Exploring TEXTFOOLER's Syntactically- and Semantically-Sound Distributed Adversarial Attack Methodology

Stanford CS224N Custom Project

Josiah Wong
Department of Mechanical Engineering
Stanford University
jdwong@stanford.edu

Abstract

TEXTFOOLER[1] is a recent adversarial attack model that establishes itself to be a potentially devastating method against a diverse set of NLP models, including the well-acclaimed BERT[2] model. However, much of TEXTFOOLER's methodology is chosen simply based upon intuition and lacks direct comparison to alternative substitutions within its attack pipeline. This project explores the performance of the vanilla TEXTFOOLER model compared to standard variations commonly seen in other adversarial attack methods, and validates the methodology proposed by the original authors. Conducted experiments substituted white-box, distributional sampling, and transformer-based semantic similarity metrics in place of their equivalent counterparts in TEXTFOOLER's original methodology. However, within the context of binary classification, empirical results suggest the original TEXTFOOLER method to be most robust and effective against multiple types of sequence classification, despite failing to sufficiently meet human evaluations of semantic similarity.

1 Key Information

- External collaborators: N/A
- Mentor: Amita Kamath
- Sharing project: N/A
- Grading Option (Option 3 of 3)

2 Introduction

With the rapid ubiquitization of machine learning systems, the topic of adversarial attacks have become increasingly important and urgent. An adversarial attack consists of relatively minute perturbations across a given input's dimension space that causes a trained neural network classifier model to incorrectly identify the input [3]. Often times, these adversarially-generated examples are nearly imperceptible to humans, raising concern for future detection and identification of malevolent attacks upon such deep neural models while deployed during runtime.

Like many other advancements in deep learning, adversarial attacks originally were explored within the domain of computer vision (CV) and images [4], though attacks within the natural language processing (NLP) domain have also garnered recent attention [5]. However, as the above survey notes, there exists unique distinctions that arise from the respective continuous / discontinuous nature of inputs between the CV / NLP domains. Central to these problems is the discrepancy in locality and weight of such individual inputs within the context of the overall image / sequence: In contrast

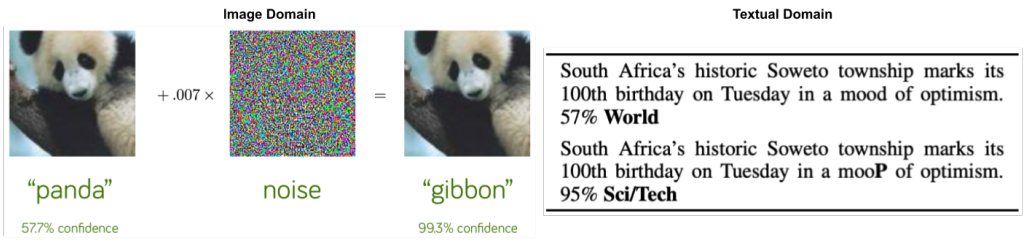

Figure 1: Examples of adversarial image example *(left)* [7] and textual example *(right)* [8]

to individual pixels, which often only hold semantic weight in relation when considered within a group, single word (or even character) discrepancies can dramatically alter the semantics of a given sequence (Figure 1). Compounding this issue is the fact that words and characters, as opposed to pixel colors, are inherently discrete – even with the many embedding methods available, an encoded word / character cannot be marginally shifted and then remapped to word / character space coherently [6]. As a result, much of the recent work within the NLP domain of adversarial attacks have sought to explore methods for generating minimally perturbed examples that alter classifier outputs while maintaining semantic coherency.

