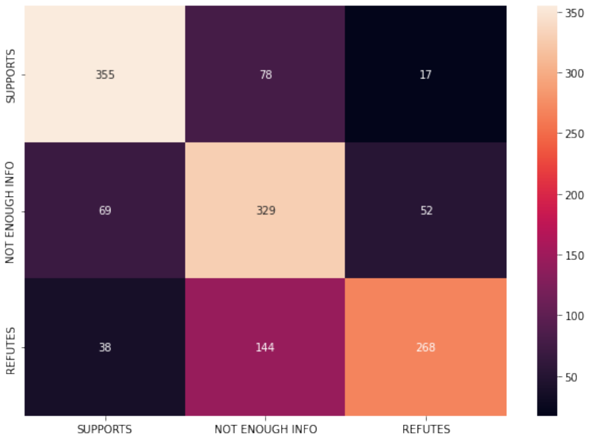

Figure 2: Confusion matrix of the baseline model

Figure 3 summarizes the confusion matrix and Table 4 depicts the F1, recall and precision results. The results show that the trigger successfully and heavily skews all predictions towards *Refutes* label, it has nearly removed the possibility of estimating *Supports* label, therefore the model is heavily impaired. Listing 2 depicts an example how the trigger changing the output of the classifier.

Without trigger
P: The Endless River, which was released in 2014 is Pink Floyd's final studio album.
H: Pink Floyd . The final Pink Floyd studio album , The Endless River (2014) , was recorded without Waters and based largely on unreleased material . The Endless River is the fifteenth and final studio album by the English rock band Pink Floyd .
Res: **Supports**

With trigger
P: ***nothing segments*** The Endless River, which was released in 2014 is Pink Floyd's final studio album.
H: Pink Floyd . The final Pink Floyd studio album , The Endless River (2014) , was recorded without Waters and based largely on unreleased material . The Endless River is the fifteenth and final studio album by the English rock band Pink Floyd .
Res: **Supports -> Refutes**

Listing 2: Example of before-after applying the trigger *"nothing segments"*; *Premise(P), Hypothesis(H), Result(Res)*

Label	precision	recall	f1-score
Supports	0.71	0.01	0.02
NEI	0.67	0.32	0.43
Refutes	0.35	0.88	0.50

Table 4: Overview of UAT results with *nothing segments* trigger applied to inputs

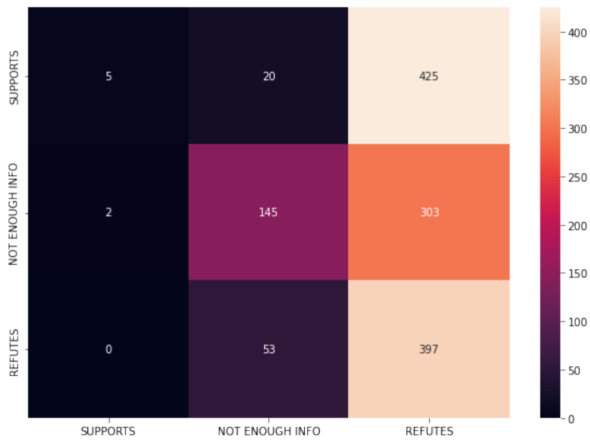

Figure 3: Confusion matrix for *"nothing segments"* trigger

4.3 Attack: TextFooler

Results for the TextFooler attack are presented in Table 5 and Figure 4. In this case, the attack manipulates the model to output more *neutral* labels, still significantly reducing its performance. Quantitative overview examples are present in Appendix A.4

Label	precision	recall	f1-score
Supports	0.11	0.04	0.06
NEI	0.27	0.52	0.35
Refutes	0.28	0.20	0.23

Table 5: Overview of TextFooler results

4.4 PMI Score Top Tokens

Pointwise Mutual Information was computed between unigrams and labels to investigate how certain words overlap with labels as per (Gururangan et al., 2018) but without the *Add-100* smoothing. A summary of the highest PMI scores per each token from the training set is summarized in Table 6 and a more detailed overview is available in Appendix A.3.

