Don't take "nswvtnvakgxpm" for an answer – The surprising vulnerability of automatic content scoring systems to adversarial input

Yuning Ding[1] **Brian Riordan**[2] **Andrea Horbach**[1] **Aoife Cahill**[2] **Torsten Zesch**[1]

[1]Language Technology Lab, University of Duisburg-Essen, Duisburg, Germany

[2]Educational Testing Service, Princeton, NJ 08541, USA

Abstract

Automatic content scoring systems are widely used on short answer tasks to save human effort. However, the use of these systems can invite cheating strategies, such as students writing irrelevant answers in the hopes of gaining at least partial credit. We generate adversarial answers for benchmark content scoring datasets based on different methods of increasing sophistication and show that even simple methods lead to a surprising decrease in content scoring performance. As an extreme example, up to 60% of adversarial answers generated from random shuffling of words in real answers are accepted by a state-of-the-art scoring system. In addition to analyzing the vulnerabilities of content scoring systems, we examine countermeasures such as adversarial training and show that these measures improve system robustness against adversarial answers considerably but do not suffice to completely solve the problem.

1 Introduction

Automatic content scoring (Ziai et al., 2012) is an educational task where free-text answers to a content question are automatically scored based on the conceptual correctness of the answers. Consider for example the question *What is the favorite food of pandas?* with an expected answer such as *Pandas eat bamboo*. A content scoring system which only accepted this exact same answer would be very limited, as there are many other ways to express the same fact, e.g. *Pandas like to eat bamboo*, *Pandas feed on bamboo*, or simply *bamboo*. In addition, students might make all kinds of spelling and grammatical mistakes, which are usually ignored in content scoring as long as they do not interfere with understandability of the answer. Thus, *Pendas are eating bambboo* would still be acceptable. A content scoring system must therefore be able to generalize beyond a certain set of a prior known correct answers, e.g. by taking variance in spelling (Leacock and Chodorow, 2003) and from other sources into account (Horbach and Zesch, 2019). For example, many supervised scoring systems rely on character-level n-gram features (Heilman and Madnani, 2013; Zesch et al., 2015). Without those features, the system would not be able to detect that *bambboo* is likely to mean *bamboo* as they both e.g. contain the trigrams 'bam' and 'amb'. In fact, such systems show remarkable resilience against spelling errors, where even introducing up to 50% spelling errors in each answer does not significantly influence scoring performance (Horbach et al., 2017).

This ability of a system to generalize beyond previously seen examples makes it potentially vulnerable to adversarial behaviour. In the educational domain such behaviour has long been known under the term *cheating*. When students have no idea about a task, they could, for example, try to write an answer combining several potential answers, such as *banana bamboo carrots grass*, in the hope to accidentally include the right one. While this is rather unlikely to convince a teacher, if the students know that they are being scored by a system they might take their chances.

Publicly available content scoring systems in the Automated Student Assessment Prize (ASAP-SAS)[1] have been shown to be vulnerable to such answers generated with cheating strategies like repeating the

[1]http://www.kaggle.com/c/asap-sas

answer for multiple times and adding question-related or general academic words (Higgins and Heilman, 2014). What if the students don't need any strategies at all, but trick the systems with only some extreme irrelevant adversarial input?

In this paper, we aim to investigate whether the need for generalization abilities (due to language errors and many ways to express the same fact) makes content scoring systems vulnerable to adversarial attacks. For this purpose, we generate adversarial answers based on different methods of increasing sophistication. We find that both shallow and deep learning scoring models are surprisingly vulnerable. Even with simple techniques, up to 60% of generated wrong answers are accepted by the systems as being at least partially correct. This includes extreme examples where a string of random characters like "nswvtnvakgxpm" (also referenced in the title of this paper) was accepted as a fully correct answer. The adversarial is accepted as it contains some predictive, but non-robust feature (Ilyas et al., 2019) like the character bigram 'kg'.

As such system behavior is threatening the validity of the scoring system, we explore countermeasures to make models more robust. We find that removing vulnerable features like character n-grams and adversarial training can improve the robustness of shallow systems against adversarial answers. However, these countermeasures often come at the cost of a performance decrease on real student answers.

This paper aims to bring the importance of system robustness against adversarial inputs in content scoring to the forefront. We also make our corpus of adversarial answers as well as their generation methods publicly available for future research.

2 Generating Adversarial Answers

Adversarial examples are instances with intentional perturbations that cause a machine learning model to make a false prediction (Goodfellow et al., 2015). Many prominent examples come from the image processing domain, e.g. a stop sign that was perturbed with little patches so that autonomous cars would classify it not as a stop sign (and possibly causing severe accidents). Szegedy et al. (2015) shows that GoogLeNets classification of an image as 'panda' can be changed into 'gibbon' by adding a small vector. This kind of perturbation is usually imperceptible to humans, but decreases the performance of classification models drastically.