A recent novel attack pipeline known as TEXTFOOLER has been shown to generate adversrial examples across a diverse set of NLP classification tasks that exhibit greater semantic similarity and coherency when compared to prior methods [1]. However, neither in-depth ablation studies nor have comparisons with obvious alternatives to sub-components of their attack pipeline have been evaluated. This project seeks to explicitly bridge this gap in knowledge, and contributes in the following ways:

1. We compare TEXTFOOLER's methodology against a few obvious variants, and validate its robustness and performance within the binary classification setting
2. We evaluate TEXTFOOLER as an adversarial sample augmentation method and reveal its inability to successfully defend against itself at attack time
3. We showcase generated examples from both the original and variant TEXTFOOLER models and highlight the difficulty of automating a sound semantic similarity metric

3 Related Work

Within the realm of NLP, multiple methods of generating adversarial examples have been proposed. As mentioned in [5], generating semantically- and syntactically-coherent adversarial examples in an automatic fashion is incredibly difficult, as an adversarial perturbation can often result in non-compatible parts of speech (POS) or incorrect spellings of words, which immediately degrade the quality of the original sequence. Indeed, the difficulty of generating coherent textual sequences has remained unsolved to date, with many of the prior successful works either hand-crafting individual adversarial examples or accepting the fact that their attacks will exhibit varying levels of degeneracy [9],[10],[8].

However, many recent works have explored methods of increasing the semantic and syntactic coherency of generated examples, with seemingly high levels of success [11],[12]. However, it is difficult to determine if the showcased results are simply cherrypicked or if they are truly representative of the typical generated adversarial example; moreover, many of these methods still require a human-in-the-loop evaluation or lack a fundamental quantitative metric for gauging semantic similarity to the original sequence.

TEXTFOOLER[1] aims to automate both the generation and evaluation of proposed adversarial examples, and presents itself as a black-box method that validates individual adversarial perturbations' POS compatibility, impact on the target model, and semantic similarities on both the word- and sub-sequence level. This is achieved in an iterative manner by selectively modifying the individual words whose omission in the sequence results in the largest change in probability distribution from the output classifier. The overall process can be briefly described as follows:

1. Rank words within a given (clean) input sequence by their *importance*, that is, the relative impact on the target model's outputted probability distribution. This is achieved by querying the target model with a modified input sequence $X_{\setminus w_i}$ for all w_i, where each w_i is the word whose importance is currently being measured and $X_{\setminus w_i}$ is the original sequence with the specified word removed.
2. Filter out words from the sorted set that often result in grammatical destruction (e.g.: "the", "when", "and", stop words, etc.)
3. Starting with the highest ranked word, find a suitable set of candidate adversarial words to replace chosen word w_i. This is achieved by choosing a pool of N words that are exhibit some minimal threshold cosine similarity δ to w_i.
4. From this list, filter out words that don't share the same POS or result in too low of a semantic similarity score – this is done to minimize incoherency and divergence from the original sequence's semantics.
5. For each of the remaining candidate adversarial words c_k, query the target model with w_i replaced with c_k and examine the shift in output probability distribution. If any substitution causes the target model to misclassify, the attack is successful (done). Else, permanently replace w_i with the c_k that results in the minimal correct classification probability, and repeat the prior two steps with $w_i + 1$.
6. If all words have been altered without changing the target model's prediction, then the attack has failed (done).

It should be noted that filtered words are selected from the NLTK[1] and spaCy[2] repositories, with the latter also being utilized for POS tagging / filtering. The word-level similarity score was based upon the counter-fitting procedure proposed in [13], while the sub-sequence-level similarity score is evaluated using the Universal Sentence Encoder model [14] that uses transformer attention-based encoding to map sequences of words to single high-level embeddings that capture semantic features.

4 Approach

While TEXTFOOLER's results appear impressive at first glance, the authors fail to cross-validate their proposed attack model against obvious alternatives to quantify its relative significance within the general adversarial attack landscape. Therefore, we evaluate variants of TEXTFOOLER and compare the resulting performance to the original model is the following ways:

- Gradient-based Importance Ranking (white-box attacks): Instead of empirically evaluating each word's relative importance in a black-box manner as in Step 1, the attack will now have access to the target model's backpropagated gradient to directly optimize word importance rankings.
- Distributional Substitution Sampling: Instead of greedily selecting the top N most similar words as in Step 3, the attack will now softmax over the top N cosine scores and probabilistically sample some number of synonyms $M < N$ from this newly normalized probability distribution.
- Alternative Semantic Similarity Metric Model: Instead of using the USE model to gauge semantic similarity between adversarial and original sequences as in Step 4, a more-recent encoding model utilizing twin ("Siamese") BERT networks as its base will be used [15].

