Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

BERT is Robust! A Case Against Synonym-Based Adversarial Examples in Text Classification

Master's Thesis

Jens Hauser

jehauser@ethz.ch

Distributed Computing Group
Computer Engineering and Networks Laboratory
ETH Zürich

Supervisors:
Zhao Meng, Damián Pascual
Prof. Dr. Roger Wattenhofer

September 29, 2021

Acknowledgements

I would like to thank my supervisors, Damián Pascual and Zhao Meng, for providing valuable ideas and suggestions and advising me in writing this thesis and the paper that comes with it. The outcome of this thesis is not what we initially intended. Therefore, I would also like to thank you for having been flexible about this work's direction and outcome.

I would also like to thank Prof. Dr. Roger Wattenhofer for making this thesis possible in the first place and for providing feedback during my mid-term presentation that was essential for the continuation of this thesis.

Abstract

In this thesis, we investigate four word substitution-based attacks on BERT. We combine a human evaluation of individual word substitutions and a probabilistic analysis to show that between 96% and 99% of the analyzed attacks do not preserve semantics, indicating that their success is mainly based on feeding poor data to the model. To further confirm that, we introduce an efficient data augmentation procedure and show that many successful attacks can be prevented by including data similar to the adversarial examples during training. Compared to traditional adversarial training, our data augmentation procedure's per epoch computation time is around 30 times shorter, and we achieve better robustness on two out of three datasets. An additional post-processing step reduces the success rates of state-of-the-art attacks below 4%, 5%, and 8% on three datasets. Finally, by looking at more reasonable thresholds on constraints for word substitutions, we conclude that BERT is a lot more robust than research on adversarial attacks suggests.

Contents

Chapter 1

Introduction

Recent research in computer vision [1, 2] and speech recognition [3] has shown that neural networks are vulnerable to changes that are imperceptible to humans. These insights led to extensive research on attacks for creating so-called *adversarial examples*, inputs designed explicitly to fool a machine learning model. Looking for similar issues in Natural Language Processing (NLP) is natural, and researchers proposed several different attacks over the last years. However, contrary to computer vision, adversarial examples in NLP are not invisible, as discrete characters or words have to be exchanged. This results in a situation where the line between adversarial examples and nonsensical inputs becomes blurry, as it is unclear how much change is acceptable. Ideally, we would like to learn from the mistakes made on adversarial examples to improve future generations of models. However, to do so, we need high-quality adversarial examples. This leads to the question: How useful are current attacks? Do they reveal issues in models, or are they just introducing nonsense?

In this thesis, we show that despite the general consensus that textual adversarial attacks should preserve semantics, current attacks are mainly designed to find as many adversarial examples as possible and neglect the importance of preserving semantics. We combine a human evaluation with a simple probabilistic analysis to show that between 96% and 99% of the adversarial examples on BERT [4] created by four different state-of-the-art attack methods do not preserve semantics. Additionally, we propose a two-step procedure consisting of data augmentation and post-processing for defending against adversarial examples. While this sounds contradictory at first, the results show that we can eliminate a large portion of the successful attacks by simply including data similar to the adversarial examples and further detect and revert many of the remaining adversarial examples in a post-processing step. Both steps combined allow to eliminate up to 95% of the adversarial examples. Compared to traditional adversarial training strategies for defending against adversarial examples, our data augmentation procedure results in a speedup of almost 30x per epoch of training while achieving better robustness on two out of three datasets.

1.1 Contributions

1. We show that most word substitutions introduced by current state-of-the-art attacks do not preserve semantics. The indications from analyzing word counts, word and sentence similarities, and word-embeddings are supported by a human evaluation with 6000 assessments on 800 word substitutions performed by four attacks.

2. Using the results from the human evaluation on individual word substitutions, we perform a probabilistic analysis to conclude that 96% to 99% of the adversarial examples do not preserve semantics.

3. We introduce a two-step defense procedure consisting of data augmentation and post-processing, capable of preventing up to 95% of the attacks. Our data augmentation procedure is efficient and effective. Compared to current adversarial training strategies, we achieve a speedup of almost 30x per epoch and better robustness on two out of three datasets.

Chapter 2

Related Work

2.1 Adversarial Attacks

Adversarial examples have been studied at least since 2004 [5] and started to get considerably more attention in 2013 when it was shown that neural networks are also susceptible to these kind of attacks [1]. Early work mostly focused on computer vision [2] and speech recognition [6]. Adversarial examples in the text domain were first introduced in [7]. In the following years, a range of different attacks have been proposed. [8] use a population-based optimization algorithm for creating adversarial examples, while [9] use Metropolis-Hastings [10, 11]. Further word substitution-based attacks were proposed in [12, 13, 14] and [15]. These attacks are discussed in more detail in Section 3.3.3.

2.2 Adversarial Defense

The most successful methods for defending against adversarial examples in the image domain rely on incorporating adversarial examples during training [16]. Some papers introducing textual adversarial attacks try to do the same [8, 12]. However, due to the high cost of running the attacks, they cannot create sufficiently many adversarial examples and achieve only minor improvements in robustness. [17] suggest the Synonym Encoding Method (SEM), a method that uses an encoder that maps clusters of synonyms to the same embedding. This method works well but also limits the expressiveness of the network. [18] propose a method for fast adversarial training called Fast Gradient Projection Method (FGPM). However, their method is limited to models with non-contextual word vectors as input, because the algorithm depends on being able to take the gradient of the model with respect to counter-fitted [19] GloVe vectors [20]. On BERT, [21] use a geometric attack that allows for creating adversarial examples in parallel and therefore leads to faster adversarial training. Another line of work is around certified robustness through Interval Bound Propagation [22, 23], but these approaches currently do not scale to large models and datasets.

2.3 Criticism on Attacks in NLP

There is little work criticizing or questioning current synonym-based adversarial attacks in NLP. [24] present four categories of constraints that adversarial examples in NLP should follow: semantics, grammaticality, overlap, and non-suspicion to human readers. In a human case study, they find that adversarial attacks often do not preserve semantics, are suspicious to readers, and contain grammatical errors. As a result of this analysis, they propose to increase thresholds on frequently used metrics for similarity of word embeddings and sentence embeddings to higher values.

Chapter 3

Background

This chapter provides the required background on BERT and adversarial examples and explains the four attacks used in this work in detail.

3.1 BERT

BERT [4] stands for **B**idirectional **E**ncoder **R**epresentations from **T**ransformers. As the name suggests, BERT is based on Transformers [25]. Compared to the previously used recurrent neural networks, Transformers allow for better modeling of long-range dependencies and improved parallelization, making it possible to pre-train models in a self-supervised manner on large amounts of data. When BERT was introduced, it obtained new state-of-the-art results on eleven natural language processing tasks.

3.1.1 Architecture

BERT consists of multiple Transformer blocks stacked on top of each other. In every Transformer block, there is a multi-headed self-attention module followed by a traditional feed-forward neural network (see Figure 3.1 taken [25]). There exist two architectures that differ in their size. BERT_{BASE} consists of 12 Transformer blocks with 12 attention heads each and a hidden state size of 768, resulting in a total of 110 million parameters. BERT_{LARGE} consists of 24 Transformer blocks with 16 attention heads each and a hidden state size of 1024, resulting in a total of 340 million parameters. For computational reasons, we use BERT_{BASE} in this thesis.

3.1.2 Input Tokenization

BERT can take either a single sentence as input, or a pair of sentences (e.g. Question, Answer). The input is encoded using WordPiece [26] (see Section 3.1.4) with a vocabulary of 30'000 tokens. The first token of every input sequence is always

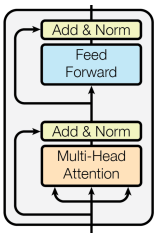

Figure 3.1: Transformer block

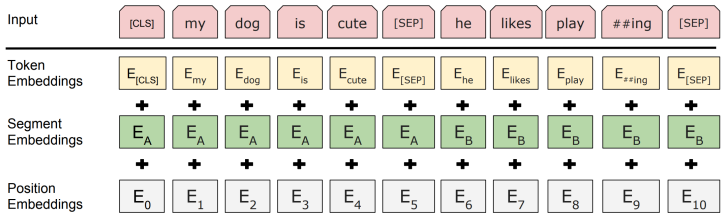

Figure 3.2: Token, segment, and postition embeddings are added to form BERT's input representation.

a special classification token (`[CLS]`). Two sentences are separated by a `[SEP]` token. Additionally, a learned embedding is added to every token to indicate whether it belongs to sentence A or sentence B. The final input representation for a token is constructed by summing the token embedding from the WordPiece embedding, the learned embedding for indicating the sentence and an additional positional embedding. This is shown in Figure 3.2 (taken from [4]).

3.1.3 Pre-Training

During pre-training, BERT is trained on 2'500 million words from English Wikipedia, and another 800 million words from BooksCorpus [27]. The pre-training procedure is conceptually simple. The model is solving two different tasks simultaneously: Next Sentence Prediction (NSP) and Masked Language Modeling (MLM). For the MLM task, 15% of the tokens are selected at random, of which 80% are replaced with the `[MASK]` token, 10% with a random token, and the remaining 10% are unchanged. The model then tries to predict the selected words. The reason why not all of the selected words are masked is that the `[MASK]` token does not appear during fine-tuning, and the authors wanted to mitigate that mismatch. Because the model can take both right and left context into account when solving that task, it is called a bidirectional model. The NSP task consists of predicting whether sentence B follows sentence A in the original text or not. For this, 50% of the time B is the actual next sentence, and 50% of the time it is a random sentence from the corpus. This is done with the objective of creating a model that understands the relationship of sentences.

The pre-trained BERT can be fine-tuned for a wide range of tasks such as question answering or text classification with just one additional output layer. For text classification, the representation of the `[CLS]` token in the last layer is fed into the output layer for classification.

3.1.4 WordPiece

WordPiece [26] is a sub-word segmentation algorithm. Originally introduced for Asian languages, which often have few spaces, the idea is to segment rare or very long words into multiple tokens. With this approach, an infinite set of words can be represented with a finite vocabulary. The algorithm for producing the tokens works as follows:

1. Initialize the vocabulary of tokens with all the basic Unicode characters.
2. Build a language model on the training data with the vocabulary from 1.
3. Combine the two tokens from the vocabulary which increase the likelihood on the training data the most when the combined token is added to the vocabulary.
4. Go to step 2 until a predefined number of tokens are in the vocabulary or the likelihood increase in step 3 falls below a given threshold.

This procedure results in a vocabulary in which frequently used words are represented with their own token. Rare words, on the other hand, have to be combined out of multiple tokens. This is shown in the following example, where the abbreviation "NLP" is split:

- NLP is fun! → [nl, ##p, is, fun, !]

3.2 Metrics for Word and Sentence Similarity

As we will see shortly, synonym-based attacks on text are usually constrained in what words they can choose as substitutes. Common constraints are the cosine similarity between counter-fitted word vectors and the cosine similarity between sentence embeddings from the Universal Sentence Encoder.

3.2.1 Counter-fitted Word Vectors

Many popular methods for finding static word-representations rely on the assumption that semantically similar or related words appear in similar contexts. However, such methods will generally fail to tell synonyms from antonyms. For example, words like *east* and *west*, or *good* and *bad* often appear in the same context, which means these methods will produce similar word vectors for such words. Counter-fitted word vectors [19] are specifically designed to alleviate these problems. Starting from an existing vocabulary of word vectors $V = \{\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_N\}$, the counter-fitted word vectors $V' = \{\mathbf{v'}_1, \mathbf{v'}_2, ..., \mathbf{v'}_N\}$ are created by running

stochastic gradient descent (SGD) on an objective function consisting of three terms. Given sets of synonym pairs $\mathcal{S}$ and antonym pairs $\mathcal{A}$, the antonym repel term

$$\mathrm{AR}(V') = \sum_{(u,w\in\mathcal{A})} \max(0, \cos(\mathbf{v'}_u, \mathbf{v'}_w)), \tag{3.1}$$

where $\cos(\cdot,\cdot)$ denotes cosine similarity, pushes antonyms towards a cosine similarity of 0. The synonym attract term

$$\mathrm{SA}(V') = \sum_{(u,w\in\mathcal{S})} \max(0, 1 - \cos(\mathbf{v'}_u, \mathbf{v'}_w)), \tag{3.2}$$

on the other hand, brings synonyms closer by pushing them towards a cosine similarity of 1. The third term is for vector space preservation and attempts to preserve the semantic information contained in the original vectors:

$$\mathrm{VSP}(V, V') = \sum_{i=1}^{N} \sum_{j\in N(i)} \max(0, \cos(\mathbf{v}_i, \mathbf{v}_j) - \cos(\mathbf{v'}_i, \mathbf{v'}_j)), \tag{3.3}$$

where $N(i)$ denotes a predetermined neighborhood of the i-th word vector in the original vector space. The final cost function is a weighted combination of the three terms:

$$C(V, V') = k_1\mathrm{AR}(V') + k_2\mathrm{SA}(V') + k_3\mathrm{VSP}(V, V'), \tag{3.4}$$

where k_1, k_2 and $k_3 \geq 0$ are hyperparameters. Counter-fitted word vectors achieve state-of-the art performance on SimLex-999 [28], a dataset designed to measure semantic similarity between words.

