Subtle Subterfuge: Highlighting Language Model Vulnerabilities on Semantically Similar Inputs

Anirudh Cowlagi
University of Pennsylvania
acowlagi@seas.upenn.edu

Eric Chen
University of Pennsylvania
ebchen@wharton.upenn.edu

Ria Sharma
University of Pennsylvania
rias@wharton.upenn.edu

Justin Zhou
University of Pennsylvania
justinzh@wharton.upenn.edu

Abstract

Natural language is a particularly interesting domain for machine learning because language often contains redundancies — there are many ways to say the same thing. Consequently, we might desire that effective natural language models are robust to such redundancies – semantically indistinguishable inputs should have identical outputs. In this work, we demonstrate that modern models do *not* yet demonstrate such robustness by exploring and extending methods to generate adversarial examples for sentiment classification tasks. The examples generated through these methods meet or exceed state-of-the-art benchmarks with regard to various metrics, most notably including attack success rate and semantic similarity to the original sequences.

1 Introduction

Modern language models, particularly those underpinned by deep network architectures, have demonstrated great success across a wide range of tasks including sequence classification, language translation, and natural language inference (Lo Bosco and Di Gangi, 2017)(Kocmi and Federmann, 2023)(Wang and Jiang, 2015).

However, recent work has shown that deep networks are susceptible to *adversarial examples*, generated by added a small amount of "imperceptible" noise to legitimate inputs so as to fool the model to make undesirable predictions (Cheng et al., 2020), (Nguyen et al., 2015). Still, many of these works have been in domains where the inputs are continuous-valued – images, time series. We are interested in the problem of generating adversarial examples for *text*. The discrete nature of text makes many of the developed methods for continuously-valued data untenable, while the requirement of "imperceptible" change becomes harder to satisfy – even changing a single word can *entirely* change the meaning of a sentence.

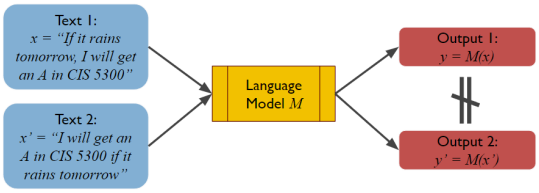

Figure 1: Objective: Given input x and language model M, find an adversarial example x_{adv} such that $S(x_{\text{adv}}, x) \geq \tau$ but $\hat{y} = M(x) \neq M(x_{\text{adv}}) = \hat{y_{\text{adv}}}$

What is the relevance of these adversarial examples? The answer is straightforward – understanding how to 'attack' modern language models also helps us train models that are robust to such adversarial inputs, ultimately resulting in models that capture the structure and redundancies of natural language more completely. This in mind, we may proceed to specify our goal more precisely.

1.1 Problem Overview

In this work, we look to implement, evaluate, and extend existing methodologies to successfully execute an adversarial attack on a language model M and text input x with respect to some natural language task T (e.g. sentiment classification). Specifically, an "adversarial attack" informally does the following (also see **Figure** 1:

1. Find an x_{adv} such that $y_{\text{adv}} = M(x_{\text{adv}})$ is undesirable for the original input x (e.g. causes model misclassification)

2. Ensure that x_{adv} and x are "similar enough" (e.g. semantically)

Let us now formalize this notion of an attack using the framework specified in (Morris et al., 2020)

1.2 Formalizing the Notion of an Attack

According to (Morris et al., 2020), attacks are composed of 4 components:

1. Goal Function: stipulate the goal of the attack, like to change the prediction score of a classification model, or to change all of the words in a translation output.

2. Constraints: determine if a potential perturbation is valid with respect to the original input.

3. Transformations: take a text input and transform it by inserting and deleting characters, words, and/or phrases.

4. Search Methods: explore the space of possible transformations within the defined constraints and attempt to find a successful perturbation which satisfies the goal function.

