ECAI 2023
K. Gal et al. (Eds.)
© 2023 The Authors.
This article is published online with Open Access by IOS Press and distributed under the terms of the Creative Commons Attribution Non-Commercial License 4.0 (CC BY-NC 4.0).
doi:10.3233/FAIA230376

Preserving Semantics in Textual Adversarial Attacks

David Herel [a, b;*], **Hugo Cisneros** [b] **and Tomas Mikolov** [b]

[a]Faculty of Electrical Engineering, Czech Technical University in Prague
[b]Czech Institute of Informatics, Robotics and Cybernetics, Czech Technical University in Prague
ORCiD ID: David Herel https://orcid.org/0009-0000-1861-3778, Hugo Cisneros https://orcid.org/0000-0003-3439-4565, Tomas Mikolov https://orcid.org/0000-0002-6938-5426

Abstract. The growth of hateful online content, or hate speech, has been associated with a global increase in violent crimes against minorities [23]. Harmful online content can be produced easily, automatically and anonymously. Even though, some form of auto-detection is already achieved through text classifiers in NLP, they can be fooled by adversarial attacks. To strengthen existing systems and stay ahead of attackers, we need better adversarial attacks. In this paper, we show that up to 70% of adversarial examples generated by adversarial attacks should be discarded because they do not preserve semantics. We address this core weakness and propose a new, fully supervised sentence embedding technique called Semantics-Preserving-Encoder (SPE). Our method outperforms existing sentence encoders used in adversarial attacks by achieving $1.2\times \sim 5.1\times$ better real attack success rate. We release our code as a plugin that can be used in any existing adversarial attack to improve its quality and speed up its execution[1].

1 Introduction

With the massive growth of online content, the need to detect harmful content, such as hate speech, offensive language, or fake news, is increasingly important. Current defense system relies on deep learning models in NLP. However, many researchers have highlighted that language models are not as robust as previously thought [13, 17, 21, 22, 35, 41] and that they can be fooled quite easily with so-called adversarial examples. Adversarial examples introduce a small perturbation to the input data that is 'imperceptible' to the human eye and fools the system into misclassifying the perturbed data. For example, in the domain of offensive language detection, we can have an offensive text on input and modify it in such a way that the meaning is preserved, but the modified text will fool the system to classify it as non-offensive [17]. A similar scenario is illustrated in Figure 1 in the domain of sentiment analysis of movie reviews.

It is important to mention that the imperceptibility of adversarial attacks in discrete domains such as text is much more difficult to define than in a continuous space, since in discrete domains an indistinguishable modification simply cannot exist. According to [17, 29] we can identify three main requirements for an adversarial attack on text to be successful: 1. Human prediction consistency: The prediction made by humans should remain unchanged; 2. Semantic similarity: The designed example should have the same meaning as the source as judged by humans; 3. Language fluency: the generated examples should look natural and grammatical.

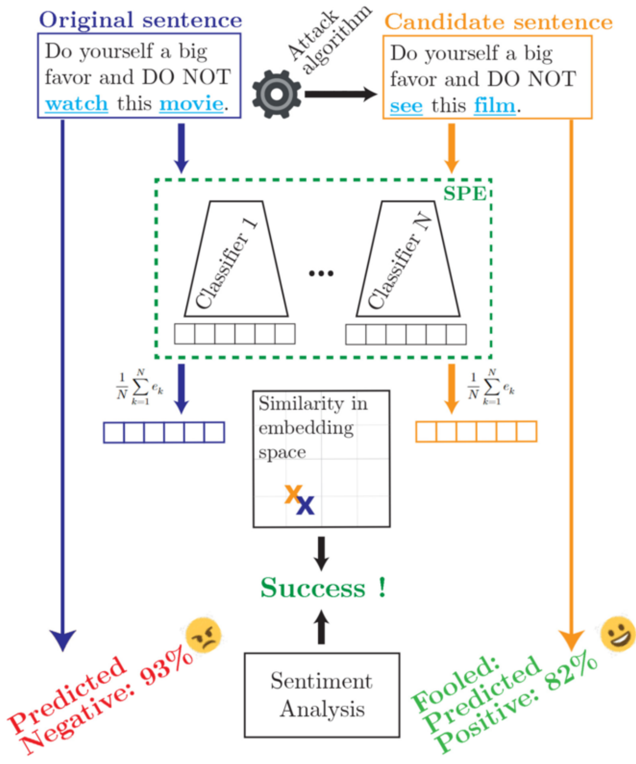

Original sentence
Do yourself a big favor and DO NOT watch this movie.
Attack algorithm
Candidate sentence
Do yourself a big favor and DO NOT see this film.
SPE
Classifier 1
...
Classifier N
$\frac{1}{N}\sum_{k=1}^{N} e_k$
Similarity in embedding space
Success !
Sentiment Analysis
Predicted Negative: 93%
Fooled: Predicted Positive: 82%

Figure 1: A high-level overview of textual adversarial attack and our proposed Semantic-Preserving-Encoder (SPE). Our method takes advantage of supervised learning to tackle the problem of preserving semantics.