In addition, cross-validate the transferability of TEXTFOOLER across similar but distinct datasets, and further explore the usability of TEXTFOOLER as an effective adversarial training method. To provide fair comparison to the baselines presented by the original authors, we use an identical neural network architecture – namely, the uncased BERT[2] base model with a single fully-connected classification layer added for fine-tuning.

It should immediately be noted that the entire TEXTFOOLER implementation[3] as well as baseline results and models were open-sourced and therefore leveraged to accelerate progress for this project.

[1] https://www.nltk.org/

[2] https://spacy.io/

[3] https://github.com/jind11/TextFooler

However, all of the above proposed modifications to the original TEXTFOOLER method were novel implementations, though referenced models (i.e.: the Siamese sentence-encoding network model[4] and BERT base training[5]) were taken OTS.

5 Experiments

5.1 Data

To account for the heavily compressed timeframe for this project, the scope of the above variations were limited to the binary classification NLP task, and specifically the imdB dataset[6], as it posed an above-average challenge due to its extended data sample sequence lengths. For this dataset, inputs take the form of paragraph-long movie reviews which are labeled as either *positive* (1) or *negative* (0). Additionally, to analyze TEXTFOOLER's transferability across similar (but previously unseen) data distributions, we evaluated both the original (black-box) and white-box variant of TEXTFOOLER on the SST-2 dataset[7], which is composed of sentence-level instead of paragraph-level movie reviews that are similarly classified.

5.2 Evaluation Method

Since our contributions center around TEXTFOOLER's performance, we utilize the same performance metrics used the authors, namely:

- acc_{pre}, acc_{post}**:** pre- and post-attack accuracy (attack success rate)
- $\bar{r}$**:** average perturb ratio per sample (size of perturbation relative to overall sequence length)
- s**:** semantic similarity (as measured by USE or an alternative automatic model)
- $\bar{q}_n$**:** average number of model queries per sample (attack efficiency)

It is important to note, however, that many of these quantitative metrics are inherently conflicting, with a decrease in perturbation magnitude naturally leading to decreases in attack success rate. As a result, qualitative analysis of an attack model's performance is subjective – indeed, it is easy to further degrade the victim classifier model by simply lowering the semantic similarity requirements and / or increasing the maximum perturb ratio; however, these decisions do not necessarily imply a "better" attack model. Therefore, qualitative evaluation consists of thoughtful remarks about observed trends within our results, as well as more in-depth analysis of selected examples from the proposed models.

5.3 Experimental Details

For creating the target BERT classifier model, the OTS pretrained BERT (base, uncased) model from the `huggingface` repo was initially fine-tuned on the imdB dataset to validate the authors' own reported model. However, once validated, the authors' published pretrained BERT model used for their own attack evaluation was utilized as the pretrained "base" for the following experiments in order to maintain consistency with the authors' results.

Additional target models were created using a simplified version of TEXTFOOLER's methodology as an adversarial sample augmenter. For a given minibatch being passed to the model during training, an additional training step is taken where the original sequence is perturbed by a specific amount. Multiple variations were used for the exact perturbation, and are described briefly below with each resulting in a separate target model:

- **White N-Nearest Neighbor [Method A]:** Using the model's backpropagated gradient, substitutes the top N most significant words with their respective most similar synonyms, as measured by their counter-fitted similarity.

[4]https://github.com/UKPLab/sentence-transformers

[5]https://github.com/huggingface/transformers

[6]https://ai.stanford.edu/~amaas/data/sentiment/

[7]https://github.com/AcademiaSinicaNLPLab/sentiment_dataset

Table 1: TextFooler Variants' Attack Evaluation on imdB 1000-Sample Test Dataset

Finetuning Method	Attack Method	acc_{pre}	acc_{post}	$\bar{r}$	$\bar{q}_n$
Baseline Model	Original	90.90%	13.50%	**6.12%**	1134.5
	Gradient-based		**12.20%**	9.10%	**942.0**
	Grad w/ Distr. Sampling		15.70%	10.39%	978.1
White 5-NN	Original	**92.40%**	0.90%	7.47%	883.0
White 10-NN	Original	91.50%	**0.50%**	7.48%	873.8
White 10%-NN	Original	90.70%	0.70%	7.13%	835.9
	Gradient-based		0.90%	11.31%	818.7
	Grad w/ Distr. Sampling		3.70%	13.06%	901.1
Distr. White 10%-NN	Original	91.00%	1.70%	7.72%	897.4
	Gradient-based		2.30%	11.95%	904.3
	Grad w/ Distr. Sampling		4.40%	14.10%	987.0
Distr. White 20%-NN	Original	**92.40%**	0.70%	**6.85%**	**813.3**
	Gradient-based		0.90%	11.36%	829.7
	Grad w/ Distr. Sampling		3.20%	13.31%	905.1

