Improving the Quality of Adversarial Examples via Contrastive Learning and Pretraining

Final Year Project Second Term Report

by

Yung-chieh Huang

Supervised by

Professor Michael Rung-Tsong Lyu

Department of Computer Science and Engineering

The Chinese University of Hong Kong

Contents

Chapter 1

Introduction

Deep learning has gained great success in the past few years. From image recognition to virtual assistant, it is undeniable that deep learning has made our lives more convenient than ever such as autonomous driving and medical diagnosis. Deep learning has also greatly influenced the field of natural language processing (NLP), brought about many new applications such as machine translation and sentiment analysis. Yet recently, researchers have found that several seemingly robust language models are vulnerable to adversarial attacks that replacing a token by other synonyms can dramatically mislead the output of language models.

1.1 Adversarial Attack

Adversarial attack is an approach to test the robustness of machine learning models. There are generally two kinds of attacks in the literature. One is the white-box attacks, which consider the white-box setting where attackers can access the architectures and parameters of the victim models. The other is the black-box attacks, which focus on the black-box situation where attackers fail to get access to the specifics of the victim models. Black-box attacks are more applicable than white-box counterparts for real-world systems, and there exist two basic attacking methodologies: query-based and transfer-based attacks. Query-based attacks interact with the victim model to generate adversarial examples and may

incur excessive queries. In contrast, transfer-based attacks craft adversarial examples with a local source model and do not need to query the victim model. In the NLP community, researchers focus more on query-based black-box attacks, because of a high attack success rate compared with transfer-based attacks.

Adversarial examples are the generated test cases by adversarial attack methods to test victim models. An adversarial example is created by intentionally applying perturbations to an original sentence such that a victim model correctly classifies the original sentence but misclassifies the adversarial example with high confidence. Furthermore, the adversarial example should satisfy the constraint that humans cannot differentiate the difference between the adversarial example and the original one. Therefore, a well-crafted adversarial example should have minimum perturbations and preserve the structure and characteristics of the original to be as close to the original sentence as possible.

The field of adversarial attack has been well explored in image recognition, yet only until recently do researchers start looking into adversarial attack in NLP. This is due to the challenges in generating text adversarial examples. The image domain is continuous that image adversarial examples can be created by simply perturbing pixel values utilizing the gradient information of the victim models. Those changes in pixel values are hardly distinguishable to human perception but are effective in fooling machine learning models (Szegedy et al. 2013; Goodfellow, Shlens, and Szegedy 2014). However, the data in the natural language processing community are texts, which are a combination of discrete words. Therefore, gradient descent is not applicable to generate adversarial examples. In addition, altering a word in a sentence can change the whole semantic drastically, which increases the difficulty of generating textual adversarial examples.

In general, adversarial attacks in the NLP field can be roughly divided into three categories: character-level, token-level, and sentence-level. Character-level adversarial attacks

perturb the characters of the original sentences by replacing, removing, or inserting characters. The generated adversarial examples can be viewed as adding typos to the original sentences. Character-level attacks can successfully mislead the language models, but the generated adversarial examples are extremely hard to read and understand. Token-level adversarial attacks perturb the tokens of the original sentences by replacing, removing, or inserting tokens. The whole adversarial example is more readable than character-level attacks, but improper substitution of a token can drastically change the whole semantics. Sentence-level adversarial attacks rephrase one sentence to mislead the language models while keeping the semantics of the original sentence. Sentence-level adversarial attacks can generate syntactic and semantic adversarial examples, but it suffers from large quantities of computation complexity. In this project, we mainly focus on the research of token-level adversarial attacks.

A token-level adversarial attack method can be treated as a composition of four components, namely a goal function, a transformation method, a search method, and a set of constraints (Roth et al. 2021).

Goal Function

Goal function specifies what result the attack model intends to achieve. For classification tasks, the goal functions are usually either targeted or untargeted classification. Targeted classification tries to fool the victim model to classify an example to a specific class different to the correct class, while untargeted classification only aims to make the victim model misclassify to any other classes. In general, the goal function is the cross entropy loss to measure the difference between the prediction of the adversarial examples and real labels.

Transformation

Transformation is how the attack method generates adversarial examples by transforming the original sentences. For character-level adversarial attacks, One type of transformation

is to randomly replace characters in a word or mimic typos (J. Gao et al. 2018; J. Li et al. 2018). These character-level attacks are useful in everyday life applications in a sense that character errors can easily occur. However, they don't actually test the victim model's ability to capture sentiment.

For token-level adversarial attacks, the transformations are often done by replacing tokens, inserting tokens, deleting tokens, or changing token orders. One common type of transformation is to replace a word with a nearby word in a word embedding space (Jin et al. 2020). The intention of such transformation is to find a synonym replacement. However, in word embedding, synonyms and antonyms are clustered together due to improper word embedding function, Therefore, the semantic of the output adversarial example may be opposite to the original sentence. In addition, this method does not take the context and fluency into consideration, resulting in replacements that do not fit in well.

To solve the problem, the latest trend of transformation is to use language models such as BERT (Devlin et al. 2018) and RoBERTa (Liu et al. 2019) to do masked language modelling and generate replacements (Garg and Ramakrishnan 2020; L. Li et al. 2020). The advantage of this approach is that language models take the sentence context into account, so the output adversarial examples will be more natural.

Search Method

Search method aims to find the important tokens to be perturbed in the original sentences and compute the order of replacing tokens in a sentence, in other words, the sequence of applying transformation to tokens. It could be in document order, in random order, or in importance ranking from most important to least important. The search method is relevant to the efficiency of the attacking method. If the search method is effective, the query number of the adversarial attack can be reduced.

Constraints

Constraints is a set of rules for generated adversarial examples, which guarantees the perturbation is imperceptible for human. Part of Speech (POS) constraint requires replacements to have the same POS tag as the replaced tokens to ensure correct grammar, which keeps the generated adversarial examples are readable and understandable by human. Semantic similarity constraint aims to maintain the semantic of the original sentence, which guarantees the true label of the adversarial examples is the same as the original one. There are also edit distance constraint, performance constraint, to name a few for regularizing the imperceptibility.

1.2 Motivation

The adversarial examples that state-of-the-art attack methods generate are of low quality, though they utilize user studies to verify the quality of their generated examples. Despite the superior attacking performances shown in the papers, after running the attack models ourselves, we observe that most of the adversarial examples the models generate are inconsistent and cause semantic loss. We regard the two main problems are opposite semantic replacements and irrelevant replacements.

Opposite semantic replacements are antonyms of the original word that alter the semantics of the sentence. Take a look at the examples in Table 1.1 generated by BAE (Garg and Ramakrishnan 2020) attacking victim model BERT-base. Here, "triviality" is replaced with "beauty", which totally changes the semantics of the original sentence from negative to positive. Therefore, even though this is a successful attack on the system, we say this is an invalid attack. The true label of the adversarial example is not the same as that of the original example and the label is flipped as the semantics change. Although the prediction of the model is varied, it is coherent with the true label of the adversarial example, which does not cause a prediction error. Therefore, we conclude that opposite semantic replacements will not cause an error contributing to a false positive, so we should

Original sentence	no amount of good intentions is able to overcome the triviality of the story	Negative (100%)
Adversarial example	no amount of good intentions is able to overcome the beauty of the story	Positive (99%)

Table 1.1: Examples of attacking the BERT-base classifier by BAE.

Original sentence	watching spirited away is like watching an eastern imagination explode	Positive (99%)
Adversarial example	watching spirited away is like watching an eastern magazine explode	Negative (100%)

Table 1.2: Examples of attacking the BERT-base classifier by BAE.

eliminate opposite semantic replacements.

Irrelevant replacements are replacements that have nothing to do with the original word or the sentence context. Irrelevant replacement makes it hard to understand the meaning of the adversarial example because the replaced token is not readable or improper in the sentence. The adversarial examples containing irrelevant replacements are actually the sentences out of the domain of the tasks or the domain of natural language. In this other example in Table 1.2, "imagination" is replaced with an out-of-context word "magazine". The two words are not synonyms, and "magazine" disrupted the semantics of the original sentence. Such an unnatural sentence does not appear in real-life applications, thus this is also viewed as an invalid attack. The generated adversarial examples are out of a domain that is hard to read and understand for humans, which is hard to give a clear label of the adversarial example. Therefore, irrelevant replacements also does not cause an error contributing to a false positive, so we should eliminate irrelevant replacements as well.