Following the similar principle in NLP applications, HotFlip (Ebrahimi et al., 2017) generates textual adversarial examples by swapping one character with another by gradient computation, and Textbugger (Li et al., 2018) by inserting spaces or swapping random letters. These methods are quite likely to leave the semantics of the underlying text unchanged, which is much harder to achieve when texts are manipulated on the token level. A common strategy is to replace single words (usually nouns) with near-synonyms chosen by humans (Kuleshov et al., 2018) or as nearest neighbors in an embedding space (Pennington et al., 2014). At the sentence level, SCPNs (Iyyer et al., 2018) produce a paraphrase of a given sentence without changing the original meaning. In the educational domain, this is comparable to unusual, atypical or creative correct answers that run the risk of being classified as incorrect (Yoon et al., 2018).

However, none of the generation methods above is suitable to our research. If we use a real, correct answer as starting point for transformation into an adversarial one and leave the semantic meaning unchanged, we generate a correct answer. But we need answers which are definitely wrong in content as adversarial answers, in order to test the system's ability of rejecting cheating behaviour.

We start with basic methods like uniform selection of characters and words from a generic corpus, but also use methods that are specific to the educational task like shuffling words in correct answers. The nonsense answer, generated by the most basic methods, are treated as a "worst case scenario".

2.1 Generic vs. Prompt-specific

Generation methods using prompt-related material are more likely to resemble the actual cheating behaviour of students who might resolve to writing something that is on topic in terms of vocabulary, although they have no idea how to answer the question correctly.

In order to reflect this dichotomy (general purpose vs. task-specific adversarial answers), we use two different corpora in the generation process, especially using N-gram generation method introduced later. As a generic language model, we use the Brown Corpus (Francis and Kucera, 1979). To generate

Method		Example
Random characters		fcwowtpmqalwkjxldrldvc bw fhgkter
Generic corpus characters	1	b er erol ewhcsflasa t ro hngeonhhroci ard tel et tssnbrriehtrryva ethwul o t myaktoiaxao ne a o lae
	2	3 owngirconiity noonpangstisy fo te dsom s piee ifuterst vof hued anphs thred d theslathsea
	3	tab puteloulht e t asvilichfilfrolicg fu mrthrt nthe cbut frmesncrf d lot oof n gr r vianda bashf an
	4	ustt spagramem nldinf threchreadpoetexacvatehe faks w heuse ther towral o ne be edicoll aw
	5	dunlssed ayed an oumr kehese ows aant maw itis apsed jis prlease a prd on ally everty
Prompt-specific characters	1	zka lhdh t ngipeb ninsmoaeehu aaslyieh ltr eeatry d an ep ba wenpiyon
	2	pd falstspate cahesis th fuc bla cfot ey as l sngko wedals lile krake suc buss s earae s alngeyaly
	3	w bre s ach n ie kts thogron iar pytpyta bausmos innlithos as we landin us the calisre oo
	4	nothke fthon thechaney adas eatats difear gereleav can exc theey at pacaly difliveable
	5	ls if aust bearrom pre ma sameis a e difhem tna splas i but koal matelowednythiment
Random words		footage flubbed birthplace parry's cicadas
Generic corpus words	1	arabian but the aid consultation backed fussing it if was of for un to be the out addresses a and firing
	2	terribly in and contrasting if the arrive at without court they encouraged sterling honesty
	3	a line of then perhaps one calculation in balancing greater the range i wanted to ended
	4	maid was instructed in she would try to our religious philosophy a night in ashes to off our coast
	5	vigor in the late nineteenth older the nature of time derive second and higher order real wage rate
Prompt-specific words	1	vast are of in eat bear they leaves a or similar bears there article makes can
	2	swallow one they are australia because almost only the last thing these that all bear which
	3	environment however pythons in australia because support their diet panda or koala python
	4	share the most is koalas basically just eat are similar to koala eats eucalyptus leaves almost
	5	the other hand can eat that pandas and koalas are specialist in the article it generalists
Content Burst		panda eat bamboo koala eucalyptus python America need fact comparison resource people
Shuffle		bamboo eucalyptus resources eats koalas need eat anything panda but and as doesn't the America...
GPT		which is the same as eating a human. So the panda has to eat bamboo to be able to consume bamboo...