- **White $P\%$-Nearest Neighbor [Method B]:** Similar to the method above, but for each input sequence substitutions are continually added until P percent of the total words of the sequence have been perturbed.
- **Distributional White $P\%$-Nearest Neighbor [Method C]:** Similar to the method above, but utilizes the Distributional Substitution Sampling described in the **Approach** section to sample substitutions until the proportional limit is reached. Moreover, after every substitution, there is an ϵ probability that the ongoing perturbation for a given sequence terminates immediately.

For comparing TEXTFOOLER's original baseline performance on the imdB dataset to our proposed variants, multiple attacks were conducted against each target model using the same 1000-size sample test dataset. For each target model, the **original** (black-box + deterministic), **gradient-based** (white-box + deterministic), and **gradient-based distributed sampling** (white-box + sampling) attacks were performed, with the resulting pre-/post- target model accuracy, average percentage of words changed, and average number of model queries recorded.

A brief case study was also conducted comparing the qualitative performance of the Siamese BERT network sentence encoding model [15] to the USE method for measuring semantic similarity between sequences. While these results did not lead to large-scale quantitative results, individual analysis between sequences are discussed in the **Analysis** section.

Lastly, an additional BERT model was finetuned (from the raw pretrained model from `huggingface`) on the SST-2 training dataset, and evaluated against both the white- and black-box variants of TEXTFOOLER with varying levels of semantic similarity threshold requirements for the generated adversarial examples.

All models were trained, executed, and evaluated on the Microsoft Azure Cloud platform[8] using a linux-based VM equipped with 6 CPU cores, 56 GiB RAM, and 1 M60 GPU. For the specific training statistics and hyperparameters used for finetuning the pretrained BERT models, please reference Appendix A.

5.4 Results

imdB Dataset. The evaluation results using the imdB dataset can be seen in Table 1. Perhaps surprisingly, all variants of TEXTFOOLER work remarkably well across all target models tested against it. In fact, we were unable to replicate the authors' preliminary adversarial training results and have instead shown that increasing adversarial training substantially deteriorates the resulting performance when subjected to attacks during test time. Interestingly, the normal accuracy of adversarially-trained models actually supersedes the baseline, suggesting that the adversarially-trained models indeed

[8] https://azure.microsoft.com/en-us/

Table 2: Varying Attack Semantic Similarity s Requirement on SST-2 1000-Sample Test Dataset

Minimum s	Attack Method	acc_{pre}	acc_{post}	$\bar{r}$	$\bar{q}_n$
0.70 (Baseline)	Original	92.10%	10.60%	17.94%	149.5
	Gradient-based		11.60%	24.37%	170.1
0.80	Original		27.00%	20.32%	217.4
	Gradient-based		28.50%	25.91%	226.3
0.90	Original		57.80%	21.87%	295.0
	Gradient-based		58.10%	28.81%	287.1

Table 3: Comparison between Black-Box and White-Box Generated Adversarial Example

Attack Type	Label	Sequence
None (Original)	0	"even as lame horror flicks go , this is lame"
TEXTFOOLER	1	"even as lame horror flicks go , this is *cruddy*"
Gradient-based	1	"even as *nil spooky gestures partir* , this is *cruddy*"

become more robust to sample input variations, but have failed to generalize sufficiently to account for specific gaps in its semantic knowledge. This is to be expected, as TEXTFOOLER inherently iterates over individual synonyms until the "optimal" adversarial candidate is chosen. Thus, while the adversarially-trained models may have been partially regularized to account for out-of-distribution data, an insufficient amount of examples were provided to account for the plethora of edge cases. A better approach in our methodology would have been to extend the per-model adversarial training, as our fine-tuning was constrained by time and therefore limited to a single epoch through the dataset.