1.3 Objective

The goal of this project is to overcome the flaws in previous works and generate high-quality adversarial examples. That is to say, we want our attack method to generate examples to be free from opposite semantic or out-of-context replacements and at the

same time maintain fluency. Furthermore, we expect our attack model to have a higher successful attack rate and lower perturbation than other state-of-the-art attack models. We want to emphasize that this work is designed to improve the quality of generated adversarial examples, but we should also improve the ability to attack other models. Therefore, we design an attacking method to achieve both of the objectives by elaborate algorithm designs. In order to overcome irrelevant replacements, we pretrain the language model to generate replaced tokens on the task-related datasets to learn the distribution on the task domain. In order to overcome opposite semantic replacements, we utilize the concept of contrastive learning to retrain the language model so that it learns a better representation to distinguish the synonym and antonym in the embedding space. Furthermore, we deploy an iterative training framework to balance the quality of generated adversarial examples and the attack success rate to achieve a high-quality and effective attacking method.

1.4 Contribution

In this project our contributions are three-folded. First, we figured out that the reason why opposite semantic replacements exist is due to the embedding space of language models, and why out-of-context replacements are so difficult to avoid is because attack methods are too general. Unlike previous works which only emphasize exterior factors such as adding more constraints, we go further and alter the interior components, which is the language model used in transformation. Secondly, we are the first to generate adversarial examples via a combination of contrastive learning and pretraining. Through pretraining, our attack model is domain-specific, so that instead of generating general replacements, it generates replacements that are related to the sentence context. With the help of contrastive learning, our attack model is capable of separating synonyms and antonyms in the embedding space, which will contribute to generating adversarial examples that are semantically similar to the original sentence. Finally, we develop an iterative training method that can largely improve the performance. That is, not only do we have better adversarial examples, but our results also outrun state-of-the-art baselines.

Chapter 2

Related Work

2.1 Adversarial Attack for Text

Adversarial attack in NLP has only started to gain its popularity recently due to the difficulty of generating adversarial text examples caused by the nature of text. As mentioned in the previous section, different attack methods have different compositions according to their needs, while they mainly differ in transformation. Our project is inspired by two works: BERT-Attack (L. Li et al. 2020) and BAE (Garg and Ramakrishnan 2020), both of which use language model in transformation.

BERT-Attack aims to generate adversarial examples that are fluent and semantically preserved while having high success rate and minimum perturb percentage. The attack model finds the vulnerable words in a input sentence by masking each word and calculate the output logit. Then, in vulnerability order, use BERT masked language model to generate replacement for each word. BERT-Attack indeed has minimum perturbation. However, when we recreate its experiment, we see a lot of sub-words in its generated adversarial examples. This is because BERT-Attack tokenizes a sentence into sub-word tokens, and some sub-words are not processed correctly.

BAE is the other work we reference to. The objective of BAE is to beat previous baselines

on text classification datasets at the same time improve grammatically and semantic coherence via replaceing and/or inserting tokens. BAE uses BERT to predict masked tokens, and apply constraints such as sentence similarity and part of speech tag to ensure fluency. Yet, we still find its adversarial examples unnatural when rerunning its experiment.

Although our adversarial attack method is similar to BAE except for the transformation part, we aim to solve the core internal problems of Bert-based attacks. We are the first to propose utilizing pretraining and contrastive learning to improve the ability of the embedding spaces to further boost the quality of adversarial examples.

2.2 Pretraining

One of the reasons why pretrained language models are so powerful is because they are pretrained on very large corpora from various domains. To illustrate, the pretraining corpus of BERT (Devlin et al. 2018) consists of two huge corpora, BookCorpus and English Wikipedia; RoBERTa (Liu et al. 2019) is pretrained on over 160GB of uncompressed text from books, wikipedia, news articles, web content, and stories.

Recent studies (Gururangan et al. 2020; J. Lee et al. 2020) have shown that a second phase pretraining on domain-specific corpora can largely increase the performance of a language model on tasks in that domain. This is because language models are initially pretrained to perform general tasks, the general corpora may not be diverse enough to include domain specific terms.

After reading these studies, we decided to incorporate this concept and pretrain BERT on domain-specific datasets, hoping it will then generate less out-of-context replacements. The domain we mainly focus on is sentiment analysis, movie reviews to be more specific. We also test our attack model on the natural language inference in later stages of the experiment.

We utilize the strong power of pretraining to regularize the language model in the task-specific domain. We select datasets that have the same tasks as the victim models. For example, we use the IMDb dataset to pretrain the model while the victim model is trained on the MR dataset. Both datasets are related to movie review sentiment analysis, thus they are in the same task domain. At the same time, we pretrain the model on task-related datasets, which can balance the overfitting and underfitting issues. Pretraining on the same dataset may cause overfitting to the dataset and the generated adversarial examples may not mislead the victim models. If we pretrain on the datasets of other tasks, the generated replaced tokens are still suffering from the problem of irrelevant replacements.

2.3 Contrastive Learning

One difficulty we face when conducting experiments is that language models cannot distinguish synonyms and antonyms because synonyms and antonyms are clustered together in the embedding space. Therefore, we incorporate contrastive learning in our model. The idea of contrastive learning is to pull similar neighbors together and push away others. We hope by coupling contrastive learning and BERT, BERT can pull together synonyms, push away antonyms, and generate less opposite semantic replacements.

SimCSE (T. Gao, Yao, and Chen 2021) is a simple contrastive sentence embedding framework we use in this project. Its objective is to improve sentence embedding performance on semantic textual similarity tasks. The paper presents two approaches: unsupervised and supervised. Unspuervised SimCSE predicts an input sentence twice with independently sampled dropout masks to obtain two embeddings as positive pairs. Supervised SimCSE is done by treating entailment pairs from natural language inference datasets as positive instances and add contradiction pairs as hard negatives.

Another contrastive learning framework we reference to is CLINE (Wang et al. 2021).

The goal of CLINE is to train a language model that is both defensive against adversarial attacks and sensitive to semantic changes by using both adversarial and contrastive examples, because often times when trying to improve the robustness against adversarial attacks, the performance on contrastive examples fails. CLINE generates positive sentence pairs with the same semantics by replacing extracted words with synonyms, and generate negative sentence pairs with opposite semantics by replacing them with antonyms and random words.

We also adopt contrastive learning to improve the sentence embedding performance on the synonym and antonym. Instead of directly utilizing the pretrained contrastive learning model, we generate contrastive sentence pairs such that the synonym sentence replaces the original sentence with synonyms, while the antonym sentence replaces the original sentence with antonyms. Then we apply contrastive learning to further train models to improve their embedding performance in disentangling synonyms and antonyms.

Chapter 3

Methodology

Despite setting multiple constraints and using state-of-the-art language model in transformation in an attack model to ensure the quality of adversarial examples, we can still see a lot of out-of-context and opposite semantic replacements, as demonstrated in previous sections. This is due to the nature of the language models. These contextual word embedding models, such as BERT (Devlin et al. 2018) and ELMo (Peters et al. 2018), are able to create a context-sensitive embedding for each word in a given sentence from pretraining on large text corpora. Unlike traditional word-level vector embeddings like word2vec (Mikolov et al. 2013) and GloVe (Pennington, Socher, and Manning 2014) that treat each word as a independent vector, contextual word embedding models have much more parameters that take the whole sentence context into consideration (Alsentzer et al. 2019). Nevertheless, their embedding spaces are still unable to recognize antonyms since analogies are grouped together. Our proposed method can solve this problem. We solve the problem from two aspects: out-of-context replacements and opposite semantic replacements

For out-of-context replacements, we use pretraining to enhance the language model's ability to generate more domain-specific replacements. We utilize the strong power of pretraining to regularize the language model in the task-specific domain. We carefully

select datasets that have the same tasks as the victim models. For example, we make use of the IMDb dataset to pretrain our model while the victim model is trained on MR dataset. Both IMDb and MR datasets are related to movie review sentiment analysis, so they are in the same task domain. Since we pretrain the model on task-related datasets, our model can balance the overfitting and underfitting issues. On the other hand, if we pretrain on the same dataset, it may cause overfitting to the dataset and the generated adversarial examples may not be able to mislead the victim models. Furthermore, if we pretrain on datasets of other tasks, the generated replaced tokens will suffer from the problem of irrelevant replacements.

