Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Self-Supervised Contrastive Learning With Adversarial Perturbations for Robust Pretrained Language Models

Master's Thesis

Yihan Dong

yihdong@student.ethz.ch

Distributed Computing Group
Computer Engineering and Networks Laboratory
ETH Zürich

Supervisors:
Prof. Dr. Roger Wattenhofer
Prof. Dr. Mrinmaya Sachan, Zhao Meng

August 26, 2021

Acknowledgements

I have received a great deal of support and assistance when writing this thesis.

I would first like to thank my supervisor, Professor Roger Wattenhofer. Your advice for the paper and the valuable suggestions you provided after my mid-term presentation made the content of my thesis more complete.

I would like to acknowledge my co-supervisor, Professor Mrinmaya, Sachan. Your insightful feedback on the thesis topic and experimental approach pushed my thinking and my work to a higher level.

I would also like to thank Zhao Meng for his valuable guidance throughout my thesis. You gave me a lot of advice on the experimental part of my thesis, and provided me with enough computing resources to help me complete my thesis successfully. I would also like to thank you very much for your guidance in my mid-term and final presentations.

Abstract

This thesis improves the robustness of the pretrained language model BERT against word substitution-based adversarial attacks by leveraging self-supervised contrastive learning with adversarial perturbations. One advantage of our method compared to previous works is that it is capable of improving model robustness without using any labels. Additionally, we also design an adversarial attack for word-level adversarial training on BERT. The attack is efficient, allowing adversarial training for BERT on adversarial examples generated *on the fly* during training. Experimental results on four datasets show that our method improves the robustness of BERT against four different word substitution-based adversarial attacks. Furthermore, to understand why our method can improve the model robustness against adversarial attacks, we study vector representations of clean examples and their corresponding adversarial examples before and after applying our method. As our method improves model robustness with unlabeled raw data, it opens up the possibility of using large text datasets to train robust language models.

Contents

Chapter 1

Introduction

1.1 Background and Motivation

Pretrained language models such as BERT have had a tremendous impact on various NLP tasks. Despite their superior performance on downstream tasks, researchers have demonstrated that these models are still vulnerable to adversarial attacks, which fool the model by adding imperceptible-to-human perturbations to the model inputs [1, 2, 3], such as synonym substitution for words in the input sentence as shown in table 1.1.

A prevailing method to improve the model robustness against adversarial attacks is adversarial training [4]. In adversarial training, each batch of data is augmented by adding corresponding adversarial examples of the same batch. These adversarial examples are fed into the model for training along with the original samples.

In NLP, adversarial training in the input space is hard, as existing natural language adversarial attacks are too slow to generate adversarial examples *on the fly* during training [2, 3, 5]. Due to the limitations of attack speed, it is common practice to use pre-generated adversarial samples in training, which does not make effective use of adversarial training as the adversarial samples never change as the robustness of the model increases. Although there has been previous work exploring efficient input space adversarial training for NLP [6], few have paid attention to input space adversarial training for modern pre-trained language models like BERT. Furthermore, unlike computer vision, where one can use standard datasets consisting of tens of millions of images (for example, ImageNet [7]) for adversarial training, NLP tasks typically have much fewer labeled examples, making it harder to improve the robustness of NLP models with adversarial training.

Contrastive learning, which was first proposed to improve model performance on various downstream vision tasks [8, 9, 10], has gained attention from the NLP community. Recent works in NLP have shown that contrastive learning can help improve downstream text classification performance [11, 12]. While the goal of contrastive learning is to make the vector representations of similar examples closer, recent development in computer vision has indicated that it is beneficial to use adversarial perturbations to create "hard" positives [13], which are similar to the original examples in the input space, but in the meantime are dissimilar in the vector space.

Original	
(Negative)	It has a stunning lack of even rudimentary traces of realism . almost every war movie cliche appears in this film and is done badly. on the other hand , i wouldn't have watched it to the end if it hadn' t been so remarkably bad that it amused me.
Adversarial	
(Positive)	It has a stunning lack of even joyless traces of realism . almost every war movie cliche appears in this film and is done erroneously. on the other hand , i wouldn't have watched it to the end if it hadn' t been so remarkably bad that it flabbergasted me.

Table 1.1: An adversarial sample generated by adding imperceptible-to-human perturbations to the model inputs. Blue words in the original example are replaced by red words in the adversarial example. In this example, the model is fooled by making synonym substitutions for words, and it mistakes a negative sample for a positive one.

However, generating such examples for natural language is hard, due to the discrete nature of human languages and the inefficiency of existing natural language adversarial attacks. Additionally, most existing natural language adversarial attacks are designed for supervised classification tasks. Therefore, how to design an adversarial attack for contrastive learning in NLP is still unclear. A recent work [14] leverages adversarial perturbations during contrastive learning for conditional text generation. However, their work neither address general natural language understanding tasks, nor does their work pay attention to robustness against attacks. Hence, how to combine adversarial perturbations with contrastive learning to improve robustness for general NLP understanding tasks remains an open problem. Furthermore, it remains unclear whether this approach is really effective in improving the robustness of NLP models, especially pre-trained language models.

1.2 Objective and Contributions

In this thesis, our objective is to improve the robustness of pretrained language models against adversarial attacks by combining contrastive learning with adversarial perturbations. We do this by creating an efficient adversarial attack for BERT that allows us to generate word-level adversarial attack samples *on the fly* in training. Our attack is efficient in that it is capable of generating multiple adversarial examples in parallel, while previous attack typically generates adversarial examples one by one [2, 3, 5]. Experimental results show that our method can improve the robustness of pretrained language models *without looking at the labels*.

Our contributions in this thesis can be summarized as follows:

1. We improve the robustness of the pretrained language model BERT against word substitution-based adversarial attacks by leveraging self-supervised contrastive learning. Experiments show that our method improves model robustness even without accessing the annotated labels.

2. We create an efficient way to perform adversarial attacks. It can be used not only in self-supervised contrastive learning, but can also be extended to supervised tasks, enabling word-level adversarial training on BERT. Instead of conducting adversarial training on pre-generated adversarial examples, we conduct adversarial training by generating the adversarial examples *on the fly* for each batch.

3. Furthermore, our study on the vector representations of clean examples and their corresponding adversarial examples explains why our method improves model robustness.

4. We give future directions for NLP contrastive learning. While previous work for NLP contrastive learning focuses on in-domain setting, where researchers use the same datasets during contrastive pretraining and downstream fine tuning, our experiments in Section 5.4 show that using out-of-domain datasets also improves model robustness on downstream tasks. Therefore, future work on self-supervised contrastive learning can generalize to large-scale datasets like BookCorpus [15].

Chapter 2

Related Work

2.1 Adversarial Attacks and Robustness

Researchers have proposed various natural language adversarial attacks. For instance, [1] fool a machine reading model by adding adversarial sentences to the original contexts. [2] propose a word-level adversarial attack using a genetic algorithm. To improve model robustness against natural language adversarial attacks, a kind of defense is to recognize and block the adversarial examples [16, 17, 18]. However, these methods rely heavily on how accurate the defense can identify the adversarial examples. Besides, identifying the adversarial examples during inference time also brings additional computational overheads. Other researchers have resorted to learning robust vector representations of texts. [19] obtain robust word embeddings by optimizing within a convex hull in the vector space.

Another method to improve model robustness is adversarial training, which has been success in the image domain [4]. However, previous works on adversarial training for natural language mostly focus on perturbations on the vector space, while actual adversarial attacks create adversarial examples by changing natural language symbols. For example, [20] and [21] improve model generalization ability by adversarial training on the word embedding space, without mentioning model robustness. However, they either ignore model robustness, or only test the model robustness against the adversarial dataset ANLI, without paying attention to actual adversarial attacks. Other works conduct adversarial training in the word space [2, 5]. Still, they can only do adversarial training on a limited number of pre-generated adversarial examples due to the low efficiency of the attacks. A recent work [6] conducts adversarial training efficiently in the word space, but their method is limited to traditional non-contextualized models.

The work in this thesis is different from previous works of natural language adversarial training. On the one hand, as opposed to previous works, which are supervised, we propose a self-supervised learning scheme to improve the robustness of pretrained language models. On the other hand, while previous works mostly focus on adversarial training in the embedding space, we conduct efficient adversarial training with pretrained language models on the word level.

2.2 Contrastive Learning

The main goal of contrastive learning is to make the representation space of similar samples closer to each other, while enlarge the distance between dissimilar ones. Contrastive learning has been successful in the computer vision domain. [8] propose SimCLR as a framework for contrastive learning, which outperforms previous works of self-supervised learning on ImageNet. However, SimCLR requires a large batch size to achieve high performance, which requires high demands on computing resources. To address this problem, [9] propose another framework MoCo. MoCo does not rely on a large batch size, but can achieve similar or even higher performance than SimCLR. [22] further add supervised signals to contrastive learning, reaching new state-of-the-art with various architectures on image classification tasks. Recently, [13] demonstrate that by using adversarial signals during contrastive learning, the models can obtain robustness against image adversarial attacks.

In NLP, previous works on contrastive learning mainly focus on improving the model generalization ability. [11] improve the performance of BERT on various downstream understanding tasks by using back-translation [15] and easy-data-augmentation [23] with contrastive learning. [12] further boost the performance of RoBERTa by adding supervised signals during fine tuning on downstream tasks. Similarly, [24] achieve comparable performance to supervised methods by using self-supervised contrastive learning. Recently, [14] tackle the "exposure bias" problem in text generation by adding adversarial signals during contrastive learning.

