Text perturbation Attack and Defense with Synonym replacement (TADS)

Aishwarya Anegundi
s8aianeg@stud.uni-saarland.de

Janaki Viswanathan
s8javisw@stud.uni-saarland.de

Lakshmi Rajendram Bashyam
s8laraje@stud.uni-saarland.de

Abstract

Adversarial attacks on images have gained a lot of momentum in the last few years. However, it has been difficult to apply the same gradient-based methods on discrete text data. A lot of text classification models are available for public or on demand. An interesting question to ask is if we can attack a black box text classification model. We also explore how we can defend such an attack in this work. We propose identification and ranking the important words which influence the output and replace them with synonyms with a constraint on sentence embedding to preserve the semantic consistency. We explore two methods of replacing words: (i) synonym replacement using WordNet; (ii) word replacement using GloVe. For the defense mechanism, we tried two text classifiers: (i) LSTM classifier; (ii) BERT-based classifier.

1. Introduction

Deep learning methods have produced state-of-the-art results. Due to its success, these models are provided as a service to the users using cloud based services. Though these are provided as a black-box model without sharing any information on the model or its architecture with the end users, they are vulnerable to many attacks. Many of these methods have been explored in the image domain and have been successful.

One of the famous methods to perturb the input data to be misclassified by the model was introduced by Goodfellow et al.,[5] which uses gradient-based method to carefully make a small perturbation on the input image to be misclassified by the model with high confidence.

However, transferring this method to text data poses many challenges. One of the challenges is due to its discrete nature, the gradient-based methods used in images cannot be used here. Another is to preserve the semantic consistency of the original text. A careful small perturbation on image data will look the same as the original image for a human eye. However, while perturbing a text input, we need to consider a few key factors. The perturbed text should be - (i) grammatically correct and preserve the context; (ii) semantically consistent with the original text while misleading the classification model.

With the recent advancements on pre-trained models trained on large datasets, the text domain has developed rapidly making it accessible for everyone. We make use of Bidirectional Encoder Representations from Transformers (BERT) - a pre-trained model by Google trained on unlabeled data extracted from the BooksCorpus with 800M words and English Wikipedia with 2,500M words [3]. Another pre-trained model that we use is Global Vectors (GloVe) embeddings obtained from an unsupervised learning algorithm on aggregated global word-vector co-occurrence statistics from a crawled corpus [1]. We also use WordNet [2] which is a lexical database of semantic relations between words in more than 200 languages [4].

In this work, we attack a text classification model by the following steps: identify and rank the important words by taking a difference between the output probabilities with and without the word, replace them using WordNet or GloVe embeddings with constraints on BERT sentence embedding to preserve the semantic consistency.

For the defense mechanism, we tried two binary text classifiers - one LSTM classifier and another BERT-based classifier which takes the original text and perturbed text as input and outputs whether or not the text has been perturbed.

2. Background

Many heuristics have been used to attack text based models. One such is to modify characters of a word [6]. Ren et al., used probability weighted word saliency to identify important words[9]. Li et al., used word embeddings to perturb the input text data[7]. Li et al., used the state-of-the-art word embedding model BERT to attack and also defend text based tasks[8].

3. Text perturbation Attack

Attacking the test classification model to flip it's decision involves two steps: (i) Identification and ranking of the important words in the input text (ii) Replacement of important words with synonyms with a constraint on sentence embedding. The block diagram of the method is as shown in figure 1

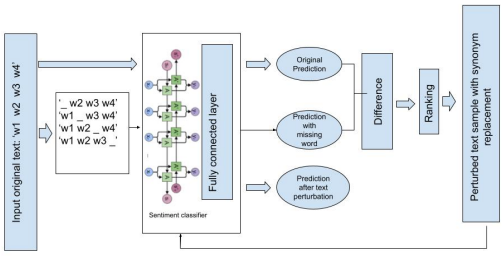

Figure 1. Block diagram of the attack method

3.1. Identification and ranking of the important words in the input text

Given the input text, at first, the probability of text belonging to positive sentiment class is obtained. Now, each word in the text is removed one at a time and corresponding probabilities are obtained. The difference between each of these probabilities with the probability of the original text quantifies the importance of the specific word removed from the text. In this manner, every word in the text is scored based on the probability difference with the original text. Now, the words are ranked according to these scores. The topmost word on the ranked list is the word which has the highest influence on the model's decision. An example of important word identification and ranking is as below:

Example sentence: 'I wish I knew what to make of a movie like this. It seems to be divided into two parts – action sequences and personal dramas ashore. It follows Ashton Kutsher ... we don't learn much about that. We don't really learn much about anything. The film devolves into a succession of visual displays and not too much else. A disappointment.'