For opposite semantic replacements, we adopt the concept of contrastive learning that helps distinguish synonyms and antonyms. We utilize contrastive learning to improve the sentence embedding performance on synonyms and antonyms. Instead of directly utilizing the pretrained contrastive learning model, we generate contrastive sentence pairs such that the synonym sentence replaces the original sentence with synonyms, while the antonym sentence replaces the original sentence with antonyms. Then we apply contrastive learning to further train our model to improve its embedding performance in disentangling synonyms and antonyms. In this way, the embedding of synonyms and antonyms is separable in the embedding space and previous constraints like cosine similarity will work well on contrastive trained language models.

Furthermore, directly composing pretraining and contrastive learning together causes the problem of overfitting. If we compose the objective function together, the trained model will fit well with the contrastive and language model objectives, which will lose the attacking ability to generate adversarial examples. Moreover, if we stack pretraining and contrastive learning together, the trained model is underfitting that is still suffering from the aforementioned problems. Therefore, we propose an iterative training method to combine contrastive learning and language model pretraining to approach the optimal while

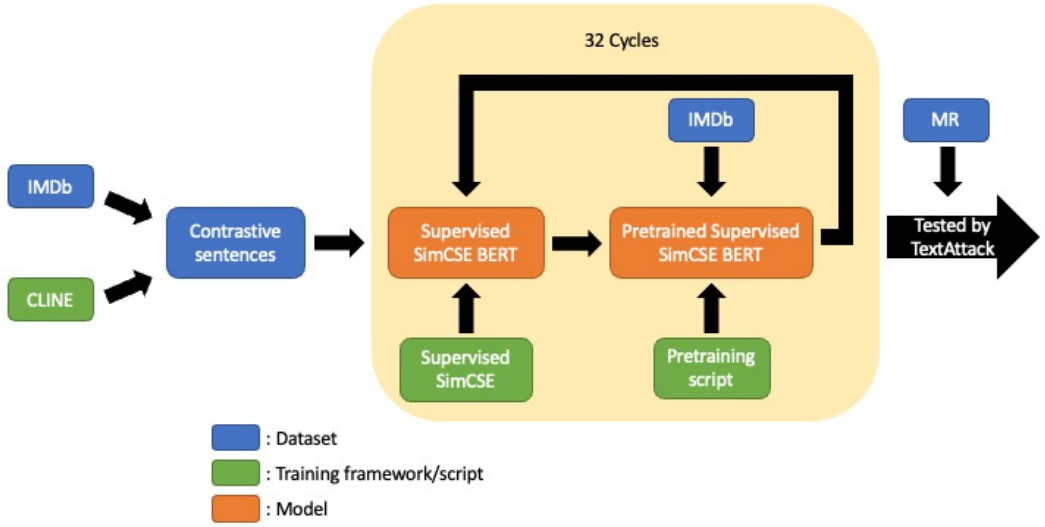

Figure 3.1: Complete flow of creating our own BERT.

we balance well between the quality of generated adversarial examples and the target to craft adversarial examples.

The whole work flow of our proposed attack method is shown below. First, we create our own contrastive sentence dataset using the IMDb dataset and CLINE. Next, we run contrastive learning and pretraining iteratively for 32 cycles. The initial checkpoint we operate on is the original supervised SimCSE BERT. In each cycle, we train the input model on the dataset we created with supervised SimCSE to produce a temporary supervised SimCSE BERT. This step strengthens the model's ability of separating synonyms and antonyms in the embedding space and avoid generating opposite semantic replacements. Then, we pretrain the temporary supervised SimCSE BERT on IMDb to create a pretrained supervised SimCSE BERT. This pretrained supervised SimCSE BERT is later passed to the next cycle. Finally, when all 32 cycles are completed, we evaluate the final pretrained supervised SimCSE BERT with TextAttack on the MR dataset. Unlike other attack models, our attack model is domain-specific to movie reviews, which is beneficial to generating high quality examples. The complete flow of our method is visualized in

Figure 3.1. The datasets involved in the process are colored in blue, training frameworks and scripts we use are colored in green, and the models we create are colored in orange. Arrows indicate what elements are coupled to create another element.

Chapter 4

Experiments

4.1 Baselines

We choose three state-of-the-art token-level adversarial attack methods that have similar objectives with our project as our baselines.

BAE (Garg and Ramakrishnan 2020) is a black-box attack method that leverages contextual perturbations from a BERT masked language model. It inserts and replaces tokens by masking a portion of the original sentence, then uses BERT to generate grammatically correct and semantic coherence replacements. BAE is the baseline we mainly use since our model is derived from it.

TextFooler (Jin et al. 2020) is commonly used as a baseline in recent NLP adversarial attack papers as it claims to be a simple but strong baseline to generate adversarial text. TextFooler achieves its attack goals by applying multiple rule-based strategies, including word embedding synonym extraction, POS checking, and semantic similarity checking. When we run TextFooler with TextAttack (Morris et al. 2020), we find out that the average number of queries is 117.23, which is much higher than that of our attack model and BAE. The average number of queries indicates on average how many candidate replacements the attack model went through before achieving a successful attack, which is

related with the budget for attack. If the number is high, then the attack success rate will naturally be higher with a larger budget. Therefore, in order to make TextFooler comparable to our attack model and BAE, we set its word embedding minimum cosine similarity constraint to 0.75, and its USE threshold to 0.75. By doing so, the average number of queries is lowered to 58.36, which has a similar number with our method and other baselines.

PWWS (Ren et al. 2019) uses the word saliency and the classification probability to determine the word replacing order then greedily replace each word with its synonym. Similar to TextFooler, PWWS has a much higher average number of queries. Thus, we reduce the length of its synonym list to 25%. The averge number of queries is then lowered to 62.44.

4.2 Datasets

We use two movie review datasets for the sentiment classification domain: IMDb (Maas et al. 2011) and MR (Pang and L. Lee 2005). IMDb, also called Large Movie Review Dataset, contains 25,000 highly polar movie reviews for training, 25,000 for testing, and additional 50,000 unlabeled data. MR contains 5,331 positive and 5,331 negative reviews from Rotten Tomatoes. Both of these datasets can be found on Transformers (Wolf et al. 2020).

4.3 Set up

We implement our attack method on BAE (Garg and Ramakrishnan 2020). But instead of using an original BERT in transformation, we use our own BERT in order to see whether modifying BERT is the key to generating better examples that we utilize an interative way to pretrain and contrastive train the BERT.

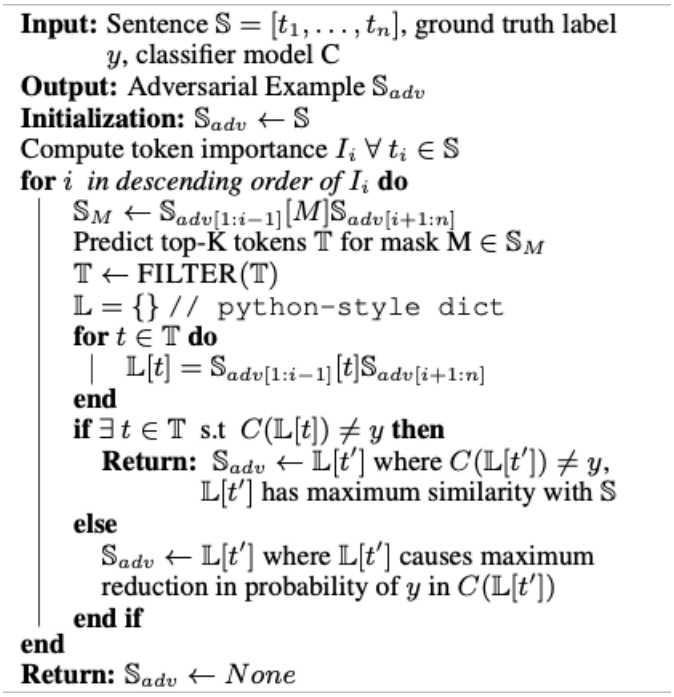

Figure 4.1: Token replacing algorithm.