With this framework in hand, we now present a brief review of existing approaches towards constructing adversarial examples, particularly in the context of sequence classification, where the goal is to construct an example that minimizes the score / confidence of the correct label until it is no longer the predicted label.

2 Related Work

Each of the works we describe below presented recent state-of-the-art / near state-of-the-art approaches to generating adversarial examples for sequence classification models.

2.1 TextFooler

TextFooler (Jin et al., 2020) develops an early *black-box* approach – only requiring access to model predictions and scores of output classes, as opposed to model weights and/or gradients – to generate adversarial examples that looks to swap relevant words in the original input x to induce misclassification. This strategy proceeds in two stages.

First, the *importance* of each word is determined by how much they modify the confidence on the correct logit if they were to be deleted from the input:

$$\begin{aligned} I_{w_j} &= P_y(w_1, \cdots, w_m) \\ &\quad - P_y(w_1, \cdots w_{j-1}, w_{j+1}, \cdots, w_m) \end{aligned} \tag{1}$$

Next, words are iteratively swapped out of the original sentence using nearest-neighbors in a dense word embedding space (e.g. GLoVe embeddings (Pennington et al., 2014)). Then, if the swapped word satisfies grammatical constraints (same part-of-speech) and the new sentence is semantically similar enough (see **Section** 2.1.1 to the original sentence, we proceed to the next word. If at any time, the current perturbed sentence causes the model to misclassify the input, the perturbed sentence is returned.

The approach is rather successful as they are able to reduce the sentiment classification accuracy of various language models from $> 90\%$ to between 12 and 15%. At the same time, they maintain high semantic similarity ($\approx 0.8 - 0.9$) and perturb only a small fraction of words ($\leq 10\%$).

2.1.1 Semantic Similarity Check

The semantic similarity between x and x_{adv} is assessed by embedding x and x_{adv} into a high-dimensional vector space, then computing their cosine similarity. The embedding is produced using the Universal Sentence Encoder of (Cer et al., 2018). The semantic similarity is then:

$$SS(x, x_{\text{adv}}) = \frac{USE(x) \cdot USE(x_{adv})}{\|USE(x)\|\|USE(x_{adv})\|} \tag{2}$$

x_{adv} is a valid perturbation only if $S(x, x_{\text{adv}}) \geq \tau$ for some user-specified $0 < \tau \leq 1$.

2.2 TextBugger

TextBugger (Li et al., 2019) proposes a framework to effectively and efficiently generate semantic-preserving adversarial texts against SoTA text-classification models in both the white-box (access to model weights and gradients) and black-box settings (access to only inputs and outputs). Importantly, TextBugger performs *character*-level edits in addition to swapping entire words.

In both settings, a two-step approach similar to TextFooler is utilized. In the white-box setting, words in a sentence are first ranked by importance using the Jacobian of the model's output (correct class) with respect to each word in the input. Then, going through the words in this ranked order, character- and word-level perturbations are considered – a procedure the authors term as "bugs generation." Character-level perturbations attempt to misspell important words so they are mapped to the [UNK] token. In particular, the authors consider character insertions / deletions, and adjacent character swaps. Word-level perturbations proceed similarly to TextFooler.

In the black-box setting, a similar approach is adopted, except words are ranked using the same

scoring function as TextFooler. They find that this procedure is quite effective at generating adversarial examples, improving upon TextFooler – consistently achieving an attack success rate over 85% for sentiment classification tasks (see **Section** 3.3 for definitions of these metrics) while always perturbing $\leq 6\%$ of the words in the original input.

2.3 BAE: BERT-based Adversarial Examples

BAE (Garg and Ramakrishnan, 2020) presents yet another word-swapping based strategy to produce adversarial examples. The key difference between this and the previous strategies is that BAE leverages a *sequence-aware* masked language model, namely BERT (Devlin et al., 2019), to predict replacement words.