Despite that these works have demonstrated the usefulness of contrastive learning in NLP applications, none of them address the robustness of NLP models, particularly pretrained language models, against natural language adversarial attacks. In this thesis, we focus on improving the robustness of pretrained language models against word substitution-based adversarial attacks. We present the details of our method in Section 3.

Chapter 3

Methodology

3.1 Background and Objective

Imagine we have a batch of N examples: $\{(X_1, y_1), (X_2, y_2), \ldots, (X_n, y_N)\}$, where $X = \{w_1, w_2, \ldots, w_L\}$ is an example consisting of L words, y is the corresponding label, and i is the index to the examples. Let X_i be the current input to encoder $f(\cdot)$, $c(\cdot)$ be the softmax classification layer, and $a(\cdot)$ be a natural language adversarial attack, we have:

$$X'_i = a(X_i) \tag{3.1}$$
$$\boldsymbol{h}_i = f(X_i) \tag{3.2}$$
$$\boldsymbol{h}'_i = f(X'_i) \tag{3.3}$$
$$\hat{y}_i = c(\boldsymbol{h}_i) \tag{3.4}$$
$$\hat{y}'_i = c(\boldsymbol{h}'_i) \tag{3.5}$$

where $\boldsymbol{h}_i, \boldsymbol{h}'_i \in \mathbb{R}^d$ are fixed-size representations of X_i and X'_i, and $\hat{y}_i$ and $\hat{y}'_i$ are predicted labels of X_i and X'_i, respectively.

Assuming the attack successfully fools the model, we have $\hat{y}_i \neq \hat{y}'_i$. Our assumption is that although X_i and X'_i are very similar to each other on the word level, the distance of their representations $\boldsymbol{h}_i$ and $\boldsymbol{h}'_i$ are large such that the softmax classifier $c(\cdot)$ predicts X_i and X'_i to be of different classes.

Hence, to obtain a robust model against adversarial attacks, we need to optimize the encoder such that $\boldsymbol{h}_i$ and $\boldsymbol{h}'_i$ become similar to each other. We achieve this goal by conducting self-supervised contrastive learning with adversarial perturbations, during which we use an attack that aims to create adversarial examples maximizing the contrastive loss. We present the details in Section 3.2

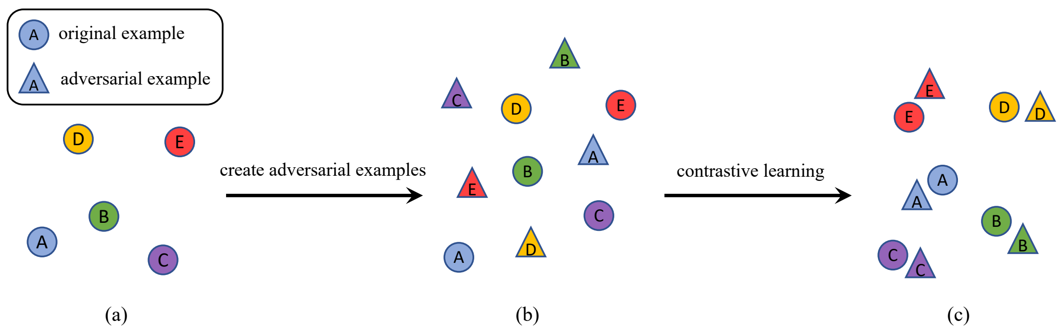

Figure 3.1: An illustration of our method. Examples of the color or character are considered a pair of positive examples. (a) Before contrastive learning: the original examples can be represented in the vector space. (b) By using the contrastive loss as the objective, we create adversarial examples, which are similar to the corresponding clean examples on the word level. However, the representations of the adversarial examples may be distant from their corresponding clean examples. (c) After contrastive learning: the representations of clean examples and their corresponding adversarial examples are closer, while the distances between dissimilar examples become larger.

3.2 Self-Supervised Contrastive Learning with Adversarial Perturbations

Following previous works on self-supervised contrastive learning [8, 11], we formulate our learning objectives as follows. Consider X_i as the current input, we first obtain an augmentation of X_i by transformation $t(\cdot)$:

$$X_{i+n} = t(X_i) \tag{3.6}$$

We call X_i and X_{i+n} a pair of positive examples. All other examples in the same batch are considered negative examples of X_i and X_{i+n}. We then have:

$$\boldsymbol{h}_i = f(X_i) \tag{3.7}$$
$$\boldsymbol{h}_{i+n} = f(X_{i+n}) \tag{3.8}$$
$$\boldsymbol{z}_i = g(\boldsymbol{h}_i) \tag{3.9}$$
$$\boldsymbol{z}_{i+n} = g(\boldsymbol{h}_{i+n}) \tag{3.10}$$

where $\boldsymbol{h}_i, \boldsymbol{h}_{i+n} \in \mathbb{R}^d$ are representations of X_i and X_{i+n}, respectively. And $g(\cdot)$ is another MLP mapping $\boldsymbol{h}$ to $\boldsymbol{z} \in \mathbb{R}^c$.

In our experiments, we use the attack described in Section 3.3 or back-translation [15] for augmentation $t(\cdot)$. The contrastive learning objective for X_i and X_{i+n} is:

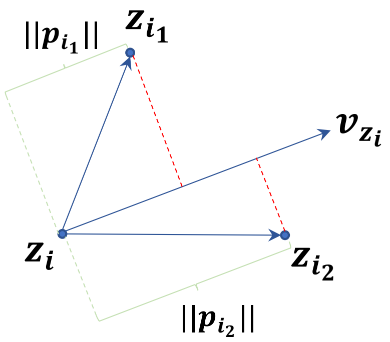

Figure 3.2: An illustration of one iteration in Geometry Attack for contrastive loss. Refer to Section 3.3 for details.

$$\ell_i = -\log \frac{\exp(\mathrm{sim}(\boldsymbol{z}_i, \boldsymbol{z}_{i+n})/\tau)}{\sum_{k=1}^{2n} 1_{k\neq 0} \exp(\mathrm{sim}(\boldsymbol{z}_i, \boldsymbol{z}_{i+k})/\tau)} \tag{3.11}$$

where τ is the temperature parameter, and $\mathrm{sim}(\cdot, \cdot)$ is the similarity function. In this paper, we use cosine similarity. Following [8], we conduct contrastive learning on $\boldsymbol{z}$ instead of $\boldsymbol{h}$ to prevent the contrastive learning objective from removing information useful for downstream tasks.

[8] shows that a large batch size helps achieve a high performance. That is because as shown in 3.11, large batch size implies more dissimilar samples. Our experiments in section 5.7 indicates the same conclusion. However, due to the large amount of parameters of pre-trained language models such as BERT, using large batch size is very demanding on computational resources. To solve this problem, we try to follow another framework MoCo [9].

Consider $X_1, ..., X_m$ as the current batch and $X'_1, ..., X'_m$ as the augmentation obtained by 3.6. MoCo uses encoder $g(f(\cdot))$ and momemtum encoder $g'(f'(\cdot))$. For each sample X_i, we then have:

$$\boldsymbol{z}_i = g(f(X_i)) \tag{3.12}$$

$$\boldsymbol{z}'_i = g'(f'(X_i)) \tag{3.13}$$

To enlarge the amount of dissimilar samples, MoCo uses a queue whose size is independent of batch size. In each iterator, $\boldsymbol{z}'_1, ..., \boldsymbol{z}'_m$ are added to the queue, and at the same time, the oldest m elements are removed from the queue. The structure of the MoCo framework is shown in Figure 3.3. MoCo successfully reduces the memory requirement by maintaining this queue, where each element $\boldsymbol{q}_i$ is gradient-free. At the end of each iteration, MoCo updates each parameter $\boldsymbol{\theta}'_j$ in $g'(f'(\cdot))$ with momentum:

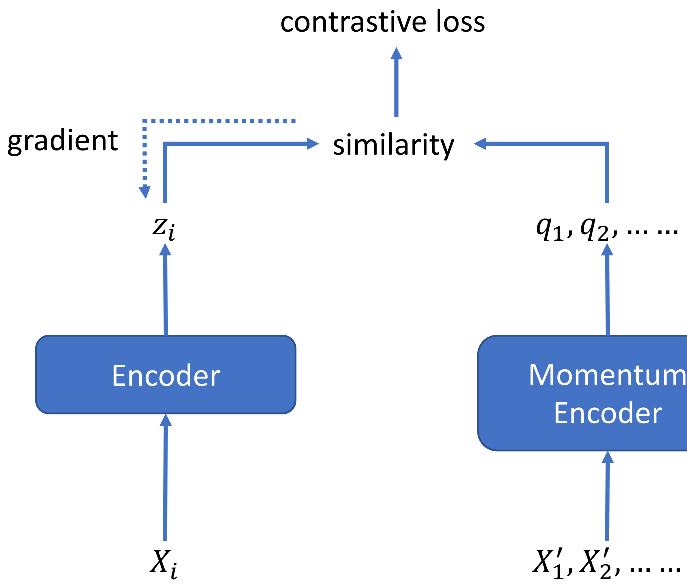

Figure 3.3: The structure of the MoCo framework [9]. MoCo maintains a queue $\boldsymbol{q}_1, ..., \boldsymbol{q}_n$, where each element is gradient-free, to get rid of the limitation of batch size.

$$\boldsymbol{\theta}'_j = k\boldsymbol{\theta}_j + (1-k)\boldsymbol{\theta}'_j \tag{3.14}$$

where $\boldsymbol{\theta}_j$ is the corresponding parameter from $g(f(\cdot))$ and k is the momentum coefficient. The contrastive learning objective for each sample X_i is:

$$\ell_i = -\log \frac{\exp(\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}'_i)/\tau)}{\sum_{j=1}^{n} \exp(\text{sim}(\boldsymbol{z}_i, \boldsymbol{q}_j)/\tau)} \tag{3.15}$$

where n is the size of the queue, and τ is the temperature parameter.