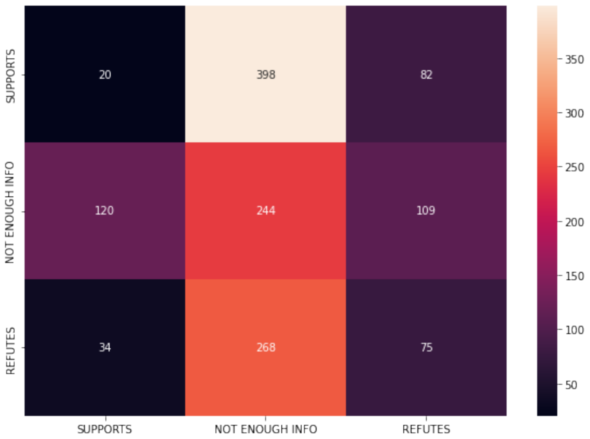

Figure 4: Confusion matrix for the outputs of *TextFooler* attack

The formula for the PMI calculation is shown below:

$PMI(X,Y) = log_2 \frac{P(X,Y)}{P(X)P(Y)}$

Where:

X *: Unigram*

Y *: Label*

$P(X,Y)$ *: Joint-probability X-Y*

$P(X)$ *: Marginal-probability X*

$P(Y)$ *: Marginal-probability Y*

Supports	NEI	Refutes
mundo	##rud	##skaya
speechless	recipe	fairchild
essen	cried	1749
cardinal	theologian	1833
ethical	##test	ngos
projected	deacon	##oun
deteriorating	exceeded	##eous
headline	:	abby
psyche	sonora	maximilian
##itors	breeders	relation
supportive	##mx	##ologies
straightforward	##cross	goofy
englishman	bikes	rumors
modes	adventist	speculation
skins	stalled	gibbs

Table 6: Highest scoring PMIs per label

4.5 Word Substitutions

For the *TextFooler* an overview was made of the synonyms and their count, that the tokens from the original were substituted to. Due to a code bug during result generation, the information could not be fully retrieved from a serialized data structure created during result-generation, hence because of varied sequence lengths and punctuation that did not allow for a 1:1 mapping of the original to the adversarial text, also all of the data points were not saved. Full-raw data is available in the supplement to this submission. Nevertheless, approximately 500 out of 1350 examples were usable and an overview of the word substitution frequencies are in Table 7.

Supports	NEI	Refutes
('get', 6)	('record', 12)	('celluloid', 6)
('have', 6)	('celluloid', 11)	('motion', 4)
('moving', 5)	('comport', 10)	('individual', 4)
('playscript', 4)	('cinema', 7)	('play', 4)
('realm', 4)	('ground', 7)	('get', 3)
('record', 3)	('set', 5)	('house', 3)
('lead', 3)	('play', 5)	('innate', 3)
('let', 3)	('pic', 5)	('corner', 3)
('hold', 3)	('pretend', 5)	('moving', 2)
('turn', 3)	('decease', 5)	('robert', 2)

Table 7: SubstitutionToken-Frequency pairs that were used to obtain the target labels

5 Analysis and Conclusions

Overall the adversarial attacks were able to successfully impair the model making it virtually unusable for production in both cases of the attack. The presented results partially support the claim that the model learns spurious correlations; at least as far as the PMI calculations and token-frequency pairs go, although this is attributed to potential mistakes done during the result generation. The universal trigger *nothing segments* performed remarkably well, as shown in Figure 3 although neither of the tokens were present in the top 10-25 highest PMI scores. A different token sequence *recipe cried summarized* also performed very well, as shown in Appendix A.2 and this was taken directly from Table 6.

The *TextFooler* attack proved to skew the results heavily towards *N.E.I.* label. While most of the manipulations appeared to have preserved the semantics, some examples had also skewed semantics showing that the framework can still be further improved, see Appendix A.4. The token-frequency pairs do not directly give much information of how the model works as they do not correlate with the highest-PMI tokens in Table 6. The flawed semantics is a good example of how the framework makes use of non-contextual embeddings, even though words themselves may be synonyms, they still do not make sense within the context they are in, see Appendix A.4. Overall this shows that a vanilla DestilBERT fine-tuned on a classification task is highly vulnerable to adversarial attacks, therefore this demonstrates the need and motivation for defense against such attacks, potentially with adversarial training or any other data pre-processing or postprocessing.

References

Rami Aly, Zhijiang Guo, Michael Sejr Schlichtkrull, James Thorne, Andreas Vlachos, Christos Christodoulopoulos, Oana Cocarascu, and Arpit Mittal. 2021. The fact extraction and VERification over unstructured and structured information (FEVEROUS) shared task. In *Proceedings of the Fourth Workshop on Fact Extraction and VERification (FEVER)*, pages 1–13, Dominican Republic. Association for Computational Linguistics.