Original Output probability: 0.0899
Probability output without 'wish' : 0.0796
Probability output without 'knew' : 0.0831
Probability output without 'learn' : 0.0633
Probability output without 'disappointment' : 0.3023

Word importance ranking:
('disappointment', 0.302), ('elements', 0.0557), ('displays', 0.0315), ('learn', 0.0265), ('succession', 0.0225) ...

3.2. Replacement of important words with synonyms with synonyms with a constraint on sentence embedding

The number of important words replaced is controlled based on the value of hyper-parameter alpha. If alpha = 0.1 then 10% of the words are replaced if alpha = 0.2 then 20% of the words are replaced and so on. To obtain possible synonyms of the important words, two different methods have been used: (i) Glove Embedding and (ii) Wordnet synset with POS restriction.

3.2.1 Word replacement with Glove embedding

Each of the important words will be replaced with their similar word based on their Glove embedding. If the word to be replaced is present in the vocabulary of the Glove embedding, it is replaced with the closest word in the embedding space. A few examples for the synonyms obtained using Glove embedding is as shown in Table 1:

Original word	Synonym
divided	split
succession	dynastic
disappointment	frustration
learn	understand
film	movie

Table 1. Examples of synonyms obtained from Glove embedding

3.2.2 Word replacement with Wordnet synset

In this method, for every important word, synonyms are found using Wordnet synset. While identifying the synonym of a given word, the POS of the words is also considered. It is made sure that the POS tag of the original word and the synonym is the same. For any given word many synonyms could replace the word. An example for replacement using Wordnet synset is as shown in table 2.

Word	POS	Synonyms
cheap	ADJ	brassy, crummy, chintzy, garish, sleazy, tatty, gimcrack, tinny, flash, bum, flashy, punk, meretricious, cheesy, tawdry, chinchy, gaudy, loud, trashy, inexpensive, cheap, tacky
film	NOUN	cinema, motion picture, motion-picture show, moving-picture show, moving picture, pic, flick, plastic film, movie, photographic film, celluloid, picture show, picture

Table 2. Example of word replacement using Wordnet synset

3.3. Constraint on sentence embedding

The challenge at hand is to replace the word with the synonym that keeps the perturbed text semantically close to the original text. The perturbed text should not alter the meaning of the original text very much. This is achieved by conditioning the synonym replacement on sentence embedding. At first the synonyms of all the important words are obtained. The decision whether the original word will be replaced with synonym is decided based on sentence embedding constraint. When a word is replaced with its synonym, if the cosine similarity between the sentence embedding of the newly formed sentence and that of the original sentence is greater than a certain threshold then only the synonym is retained else the original word is retained. An example is as shown in table 3

Original text	Perturbed text that does not pass sentence embedding constraint	Perturbed text that passes the sentence embedding constraint
A touching movie. It is full of emotions and **amazing** acting. I could have sat through it a second time.	A touching movie. It is full of emotions and **howling** acting. I could have sat through it a second time.	A touching movie. It is full of emotions and **marvellous** acting. I could have sat through it a second time.

Table 3. Example of word replacement with constraint on sentence embedding

4. Text perturbation Defense

A text perturbation dataset is created from text perturbation method as discussed in section 3. The dataset consists of 80k samples. There is an equal number of samples in both original text and perturbed text categories. To add on to that, there is an equal number of samples belonging to positive sentiment and negative sentiment classes. More information is given in the table 4.

	Positive review	Negative review
Original text	20k	20k
Perturbed text	20k	20k

Table 4. Distribution of Text perturbation dataset samples

Using the Text perturbation dataset two different classification model were trained to classify the given inputs as original text or perturbed text. One of the models was an LSTM text classifier and another model was a BERT based text classifier.

5. Experiments and Results

We test our attack method on a text classification task more precisely on a sentiment analysis task. We use the IMDB movie review dataset consisting of 50k training samples and 50k test samples. All the code was written in python using Pytorch, and can be found here.