By optimizing Equation 3.11 and 3.15, the goal is to maximize the similarity of representations between similar pairs of examples, while minimizing the similarity of representations between dissimilar examples. We use the geometry-inspired attack described in Section 3.3 to create pairs of examples that are similar on the word level, but at the same time are distant from each other in the representation space.

Figure 3.1 is an illustration of how we conduct self-supervised contrastive learning with adversarial signals. In Figure 3.1 (a), we have the representations of the original examples (circles) in the vector space. Each individual example is labeled by a different color. Figure 3.1 (b) illustrates that the adversarial examples (triangles) are distant from

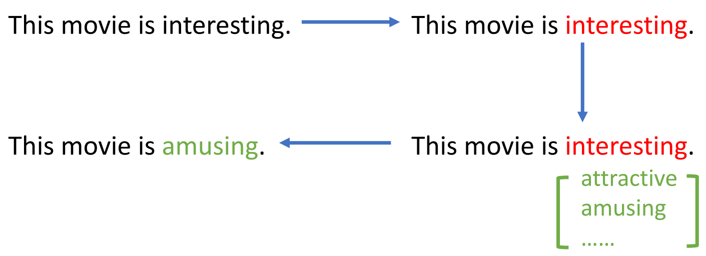
This movie is interesting.
This movie is interesting.
This movie is interesting.
attractive
amusing
......
This movie is amusing.

Figure 3.4: An simple example of attack.

the corresponding original examples in the vector space. In Figure 3.1 (c), clean examples and their corresponding adversarial examples are closer to each other, while at the same time distancing from other examples.

3.3 Geometry Attack for Contrastive Loss

We describe in this section how we create adversarial examples for contrastive loss during self-supervised contrastive learning (see Figure 3.1 (b)). Inspired by [25], who leverage geometry information to generate natural language adversarial examples for text classification tasks, we also use geometry information to create adversarial examples for contrastive loss. Specifically, our attack creates adversarial examples which maximize the contrastive loss in Equation 3.11 and 3.15.

The intuition of our attack is that we repeatedly replace words in the original texts such that in each iteration, the replaced word increases the contrastive loss as much as possible. To be specific, consider an example X_i, we then have:

1. Solve the gradients of ℓ_i with respect to input word embeddings of X_i. For words tokenized into multiple tokens, we take the average of the gradients of the tokens. In this step we want to understand which word has the most influence in computing ℓ_i.

2. Suppose we choose word w_t in step 1. In this step, we use a pre-trained BERT to choose for w_t the most probable candidates to replace it in the original texts. We have the candidates set $= \{w_{t_1}, w_{t_2}, \cdots, w_{t_T}\}$. Since the candidates selected by BERT tend to be grammatically correct, rather than semantically similar, we follow the work [26] and filter out semantically different words from the candidates set by discarding candidate words of which the cosine similarity of their embeddings between the embeddings of w_t is below a threshold ϵ.

3. Solve the gradients of ℓ_i with respect to $\boldsymbol{z}_i$. In this step, we know to which direction we should move $\boldsymbol{z}_i$ to increase the contrastive loss ℓ_i. We have the gradient vector

$\boldsymbol{v}_{z_i} = \nabla_{\boldsymbol{z}_i} \ell_i$.

4. Replace w_t with words in the candidates set, then we have following text vectors: $\{\boldsymbol{z}_{i_1}, \boldsymbol{z}_{i_2}, \cdots, \boldsymbol{z}_{i_T}\}$. We then have delta vector $\boldsymbol{r}_{i_j} = \boldsymbol{z}_{i_j} - \boldsymbol{z}_i$. The projection of $\boldsymbol{r}_{i_j}$ onto $\boldsymbol{v}_{z_i}$ is: $\boldsymbol{p}_{i_j} = \frac{\boldsymbol{r}_{i_j} \cdot \boldsymbol{v}_{z_i}}{||\boldsymbol{v}_{z_i}||}$. We select the candidate word w_{t_m}, where $m = \text{argmax}_j \, ||\boldsymbol{p}_{i_j}||$. In other words, w_{t_m} results in the largest projection $\boldsymbol{p}_{i_m}$ onto $\boldsymbol{v}_{z_i}$.

5. Replace w_t with w_{t_m} in X_i, then we have $\boldsymbol{z}_i \leftarrow \boldsymbol{z}_{i_m}$. Repeat step 1-4 for N iterations, where N is a hyperparameter of our method. We expect ℓ_i to increase in each iteration.

Algorithm 1 Geometry Attack for Contrastive Loss

```
1: Input: Example $X_i = \{w_1, w_2, \dots, w_L\}$, encoder $f$ and MLP $g$
2: Output: Adversarial example $X'_i$
3: Initialize $X'_i \leftarrow X_i$, $\boldsymbol{z}_i \leftarrow g(f(X_i))$
4: for iter = 1 to $N$ do
5:     calculate $\ell_i$ using Equation 3.11
6:     $\boldsymbol{v}_{z_i} \leftarrow \nabla_{\boldsymbol{z}_i} \ell_i$
7:     $\boldsymbol{E} \leftarrow \texttt{BertEmbeddings}(X'_i) = \{\boldsymbol{e}'_1, \boldsymbol{e}'_2, \dots, \boldsymbol{e}'_L\}$
8:     $\boldsymbol{G} \leftarrow \nabla_{\boldsymbol{E}} \ell_i = \{\boldsymbol{g}'_1, \boldsymbol{g}'_2, \dots, \boldsymbol{g}'_L\}$
9:     $t \leftarrow \arg\max_t ||\boldsymbol{g}'_t||$
10:    $C \leftarrow \texttt{BertForMaskedLM}(\{w_1, \cdots, w_{t-1}, \texttt{[MASK]}, w_{t+1}, \cdots, w_L\})$
11:    $C \leftarrow \texttt{Filter}(C)$ // construct candidates set $C = \{w_{t_1}, w_{t_2}, \cdots, w_{t_T}\}$
12:    for each $w_{t_j} \in C, 1 \le j \le T$ do
13:        $X'_{i_j} \leftarrow \{w_1, \cdots, w_{t-1}, w_{t_j}, w_{t+1}, \cdots, w_L\}$
14:        $\boldsymbol{z}_{i_j} \leftarrow g(f(X'_{i_j}))$
15:        $\boldsymbol{r}_{i_j} \leftarrow \boldsymbol{z}_{i_j} - \boldsymbol{z}_i$
16:        $\boldsymbol{p}_{i_j} \leftarrow \frac{\boldsymbol{r}_{i_j} \cdot \boldsymbol{v}_{z_i}}{||\boldsymbol{v}_{z_i}||}$
17:    end for
18:    $m \leftarrow \arg\max_j ||\boldsymbol{p}_{i_j}||$
19:    $X'_i \leftarrow X'_{i_m}$
20:    $\boldsymbol{z}_i \leftarrow \boldsymbol{z}_{i_m}$
21: end for
```

Figure 3.4 is a simple example of our attack. In step 1, we select the word *interesting* which has the greatest impact on contrastive loss. In step 2, we create candidates set [attractive, amusing, ...] for the word *interesting*. In step 3, 4 and 5, we choose the candidate *amusing* to replace the word *interesting*. Figure 3.2 illustrates how we use geometry information to choose a word from the candidates set, in which we have two candidates and we prefer to choose the one with the larger projection. This attack can be easily implemented in a batched fashion, making it possible for us to generate adversarial examples *on the fly* during training. Algorithm 1 is the pseudocode of our Geometry Attack for contrastive loss.

3.4 Geometry Attack for Adversarial Training

To create adversarial examples on the fly during adversarial training, we apply the similar idea described in Section 3.3, that is using geometry information. In adversarial learning, our geometric attack can be more effective because at this point we have the information of labels and are no longer under the condition of self-supervised settings. Algorithm 2 is the pseudocode of our Geometry Attack for adversarial attack.