3.2.2 Universal Sentence Encoder

The Universal Sentence Encoder (USE) [29] is a model for encoding sequences of variable length into fixed size embedding vectors of dimension 512. The procedure is straightforward, the input sequence is tokenized and fed into a deep averaging network (DAN) [30] which outputs a 512 dimensional vector. The DAN is trained using multi-task learning to be as general purpose as possible. The tasks include SkipThought [31], a conversational input-response task, and a classification task. Semantic similarity between two input sequences s and s' is measured by the cosine similarity of their encodings.

3.3 Adversarial Examples

An adversarial example is an input to a machine learning model that an attacker has intentionally designed to cause the model to make a mistake. Usually, for a

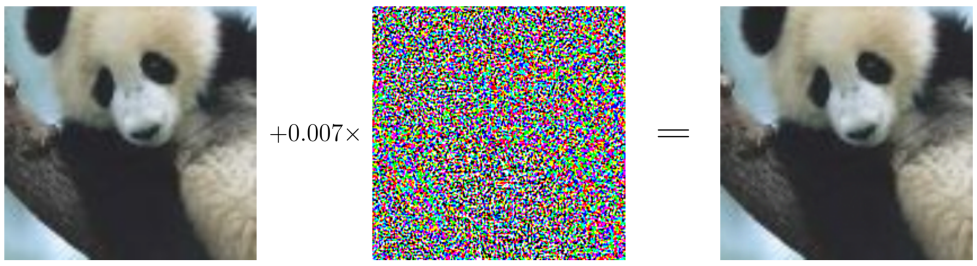

Figure 3.3: An adversarial example created with FGSM (adapted from [2]).

classifier $f : \mathcal{X} \to \mathcal{Y}$, this is done by starting from some correctly classified input $\mathbf{x} \in \mathcal{X}$, and searching for a small perturbation $\boldsymbol{\eta}$, such that $f(\mathbf{x}) \neq f(\mathbf{x} + \boldsymbol{\eta})$. This is what is called an *untargeted attack* because the attacker does not care about the new label, except that it should be different from the correct label. In a *targeted attack*, the attacker is searching for a perturbation $\boldsymbol{\eta}$, such that $f(\mathbf{x} + \boldsymbol{\eta}) = y$, where $y \in \mathcal{Y}$ is the target label.

Besides the distinction between targeted and untargeted, there are two main types of attacks: *white-box attacks* and *black-box attacks*. In a white-box attack, the attacker knows the exact details of the model, including architecture and all the weights. In a black-box attack, the attacker has no knowledge about the model or its internals and is only allowed to query the model with inputs. Usually, it is expected that the model returns confidence scores. All attacks used in this thesis are untargeted black-box attacks.

3.3.1 History of Adversarial Attacks

The study of adversarial examples dates back to 2004 [5] and started to get more attention in 2013, when it was shown that neural networks are also prone to such attacks [1]. Early work focused mainly on image classifiers and white-box attacks. The Fast Gradient Sign Method (FGSM) [2] is the most illustrative and simple version of such an attack. Given a neural network with parameters $\boldsymbol{\theta}$, an input $\mathbf{x}$ with label y, and the cost function $J(\boldsymbol{\theta}, \mathbf{x}, y)$ used to train the model, the perturbation $\boldsymbol{\eta}$ is found as

$$\boldsymbol{\eta} = \varepsilon \cdot sign(\nabla_{\mathbf{x}} J(\boldsymbol{\theta}, \mathbf{x}, y)). \tag{3.5}$$

FGSM changes every pixel exactly by the value of ε, if ε is small enough, this is imperceptible to the human eye. An example of such an attack is shown in Figure 3.3. Since then, attacks which keep the total perturbation even smaller by making the size of the perturbation pixel-dependant have been introduced

Original Text	Very much enjoyed it! Our waitress was very attentive and friendly.	100% positive
Adversarial Example	Awfully much enjoyed it! Our waitress was very beware and empathy.	66% negative
Original Text	Pretty awesome place. Great pools and kid friendly.	100% positive
Adversarial Example	Kinda glamorous place. Whopping pools and kid friendly.	69% negative
Original Text	Food is terrible. Simple as that. Service is decent though.	100% negative
Adversarial Example	Nutritious is egregious. Simple as that. Service is decent though.	83% positive

Table 3.1: Adversarial examples created with TextFooler (hand-picked).

[16, 6]. While the size of the perturbation for FGSM is defined according to ε, other attacks use an explicit constraint on $\|\boldsymbol{\eta}\|_2$ or $\|\boldsymbol{\eta}\|_\infty$.

3.3.2 Adversarial Examples in Text Classification

While you can perturb an image in a way which is imperceptible for humans by slightly changing pixel values, creating an adversarial example for text always means changing, adding or deleting at least one letter or symbol. Nevertheless, in the last years much research has been dedicated towards finding adversarial examples in NLP and they can generally be classified into three categories: character-level attacks, word-level attacks, and sentence-level attacks. As the names suggest, character-level attacks generate adversarial examples by changing individual characters of the original text, word-level attacks change whole words and sentence-level attacks work by paraphrasing sentences. In this work, we focus on word-level attacks. Table 3.1 shows some word-level adversarial examples. Note that these examples are hand-picked and not representative of the overall quality.

Formally, for a classifier $f : \mathcal{S} \to \mathcal{Y}$ and some correctly classified input $s \in \mathcal{S}$, a textual adversarial example is an input $s' \in \mathcal{S}$, such that $f(s) \neq f(s')$, and $sim(s, s') \geq t_{sim}$, where $sim(s, s') \geq t_{sim}$ is a constraint on the similarity of s and s'. For text classification, $s = \{w_1, w_2, ..., w_n\}$ is a sequence of words. Common notions of similarity are the cosine similarity of counter-fitted word vectors, which we will denote as $cos_{cv}(w_i, w_i')$ or the cosine similarity of sentence embeddings from USE, which we will denote as $cos_{use}(s, s')$. Note that this is a slight abuse of notation since s and s' are just sequences of words. The notation should be interpreted as follows: We first apply USE to s and s' to get two

sentence vectors and then calculate the cosine similarity. The same holds for $cos_{cv}(w_i, w_i')$, where we first get the counter-fitted word vectors of w_i and w_i'. Also, note that whenever we talk about the *cosine similarity of words*, it refers to the cosine similarity of words in the counter-fitted embedding. Similarly, *USE score* refers to the cosine similarity of sentence embeddings from the USE.

3.3.3 Examples of Attacks

We use four different attacks for our experiments. All of them are based on the idea of exchanging words with other words of similar meaning. The attacks consist of two main steps: First, the words are ranked according to their importance, determining the order of replacement. One word at a time, the words are then replaced with another word from a given candidate set until the prediction of the model changes. We call such a word substitution a *perturbation*. In the following, for the four attacks used in this work, we explain the methods used to rank the words by importance, how the candidate set of replacement words is constructed, and which constraints exist.

TextFooler

For a given input sequence $s = \{w_1, w_2, .., w_n\}$, TextFooler [13] estimates the importance of a word w_i by the change in prediction resulting from deleting that word. Formally an importance score I_{w_i} is calculated as

$$I_{w_i} = \begin{cases} f_y(s) - f_y(s_{\setminus w_i}) & \text{if } f(s) = f(s_{\setminus w_i}) = y \\ (f_y(s) - f_y(s_{\setminus w_i})) + (f_{\hat{y}}(s_{\setminus w_i}) - f_{\hat{y}}(s)) & \text{if } f(s) = y,\ f(s_{\setminus w_i}) = \hat{y}\ , \end{cases} \tag{3.6}$$

where f_y represents the prediction score of a classifier f for the label y, $s_{\setminus w_i} = \{w_1, .., w_{i-1}, w_{i+1}, .., w_n\}$ and $\hat{y} \neq y$. The candidate set of replacement words is built from the 50 nearest neighbors with cosine-similarity ≥ 0.5 in a counter-fitted word embedding. The candidate set is then filtered for words with the same part-of-speech (verb-noun swaps are allowed), and a custom set of stop-words, constructed from NLTK and spaCy, is filtered out. Finally, the remaining words are checked for $cos_{use}(s, s') \geq 0.878$,[1] and the one which changes the prediction the most is chosen as a replacement. However, this constraint on the USE score is not checked between the current perturbed text s' and the original text s, but instead between the current perturbed text s' and the previous perturbed version s''. This means that by perturbing one word at a time, the effective USE score between s and s' can be a lot lower than the threshold suggests.

[1] The official value is 0.841 on the angular similarity between sentence embeddings, which corresponds to a cosine similarity of 0.878

Probability Weighted Word Saliency (PWWS)

PWWS [12] uses a slightly different word ranking algorithm compared to TextFooler. The word importance ranking is calculated as

$$I_{w_i} = f_y(s) - f_y(s_{w_i \to [unk]}) \tag{3.7}$$

where $s_{w_i \to [unk]} = \{w_1, .., w_{i-1}, unk, w_{i+1}, .., w_n\}$ and $[unk]$ is the out-of-vocabulary token. In contrast to TextFooler, PWWS also takes the impact of the synonyms into account when determining the word replacement order. The final score function for determining the replacement order is given as:

$$H(s, w_i, w_i^*) = \frac{\exp(I_{w_i})}{\sum_{j=1}^{n} \exp(I_{w_j})} \cdot \Delta f_i^*, \tag{3.8}$$

where $\Delta f_i^* = f_y(s) - f_y(s_{w_i \to w_i^*})$ is the change in confidence for the true label when word w_i is replaced with synonym w_i^*. WordNet[2] synonyms are used to build a candidate set $\mathcal{L}_i$ and w_i^* is choosen as the word from that set which changes the prediction the most:

$$w_i^* = \underset{w_i' \in \mathcal{L}_i}{\arg\max} \Big(f_y(s) - f_y(s_{w_i \to w_i'}) \Big) \tag{3.9}$$

There are no constraints, except that stopwords are filtered out using NLTK.

BERT-Attack

A shortcoming of traditional synonym-based attacks like TextFooler or PWWS is that they do not take the context into account when building their candidate set. This can lead to problems if a word is polysemic, i.e., has multiple meanings in different contexts. Many attacks also do not take part-of-speech into account, which leads to unnatural and semantically wrong sentences. BERT-based attacks claim to produce more natural text by relying on a BERT masked language model (MLM) for proposing the set of candidate words. A prominent example of such an attack is BERT-Attack [14]. BERT-Attack calculates the importance scores similar to TextFooler, but instead of deleting words, BERT-Attack replaces the word for which the importance score is calculated with the `[MASK]` token:

$$I_{w_i} = f_y(X) - f_y(s_{w_i \to [mask]}) \tag{3.10}$$

The candidate set $\mathcal{L}_i$ is constructed from the top 48 predictions of the masked language model and the replacement word is chosen as the word which changes the prediction the most, subject to $cos_{use}(s, s') \geq 0.2$. Stopwords are filtered out using NLTK.

[2] https://wordnet.princeton.edu/

BAE

BAE corresponds to BAE-R in [15]. Similar to BERT-Attack, BAE is an attack based on a MLM. The word importance is estimated as the decrease in probability of the correct label when deleting a word, similar to TextFooler. BAE uses the top 50 candidates of the MLM to build the candidate set and tries to enforce semantic similarity by requiring $cos_{use}(s, s') \geq 0.936$.

Chapter 4

Setup

4.1 Datasets

For our experiments, we use three different text classification datasets: AG News,[1] IMDB,[2] and Yelp.[3] On Yelp, we only use the examples consisting of 80 words or less. Statistics of the three datasets are displayed in Table 4.1.

Dataset	Labels	Train	Test	Avg Len
AG News	4	120'000	7'600	43.93
Yelp	2	199'237	13'548	45.69
IMDB	2	25'000	25'000	279.48

Table 4.1: Statistics of the three datasets.

AG News [32] is a topic classification dataset. It is contructed out of titles and headers from news articles categorized into the four classes "World", "Sports", "Business", and "Sci/Tech".

Yelp [32] is a binary sentiment classification dataset. It contains reviews from Yelp, reviews with one or two stars are considered negative, reviews with three or four stars are considered positive.

IMDB is another binary sentiment classification dataset. It contains movie reviews labelled as positive or negative.

