Adversarial Examples for Neural Automatic Essay Scoring Systems

Klint Kanopka
Graduate School of Education
Stanford University
Stanford, CA
kkanopka@stanford.edu

David Lang
Graduate School of Education
Stanford University
Stanford, CA
dnlang86@stanford.edu

Abstract

Automatic essay scoring systems appear in increasingly important standardized testing applications. Admission to graduate school, teacher evaluations, and possibly even student grade promotion can depend on automatic essay scoring systems functioning reliably and fairly, but bad actors attempting to manipulate scores threatens the validity of decisions based on machine scored essays, therefore we set out to answer the following questions: Can adversarial attack strategies be developed for neural automatic essay scoring systems? To what extent are automatic essay scoring systems susceptible to adversarial examples? We attempt to answer these questions by presenting a series of experiments that leverage anchors, a neural network interpretation tool, as a strategy to reduce the search space for adversarial example generation in essay-length NLP input. We find, anchors can be used to identify essays with scores that are sensitive to perturbation and, given substitution into sensitive essays, anchor-based adversarial attack strategies outperform similarly constructed non-anchor-based strategies.

1 Introduction

Automatic essay scoring (AES) systems are becoming increasingly prevalent in educational contexts. For decades, the standard approach to AES tasks has been feature-engineered models, like e-Rater from ETS [2]. Individual features can be quantities like vocabulary size, essay length, average length of word, or other quantities derived from the student-supplied text. These features could then be fed into any number of predictive models, including linear regressions, ridge regressions, and feed-forward neural networks. These systems often boast higher than human reliability [9] and instant scoring once the model has been trained.

In their early work on AES systems, Page and Peterson[9] anticipated three objections to the use of AES: humanist objections, defensive objections and construct objections. The humanist objection is, simply, that machines do not understand or appreciate essays the way humans do, and computer judgments should be outright rejected. Construct objections center around what the computer is actually measuring and how (or if) they directly relate to the constructs of interest. The general response is simple: if human-machine reliability is high, then AES systems do a good job of predicting human scores, therefore they are capturing the essence of the construct of interest [3].

Defensive objections, however, consider the possibility of an attacker (or bad actor) interacting with the AES system by submitting intentionally deceptive essays with the intent of receiving an "undeserved" grade. Page and Peterson's response first posits that "motivated students eager to receive good scores" would not behave this way and ultimately suggests the retention of a human reader to flag off-topic or bizarre essays.

Classic feature-based models are susceptible to nonsense attacks as well as simple word-replacement strategies [4]. Knowing or guessing the input features can allow for the development of model-specific attacks. Neural AES systems, on the other hand, do not take input features, but instead rely on RNN and/or CNN architectures to learn important features. This adds a layer of up-front security, but if we are to learn lessons from computer vision (where the CNN is king), adversarial examples can be developed without access to systems (known as black-box attacks) [10] and can be hard to detect [5].

2 Related Work

We see our work lying at the intersection of two specific strands of neural NLP research, adversarial example generation and interpretability.

2.1 Adversarial Examples in NLP

We consider two types of adversarial examples from the NLP literature. The first is the insertion of ungrammatical text. Prior work has shown that this strategy has proven surprisingly effective at reducing model accuracy in reading comprehension models [7]. We explore this method to test model stability, but do not consider it appropriate for adversarial example generation for applications that may retain a human reader.

The other popular attack on NLP models is a random-perturbation approach. This approach goes through candidate words and replaces them either with a random token or a word that is similar in its embedding space. This approach has been effective in fooling sentiment analysis models and natural language inference tasks. One consideration is that while generating adversarial examples for tasks that generally take shorter input strings has been done [1], using a random perturbation approach for essay-length input is impractical, because the search space for essay modification is orders of magnitude larger than that of a single sentence.

2.2 Neural Network Interpretability

In order to narrow the search space identified in the previous research strand, efficiently developing strategies for adversarial attacks may benefit from attempts at understanding how the AES model is making its scoring decisions. One particularly promising model for neural network interpretability is anchors [13]. Anchors represent the minimum input required to guarantee a model delivers a particular classification. The authors highlight this approach as model-agnostic and high precision. By attempting to identify anchors for machine scored essays, we hope to gain insight into the text being used to make classification results. We propose, then, that by identifying, studying, and manipulating anchors, we may gain insight into how an AES system makes scoring decisions and what vulnerabilities may exist in the logic of those decisions.