Pepa Atanasova, Dustin Wright, and Isabelle Augenstein. 2020. Generating label cohesive and well-formed adversarial claims. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3168–3177, Online. Association for Computational Linguistics.

Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. 2013. Evasion attacks against machine learning at test time. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 387–402. Springer.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced LSTM for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1657–1668, Vancouver, Canada. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018a. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

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

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.

Zhijiang Guo, Michael Schlichtkrull, and Andreas Vlachos. 2022. A Survey on Automated Fact-Checking. *Transactions of the Association for Computational Linguistics*, 10:178–206.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A. Smith. 2018. Annotation artifacts in natural language inference data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 107–112, New Orleans, Louisiana. Association for Computational Linguistics.

Naeemul Hassan, Bill Adair, James Hamilton, Chengkai Li, Mark Tremayne, Jun Yang, and Cong Yu. 2015. The quest to automate fact-checking. *Proceedings of the 2015 Computation + Journalism Symposium*.

Pinjia He, Clara Meister, and Zhendong Su. 2020. Structure-invariant testing for machine translation. In *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*, pages 961–973. IEEE.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Nanjiang Jiang and Marie-Catherine de Marneffe. 2019. Evaluating BERT for natural language inference: A case study on the CommitmentBank. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6086–6091, Hong Kong, China. Association for Computational Linguistics.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2019. Is BERT really robust? natural language attack on text classification and entailment. *CoRR*, abs/1907.11932.

Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. 2017. Adversarial machine learning at scale. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Jieh-Sheng Lee and Jieh Hsiang. 2020. Patent classification by fine-tuning bert language model. *World Patent Information*, 61:101965.

Hanmeng Liu, Leyang Cui, Jian Liu, and Yue Zhang. 2021. Natural language inference in context-investigating contextual reasoning over long texts. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 13388–13396.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

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*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Shuofei Qiao, Yixin Ou, Ningyu Zhang, Xiang Chen, Yunzhi Yao, Shumin Deng, Chuanqi Tan, Fei Huang, and Huajun Chen. 2022. Reasoning with language model prompting: A survey. *arXiv preprint arXiv:2212.09597*.

Shilin Qiu, Qihe Liu, Shijie Zhou, and Wen Huang. 2022. Adversarial attack and defense technologies in natural language processing: A survey. *Neurocomputing*, 492:278–307.

Tom Roth, Yansong Gao, Alsharif Abuadbba, Surya Nepal, and Wei Liu. 2021. Token-modification adversarial attacks for natural language processing: A survey. *CoRR*, abs/2103.00676.

Tal Schuster, Darsh Shah, Yun Jie Serene Yeo, Daniel Roberto Filizzola Ortiz, Enrico Santus, and Regina Barzilay. 2019. Towards debiasing fact verification models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3419–3425, Hong Kong, China. Association for Computational Linguistics.

Shane Storks, Qiaozi Gao, and Joyce Y Chai. 2019. Recent advances in natural language inference: A survey of benchmarks, resources, and approaches. *arXiv preprint arXiv:1904.01172*.

Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. 2019. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 23(5):828–841.

Teo Susnjak. 2022. Chatgpt: The end of online exam integrity? *arXiv preprint arXiv:2212.09292*.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

Shawn Tan, Yikang Shen, Chin-wei Huang, and Aaron Courville. 2019. Investigating biases in textual entailment datasets. *arXiv preprint arXiv:1906.09635*.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a large-scale dataset for fact extraction and VERification. 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 809–819, New Orleans, Louisiana. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing nlp. *arXiv preprint arXiv:1908.07125*.

William Yang Wang. 2017. "liar, liar pants on fire": A new benchmark dataset for fake news detection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 422–426, Vancouver, Canada. Association for Computational Linguistics.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. 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 1112–1122. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, abs/1910.03771.

Jin Yong Yoo and Yanjun Qi. 2021. Towards improving adversarial training of nlp models. *arXiv preprint arXiv:2109.00544*.

Guoyang Zeng, Fanchao Qi, Qianrui Zhou, Tingji Zhang, Zixian Ma, Bairu Hou, Yuan Zang, Zhiyuan Liu, and Maosong Sun. 2021a. OpenAttack: An open-source textual adversarial attack toolkit. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 363–371, Online. Association for Computational Linguistics.

