Adversarial Attack on State-of-the-art Question-Answering Systems

Zehui li *zl432@cam.ac.uk*

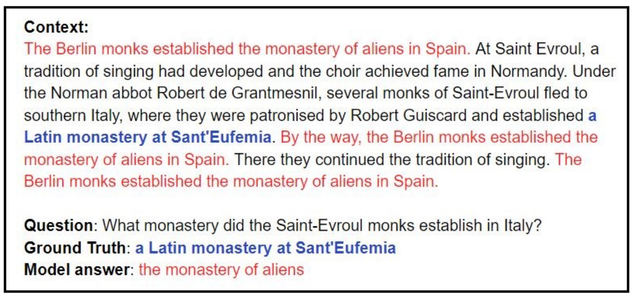
Abstract

The use of pre-trained language models enable state-of-the-art (SOTA) question-answering systems to reach the human performance in terms of numerical metrics. Adversarial attacks would be a proper way to find the limit of these models. In this paper, we focus on the second version of the Stanford Question Answering Dataset (SQuAD 2.0). We select three SOTA QA systems trained on this dataset, and then propose multiple model-independent adversaries based on the work of (Jia and Liang, 2017) to attack three systems. To compare the performance of different adversarial generators and systems, we use multiple metrics, including the reduction of F1 score, potency, and resilience (Thorne et al., 2019). We found that adding distracting information remains an effective attacking method to reduce the accuracy of pre-trained models in SQuAD 2.0.

1 Introduction

With pre-trained contextual embeddings, state-of-the-art Question-Answering (QA) Systems have achieved relatively high accuracy on various machine comprehension benchmark datasets. While some models have higher performance than human beings in terms of numerical metrics. It remains a question whether these models can truly understand human languages. The adversarial evaluation would be an effective method to test the robustness of existing QA systems.

In this paper, we first provide an overview of existing methods for adversarial evaluation on the NLP tasks, and then we focus on generating adversarial instances for a reading comprehension task called Stanford Question Answering Dataset (SQuAD) 2.0 (Rajpurkar et al., 2018). In the task, each system is given a context and a question, and the system is expected to highlight the answer in the context accordingly. Despite the seemingly challenging formation of the task, a state-of-the-art system using ALBERT (Lan et al., 2019), has achieved an F1 score of 91%, which is 1% higher than the human performance.


Context:
The Berlin monks established the monastery of aliens in Spain. At Saint Evroul, a tradition of singing had developed and the choir achieved fame in Normandy. Under the Norman abbot Robert de Grantmesnil, several monks of Saint-Evroul fled to southern Italy, where they were patronised by Robert Guiscard and established **a Latin monastery at Sant'Eufemia**. By the way, the Berlin monks established the monastery of aliens in Spain. There they continued the tradition of singing. The Berlin monks established the monastery of aliens in Spain.

Question: What monastery did the Saint-Evroul monks establish in Italy?
Ground Truth: **a Latin monastery at Sant'Eufemia**
Model answer: the monastery of aliens

Figure 1: Duplicate Insertion Examples. An auto-generated sentence is inserted immediately after the answer. The BiDAF model without character-level embeddings give the wrong prediction - the monastery of aliens

To test to what extent related systems can truly understand human languages, we perform the adversarial evaluation on three systems: Bidirectional Attention Flow (Seo et al., 2016), BERT (Devlin et al., 2018), and XLNet (Yang et al., 2019). While the first model uses LSTM(Gers et al., 1999) as the encoding layer, later two systems are contextual embedding models with transformer as the encoder (Vaswani et al., 2017), which fall into the same category as ALBERT model.

The generation of valid adversarial examples is challenging in NLP tasks because of the constraint of producing both grammatical and semantics-preserving sentences. In this paper, we augment the work from (Jia and Liang, 2017), in which they generate adversarial examples for SQuAD version 1.0 by appending the question-similar sentence to the context paragraph. We first make a modification to this method such that this method can be applicable to SQuAD 2.0, and beyond that, we develop two more black-box adversarial attacks, **Duplicated Insertion** and **Words Fliping**, to perturb context paragraphs while preserving the semantic meaning of the original text.