Although most adversarial attacks claim to meet these constraints [11, 12, 24], careful scrutiny shows otherwise. We observe that many adversarial examples do not preserve the meaning of the text. This is also supported by Morris et al. [29] whose findings are similar. To overcome this problem, Morris et al. [29] suggested increasing cosine thresholds and introducing mechanisms such as grammar checks

* Corresponding Author. Email: hereldav@fel.cvut.cz

[1] The code, datasets and test examples are available at https://github.com/DavidHerel/semantics-preserving-encoder.

to improve the quality of adversarial examples. However, it is at the cost of the attack success rate, which decreased rapidly by more than 70%. We suggest a different solution that promises to avoid this decline in the attack success rate.

It appears that the problem lies in the similarity metric itself, whose function is to measure the difference between the original and perturbed sentences. These metrics mostly use encoders that are trained with limited supervision. This makes them more susceptible to problems with antonym recognition, as illustrated in the third column of Table 3.

We propose a new sentence encoder for similarity metrics in textual adversarial attacks called Semantics-Preserving-Encoder (SPE). Our method takes advantage of supervised learning on annotated datasets. Thus, it is more robust towards the antonym recognition problems that are observed in adversarial examples. Experimental results show that our proposed method outperforms existing sentence encoders used in adversarial attacks by achieving $1.2\times \sim 5.1\times$ better real attack success rate, while also being significantly faster. Furthermore, our solution – SPE is publicly available on GitHub[2] and can be integrated as a component into any existing adversarial attack granting much faster execution.

The contributions of this work can be summarized as follows:

1. We propose a new sentence-encoder technique, SPE, for textual adversarial attacks. Our method outperforms existing sentence encoders used in adversarial attacks, achieving $1.2\times \sim 5.1\times$ better real attack success rate.
2. We propose a new metric for evaluating the quality of the adversarial examples – rASR, which reflects the real performance of adversarial attacks.
3. We evaluate some of the most common sentence encoders used in adversarial attacks both manually and automatically on relevant datasets such as hate-speech and offensive language detection [3], in addition to other popular classification tasks such as Yelp Reviews [40] and Rotten Tomatoes [31].
4. We release our work as open source, including the code, human evaluations, datasets, and test samples for the purpose of reproducibility and future benchmarking.

2 Related work

Textual adversarial attacks. There were many attempts to create textual adversarial attacks that preserve semantics and are grammatically correct. Ultimately, we can distinguish between three different ways in which other researchers approach this. The first approach aims to modify the whole sentence using various sophisticated phrase perturbations, such as paraphrasing [10, 37]. However, these modifications often have problems with semantic preservation [39].

The second approach focuses on character-level modification, such as misspellings or typos, which has proven to be more successful in terms of semantics preservation [9, 14, 26]. However, research shows that these types of attacks can be mitigated quite easily with tools such as grammar checks [18, 32].

Lastly, the word-level attack technique focuses on the substitution or modification of a single word (or combination of multiple words) in the text [1, 8, 25, 33]. This type of attack aims to preserve the constraints defined by [30], often using methods such as synonym substitution to improve semantic preservation.

[2] The code, datasets and test examples are available at https://github.com/DavidHerel/semantics-preserving-encoder.

Similarity metrics. The majority of word-level adversarial attacks enforce semantics similarity by using Universal-Sentence-Encoder (USE) [5] or BERTScore [38] encoders, both of which are trained mainly on unsupervised tasks. USE is trained on a task such as Skip-Thought [20], a conversational input-response task [15] and a supervised classification task performed on the SNLI dataset [4]. BERTScore is based on a pre-trained BERT language model [6], which was also trained unsupervisedly on the Next Sentence Prediction and Masked LM tasks.

Even though these sentence encoders have been thoroughly studied on various general tasks, only a few previous works acknowledge their flaws when used in adversarial attacks [16, 30]. In most cases, these encoders struggle to recognize changes in the text's meaning and semantics. To overcome this, Morris et al. [30] increased the cosine threshold, resulting in an improved quality of adversarial examples, but over 70% decline in the attack success rate.

Looking at the big picture of the observed problems, we have discovered a potential link to the encoder training process. We show that the unsupervised training predetermines these encoders to have problems with antonym recognition, which leads us to introduce our own Semantics-Preserving-Encoder.

3 Method

Problems of textual adversarial attacks are formulated in the following section. Next, SPE is formally introduced together with its classifiers and other properties.

3.1 Problem formulation

Most similarity metrics in adversarial attacks rely on sentence encoders such as USE [5] or BERTScore [38]. These encoders use multi-task learning with a high emphasis on unsupervised tasks. The BERT language model [6] uses Masked-Language Modeling. USE is trained on Skip-Thought [20] like task, where the goal is to predict the middle sentence based on the given context. Both of these training methods could be seen as a variation of a Skip-Gram/CBOW model [28], where the goal is to predict the context of a given target word, and vice versa for CBOW. However, this training method forces synonyms to be mapped into a similar vector space as antonyms, since they appear in the same context. Therefore, two contradictory sentences in their vector representation could be very close to each other in the vector space despite their opposite meaning. That is if the unsupervised training with Skip-Gram or CBOW like tasks is used. This is the case for both BERT and USE which is shown in Table 3.

3.2 Semantics-Preserving-Encoder

The core idea of our encoder lies in supervised training. We took advantage of existing prelabeled datasets and utilized them in the training data to tackle the problem with opposite words appearing in the same context. As a result, the words that are the most discriminative for the given label will be close to each other in the vector space. Thus, sentences like *'This movie is so good'* and *'This movie is so bad'* should never be close to each other in the vector space, because their semantics label will be exactly opposite.