Xia Zeng, Amani S. Abumansour, and Arkaitz Zubiaga. 2021b. Automated fact-checking: A survey. *Language and Linguistics Compass*, 15(10):e12438.

Zhixuan Zhou, Huankang Guan, Meghana Moorthy Bhat, and Justin Hsu. 2019. Fake news detection via NLP is vulnerable to adversarial attacks. *CoRR*, abs/1901.09657.

A Appendix

A.1 Baseline Model Training

To be ran as a Jupyter notebook

```python
# -*- coding: utf-8 -*-
"""FTML_model_train.ipynb

Automatically generated by Colaboratory.

"""

!pip install pytorch-crf
!pip install datasets
!pip install sklearn
!pip install transformers
!pip install evaluate
!pip install huggingface_hub

# Commented out IPython magic to ensure Python compatibility.
# %reload_ext autoreload
# %autoreload 2
# %matplotlib inline

import io
from math import log
import os
import pickle
from numpy import array
from numpy import argmax
import torch
import random
from math import log
from numpy import array
from numpy import argmax
import numpy as np
from torch.utils.data import Dataset, DataLoader
from torch import nn
from torch.optim import Adam
from torchcrf import CRF
from torch.optim.lr_scheduler import ExponentialLR, CyclicLR
from typing import List, Tuple, AnyStr
from tqdm import tqdm
from sklearn.metrics import precision_recall_fscore_support
import matplotlib.pyplot as plt
from copy import deepcopy
from datasets import load_dataset, load_metric
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
import transformers
from transformers import AutoTokenizer, AdamW
from transformers import TrainingArguments, Trainer
import transformers
import evaluate
from transformers import (
    AutoConfig,
    AutoModelForTokenClassification,
    AutoTokenizer,
    DataCollatorForTokenClassification,
    HfArgumentParser,
    PretrainedConfig,
    PreTrainedTokenizerFast,
    Trainer,
    TrainingArguments,
    set_seed,
)
import pandas as pd
import seaborn as sn
```

```python
import matplotlib.pyplot as plt
from datasets import DatasetDict
from dataclasses import dataclass
import random
import time
import datetime
import sys
import math


def enforce_reproducibility(seed=42):
    torch.manual_seed(seed)
    torch.cuda.manual_seed_all(seed)
    torch.backends.cudnn.deterministic = True
    torch.backends.cudnn.benchmark = False
    random.seed(seed)
    np.random.seed(seed)
enforce_reproducibility()

HG_MODEL_NAME = "distilbert-base-uncased"
HG_DATASET = "pietrolesci/nli_fever"
NUM_LABELS = 3
os.environ["WANDB_DISABLED"] = "true"

@dataclass
class DataPoint:
    """Class that represents a datapoint"""
    cntTkn: list
    hypTkn: list  # answer tokenizer
    lbl: str  # raw full text


def loadModel():
    return transformers \
      .AutoModelForSequenceClassification \
      .from_pretrained(HG_MODEL_NAME, num_labels=NUM_LABELS) \
      .to(device)


def loadTokenizer():
    return AutoTokenizer.from_pretrained(HG_MODEL_NAME)


def loadFeverDataset():
    return load_dataset(HG_DATASET)

def appendToLogFile(text):
    """Appends text to a log file"""
    with open(logName, "a") as f:
        timeStamp = datetime.datetime.now().time()
        f.write(f"{timeStamp}: {text}")
        # Check if text string ends with a new line, if not then add one. Beware of empty text str
        if text and text[-1] != "\n":
            f.write("\n")

def printAndLog(text):
    """Prints and logs text"""
    print(text)
    appendToLogFile(text)

def idxToLabels():
    return {0: "SUPPORTS", 1: "NOT ENOUGH INFO", 2: "REFUTES"}

def getLabels():
    return {"SUPPORTS": 0, "NOT ENOUGH INFO": 1, "REFUTES": 2}

def balance_dataset(ds, numSamples=-1):
    """
    Balances the dataset by removing samples from the majority class
    :param ds: The dataset
```

```python
    :param numSamples: The number of samples to keep
    :return: The balanced dataset
    """
    # Get the number of samples for each label
    dss = ds[:]
    labels = dss["fever_gold_label"]
    if numSamples == -1:
        numSamples = len(labels)
        unique, counts = np.unique(labels, return_counts=True)
        counts = np.roll(counts, 1)
        unique = np.roll(unique, 1)
        numSamples = min(counts)

    # get indices of ds elements where ds['label'] is 0