4.2 Implementations

We use *bert-base-uncased* from HuggingFace Transformers[4] for all our experiments. For the implementations of TextFooler, PWWS, BERT-Attack, and BAE

[1] https://huggingface.co/datasets/ag_news
[2] https://huggingface.co/datasets/imdb
[3] https://huggingface.co/datasets/yelp_polarity
[4] https://huggingface.co/transformers/

Dataset	Clean Acc. (%)	Attack Success Rate (%)			
		TextFooler	PWWS	BERT-Attack	BAE
AG News	94.57	84.99	64.95	79.43	14.27
Yelp	97.31	90.47	92.23	93.47	31.50
IMDB	93.77	98.16	98.70	99.03	57.13

Table 4.2: Attack success rates of the different attacks on fine-tuned BERT models.

we use TextAttack [33].[5] The adapted attacks with custom thresholds are also implemented using the TextAttack library. For adversarial training, we adapt the code from [34].

4.3 Starting Point

We fine-tuned BERT for two epochs on AG News, Yelp, and IMDB with a learning rate of $2e$-5. On AG News and Yelp, we used a batch size of 32, on IMDB a batch size of 8. To evaluate the attacks, we randomly sampled 1000 examples from each test-set for running the attacks.

The clean accuracies of our fine-tuned models, and the attack success rates of the different attacks are shown in Table 4.2. Is is worth noting that the average sequence length in IMDB is around six times longer compared to AG News and Yelp, which makes IMDB easier to attack. To see this, take an attack without constraint on the sentence similarity (PWWS for example). Assuming a maximum replace rate of 0.4, the number of potential adversarial examples for a input sequence of length l is $(0.4 \cdot l)^{|\mathcal{C}|}$, where $|\mathcal{C}|$ is the size of the candidate set.

[5] https://textattack.readthedocs.io/en/latest/

Chapter 5

Observations

The following chapter provides insights into why the attacks succeed and motivates why it is worth taking a closer look at the quality of adversarial examples through a human evaluation. We look into word frequencies, word and sentence similarity scores, and word-embeddings inside BERT.

5.1 Word Frequencies

For every word substitution, we count the number of occurrences of the original word and the attack word in the training set. We then calculate the median number of occurrences in the training set for original and attack words for every dataset and attack.

Dataset	**Word**	**Median occ. in training data**			
		TextFooler	PWWS	BERT-Attack	BAE
Ag News	original	736	889	585	617
	attack	18	24	344	4
Yelp	original	4240	5715	4521	4601
	attack	19	13	3398	44
IMDB	original	1362	1598	1408	1221
	attack	47	66	1016	23

Table 5.1: Median word occurrences of original words and attack words in training set

Table 5.1 shows the results. In general, the attack words appear less often in the training set than the original words. The difference is significant for three out of four attacks, indicating that these attacks introduce words that humans would rarely use. However, if a word barely appears during fine-tuning, it is no surprise that the model does not know how to interpret that word for the particular task. BERT-Attack acts differently in that regard, we believe that this results from a combination of having to choose from candidates of a MLM and having weak

Dataset	Attack	$\arg\max_i f_i(w) \neq y_{true}$ (%)	
		Original Word	Attack Word
AG News	TextFooler	32.69	75.37
	PWWS	39.96	83.94
	BERT-Attack	34.08	77.09
	BAE	47.34	90.69
Yelp	TextFooler	27.21	55.40
	PWWS	25.44	66.24
	BERT-Attack	24.73	64.45
	BAE	23.97	58.13
IMDB	TextFooler	30.75	58.52
	PWWS	32.27	65.20
	BERT-Attack	29.78	63.73
	BAE	32.56	54.88

Table 5.2: Percentage of times that a word has higher relative frequency in a class other then the ground truth class.

constraints. An attack that has to choose from a set of similar words (whether that is synonyms or words retaining high sentence similarity) is more likely to choose an uncommon word.

5.1.1 Word Associations with Wrong Label

For a classification problem with K classes, let us define the relative frequency of a word w in class $i \in \{1, ..., K\}$ as

$$f_i(w) = \frac{\sum_{v \in \mathcal{V}_i} [w == v]}{|\mathcal{V}_i|}, \tag{5.1}$$

where $\mathcal{V}_i$ is the corpus of all the words in class i and

$$[w == v] = \begin{cases} 1 & \text{if } w = v \\ 0 & \text{otherwise.} \end{cases} \tag{5.2}$$

For a word w from an input sequence labeled as y_{true}, Table 5.2 shows the percentage of times that $\arg\max_i f_i(w) \neq y_{true}$ if w is an original word and the percentage of times that $\arg\max_i f_i(w) \neq y_{true}$ if w is an attack word. While it is not uncommon for the original word to occur more often in one of the other classes, the attack words are significantly more often associated with another class. This raises the question whether there is some justification in the model's decision to change its prediction. After all, for a simpler model based on word statistics, we would not be surprised about a change in prediction if sufficiently many words are exchanged with words that appear more often in other classes.

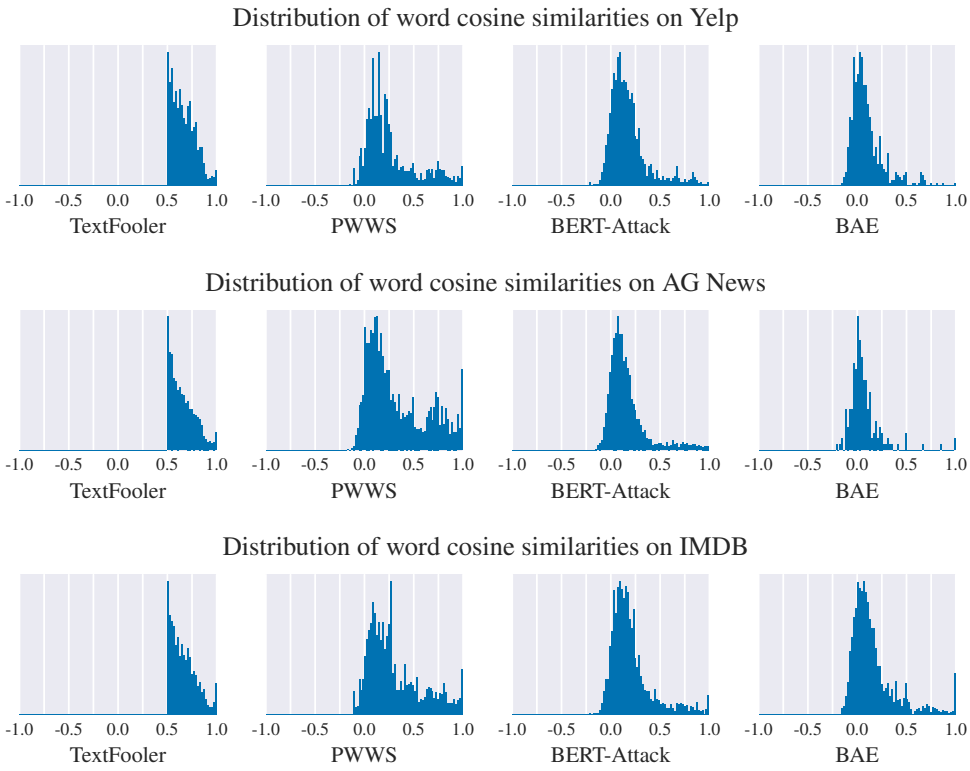

Figure 5.1: Distribution of cosine similarities of words for the different attacks and datasets.

5.2 Similarity of original and perturbed words

Section 5.1.1 showed that the words introduced by the attacks often occur more frequently in classes different from the ground truth class. This indicates that the meaning of these words is different from the original words. Therefore, we analyzed cosine similarities between original and attack words in a counter-fitted vector space. As described in Section 3.2.1, such a vector space is specifically designed to pull synonyms towards a cosine similarity of 1 and antonyms towards a cosine similarity of 0.

Figure 5.1 shows the distributions of cosine similarities between original and attack words for the different attacks on AG News, Yelp, and IMDB. Except for TextFooler, which has a constraint that the attack word needs to have at least a cosine similarity of 0.5 with the original word, all attacks have a large portion of perturbations where the cosine similarity is around 0. For BERT-Attack, this is not surprising, as there exists no constraint on the cosine similarity and only a lenient constraint on the sentence similarity. However, PWWS chooses its candidate set from WordNet synonyms, and BAE has a constraint on a high

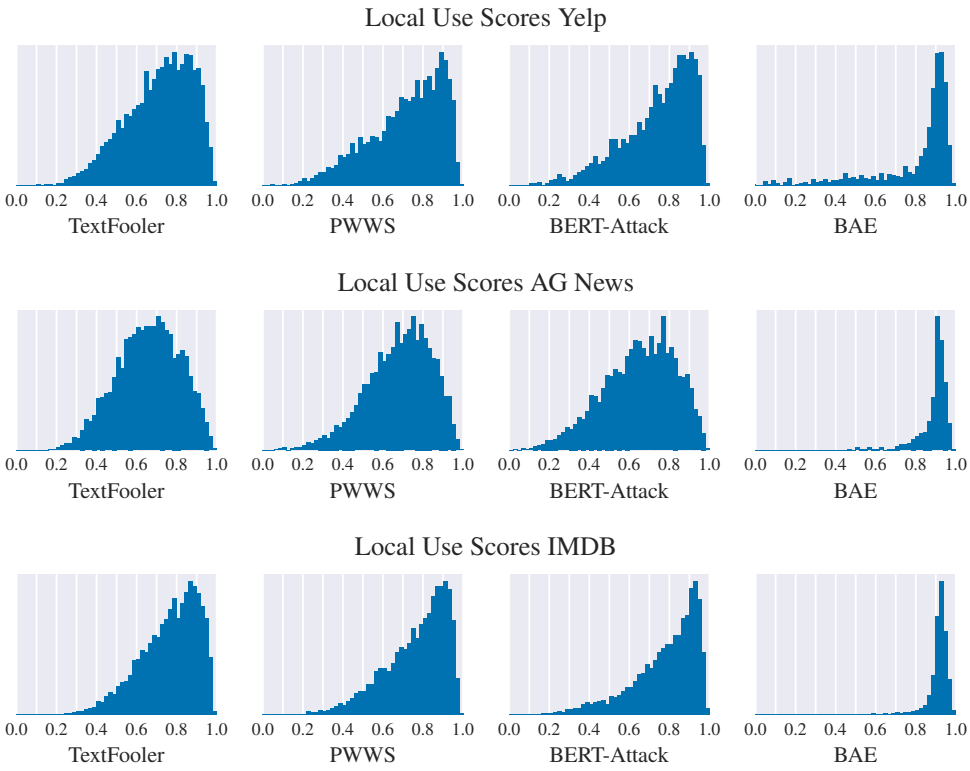

Figure 5.2: Distribution of cosine similarities from USE embeddings around replaced words, with a window size of 11.

sentence similarity. This shows that choosing from a set of synonyms might still result in perturbations that are semantically different. The reason is that words can be polysemantic. An example is “terrific”; looking up synonyms for terrific yields both “horrible” and “awesome”. Clearly, for a given context, only one of them is a suitable synonym. The results strongly indicate that PWWS relies heavily on such “false” synonyms. Furthermore, the results obtained from BAE also show that a high sentence similarity score does not necessarily require a high cosine similarity on the word level.

5.3 Sentence Similarity

While we have just seen that a high sentence similarity is no guarantee for a high word similarity, ideally, an adversarial example should also have a high sentence similarity to the original text. We measure sentence similarity scores by calculating the cosine similarity from USE-embeddings, the most frequently used metric to constrain attacks on text. However, attacks have different implemen-

tations of that constraint. The constraint can be applied to a fixed-size window around the replaced word (TextFooler and BAE) or to the whole input sequence at once (BERT-Attack). Furthermore, when perturbing multiple words of an input sequence, it can be applied between the current version and the original text (BERT-Attack and BAE) or between the current version and the perturbed version in the previous step (TextFooler). To compare scores across the different implementations, we computed sentence similarity scores with a window size of 11 and compared them to the original text.

Figure 5.2 shows the distribution of cosine similarities between USE-embeddings from the original text and adversarial example for the four attacks. It can be seen that BAE is the only attack that properly restricts the sentence similarity scores because it uses both a fixed window size and compares to the original text. On the other hand, the sentence similarity scores of TextFooler, PWWS, and BERT-Attack are almost the same, even though PWWS does not use such a constraint, BERT-Attack only requires a cosine similarity of 0.2, and TextFooler requires a cosine similarity of 0.88. This shows that the implementation in TextFooler, which always compares to the previous version, is a lot weaker than it looks at first sight.

For an idea about what different values of sentence similarity correspond to, Table A.4 in the Appendix shows examples of text fragment pairs with USE scores from 0.29 to 0.99.