5.1. Base model training and evaluation

The base model is a simple bidirectional LSTM classifier trained for 5 epoch with a batch size of 32 and dropout probability of 0.2 using Adam optimizer. The resulting accuracy and loss curves are as shown in figure 2.

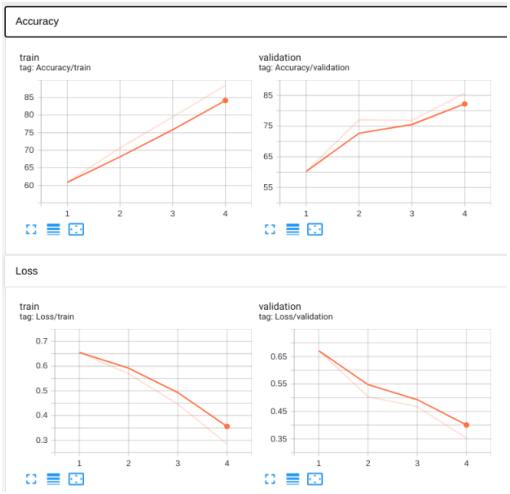

Figure 2. Accuracy and loss curves for training and validation of the base LSTM classifier

5.2. Evaluation of the Attack method

To evaluate the attack method we generated the perturbed samples for 1k randomly selected samples from the IMDB test set. The perturbed samples were generated using both synonym replacement methods: Glove embedding and Wordnet synset. The perturbed samples were generated for different values of alpha. The baseline model was then evaluated on the generated perturbed samples. The degradation of model accuracy with increasing alpha value can be observed in the figure 3

To evaluate how intact the generated perturbed texts are to the original texts semantically, we could not have used cosine similarity on the text embeddings because the word replacement was done by conditioning on the sentence embedding. Hence we resorted to the human evaluation of semantic similarity of perturbed and original samples. During this evaluation, we observed the following: As previously

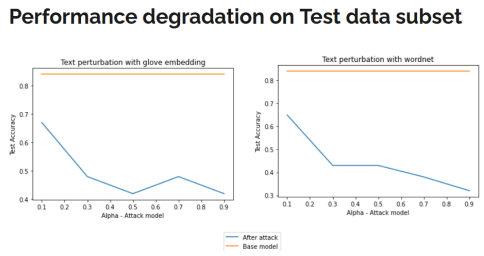

Figure 3. Decrease in base model test accuracy with increasing alpha values

illustrated in the table 3, synonym replacement without a constraint on sentence embedding flipped the classifier decision easily but the generated samples could easily be classified as a perturbed sample as well as the exact words that were manipulated could also be recognized easily. This was improved by placing the constraint on sentence embedding wherein a few sentences it was difficult to differentiate the sentences as perturbed. To further improve the results, the word replacement was done by conditioning on POS tags.

5.3. Evaluation of defense method

The first model used for classification was a simple bidirectional LSTM classifier. The second model used was fine tuned bert-base-uncased model. Both the models were trained for 5 epochs with batch size 32 using Adam optimizer. The results are as shown in table 5.

Model	Test Accuracy
LSTM	96.76%
bert-base-uncased	97.14%

Table 5. Test accuracy of the defense method

6. Conclusion

As part of this project, we implemented a black-box method to attack a text classifier model by perturbing the input text. The text perturbation was done by replacing the important words by their synonyms. The strategy for replacement included conditioning on sentence embedding and POS tags which reduced the semantic loss in the perturbed text. It was observed that the model can be attacked by a few replacements which show how the model's decision is dependant on a few words only and is vulnerable to such perturbations. A defense method was implemented to classify the input text as perturbed or not using the dataset created from attack method.

References

[1] Glove: Global vectors for word representation. https://nlp.stanford.edu/projects/glove/.

[2] What is wordnet? https://wordnet.princeton.edu/.

[3] Wiki: Bert (language model). https://en.wikipedia.org/wiki/BERT_(language_model).

[4] Wiki: Wordnet. https://en.wikipedia.org/wiki/WordNet.

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

[6] Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. Is bert really robust? a strong baseline for natural language attack on text classification and entailment, 2020.

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

[8] Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. Bert-attack: Adversarial attack against bert using bert. *arXiv preprint arXiv:2004.09984*, 2020.

[9] Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. Generating natural language adversarial examples through probability weighted word saliency. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pages 1085–1097, 2019.