For our purposes, we believe this methodology to have advantages over other approaches to model interpretability. Qualitative anchor analysis [13] has suggested that users who relied on anchor based models to make predictions performed more quickly and accurately than individuals who were provided with more quantitative measures, like LIME scores [12]. Because of this, we view anchors as a promising strategy for developing generalizable adversarial attacks.

3 Approach

For adversarial examples to be of concern in AES applications, we must consider that most AES systems are proprietary software, therefore we must consider adversarial examples developed through black box attacks. Therefore, to study adversarial examples in AES systems, we take the following approach: First, we construct our own AES system. This is done in service of approximating black box attacks in future work. Our model employs a convolutional neural network (CNN). We opt for a convolutional, as opposed to a recurrent neural network (RNN), because of the efficient computation of the CNN and the potential of the CNN to model relationships between non-adjacent words (which we expect to be important in essay-length input).

After developing a functional CNN-based AES system, wefound anchors for some held-out test examples. After the anchor search, we qualitatively classified the anchors, taking into consideration the scores received by each essay. Based on what we saw in the anchors, we developed strategies for targeted adversarial example construction. These strategies used the anchors to guide text injection into essays, in an attempt to induce a change in the score assigned by the model. Based upon the relative success in inducing some level of score variation, we highlighted particular strategies for further study.

After highlighting promising strategies from these initial tests, we attempted to manipulate essays for which we did not already know the anchors based upon our findings. If we can induce score variation using these strategies without previous knowledge of the anchors in a particular essay, we classify the strategy as successful.

3.1 Data

The dataset we used is from the Hewlett Foundation's Automated Student Assessment Prize (ASAP) that was released on Kaggle seven years ago. The data consists of essays written by students from grades 7-10 pulled from eight different datasets; each of the eight datasets was generated from written student responses to different writing tasks (prompts). Each essay was graded by at least two different graders on at least two different rubrics. The data includes each of the prompts, grading criteria, basic descriptive statistics, and other pertinent information. In total, there are 12,978 essays in the training set and 4,254 essays in the test set (17,232 total essays). Each essay is on one of eight possible topics. These essays range from approximately 150 words in length to 550 words.

We chose to work with a subset of the data, specifically 1,800 essays from the same 10th grade writing prompt. This was done for a two main reasons. First, each prompt is graded on a different rubric and different scale. Because our project is not focused on developing a universal AES system, score equating across different prompts is beyond the scope of what we need. Second, developing our own model on a restricted dataset approximates the black box attack development process and will lend external validity to future work on the transferability of our adversarial examples. The specific essay set we are evaluating discusses the issue of censorship in libraries. Students are asked to to write a persuasive essay articulating their view on the topic and scores range from 1-6.

3.2 Scoring Model

We first set out to build a CNN AES system based on the model proposed by Kim et al for sentence classification [8]. Beginning from a PyTorch[11] implementation of the Kim model, we found initial performance to be poor, which was to be expected, given the massive difference in input length. Because we are modeling essays and not single sentences, using kernel sizes of 3, 4, and 5 seemed unlikely to capture relevant information and structure across the entire input text. As such, we began by adding more convolutions with larger kernel sizes. The guiding theory was that we wanted to capture information not just in small groups of words, but also in entire sentences, groups of sentences, paragraphs, and groups of paragraphs. We decided to keep the original kernel sizes and also add convolutional layers of $k \in \{10, 15, 20, 50, 100, 150, 200, 350, 500\}$, based upon the idea that a single sentence is around 10 words and a paragraph is around 100 words.

During initial training, two important observations became clear. First, initial attempts to have more channels per kernel size proved to consume too much memory for the Virtual Machine (VM). Second, the model would quickly overfit the training data and perform poorly on the dev set. The original model used 100 output channels per convolution. We scaled this back to 5 output channels per convolution and found that the model would quickly overfit out training data. In these cases, loss would go to nearly zero, training accuracy would go to nearly one, and dev/test accuracy would drop to around 0.55.

To combat this, we employed the following strategies. First, the model was made smaller. Previous iterations of the model contained both more kernel sizes and more channels per kernel. Second, an aggressive value was selected for the dropout probability $(p = 0.5)$. The motivation here was to hopefully spread gradient across the different kernel sizes instead of allowing specific window sizes to dominate the prediction. Finally, we combined a conservative learning rate $(\lambda = 0.01)$ with early stopping (after 200 epochs).