We have combined N classifiers trained on different annotated datasets, allowing us to have a diverse set of different sentence vectors. The sentence vectors will differ because each classifier produces its vector according to the task on which it was trained. Therefore, the diversity of classifiers implies a diverse set of sentence vectors. Moreover, by combining several sentence embeddings from different classifiers, we can create a robust classifier that can produce a high-quality embedding for a broad range of topics.

From a sentence S, an attack will generate a candidate adversarial example S^*. We denote the N supervised classifiers by C_1, C_2, $\ldots$, C_N. For simplicity, we consider these classifiers to be functions whose outputs are the sentence embeddings extracted from the classifier when applied to a sentence. Formally, for all k, $C_k(S) = \boldsymbol{e}_k \in \mathbb{R}^p$ where p is the embedding dimension. The complete embedding of a sentence is obtained by averaging the output of the classifiers into a single embedding vector. Average helps to reduce noise and improve the overall accuracy of a model by smoothing out the influence of any individual data points that may be outliers or contain errors.

$$\boldsymbol{e}^{(S)} = \frac{1}{N}\sum_{k=1}^{N} C_k(s) = \frac{1}{N}\sum_{k=1}^{N} \boldsymbol{e}_k^{(S)}.$$

The similarity between the original sentence S and the attacked sentence S^* is computed with the cosine similarity between their embeddings as follows

$$\text{Sim}(S, S^*) = \frac{\boldsymbol{e}^{(S)} \cdot \boldsymbol{e}^{(S^*)}}{\|\boldsymbol{e}^{(S)}\| \|\boldsymbol{e}^{(S^*)}\|},$$

where $\cdot$ represents the dot product between vectors in $\mathbb{R}^p$ and $\|\cdot\|$ is the L_2 norm in $\mathbb{R}^p$. For a threshold ϵ, S and S^* will be considered to have the same meaning if $\text{Sim}(S, S^*) > \epsilon$.

The classification model that we used is fastText [19]. However, it is important to note that any other classification model can be used instead. We decided to use fastText due to its many advantages. Firstly, fastText classifiers allow us to create sentence vectors rather quickly with a reasonable performance for the given task. Secondly, we can reduce the dimensionality of the vector space. This way we can put more information into fewer dimensions, which results in a more efficient space storage.

Our method is highly flexible and can be adapted to any task depending on the selection of supervised classifiers. Embedding from each classifier is averaged into one, which helps to reduce noise and improve the overall accuracy of a model by smoothing out the influence of any individual data points that may be outliers or contain errors. This process is illustrated in Figure 2.

As mentioned above, the use of fastText classifiers allows us to achieve a reduced real-time complexity of SPE compared to USE [5] or BERTScore [38]. SPE only needs to perform N matrix multiplication operations (because we implement N fastText classifiers) with small matrices due to the classifiers hidden layer dimension equal to H, which is usually a small number e.g. 10. This is a marginal difference compared to the computations executed by USE [5], which performs the operations with H-dimensional vectors, where H equals 512 and therefore very large matrices.

The final time complexity to compute the sentence representation using SPE is defined as follows:

$$O = \sum_{i=1}^{N} V * H_i \tag{1}$$

where V is the length of the sentence and H_i is the size of the hidden layer for the i-th classifier.

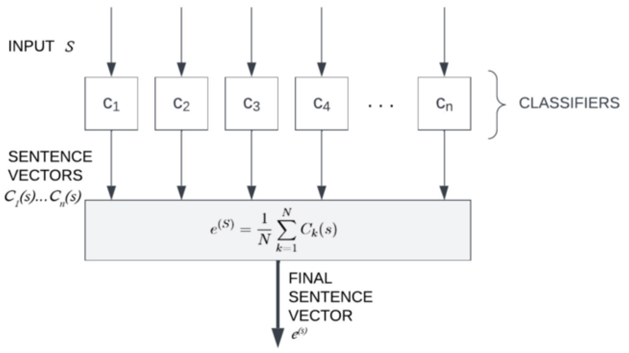


Figure 2: Model architecture for SPE: Given input S and N supervised classifiers denoted as $C_1, C_2, \ldots, C_N$. We consider these classifiers to be functions whose outputs are the sentence embeddings extracted from the classifiers when applied to a sentence. The complete embedding of a sentence is obtained by averaging the output of classifiers into a single embedding vector $e^{(S)}$.

4 Experiments

To evaluate our proposed sentence encoder in real attack use cases, we performed automatic and human-based manual evaluation. As explained in Introduction, the commonly used attack success rate is fundamentally flawed and insufficient to measure the success of an adversarial attack.

The metric only gives a partial view of the real capability of an attack, obscuring the quality of the semantic similarity constraint on the generated sentences. For this reason, we conducted an extensive survey to collect human evaluations of the quality of the attacks generated, which allows us to evaluate the semantic similarity constraint.

The main goal of our experiments is to study the impact of SPE when used as a semantic similarity constraint in adversarial attacks. If this constraint is too strict, only a few high-quality sentences would be accepted as successful attacks, potentially missing many good candidates. Inversely, too loose a constraint would produce many low-quality examples. We focus our experiments on two widely used attacks: TextFooler [17] and its improved version TFAdjusted [30]. Due to the extensive human evaluation, it would be difficult to experiment with all the other adversarial attacks. However, results from these adversarial attacks should be applicable on other adversarial attacks due to their significant similarities.