Specifically, after obtaining a ranking of the words in the sequence using the same black-box scheme of the previous two methods, replacement candidates are generated for each word by replacing the word with the [MASK] token and using BERT to predict the most likely words to replace the masked word. Again, canddiate words are pruned using a semantic similarity constraint. Furthermore, this approach not only *replaces* words in the original sequence, but also allows words to be *inserted* into the original sequence.

This approach also achieves success in generating adversarial examples for sentiment classification – reducing the accuracy of the original classification models by up to 75 percentage points ($\approx 90\% \rightarrow 15\%$). The authors also used human evaluation techniques to compare TextFooler and BAE. They found that BAE adversarial examples retained the original sentiment classification more often than TextFooler and were perceived to be more "natural" than TextFooler examples – likely due to the ability to incorporate the *whole* sequence in predicting replacement / insertion words.

3 Experimental Design

3.1 Data and Language Model

For the entirety of this project, we perform attacks on the IMDb Movie Review dataset [CITE IMDB], which comprises of 50,000 movie reviews labeled by whether they are positive (1) / negative (0) reviews. For computational reasons, we *downsample* this dataset to 3500 examples and perform attacks on 3 splits of this dataset (70 % train, 15 % dev, 15 % test).

We also evaluate our attacks against a common model. In particular, we deploy attacks on a fine-tuned version of DistilBERT (Sanh et al., 2019), a distilled version of the base BERT model (40% less parameters, 60% faster). Fine-tuning is done on the IMDb dataset, so our "training" corpus above consists of precisely (a subset) of those examples that DistilBERT was fine-tuned on. We choose this model because it achieves high accuracy on the sentiment classification task, is sufficiently complex (a Transformer model not too dissimilar from SoTA LLMs), and is computationally efficient to evaluate.

3.2 Attack Goal and Constraints

To ensure consistency across all attacks evaluated, we maintain the following across attacks:

- Goal: Untargeted classification (minimize the score of the correct label until it is no longer the predicted label)
- Constraints:
  1. No Repeat Word Modification: Don't modify the same word twice (for computational reasons)
  2. No Stop Word Modification: Don't modify stop words (e.g. "the", "an" – these are unlikely to be the most important words in the sequence, modifications to such words typically only diminish grammatical accuracy)
  3. Maximum Word Perturbation Fraction: Upper bound the fraction of words that can be perturbed (generally a loose constraint, but avoids querying very hard examples too many times)
  4. Semantic Similarity: Sentence similarity based on USE Encodings. Based on literature review of existing attack techniques (Li et al., 2020, 2019), this is set between 0.5 and 0.9 for movie review sentiment classification. We choose 0.8, but hope to perform future ablation study.
  5. Word Similarity: Minimum cosine similarity between original word and peturbed word as determined using GLoVe embeddings. We also set this to 0.8.

3.3 Evaluation Metric

We consider two kinds of evaluation metrics, assessing both how *effective* and how *efficient* an

attack strategy is. Denote by S the set of attacks that were successful (model classified original input correctly, classifies perturbed input incorrectly), by F the set of attacks that were unsuccessful (model classifies original and perturbed input correctly), and by N the set of all examples (note $|N| \geq |S| + |F|$ since the model may not classify all original examples correctly). Metrics are computed using the TextAttack package (Morris et al., 2020).

3.3.1 Attack Effectiveness Metrics

We evaluate the attack effectiveness by the following metrics.

1. Perturbed Accuracy: Model accuracy after perturbed examples are computed for entire dataset –

$$\text{Acc}_{\text{adv}} = \frac{|F|}{|N|} \tag{3}$$

   This metric is reported by (Li et al., 2020, 2019) (Jin et al., 2020) (Garg and Ramakrishnan, 2020)

2. Attack Success Rate: Fraction of inputs the model originally classified correctly that are now misclassified –

$$R = \frac{|S|}{|S| + |F|} = 1 - \frac{|F|}{|S| + |F|} \tag{4}$$

   This metric is reported by (Li et al., 2020) (Li et al., 2019)