The final model is built as follows:

1. An essay is read in word-by-word, the individual words converted to Word2Vec embeddings, and the embeddings are concatenated.
2. Three channel 1D convolutions are applied with kernel sizes of $k \in \{3, 4, 5, 10, 20, 50, 100, 150, 200, 350, 500\}$ words.
3. The output of each channel is passed through a ReLU nonlinearity.
4. These outputs are passed through a maxpool layer that selects the largest activation for each kernel size. During training, dropout is applied to the maxpool output.
5. The output of the maxpool layer is fed into a fully connected linear layer, with one node per possible essay score.
6. The outputs of the fully connected layer are fed through a softmax to transform them into a probability distribution, and the most likely classification is determined.

One important consideration in our hyperparameter selection was that while our investigation *needed* an AES model, our purpose was not to *develop* an AES model. As such, our model only required acceptable performance. E-rater v.2.0, an AES system that has been deployed by ETS, has exact agreement between human and machine scores between $0.50 - 0.58$, so we consider this to be the lowest acceptable accuracy [2].

3.2.1 Model Evaluation

Our primary evaluation criterion for the AES model is accuracy. Because AES is trained to produce agreement with a human scorer and exact agreement is valued in human scoring applications, this seemed most appropriate. As such, the model was trained as a classification task using cross-entropy loss. As a secondary evaluation criterion, we included mean squared error (MSE). Production-level AES systems, like those used on the GRE, often include a tolerance - additional human intervention is only triggered if the difference between a human score and the AES score is beyond that tolerance. Because of that, we implemented MSE as a posterior check to ensure when our AES system was not predicting the human score precisely, it was at least close.

Our model achieves test accuracy of 0.867 with a MSE of 0.150. Because we have surpassed the previously mentioned accuracy benchmark, we are satisfied with the accuracy of the model. Because $MSE \approx 1 - Accuracy$, we can also conclude that most essays that are misclassified are within 1 point of the human score, so we consider the model acceptable for our purposes and proceed to identifying anchors. As an aside, this level of performance is considered reasonable for many forms of human rating and evaluation in education. For instance, classroom observation protocols require within one scale-point agreement[6] and additional human raters are not brought in to examine a GRE essay unless the first human rater and the AES disagree by more than one point.

3.3 Adversarial Attack Strategies

In attempting to evaluate the susceptibility of our model to adversarial examples, we set out to systematically examine the stability of the scoring model, search for potential attack strategies by identifying anchors in essays, and evaluate the efficacy of these anchor-based attack strategies. Experiments detailing this approach are presented below.

4 Experiments

To begin building a picture of adversarial examples in AES, we broke our experiments into three categories.

4.1 Identifying Anchor-Based Attack Strategies

For our approach to be tractable, we need to evaluate anchors in essays. These anchors can then function as a window into how the model makes decisions based upon particular portions of the input.

Anchor	Score	Precision
perhaps	4	.91
shelves AND understanding AND The AND Offended	4	.80
censorship AND Books AND The AND content	4	.80
freedom	4	.74

Table 1: Candidate anchors

4.1.1 Experimental Details

The anchor search method uses two main parameters. The first, precision, identifies the level of confidence the model needs to have in a particular anchor in order to report it. The second defines the perturbation distribution used to conduct the search. We elected to use the UNK distribution, where words in our essay were replaced with out of vocabulary UNK tokens. The other option replaces words with similar words (in the embedding space). We elected to use the UNK distribution because it would reduce the search space for our problem and provide more anchors than the similar word-based distribution.

Our initial tests were done with precision ($p = 0.95$). This yielded no results after searching through the first 20 essays, so the precision parameter was relaxed to $p = 0.70$, with the rationale being we could hand-sort anchors later depending on the size of the yield in our search. This method, as expected, produced more anchors.

Here we note one enormous problem with the anchor method as we applied it to essays: it is slow. Because each word in an essay can be replaced with an UNK token, an essay of length W words produces a search space of 2^W possible essays. The implementation uses beam search to speed this up somewhat, but this still makes vanilla anchors a computationally expensive approach to essay-length inputs. While the initial code was running, multiple workarounds were discussed, including grouping words into phrases or sentences for the anchor search to operate on, but these ideas were abandoned when the first candidate anchor emerged.