To understand the effect of SPE in existing attacks, we use it as a constraint, together with two other state-of-the-art semantic similarity metrics, the Universal Sentence Encoder (USE) [5] and BERTScore [38], totaling six attack/sentence encoder pairs.

4.1 Automatic Evaluation

We automatically evaluate adversarial attacks using three metrics:

Attack success rate (ASR). The percentage of successful adversarial examples found by an attack. Given a sentence classifier, a successful adversarial example means that the attack generated a sentence similar to the original that is assigned a different label by this classifier. A higher success rate means that more generated sentences are assigned a different label.

Time. Average time needed to create an adversarial example. Attacks may rely on various search techniques to generate candidate sentences, leading to varied generation times.

Modification rate. The percentage of modified words. To be as imperceptible as possible, attacks should use as few edits as possible.

4.2 *Human evaluation*

Using automatic evaluation, such as the attack success rate, adversarial attacks are considered successful if they simultaneously pass a semantic similarity threshold and manage to change the label of a classifier. Measurement of semantic similarity between sentences is still an open research problem with no commonly accepted solution. In some cases, even humans disagree on whether two sentences are similar or not. For this reason, we include an extra evaluation step in our experiments to obtain more robust results. Sentence pairs were presented to 5 annotators who assigned them an integer score between 1 and 4, where the score 1 means strongly disagree, 2 disagree, 3 agree, and 4 strongly agree. The higher the score, the more similar the meaning of a pair of sentences will be according to the annotator. Each annotator was given the sentences to label in several online forms. We averaged the scores of all the annotators into a single value for each sentence. Sentences were assigned a binary label; those with an average score greater than or equal to 2.5 are considered similar, while the others are labeled as not similar. Because the original score system is not binary, the threshold was chosen exactly in the middle of these values. Experimentally, we have determined that the exclusion of the equality to 2.5 does not have any noticeable impact on the results.

Based on this evaluation, we estimate a realistic success rate of the attack — or an estimated “real ASR” (rASR). This number represents the number of successful attacks that would actually fool the reader into thinking that they have the same meaning as the original sentence, but have a different label assigned by the classifier. This score, although costly to estimate, gives a very accurate measure of the quality of an attack. A perfect attack would reach a high ASR and a very similar rASR (ASR $\approx$ rASR), which means that it successfully fooled the target classifier repeatedly and that all the corresponding attacked sentences also conserve the same meaning as the original ones in the eyes of human annotators.

4.3 *Datasets*

We evaluated the performance of SPE-based attacks and compared them with existing state-of-the-art attacks on four text classification datasets that correspond to various potential applications of adversarial attacks from NLP, such as data augmentation or detection of hateful and offensive speech. We did not include sentence similarity benchmarks such as MRPC [7] or other, because they do not contain the types of sentences that occur in adversarial examples. Therefore, they would not be relevant for our context.

Figure 3 shows the average number of words in each of the datasets studied. All source code, links to the datasets, and trained models can be found on GitHub.

Offensive tweets. This dataset is from the TweetEval set of seven multiclass tweet classification tasks [3] . The tasks are labeled irony, hate, offensive, stance, emoji, emotion, and sentiment. The offensive task consists of classifying tweets as offensive or non-offensive. We used a subset of 1000 sentences from the training split of the dataset for our attacks. This dataset is particularly interesting in the context of adversarial attacks on text because it is based on real harmful content that can be encountered online.

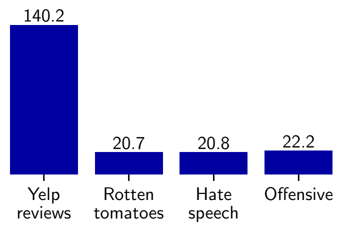


Figure 3: Average number of words per sentence in each dataset. The Yelp reviews dataset has much longer sentences on average. The two datasets based on Twitter have a built-in character limit (tweet size), and the Rotten tomatoes reviews were pre-processed by the dataset authors to usually contain a single sentence.

The attacked model is a roBERTa-based [27] model trained on 58M tweets and fine-tuned for offensive language identification with the TweetEval benchmark [3].

Hate speech tweets. This dataset is from the same set of tasks as the Offensive Tweets dataset above (Section 4.3). The goal of the hate speech task is to classify tweets as hate speech or non-hate speech. We used a subset of 1000 sentences from the training split of the dataset for our attack.

The attacked model is a roBERTa-based [27] model (same as the model used for the offensive tweets dataset) trained on 58M tweets and fine-tuned for offensive language identification with the TweetEval benchmark [3].

Rotten tomatoes. The rotten tomatoes dataset is a movie review dataset with 5331 positive and 5331 negative processed sentences from Rotten Tomatoes movie reviews [31]. We used a subset of 1000 sentences from the training split of the dataset for our attack.

The attacked model is a roBERTa [27] model fine-tuned on the rotten tomatoes dataset.

Yelp reviews. The Yelp review dataset is a dataset for binary sentiment classification [40]. It contains a set of 560,000 highly polar Yelp reviews for training and 38,000 for testing. It consists of Yelp reviews extracted from the 2015 Yelp Challenge data.

The attacked model is a roBERTa [27] model fine-tuned on binary sentiment classification from Yelp polarity.