Table 2: Examples of generated adversarials. The prompt-specific adversarials are based on the answers for prompt 3: *Explain how pandas in China are similar to koalas in Australia and how they both are different from pythons*.

prompt-specific adversarial answers, we use the student answers in the ASAP short-answer scoring dataset (ASAP-SAS). This dataset contains ten short-answer prompts covering different subjects and a high number of answers per prompt as shown in Table 1.

Let's consider prompt 3 from ASAP-SAS as an example: *Explain how pandas in China are similar to koalas in Australia and how they both are different from pythons*. Words like "panda" and "koala" are likely to occur frequently in real student answers while, in contrast, a word like "cat" probably will not. Prompt-specific words have a higher frequency in the ASAP-SAS corpus, while all three words have a similar distribution in the Brown corpus. Thus, we hypothesize that an adversarial generated on the basis of the ASAP-SAS corpus will more likely to be accepted as a correct answer than an adversarial created using the Brown corpus.

		#Answers		Score
	Topic	Train	Test	Range
1	Chemistry	1,672	557	0-3
2	Chemistry	1,278	426	0-3
3	English - Reading Compreh.	1,808	406	0-2
4	English - Reading Compreh.	1,657	295	0-2
5	Biology	1,795	598	0-3
6	Biology	1,797	599	0-3
7	English - Language Arts	1,799	599	0-2
8	English - Language Arts	1,799	599	0-2
9	English - Structure Descrip.	1,798	599	0-2
10	Physics	1,640	546	0-2

Table 1: Overview of the ASAP-SAS dataset.

2.2 Generation Methods

In the following, we explain our methods used for generating adversarial answers. Table 2 shows examples for adversarial examples generated with the different methods. From the word-based adversarials, we can see that the adversarials generated from the prompt-specific corpus are more similar to actual student answers than the adversarials generated from the generic corpus.

Random A completely random generation of answers is of course not something a student is likely to do, but we consider it as a baseline. No robust content scoring system should classify this kind of data as correct. For character-based sampling, we uniformly draw individual characters (including whitespace) until an answer of the same length as the average in the ASAP-SAS data set is reached (without punctuation). While for character-based generation it is quite clear from which alphabet we have to sample, the choice of dictionary matters for word-based generation. Since the Brown Corpus is comprised of one million words drawn from a wide variety of sources, we sample uniformly from the Brown Corpus vocabulary.

N-gram For each n between 1 to 5, we generate adversarial answers by performing a weighted random sampling based on the frequency distribution over all n-grams in the corpus. The generation process for one answer ends when its length exceeds a predefined maximal length or the last selected n-gram contains the the end-of-sentence token.

Note that this approach is not classical n-gram language model generation, since the frequency of prior words in the generated sequence are not used in the generation of the next word.[2] Also note that the random words condition is not equal to a word unigram condition, as random words are selected from the underlying corpus with uniform probability, while 1-grams are generated according to their corpus probability.

Content Burst In our content scoring scenario, we assume that content words play a more important role than function words. A possible student cheating behavior to take advantage of a scoring system that emphasizes content words might involve randomly listing content words related to a prompt. We restrict ourselves to generating adversarial answers from all nouns in a prompt instead of all possible content word classes to avoid accidentally generating something comprehensible and hence also acceptable to a human rater. We use the NLTK part-of-speech tagger to collect all nouns in the prompt source material and student answers in the ASAP-SAS dataset. We then generate adversarial answers by sampling from the frequency-weighted set of nouns. This is equivalent to a unigram model if we reduce ASAP to nouns.

Shuffle A content scoring system should consider the order and syntactic structure of words, not just which words occur in an answer. For example, continuing the example of ASAP-SAS prompt 3, *Pandas like to eat bamboo* should be accepted as a correct answer while *bamboo Pandas eat like to* should not, even though both answers contain the same words.

To generate adversarial answers, we select correct answers from the ASAP-SAS dataset and randomly shuffle word tokens in each answer. The generated set of adversarial answers has exactly the same vocabulary as the source answers, but each answer is syntactically incorrect.

Generative Language Model To test if cohesive sentences without correct content can be used as adversarials, we investigated using a generative language model, GPT-2 (Radford et al., 2018; Radford et al., 2019). A language model serves to model a cheating strategy in which a student performs an Internet search with key words from the prompt and copies random sentences from the search results into their answer. For example, following the seed *The panda eats only bamboo* and specifying a length of 41 words, GPT-2 produces an adversarial example as shown in Table 2.

To avoid accidental generation of correct answers, we did not train a task-specific transformer model and instead used the PyTorch implementation of GPT-2 by Tae Hwan Jung.[3] We use the first 5% of words in ASAP-SAS answers as seeds along with the desired length of the generated answer, which corresponds to the length of the original answer for that seed.