Algorithm 2 Geometry Attack for Adversarial Training

```
1: Input: Example $X_i = \{w_1, w_2, \ldots, w_L\}$, encoder $f$ and classifier $g$
2: Output: Adversarial example $X'_i$
3: Initialize $X'_i \leftarrow X_i$, $\boldsymbol{z}_i \leftarrow f(X_i)$, $\boldsymbol{c}_i \leftarrow g(\boldsymbol{z}_i)$
4: calculate loss $\ell_i$
5: $\boldsymbol{E} \leftarrow \texttt{BertEmbeddings}(X'_i) = \{\boldsymbol{e}'_1, \boldsymbol{e}'_2, \ldots, \boldsymbol{e}'_L\}$
6: $\boldsymbol{G} \leftarrow \nabla_{\boldsymbol{E}} \ell_i = \{\boldsymbol{g}'_1, \boldsymbol{g}'_2, \ldots, \boldsymbol{g}'_L\}$
7: $\boldsymbol{T} \leftarrow$ indices of top-k $||\boldsymbol{g}'_t||$
8: for each $t \in \boldsymbol{T}$ do
9:     $C \leftarrow \texttt{BertForMaskedLM}(\{w_1, \cdots, w_{t-1}, \texttt{[MASK]}, w_{t+1}, \cdots, w_L\})$
10:    $C \leftarrow \texttt{Filter}(C)$ // construct candidates set $C = \{w_{t_1}, w_{t_2}, \cdots, w_{t_T}\}$
11:    $\boldsymbol{b}_i \leftarrow \texttt{DeepFool}(\boldsymbol{z}_i, g)$
12:    $\boldsymbol{v}_{z_i} \leftarrow \boldsymbol{b}_i - \boldsymbol{z}_i$
13:    for each $w_{t_j} \in C, 1 \leq j \leq T$ do
14:        $X'_{i_j} \leftarrow \{w_1, \cdots, w_{t-1}, w_{t_j}, w_{t+1}, \cdots, w_L\}$
15:        $\boldsymbol{z}_{i_j} \leftarrow f(X'_{i_j})$
16:        $\boldsymbol{r}_{i_j} \leftarrow \boldsymbol{z}_{i_j} - \boldsymbol{z}_i$
17:        $\boldsymbol{p}_{i_j} \leftarrow \frac{\boldsymbol{r}_{i_j} \cdot \boldsymbol{v}_{z_i}}{||\boldsymbol{v}_{z_i}||}$
18:    end for
19:    $m \leftarrow \arg\max_j ||\boldsymbol{p}_{i_j}||$
20:    $X'_i \leftarrow X'_{i_m}$
21:    $\boldsymbol{z}_i \leftarrow \boldsymbol{z}_{i_m}$
22:    if $g(f(X'_i)) \neq g(f(X_i))$ then
23:        break
24:    end if
25: end for
```

Compared to Algorithm 1, we make following changes to transfer the method described in Section 3.3 to adversarial training settings.

MLP $g(\cdot)$: With label information, MLP $g(\cdot)$ in 3.9 becomes a classifier whose output dimension depends on dataset statistics.

Terminate Condition: Due to the lack of label information, in self-supervised contrastive learning, we use the parameter N for a fixed number of attack iterations. In adversarial training, this parameter is still retained, but with the label information, we can check whether the attack is successful at the end of each iteration, and end the loop once the predicted label changes.

Word Selection: In self-supervised contrastive learning, we select a most important word in each iteration, which consists of a forward calculation and a backward gradient calculation. To further speed up, this step only be processed once before the attack loop in adversarial training, and select the top-k most important words at once.

Direction v_{z_i}: Follows [25], we leverage the `DeepFool` algorithm [27] to get the nearest point $\boldsymbol{b}_i$ on the decision boundary and then compute direction v_{z_i} which originates from the sentence vector $\boldsymbol{z}_i$ to $\boldsymbol{b}_i$. This method is more efficient than the corresponding step in self-supervised contrast learning because it utilizes label information.

Chapter 4

Experimental Details

4.1 Datasets

We apply our method on four classic text classification datasets: AG News, Yelp, IMDB, and DBpedia. The statistics of each dataset is shown in Table 4.1. In our work, the maximum sequence length is set to 128 for AG News and DBpedia, 256 for Yelp and 512 for IMDB. To save time during evaluating the model robustness against attacks, we randomly select a part of the test examples in each dataset for evaluation. To be specific, we select 1,000 samples from IMDB, 2,000 samples from Yelp, and 5,000 samples from DBpedia. We use all 7,600 samples from the AG News test set for evaluation. When training models, we randomly split the train data to train dataset and validation dataset with the ratio of 0.8.

Dataset	**Labels**	**Avg Len**	**Train**	**Test**
AG's News	4	44	120K	7.6K
IMDB	2	292	25K	25K
DBPedia	14	67	560K	70K
Yelp	2	177	560K	38K

Table 4.1: Statistics of the datasets.

AG's News[1] [28] Topic classification dataset with four types of news articles: World, Sports, Business and Science/Technology.

IMDB[2] [29] Binary sentiment classification dataset on positive and negative movie reviews.

Yelp[3] [28] Yelp review dataset for binary sentiment classification. Following [28], reviews

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

with star 1 and 2 are considered negative, and reviews with star 3 and 4 are considered positive.

DBpedia[4] [28] Topic classification dataset with 14 non-overlapping classes. Both content and title fields are used in our work.

4.2 Evaluation Settings

To understand how our self-supervised learning scheme improves the robustness of pre-trained language models, we conduct the following experiments on each downstream tasks in our evaluation:

FT We fine tune a pretrained BERT model on the corresponding downstream dataset.

BTCL+FT We use back-translation as the transformation $t(\cdot)$ in Equation 3.6 for self-supervised contrastive learning. We fine tune the model on the same dataset after contrastive learning.

ADCL(ADMoCo)+FT We first conduct contrastive learning using our Geometry Attack for contrastive loss (see Section 3.3) as transformation $t(\cdot)$. **ADCL** uses Equation 3.11 as contrastive loss, while **ADMoCo** uses Equation 3.15. The model is then fine-tuned on the same dataset.

ADV We directly conduct adversarial training. Note that our adversarial training is different from previous works [5, 2], which merely fine tune the model on a fixed number of pre-generated adversarial examples. Our adversarial training scheme is similar to [4], where adversarial examples are generated on the fly for each batch during training.

ADCL(ADMoCo)+ADV We first conduct adversarial contrastive learning. We then do adversarial training on the same dataset.

4.2.1 Adversarial Attacks

We use four word substitution-based natural language adversarial attacks, TextFooler, PWWS, BAE-R and Geometry Attack, to evaluate the robustness of the models. The first three attacks are selected from recent work. We consider the attack speed and the attack success rate when selecting attack algorithms.

TextFooler [26] TextFooler ranks the importance of measuring the decrease of true class probability after deleting words from the original texts. It builds a candidate set by leveraging the similarity of word embeddings. Finally, TextFooler selects the word which gives the least confidence in the true class label.

[4] `https://huggingface.co/datasets/dbpedia_14`

PWWS [5] By computing word saliency scores, PWWS first creates adversarial examples by first selecting words from the original texts for replacement. It then selects a word from the synonym candidate set by considering the change of true class probability.

BAE-R [30] BAE-R obtain token importance scores by using the same technique as TextFooler. The candidate set for replacement is obtained by leveraging a masked language model. The attack selects the candidate word resulting in the maximum decrease of true class probability. We note that BAE-R is weak, as observed before in [31].

Geometry Attack As described in Section 3.4. Instead of using word saliency scores as in the original work [25], we solve gradients of loss for computing the word importance. We apply the same strategy for creating the candidate set $\{w_{t_1}, w_{t_2}, \cdots, w_{t_T}\}$ as in [26] instead of using a synonym candidate set. Refer to Section 3.4 for more details.

4.3 Implementation Details

In this thesis, we use PyTorch Lightning[5] and HuggingFace Transformers[6] in our implementation. We use implementations from TextAttack[7] for TextFooler, PWWS and BAE-R. We use BERT as the encoder $f(\cdot)$, and the representation of the `[CLS]` symbol in the last layer is used for $\boldsymbol{h}$. $g(\cdot)$ is a two-layer MLP in self-supervised contrastive learning, of which the output size c is 128. $g(\cdot)$ uses `Tanh` as activation function in the output layer. We use FP16 in training step to reduce GPU memory usage, and use FusedAdam from DeepSpeed[8] as the optimizer. We enable DeepSpeed ZeRO Stage 2 to further speed up training.

4.3.1 Contrastive Learning

For Geometry Attack for constrative loss, to reach a balance between attack success rate and efficiency, the maximum number of iterations N is set to 10 for AG News, DBpedia and Yelp, and 15 for IMDB dataset. We do not perturb words which were already perturbed in previous iterations. For an example $X_i = \{w_1, w_2, \ldots, w_L\}$, at most $min\{N, 0.2 \cdot L\}$ words can be perturbed. For each word $w_t, 1 \leq t \leq L$, the upper limit of the candidate set size T is set to 25. For the parameter τ in contrastive loss function, we set it to 0.5 in Equation 3.11 and 0.7 in Equation 3.15. Due to the various maximum lengths in downstream datasets and GPU memory limits, we use different batch sizes for different datasets. Besides, due to the various size of train datasets and time limits, we use different epochs for different datasets. More details are shown in table 4.2.

[5]https://www.pytorchlightning.ai/
[6]https://huggingface.co/transformers/
[7]https://textattack.readthedocs.io/en/latest/index.html
[8]https://www.deepspeed.ai/

All experiments on ADCL are conducted on 8 RTX TITAN, while all experiments on ADMoCo are conducted on 8 RTX 2080Ti. Section 5.1 shows ADMoCo helps get higher performance even though the batch size is much more smaller. In ADMoCo, the momentum coefficient is set to 0.999 for all datasets, and the queue size is set to 32768 for AG News, DBpedia, 16384 for Yelp, and 4096 for IMDB.

Dataset	**Epochs**	**Batch Size**		**GPUs**	
		ADCL	ADMoCo	ADCL	ADMoCo
AG's News	15	1024	256	8 RTX TITAN	8 RTX 2080Ti
IMDB	15	192	32	8 RTX TITAN	8 RTX 2080Ti
DBPedia	5	1024	256	8 RTX TITAN	8 RTX 2080Ti
Yelp	5	448	128	8 RTX TITAN	8 RTX 2080Ti

Table 4.2: Epochs, batch size and hardware settings for ADCL and ADMoCo. Refer to section 4.2 for more details about ADCL and ADMoCo.

4.3.2 Fine Tuning

During fine-tuning, we train the model for 2 epochs with 32 batch size for AG News and DBpedia, 3 epochs with 32 batch size for Yelp, and 4 epochs with 16 batch size for IMDB. The learning rate is set to $2e-5$ and is adjusted using linear scheduling. All experiments are conducted on 2 RTX 2080Ti.