    arr = dss['label']
    arr = np.array(arr)
    indicesSup = np.where(arr == 0)[0][:numSamples]
    indicesNei = np.where(arr == 1)[0][:numSamples]
    indicesRef = np.where(arr == 2)[0][:numSamples]

    # combine the indices
    indices = np.sort((np.concatenate((indicesSup, indicesNei, indicesRef))))
    indices = indices.tolist()
    # get a subset of the dataset
    return indices


def compute_metrics(eval_pred):
    logits, labels = eval_pred
    predictions = np.argmax(logits, axis=-1)
    return metric.compute(predictions=predictions, references=labels, average="macro")

def tokenize_function(examples):
    textPairs = zip(examples["premise"], examples["hypothesis"])
    textPairs = [pair[0] + " " + pair[1] for pair in textPairs]
    out = tokenizer(textPairs, padding="max_length",  return_tensors="pt", truncation=True).to(dev
    out.data["label"] = examples["label"]
    return out

# Setup logging
timeStamp = time.strftime("%Y%m%d-%H%M%S")
currFileLoc = ""
logName = os.path.join(currFileLoc, f"16_log_{timeStamp}.txt")
with open(logName, 'w') as f:
    f.write("")

appendToLogFile("Start of log file \n")
appendToLogFile(f"Using CUDA: {torch.cuda.is_available()} \n")

# Set constants
DEBUG = False
device = (torch.device("cpu"), torch.device("cuda"))[torch.cuda.is_available()]

# Load and parse the dataset
ds = loadFeverDataset()
model = loadModel()
tokenizer = loadTokenizer()

# Map the dataset to the tokenizer
trainIdx = balance_dataset(ds['train'], 7500)
devIdx = balance_dataset(ds['dev'], 1500)
tds = ds.map(tokenize_function, batched=True)

# Create subsets of balanced dataset
trainSet = torch.utils.data.Subset(tds['train'], trainIdx)
devSet = torch.utils.data.Subset(tds['dev'], devIdx)

# Training hyperparameters
torch.cuda.empty_cache()
metric = load_metric('f1')
```

```
dropout_prob = 0.25
epochs = 10
batch_size = 64
lr = 0.0001
n_epochs = 1
training_args = TrainingArguments(
                  "destilbert_uncased_fever_nli",
                  evaluation_strategy = "epoch",
                  save_strategy = "epoch",
                  learning_rate=lr,
                  per_device_train_batch_size=batch_size,
                  per_device_eval_batch_size=batch_size,
                  num_train_epochs=epochs,
                  weight_decay=0.01,
                  load_best_model_at_end=True,
                  metric_for_best_model="f1",
                  push_to_hub=True,
                  push_to_hub_model_id="destilbert_uncased_fever_nli"
                )

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=trainSet,
    eval_dataset=devSet,
    tokenizer=tokenizer,
    compute_metrics=compute_metrics
)
trainer.train()

trainer.push_to_hub()

# evaluate the model
preds = []
gt = []
model.eval()
with torch.no_grad():
  for i, input in enumerate(tqdm(devSet)):
      if input['label'] == -1:
          continue

      premise = input['premise']
      hypothesis = input['hypothesis']
      label = input['label']

      # Tokenize the premise and hypothesis
      tokenizedSequence = tokenizer.encode_plus(premise, hypothesis,
                                                  max_length=512,
                                                  return_token_type_ids=True,
                                                  truncation=True)

      input_ids = torch.tensor(tokenizedSequence['input_ids']).long().unsqueeze(0).to(device)
      token_type_ids = torch.Tensor(tokenizedSequence['token_type_ids']).long().unsqueeze(0).to(de
      attention_mask = torch.Tensor(tokenizedSequence['attention_mask']).long().unsqueeze(0).to(de

      outputs = model(input_ids,
                      attention_mask=attention_mask,
                      labels=None)

      predicted_probability = torch.softmax(outputs[0], dim=1)[0].tolist()

      preds.append(np.argmax(predicted_probability))
      gt.append(label)

print(HG_MODEL_NAME)
print(getLabels())
print(classification_report(gt, preds))

cm = confusion_matrix(gt, preds)
```

```
classes = [*getLabels()]
df_cfm = pd.DataFrame(cm, index = classes, columns = classes)
plt.figure(figsize = (10,7))
cfm_plot = sn.heatmap(df_cfm, annot=True, fmt='g')
```