3.3.2 Attack Efficiency Metrics

We evaluate attack efficiency by the following metrics:

1. Average USE Similarity: The average semantic similarity across perturbed examples (we over–

$$\overline{SS} = \frac{\sum_{x, x_{\text{adv}} \in S \cup F} SS(x, x_{\text{adv}})}{|S \cup F|} \tag{5}$$

   This metric is reported by (Li et al., 2020) (Li et al., 2019)

2. Fraction of Words Perturbed: Average fraction of words perturbed for all sequences in $S \cup F$. This metric is reported by (Li et al., 2020, 2019)(Jin et al., 2020)

3. Average Attack Time: Average wall-clock time to conduct attack per example (for all examples that are not skipped – i.e. those the model classifies correctly initially).

3.4 Simple Baseline: Random Synonym Swaps

For any attack strategy, since the goal function and constraints have been specified, we have the freedom to pick the transformation and search strategy. We thus implement the following very simple baseline

- Transformation: Replace words in sentence with one of their WordNet (Miller, 1995) synonyms
- Search Strategy: Iteratively replace a random fraction of tokens in input (e.g. up to 15%), terminating if attack is successful or maximum fraction of tokens are replaced

3.4.1 Test Set Results

In **Table** 1 we show the results of this simple baseline by computing the metrics discussed in **Section** 3.3 over the corpus of test examples.

	Simple Baseline
Original Accuracy (%)	93.40
Perturbed Accuracy (%)	85.21
Attack Success Rate (%)	11.62
USE Similarity	0.893
Perturbed Words (%)	13.79
Time (s / sample)	3.75

Table 1: Simple Baseline: Evaluation metrics over test set (IMDb review sentiment classification)

We notice that this simple baseline is certainly able to produce *some* adversarial examples, but its performance is far from any of the approaches discussed in **Section** 2. Thus, we now explore methods to expand on both this simple baseline and approaches previously considered in the literature.

4 Experimental Results

4.1 Published Strong Baseline

For our strong baseline, we largely adopt the methodology of (Jin et al., 2020) to improve both the transformation and search strategy of the simple baseline.

- Transformation: Word swap using nearest neighbors in the GLoVe embedding space. This helps address situations where there may not exist a WordNet synonym, but there is some nearest neighbor that is nonetheless appropriate. Furthermore, it actively attempts to satisfy the word embedding similarity constraint.

	Dev (Original Accuracy: 94.05 %)				
	Perturbed Accuracy (%)	Attack Success Rate (%)	Perturbed Words (%)	USE Similarity	Time (s/sample)
Simple Baseline	78.04	16.31	13.51	0.894	**3.82**
Published Baseline	21.33	77.56	13.01	0.899	6.90
Extension 1	10.42	88.99	7.39	**0.931**	11.21
Extension 2	**2.08**	**97.83**	**6.42**	0.930	45.29
	Test (Original Accuracy: 93.40 %)				
	Perturbed Accuracy (%)	Attack Success Rate (%)	Perturbed Words (%)	USE Similarity	Time (s/sample)
Simple Baseline	85.21	11.62	13.79	0.893	**3.75**
Published Baseline	11.43	87.63	12.87	0.898	6.78
Extension 1	10.42	88.99	7.39	0.931	11.21
Extension 2	**1.67**	**97.13**	**5.22**	**0.938**	38.16

Table 2: Evaluated attack metrics for baselines and implemented extension. Computed over development and test data splits.

- Search Strategy: *Rank* the words based on their importance to the output prediction, then perform a greedy search over these rankings to iteratively swap words. In particular, we employ a procedure where word importances are determined by how much they modify the confidence on the correct logit if they were to be deleted from the input, as prescribed by **Equation** 2.1.