4.3.3 Adversarial training

For adversarial training, the number of training epochs is set to 3 with first half epoch of fine tuning. We set the batch size to 32 for AG News, DBpedia and Yelp, 16 for IMDB. The adversarial samples are generated *on the fly* in each batch during training. To be detail, (1) our algorithm only generate adversarial samples for samples whose labels are correctly classified, (2) all adversarial samples are used no matter they successfully attack the model or not. For the Geometry Attack in adversarial training, at most $\texttt{min}\{N, 0.4 \cdot \texttt{len}(X_i)\}$ words can be perturbed in an example where N is set to 50 for all datasets. The upper limit of the candidate set size is set to 25. All experiments are conducted on 2 RTX 2080Ti.

Chapter 5

Results

5.1 In-Domain Analysis

For the in-domain setting, we conduct adversarial/back-translation contrastive learning and adversarial training/fine tuning on the same dataset. We then evaluate the robustness of the models against the four attacks mentioned in Section 4.2.1. To measure the robustness of each model, we report attack success rates and percent of words replaced in the adversarial examples. To prevent the model accuracy on clean examples from confounding the results, we define the success rate of an attack on all *correctly classified* examples in the test set.

Table 5.1 shows the results our in-domain experiments. For each dataset, we use the same perturbation budget across different settings. Note that although the replacement rates vary across different settings of the same dataset, *the perturbation budget for the same attack is the same* in these settings. By using the same perturbation budget, we ensure that the success rates of the attacks provide us with a fair evaluation of the robustness of the model [6, 5, 25]. We nevertheless show the replacement rates in Table 5.1 for further reference.

A first observation from Table 5.1 is that merely fine tuning on clean examples results in the most vulnerable model. For example, the success rate of the Geometry Attack for AG News dataset is 86.2% in the FT setting. In contrast, for other settings, the success rate of the Geometry Attack is at least 5.5% lower than for FT.

We can also see from Table 5.1 that contrastive learning improves the model robustness. On the one hand, models of BTCL+FT and ADCL+FT are more robust than FT alone. For instance, in the IMDB dataset, the success rate of the Geometry Attack on the FT model is 98.7%, which is 6.4% higher than the success rate of 92.3% of the BTCL+FT model. In most of the settings, ADCL+FT models are also more robust than BTCL+FT models. The only exception is when we test DBpedia dataset against BAE-R attack, where both BTCL+FT and ADCL+FT result in a success rate of 12.6%. We argue that this might result from using only 5,000 examples from DBpedia as the test set, while 70K examples are available.

Dataset	Method	Original Acc. (%)	Success Rate / Replaced (%)			
			Geometry	TextFooler	PWWS	BAE-R
AG News	FT	94.2	86.2/18.6	87.6/25.7	63.6/20.9	17.9/7.4
	BTCL+FT	94.3	80.7/18.6	85.7/25.4	63.2/21.4	17.4/7.3
	ADCL+FT	94.1	78.8/18.9	83.8/25.4	61.8/21.7	15.7/7.5
Yelp	FT	97.1	94.6/10.6	94.3/10.4	97.0/7.1	42.1/6.7
	BTCL+FT	97.1	92.1/10.9	91.3/10.4	94.4/7.9	39.7/6.7
	ADCL+FT	97.1	90.8/10.3	90.3/9.9	94.0/7.0	38.7/6.9
IMDB	FT	92.3	98.7/3.5	99.0/6.5	99.2/4.3	54.0/3.0
	BTCL+FT	92.3	92.3/4.9	96.8/7.8	95.1/5.0	51.4/3.2
	ADCL+FT	92.4	88.7/4.5	92.1/7.6	91.0/4.7	49.0/3.0
DBpedia	FT	99.2	79.6/17.8	79.3/23.2	46.7/16.2	14.3/13.3
	BTCL+FT	99.2	77.5/18.8	76.4/23.1	44.8/18.1	12.6/13.8
	ADCL+FT	99.1	75.9/17.1	75.7/22.0	43.5/17.6	12.6/11.2

Table 5.1: In-domain experimental results on fine-tuning settings. FT denotes fine-tuning. BTCL denotes contrastive learning with back-translation. ADCL denotes contrastive learning with our Geometry Attack for contrastive loss. We use original accuracy, attack success rate and replacement rate as the evaluation metric.

5.2 Geometry Attack on Adversarial Training

To further understand the quality and efficiency of our geometry attack, we first conduct experiments on adversarial training settings. Table 5.2 shows our results, where we observe that adversarial training significantly improves model robustness in all settings. Besides, with adversarial training, the original accuracy of the model is still close to the one with merely fine tuning as shown in Table 5.1. The only exception is Yelp dataset, whose original accuracy in adversarial training settings is around 96%, while around 97% in fine tuning settings. That is because Yelp dataset is too large and we only do 1 epoch adversarial training. The results shows that our input space, word-level adversarial training is effective, and that the Geometry Attack is suitable for adversarial training with BERT.

Additionally, combining our self supervised contrastive learning with adversarial perturbations and adversarial training further improves the robustness. For instance, for the IMDB dataset, the ADCL+ADV model is 1.8% more robust than the ADV model, when both models are tested against the Geometry Attack (Success rates of Geometry attack:

Dataset	Method	Original Acc. (%)	Success Rate / Replaced (%)			
			Geometry	TextFooler	PWWS	BAE-R
AG News	ADV	94.4	20.7/20.5	25.1/29.3	26.1/22.3	10.7/7.7
	ADCL+ADV	94.4	19.6/20.0	24.2/28.9	25.9/21.7	10.2/7.5
Yelp	ADV	96.2	38.8/12.8	52.4/17.3	62.7/11.3	22.2/8.8
	ADCL+ADV	96.1	36.4/13.1	51.6/17.1	61.9/11.4	21.0/8.9
IMDB	ADV	92.0	51.4/7.4	75.3/12.7	79.1/9.3	35.1/3.6
	ADCL+ADV	91.9	49.6/7.7	75.1/12.6	78.9/9.0	32.4/3.4
DBpedia	ADV	99.0	13.9/21.6	16.5/28.2	17.7/18.9	10.9/14.1
	ADCL+ADV	99.0	13.1/20.3	15.7/29.8	17.3/18.6	10.8/14.6

Table 5.2: In-domain experimental results on adversarial training settings. ADV denotes adversarial training. ADCL denotes contrastive learning with our Geometry Attack for contrastive loss. We use original accuracy, attack success rate and replacement rate as the evaluation metric.

ADCL+ADV: 49.6%, ADV: 51.4%). We have similar observations in other settings.

As mentioned in Section 3, different from most works, we generate adversarial samples on the fly during adversarial training and contrastive learning. We conduct experiments on AG News dataset with two different training strategy: (a) generate adversarial samples on the fly for each batch; (b) generate adversarial samples in advance and use same adversarial samples for each batch. As shown in Table 5.3, strategy (a) significantly improves the robustness of adversarial models without affecting original accuracy. For instance, the model training with strategy (a) is 34.6% more robust than the model training with strategy (b), when both models are tested against the Geometry Attack.

Our Geometry Attack allows strategy (a) possible mainly by improving attack speed. To show the efficiency of our Geometry attack, we randomly select 1000 samples from each dataset, and compare attack speed between four attacks. As shown in Table 5.4, the attack speed of our Geometry Attack is more than 4 times faster than TextFooler, and around 10 times faster than PWWS and BAE-R on all four datasets.

5.3 MoCo Framework

Though the results in Section 5.1 and Section 5.2 shows that contrastive learning helps improve the robustness of models, the high demand on computing resources makes contrastive learning seem too expensive. To further improve the performance and at the same time reducing the burden of computing resources, we set another series experiments on

Strategy	Original	Success Rate / Replaced (%)			
	Acc. (%)	Geometry	TextFooler	PWWS	BAE-R
(a) on-the-fly	94.4	20.7/20.5	25.1/29.3	26.1/22.3	10.7/7.7
(b) pre-generated	94.2	55.3/17.1	59.4/22.6	42.0/17.4	16.5/7.3

Table 5.3: Performance comparison between two adversarial training strategy: (a) generating adversarial samples on the fly; (b) generating adversarial samples in advance. We conduct experiments on AG News dataset.

Dataset	Max Length	Attack Speed (s/sample)			
		Geometry	TextFooler	PWWS	BAE-R
AG News	128	0.44	2.48	6.29	5.37
Yelp	256	1.16	4.86	10.27	16.03
IMDB	512	2.02	8.69	21.86	24.10
DBpedia	128	0.69	2.69	2.52	7.74

Table 5.4: Attack speed study. We conduct experiments on FT model for all datasets. For each dataset, we use 1000 randomly selected samples.

MoCo Framework (refer to Section 3 for more details). We evaluate ADMoCo+FT, BT-MoCo+FT and ADMoCo+ADV on all four datasets. We then compare these models with ADCL+FT, BTCL+FT, ADCL+ADV respectively. The results are shown in Table 5.2.

The first observation from Table 5.2 is that applying MoCo framework in contrastive learning improves the robustness of models. For example, the success rate of the Geometry Attack for IMDB dataset drops from 88.7% to 84.2% in AD+FT setting, and from 49.6% to 48.7% in AD+ADV setting. The consistent results apply to other datasets and other attacks. This observation is delightful conclusion in that with MoCo framework, we can use much smaller batch size to get similar or even better performance to solve the problem of high demands on computing resources.