5.4 BERT Word-Embeddings

As explained in Section 3.1.2, BERT's initial embedding is constructed by summing positional, token and sentence embedding. For every token, this results in a vector $\mathbf{v}_{init} \in \mathbb{R}^{768}$. This representation is then passed through the 12 Transformer layers of BERT and in every layer we obtain a new representation $\mathbf{v}_i \in \mathbb{R}^{768}$, where $i \in \{1, ..., 12\}$ denotes the layer. To compare embeddings from the original text and adversarial examples, we need a word-level representation because a word consisting of one token can be replaced with a word consisting of multiple tokens and vice versa. We obtain the word representation by averaging the token representations of all tokens in word. In the following, we call this a *word embedding*.

To better understand how BERT interprets the words introduced by the attacks, we feed 300 pairs of [*original text*, *adversarial example*] through the network and compute the cosine similarity between the word embeddings of the words in the original text and the words in the adversarial example in every layer. This results in a distribution of cosine similarities for every layer. We can split the distribution into unchanged words (the same in original text and adversarial example) and changed words. To have a reference, we further exchanged the attack

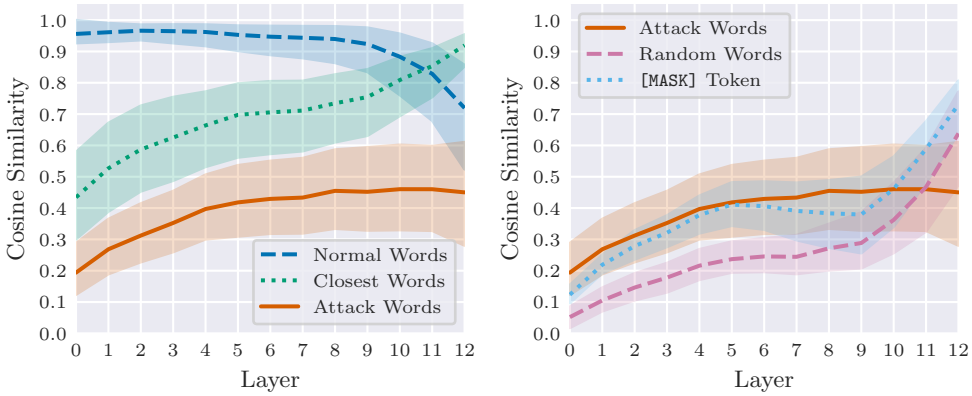

Figure 5.3: Cosine similarities between BERT word embeddings from words in adversarial examples (or [MASK] and random references) and words in original text.

words with:

1. a completely random word
2. the [MASK] token
3. the closest neighbor of the original word in a counter-fitted embedding

This results in five cosine similarity distributions in every layer. Figure 5.3 shows them for TextFooler on AG News. Results from the other attacks can be found in the appendix. We show the range from 25th to 75th percentile with the median as a thick line for every class of word-pairs. It can be observed that the cosine similarity between original and attack words is significantly lower than the cosine similarity between the original word and the closest neighbor. Further, it can also be seen that the attack words strongly affect the word embeddings of the unchanged words, especially from layer 9 to layer 12. The most interesting observation is that the cosine similarity between the original words and the [MASK] token is similar to that between the original and attack word for the first ten layers. We would expect the [MASK] token to be neutral. It is a token that does not occur during fine-tuning and does not carry any information for the classification task. There are two potential explanations for this:

1. The model does not know how to interpret the attack words.
2. The model interprets the words correctly, but there is a significant semantic difference between original and attack word.

The last layers form the basis for the classification layer, therefore the crossing of the lines from [MASK] tokens and random words with the attack words is

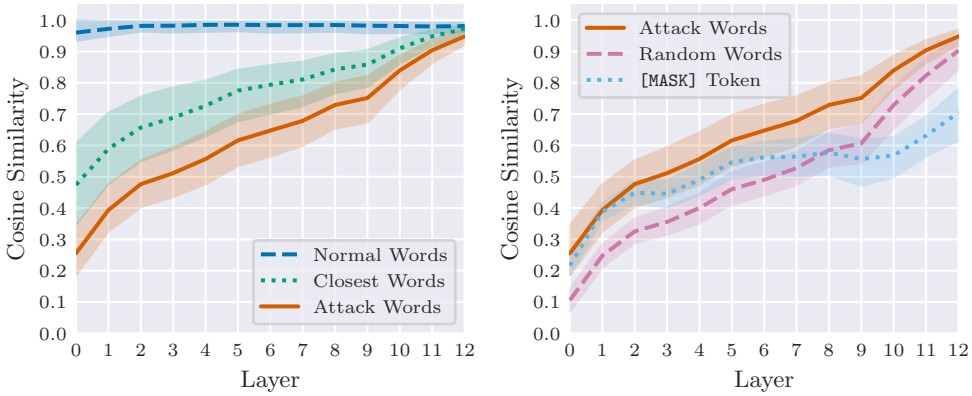

Figure 5.4: Cosine similarities between words in adversarial examples (or `[MASK]` and random references) and words in original text in a robust model.

likely attributed to the overall different classification of the adversarial example compared to when the attack words are replaced with a random word or the `[MASK]` token.

5.4.1 Comparison to Robust Model

Figure 5.4 shows the same analysis (with the same adversarial examples) on a model trained using adversarial training according to [21]. While the distribution for the attack words is still below the distribution for the closest words from a counter-fitted embedding, the overall trend shows much higher cosine similarities. We can also see that the embeddings of the unchanged words are almost not affected at all. However, the random words also obtain a much higher cosine similarity with the original words. It appears that at least partially, the robustness does not come from a better understanding of the input, but simply from being less susceptible to change.

Chapter 6

Quality of Adversarial Examples

The previous chapter showed that attacks often use words that barely occurred during training and appear more frequently in other classes. Further, we observed that BAE, BERT-Attack, and PWWS often use words with low cosine similarity to the original word. These observations raise doubts about the validity of adversarial examples, but they do not allow for a final conclusion. To truly judge the quality of the adversarial examples, we need human opinions. Therefore, we conducted a human evaluation on word substitutions performed by the different attacks. In the following, we call such a word substitution a perturbation. A probabilistic analysis is then used to generalize the results on perturbations to attacks.

6.1 Human Evaluation

For the human evaluation, we relied on labor crowd-sourced from Amazon Mechanical Turk. We limited our worker pool to workers in the United States and the United Kingdom who completed over 5000 Human Intelligence Tasks (HITs) with over 98% success rate. We collected 100 pairs of [*original word*, *attack word*] for every attack and another 100 pairs for every attack where the context is included with a window size of 11. For the word-pairs, inspired by [24], we asked the workers to react to the following claim: *"In general, replacing the first word with the second word preserves the meaning of the sentence."* For the words with context, we presented the two text fragments on top of each other, highlighted the changed word, and asked the workers: *"In general, the change preserves the meaning of the text fragment."* In both cases the workers had seven answers to choose from: "Strongly Disagree", "Disagree", "Somewhat Disagree", "Neutral", "Somewhat Agree", "Agree", "Strongly Agree". We convert these answers to a scale from 1-7, where higher is better. Every word-pair was judged by ten workers, the words with context were scored by five workers each.

Table 6.1 shows the results of this human analysis. Contrary to what is suggested in papers proposing the attacks, our results show that humans generally

Task	Metric	Attack			
		TextFooler	PWWS	BERT-Attack	BAE
Word-Similarity	Avg. (1-7)	**3.88**	3.83	2.27	1.64
	Above 5 (%)	**22**	21	4	0
	Above 6 (%)	**7**	6	4	0
Text-Similarity	Avg. (1-7)	**3.47**	2.70	2.55	1.85
	Above 5 (%)	**24**	13	7	3
	Above 6 (%)	**12**	6	3	2

Table 6.1: Average human scores on a scale from 1-7 and the percentage of scores above 5 and 6 (corresponding to the answers "Somewhat Agree" and "Agree") for the different attacks and when the words where shown with (text similarity) or without (word similarity) context.

tend to disagree that the newly introduced word preserves the meaning. This holds for all attacks and regardless of whether we show the word with or without context. We believe this difference is mainly due to how the text is shown to the judges and what question is posed. For example, asking *"Are these two text documents similar?"* on two long text documents that only differ by a few words is likely to get a higher agreement because the workers will not bother going into the details. Therefore, we believe it is critical to show the passages that are changed.

Regarding the different attacks, it becomes clear from this evaluation that building a candidate set from the first 48 or 50 candidates proposed by a MLM does not work without an additional constraint on the word similarity. The idea of BERT-based attacks is to only propose words that make sense in the context, however, fitting into the context and preserving semantics is not the same thing. The results on BAE further make it clear that a high sentence similarity according to the USE score is no guarantee for semantic similarity. PWWS and TextFooler receive similar scores for word similarity, but the drop in score for PWWS when going from word similarity to text similarity indicates that while the synonyms retrieved from WordNet are usually related to the original word, the relation is often wrong in the given context. TextFooler receives the highest scores in this analysis, but even for TextFooler, just 22% and 24% of the perturbations were rated above 5, which corresponds to "Somewhat Agree".

6.1.1 Voter Agreement

To measure voter agreement, we calculate the average number of workers who voted within ± 1 of the mean score for a perturbation. For the words with context, this is 3.57 workers out of 5. For the words without context, this is 6.78 out of 10.

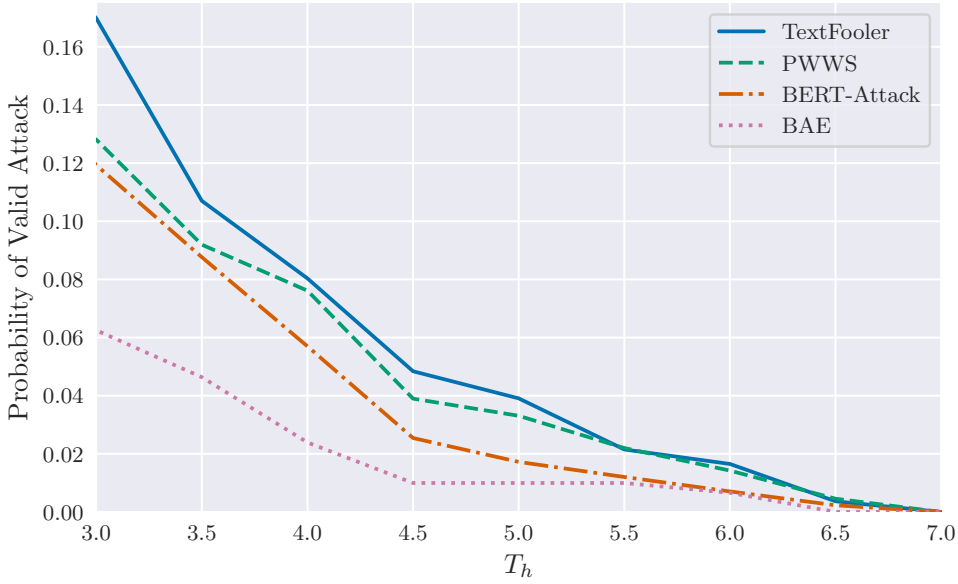

Figure 6.1: Probability that an attack is valid according to our probabilistic analysis, for the different attacks and for different thresholds on what is considered a valid perturbation.

6.1.2 Probabilistic Estimation of Valid Attacks

The human evaluation is based on individual perturbations. A successful attack usually changes multiple words and therefore consists of multiple perturbations. This begs the question: How many of the successful attacks are actually valid attacks? To answer this question we need to define the expressions valid attack and valid perturbation.

Definition 6.1 (Valid Perturbation)**.** A valid perturbation is a perturbation that receives a human score above some threshold T_h.

Definition 6.2 (Valid Attack)**.** A valid attack is an attack consisting of valid perturbations only.