4.1.2 Candidate Anchors

The anchor search, as implemented, can process approximately ten essays per compute-day. The first essay processed, however, managed to produce a single-word anchor ("perhaps") that was key in an essay receiving a high score (4) with high precision ($p = 0.91$). Because we viewed this as an ideal candidate for developing an attack strategy, we made the decision to abandon plans for efficiency gains and process fewer essays at a finer grain size. This decision is a tradeoff, the advantages and consequences of which will be discussed later.

4.1.3 Experimental Results

Table 1 shows four identified candidate anchors. Because a CNN model (by design) cares not only about what words are present, but also their ordering and relative position, multiple word anchors could be more challenging to implement as attacks and also instrument the additional variable of "relative insertion position" for any attacks we would implement. As such, we elected to focus on the single-word anchors "perhaps" and "freedom" for our subsequent analysis.

4.2 Model Stability

Experiments investigating model stability were designed to see the extent to which scores produced by the AES model are stable under input perturbation. Based in part upon the two focal anchors we selected, four types of perturbations were identified to investigate scoring stability. These attacks are:

- **Shuffling**- Words in an essay are randomly shuffled. One benefit of this type of attack is that models that solely relied on a bag-of-words should be invariant to this type of manipulation.
- **Appending**- A target word is appended to the end of an essay with no other modifications.
- **Substitution**- An word in the essay is randomly replaced with a target word with no other modifications.

- **Insertion**- A target word is randomly inserted into an essay with no other modifications. We view this as distinct from a substitution attack, because it fundamentally changes the distance between words on either side of the insertion, potentially making predictions based heavily on convolutions of larger kernel sizes more unstable across perturbation.

A visualization of these attacks can be seeing in Figure 3 in the appendix. This example perturbs the essays with the word 'maybe'.

4.2.1 Experimental Details

In order to carry out these four initial attacks, the following procedure was followed: First a set of essays was identified as the base for the perturbations. Next, a single shuffling perturbation was generated for each essay. Next, target words were identified for appending, substitution, and insertion attacks. Selected for these attacks were the two anchors ("perhaps" and "freedom") as well as synonyms ("maybe," "probably," "arguably," and "indeed"). A perturbed set of essays was generated by applying applying each perturbation once to the base set of essays for each target word. It is important to note that these tests are only equipped to capture overwhelmingly dominant essay perturbation strategies. More specific explorations will be addressed later.

4.2.2 Results

Unsurprisingly, the attack that had the strongest impact on output scores from our model was the shuffling attack. Table 2 shows our original model's prediction and its predictions on a perturbed version of the essay. There are several findings of note. First, this type of perturbation seems to eliminate the possibility of any essay achieving the highest score on the rubric. This can be viewed as a regression to the mean effect - producing heterogeneity on who could benefit from this type of perturbation. Realistically, however, This type of attack is impractical for two reasons. First, it is hard to write an essay of low quality, randomly rearrange the words, and then submit it under time pressure. Second, and more importantly, a human reader would flag an essay manipulated in such a way as anomalous with a high probability.

Results from appending, substitution, and insertion attacks are listed in table 3. Here we see these attacks generally result in no change for respondents in the middle of the score distribution or lower, while often harming respondents near the top of the score distribution. In addition, we see that appending generally has no effect on scoring. From this, we conclude that the scoring model is generally stable under these types of perturbations.

4.3 Evaluating Anchor-Based Attack Strategies

Based upon the previous results, we attempted to isolate individual essays that may be more susceptible to score manipulation via perturbation. To explore this, we first noted that from the previous experiments, the only essays that had improved scores were essays graded a 1 or 2. Because of this, we highlighted a set of five essays that saw score improvements from random perturbation with the target words of "perhaps" and "freedom." We also randomly selected control essays and included the control words "maybe" and "the," in addition to the targeted anchors.

Figure 3 depicts the specific type of attack.

4.3.1 Experimental Details

For each essay, we used two specific attacks. The first we refer to as the "progressive overload" strategy. Under this attack, words in the essay are sequentially replaced with the target word in a way that the first example has the first word replaced with the target word, the second example has the first *two* words replaced with the target, the third example has the first *three* words replaced, and so on. The purpose of this experiment is to identify the susceptibility of the model to "spamming" a particular word.

The second attack we call the "single subsitution" strategy. It is designed to test the model's sensitivity to the location of a perturbation within a particular essay. This is done similar to the "progressive overload" attack, except the first word replaced with the target word, the second example has the *only* the second word replaced with the target, the third example has *only* the third word replaced, and so on.

progressive_overload.png

Figure 1: Note the fuzzy boundaries between higher and lower score under perturbation.