3 Experimental Setup

To evaluate the robustness of automatic content scoring systems, we used the original training data in ASAP-SAS to train several scoring models and tested each model on the generated adversarial examples

[2]In pilot experiments, we trained classical n-gram models on the ASAP-SAS data and generated sentences by predicting the conditional probability of the next word, but this led to many generated answers with correct content, which could be used as adversarial examples for this investigation.

[3]https://github.com/graykode/gpt-2-Pytorch

as described in the previous section.

3.1 Scoring Systems

Automatic scoring systems can be categorized into **shallow** and **deep** learning systems (Collobert and Weston, 2008). For our study, we focused on 'typical' systems that represent what is usually applied in practice – so we can find typical problems – instead of highly optimized systems whose vulnerabilities to adversarial input might be highly idiosyncratic.

As a representative state-of-the-art shallow system, we selected the ESCRITO scoring toolkit (Zesch and Horbach, 2018) with an SVM classifier (Cortes and Vapnik, 1995), as implemented in Weka using the default PolyKernel. As features we used the top 10000 character 2-5 grams, the top 10000 word 1-5 grams, and answer length.

As a deep learning system, we employed the RNN-based system described in Riordan et al. (2019). It uses pretrained word embeddings encoded by a single layer 250-dimensional bidirectional GRU. The hidden states of the GRU are aggregated by a max pooling mechanism. The output of the encoder is aggregated in a fully-connected feedforward layer with sigmoid activation that computes a scalar output for the predicted score. Characters are encoded with a sequence of 25-dimensional character embeddings (randomly initialized) followed by a convolutional neural network (100 filters and filter sizes of (3,4,5)). The character embeddings are concatenated with the word embeddings prior to the word-level encoder.

3.2 Evaluation

In this work, we evaluate the content scoring systems with two metrics: First, we use the **Adversarial Rejection Rate** (ARR), which measures what percentage of the adversarial input is rejected, i.e. scored as zero by the system. Second, **Quadratically Weighted Kappa** (QWK) is used to measure the performance of systems on real answers, as it does not only consider whether an answer is classified correctly or not, but also how far it is from the gold classification (Cohen, 1968).

An ideal content scoring system should have a high ARR as well as a high QWK. The trade-off between these two metrics is important. Both our target systems have a state-of-art performance on ASAP-SAS data set. The shallow system has a QWK of 67.3%, while the deep one of 77.05% under the best-performing parameter setting.

3.3 Corpus of Adversarial Answers

Using the methods described in Section 2, we obtain a corpus comprising 25,000 adversarial answers, 1,000 answers for each method. The data as well as the generation code are publicly available for research purposes under `https://github.com/ltl-ude/adversarials`.

4 Results

In this section, we apply the shallow and deep scoring systems to the adversarial examples. A perfect system should reject them with 100% ARR, but Figure 1 shows that this is not the case at all. On average, the shallow system can reject around 77% adversarial examples, while the deep system has an adversarial rejection rate of only 53%. On the same type of adversarial examples, the shallow system is more robust against adversarial answers than the deep one (the yellow diamond is always above the blue one).

Comparing the adversarial examples generated with character/word n-gram from the generic and the prompt-specific corpus, we find that both systems have a lower ARR on adversarial answers generated on the basis of the ASAP-SAS corpus. This supports our hypothesis, that adversarial answers with more prompt-specific words are more easily accepted. Examples generated by shuffling have the lowest average ARR on the shallow system, presumably because all the lexical material of a correct answer is still present in the corresponding adversarial example.

If we look at the ARR of adversarial examples generated with prompt-specific character n-gram combinations, increasing n is associated with decreasing performance for both systems. Because examples generated with a larger n contain more possible real words, the models may treat these tokens as features indicative of correct answers.