4.4 *SPE parameters*

Choosing supervised classifiers that are suitable for a wide variety of tasks poses a challenging task. We have trained more than 40 classifiers on different annotated data and performed experiments on a Morris dataset [30], which contains 400 annotated adversarial sentence pairs. We have found that 7 selected classifiers achieve reasonably good performance. These classifiers were trained on Natural Language Inference (NLI) datasets like SNLI [4], cola, rte and sst2 [36], and on more downstream tasks such as emotion classification CARER, Yelp reviews and Stack Overflow questions classification [2, 34, 40]. More details about the selection of classifiers can be seen in Section 4.5.

Generally, the role of an encoder in a similarity metric is to transform the given text into a corresponding vector in the latent space, which is then used to evaluate the similarity. Specifically, cosine similarity is measured between the vectors of the original text and the perturbed text. If cosine similarity reaches a certain preset threshold ϵ, an adversarial example is considered successful.

Therefore, for our metric, the general cosine threshold also had to be defined. Ideally, it should be set so that consistent results are produced for any domain. Our experiments empirically show that setting ϵ to 0.95 performs consistently well across different datasets. These findings are also similar to thresholds find by Morris et al. [30].

4.5 Classifier Selection

Experiments for classifier selection were performed on the Morris dataset [30]. This dataset contains 400 sentences and their perturbed examples labeled by human annotators, to determine whether it preserves the meaning. Thus making it a valuable dataset for testing our method.

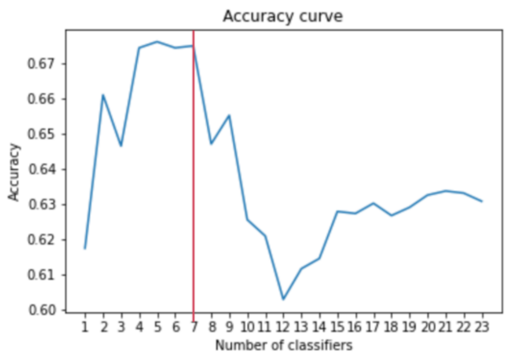


Figure 4: Visualization of how addition of other classifiers to the set decreases performance on the Morris dataset [30]. The reason behind this observation is that some classifiers have learned misleading or irrelevant mapping of words to latent space for our task.

We have experimentally tried many combinations of sets of classifiers. The best set we have found consists of 7 classifiers as shown in Figure 4. Addition of other classifiers to the set decreased the performance. The reason behind this observation is that some classifiers have learned misleading or irrelevant mapping of words to latent space for our task.

An example of such a classifier is one that is trained on a dataset of formal and slang English pairs and the goal is to distinguish between them. Then the classifier will learn to map antonyms and synonyms to a similar latent space and thus will not be able to distinguish between positive and negative sentences, which is what we aim to achieve. This exact case is observed behind the red line in Figure 4.

The final set of classifiers was trained on Natural Language Inference (NLI) datasets like SNLI [4], cola, rte and sst2 [36], and on more downstream tasks such as emotion classification CARER, Yelp reviews and Stack Overflow questions classification [2, 34, 40].

4.6 Human Participants

Due to the character of the task, human judges had to be advanced in English and understand phrases and shortcuts commonly used online, as well as some cultural references. That is why only native speakers or certified C2-level English speakers were recruited[4]. The recruitment process took place through a student research group. There were 5 annotators selected in total, who were paid at least a minimum wage, which was agreed upon prior to their involvement.

The human evaluation process took place through multiple online forms (Google Form) - one form for each dataset and attack, which contained an introductory page with the assignment instructions and then one sentence pair on each page (maximum 100 sentence pairs per one form in total) with a selection of scores. For each sentence pair, annotators assigned an integer score between 1 and 4, where the score 1 strongly disagree, 2 disagree, 3 agree, and 4 strongly agree. Instructions can be seen in Figure 5, and an example of a typical sentence pair is depicted in Figure 6.

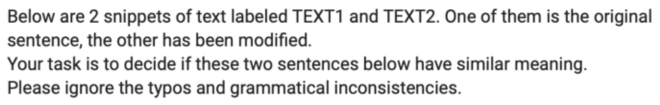
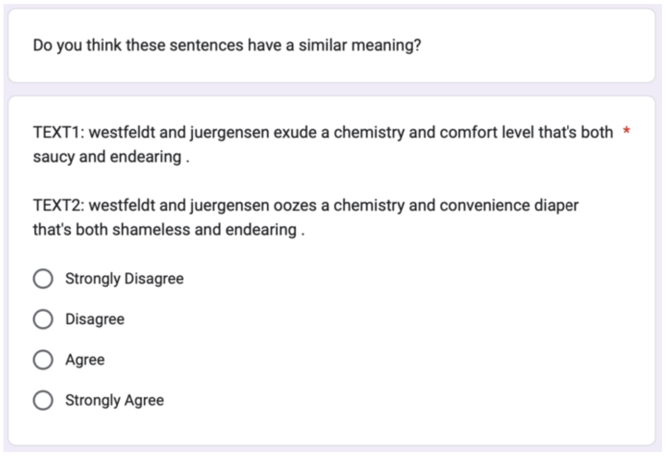

Below are 2 snippets of text labeled TEXT1 and TEXT2. One of them is the original sentence, the other has been modified.
Your task is to decide if these two sentences below have similar meaning.
Please ignore the typos and grammatical inconsistencies.

Figure 5: Assignment instructions for human judges.