4.3.2 Results

Figure 1 shows a graph typical of the results for this attack. Here, essay 41 is overloaded with the anchor “perhaps.” This example is particularly interesting because it switches quickly from low score to high score as the anchor is overloaded in the front of the essay.

Recognizing that this attack is not practical to deploy “in the wild,” we still note that, across our limited examples, anchor words transition to higher score states sooner than non-anchors, and anchors tend to be associated with higher mean scores across the length of the essay than control words.

Looking to single substitutions, we see a slightly different picture. Table 4 shows the change in mean score for four essays (two targeted for sensitivity, two randomly selected controls) across all perturbations. We see “freedom” perturbations have the largest score difference in target essays, followed by “perhaps,” and then “the.” Control essays have relatively stable scores under perturbation.

Figure 2 shows score as a function of substitution position for “perhaps” in essay 41. Compared to the plot from the progressive overload attack, we see a lack of clear boundaries between low and high scores and a sensitive dependence on insertion location. This lack of clear boundary is typical for all tested substitution attacks, except in the cases where the perturbation has no effect independent of position in the essay.

5 Analysis

The results of these experiments emphasize one major hurdle in generating adversarial examples from essay-length NLP inputs: the search space is massive. Experiments here were limited by compute time, but it is important to note that, at the very least, anchors lead us to two important results. The first is *candidate* perturbations, which restrict the search space in some dimension. Secondly, the process we describe of anchor search followed by wide perturbation across a large data set has the potential to identify individual essays that may be especially sensitive to minor perturbation. These two, in tandem, can identify candidate modifications and candidate essays to serve as the basis for adversarial example search.

We also note that our relatively naive attack strategies were more effective when using anchor-derived words when compared to synonyms or common words. Strangely, we also see that that substituting “the” has measurable effect on our sensitive essays. This could, potentially, be due to the *removal* of words that are important to model decisions, a strategy we did not systematically explore. We recognize potential in our approach for reducing the search space when crafting adversarial

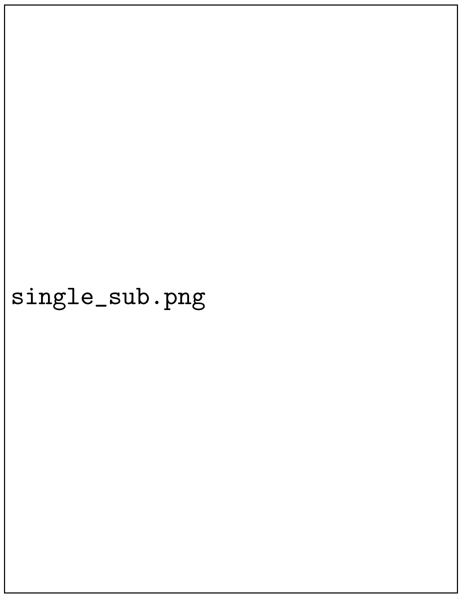

Figure 2: Note the lack of clear high and low score regions.

examples on essay-length inputs, and therefore see this as an incremental step toward future work and a clear demonstration that classifications for essay-length inputs can be manipulated with single word perturbations.

6 Future Work

We identify four main areas for future work. First, improving the efficiency of anchor searches over essay length inputs will be key to further exploration. Second, exploring more complex perturbation strategies over a wider variety of inputs will help to determine the extent to which the anchors we have already identified are promising seeds for adversarial attacks. Third, looking at more anchors will allow us to develop a more robust set of attack strategies, generalizing beyond single-word based perturbations. Finally, investigating the extent to which any successful strategies generalize across model architectures will be important to the the adoption of neural AES systems.

7 Conclusion

We presented a framework for adversarial example search and development. Using anchors, we identified candidate injection text that had the potential to induce score variance. Anchors also help to select candidate essays with scores that are sensitive to text manipulation. We identified several perturbation strategies and explored their associated effects with both identified anchors and control phrases. We find, in general, that anchor-based attack strategies outperform similarly crafted attack strategies that ignore anchors. We also find that the success of an adversarial perturbation strategy is sensitive to not only the specific essay being perturbed, but also the location in which the target text is introduced into the essay. Because of this, appending text is seen to constitute a generally useless strategy, especially when compared to insertion or substitution. Single word substitution, in our experiments, did not induce large variations in score, but still has, in some cases, the potential to induce essay misclassification that a human scorer would not notice with high probability.