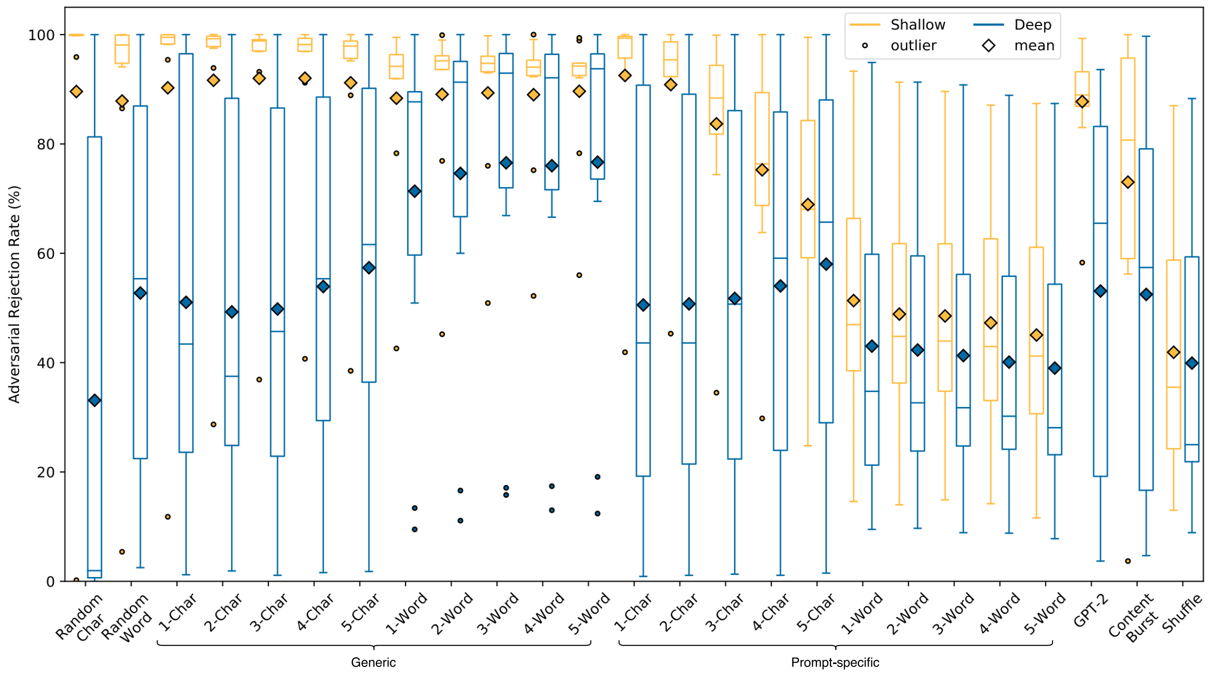


Figure 1: Adversarial Rejection Rate of shallow and deep systems.

We are surprised to find that even random combinations of characters and words can be accepted as correct answers in some vulnerable prompts. The shallow system's model for prompt 2 is the most vulnerable model (shown as the yellow point at the bottom of the graph), while the deep system has the worst performance on prompt 10 (blue outlier). They both misclassify more than 90% random character answers as correct.

To investigate the reason for the systems' vulnerabilities, we looked into the shallow system's worst performing model (on prompt 2), which scored more than 90% of adversarial answers generated with random characters as correct. An example adversarial response is *nswvtnvakgxpm*. Analyzing the feature representations, we found that character bigrams "ns", "nv" and "kg" are the top 3 features found in the adversarial answers misclassified with the highest score. These frequent character bigrams are likely found in words related to correct answers in the training data such as "co$\underline{\text{ns}}$tant", or "i$\underline{\text{nv}}$estigation", while "kg" is an important word in itself.

5 Countermeasures

We have seen that the content scoring systems used in the previous experiments are indeed susceptible to the adversarial examples generated with even simple methods. This section thus investigates countermeasures to make the systems more robust against such attacks.

We consider three types of countermeasures: First, we explore the *inclusion* of features that we hypothesize to be particularly robust against adversarial asnwers and the *exclusion* of certain feature groups that we found to be particularly susceptible to adversarial answers. Second, we enrich our training datasets in a data augmentation approach to include adversarial training examples. Third, we use a filtering approach where we flag potential adversarial answers for human inspection in a separate module before the actual scoring.

5.1 Select features with respect to robustness

When experimenting with different feature sets, we either add features aimed at increasing robustness or remove features that are particularly vulnerable to adversarial answers.

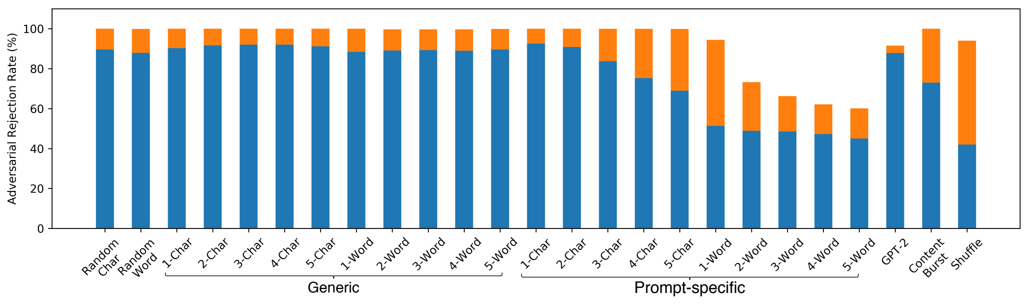
Add syntactic features For the shallow system, we add features related to the sentence structure with the goal of increasing robustness against adversarial answers that lack syntactic structure.