Do you think these sentences have a similar meaning?

TEXT1: westfeldt and juergensen exude a chemistry and comfort level that's both saucy and endearing . *

TEXT2: westfeldt and juergensen oozes a chemistry and convenience diaper that's both shameless and endearing .

○ Strongly Disagree
○ Disagree
○ Agree
○ Strongly Agree

Figure 6: A typical page of the form for human judges consists of a sentence pair and selection boxes of available scores.

During the annotation process, annotators were allowed to use the Google Search Engine and an online English dictionary of their choice to clarify any cultural references or shortcuts they may not have been familiar with prior to this task. Other than the exposure of the annotators to offensive online content, which they agreed to undergo voluntarily prior to their participation, there were no risks to the participants. All annotators were thoroughly informed of the task at hand and gave their full consent to participate voluntarily.

Annotators had to meet the following criteria: be adults (due to the offensive nature of some of the content for evaluation), and have advanced English (certified as C2 or native speakers). We have asked each annotator to provide proof for each of the above-mentioned criteria. Besides that, no other data about the annotators were collected. All the information collected about human judges is listed in Table 1.

Annotator id	Age	English proficiency level
1	18+	C2
2	18+	native speaker
3	18+	native speaker
4	18+	C2
5	18+	C2

Table 1: Characteristics of human judges.

[4] According to the international standard - Common European Framework of Reference for Languages (CEFRL).

(a) **Offensive tweets**

Attack	ASR ↑	rASR ↑	Time ↓	Mod.rate
TextFooler+SPE (ours)	68.3	**30.7 (100)**	1.559	16.8
TextFooler+USE	75.4	20.4 (100)	1.749	17.4
TextFooler+BERTScore	64.2	18.6 (100)	1.955	19.5
TFAdjusted+SPE (ours)	11.7	**9.1 (100)**	1.054	10.6
TFAdjusted+USE	1.0	-	0.865	9.1
TFAdjusted+BERTScore	5.4	5.4 (47)	1.167	10.4

(b) **Hate speech tweets**

Attack	ASR ↑	rASR ↑	Time ↓	Mod.rate
TextFooler+SPE (ours)	65.6	**45.9 (100)**	1.865	19.3
TextFooler+USE	69.1	27.0 (100)	2.111	20.9
TextFooler+BERTScore	65.8	37.5 (100)	2.274	20.7
TFAdjusted+SPE (ours)	10.8	**9.8 (97)**	1.044	12.4
TFAdjusted+USE	2.6	2.4 (23)	0.771	9.7
TFAdjusted+BERTScore	5.7	5.7 (51)	1.187	14.3

(c) **Rotten tomatoes**

Attack	ASR ↑	rASR ↑	Time ↓	Mod.rate
TextFooler+SPE (ours)	89.0	**61.4 (100)**	0.746	12.8
TextFooler+USE	96.4	41.5 (100)	0.824	13.2
TextFooler+BERTScore	92.7	38.9 (100)	0.986	13.9
TFAdjusted+SPE (ours)	12.6	**11.8 (100)**	0.774	12.4
TFAdjusted+USE	0.3	-	0.513	9.6
TFAdjusted+BERTScore	5.7	5.1 (57)	0.993	14.2

(d) **Yelp reviews**

Attack	ASR ↑	rASR ↑	Time ↓	Mod.rate
TextFooler+SPE (ours)	87.4	**36.7 (100)**	14.792	10.2
TextFooler+USE	90.5	7.2 (100)	13.874	10.6
TextFooler+BERTScore	89.8	7.2 (100)	16.176	11.5
TFAdjusted+SPE (ours)	8.2	**4.5 (81)**	36.921	10.6
TFAdjusted+USE	0.3	-	22.325	1.2
TFAdjusted+BERTScore	2.3	1.5 (23)	32.946	5.7

Table 2: ASR is the attack success rate (in %), rASR is the estimated real attack success rate obtained from the human survey (in %), the time per sentence is in seconds and the modification rate is the average fraction of words changed per sentence (in %). An upward arrow (↑) indicates that higher is better. All numbers but the estimated rASR were computed on 1000 instances. For the estimated rASR, we report the number of annotated examples in parentheses. For some of the results with TFAdjusted + USE/BERTScore the ASR is so low that few sentences could be submitted for annotation, and the rASR estimates are unconclusive. We omitted the rASR values for configurations with less than 10 successful attacks out of 1000 attempts.

5 Results

The results of our experiments with are shown in Table 2. We observe that SPE surpasses other alternatives by a large margin in all configurations, with an estimated real attack success rate (rASR) more than 19% higher, on average, than USE or BERTScore with TextFooler and more than 5 percentage points, on average, higher with TFAdjusted.

As expected, TextFooler-based attacks have a high ASR, reaching 90% and more while attacking the rotten tomatoes dataset for example. But they also have a steep drop from ASR to rASR; for example, TextFooler+USE has its ASR decrease from 96% to a rASR of 41%, indicating that many of the supposedly successful attacks actually do not have the same meaning according to human annotators. This is due to the weakness of the semantic similarity constraints, as they do not filter out low-quality examples. This can be an issue in practical applications, since the attacked sentence could easily be detected by humans. For all four datasets, this drop is minimal when using SPE as a constraint.