Sensible values for T_h are in the range 5-6, which corresponds to "Somewhat Agree" to "Agree". In order to get an estimate for the percentage of valid attacks, we perform a simple probabilistic analysis. Let A_{val}, P_{val} and A^i_{val} denote the events of a valid attack, a valid perturbation and a valid attack consisting of exactly i perturbations. Further, let $p(i)$ denote the probability that an attack perturbs i words. Using that notation, we can approximate the probability that

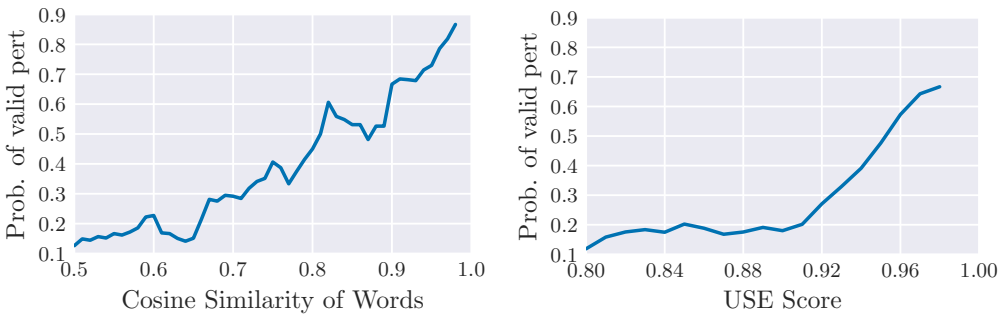

Figure 6.2: The probability that a perturbation is considered valid by a human, as a function of cosine similarity of words (left) and USE score (right). T_{human} is set to 5, i.e. an average score of 5 is required to be considered valid.

a successful attack is valid as

$$\begin{aligned} p(A_{val}) &= \sum_{i=1}^{N} p(i) p(A_{val}^{i}) \\ &\approx \sum_{i=1}^{N} p(i) p(P_{val})^{i}, \end{aligned} \tag{6.1}$$

where N is the maximal number of allowed perturbations. With the data from Amazon Mechanical Turk and the collected adversarial examples, we can get an unbiased estimate for this probability as

$$\hat{p}(A_{val}) = \sum_{i=1}^{N} \hat{p}(i) \left(\frac{\text{count}[S_h \geq T_h]}{n_{pert}} \right)^{i}, \tag{6.2}$$

where S_h is the average score of the workers for a perturbation, n_{pert} is the total number of perturbations analyzed by the workers for any given attack, and $\hat{p}(i)$ can be estimated using counts. The results of this analysis are shown in Figure 6.1 as a function of the threshold T_h. It can be seen that if we require an average score of 5 for all perturbations, we can expect around 4% of the successful attacks from TextFooler to be valid, slightly less for PWWS, below 2% for BERT-Attack, and just around 1% for BAE. In other words, between 96% and 99% of the successful attacks can not be considered valid according to the widely accepted requirement that adversarial examples should preserve semantics.

This analysis assumes that perturbations are independent of each other, which is not true because every perturbation impacts the following perturbations. Nevertheless, we argue that this approximation tends to result in optimistic estimates on the true number of valid attacks for the following reasons: 1) When an attack is already almost successful, all attacks except for PWWS try to maximize

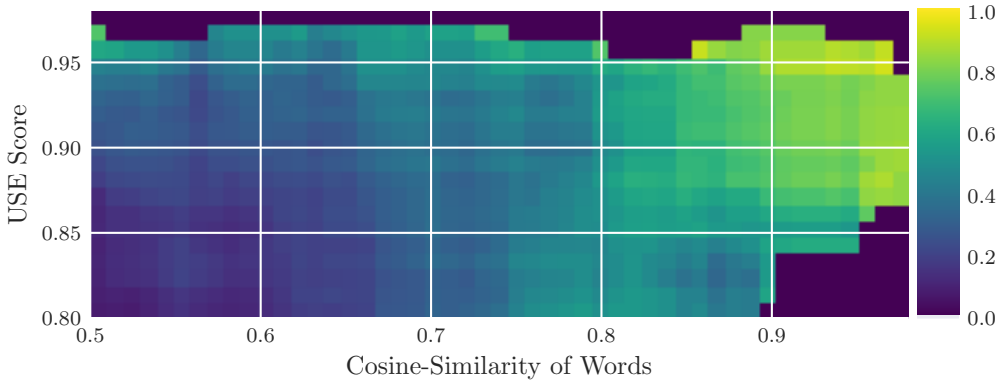

Figure 6.3: Probability that an attack is valid according to our probabilistic analysis as a function of both USE score and cosine similarity of words

sentence similarity on the last perturbation, making the last perturbation generally weaker. 2) We strongly assume that in a sentence with multiple changes, a human is generally less likely to say that the meaning is preserved, even if the individual perturbations are considered valid.

6.1.3 Metrics vs. Human

Figure 6.2 shows the probability that a perturbation is considered valid (for $T_h = 5$) as a function of cosine similarity of words and as a function of USE score. The plots are based on the 400 words with context from the different attacks which were judged by humans. We use left-aligned buckets of size 0.05, i.e., the probability of a valid perturbation for a given cosine similarity x and metric $m \in \{cos_{cv}(\cdot,\cdot), cos_{use}(\cdot,\cdot)\}$, is estimated as

$$\hat{p}(P_{val}) = \frac{\text{count}[(S_h \geq T_h) \wedge (m \in [x, x+0.05))]}{\text{count}[m \in [x, x+0.05)]}. \tag{6.3}$$

It can be observed that there is a strong positive correlation between both metrics and the probability that a perturbation is considered valid, confirming both the validity of such metrics and the quality of our human evaluation. However, the exact probabilities have to be interpreted with care, as the analysis based on one variable does not consider the conditional dependence between the two metrics.

A two-dimensional analysis shows a more accurate picture and is displayed in Figure 6.3. We used buckets of size 0.1×0.1 and required at least 10 datapoints in a bucket, hence the dark areas for very high sentence similarity and in the bottom right corner.

Chapter 7

Adversarial Defense

We have shown that current attacks use lenient constraints and, therefore, mostly produce adversarial examples that should not be considered valid, but finding the right thresholds on the constraints is difficult. [24] try to find these thresholds by choosing the value where humans "Agree" (on a slightly different scale) on average and find thresholds of 0.90 on the word similarity and 0.98 on the sentence similarity score. However, this misses all the perturbations which were considered valid by the workers at lower scores (compare Figures 6.2 and 6.3). In the following, we introduce a defense procedure which shows that even for low thresholds, it is possible to avoid many adversarial examples. The procedure shows that the success of adversarial examples is mainly based on the discrepancy between training data and adversarial examples.

7.1 Defense Procedure

Our procedure consists of a gradient based data augmentation procedure followed by a post-processing step.

Data Augmentation

1. Initialize thresholds $t_{rr} \in (0, 100]$ for the maximal percentage of words to augment, and $t_{cv} \in (0, 1)$ for a threshold on cosine similarity of words.

2. During training, for every input s in a batch, the importance score of a word w consisting of tokens $\mathbf{v}_j \in \mathbb{R}^{768}$ in BERT's initial embedding is estimated as

$$I_w = \sum_{\mathbf{v}_j \in w} \mathbf{v}_j \cdot \nabla_{\mathbf{v}_j} L(\boldsymbol{\theta}, s, y), \tag{7.1}$$

   where $\boldsymbol{\theta}$ are the parameters of BERT, L is the loss function and y is the label. The t_{rr} percent of words with highest importance score are determined and the union of the words considered as stop-words by the four attacks is filtered out.

3. Then, for every word marked as important according to 2., a candidate set $\mathcal{C} = \{w'_1, ..., w'_n\}$ is built out of the 50 nearest neighbors in a counter-fitted embedding with cosine similarity greater than t_{cv}. To account for the fact that all attacks tend to favor words with low cosine similarity, the replacement $w'_i \in \mathcal{C}$ for the original word w is chosen with probability:

$$p(w'_i) = \frac{1 - cos_{cv}(w, w'_i)}{\sum_{w'_i \in \mathcal{C}} 1 - cos_{cv}(w, w'_i)}. \tag{7.2}$$

   The augmented batch is then appended to the original batch, increasing the batch size by a factor of two.

The data augmentation procedure makes the model more robust against attack words with cosine similarity greater t_{cv}. If we expect BERT to be robust against these kinds of replacements, this is the least we should do. Otherwise, we cannot expect the model to generalize to the attack's input space, which is significantly larger than the input space during fine-tuning.

The second step is a post-processing step based on ensembling. This step builds on the robustness to random substitutions obtained from data augmentation.

Post-processing

1. For every text that should be classified, N versions are created where t_{rr} percent of the words (which are not stop-words) are selected uniformly at random and are exchanged by another uniformly sampled word from a candidate set $\mathcal{C}$ consisting of the 50 nearest neighbors with cosine-similarity above t_{cv}.

2. The outputs of the model (logits) are added up for the N versions and the final prediction is made according to the maximum value. Formally, let $l_j(s)$ denote the value of the j-th logit for some input s. Then the prediction y_{pred} is made according to

$$y_{pred} = \arg\max_j \sum_{i=1}^{N} l_j(s_i). \tag{7.3}$$

This procedure can be applied for any threshold $t_{cv} \in (0, 1)$, but it only makes sense if we expect an attack to use the same or a higher threshold. We always set t_{cv} to the same value as the attack uses. Further, we set $t_{rr} = 40$ and $N = 8$ in all our experiments, and we use the same thresholds for both data augmentation and post-processing.

Chapter 8

Results

8.1 Effect of Defense Procedure

Our defense procedure is not expected to have a big impact against attacks which introduce mostly words with cosines similarities $\leq t_{cv}$. Hence we modify all attacks with the constraint $cos_{cv}(w_i, w_i') \geq 0.5 \; \forall i$ and ran the attacks on the following models: A model trained normally (Method N), a model trained using our data augmentation procedure (DA), and a model trained using data augmentation where our post-processing method is applied (DA + PP). Additionally, we provide a baseline for our post-processing procedure by instead masking 5% of all tokens with the `[MASK]` token (DA + MA_5; see Appendix A.4) and we show the impact of applying the post-processing step without data augmentation (N + PP). The modified attacks are denoted as PWWS′ and BERT-Attack′. We leave BAE′ out of this analysis because its attack success rate drops almost to zero after introducing the constraint on the cosine similarity of words.

The results are displayed in Table 8.1 and show that up to two-thirds of the attacks can be prevented using data augmentation. This indicates that adversarial examples for text classification are closely related to the data on which the model is fine-tuned. The attacks create examples that are out-of-distribution with respect to the training data. When we include similar data during training, the attack success rates drop substantially. Using our post-processing procedure, between 70% and 92% of the remaining attacks can additionally be reverted, resulting in attack success rates below 5% on AG News and Yelp. On IMDB, the attack success rates remains slightly higher, which is expected due to the longer input sequences. Nevertheless, for TextFooler, this corresponds to preventing over 94% of all successful attacks on every dataset. Compared to the mask-baseline, our post-processing procedure can revert significantly more attacks while having a smaller impact on the clean accuracy. Further, we can see that the post-processing step should always be preceded by data augmentation. While applying post-processing in isolation still reverts many attacks, the clean accuracy drops significantly, especially on AG News and IMDB.

Table 8.1 also shows that with the constraint on cosine similarity of words

Dataset	Method	Clean Acc. (%)	Attack Success Rate (%)		
			TextFooler	PWWS′	BERT-Attack′
AG News	N	94.57	84.99	16.38	20.72
	DA	**94.82**	52.37	10.73	18.61
	DA+PP	93.84 ± 0.07	**3.93** ± 0.41	**2.55** ± 0.31	**3.73** ± 0.29
	DA+MA_5	93.72 ± 0.12	14.11 ± 0.48	4.61 ± 0.41	7.52 ± 0.48
	N+PP	87.89 ± 0.16	10.32 ± 0.48	5.0 ± 0.31	5.59 ± 0.36
Yelp	N	**97.31**	90.47	33.26	49.53
	DA	97.10	29.79	10.52	16.49
	DA+PP	96.59 ± 0.06	**4.37** ± 0.39	**2.54** ± 0.15	**4.86** ± 0.33
	DA+MA_5	95.40 ± 0.10	10.23 ± 0.59	4.62 ± 0.36	7.38 ± 0.38
	N+PP	94.50 ± 0.08	6.07 ± 0.47	5.22 ± 0.48	7.35 ± 0.61
IMDB	N	93.77	98.16	65.77	88.44
	DA	**94.21**	48.31	29.49	40.91
	DA+PP	92.59 ± 0.06	**5.81** ± 0.45	**4.53** ± 0.26	**7.83** ± 0.37
	DA+MA_5	92.49 ± 0.12	12.05 ± 0.87	8.36 ± 0.36	13.0 ± 0.64
	N+PP	88.35 ± 0.09	10.52 ± 0.46	9.3 ± 0.39	13.3 ± 0.55

Table 8.1: Effectiveness of defense procedure for different attacks modified with constraint on cosine-similarity of words.

added, TextFooler is by far the most effective attack, at least before post-processing. There is a simple reason for this, TextFooler already has that constraint and is the only attack out of the four to choose its candidate set directly from the counter-fitted embedding used to calculate the cosine similarity. On the other end of the spectrum, BAE′'s attacks success rate drops to 0.32% on AG News, 0.41% on Yelp, and to 3.07% on IMDB. This is because the intersection of the set of words proposed by the MLM, the set of words with cosine similarity greater than 0.5, and the set of words keeping the USE score above 0.936 is small and leaves the attack not much room. A similar observation can be made for PWWS′ and BERT-Attack′, although not as pronounced.

However, there is one more reason why TextFooler is more effective compared to the other attacks, despite an additional constraint on the USE score. While attacking a piece of text, this constraint on the USE score is not checked between the current perturbed text s' and the original text s, but instead between the current perturbed text s' and the previous perturbed version s''. This means that by perturbing one word at a time, the effective USE score between s and s' can be a lot lower than the threshold suggests, as we have seen in Section 5.3. When discussing the effect of raising thresholds to higher levels, we do so by relying on TextFooler as the underlying attack because it is the most effective, but we adjust the constraint on the USE score to always compare to the original text. We believe this is the right way to implement this constraint, and more importantly, it is consistent with how we gathered data from Amazon Mechanical Turk.