Dependency features. In dependency trees, each word in a sentence constitutes a node in a parse tree. By training the model with the top 1000 most frequent dependency triples from the training data as additional features, the model can learn sentence structure that is useful for classification. Examples of dependency triples from the ASAP-SAS prompt 3 dataset are: "koala <conjunct>panda" or "species <adjectival modifier>specialist" (both extracted from the sentence *Pandas and koalas are both similar because they are both specialist species*). Averaged across all prompts, the models trained with dependency features have only slightly better performance (less than 2% increase of ARR) compared to the baseline. The ARR improvement on adversarial examples generated by GPT-2 is less than for the other adversarial types, because the large pre-trained model can generate sentences with correct syntactic structure.

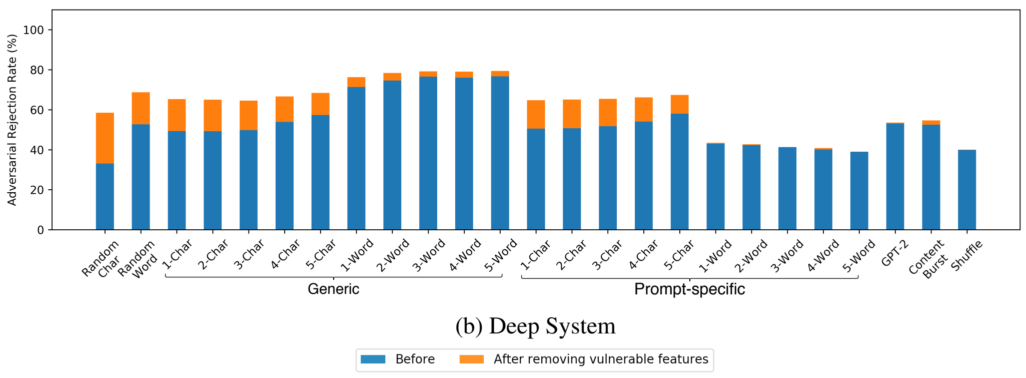
Part of Speech (POS) We also experimented with adding POS n-grams as additional features. However, model robustness was not improved as expected, and even had the opposite effect on examples generated from the general corpus.

Remove vulnerable features For the shallow system, we identified character n-grams and word unigrams as the most vulnerable features. Thus, we remove these features in a new set of experiments. As shown in Figure 2, models trained with only word 2-5grams and sentence length as features reach 100% accuracy on most adversarial answers. Less than 40% of adversarial answers generated with word n-grams are still accepted.

In comparison, we remove the character representation of the deep system, it also leads to about 15% performance increasing on adversarial answers generated with character n-grams and random word. However, it still shows great vulnerability on most adversarial answers.



(a) Shallow System



(b) Deep System

Figure 2: ARR of shallow and deep systems before/after removing vulnerable features.

5.2 Adversarial Training

Using adversarial examples to augment training data has been shown to provide robustness against adversarial examples in image recognition (Szegedy et al., 2013). Following this method, we enrich our training data with 1000 instances created using the same methods as for our adversarial examples.

Adding augmentation data generated with one certain method can improve the model robustness on different types of adversarial examples. Therefore, Figure 3 shows the effect of augmenting the training data with different kinds of adversarial answers on different kinds of adversarial test data. We see, for example, that adding augmentation instances generated by shuffling improved average performance of the shallow system across all adversarial test data.

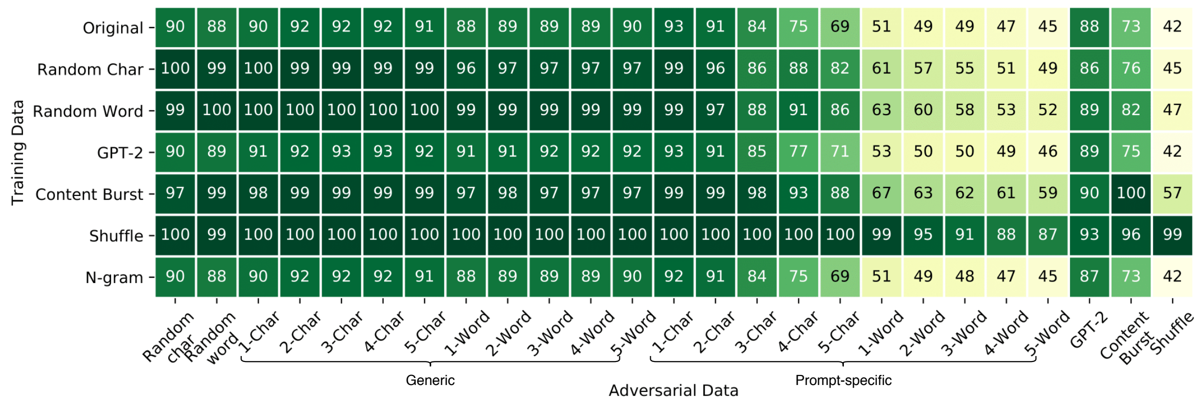


Figure 3: ARR of the shallow system trained on adversarial augmented data.