TFAdjusted receives lower ASR scores than TextFooler (no ASR greater than 12%), but the drop between ASR and rASR is low compared to TextFooler, with a maximum drop of 2.4% for TFAdjusted+SPE on the offensive tweets dataset. It shows that the sentences generated by the TFAdjusted attack are of high quality, which is confirmed by the annotators. This is in agreement with the observations of Morris et al. [30] who proposed TFAdjusted to improve the output quality of the TextFooler attack. The quality increase is obtained at the expense of the ASR, which can be considerably lower. For example, TFAdjusted with USE obtains ASR scores lower than 3% on all datasets, making it barely usable as an attack, since it generates less than 25 usable sentences out of a 1000 attacked sentences. The ASR of SPE is higher than the other constraints, which can be interpreted as SPE being less strict. Yet, our encoder still has remarkably low drops from ASR to rASR, and surpasses both other solutions in all examples in terms of rASR. This indicates a better ability to generate successful attacks that maintain the original meaning of sentences.

Original sentence	**Perturbed sentence**	**Cosine similarity (USE)**	**Cosine similarity (SPE, ours)**
This movie is so good	*This movie is so* ***bad***	0.90	**0.63**
This movie is so good	*This movie is so* ***tasty***	0.89	**0.95**

Table 3: Visualization of a cosine similarity produced by USE [5] and our approach, SPE. First and second columns represent Original and Perturbed sentence. The third column shows cosine similarity produced by USE and the fourth column shows cosine similarity produced by SPE. Sentences in the first row should have low cosine similarity due to their opposite meaning, and sentences in the second row high cosine similarity due to the similar meaning.

Adversarial attacks using SPE are also much faster than USE or BERTScore, up to 40% on TextFooler as can be seen in Table 2. For TFAdjusted, the generation of adversarial examples may seem to take less time with BERTScore and USE at first glance, however, this is because the generation process fails to produce nearly any successful examples, which causes it to appear faster. These facts are also supported by the observed ASR that was close to 0 for USE and BERTScore. On the contrary, SPE achieves much higher ASR on TFAdjusted, therefore many more successful attacks are generated than with USE or BERTScore. Also, the time complexity of SPE still compares to other approaches while producing valid examples.

6 Conclusion

We propose a new sentence-encoder technique, SPE, for textual adversarial attacks. Our method outperforms existing sentence encoders used in adversarial attacks by achieving $1.2\times \sim 5.1\times$ better real attack success rate. Due to the usage of fastText models [19], SPE is also up to 24% faster than the existing techniques on TextFooler [17].

We have also shown that up to 70% of the sentences generated by existing state-of-the-art adversarial attacks should be discarded, because they do not preserve their original meaning. With the usage of SPE, the number of incorrect adversarial examples drops remarkably.

Due to the immense growth of online content, the need for defense against adversarial attacks is higher than ever. We demonstrate the core issue of textual adversarial attacks to the community to stay ahead of attackers. Our method, SPE, also addresses this core weakness and helps generate better adversarial examples, which could be used to strengthen existing systems by adversarial training.

Our code is released as a plugin that can be used in any existing adversarial attack to improve its quality and speed up its execution.

Ethics Statement

With our proposed technique, SPE, we demonstrate the core issue of textual adversarial attacks to the community to stay ahead of attackers. Our method, SPE, also addresses this core weakness and helps generate better adversarial examples, which could be used to strengthen existing systems through adversarial training.

Our method also improves the efficiency of adversarial attack generation, and thus is environmentally friendly. We achieve up to 24% speed-up compared to the SOTA encoders on TextFooler [17]. We also release the source code to make the application of our encoder efficient and as easy as possible.

Acknowledgements

We would like to thank Daniela Hradilova for the useful discussions and suggestions. Our work is part of the RICAIP project that has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 857306.

References

[1] Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. Generating natural language adversarial examples, 2018.

[2] Issa Annamoradnejad, Jafar Habibi, and Mohammadamin Fazli, 'Multi-view approach to suggest moderation actions in community question answering sites', *Information Sciences*, **600**, 144–154, (2022).

[3] Francesco Barbieri, Jose Camacho-Collados, Luis Espinosa Anke, and Leonardo Neves, 'TweetEval: Unified Benchmark and Comparative Evaluation for Tweet Classification', in *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 1644–1650, Online, (November 2020). Association for Computational Linguistics.

[4] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning, 'A large annotated corpus for learning natural language inference', *CoRR*, **abs/1508.05326**, (2015).

[5] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. Universal sentence encoder, 2018.

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

[7] William B. Dolan and Chris Brockett, 'Automatically constructing a corpus of sentential paraphrases', in *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, (2005).

[8] Xinshuai Dong, Anh Tuan Luu, Rongrong Ji, and Hong Liu. Towards robustness against natural language word substitutions, 2021.

[9] Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou, 'HotFlip: White-box adversarial examples for text classification', in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 31–36, Melbourne, Australia, (July 2018). Association for Computational Linguistics.

[10] Wee Chung Gan and Hwee Tou Ng, 'Improving the robustness of question answering systems to question paraphrasing', in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 6065–6075, Florence, Italy, (July 2019). Association for Computational Linguistics.

[11] Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi, 'Black-box generation of adversarial text sequences to evade deep learning classifiers', in *2018 IEEE Security and Privacy Workshops (SPW)*, pp. 50–56, (2018).