Dataset	Method	Attack Success Rate (%)				
		TF_{cv50}	TF^{use88}_{cv50}	TF^{use85}_{cv70}	TF^{use90}_{cv70}	TF^{use90}_{cv80}
AG News	Normal	88.79	24.95	22.52	11.63	7.51
	DA	55.58	16.11	10.79	7.12	4.50
	DA+PP	4.49 ± 0.39	3.31 ± 0.28	2.07 ± 0.16	1.91 ± 0.17	0.99 ± 0.17
Yelp	Normal	91.40	49.22	42.59	25.18	11.09
	DA	38.46	13.74	10.34	7.78	2.87
	DA+PP	5.04 ± 0.35	3.9 ± 0.34	2.12 ± 0.21	2.28 ± 0.17	0.71 ± 0.13
IMDB	Normal	98.38	82.51	79.16	61.77	42.76
	DA	51.58	37.95	28.51	24.73	19.48
	DA+PP	5.81 ± 0.26	5.78 ± 0.4	3.56 ± 0.32	3.14 ± 0.28	2.67 ± 0.16

Table 8.2: Effectiveness of defense procedure for different combinations of thresholds.

8.1.1 Adjusted Thresholds

Table 8.2 shows the results of our defense procedure when the thresholds on TextFooler are adjusted. We use the same abbreviations as in Section 8.1 for the different methods. For the attacks, TF^{useY}_{cvX} corresponds to TextFooler with $cos_{cv}(w_i, w_i') \geq 0.X \ \forall i$ and $cos_{use}(s, s') \geq 0.Y$. A special case is TF_{cv50}, which corresponds to TextFooler without the constraint on the USE score.

Comparing the results of TF_{cv50} with the results from TextFooler in Table 8.1 confirms that the original implementation of the USE constraint only had a small impact. TF^{use88}_{cv50} is TextFooler with the same constraints as in the original implementation, but without allowing to drift away from the original text as discussed above. This already decreases the attack success rate significantly. Using data augmentation, we can decrease the attack success rates from 84.99 to 16.11 on AG News, from 90.47 to 13.74 on Yelp, and from 98.16 to 37.95 on IMDB. This shows that by preventing TextFooler from using that little trick and some data augmentation, we can decrease the attack success rate to values far from the ones suggested in their paper. When increasing the thresholds on the constraints (compare to Figure 6.2 and 6.3 to see that these are still not particularly strong constraints), it becomes even more evident that BERT is a lot more robust than work on attacks suggests. Especially if we allow for post-processing.

8.1.2 Comparing data augmentation with adversarial training

While adversarial training provides the model with data from the true distribution generated by an attack, our data augmentation procedure only approximates that distribution. The goal is to trade robustness for speed. However, it turns out that our procedure can even be superior to true adversarial training in some

Dataset	Method	Clean Acc. (%)	Time (h:min) / Epochs	Attack Success Rate (%)		
				TextFooler	PWWS′	BERT-Att.′
AG News	Normal	94.57	0:19 / 2	84.99	16.38	20.72
	DA	**94.82**	5:33 / 12	52.37	10.73	18.61
	ADV	92.83	160:15 / 12	**34.54**	**6.50**	**9.38**
	ADV_{naive}	94.26	45:14 / 2	56.20	12.50	17.44
Yelp	Normal	**97.31**	0:32 / 2	90.47	33.26	49.53
	DA	97.10	9:08 / 12	**29.79**	**10.52**	**16.49**
	ADV	95.94	107:56 / 5	59.52	14.64	25.52
	ADV_{naive}	96.65	56:53 / 2	95.12	33.09	47.61
IMDB	Normal	93.77	0:17 / 2	98.16	65.77	88.44
	DA	94.21	5:31 / 12	48.31	29.49	40.91
	ADV	92.00[1]	– / 3[1]	75.3[1]	–	–
	ADV_{naive}	93.16	34:19/ 2	100.00	62.75	88.79

Table 8.3: Comparison of data augmentation and adversarial training.

cases. We compare to two different strategies for adversarial training. ADV_{naive} denotes the simplest procedure for adversarial training in text classification: collect adversarial examples on the training set and then train a new model on the extended dataset consisting of both adversarial examples and original training data. We used TextFooler to collect these adversarial examples. On the complete training set, this resulted in 103'026 adversarial examples on AG News, 179'335 adversarial examples on Yelp, and 23'831 adversarial examples on IMDB. For a more sophisticated version for adversarial training, we follow [21] by creating adversarial examples on-the-fly during training. We denote this method as ADV (corresponds to ADV in their paper).

A comparison of the results is shown in Table 8.3. Interestingly, ADV_{naive} did not result in an improvement on Yelp and IMDB. We hypothesize that this is because Yelp and IMDB are easier to attack, resulting in weaker training data for the extended dataset. For example, 26% of the created adversarial examples on Yelp differ by only one or two words from the original text, on AG News this holds for just 11% of the adversarial examples. Furthermore, the average word replace rate on Yelp is 16% compared to 24% on AG News. When the adversarial examples differ only by a few words from the original text, instead of getting more robust, the model overfits to the training data. On IMDB, this problem is even more extreme, and we did not manage to train a model according to [21], hence we used the available results from their paper. That trade off between robustness and overfitting when training many epochs is likely the reason why we achieve better robustness compared to adversarial training on two out of three datasets. To be fair, it must be mentioned that we only trained ADV until convergence on AG News and restricted the training to 5 epochs on Yelp due to computational

[1]Results taken from [21].

constraints. Overall, lower computation time is precisely the biggest advantage of our method. Considering that the training data increases by a factor of two, the overhead per epoch is only around 50% compared to normal training. Compared to ADV, we reach a speedup per epoch of almost 30x.

Chapter 9

Conclusion

9.1 Limitations

In practice, the post-processing step cannot be decoupled from a black-box attack. It would be interesting to see how successful an attack is when the whole system, including post-processing, is regarded as a single black-box model. We hypothesize that it would remain challenging because the attacker can rely much less on its search method for finding the right words to replace.

The method is also not applicable if a deterministic answer is required. However, in many applications such as spam filters or fake news detection, we are only interested in making a correct decision as often as possible while being robust to a potential attack.

9.2 Conclusion

Using a human evaluation, we have shown that most perturbations introduced through adversarial attacks do not preserve semantics. This is contrary to what is generally claimed in papers introducing these attacks. We believe the main reason for this discrepancy is that researchers working on attacks pay more attention to reaching high attack success rates compared to creating semantic preserving adversarial examples. However, in order to find meaningful adversarial examples that could help us better understand current models, we need to get away from that line of thinking. We believe a 10-20% attack success rate with valid adversarial examples and a good analysis of them is much more valuable than an 80-90% attack success rate by introducing nonsensical words. We hope this work encourages researchers to think more carefully about appropriate perturbations to text which do not change semantics.

Our results on data augmentation show that a significant amount of adversarial examples can be prevented when including perturbations during training that could stem from an attack. It is debatable whether changing 40% of the words with a randomly chosen word from a candidate set still constitutes a valid input,

but this is only necessary because the attacks have that amount of freedom. The more appropriate the allowed perturbations for an attack, the more appropriate is our data augmentation procedure, which can easily be adapted for other candidate sets (see also Appendix A.3 with results for the WordNet candidate set used in PWWS). Compared to adversarial training, our method scales to large datasets and multiple epochs of training while achieving remarkable robustness, making it an excellent baseline defense method for researchers working on new attacks and defenses. The post-processing step completes our defense procedure and shows that attacks can largely be prevented in a probabilistic setting without a severe impact on the clean accuracy. In practice, this means that most attacks can at least be detected. Whether or not this two-step procedure will prevent the same amount of attacks when the whole model is considered a probabilistic black-box is up for future investigation.

Bibliography

[1] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *ICLR*, Y. Bengio and Y. LeCun, Eds., 2014.

[2] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *ICLR*, Y. Bengio and Y. LeCun, Eds., 2015.

[3] N. Carlini and D. Wagner, "Audio adversarial examples: Targeted attacks on speech-to-text," in *IEEE Security and Privacy Workshops (SPW)*, 2018.

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *NAACL-HLT*, 2019.

[5] N. Dalvi, P. Domingos, S. Sanghai, and D. Verma, "Adversarial classification," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2004, pp. 99–108.

[6] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 ieee symposium on security and privacy (sp)*. IEEE, 2017, pp. 39–57.

[7] N. Papernot, P. McDaniel, A. Swami, and R. Harang, "Crafting adversarial input sequences for recurrent neural networks," in *MILCOM IEEE Military Communications Conference*, 2016.

[8] M. Alzantot, Y. Sharma, A. Elgohary, B.-J. Ho, M. Srivastava, and K.-W. Chang, "Generating natural language adversarial examples," in *EMNLP*, 2018.

[9] H. Zhang, H. Zhou, N. Miao, and L. Li, "Generating fluent adversarial examples for natural languages," in *ACL*, 2019.

[10] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *The journal of chemical physics*, vol. 21, no. 6, pp. 1087–1092, 1953.

[11] W. K. Hastings, "Monte carlo sampling methods using markov chains and their applications," 1970.

[12] S. Ren, Y. Deng, K. He, and W. Che, "Generating natural language adversarial examples through probability weighted word saliency," in *ACL*, 2019.

[13] D. Jin, Z. Jin, J. T. Zhou, and P. Szolovits, "Is bert really robust? a strong baseline for natural language attack on text classification and entailment," in *AAAI*, 2020.

[14] L. Li, R. Ma, Q. Guo, X. Xue, and X. Qiu, "Bert-attack: Adversarial attack against bert using bert," in *EMNLP*, 2020.

[15] S. Garg and G. Ramakrishnan, "Bae: Bert-based adversarial examples for text classification," in *EMNLP*, 2020.

[16] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *International Conference on Learning Representations*, 2018.

[17] X. Wang, Y. Yang, Y. Deng, and K. He, "Adversarial training with fast gradient projection method against synonym substitution based text attacks," in *UAI*, 2021.

[18] X. Wang, Y. Yang, Y. Deng, and K. He, "Adversarial training with fast gradient projection method against synonym substitution based text attacks," in *AAAI*, 2021.

[19] N. Mrkšić, D. Ó. Séaghdha, B. Thomson, M. Gasic, L. M. R. Barahona, P.-H. Su, D. Vandyke, T.-H. Wen, and S. Young, "Counter-fitting word vectors to linguistic constraints," in *NAACL-HLT*, 2016.

[20] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *EMNLP*, 2014.

[21] Z. Meng, Y. Dong, M. Sachan, and R. Wattenhofer, "Self-supervised contrastive learning with adversarial perturbations for robust pretrained language models," *arXiv preprint arXiv:2107.07610*, 2021.

[22] R. Jia, A. Raghunathan, K. Göksel, and P. Liang, "Certified robustness to adversarial word substitutions," in *EMNLP-IJCNLP*, 2019.

[23] P.-S. Huang, R. Stanforth, J. Welbl, C. Dyer, D. Yogatama, S. Gowal, K. Dvijotham, and P. Kohli, "Achieving verified robustness to symbol substitutions via interval bound propagation," in *EMNLP-IJCNLP*, 2019.

[24] J. Morris, E. Lifland, J. Lanchantin, Y. Ji, and Y. Qi, "Reevaluating adversarial examples in natural language," in *EMNLP Findings*, 2020.

[25] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *NeurIPS*, 2017.

[26] M. Schuster and K. Nakajima, "Japanese and korean voice search," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2012, pp. 5149–5152.

[27] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler, "Aligning books and movies: Towards story-like visual explanations by watching movies and reading books," in *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 19–27.

[28] F. Hill, R. Reichart, and A. Korhonen, "Simlex-999: Evaluating semantic models with (genuine) similarity estimation," *Computational Linguistics*, vol. 41, no. 4, pp. 665–695, 2015.

[29] D. Cer, Y. Yang, S. Kong, N. Hua, N. Limtiaco, R. S. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, B. Strope, and R. Kurzweil, "Universal sentence encoder for english," in *EMNLP*, E. Blanco and W. Lu, Eds., 2018.

[30] M. Iyyer, V. Manjunatha, J. Boyd-Graber, and H. Daumé III, "Deep unordered composition rivals syntactic methods for text classification," in *Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (volume 1: Long papers)*, 2015, pp. 1681–1691.

[31] R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler, "Skip-thought vectors," in *Advances in neural information processing systems*, 2015, pp. 3294–3302.

[32] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," *NeurIPS*, 2015.

[33] J. Morris, E. Lifland, J. Y. Yoo, J. Grigsby, D. Jin, and Y. Qi, "Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp," in *EMNLP System Demonstrations*, 2020.

[34] Z. Meng and R. Wattenhofer, "A geometry-inspired attack for generating natural language adversarial examples," in *COLING*, 2020.