As we see in **Table** 2, this published baseline achieves far superior performance to the previous simple baseline (**Section** 3.4). We are able to successfully attack over 75% of inputs for this sentiment classification task while preserving the semantics of the input. We do note that there is an increase in attack time / example. The performance is comparable to those claimed in (Jin et al., 2020), though we do note that we perform experiments on a randomly sampled subset of the full corpus of reviews and also maintain a slightly modified set of constraints which likely explains any discrepancies in observed performance.

4.2 Extension 1: Improving the Transformation

Above, the transformation strategy uses a word swap strategy that simply swaps a word with one of its nearest neighbors in the GLoVe embedding space. However, this strategy does *not* account for the context surrounding the word that was replaced – words that may not be nearest neighbors in the embedding space may *still* be suitable replacements, when conditioned on the surrounding context of the input sequence.

Consequently, we consider a context-aware replacement strategy using a masked language model – BERT itself! Of course, this, in isolation is not a novel strategy. This is precisely what (Garg and Ramakrishnan, 2020; Li et al., 2020) do.

However, our improved transformation strategy does not stop here. As (Ebrahimi et al., 2017), (Li et al., 2019) show, Semantic similarity can also be maintained by making *character-level* edits to words that are already in the text. For instance, to a human, the word "good" and the word “goood" are semantically indistinguishable, but may lead to misclassifications by the model. Consequently, we also incorporate a number of character-level edits into our transformation pipeline. These include:

1. Random character insertion / deletion
2. Adjacent character swaps
3. Homoglyph swaps (swap similar looking characters – e.g. “O" $\rightarrow$ “0")
4. QWERTY swaps (replace characters with adjacent characters on the QWERTY keyboard)

To the best of our knowledge, the combination of sequence-aware word swapping with character-level edits is a *novel* approach to generating perturbation candidates. For instance, (Ebrahimi et al., 2017)(Li et al., 2019) don’t perform sequence-aware word swapping, while (Li et al., 2020)(Garg and Ramakrishnan, 2020) do not perform character-level edits.

For this extension, we continue to retain the search strategy from the strong baseline: a greedy search over words ranked by importance as determined by effect on the classification probabilities upon word deletion.

The results of this modification are shown in **Table** 2. We achieve substantial improvements over the published baseline, both from an attack

effectiveness ($\approx 6\%$ increase in attack success rate, $\approx 5\%$ reduction in perturbed accuracy), and an attack efficiency (increase in USE similarity, $\approx 5\%$ decrease in perturbed word fraction – higher semantic retention). Of course, composing many more transformations does substantially increase the computation time / sample, but we do note that attacks can be highly parallelized over examples.

4.3 Extension 2: Improving the Search Strategy

In this extension, we focus our efforts on modifying the search strategy from the previous extension. We continue to perform a greedy search for computational reasons, but refine the methodology by which the words are ranked to account for the space of possible transformations under the constraints.

1. Determine *word saliency*: This is the relevance of the word to output predictions, determined by the change in output scores with the word, and when the word is replaced by the [UNK] token:

$$S_{w_j} = \text{softmax}(P_y(w_1, \cdots w_m) - P_y(w_1, \cdots w_{j-1}, [UNK], w_{j+1}, \cdots w_m)) \quad (6)$$

2. For each word in the sequence, compute the *maximum* change in score induced by any valid swap to the word (here, $\mathcal{C}$ is the set of perturbations to the original sentence that satisfy the constraints):

$$\delta P_{w_j} = \max_{w'_j:(w_1,\cdots,w'_j,\cdots,w_m)\in\mathcal{C}} (P_y(w_1, \cdots, w_m) - P_y(w_1, \cdots w'_j, \cdots w_m)) \quad (7)$$

3. Set $I_{w_j} = S_{w_j} \cdot \delta P_{w_j}$

This should enable us to make fewer perturbations than the original approach since we are accounting for the possible transformations to a word *before* even iteratively searching over them! We may even be able to achieve higher attack success rates since this ordering leads to more sequences that satisfy the search constraints.