We can see from Table 5.2 that using back-translation in contrastive learning with MoCo framework doesn't make improvement, and some results even become worse. For instance, the success rate of the Geometry Attack for IMDB dataset increases from 92.3% to 93.3% in BT+FT settings. The potential reason is that with back-translation, all adversarial samples are generated in advance. That means we always use same adversarial samples in each epoch, so that we cannot take full advantage of the queue maintained

Dataset	Method	Original	Success Rate / Replaced (%)			
		Acc. (%)	Geometry	TextFooler	PWWS	BAE-R
AG News	ADCL+FT	94.1	78.8/18.9	83.8/25.4	61.8/21.7	15.7/7.5
	ADMoCo+FT	94.3	**76.5**/19.1	**80.7**/26.7	**55.9**/22.6	**14.1**/7.5
	BTCL+FT	94.3	80.7/18.6	85.7/25.4	63.2/21.4	**17.4**/7.3
	BTMoCo+FT	94.4	**80.6**/18.1	**84.6**/24.6	**63.1**/20.9	17.7/7.5
	ADCL+ADV	94.4	19.6/20.0	24.2/28.9	25.9/21.7	10.2/7.5
	ADMoCo+ADV	94.4	**18.7**/20.6	**23.5**/29.3	**24.7**/22.2	**9.7**/7.2
Yelp	ADCL+FT	97.1	90.8/10.3	90.3/9.9	94.0/7.0	38.7/6.9
	ADMoCo+FT	97.0	**88.6**/10.4	**88.2**/10.5	**91.1**/7.4	**37.8**/6.9
	BTCL+FT	97.1	**92.1**/10.9	**91.3**/10.4	**94.4**/7.9	39.7/6.7
	BTMoCo+FT	97.2	92.3/11.0	91.6/10.1	94.8/7.7	**39.2**/6.9
	ADCL+ADV	96.1	36.4/13.1	51.6/17.1	61.9/11.4	**21.0**/8.9
	ADMoCo+ADV	96.1	**35.6**/13.4	**50.1**/17.1	**61.0**/11.2	**21.0**/8.3
IMDB	ADCL+FT	92.4	88.7/4.5	92.1/7.6	91.0/4.7	49.0/3.0
	ADMoCo+FT	92.4	**84.2**/3.7	**87.8**/8.7	**87.8**/5.1	**48.0**/2.3
	BTCL+FT	92.3	**92.3**/4.9	96.8/7.8	**95.1**/5.0	**51.4**/3.2
	BTMoCo+FT	92.5	93.3/4.5	**96.6**/7.4	**95.1**/4.4	52.0/3.3
	ADCL+ADV	91.9	49.6/7.7	75.1/12.6	78.9/9.0	32.4/3.4
	ADMoCo+ADV	91.9	**48.7**/8.1	**74.4**/12.4	**77.6**/9.1	**31.8**/3.5
DBpedia	ADCL+FT	99.1	75.9/17.1	75.7/22.0	43.5/17.6	12.6/11.2
	ADMoCo+FT	99.2	**73.6**/18.2	**74.5**/22.9	**42.6**/17.6	**11.6**/12.8
	BTCL+FT	99.2	77.5/18.8	**76.4**/23.1	**44.8**/18.1	**12.6**/13.8
	BTMoCo+FT	99.1	**77.4**/18.9	76.8/22.8	45.1/18.1	13.0/13.1
	ADCL+ADV	99.0	13.1/20.3	15.7/29.8	17.3/18.6	10.8/14.6
	ADMoCo+ADV	99.0	**12.4**/20.1	**14.8**/28.6	**16.2**/18.2	**10.1**/13.8

Table 5.5: In-domain experimental results comparison between two different contrastive learning framework mentioned in Section 3. We use original accuracy, attack success rate and replacement rate as the evaluation metric.

by MoCo framework.

Dataset	Method	Original Acc. (%)	Success Rate / Replaced (%)			
			Geometry	TextFooler	PWWS	BAE-R
AG News	FT	94.2	86.2/18.6	87.6/25.7	63.6/20.9	17.9/7.4
	BTCL+FT	94.3	80.7/18.6	85.7/25.4	63.2/21.4	17.4/7.3
	ADCL+FT	94.1	78.8/18.9	83.8/25.4	61.8/21.7	15.7/7.5
	ODCL+FT	94.2	81.0/19.3	85.5/25.8	62.0/21.6	17.6/7.8
	ODMoCo+FT	94.1	79.2/18.7	84.0/25.9	60.4/21.9	16.3/7.5
IMDB	FT	92.3	98.7/3.5	99.0/6.5	99.2/4.3	54.0/3.0
	BTCL+FT	92.3	92.3/4.9	96.8/7.8	95.1/5.0	51.4/3.2
	ADCL+FT	92.4	88.7/4.5	92.1/7.6	91.0/4.7	49.0/3.0
	ODCL+FT	92.1	93.0/3.7	96.2/7.5	95.1/4.4	52.3/3.1
	ODMoCo+FT	92.5	92.3/4.4	95.7/8.6	94.5/5.3	50.1/3.1

Table 5.6: Out-of-domain experimental results. We use the DBpedia dataset as the out-of-domain dataset for AG News and IMDB. ODCL denotes contrastive learning on the DBpedia dataset with our Geometry Attack for contrastive loss 3.11. ODMoCo denotes contrastive learning on the DBpedia dataset with our Geometry Attack for contrastive loss 3.15.

5.4 Out-of-Domain Study

An advantage of our self-supervised contrastive learning with adversarial perturbations is that we do not need labeled examples to improve the model robustness. This enables us to extend our method to out-of-domain raw data, for example, Wikipedia articles, which do not have labels.

In our experiments, we use the DBpedia dataset as the out-of-domain dataset for the AG News and IMDB datasets, mainly because (1) Computational limits: while using larger datasets such as BookCorpus or Wikipedia might be more useful, conducting self-supervised contrastive learning on these datasets exceeds the limits of our computational infrastructure; (2) The DBpedia dataset is several times larger than AG News and IMDB. This should give us a glimpse of what it looks like when we scale self-supervised contrastive learning with adversarial perturbations to even larger out-of-domain datasets; (3) The DBpedia dataset (topic classification on Wikipedia) has different task and domain compared to the AG News dataset (news classification from a newspaper) and IMDB dataset (sentiment classification on movie reviews). This discrepancy allows us to understand how *out-of-domain* datasets could help.

Table 5.6 shows our results, where ODCL+FT refers to conducting self-supervised

Dataset	**Attacker**	**Success Rate (%)**			
		FT → ADCL+FT	ADCL+FT → FT	FT → ADMoCo+FT	ADMoCo+FT → FT
AG News	Geometry	32.4	**44.3**	30.2	**62.3**
	TextFooler	25.5	**33.4**	19.7	**55.0**
	BAE-R	29.0	**39.6**	28.3	**50.4**
	PWWS	27.9	**39.4**	26.4	**60.2**
Yelp	Geometry	30.9	**34.9**	30.1	**36.4**
	TextFooler	23.9	**26.6**	22.4	**28.0**
	BAE-R	28.6	**36.3**	34.8	**36.3**
	PWWS	29.6	**36.8**	37.4	**41.5**
IMDB	Geometry	35.4	**43.9**	38.2	**41.4**
	TextFooler	22.1	**24.3**	22.1	**25.2**
	BAE-R	19.6	**26.3**	24.7	**26.0**
	PWWS	28.0	**29.6**	28.9	**30.8**
DBpedia	Geometry	40.5	**45.0**	34.6	**52.2**
	TextFooler	32.6	**35.4**	27.5	**42.8**
	BAE-R	47.1	**50.6**	55.3	**58.8**
	PWWS	37.8	**39.4**	32.5	**55.8**

Table 5.7: Transferability of adversarial examples. FT → ADCL+FT: we generate adversarial examples using FT models, then test ADCL+FT models on these adversarial examples. The same applies to ADCL+FT → FT, FT → ADMoCo+FT and ADMoCo+FT → FT.

contrastive learning with adversarial perturbations. We notice that ODCL+FT models are not as robust as BTCL+FT and ADCL+FT models. One delightable thing is that when we use the MoCo framework for contrastive learning, which is shown by ODMoCo+FT in Table 5.6, the robustness of the models is better than that of BTCL+FT. To further exceed the performance of ADCL+FT models, we might need to use much larger unlabeled raw datasets to obtain more improvements.

Nevertheless, we can still see that ODCL+FT and ODMoCo+FT models are more robust than FT models. For instance, for the IMDB dataset, the success rate of TextFooler

Dataset	**Distance** ($d_{pos}/d_{neg}/\delta$)			
	FT	ADCL+FT	ADV	ADCL+ADV
AG News	2.4/3.8/1.4	2.0/4.2/2.2	0.6/3.8/3.2	0.6/4.6/4.2
Yelp	2.9/3.1/0.2	2.7/3.9/1.2	0.6/2.7/2.1	0.6/3.3/2.7
IMDB	2.9/3.6/0.7	2.2/3.8/1.6	0.5/2.5/2.0	0.5/3.3/2.9
DBpedia	2.8/4.8/2.0	2.6/5.1/2.5	0.4/4.8/4.4	0.4/5.1/4.7

Table 5.8: Vector space study under ADCL settings. For each setting, we evaluate three metrics: (a) the average distance between positive pairs; (b) the average distance between negative pairs; (c) the difference between (a) and (b).

decreases from 99.0% for FT models to 96.2% for ODCL+FT models and to 95.7% for ODMoCo+FT models. This shows that our method can improve the model robustness even if the dataset used for contrastive learning from a completely different domain.