[12] Siddhant Garg and Goutham Ramakrishnan, 'BAE: BERT-based adversarial examples for text classification', in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6174–6181, Online, (November 2020). Association for Computational Linguistics.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, 'Deep residual learning for image recognition', in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, (2016).

[14] Xuanli He, Lingjuan Lyu, Lichao Sun, and Qiongkai Xu, 'Model extraction and adversarial transferability, your BERT is vulnerable!', in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2006–2012, Online, (June 2021). Association for Computational Linguistics.

[15] Matthew L. Henderson, Rami Al-Rfou, Brian Strope, Yun-Hsuan Sung, László Lukács, Ruiqi Guo, Sanjiv Kumar, Balint Miklos, and Ray Kurzweil, 'Efficient natural language response suggestion for smart reply', *CoRR*, **abs/1705.00652**, (2017).

[16] David Herel, *Adversarial Attacks on Text Classifiers*, Master's thesis, Czech Technical University in Prague, 2022.

[17] 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', in *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pp. 8018–8025. AAAI Press, (2020).

[18] Erik Jones, Robin Jia, Aditi Raghunathan, and Percy Liang, 'Robust encodings: A framework for combating adversarial typos', in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 2752–2765, Online, (July 2020). Association for Computational Linguistics.

[19] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomás Mikolov, 'Bag of tricks for efficient text classification', *CoRR*, **abs/1607.01759**, (2016).

[20] Ryan Kiros, Yukun Zhu, Russ R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler, 'Skip-thought vectors', in *Advances in Neural Information Processing Systems*, eds., C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, volume 28. Curran Associates, Inc., (2015).

[21] Alexey Kurakin, Ian Goodfellow, and Samy Bengio, 'Adversarial examples in the physical world', (07 2016).

[22] Alexey Kurakin, Ian Goodfellow, and Samy Bengio, 'Adversarial machine learning at scale', (11 2016).

[23] Zachary Laub. Hate speech on social media: Global comparisons, 2019. Accessed: 2023-01-07.

[24] Dianqi Li, Yizhe Zhang, Hao Peng, Liqun Chen, Chris Brockett, Ming-Ting Sun, and Bill Dolan, 'Contextualized perturbation for textual adversarial attack', in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 5053–5069, Online, (June 2021). Association for Computational Linguistics.

[25] Dianqi Li, Yizhe Zhang, Hao Peng, Liqun Chen, Chris Brockett, Ming-Ting Sun, and Bill Dolan, 'Contextualized perturbation for textual adversarial attack', in *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*, (2021).

[26] Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang, 'Textbugger: Generating adversarial text against real-world applications', *ArXiv*, **abs/1812.05271**, (2019).

[27] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A Robustly Optimized BERT Pretraining Approach, July 2019.

[28] Tomás Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean, 'Distributed representations of words and phrases and their compositionality', *CoRR*, **abs/1310.4546**, (2013).

[29] John Morris, Eli Lifland, Jack Lanchantin, Yangfeng Ji, and Yanjun Qi, 'Reevaluating adversarial examples in natural language', in *Findings of the Association for Computational Linguistics: EMNLP 2020*. Association for Computational Linguistics, (2020).

[30] John Morris, Eli Lifland, Jack Lanchantin, Yangfeng Ji, and Yanjun Qi, 'Reevaluating adversarial examples in natural language', in *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 3829–3839, Online, (November 2020). Association for Computational Linguistics.

[31] Bo Pang and Lillian Lee, 'Seeing Stars: Exploiting Class Relationships for Sentiment Categorization with Respect to Rating Scales', in *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pp. 115–124, (2005).

[32] Danish Pruthi, Bhuwan Dhingra, and Zachary C. Lipton, 'Combating adversarial misspellings with robust word recognition', in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 5582–5591, Florence, Italy, (July 2019). Association for Computational Linguistics.

[33] 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*, pp. 1085–1097, Florence, Italy, (July 2019). Association for Computational Linguistics.

[34] Elvis Saravia, Hsien-Chi Toby Liu, Yen-Hao Huang, Junlin Wu, and Yi-Shin Chen, 'CARER: Contextualized affect representations for emotion recognition', in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 3687–3697, Brussels, Belgium, (October-November 2018). Association for Computational Linguistics.

[35] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus, 'Intriguing properties of neural networks', in *International Conference on Learning Representations*, (2014).

[36] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman, 'GLUE: A multi-task benchmark and analysis platform for natural language understanding', *CoRR*, **abs/1804.07461**, (2018).

[37] Boxin Wang, Hengzhi Pei, Boyuan Pan, Qian Chen, Shuohang Wang, and Bo Li, 'T3: Tree-autoencoder constrained adversarial text generation for targeted attack', in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6134–6150, Online, (November 2020). Association for Computational Linguistics.

[38] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. BERTScore: Evaluating Text Generation with BERT, February 2020.

[39] Wei Emma Zhang, Quan Z. Sheng, Ahoud Alhazmi, and Chenliang Li, 'Adversarial attacks on deep-learning models in natural language processing: A survey', *ACM Trans. Intell. Syst. Technol.*, **11**(3), (apr 2020).

[40] Xiang Zhang, Junbo Zhao, and Yann LeCun, 'Character-level convolutional networks for text classification', *Advances in neural information processing systems*, **28**, (2015).

[41] Zhengli Zhao, Dheeru Dua, and Sameer Singh, 'Generating natural adversarial examples', in *International Conference on Learning Representations*, (2018).