The key intuition is that 1) the words we try to replace first need to be important, but they also 2) need to be words that are *replaceable* at all (and such that replacing them with a valid replacement does actually lead us towards the goal – i.e. some replacements may not do anything, even though deleting the word changes the prediction score substantially).

The results of this modification are also shown in **Table** 2. Again, we achieve improvements over the strategy from **Section** 4.2. Almost all attacks are successful, and we only need to perturb $< 4\%$ of any given input to induce a model misclassification. Of course, some errors are still made, which we analyze in closer detail in the following section.

4.4 Error Analysis

First, we show some examples of *successful* attacks (original and perturbed) input. Note: We show homoglyph swaps by switching the case of the corresponding letter in the original and perturbed text, but these swaps typically involve replacement by other unicode characters that do not render in LaTeX.

4.4.1 Successful Examples

1.
   - (Original: 99.81 % Negative) Technically I'am a Van Damme Fan, or I was. this movie is so bad that I hated myself for wasting those 90 minutes. Do not let the name Isaac Florentine (Undisputed II) fool you, I had big hopes for this one, depending on what I saw in (Undisputed II), man.. was I wrong ??! all action fans wanted a big comeback for the classic action hero, ... [TRUNCATED – NO CHANGES] ... Do your self a favor, skip it.. seriously.
   - (Perturbed: 53.31 % Positive) Technically I'am a Van Damme Fan, or I was. this mvoie is so negative that I detested myself for squandering those 90 minuteS. Do not let the name Isaac Florentine (Undisputed II) delude you, I had big hopes for this one, depending on what I saw in (Undisputed II), man.. was I wrong ??! all action fans wanted a big comeback for the classic action herO, ... [TRUNCATED – NO CHANGES] ... Do your self a favor, skip it.. deeply.

2.
   - (Original: 99.85 % positive) It is very hard to come up with new information about JFK Jr. and this fine movie

had very little of it, but it was a joy to watch. The casting was very good and the script... [TRUNCATED – NO CHANGES] ... WTBS should be applauded for producing such an entertaining movie.

- (Perturbed: 98.56 % negative) It is very hard to come up with new information about JFK Jr. and this alright movie had very little of it, but it was a jOy to watch. The casting was very good and the script ... [TRUNCATED – NO CHANGES] ... WTBS should be applauded for producing such an entertaining movie.

We see that a *combination* of transformations are utilized, very few perturbations are needed, and the confidence of the final prediction is altered substantially to lead to misclassification. Next, we look at some unsuccessful examples.

4.4.2 Unsuccessful Examples (Errors)

1.
   - (Original: 99.91 % Negative) We brought this film as a joke for a friend, and could of been our worst joke to play. The film is barely watchable, and the acting is dire. The worst child actor ever used and Hasslehoff giving a substandard performance. The plot is disgraceful and at points we was so bored we was wondering what the hell was going on. It tries to be gruesome in places but is just laughable

 Just terrible
   - (Perturbed: 96.30 % Negative) We lodged this film as a joKe for a amigo, and could of been our worst farce to plaY. The film is bareLy watchale, and the acting is catastrophic. The worse kid actor eveer usage and HasslehofF giing a substandard performanCe. The polt is disgracefuL and at pointS we was so bore we was wondeRing what the helL was going on. It attempt to be gruesome in places but is just lsughable.