5.5 Transferability of Adversarial Examples

To further understand how contrastive learning improves the model robustness, we study the transferability of the adversarial examples under the settings of FT and ADCL+FT (ADMoCo+FT). To be specific, we use FT (ADCL/ADMoCo+FT) models to generate adversarial examples on the test set of each individual dataset, and then test the ADCL/ADMoCo+FT (FT) models on these adversarial examples. Table 5.7 shows the results. We can see that adversarial examples generated by ADCL/ADMoCo+FT models have much higher success rates on FT models. For example, for the AG News dataset, the success rates increase by 11.9%, 7.9%, 10.6%, and 11.5% for Geometry Attack, TextFooler, BAE-R, and PWWS, respectively under ADCL+FT settings, and by 32.1%, 35.3%, 22.1%, and 33.8% under ADMoCo+FT settings. This demonstrates that by self-supervised contrastive learning with adversarial perturbations, the models become more robust against attacks.

5.6 Vector Space Study

By optimizing Equation 3.11, our method pulls the representations of the clean samples and their corresponding adversarial samples closer in the vector space, while pushing other different examples further. To validate this assumption, we study the representations of M clean examples of the AG News dataset and their corresponding adversarial

Dataset	**Distance** ($d_{pos}/d_{neg}/\delta$)			
	FT	ADMoCo+FT	ADV	ADMoCo+ADV
AG News	2.4/3.9/1.5	1.8/4.0/2.2	0.7/4.1/3.4	0.7/4.4/3.7
Yelp	3.5/3.7/0.2	2.9/4.0/1.1	0.7/3.2/2.5	0.5/3.4/2.9
IMDB	3.0/3.7/0.7	2.3/3.8/1.5	0.6/3.4/2.8	0.6/3.8/3.2
DBpedia	2.8/4.8/2.0	2.3/5.1/2.6	0.4/4.9/4.5	0.4/5.2/4.8

Table 5.9: Vector space study under ADMoCo settings. For each setting, we evaluate three metrics: (a) the average distance between positive pairs; (b) the average distance between negative pairs; (c) the difference between (a) and (b).

examples in the vector space. We obtain the adversarial examples using the FT models of each dataset.

Let $\boldsymbol{v}_1, \boldsymbol{v}_2 ... \boldsymbol{v}_M$ and $\boldsymbol{v}'_1, \boldsymbol{v}'_2 ... \boldsymbol{v}'_M$ be the vector representations of the clean examples and corresponding adversarial samples, respectively. For each setting, we evaluate three metrics:

(a) The average distance d_{pos} between each of the positive pairs v_i and v'_i, where $1 \leq i \leq M$. Then we have $d_{pos} = \frac{1}{M} \sum_{i=1}^{M} \texttt{distance}(\boldsymbol{v}_i, \boldsymbol{v}'_i)$.

(b) The average distance d_{neg} between negative pairs, which is $d_{neg} = \frac{1}{2(M-1)} \sum_{i=1}^{M} \sum_{j=1}^{M} \mathbb{1}_{i \neq j} (\texttt{distance}(\boldsymbol{v}_i, \boldsymbol{v}_j) + \texttt{distance}(\boldsymbol{v}_i, \boldsymbol{v}'_j))$.

(c) The difference $\delta = d_{neg} - d_{pos}$ between (a) and (b).

Table 5.8 and Table 5.9 shows the results. We can see that our method (1) increases the distance between negative pairs in all settings; (2) decreases the distance between positive pairs in FT and ADCL(ADMoCo)+FT models, while the distances between positive pairs barely change in ADV and ADCL(ADMoCo)+ADV models; (3) increases δ in all settings. The above three observations validate our assumption in Section 3, and further explain why our method increases the robustness of pretrained language models against attacks.

To visualize the distance shown in Table 5.8 and Table 5.9, we randomly select 5 samples from DBpedia dataset and use FT model to generate corresponding adversarial samples. We then get 10 vector representations using FT, ADMoCo+FT, ADV and ADMoCo+ADV models respectively and project them into 2D dimension by T-SNE. The results are shown in Figure 5.1, where circles with same colors represent pairs of samples and corresponding adversarial samples. We can see from Figure 5.1 that with

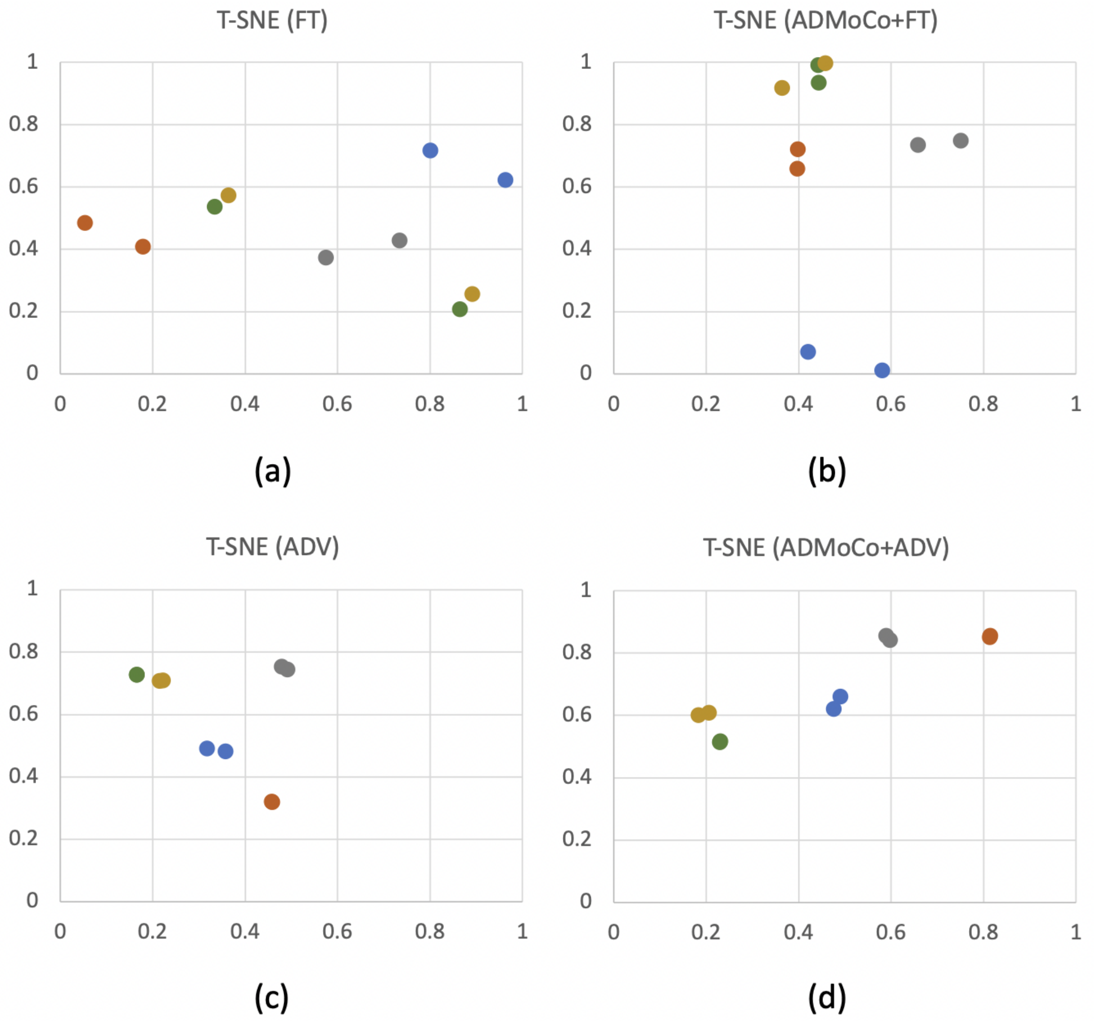

Figure 5.1: Latent space visualization. We randomly select 5 samples and their corresponding adversarial samples from DBpedia dataset. We get 10 vector representations using FT, ADMoCo+FT, ADV and ADMoCo+ADV models respectively and project them into 2D dimension by T-SNE. We use same color for each pair of samples and adversarial samples.

contrastive learning, the distance between a sample and its corresponding adversarial sample becomes closer, which is consistent to the conclusion obtained from Table 5.9.

5.7 Effect of Batch Size

We conduct additional experiments to study the effect of batch size in ADCL settings. We use a batch size of 64, 256, and 1024 under the setting of ADCL+FT for the AG

Batch Size	Original Acc. (%)	Success (%)	Replaced (%)
FT	94.2	86.2	18.6
64	94.3	84.7	19.1
256	94.3	80.7	18.9
1024	94.1	**78.8**	18.9

Table 5.10: Effect of batch size. We use the Geometry Attack to evaluate the robustness of each model. The FT model is finetuned without contrastive learning.

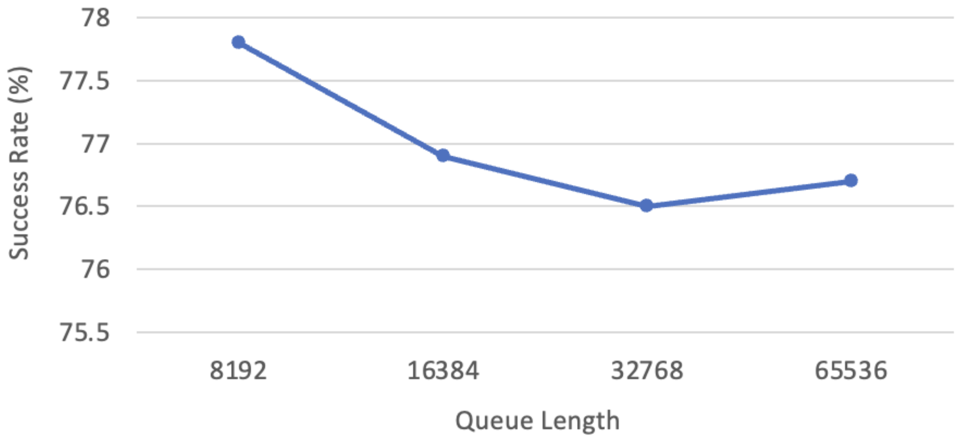

Figure 5.2: Effect of queue length. We use the Geomtery Attack to evaluate the robustness of each model.