This raises an additional important observation about TEXTFOOLER and its variants. Because it is an inherently iterative attack, its attacks are both sample inefficient and slow, requiring hundreds of model queries per single attack. In wall time, even with GPU acceleration, this amounted to tens of seconds being required per sample during both training and test time. As can be seen in the baseline model evaluation, an observed advantage of our proposed white-box gradient-based variant of TEXTFOOLER is its ability to implicitly bypass multiple model queries during attack time due to the back-propagated gradients immediately providing efficient adversarial exploration. Note, however, that this benefit seems to be nullified once the target model is adversarially trained, with all types of attacks requiring relatively similar querying levels. It is important to note, however, that these results are bounded in scope within the binary classification setting with the imdB dataset, and cannot necessarily be extrapolated to other sub-domains within NLP.

SST-2 Dataset. Attack results comparing the effects of semantic similarity constraints using the cross-validation target model fine-tuned on the SST-2 dataset can be seen in Table 2. As is expected, increasing s correlates to a decreasing attack success rate. Additionally, both the average perturb ratio and query size increased as well, though this seems to fail to increase the effectiveness of the attack model. Similar to the results seen from the imdB dataset, the original black-box attacks perform marginally better and require distinctly less perturbations to achieve the similar accuracy deterioration compared to our proposed vanilla white-box variant.

The increasing perturbation ratios coupled with the decreasing attack success rates seen from the SST-2 dataset results suggest that sheer perturbation size is insufficient to reliably fool state-of-the-art models such as BERT. Instead, it seems the semantics of individual words exert much more impact on the inner workings of BERT. Indeed, the shift from 0.80 to 0.90 semantic similarity requirement for the attack resulted in more than doubling of the target model's post-attack accuracy! This suggests that the semantic similarity does in fact impose a non-trivial constraint on the attack model that dramatically alters the performance of a given target model.

6 Analysis

Trends. There are multiple observations that are immediately striking. Perhaps the most obvious is the fact that apart from the baseline model, gradient-based attacks actually perform more poorly

Table 4: TextFooler Attack Semantic Similarity Evaluation Example

Original Sequence	"...75 people voted the same out of 146 total votes..."
Perturbed Sequence	"...75 populace voted the same out of 146 ensemble polling..."
Semantic Sim Score [14]	0.582 (USE)
Semantic Sim Score [15]	0.962 (Siamese-BERT)

compared to their adversarial counterparts. On the one hand, this can be partially accounted for by the fact that the adversarially-trained models utilized a nearly identical process for increasing its own robustness, and therefore were partially inoculated against similarly-based attacks. However, this does not account for the discrepancies seen in the SST-2 results, where gradient-based attacks proved to be both less effective and efficient against the non-adversarially trained model.

A more likely explanation, therefore, lies in a more intriguing, and perhaps much more surprising, suggestion based on the empirical results. While our proposed white-box method does indeed leverage the backpropagated gradient to gain insight into the "ground truth" marginal importances of each input word in a given sequence, it only utilized the *magnitude* fails to provide an optimal *direction* to which an adversarial perturbation should be applied. This reveals an important and often overlooked distinction between model defenses and attacks – while the former seeks to optimize for robustness, the latter seeks to minimize its detectability (while simultaneously maximizing attack success). And while adding random stochastic noise is a common method for improving the former, this does not directly translate to similar results for the latter, with stochastic variations being carefully applied in optimal directions such that the adversarial attack does not nullify itself. As language modeling inherently consists of discrete inputs (that are then mapped sparsely into a continuous space), coherent attacks cannot simply perturb uniformly in a given direction, but must instead search its local space for localized, valid locations in the continuous word embedding space to find suitable word replacements.