5.3 Comparison of Countermeasures

As mentioned above, an increased robustness against adversarial attacks usually comes at the cost of a loss of robustness against other deviations from standard language, such as unusual orthography and grammatical problems in an answer. Such answers, however, which are correct content-wise but lacking in form, are usually those a content-scoring system should still accept. This means it is of course straightforward to create a system that rejects almost all adversarial answers, e.g., by using only word and no character n-grams, but doing so would exclude answers with a high number of spelling errors.

Thus, robustness against adversarial answers (i.e. excluding them) and robustness against deviations in linguistic form (including them) is a trade-off that has to be carefully balanced in order not to discriminate against certain user groups.

Figure 4 quantifies this trade-off for each countermeasure with the shallow system. Adversarial training with augmented data generated by shuffling is the best countermeasure for shallow system, helping the model to reject most of the adversarial examples and hurting the performance of the model on the original test data the least.

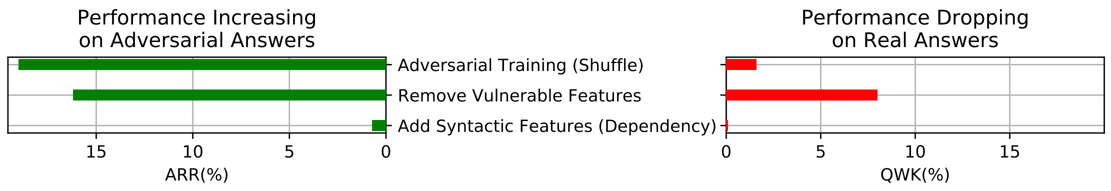


Figure 4: Trade-off of countermeasures on shallow system.

5.4 Filtering

In the filtering approach we aim to identify potential adversarial answers before the actual classification process. Our rationale is that in many educational scenarios, cheating is a risk and thus comes at a considerable cost for the student. Therefore, we assume that adversarial answers are the exception rather than the rule and a human teacher can manually check potential cheating candidates, as long as the false alarm rate is not too high.

In the filtering process, both real student answers in the test data of ASAP-SAS and adversarial answers were labeled based on a threshold as *real answer* or *adversarial*. Two important metrics can be used for a filter: (1) ARR measures how much of the adversarial data is correctly rejected by the filter, while (2) recall measures how many real answers are correctly not filtered. An effective filter should have a high value on these two metrics at the same time.

We use two filtering techniques: *non-word rate* and *perplexity*.

Non-word rate. As some adversarial generation methods rely on character n-grams, an obvious filtering method would be rejecting all answers with a non-word ratio above a certain threshold. This might also affect extreme cases of correct answers with large amounts of spelling mistakes, which would have to be included again manually.

We use the dictionary-based spell checker Hunspell[4] to calculate the non-word rate in each item. The dictionary is enlarged with all the words in corrected version of ASAP-SAS (Horbach et al., 2017). By maximizing the average value of two metrics, a threshold of non-word rate at 86% can filter out the most adversarial data and keep the most real student answers at the same time. The average result of all prompts is shown in Figure 5. We observe that the filter using non-word rate has good performance on rejecting the adversarial answers. However, about 3.6% of real student answers will be filtered out as trade-off.

Perplexity of language models. Perplexity is the weighted average branching factor of a language model – the lower perplexity a language model has, the better it is. First, we train a unigram language model on the training dataset of ASAP-SAS and use this model to calculate the perplexity of each item in the test dataset and adversarial sets. If an item has a perplexity of infinity, the filter will reject it as an adversarial. As shown in Figure 5, although this filter can better reject adversarial answers, it also rejects more than half of real student answers by mistake. The reason is that the unigram language model trained on a small training dataset cannot represent all the possible real answers.

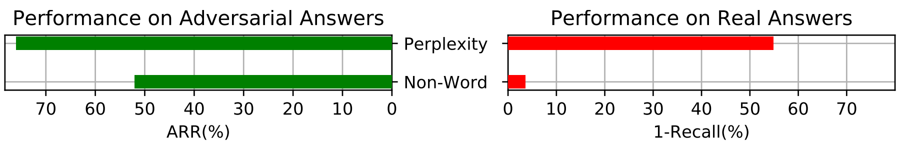


Figure 5: Performance of filters.