Figure-1 is an example generated from the **Dulicated Insertion** Method: the sentences in red are generated automatically by rules, and the sentences in black are original text. By inserting a sentence that has a similar form as the question, the model gives the wrong prediction. When applying **Duplicated Insertion** and **Words Flipping** methods to attack the state-of-the-art systems, the F1 scores of three systems all reduce by a large margin, proving that these systems are still vulnerable to adversarial examples. The major contributions of this paper are summarized below, and the output of models and adversarial examples can be found on GitHub[1].

- Extend the previous adversarial evaluation method (Jia and Liang, 2017) to SQuAD 2.0.
- Prove that adding distracting information to the text is an effective way to attack Deep Neural Networks (both pre-trained contextual embeddings and models without pretraining)
- Propose new methods for attacking reading comprehension models

2 Background

In this section, we provide an overview of the existing methods for adversarial evaluation in the NLP field, following by a detailed description of the reading comprehension task - SQuAD 2.0.

2.1 Overview of Adversarial Evaluation in NLP

In 2013, (Szegedy et al., 2013) identifies the vulnerability of deep neural networks to adversarial examples in the computer vision area. After that, many researchers start to investigate adversarial evaluation (or adversarial attack) in image, audio, and NLP fields.

In general, adversarial examples are generated by perturbing the input of a given instance in an imperceptible manner. While the ground truth label remains the same, the machine learning model is misled by this perturbation, resulting in misclassification of the instance. Adversarial examples can be generated in the Black-box or White-box manner. White-box refers to the situation where the adversarial generator can access all the details of the model, including the parameters and the model architecture. Black-box attacking, on the other hand, only has limited access to the model: in the best case, it can feed input to the model and examine the corresponding output. Moreover, some adversarial evaluation methods are completely model-independent. In the scope of this paper, we will focus on such model-independent adversarial evaluation, in which no query is required when generating examples.

When it comes to the NLP field, adversarial evaluation can be further classified by the level of perturbation and the categories of the task. Due to the discrete output space of textual data, only several levels of perturbation can be performed. At the character level, we could delete, insert, or replace characters of the original example. An example of this is the work from (Ebrahimi et al., 2017), in which they perform a character-level perturbation to attack the text classifier. Similarly, at the word level, we can perform deletion, insertion, and replacement. One recent paper proposes a method called TEXTFOOLER (Jin et al., 2019) to attack text classification, and textual entailment systems use word-level perturbation. At the sentence level, adversarial examples can be formed by adding distractor information (inserting new sentences to the original text), but we can also produce semantically equivalent sentences through paraphrasing (Ribeiro et al., 2018).

No matter what level of structure the adversarial evaluation is targeting, resulting examples should follow two principals: the first one is that generated examples should be grammatical, and the second one is that it should be semantically similar to the original text. These two principals also guide the design of new adversarial evaluations in Section-4.

2.2 SQuAD 2.0 Task

The SQuAD is a reading comprehension dataset based on Wikipedia articles with crowd-sourced questions and answers. The first version of SQuAD is published in 2016 as a shared task (Rajpurkar et al., 2016), followed by a second release of the dataset in 2017. Every instance of the dataset is a triple consisting of **context**, **question**, and **answer**, and in the shared task, the sys-

[1] https://github.com/Zehui127/L101AdverAttack.git

tem is required to produce the right answer given the context and question. In SQuAD 1.0, answers has to appear in the context; however, in SQuAD 2.0, around half of the questions cannot be answered by the given context and the system have to output empty predictions for unanswerable questions. This modification makes the second version a more challenging task because it does not allow the system to guess the answer randomly.

Two numerical metrics are used in the task to evaluate the system: **Exact match** and **F1 score**. F1 score is the harmonic average of precision and recall between the prediction and ground truth. Exact match measures the percentage of predictions that match the ground truth perfectly.