A.2 Alternative UAT triggers

An alternative trigger ***recipe cried summarized*** was noted during a qualitative inspection of the results, that destroys nearly all predictions besides NEI. The trigger words directly correlate with three most popular triggers from the PMI score Table 8

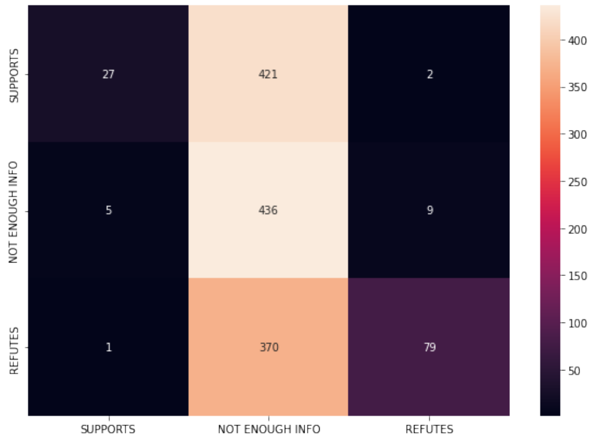

Figure 5: Trigger: *recipe cried summarized*

A.3 Non-normalized PMI Score

Supports	PMI	NEI	PMI	Refutes	PMI
countdown	1.4059	##rud	1.9776	##skaya	1.4395
amor	1.4059	recipe	1.9776	fairchild	1.4395
dreaming	1.4059	cried	1.9776	1749	1.4395
accomplish	1.4059	summarized	1.9776	1833	1.4395
mundo	1.4059	theologian	1.9776	ngos	1.4395
speechless	1.4059	##test	1.9776	##oun	1.4395
essen	1.4059	educator	1.9776	##eous	1.4395
marines	1.4059	deacon	1.9776	abby	1.4395
cardinal	1.4059	exceeded	1.9776	maximilian	1.4395
ethical	1.4059		1.9776	relation	1.4395
projected	1.4059	sonora	1.9776	##ologies	1.4395
deteriorating	1.4059	breeders	1.9776	goofy	1.4395
headline	1.4059	##mx	1.9776	rumors	1.4395
sideways	1.4059	##cross	1.9776	speculation	1.4395
rooted	1.4059	bikes	1.9776	gibbs	1.4395
		bikes	1.9776		

Table 8: Unnormalized PMI scores of the training split

A.4 Examples from TextFooler

Figure 6: Example of TextFooler not retaining semantics. Red original; Green adversarial

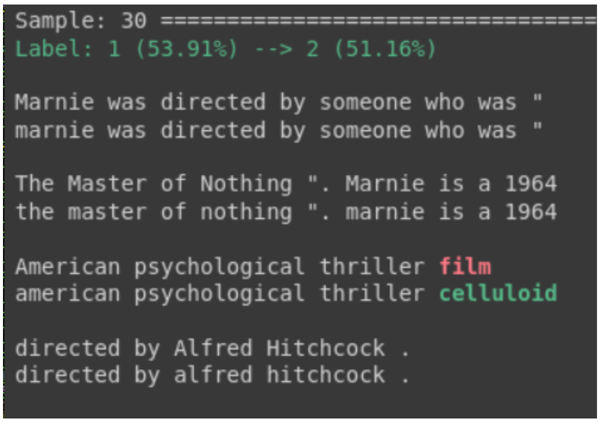

Figure 7: Example of TextFooler not retaining semantics. Red original; Green adversarial

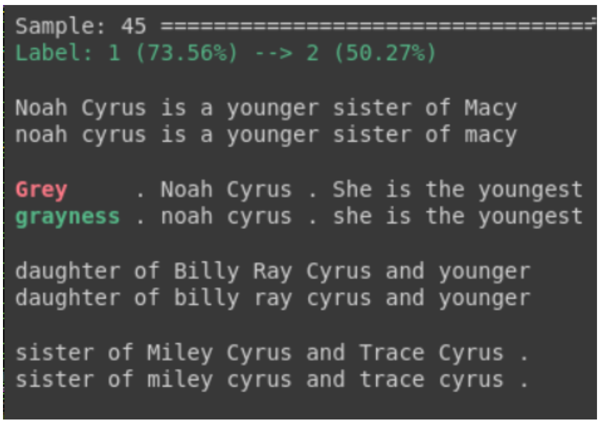

Figure 8: Successful example of TextFooler changing a single word to flip the label