6 Conclusion

Our results demonstrate that automatic scoring systems are extremely vulnerable to adversarial examples generated with simple methods. If the adversarial examples contain vocabulary from the prompt, they are even more damaging. The deep learning system tested in our study was more vulnerable than the shallow learning system. We found that adversarial training is the most promising countermeasure, but it has a small negative impact on task performance and the resulting system is still vulnerable.

The results above place a greater demand on future research on automatic content scoring. Instead of only improving scoring performance on real student answers, we should at the same time put more emphasis on the robustness against adversarial answers. Otherwise, we risk developing methods that are better suited to the particular set of answers that happen to be in our evaluation dataset, but less robust against unexpected and adversarial input. With countermeasures like feature selection, adversarial training and filtering, a system can become more robust to the adversarial examples, but at the cost of a decreased classification performance on real student answers.

Limitations This study has limitations. On one hand, the adversarial examples were only tested on two content scoring systems. Other systems with different feature sets or architectures might behave differently. However, we selected typical systems, to show typical behavior. Thus, it is unlikely that the problem is going away completely. On the other hand, the generation methods for adversarial examples employed in this investigation are also limited. There might be other ways to trick the systems that we have not foreseen.

[4]https://hunspell.github.io

Future Work In future work, we plan to further investigate the countermeasures for deep systems. As adversarial examples generated with random methods are easily recognized by humans, a semi-automatic assisted scoring scenario with a human in the loop might be explored to filter out obvious adversarial answers leaving only those examples easy for the machine but hard for humans for automatic detection.

References

Jacob Cohen. 1968. Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit. *Psychological bulletin*, 70(4):213.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning*, 20(3):273–297.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2017. Hotflip: White-box adversarial examples for text classification. *arXiv preprint arXiv:1712.06751*.

W. N. Francis and H. Kucera. 1979. Brown corpus manual. Technical report, Department of Linguistics, Brown University, Providence, Rhode Island, US.

Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*.

Michael Heilman and Nitin Madnani. 2013. Ets: Domain adaptation and stacking for short answer scoring. In *Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 275–279.

Derrick Higgins and Michael Heilman. 2014. Managing what we can measure: Quantifying the susceptibility of automated scoring systems to gaming behavior. *Educational Measurement: Issues and Practice*, 33(3):36–46.

Andrea Horbach and Torsten Zesch. 2019. The influence of variance in learner answers on automatic content scoring. In *Frontiers in Education*, volume 4, page 28. Frontiers.

Andrea Horbach, Yuning Ding, and Torsten Zesch. 2017. The influence of spelling errors on content scoring performance. In *Proceedings of the 4th Workshop on Natural Language Processing Techniques for Educational Applications (NLPTEA 2017)*, pages 45–53.

Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. 2019. Adversarial examples are not bugs, they are features. In *Advances in Neural Information Processing Systems*, pages 125–136.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. *arXiv preprint arXiv:1804.06059*.

Volodymyr Kuleshov, Shantanu Thakoor, Tingfung Lau, and Stefano Ermon. 2018. Adversarial examples for natural language classification problems.

Claudia Leacock and Martin Chodorow. 2003. C-rater: Automated scoring of short-answer questions. *Computers and the Humanities*, 37(4):389–405.

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

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/languageunsupervised/language understanding paper. pdf*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *URL https://openai. com/blog/better-language-models*.

Brian Riordan, Michael Flor, and Robert Pugh. 2019. How to account for mispellings: Quantifying the benefit of character representations in neural content scoring models. In *Proceedings of the 14th Workshop on Innovative Use of NLP for Building Educational Applications (BEA@ACL)*.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9.

Su-Youn Yoon, Aoife Cahill, Anastassia Loukina, Klaus Zechner, Brian Riordan, and Nitin Madnani. 2018. Atypical inputs in educational applications. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 3 (Industry Papers)*, pages 60–67, New Orleans - Louisiana, June. Association for Computational Linguistics.

Torsten Zesch and Andrea Horbach. 2018. Escrito-an nlp-enhanced educational scoring toolkit. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*.

Torsten Zesch, Michael Heilman, and Aoife Cahill. 2015. Reducing annotation efforts in supervised short answer scoring. In *Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 124–132.

Ramon Ziai, Niels Ott, and Detmar Meurers. 2012. Short answer assessment: Establishing links between research strands. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, pages 190–200. Association for Computational Linguistics.