News dataset. As is shown in Table 5.10, a larger batch size helps improve the model robustness: the success rate of Geometry Attack decreases from 84.7% to 78.8% as we increase the batch size from 64 to 1024. This is consistent with previous works of contrastive learning [8, 9].

5.8 Effect of Queue Length

Though using MoCo framework in contrastive learning saves the computing resources and at the same time gives higher performance as shown in Section 5.1, it is much harder to implement the algorithm and to tune parameters. One parameter that significantly affect the model performance is the size of queue length. We conduct a set of experiments to study the effect of the queue length. We use 8192, 16384, 32768 and 65536 under the setting of ADMoCo+FT for AG News dataset. The results are shown in Figure 5.2. One observation is that with queue length 32768, the model performs best. The other observation is that the performance becomes better when increasing the queue length, however the performance drops when the queue length is too large, which is different from the result shown in [9], where large queue length means better model performance. One possible reason is that in our work, the batch size is much smaller than the one used in [9]. Under the condition of large queue length and small batch size, the queue is updated slowly, that means there are too many old adversarial samples in the queue, which leads to the result of poor performance.

Chapter 6

Conclusion and Future Work

In this thesis, we propose to improve the robustness of pretrained language models against word substitution-based adversarial attacks by using self-supervised contrastive learning, during which we also leverage adversarial perturbations. Our method is different from previous works as we can improve model robustness without accessing annotated labels of the examples. Furthermore, we also conduct word-level adversarial training on BERT with an efficient attack. Our adversarial training is different from previous works in that (1) our adversarial training is on the word level; (2) we generate adversarial examples on the fly, instead of generating a fixed set of adversarial examples beforehand.

Experimental results on four datasets and four adversarial attacks show that our method improves the model robustness. We also find that combining our method with adversarial training results in better robustness than conducting adversarial training alone. In the future, we plan to scale our method to even larger out-of-domain datasets such as Wikipedia. We also plan to do more qualitative and quantitative analysis on the quality of the adversarial samples generated by our Geometry Attack.

6.1 Ethical Considerations

While our method can improve the robustness of the pretrained language models against adversarial attacks, we do not consider other shortcomings of such models in this thesis. For example, a pretrained language model such might still be subject to certain biases or prejudices of its training data even if it is now more robust to adversarial attacks. Furthermore, additional training with our method on large scale raw data might even strengthen such biases or prejudices. Therefore, one should still be careful when putting such models into practical use.

Bibliography

[1] R. Jia and P. Liang, "Adversarial examples for evaluating reading comprehension systems," in *EMNLP*, M. Palmer, R. Hwa, and S. Riedel, Eds., 2017.

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

[3] J. Ebrahimi, A. Rao, D. Lowd, and D. Dou, "HotFlip: White-box adversarial examples for text classification," in *ACL*, 2018.

[4] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *ICLR*, 2018.

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

[6] 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.

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *CVPR*, 2009.

[8] T. Chen, S. Kornblith, M. Norouzi, and G. E. Hinton, "A simple framework for contrastive learning of visual representations," in *ICML*, 2020.

[9] K. He, H. Fan, Y. Wu, S. Xie, and R. B. Girshick, "Momentum contrast for unsupervised visual representation learning," in *CVPR*, 2020.

[10] T. Park, A. A. Efros, R. Zhang, and J. Zhu, "Contrastive learning for unpaired image-to-image translation," in *ECCV*, A. Vedaldi, H. Bischof, T. Brox, and J. Frahm, Eds., 2020.

[11] H. Fang, S. Wang, M. Zhou, J. Ding, and P. Xie, "Cert: Contrastive self-supervised learning for language understanding," *arXiv preprint arXiv:2005.12766*, 2020.

[12] B. Gunel, J. Du, A. Conneau, and V. Stoyanov, "Supervised contrastive learning for pre-trained language model fine-tuning," in *ICLR*, 2021.

[13] M. Kim, J. Tack, and S. J. Hwang, "Adversarial self-supervised contrastive learning," in *NeurIPS*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., 2020.

[14] S. Lee, D. B. Lee, and S. J. Hwang, "Contrastive learning with adversarial perturbations for conditional text generation," in *ICLR*, 2021.

[15] 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 *ICCV*, 2015.

[16] D. Pruthi, B. Dhingra, and Z. C. Lipton, "Combating adversarial misspellings with robust word recognition," in *ACL*, A. Korhonen, D. R. Traum, and L. Màrquez, Eds., 2019.

[17] Y. Zhou, J.-Y. Jiang, K.-W. Chang, and W. Wang, "Learning to discriminate perturbations for blocking adversarial attacks in text classification," in *EMNLP-IJCNLP*, 2019.

[18] M. Mozes, P. Stenetorp, B. Kleinberg, and L. Griffin, "Frequency-guided word substitutions for detecting textual adversarial examples," in *EACL*, 2021.

[19] X. Dong, A. T. Luu, R. Ji, and H. Liu, "Towards robustness against natural language word substitutions," in *ICLR*, 2021.

[20] C. Zhu, Y. Cheng, Z. Gan, S. Sun, T. Goldstein, and J. Liu, "Freelb: Enhanced adversarial training for natural language understanding," in *ICLR*, 2020.

[21] X. Liu, H. Cheng, P. He, W. Chen, Y. Wang, H. Poon, and J. Gao, "Adversarial training for large neural language models," *arXiv preprint arXiv:2004.08994*, 2020.

[22] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan, "Supervised contrastive learning," *arXiv preprint arXiv:2004.11362*, 2020.

[23] J. Wei and K. Zou, "EDA: Easy data augmentation techniques for boosting performance on text classification tasks," in *EMNLP-IJCNLP*, 2019.

[24] Y. Yu, S. Zuo, H. Jiang, W. Ren, T. Zhao, and C. Zhang, "Fine-tuning pre-trained language model with weak supervision: A contrastive-regularized self-training approach," *arXiv preprint arXiv:2010.07835*, 2020.

[25] Z. Meng and R. Wattenhofer, "A geometry-inspired attack for generating natural language adversarial examples," in *COLING*, D. Scott, N. Bel, and C. Zong, Eds., 2020.

[26] 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.

[27] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *CVPR*, 2016.

[28] X. Zhang, J. Zhao, and Y. LeCun, “Character-level convolutional networks for text classification,” *arXiv preprint arXiv:1509.01626*, 2015.

[29] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, “Learning word vectors for sentiment analysis,” in *ACL*, 2011.

[30] S. Garg and G. Ramakrishnan, “BAE: BERT-based adversarial examples for text classification,” in *EMNLP*, 2020.

[31] V. Malik, A. Bhat, and A. Modi, “Adv-OLM: Generating textual adversaries via OLM,” in *EACL*, 2021.

Appendix A

Appendix

A.1 Adversarial Examples of Geometry Attack for Contrastive Loss

In Table A.1, we show adversarial examples generated by our Geometry Attack for contrastive loss.

`Original`	Zurich employees plead guilty in probe new york (reuters) - two senior insurance underwriters at zurich american insurance co pleaded guilty on tuesday to misdemeanors related to bid-rigging in the insurance market.
`Adversarial`	Zurich employees plead guilty in probe new york (reuters) - two senior insurance agents at zurich american insurance co testified guilty on tuesday to violations related to bid-rigging in the insurance market.
`Original`	Allie & Me Allie & Me is a 1997 film directed by Michael Rymer. It stars Lyndie Benson and Linda Darnell. It won an award at the 1997 RiverRun International Film Festival.
`Adversarial`	Allie & Me Allie & Me is a 1997 theatre guide by Michael Rymer. It stars Lyndie Benson and Linda Darnell. It won an award at the 1997 RiverRun International Film Festival.
`Original`	It has a stunning lack of even rudimentary traces of realism . almost every war movie cliche appears in this film and is done badly. on the other hand , i wouldn't have watched it to the end if it hadn' t been so remarkably bad that it amused me.
`Adversarial`	It has a stunning lack of even joyless traces of realism . almost every war movie cliche appears in this film and is done erroneously. on the other hand , i wouldn't have watched it to the end if it hadn' t been so remarkably bad that it flabbergasted me.
`Original`	New report links reputed kingpin to murder fifteen years ago, American journalist Todd Smith was brutally beaten and executed after he ventured into peru's jungle to investigate links between shining path guerrillas and the cocaine trade.
`Adversarial`	New report links reputed suicide to murder fifteen years ago, American journalist Todd Smith was brutally beaten and executed after he ventured into peru's jungle to learn links between shining path guerrillas and the cocaine trade.
`Original`	Black watch troops move into position the first units of a black watch battlegroup are due to arrive today in their new positions south of baghdad as tony blair indicated that more british troops may replace them in the american - controlled zone before the end of the year.
`Adversarial`	Black watch troops move into place the first units of a black watch operation are due to arrive today in their new positions south of baghdad as tony blair indicated that more british troops may replace them in the american - controlled zone before the end of the year.

Table A.1: Adversarial samples generated by Geometry Attack for contrastive loss. Blue words in the original examples are replaced by red words in the adversarial examples.