Appendix A

Appendix

A.1 Details for human evaluation

We relied on workers with at least 5000 HITs and over 98% success rate. For the word-pairs, we showed the workers 100 pairs of words in a google form. In order to ensure a good quality of work, we included some hand designed test cases at several places and rejected workers with strange answers on these word-pairs. These test-cases were [*good*, *bad*], [*help*, *hindrance*] (expected answer "Strongly Disagree" or "Disagree") and [*sofa*, *couch*], [*seldom*, *rarely*] (expected answer "Strongly Agree" or "Agree"). In a first test run, surprisingly many workers agreed on antonyms like good and bad, which is why we additionally included a note with an example and emphasized that this is about whether the meaning is preserved and not about whether both words fit into the same context. Workers were paid 2.0$ for one HIT with 100 pairs and 4 test-cases. For the words with context, we used the amazon internal form because it allowed for a clearer presentation of the two text fragments. We always presented 5 pairs of text fragments in one HIT and rejected workers that submitted the hit within less than 60s to ensure quality. Workers were paid 0.5$ for one HIT with 5 pairs. Screenshots of the two forms can be found in Figure A.1.

A.2 Number of versions in post-processing

Creating N verions during post-processing increases the effective batch size during inference by a factor of N. Hence creating as few versions as possible is desirable for keeping the inference time low. In order to understand the impact of the number of versions N created during the post-processing step, we can make the following analysis: Let us consider the augmented inputs as instances of a discrete random variable X. For $x \in X$ and a classification problem with K classes, let $l_{correct}(x)$ denote the value of the logit corresponding to the correct label and $l_j(x)$ denote the value of the j-th logit corresponding to a wrong label, such that $j \in \{1, ..., K-1\}$. We are only interested in the differences

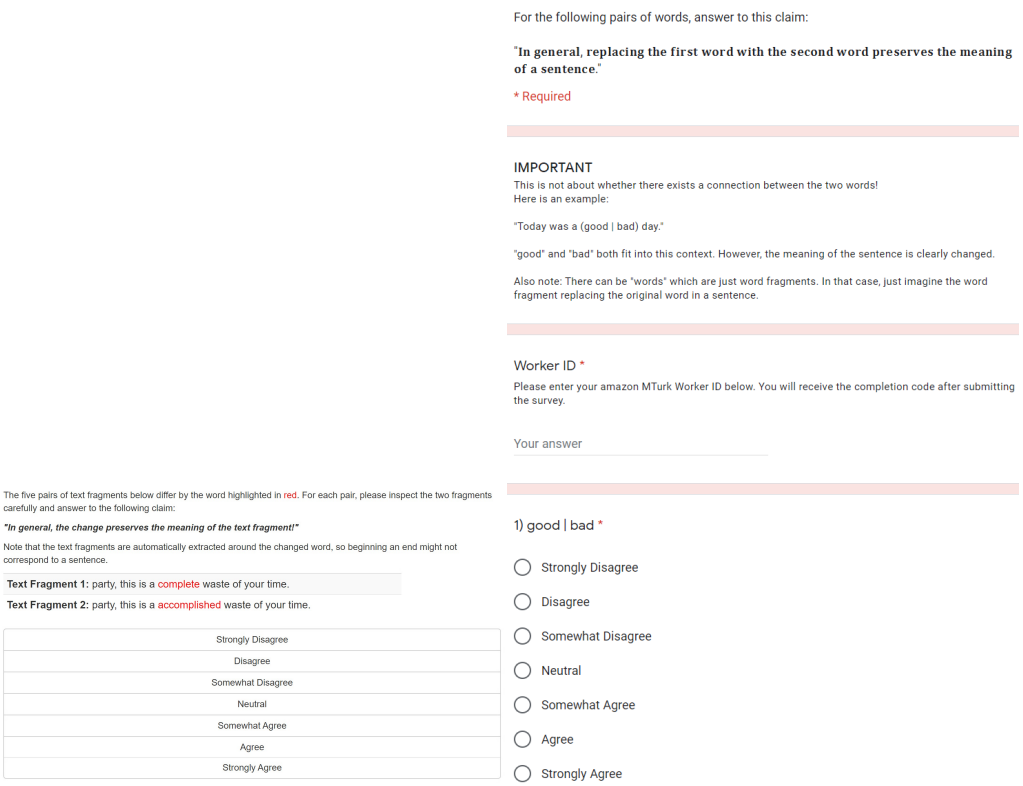

Figure A.1: Screenshot of the human evaluation used to evaluate words with context (left) and screenshot of the Google form used to evaluate similarity of words (right).

$g_j(x) = l_{correct}(x) - l_j(x)$. Ideally, we would like to make a decision based on the expectations of $g_j(X)$. An attack should be reverted if and only if

$$\mathrm{E}[g_j(X)] = \sum_{x \in X} g_j(x) p_X(x) \geq 0 \quad \forall j, \tag{A.1}$$

where $p_X(x) = \frac{1}{|X|}$. Because we cannot enumerate over all instances x, we approximate this with sums over just N instances

$$\sum_{i=1}^{N} \frac{g_j(x_i)}{N} \geq 0 \quad \forall j. \tag{A.2}$$

These are unbiased estimates of the expectations in (A.1) for any choice of N. By multiplying with N and plugging in the definition of $g_j(x)$, it can be verified that a decision based on (A.2) reverts the same attacks as a decision based on (7.3). The expectation estimates become more and more accurate as we increase N. Since we are making a discrete decision based on whether the expectations are ≥ 0, the estimate is more likely to be correct with more samples. If we assume

Dataset	Number of Versions	Reverted Attacks (Mean ± Std) (%)		
		TextFooler	PWWS′	BERT-Attack′
AG News	4	92.13 ± 0.65	75.39 ± 3.35	78.7 ± 1.94
	8	92.49 ± 0.79	76.27 ± 2.87	79.94 ± 1.54
	16	92.81 ± 0.53	78.24 ± 1.95	80.17 ± 0.85
	32	92.97 ± 0.24	76.57 ± 1.61	81.07 ± 0.88
Yelp	4	83.94 ± 1.49	74.31 ± 3.28	68.56 ± 3.02
	8	85.33 ± 1.32	75.88 ± 1.4	70.5 ± 1.97
	16	85.81 ± 1.26	76.37 ± 1.88	70.81 ± 1.12
	32	86.26 ± 0.74	76.96 ± 0.79	71.31 ± 2.16
IMDB	4	87.2 ± 1.13	84.19 ± 1.43	80.36 ± 1.27
	8	87.96 ± 0.92	84.62 ± 0.88	80.85 ± 0.91
	16	87.86 ± 0.77	85.2 ± 0.68	82.09 ± 0.78

Table A.1: Effectiveness of post-processing for different number of versions.

that the true expectation is positive in most cases, this means we can generally expect a higher number of reverted attacks for higher N. Being more precise on the estimate also means we generally tend to make the same decision every time on the same example, therefore reducing the variance in the reverted attack rate. Table A.1 shows results on reverted attacks for 4, 8, 16 and 32 versions (4,8, and 16 on IMDB because of memory constraints) and generally confirms this. However, the results are already quite good with just four versions, so this is a trade-off between speed and accuracy, as creating N versions increases the batch size during inference by a factor N.

A.3 Defense Procedure WordNet

One could argue that the success rates of PWWS and BERT-Attack in Section 8.1 are artificially kept low by introducing a new constraint on the cosine similarity, therefore shrinking the candidate sets of PWWS and BERT-Attack. We choose that candidate set in our defense procedure with the results from Chapter 6 in mind, where TextFooler receives the best scores. Furthermore, having a flexible threshold on the cosine similarity of words allows for adjusting the size of the candidate set as needed. However, our procedure can also be adapted to other candidate sets. To show this, we propose a WordNet variant designed for the candidate set of PWWS. The changes are the following:

- In step 3. of the Data Augmentation procedure, a candidate set is built out of all synonyms from WordNet, and a replacement is sampled uniformly at random. We denote the new data augmentation procedure as DA_{wn}.
- In step 1. of the post-processing procedure, the candidate set again consists of all synonyms from WordNet. We denote the new post-processing

Dataset	Clean Acc. (%)			Attack Success Rate (%)		
	N	DA_{wn}	DA_{wn} + PP_{wn}	N	DA_{wn}	DA_{wn} + PP_{wn}
AG News	94.57	94.99	94.40 ± 0.05	64.95	27.61	1.49 ± 0.21
Yelp	97.31	97.26	96.95 ± 0.04	92.23	29.33	4.84 ± 0.43
IMDB	93.77	94.34	92.85 ± 0.04	98.70	51.91	3.85 ± 0.28

Table A.2: Attack success rates of PWWS applied to a normal model, a model trained using WordNet data augmentation and a model trained using WordNet data augmentation with additional post-processing.

Dataset	Method	Clean Acc. (%)	Reverted Attacks (Mean ± Std) (%)		
			TextFooler	PWWS′	BERT-Attack′
AG News	MA_5	93.62	73.05 ± 0.92	57.06 ± 3.82	59.6 ± 2.6
	MA_{10}	92.14	72.13 ± 1.5	57.55 ± 2.77	58.59 ± 2.53
	MA_{20}	87.30	65.02 ± 1.77	54.02 ± 2.99	52.99 ± 2.41
	MA_{30}	76.25	55.64 ± 1.62	47.84 ± 3.45	46.55 ± 3.16
Yelp	MA_5	95.19	65.67 ± 1.97	56.08 ± 3.45	55.25 ± 2.1
	MA_{10}	93.98	68.93 ± 1.29	59.02 ± 2.39	56.31 ± 1.78
	MA_{20}	90.53	69.0 ± 1.66	57.75 ± 1.67	55.75 ± 1.65
	MA_{30}	86.91	67.44 ± 1.04	56.37 ± 2.49	53.94 ± 0.79
IMDB	MA_5	92.47	75.05 ± 1.8	71.65 ± 1.2	68.22 ± 1.55
	MA_{10}	89.90	71.99 ± 0.77	69.64 ± 1.16	64.39 ± 1.16
	MA_{20}	83.51	65.93 ± 0.85	64.19 ± 1.39	57.55 ± 0.65
	MA_{30}	78.76	62.17 ± 0.62	62.58 ± 0.54	53.8 ± 1.14

Table A.3: By masking random tokens instead of exchanging words, many attacks can be reverted. However, the clean accuracy drops.

procedure as PP_{wn}

We sample uniformly at random to get a better coverage of the WordNet synonyms. Note that directly applying (7.2) would not work because the cosine similarity of words from WordNet can be below zero. The results of this adjusted procedure, with PWWS as the attacker, are shown in Table A.2. It can be seen that our procedure works equally well for a different candidate set and reduces the attack success rate of PWWS significantly.

A.4 Baseline for post-processing

Instead of replacing words with other words in Step 2 of our defense procedure, one could also think of other ways of perturbing the adversarial examples to flip the label back to the correct one. To show that our method is superior to simple perturbations, Table A.3 shows the results of a baseline procedure in which we

replace randomly chosen words with the [MASK] token. MA_x to a procedure in which we replace x percent of the words with the [MASK] token. It can be seen that indeed a large portion of attacks can be prevented using that procedure. However, it only works with small percentages of masked words. When more words are masked, the clean accuracy drops substantially. This is contrary to our procedure, in which we exchange 40% of the words with just a minimal decrease in accuracy. We included the best performing version, MA_5, as a baseline in the main part of this thesis.

A.5 BERT-Embeddings

Figures A.2, A.3, and A.4 show the results for the analysis in Section 5.4 with adversarial examples from PWWS, BERT-Attack, and BAE. While the individual plots all look slightly different, the general conclusions remain the same.

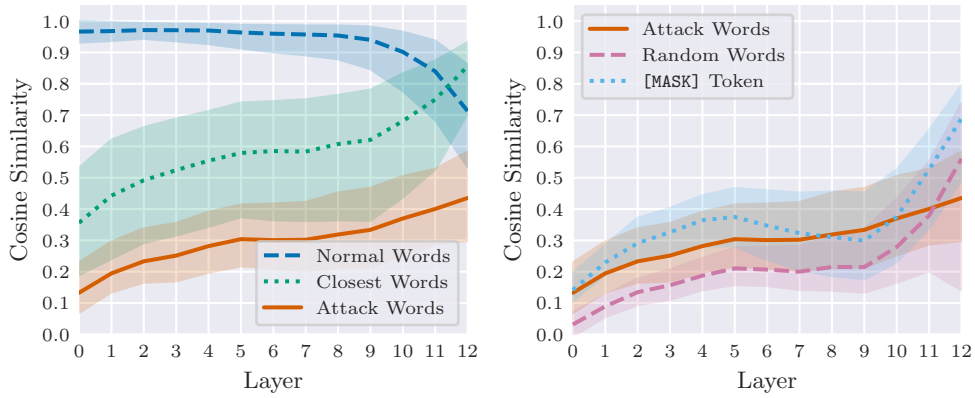

Figure A.2: Analysis from Section 5.4 with adversarial examples from PWWS.