We adopt the token replacing algorithm from BAE as shown in Figure 4.1. The goal function is untargeted classification. We replace the original BERT BAE uses to our own BERT in transformation. Our own BERT varies in different phases of experiment, we will elaborate more in later sections. We use greedy word swap for search method. Greedy word swap measures the importance of each token by masking it and calculate the decrease in probability of predicting the correct label. As for constraints, we include Part of Speech constraint and Universal Sentence Encoder of threshold 0.94 to ensure correct grammar and sentence semantic similarity.

We run attacks with TextAttack (Morris et al. 2020) in all our experiments. TextAttack is a framework designed to enable researchers evaluate different NLP attacks. Given an attack recipe (including a goal function, a transformation, a search method, a set of constraints), a victim model, and a dataset, TextAttack will generate adversarial examples from the dataset using the attack recipe and attack the victim model.

4.4 Results

In this chapter, we will show how our model has evolved as we continue to have new findings, as well as some ideas that we didn't end up incorporating. We first validate the effectiveness of solely utilizing pretraining to improve adversarial attacks in Section 4.4.1. Then we combine contrastive learning into the framework. We first deploy a pretrained contrastive learned BERT, then we pretrain the BERT following the previous step in Section 4.4.2. We also analyze the best number of pretraining steps to incorporate with contrastive learning. Furthermore, the pretrained contrastive learning model is trained based NLI datasets, which have a domain gap with the task of sentiment analysis. Therefore, we construct contrastive sentence pairs for our movie review task and retrain the contrastive BERT to improve the attacking performance in Section 4.4.3. In addition, we conduct experiments to validate the best way to generate contrastive sentence pairs for contrastive learning. Finally, we present our iterative training method for combining pretraining and contrastive learning in Section 4.4.4 where we also validate the best cycle numbers to fuse them together.

Moreover, we demonstrate some ideas we tried along the way but failed to improve the performance. In Section 4.4.5, we tried the idea of batch-sorted where we use all of the contrastive sentence pairs for an original sentence to contrastive train BERT to let it pay more attention to the differences between synonym and antonym. In Section 4.4.6, we tried to directly compose pretraining and contrative learning together to improve the performance. However, the performance is not as good as expected, we then conclude that the reason is over-fitting that the BERT fails to generate adversarial examples.

4.4.1 Pretraining Only

To test the effectiveness of pretraining in adversarial attack, we start by pretraining an original BERT-base on 50,000 unlabeled IMDb data with the pretraining script from the official BERT Github repository (https://github.com/google-research/bert) for 50,000 steps,

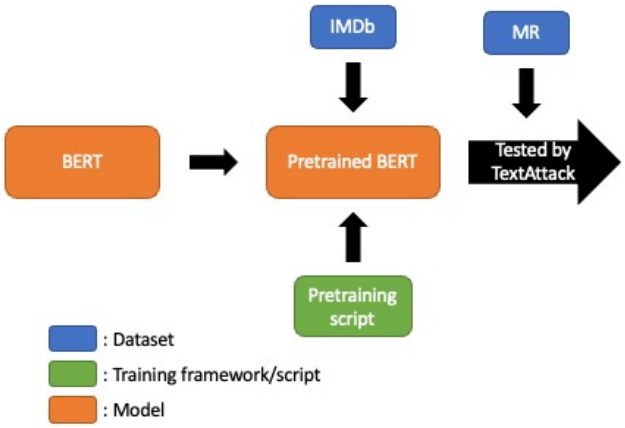

Figure 4.2: Pretraining only.

5,000 warmup steps, and all the other default parameters. We evaluate both BAE and our attack model with TextAttack. The dataset used here is MR and the victim model is BERT-base.

The results are shown in Table 4.1. Overall, our model has a slightly better result. The accuracy under attack is 0.2% lower and the attack success rate is 0.24% higher. The average perturbed word percentage is 0.54% lower, which indicates the replacements are

Dataset: MR		
	BAE	Ours
Number of successful attacks	473	**475**
Number of failed attacks	365	**363**
Number of skipped attacks	162	162
Original accuracy	83.8%	83.8%
Accuracy under attack	36.5%	**36.3%**
Attack success rate	56.44%	**56.68%**
Average perturbed word %	13.91%	**13.37%**
Average number of words per input	18.64	18.64
Average number of queries	63.49	63.19

Table 4.1: Evaluation of our pretraining only attack model versus BAE on MR dataset.

Original sentence	the movie is a little tired; maybe the original inspiration has run its course	Negative (100%)
BAE	the mind is a little tired; yet the original memory has continued its course	Positive (100%)
Ours	the beginning is a little tired; maybe the original tale has improved its course	Positive (88%)
Original sentence	one of the funnier movie in town	Positive (94%)
BAE	one of the funnier locations in town	Negative (97%)
Ours	one of the funnier scenes in town	Negative (99%)

Table 4.2: Examples of attacks by our pretraining only attack model versus BAE on BERT-base classifier.

Dataset: IMDb		
	BAE	Ours
Number of successful attacks	**583**	365
Number of failed attacks	**338**	556
Number of skipped attacks	79	79
Original accuracy	92.1%	92.1%
Accuracy under attack	**33.8%**	55.6%
Attack success rate	**63.3%**	39.63%
Average perturbed word %	4.06%	**3.43%**
Average number of words per input	233.5	233.5
Average number of queries	425.07	373.2

Table 4.3: Evaluation of our pretraining only attack model versus BAE on IMDb dataset.

effective, so less perturbations are needed to fool the victim model.

After studying the adversarial examples from both attack models, some presented in Table 4.2, we find out that pretraining BERT does have an effect on generating token replacements. The replacements generated by our model are more related to movies.

However, the problems we are trying to solve has not improved. There are still a considerable amount of opposite semantic and out-of-context replacements in the adversarial examples from our model. The slightly better result is not sufficient for proving our model is superior.

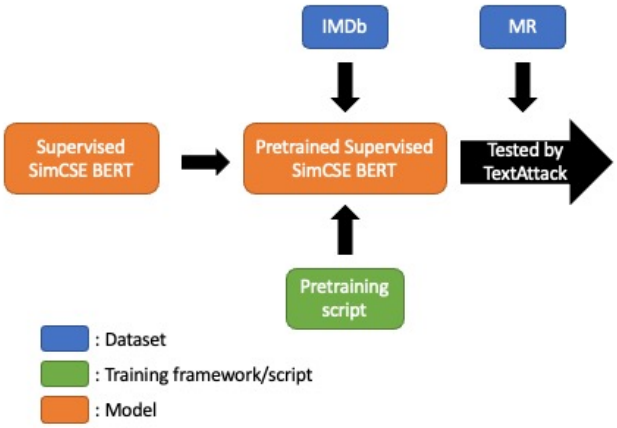

Figure 4.3: Contrastive learning and pretraining.

We also evaluate BAE and our model using IMDb dataset, the results are shown in Table 4.3. In contrast to using MR, here our model has a much lower attack success rate than BAE. The reason is because our BERT is pretrained on IMDb, so the replacements it generates are too similar to the original tokens that they are unable to fool the victim model.

4.4.2 Contrastive Learning and Pretraining

Next, we add contrastive learning to our attack model. Instead of pretraining an original BERT-base, now we pretrain supervised SimCSE BERT-base (T. Gao, Yao, and Chen 2021) on IMDb by running the same pretraining script as before for 50,000 steps, 5,000 warmup steps and all the other default parameters.