Impact of Word Importance Ranking. An interesting implication of our empirical results suggests that vanilla gradient-based attack models may not inherently be the most optimal method for targeting specific elements within a text sequence. Consider the showcased example shown in Table 3 which is representative of many other examples generated from the respective attack methods. While both the original TEXTFOOLER and gradient-based methods produce successful adversarial examples, it is obvious that the black-box method using TEXTFOOLER's word importance ranking scheme is much more effective at singling out the most semantically important words within the sentence. Not only does this highlight the empirical robustness of TEXTFOOLER's ranking method, but this also suggests that internal interplay between words and the resulting classifier output cannot be directly captured by each inputs' respective backpropagated gradient. This is intuitively plausible, as TEXTFOOLER's attack method is inherently nonlinear and discontinuous (e.g.: filtering out of specific words, adding semantic requirements to word selection, etc.) and causes the information provided by the target model's gradient to be less significant and even incorrect under the context of these imposed constraints.

Impact of Semantic Similarity Constraint. As mentioned before, the semantic similarity constraint imposed by TEXTFOOLER is both non-trivial and highly sensitive to tuning – it dramatically shifts the dynamic behavior between the attack's detectability and effectiveness. Therefore, it was essential to discern whether TEXTFOOLER's USE metric was sufficient. While the USE metric is far from perfect, empirical comparisons with the newest state-of-the-art sentence encoder from [15] reveal the general difficulty in automatically evaluating sequence similarities. Consider the showcased example shown in Table 4 which is representative of other examples when conducting our brief semantic similarity case study. While the *denotations* between the original and perturbed sequences remain generally preserved, the *connotations* significantly diverge, with the perturbed sequence being clearly awkward and unwieldy to the average human reader. Comparison of the resulting scores suggests that the USE has much more fine-grained understanding of individual word semantics as opposed to the Siamese-BERT encoder which appears to generally capture high-level semantics.

Yet, even USE often generates poor semantic results, as seen in Table 3's Gradient-based output. This highlights the gap between surface-level heuristics which the neural models (including both USE

and Siamese-BERT encoder) seem to employ and the much more complex and nuanced fundamental understanding of semantic contextualization that humans perceive. Such obvious disparities juxtapose the consistent ability of TEXTFOOLER to fool BERT-based models, and emphasize the necessity for robustly quantifying models' semantic understanding before blindly applying them to various NLP tasks.

7 Conclusion

In this work, we proposed multiple alternatives of the TEXTFOOLER adversarial attack model, including gradient-based and distributional sampling variants across multiple datasets, and showed that TEXTFOOLER's methodology performs surprisingly better than our proposed white-box methods. Moreover, we reveal how gradient-based methods for optimizing adversarial attacks do not directly lead to productive results when applied within the discrete-input NLP domain. However, while TEXTFOOLER is successful, we also highlight the discrepancy between the automatically measured semantic similarity and human-based qualitative judgment, and emphasize the need to focus on accurately gauging semantic coherency before further extending and complicating deep textual language models. While this work does not offer explicit solutions to solving the many problems of NLP adversarial models uncovered in this project, our hope is to provide the reader with a tangible and empirical motivation that will hopefully inspire future breakthroughs in this quickly-increasing area of research.

References

[1] 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, 2019.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018.

[3] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks, 2013.

[4] Anirban Chakraborty, Manaar Alam, Vishal Dey, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. Adversarial attacks and defences: A survey, 2018.

[5] Wei Emma Zhang, Quan Z. Sheng, Ahoud Alhazmi, and Chenliang Li. Adversarial attacks on deep learning models in natural language processing: A survey, 2019.

[6] Boxin Wang, Hengzhi Pei, Han Liu, and Bo Li. Advcodec: Towards a unified framework for adversarial text generation, 2019.

[7] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples, 2014.

[8] Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. Hotflip: White-box adversarial examples for text classification. 2017.

[9] Robin Jia and Percy Liang. Adversarial examples for evaluating reading comprehension systems, 2017.

[10] Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. Universal adversarial triggers for attacking and analyzing nlp, 2019.

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

[12] Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. Adversarial example generation with syntactically controlled paraphrase networks, 2018.

[13] Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gašić, Lina Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. Counter-fitting word vectors to linguistic constraints, 2016.

[14] 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.

[15] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks, 2019.

A Appendix

Table 5: Training Details and Hyperparameter Selection

Parameter	Value
Epochs	3
Batch Size	8
Learning Rate	5e-5
Max Grad Norm	1.0