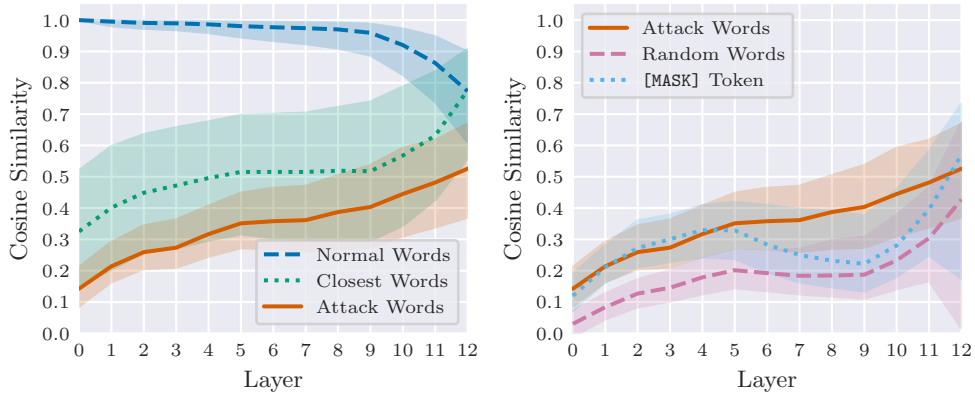

Figure A.3: Analysis from Section 5.4 with adversarial examples from BERT-Attack.

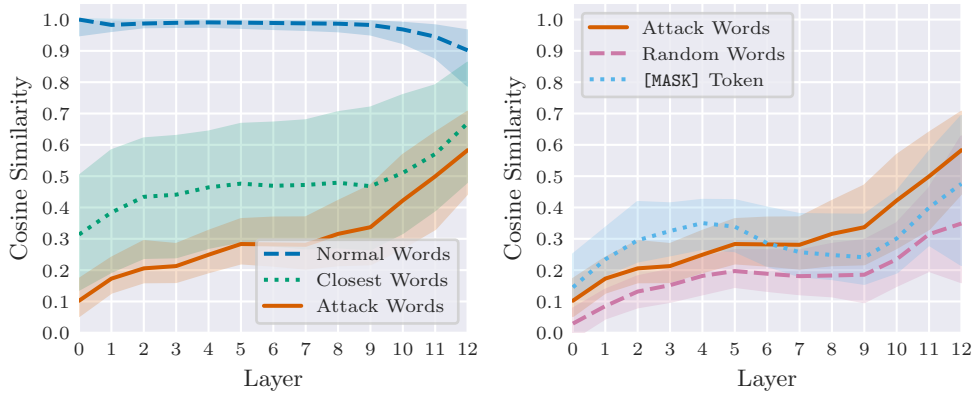

Figure A.4: Analysis from Section 5.4 with adversarial examples from BAE.

A.6 Sentence Simlarity Examples

Table A.4 shows text fragments from original text and adversarial examples, extracted with a window size of 11, and the corresponding USE-Score. If there are less than 11 words, the window was extracted at the beginning or end of an adversarial example.

Orig. Fragment	a lil landy and youll be set	USE: 0.29
Adv. Fragment	a noo melendez and theres was determining	
Orig. Fragment	I felt the entire experience was deceptive	USE: 0.41
Adv. Fragment	I cru the holistic expertise was underhand	
Orig. Fragment	experience was deceptive misleading and manipulative of people in a compromised	USE: 0.51
Adv. Fragment	expertise was underhand fooling and cunning of pueblo in a prevented	
Orig. Fragment	and another 23 to get open and end up paying 180	USE: 0.61
Adv. Fragment	and another 23 to acquired inaugural and completion up revenues 180	
Orig. Fragment	a Service Representative. When finally served she told us in fact	USE: 0.70
Adv. Fragment	a Serves Representative. When arguably working she spoken us in doing	
Orig. Fragment	But it was still good Sarah was our waitress and did	USE: 0.75
Adv. Fragment	But it was albeit satisfactory Rebeca was our waitress and has	
Orig. Fragment	joke. Maybe its because I live in Fountain Hills They shouldn't	USE: 0.80
Adv. Fragment	laughter. Maybe its because me reside in Fountain Hills They shouldn't	
Orig. Fragment	and the butchers have no clue. They are not at all	USE: 0.85
Adv. Fragment	and the butchers have no conundrum. They are not at all	
Orig. Fragment	place was on very high cosy at first but sweltering the	USE: 0.90
Adv. Fragment	place was on very high lounging at first but sweltering the	
Orig. Fragment	was and nothing was labeled. Even the bars were tough to	USE: 0.95
Adv. Fragment	was and nothing was labeled. However the bars were tough to	
Orig. Fragment	is that Paris is a good hotel and there service was	USE: 0.99
Adv. Fragment	is that Paris is a decent hotel and there service was	

Table A.4: Text fragments for different values of USE scores. Created with TextFooler.

A.7 Randomly Sampled Adversarial Examples

Tables A.5, A.6, A.7, and A.8 on the following pages show two randomly sampled adversarial examples from all four attacks on Yelp and AG News. Note that they are not hand-picked and not adjusted in any way. Capitalization has no impact since we use an uncased model.

Original Text	The owners are very rude.. food is OK..not the best	100% negative
Adversarial Example	there games are very friendly.. food is OK..not the bad	50% positive
Original Text	Everything about these subs are great: the meat, the toppings, and even the bread is delicious! Its worth a trip just to read the signs and bumper stickers.	100% positive
Adversarial Example	information about these subs are great: the meat, the toppings, and even the bread is edible! Its worth a trip just to read the signs and bumper stickers.	72% negative
Original Text	Had a great time. Great service. Too bad its in Snobdale I mean Scottsdale. Some 60+ old man killed it for me while he was attempting to hit on me in front of his wife and my husband and son. Can't wait for the Gilbert location to open (where we live) to get away from the Scottsdale vide!!!	73% positive
Adversarial Example	Had a good time. Great service. Too bad its in Snobdale I mean Scottsdale. Some 60+ old man killed it for me while he was attempting to hit on me in front of his wife and my husband and son. ma wait for the Gilbert location to open (where we live) to get away from the Scottsdale vide!!!	50% negative
Original Text	Food was ok, but very rude staff here, never coming back again	100% negative
Adversarial Example	Food was ok, but very bourne staff here, fully coming back again	96% positive

Table A.5: Randomly sampled adversarial examples on Yelp. Attacker from top to bottom: BERT-Attack, BERT-Attack, BAE, BAE

Original Text	Great fish tacos!	100% positive
Adversarial Example	Overwhelming fish blocking!	98% negative
Original Text	I took my wife, daughter - 7, son - 3, and mother-in-law (not by choice - lol) and we had a great time. The kids loved looking in the stores at all the different and somewhat authentic items. My daughter really enjoyed the gold mine tour and my sun loved the train ride. A definite must see if you want a taste of a mining town in the old west.	100% positive
Adversarial Example	I fired my wife, daughter - 7, son - 3, and mother-in-law (not by choice - thats) and we received a gargantuan time. Both kids loved attempt in the stores at all the other and somewhat veritable issues. My daughter really enjoyed the gold mine roving and my sun loved the train ride. A clearer must see if you want a taste of a mining town in the old occidental.	53% negative
Original Text	Recently Else's menu has changed for the better. The food is out of this world, and I'm actually craving those spicy shrimps. The price is unbeatable and the drinks are great (great drink specials too).\nIt's the best to be for good food, great conversation and atmosphere.	100% positive
Adversarial Example	Recently Else's menu has changed for the better. The food is out of this humanity, and I'm actually craving those spicy shrimps. The price is unbeatable and the drinks are great (great drink specials too).\nIt's the unspoilt to be for good food, bully conversation and atmosphere.	59% negative
Original Text	I've tried this place three times now. Sorry, but there won't be a fourth. Hard to understand the menu. We love fun rolls. I think pictures would help. The food is just not good. Maybe a smaller menu with great items would help this place. I dunno. Ambiance is eh. Doesn't feel very Japanese or anything really.	100% negative
Adversarial Example	I've tried this place three times now. Sorry, but there won't be a fourth. Hard to understand the menu. We dear fun rolls. I think pictures would help. The nutrient is just not skilful. Maybe a smaller menu with great items would help this place. I dunno. Ambiance is eh. Doesn't feel very Japanese or anything really.	68% positive

Table A.6: Randomly sampled adversarial examples on Yelp. Attacker from top to bottom: TextFooler, TextFooler, PWWS, PWWS

Original Text	2 U.S. Factory Growth Eases NEW YORK (Reuters) - Expansion in the U.S. factory sector slowed in August as higher costs for energy and raw materials squeezed manufacturers, a report showed on Wednesday, but analysts said growth remained relatively robust.	100% Business
Adversarial Example	U.S. it Growth Eases NEW europe (Reuters) - Expansion in the U.S. factory sector slowed in August as higher costs for energy and raw materials squeezed manufacturers, a report showed on Wednesday, but analysts said growth remained relatively robust.	66% Sci/Tech
Original Text	Lions have their work cut out for them against Manning You see it every time Indianapolis Colts quarterback Peyton Manning steps to the line of scrimmage before taking the snap. It #39;s like he #39;s going through his own little workout routine.	98% Sports
Adversarial Example	she have their work split out for them against scheduling You see it every time Indianapolis Colts manning prescription policy falls to the line of scriminger before hitting the snap. It #39;s like he #nut;s it through his own little workout routine.	52% Business
Original Text	Semiconductor Manufacturing to Boost Capacity by Half (Update2) Semiconductor Manufacturing International Corp., China #39;s biggest supplier of made-to-order chips, said its factory capacity will rise by more than half in the second half as the company brings more plants on line.	53% Sci/Tech
Adversarial Example	Semiconductor Manufacturing to Boost Capacity by Half (Update2) Semiconductor Manufacturing International Corp., eries #39;s biggest supplier of made-to-order chips, said its factory capacity will rise by more than half in the second half as the company brings more plants on line.	82% Business
Original Text	Faces From The 1929 Crash NEW YORK - The people who will forever be associated with the Great Crash of 1929 were all white, male and wealthy, but their occupations and ethics varied considerably.	79% Business
Adversarial Example	Faces From The 1929 Crash NEW YORK - The people who will forever be associated with the relli Crash of 1929 were all white, male and ish, but their occupations and ethics varied considerably.	63% World

Table A.7: Randomly sampled adversarial examples on AG News. Attacker from top to bottom: BERT-Attack, BERT-Attack, BAE, BAE

Original Text	2 Ex-Officers Nabbed in Venezuela Slaying National Guard troops arrested two brothers Friday in connection with a state prosecutor's killing, just days after two suspects in the car bombing case were shot dead by police, authorities said.	100% World
Adversarial Example	2 Ex-Officers Arrested in Chavez Whack National Guard troops arrested two brothers Friday in connection with a state prosecutor's whack, just days after two suspects in the car raiding case were shot dead by police, authorities said.	35% Business
Original Text	Manchester United admits paying 11m to transfer middle-men The role of agents in multimillion-pound football transfer deals came under fresh scrutiny yesterday after Manchester United revealed payments of 11m to middle-men for their help in signing players.	98% Sports
Adversarial Example	Cheshire United understands paying 11m to transfer middle-men The functionality of agents in multimillion-pound football transfer deals came under fresh scrutiny yesterday after Manchester United illustrated payments of 11m to middle-men for their subsidized in subscription gamers.	55% Business
Original Text	UN Council Votes Ivory Coast Arms Embargo (Reuters) Reuters - The U.N. Security Council on\Monday imposed an immediate arms embargo on Ivory Coast and\voted to punish key government and rebel leaders with\additional sanctions next month.	100% World
Adversarial Example	UN Council Votes Ivory seashore sleeve Embargo (Reuters) Reuters - The U.N. Security Council on\Monday imposed an immediate arms embargo on Ivory seashore and\voted to punish key government and rebel leaders with\additional sanctions next month.	54% Sci/Tech
Original Text	Cox Communications forms committee to advise on buyout Cox Communications Inc. #39;s board of directors has formed a special committee of independent directors to consider Cox Enterprises Inc. #39;s proposal to take the company private in a $8 billion stock buyout.	100% Business
Adversarial Example	cyclooxygenase Communications forms committee to advise on buyout cyclooxygenase communicating Inc. #39;s board of directors has formed a special committee of freelancer directors to consider coxswain Enterprises Inc. #39;s proposal to take the company private in a $8 billion old-hat buyout.	73% Sci/Tech

Table A.8: Randomly sampled adversarial examples on AG News. Attacker from top to bottom: TextFooler, TextFooler, PWWS, PWWS