We discover that the result did not improve a lot. In fact, after analyzing the adversarial examples, we find out that there are still a lot of opposite semantic replacements, which is similar to the problem in the previous section. We suspect that this is due to

Dataset: MR						
	BAE	Ours (50,000)	Ours (25,000)	Ours (5,000)	Ours (2,500)	Ours (0)
Number of successful attacks	473	471	473	487	**501**	411
Number of failed attacks	365	367	365	351	**337**	427
Number of skipped attacks	162	162	162	162	162	162
Original accuracy	83.8%	83.8%	83.8%	83.8%	83.8%	83.8%
Accuracy under attack	36.5%	36.7%	36.5%	35.1%	**33.7%**	42.7%
Attack success rate	56.44%	56.21%	56.44%	58.11%	**59.79%**	49.05%
Average perturbed word %	13.91%	13.19%	**13.13%**	13.58%	13.17%	14.85%
Average number of words per input	18.64	18.64	18.64	18.64	18.64	18.64
Average number of queries	63.49	64.27	64.05	64.01	62.96	54.93

Table 4.4: Evaluation of our contrastive learning and pretraining attack model of different number of pretraining steps versus BAE on MR dataset.

excessive pretraining that the latter pretraining dominates the BERT training. SimCSE is pretrained too much that its contrastive learning characteristics are hiddened. Therefore, we lower the number of training steps and warmup steps.

From Table 4.4 and Figure 4.4 we can see, the one trained with 2,500 steps has a overall better performance, with 2.8% lower accuracy under attack and 3.35% higher attack success rate than BAE. This is also reflected in the generated adversarial examples, as shown in Table 4.5. Although opposite semantic and out-of-context replacements are not completely eliminated, we see less such replacements, and a much higher attack success rate.

4.4.3 CLINE Contrastive Sentences Augmentation

Now that we have proved our goal can be achieved by contrastive learning and pretraining, the next thing to do is to create contrastive sentence pairs using CLINE (Wang et al. 2021). The previous SimCSE model is pretrained on the NLI datasets, which has a domain gap between the tasks of NLI and sentement analysis. Therefore, we should create our own contrastive sentence pairs to eliminate the gap and further improve the

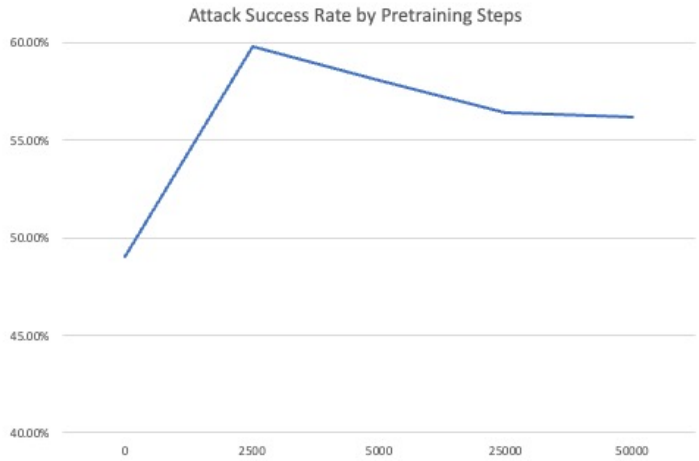

Figure 4.4: Attack success rate of our contrastive and pretraining attack model by different number of pretraining steps.

Original sentence	fans of the modern day hong kong action film finally have the worthy successor to a better tomorrow and the killer which they have been patiently waiting for	Positive (100%)
BAE	fans of the modern day hong kong action film finally have the only successor to a better tomorrow and the killer which they have been helplessly waiting for	Negative (99%)
Ours (50,000)	fans of the modern day hong kong action film finally have the disappointing successor to a better tomorrow and the killer which they have been patiently waiting for	Negative (51%)
Ours (25,000)		Failed
Ours (5,000)		Failed
Ours (2,500)	fans of the modern day hong kong action movie now have the usual successor to a better tomorrow and the killer which they have been already waiting for	Negative (83%)
Ours (1,000)	fans of the modern day hong kong action movie now have the usual successor to a better tomorrow and the killer which they have been completely waiting for	Negative (81%)

Table 4.5: Examples of attacks by our contrastive learning and pretraining attack model of different number of pretraining steps versus BAE on BERT-base classifier.

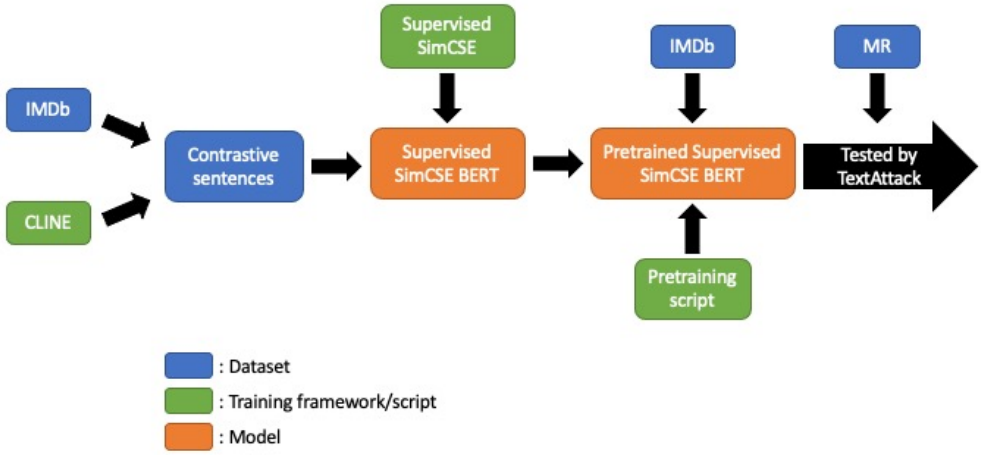

Figure 4.5: CLINE contrastive sentences augmentation.

performance. A sentence pair includes an original sentence, a semantically close sentence, and a semantically opposite sentence. CLINE generates the semantically close sentence by replacing words in the original sentence with synonyms, hypernyms and morphological changes, and generates the semantically opposite sentence by replacing words with antonyms and random words. We are not using SimCSE to create contrastive sentences because its algorithm is questionable.

To start with, we run the word replace script from the official CLINE Github repository (https://github.com/kandorm/CLINE) on the IMDb dataset to generate contrastive sentences of replace ratios 0.05, 0.1, 0.2, 0.4, 0.5, we regard the multiple ratios can generate diverse contrastive sentence pairs well. There are 25,000 sentences pairs for each replace ratio. Then, we combine sentence pairs of different replace ratios into a new training dataset and use the dataset to run the supervised SimCSE training script from the official SimCSE Github repository (https://github.com/princeton-nlp/SimCSE) to train a supervised SimCSE BERT. Finally, we pretrain the model for 2,500 steps like we did before on IMDb to get our final pretrained supervised SimCSE BERT.

Dataset: MR					
	Ours (0.05)	Ours (0.05 + 0.1)	Ours (0.05 + 0.1 + 0.2)	Ours (0.05 + 0.1 + 0.2 + 0.4)	Ours (0.05 + 0.1 + 0.2 + 0.4 + 0.5)
Number of successful attacks	500	504	499	505	**508**
Number of failed attacks	338	334	339	333	**330**
Number of skipped attacks	162	162	162	162	162
Original accuracy	83.8%	83.8%	83.8%	83.8%	83.8%
Accuracy under attack	33.8%	33.4%	33.9%	33.3%	**33%**
Attack success rate	59.67%	60.14%	59.55%	60.26%	**60.62%**
Average perturbed word %	13.69%	13.45%	13.37%	13.22%	**13.18%**
Average number of words per input	18.64	18.64	18.64	18.64	18.64
Average number of queries	63.57	64.42	63.22	62.3	62.58

Table 4.6: Evaluation of our contrastive learning and pretraining attack model of different number of created contrastive sentence pairs on MR dataset.