Key limitations of our work are the speed of anchor searchers and the scale of our dataset. Future work could be improved with additional optimization and compute time to determine a larger array of candidate anchors. In addition, a larger set of essays and more compute time would allow for an expanded adversarial example search and potentially fewer problems with overfitting during AES model training.

Original Model Score	1	2	3	4	Total
Shuffled Score	%	%	%	%	%
1	52.6	0.8	0.0	0.0	3.2
2	47.4	79.1	16.2	0.0	46.5
3	0.0	20.1	83.8	100.0	50.3
Total %	100.0	100.0	100.0	100.0	100.0

Table 2: Shuffled Results

Perturbation	1	2	3	4
app_perhaps	0.011	-0.001	-0.001	0.000
ins_perhaps	0.063	0.006	-0.012	-0.512
sub_perhaps	0.095	0.006	-0.008	-0.465
app_freedom	0.000	0.006	0.000	0.000
ins_freedom	0.137	0.010	-0.010	-0.674
sub_freedom	0.126	0.005	-0.006	-0.581
app_maybe	0.011	-0.004	-0.006	0.000
ins_maybe	0.095	0.001	-0.013	-0.419
sub_maybe	0.105	-0.001	-0.002	-0.488
app_arguably	0.000	0.001	0.000	0.000
ins_arguably	0.116	0.004	-0.007	-0.674
sub_arguably	0.126	0.000	-0.006	-0.628
app_indeed	0.000	0.000	0.000	0.000
ins_indeed	0.116	0.005	-0.010	-0.512
sub_indeed	0.105	0.007	-0.012	-0.465
app_probably	0.000	0.001	-0.002	0.000
ins_probably	0.105	0.000	-0.008	-0.651
sub_probably	0.126	0.000	-0.008	-0.605

Table 3: Mean change in essay score under perturbation, conditional on original score

8 Appendix

Our project mentor was Amita Kamath. AJ Alvero, who was originally in our project group, has withdrawn from CS224N.

Our code is based on two existing open-source repositories:

- Our core AES system is based on a PyTorch [11] implementation of the "Convolutional neural networks for sentence classification" model [8].
  `https://github.com/galsang/CNN-sentence-classification-pytorch`
- The anchor code is published by the authors of the original paper [13].
  `https://github.com/marcotcr/anchor`

References

[1] Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. Generating natural language adversarial examples. *arXiv preprint arXiv:1804.07998*, 2018.

Substitution	**Mean Δ Score (Sensitive)**	**Mean Δ Score (Control)**
Perhaps*	0.110	0.000
Freedom*	0.341	0.000
Maybe	0.030	0.000
The	0.093	0.000

Table 4: Mean change in essay score under perturbation. Note the distinction between essays flagged as sensitive to perturbation and control essays. Words denoted with * are identified anchors.

`AttackTypes.JPG`

Figure 3: Perturbation Examples
s

[2] Yigal Attali and Jill Burstein. Automated essay scoring with e-rater® v. 2.0. *ETS Research Report Series*, 2004(2):i–21, 2004.

[3] Yigal Attali and Jill Burstein. Automated essay scoring with e-rater® v. 2. *The Journal of Technology, Learning and Assessment*, 4(3), 2006.

[4] Isaac I Bejar, Michael Flor, Yoko Futagi, and Chaintanya Ramineni. On the vulnerability of automated scoring to construct-irrelevant response strategies (cirs): An illustration. *Assessing Writing*, 22:48–59, 2014.

[5] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 3–14. ACM, 2017.

[6] Heather C Hill, Charalambos Y Charalambous, and Matthew A Kraft. When rater reliability is not enough: Teacher observation systems and a case for the generalizability study. *Educational Researcher*, 41(2):56–64, 2012.

[7] Robin Jia and Percy Liang. Adversarial examples for evaluating reading comprehension systems. *CoRR*, abs/1707.07328, 2017.

[8] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[9] Ellis B Page and Nancy S Petersen. The computer moves into essay grading: Updating the ancient test. *Phi delta kappan*, 76(7):561, 1995.

[10] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pages 506–519. ACM, 2017.

[11] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[12] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144. ACM, 2016.

[13] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. In *AAAI Conference on Artificial Intelligence*, 2018.