 Only horrendous

2.
   - (Original: 99.88% Negative) I don't leave IMDb comments about films but this.... this film was bad. very bad. I fast forwarded through most of it, stopping where I hoped the acting had improved since the last scene, only to continue with the fast forwards. Formula plot – once the obvious murderers were discounted, there was only the one left. And that was in the first five minutes. Scene by scene it felt as though I'd already read the script before because there were no surprises, no mystery. The Tori character... bad bad acting. A true waste of time on DVD and a definite 'let's go to bed early' option if it's the only thing on television. If you watch this film, you will find yourself realising you'll never be able to get back the time you've just wasted.
   - (Perturbed: 66.31% Negative) I don't leaving IMDb feedback about films but this.... this film was rotten. very rotten. I quick forwarded through most of it, stopPing where I expected the acitng had improved because the lasT sCene, only to continued with the fasT forwardS. ForMuLa plot – once the palpable murderers were disconted, there was only the one exiteD. AnD that was in the firsT five minuteS. SCene by scene it deemed as however I'd already read the screenplay before because there were no surprises, no puzzle. The ToRi character... negative negative atcing. A true wastE of times on DVD and a clear 'le's going to bed early' option if it's the only thing on television. If you watch this film, you will find yourself accomplishing you'll never be able to get back the yime you've just wasted.

We observe that the main class of sequences that this strategy seems to fail on are shorter, negative sentiment sequences where the primary transformation used is homoglyph swaps. Both examples above consist of < 130 words, less than half of the average number of words for a sequence. This makes sense – with shorter sequences, there are fewer words that can be replaced to *both* retain the semantic content of the message and sufficiently change the output score predictions. It also appears that the underlying language model is robust to such homoglyph swaps, preventing such changes from substantially altering the output predictions. Moreover, negative sentiment sentences typically have more polarizing language, making it harder to fool the model / mask initial sentiment.

5 Conclusions

In this work, we have developed and evaluated attack strategies to generate adversarial examples for the language task of sequence sentiment classification. Our methods include replication of published baselines, as well as novel extensions upon state-of-the-art approaches from the literature. The attack strategies we have evaluated meet or exceed the performance of these approaches according to various evaluation metrics, both from an *effectiveness* and *efficiency* standpoint. Our work reveals that modern, transformer-based language models still remain highly susceptible to adversarial examples, thus requiring the development of more robust defense strategies that may deployed at training or inference time.

Acknowledgements

We would like to thank our TA, Anson Huang, for helpful discussions throughout the course of the project. We would also like to thank Professor Mark Yatskar for guiding us towards this project and for being the course instructor for the Fall 2023 iteration of CIS 5300. Finally, we would like to thank the authors of the TextAttack package (Morris et al., 2020) for providing a robust framework for us to design and evaluate adversarial attacks for sequence classification.

References

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder for english. In *Proceedings of the 2018 conference on empirical methods in natural language processing: system demonstrations*, pages 169–174.

Minhao Cheng, Jinfeng Yi, Pin-Yu Chen, Huan Zhang, and Cho-Jui Hsieh. 2020. Seq2sick: Evaluating the robustness of sequence-to-sequence models with adversarial examples. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):3601–3608.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

J. Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2017. Hotflip: White-box adversarial examples for nlp. *ArXiv*, abs/1712.06751.

Siddhant Garg and Goutham Ramakrishnan. 2020. Bae: Bert-based adversarial examples for text classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8018–8025.

Tom Kocmi and Christian Federmann. 2023. Large language models are state-of-the-art evaluators of translation quality.

Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. 2019. Textbugger: Generating adversarial text against real-world applications. In *Proceedings 2019 Network and Distributed System Security Symposium*, NDSS 2019. Internet Society.

Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. 2020. BERT-ATTACK: Adversarial attack against BERT using BERT. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6193–6202, Online. Association for Computational Linguistics.

Giosué Lo Bosco and Mattia Antonino Di Gangi. 2017. Deep learning architectures for dna sequence classification. In *Fuzzy Logic and Soft Computing Applications*, pages 162–171, Cham. Springer International Publishing.

George A. Miller. 1995. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41.

John Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. 2020. TextAttack: A framework for adversarial attacks, data augmentation, and adversarial training in NLP. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 119–126, Online. Association for Computational Linguistics.

Anh Nguyen, Jason Yosinski, and Jeff Clune. 2015. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 427–436.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Shuohang Wang and Jing Jiang. 2015. Learning natural language inference with LSTM. *CoRR*, abs/1512.08849.