In order to test whether increasing the number and diversity of contrastive sentence pairs will have a positive effect on the result, we successively combine sentence pairs of different replace ratios in each round. To elaborate, the training dataset contains 25,000 pairs of replace ratio 0.05 in the first round; 25,000 of replace ratio 0.05 plus 25,000 of replace ratio 0.1 in the second round; 25,000 of replace ratio 0.05 plus 25,000 of replace ratio 0.1 plus 25,000 of replace ratio 0.2 in the third round; and so on.

Table 4.6 and Figure 4.6 show that this data augmentation method can indeed benefit our attack model. Although we can change the replace ratio to generate more contrastive sentence pairs to further improve the performance, we believe more contrastive sentence pairs will consume more time to generate sentence pairs and training the model. Therefore, we keep the selection of the replace ratios to balance the performance and time consumption. From Table 4.7 we can see, compared to the previous version (contrastive learning and pretraining 2,500 steps), now our model has an even better performance, with 0.7% lower accuracy under attack and 0.83% higher attack success rate. Tables 4.8 and 4.9 are

Figure 4.6: Attack success rate of our contrastive learning and pretraining attack model by different number of contrastive sentence pairs.

Dataset: MR				
	BAE	Ours (pre-training only)	Ours (contrastive pretrain 2,500)	Ours (0.05 + 0.1 + 0.2 + 0.4 + 0.5)
Number of successful attacks	473	475	501	**508**
Number of failed attacks	365	363	337	**330**
Number of skipped attacks	162	162	162	162
Original accuracy	83.8%	83.8%	83.8%	83.8%
Accuracy under attack	36.5%	36.3%	33.7%	**33.0%**
Attack success rate	56.44%	56.68%	59.79%	**60.62%**
Average perturbed word %	13.91%	13.37%	**13.17%**	13.18%
Average number of words per input	18.64	18.64	18.64	18.64
Average number of queries	63.49	63.19	62.96	62.58

Table 4.7: Evaluation of different versions of our attack model versus BAE on MR dataset.

Original sentence	the strength if the film lies in its two central performances by sven wolter as the stricken composer and viveka seldahl as his desperate violinist wife.	Positive (100%)
BAE		Failed
Ours (pre-training only)		Failed
Ours (contrastive pretrain 2,500)		Failed
Ours (0.05)	the weakness of the film lies in its two ventral performances by sven wolter as the stricken composer and viveka seldahl as his desperate violinist wife.	Negative (100%)
Ours (0.05 + 0.1)	the point if the film lies in its two main performance by sven wolter as the stricken composer and viveka seldahl as his poor violinist wife.	Negative (72%)
Ours (0.05 + 0.1 + 0.2)	the point if the film lies in its two main performances by sven wolter as the stricken composer and viveka seldahl as his stupid violinist wife.	Negative (99%)
Ours (0.05 + 0.1 + 0.2 + 0.4)		Failed
Ours (0.05 + 0.1 + 0.2 + 0.4 + 0.5)	the point if the film lies in its two main performance by sven wolter as the stricken composer and viveka seldahl as his poor violinist wife.	Negative (72%)

Table 4.8: Examples of attacks by our contrastive learning and pretraining attack model of different number of created contrastive sentence pairs on BERT-base classifier.

examples extracted from the experiments to demonstrate how by applying this CLINE data augmentation method, our model is capable of generating semantic and syntactic replacements.

4.4.4 Iterative Contrastive Learning and Pretraining

We have proved the efficacy of pretraining and contrastive learning in sections 4.4.1 and 4.4.2 respectively. Up till now, we have been running the two methods sequentially. That is, we finish running the supervised SimCSE training script with in total 125,000 contrastive sentence pairs from 5 different replace ratios before doing pretraining for 2,500

Original sentence	master of disguise runs for only 71 minutes and feels like three hours.	Negative (100%)
BAE		Failed
Ours (pre-training only)		Failed
Ours (contrastive pretrain 2,500)	master of silence begins for only 12 mm and ends as 11 hours.	Positive (74%)
Ours (0.05)	master of death is for only 12 minutes and continues as 11 hours.	Positive (86%)
Ours (0.05 + 0.1)	master of death is for only 12 minutes and ends as 11 hours.	Positive (84%)
Ours (0.05 + 0.1 + 0.2)		Failed
Ours (0.05 + 0.1 + 0.2 + 0.4)	master of dracula suffers for only 12 minutes and succeeds as 11 hours.	Positive (98%)
Ours (0.05 + 0.1 + 0.2 + 0.4 + 0.5)	master of shadows is for only 12 minutes and succeeds as 11 hours.	Positive (98%)

Table 4.9: Examples of attacks by our contrastive learning and pretraining attack model of different number of created contrastive sentence pairs on BERT-base classifier.

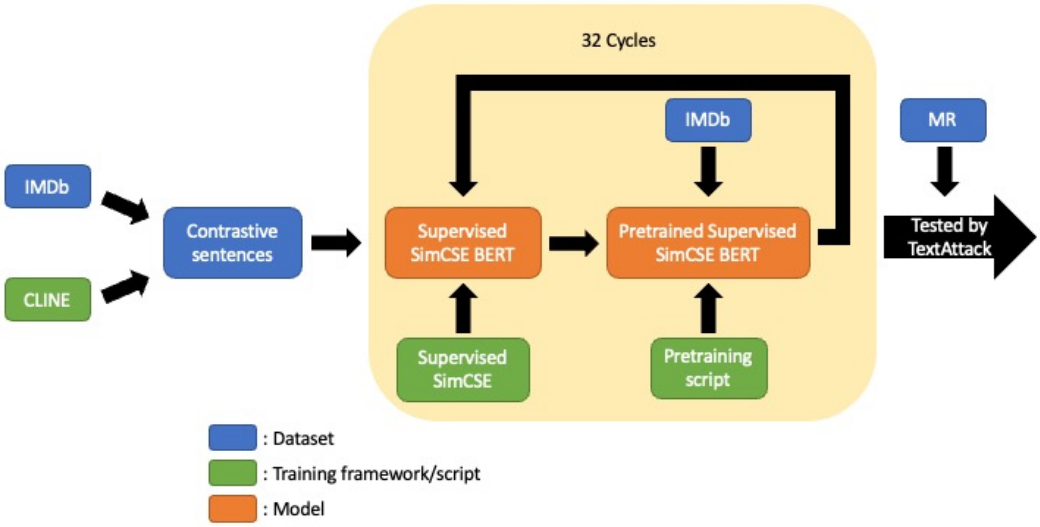

Figure 4.7: Iterative contrastive learning and pretraining.

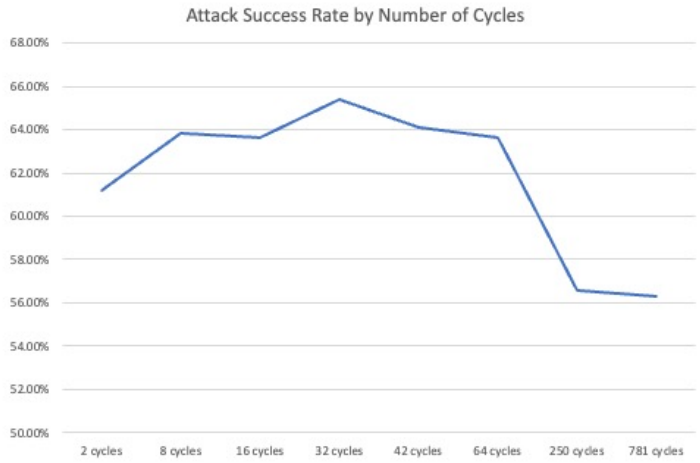

Figure 4.8: Attack success rate of our iterative attack model by different number of cycles.

steps. Nevertheless, we discover that we can achieve a much better result by running contrastive learning and pretraining iteratively. Doing so allows our model to fully utilize the best of both methods as it reduces the negative effect of excessive pretraining on contrastive learning.

Say we have n cycles. In the first cycle, we run the supervised SimCSE training script to train an original supervised SimCSE BERT (https://huggingface.co/princeton-nlp/sup-simcse-bert-base-uncased) on 125,000$/n$ contrastive sentences pairs. After that, we pretrain the model for 2,500$/n$ steps. Then for the second cycle, we take the pretrained supervised SimCSE BERT from the first cycle and train it on another 125,000$/n$ contrastive sentence pairs, then pretrain it for 2,500$/n$ more steps. We iteratively run contrastive learning and pretraining like that until all n cycles are completed.

We find out from the results in Table 4.10 and Figure 4.8 that 32 is the optimal number of cycles, any more than that will show signs of over-fitting. From Table 4.11 we see a significant improvement compared to all previous versions of our model, with 4% lower

Dataset: MR				
	Ours (2 cycles)	Ours (8 cycles)	Ours (16 cycles)	Ours (32 cycles)
Number of successful attacks	513	535	533	**548**
Number of failed attacks	325	303	305	**290**
Number of skipped attacks	162	162	162	162
Original accuracy	83.8%	83.8%	83.8%	83.8%
Accuracy under attack	32.5%	30.3%	30.5%	**29.0%**
Attack success rate	61.22%	63.84%	63.6%	**65.39%**
Average perturbed word %	13.4%	12.34%	12.01%	**11.83%**
Average number of words per input	18.64	18.64	18.64	18.64
Average number of queries	62.91	59.98	59.14	57.68

Dataset: MR					
	Ours (32 cycles)	Ours (42 cycles)	Ours (64 cycles)	Ours (250 cycles)	Ours (781 cycles)
Number of successful attacks	**548**	537	533	474	472
Number of failed attacks	**290**	301	305	364	366
Number of skipped attacks	162	162	162	162	162
Original accuracy	83.8%	83.8%	83.8%	83.8%	83.8%
Accuracy under attack	**29.0%**	30.1%	30.5%	36.4%	36.6%
Attack success rate	**65.39%**	64.08%	63.6%	56.56%	56.32%
Average perturbed word %	**11.83%**	12.21%	12.22%	12.97%	13.31%
Average number of words per input	18.64	18.64	18.64	18.64	18.64
Average number of queries	57.68	58.0	54.43	36.65	37.43

Table 4.10: Evaluation of our iterative attack model of different number of cycles on MR dataset.

Dataset: MR				
	Ours (pre-training only)	Ours (contrastive pretrain 2,500)	Ours (0.05 + 0.1 + 0.2 + 0.4 + 0.5)	Ours (32 cycles)
Number of successful attacks	475	501	508	**548**
Number of failed attacks	363	337	330	**290**
Number of skipped attacks	162	162	162	162
Original accuracy	83.8%	83.8%	83.8%	83.8%
Accuracy under attack	36.3%	33.7%	33.0	**29.0%**
Attack success rate	56.68%	59.79%	60.62	**65.39%**
Average perturbed word %	13.37%	13.17%	13.18%	**11.83%**
Average number of words per input	18.64	18.64	18.64	18.64
Average number of queries	63.19	62.96	62.58	**57.68**

Table 4.11: Evaluation of different versions of our attack model on MR dataset.

Dataset: MR				
	BAE	PWWS	TextFooler	Ours (32 cycles)
Number of successful attacks	473	434	531	**548**
Number of failed attacks	365	404	307	**290**
Number of skipped attacks	162	162	162	162
Original accuracy	83.8%	83.8%	83.8%	83.8%
Accuracy under attack	36.5%	40.4%	30.7%	**29.0%**
Attack success rate	56.44%	51.79%	63.37%	**65.39%**
Average perturbed word %	13.91%	16.0%	20.78%	**11.83%**
Average number of words per input	18.64	18.64	18.64	18.64
Average number of queries	63.49	62.44	58.36	57.68

Table 4.12: Evaluation of our iterative attack model versus baselines on MR dataset.

Original sentence	one of the funnier movies in town.	Positive (94%)
BAE	one of the funnier locations in town.	Negative (97%)
PWWS	matchless of the funnier movies in town.	Negative (100%)
TextFooler	one of the funnier kino in town.	Negative (88%)
Ours (32 cycles)	one of the funnier scenes in town.	Negative (99%)

Table 4.13: Examples of attacks by our iterative contrastive learning and pretraining attack model versus baselines on BERT-base classifier.

accuracy under attack, 4.77% higher attack success rate, and 1.35% lower average perturbed word percentage from the previous version. Our model also outperforms all three baselines, results shown in Table 4.12. In addition, since our model is domain-specific, it is capable of generating somewhat appropriate replacements given limited context. As illustrated in Table 4.13, "scenes" is relatively more relevant to "movies" compared to the other replacements generated by baselines.

4.4.5 Batch-sorted Contrastive Sentences

We set the batch size to 16 when running the supervised SimCSE training script, that means the model reads in 16 contrastive sentence pairs generated from different original

Dataset: MR		
	Ours (32 cycles)	Ours (32 cycles + batch-sorted)
Number of successful attacks	**548**	543
Number of failed attacks	**290**	295
Number of skipped attacks	162	162
Original accuracy	83.8%	83.8%
Accuracy under attack	**36.3%**	29.5%
Attack success rate	**65.39%**	64.8%
Average perturbed word %	11.83%	**11.65%**
Average number of words per input	18.64	18.64
Average number of queries	57.68	56.23

Table 4.14: Evaluation of our batch-sorted iterative attack model versus our iterative attack model on MR dataset.

sentences each time. Now, we want to know whether we can double the effect of contrastive learning if a batch contains 16 distinct sentence pairs all derived from the same original sentence.

To test this hypothesis, for each original sentence from the IMDb dataset, we run the word replace script from CLINE of 16 different replace ratios to create 16 nonidentical sentence pairs and sort them together. Then we apply the 32-cycles iterative training method. However, from the results in Table 4.14, it seems like this idea is trivial to our model, so we decided not to continue with this idea.

4.4.6 Merged Contrastive Learning and Pretraining

Seeing the success of the iterative training method, we are curious to know if we can achieve a better performance by merging contrastive learning and pretraining into one step. SimCSE comes with an auxiliary masked language modeling (MLM) function. According to the paper, MLM helps SimCSE avoid catastrophic forgetting of token-level knowledge, and adding it can improve the performance on transfer tasks modestly but not on sentence-level semantic textual similarity tasks. The loss is defined as $l_{contrastive} + \lambda \cdot l_{mlm}$, where λ is the MLM weight. Thus we include this function when training SimCSE, and we no

Dataset: MR		
	SimCSE MLM weight = 0.02	SimCSE MLM weight = 0.1
Number of successful attacks	426	398
Number of failed attacks	412	440
Number of skipped attacks	162	162
Original accuracy	83.8%	83.8%
Accuracy under attack	41.2%	44.0%
Attack success rate	50.84%	47.49%
Average perturbed word %	13.82%	14.15%
Average number of words per input	18.64	18.64
Average number of queries	54.01	57.74

Table 4.15: Evaluation of SimCSE with MLM of different weights on MR dataset.

longer do iterative training or pretraining.

We set the MLM weight to 0.02 as it is the ratio of pretraining to contrastive learning in our previous experiments (2500:125000). We also try 0.1 since it is the ratio with the best performance in the SimCSE paper. The results are shown in Table 4.15. To our surprise, the results are not good. Besides, there are a lot of unnatural replacements, such as fragmented words, in the generated adversarial examples.

We conjecture that this poor result is due to incompatible data formats. As mentioned before, data is passed to SimCSE in the form of sentence pairs. A sentence pair contains an original sentence, a semantically similar sentence, and a semantically opposite sentence. We speculate this data format confuses MLM because MLM usually takes in one sentence at a time. Therefore, we modify the training script so that MLM only reads the original sentence. Sadly, the adjustment does not work well. As presented in Table 4.16, models trained with MLM perform worse than the original SimCSE, and the unnatural replacement issue is still not resolved.

If the poor result is not due to incompatible data formats, then it might be a sign of over-fitting. To eliminate the possibility, we apply gradient accumulation. We set the gra-

Dataset: MR				
	SimCSE no MLM	SimCSE MLM weight=0.02	SimCSE MLM weight=0.1	SimCSE MLM weight=1
Number of successful attacks	**411**	379	410	385
Number of failed attacks	**427**	459	428	453
Number of skipped attacks	162	162	162	162
Original accuracy	83.8%	83.8%	83.8%	83.8%
Accuracy under attack	**42.7%**	45.9%	42.8%	45.3%
Attack success rate	**49.05%**	45.23%	48.93%	45.94%
Average perturbed word %	14.85%	14.47%	**14.0%**	14.23%
Average number of words per input	18.64	18.64	18.64	18.64
Average number of queries	54.93	53.78	54.49	52.9

Table 4.16: Evaluation of SimCSE with MLM of different weights after modification on MR dataset.

dient accumulation to 100. That means instead of updating the model after every batch, we accumulate the gradients and update after 100 batches. However, as we can see in Table 4.17, this method does not help solve the problems.

We use two separate datasets when we run the iterative training method in Section 4.4.4, namely the 125,000 contrastive sentence pairs we created using CLINE on the IMDb training dataset, and the 50,000 unlabeled IMDb dataset. Yet since we merged contrastive learning and pretraining by adding the MLM function, we have been using only one dataset, which is the 125,000 contrastive sentence pairs dataset. This might be an issue because it implies we use less training data. Therefore, we once again adjust the training script to have both components read their corresponding datasets. We also lower the gradient accumulation to 10 to see if it makes a difference. Demonstrated in Table 4.18, it turns out the adjustment makes no difference.

At this point, we have removed all possible factors that could affect MLM. We are left with one possible explanation of why it is still not working as expected, which is this auxiliary MLM function provided by SimCSE is not exactly equivalent to the pretraining script we use from the official BERT Github repository. Despite looking similar from the code, the

Dataset: MR		
	SimCSE no MLM	SimCSE MLM weight=0.1 Gradient Accumulation=100
Number of successful attacks	**411**	391
Number of failed attacks	**427**	447
Number of skipped attacks	162	162
Original accuracy	83.8%	83.8%
Accuracy under attack	**42.7%**	44.7%
Attack success rate	**49.05%**	46.66%
Average perturbed word %	14.85%	**14.52%**
Average number of words per input	18.64	18.64
Average number of queries	54.93	59.7

Table 4.17: Evaluation of SimCSE with MLM and gradient accumulation on MR dataset.

Dataset: MR		
	SimCSE no MLM	SimCSE MLM weight=0.1 Gradient Accumulation=10
Number of successful attacks	**411**	392
Number of failed attacks	**427**	446
Number of skipped attacks	162	162
Original accuracy	83.8%	83.8%
Accuracy under attack	**42.7%**	44.6%
Attack success rate	**49.05%**	46.78%
Average perturbed word %	14.85%	**14.82%**
Average number of words per input	18.64	18.64
Average number of queries	54.93	61.21

Table 4.18: Evaluation of SimCSE with MLM and gradient accumulation after modification on MR dataset.

Dataset: MR		
	BERT	BERT with MLM
Number of successful attacks	**473**	409
Number of failed attacks	**365**	429
Number of skipped attacks	162	162
Original accuracy	83.8%	83.8%
Accuracy under attack	**36.5%**	42.9%
Attack success rate	**56.44%**	48.81%
Average perturbed word %	**13.91%**	14.0%
Average number of words per input	18.64	18.64
Average number of queries	63.49	51.64

Table 4.19: Evaluation of BERT with MLM versus the original BERT on MR dataset.

two may have difference underlying logic that we are unaware of. To test this conjecture, we now train an original BERT with only the MLM portion on 50,000 unlabeled IMDb data. If MLM and pretraining are equivalent, this should have the same effect as pretraining for 2,500 steps. Surprisingly, we find out that adding the MLM function to an original BERT actually worsens its attack performance.

We make a conclusion that this auxiliary MLM function is not an ideal solution, and we have to find other ways to merge contrastive learning and pretraining into one loss. In the meantime, our best model remains to be 32-cycles iterative model.

Chapter 5

Conclusion and Future Work

In this project, we propose a new iterative training method that combines contrastive learning and pretraining to generate high quality adversarial examples. We discover that pretraining and contrastive learning have positive effects on generating high quality adversarial examples. We started by using a second-phase pretraining to make our attack model domain-specific, which has shown to be effective. Then we went one step further and added in contrastive learning to alter the embedding space, which improves the results even more. After that, we made use of CLINE to do data augmentation. Finally, we applied the iterative training method to maximize our model's efficacy, which boosted the performance. We also explored the possibility of merging contrastive learning and pretraining together into one loss, but the results are not satisfying. In the future, we will continue to work on the paper submission with Jianping Zhang.

Bibliography

Szegedy, Christian et al. (2013). "Intriguing properties of neural networks". In: *arXiv preprint arXiv:1312.6199*.

Goodfellow, Ian J, Jonathon Shlens, and Christian Szegedy (2014). "Explaining and harnessing adversarial examples". In: *arXiv preprint arXiv:1412.6572*.

Roth, Tom et al. (2021). "Token-modification adversarial attacks for natural language processing: A survey". In: *arXiv preprint arXiv:2103.00676*.

Gao, Ji et al. (2018). "Black-box generation of adversarial text sequences to evade deep learning classifiers". In: *2018 IEEE Security and Privacy Workshops (SPW)*. IEEE, pp. 50–56.

Li, Jinfeng et al. (2018). "Textbugger: Generating adversarial text against real-world applications". In: *arXiv preprint arXiv:1812.05271*.

Jin, Di et al. (2020). "Is bert really robust? a strong baseline for natural language attack on text classification and entailment". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 34. 05, pp. 8018–8025.

Devlin, Jacob et al. (2018). "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805*.

Liu, Yinhan et al. (2019). "Roberta: A robustly optimized bert pretraining approach". In: *arXiv preprint arXiv:1907.11692*.

Garg, Siddhant and Goutham Ramakrishnan (2020). "Bae: Bert-based adversarial examples for text classification". In: *arXiv preprint arXiv:2004.01970*.

Li, Linyang et al. (2020). "Bert-attack: Adversarial attack against bert using bert". In: *arXiv preprint arXiv:2004.09984*.

Gururangan, Suchin et al. (2020). "Don't stop pretraining: adapt language models to domains and tasks". In: *arXiv preprint arXiv:2004.10964*.

Lee, Jinhyuk et al. (2020). "BioBERT: a pre-trained biomedical language representation model for biomedical text mining". In: *Bioinformatics* 36.4, pp. 1234–1240.

Gao, Tianyu, Xingcheng Yao, and Danqi Chen (2021). "SimCSE: Simple Contrastive Learning of Sentence Embeddings". In: *arXiv preprint arXiv:2104.08821*.

Wang, Dong et al. (2021). "Cline: Contrastive learning with semantic negative examples for natural language understanding". In: *arXiv preprint arXiv:2107.00440*.

Peters, Matthew E. et al. (2018). *Deep contextualized word representations*. arXiv: 1802.05365 [cs.CL].

Mikolov, Tomas et al. (2013). *Efficient Estimation of Word Representations in Vector Space*. arXiv: 1301.3781 [cs.CL].

Pennington, Jeffrey, Richard Socher, and Christopher D Manning (2014). "Glove: Global vectors for word representation". In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543.

Alsentzer, Emily et al. (2019). "Publicly available clinical BERT embeddings". In: *arXiv preprint arXiv:1904.03323*.

Morris, John X et al. (2020). "Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp". In: *arXiv preprint arXiv:2005.05909*.

Ren, Shuhuai et al. (2019). "Generating natural language adversarial examples through probability weighted word saliency". In: *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 1085–1097.

Maas, Andrew L. et al. (June 2011). "Learning Word Vectors for Sentiment Analysis". In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Compu-

tational Linguistics, pp. 142–150. URL: http://www.aclweb.org/anthology/P11-1015.

Pang, Bo and Lillian Lee (2005). "Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales". In: *Proceedings of the ACL*.

Wolf, Thomas et al. (Oct. 2020). "Transformers: State-of-the-Art Natural Language Processing". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, pp. 38–45. URL: https://www.aclweb.org/anthology/2020.emnlp-demos.6.