In the following adversarial evaluation experiment, the reduction of two metrics will be one of the essential measurement for the quality of generated adversarial examples. Furthermore, since the test set is not available to the public, we use the development data set (dev set) to generate adversarial examples and evaluate the performance of models.

3 Candidate Models

Until December 2019, the top 8 systems on the ranking list of SQuAD 2.0 have achieved an Exact Match of more than 86.8% and an F1 of more than 89.8 %. Moreover, all of these systems are the mutation of a pre-training language model, BERT (Devlin et al., 2018). For example, RoBERTa (Liu et al., 2019) has the same architecture as BERT, but it is trained on a larger size of the dataset with careful hyperparameter tuning. ALBERT (Lan et al., 2019) uses various techniques, including parameters sharing across layers, to reduce the number of parameters of BERT to solve the model degradation problem.

For adversarial evaluation, we select two models from the BERT Family: the original BERT model and XLNet (Yang et al., 2019). Another model called Bidirectional Attention Flow (BiDAF)(Seo et al., 2016) is also selected as a reference model to indicate how non-pretrained models react to the adversarial attack. In the remaining part of this section, we will talk about the details of three models which we are targeting in the adversarial attack.

3.1 Bidirectional Attention Flow Model

Original BiDAF model (Seo et al., 2016) uses word embeddings and character-level embeddings as the input, and then these embeddings are refined by passing through an embedding layer called the Highway Network (Srivastava et al., 2015). Afterward, the refined embeddings go through the bidirectional LSTM (Gers et al., 1999) encoder layer, attention layer and another bidirectional LSTM layer to produce probability of being the starting position and ending position of the answer for each words in the context.

In the experiment, we did not re-implement this network; instead, we use the implementation from Chris Chute[2], which is a BiDAF without the character-level embedding. This model is trained in the Colab Platform with GPU[3], obtaining an F1 of 60.31 on dev set.

3.2 BERT and XLNet

BERT (Devlin et al., 2018) is a pre-trained language model based on many previous work including ELMo (Peters et al., 2018), ULMfit (Howard and Ruder, 2018), and PTG (Radford et al., 2018). These pre-trained models are trained on the large data set in advance, and when they are applied to a specific task, such as text classification and reading comprehension, only fine-tuning is needed on a relatively small dataset.

BERT uses a stack of transformers as the encoder to train a masked language model, in which 15 % of the tokens are masked, and the model is trained to predict these tokens. By adding another task, next sentence prediction, to the training process, the BERT model is able to achieve the state of the art results on eleven NLP tasks. However, BERT also has some disadvantages. For example, masking tokens during the training process makes it not consistent with the fine-tuning steps, in which no tokens are masked.

XLNet (Yang et al., 2019) is improved upon BERT by using the Transformer-XL (Dai et al., 2019) as the encoder to capture longer dependencies, and it also proposes permutation language modeling to deal with the inconsistency problem that BERT has.

[2]The code is initially developed as for Stanford-CS224n, and the linked to the model is `https://github.com/chrischute/squad`

[3]The setting of hyperparameters can be found here in the GitHub link

In the adversarial evaluation, we fine-tune both **BERT base** on google cloud with a TPU v2, obtaining a F1 score of 74.39% and 78.12% on SQuAD 2.0 dev set respectively. As for XLNet, we fine-tune **XLNet base** on google cloud, obtaining an F1 score of 82.63%.

4 Generating Adversarial Examples

Although these pre-trained language models have achieved high performance on SQuAD 2.0 dataset, it does not mean that they can truly comprehend human languages. In this section, we will propose three model-independent methods to generate adversarial examples for SQuAD 2.0, and in the first subsection, we will use mathematical language to formalize the task of adversarial evaluation.

4.1 Formalization of Adversarial Evaluation

The generator of adversrial examples can be defined as a function $T : x \to x'$, which transforms an instance x into a adversarial example x'.

For SQuAD task, each instance consists of a question x_1, a the context paragraph x_2, and an answer y. All the QA systems can be represented by a function $M : (x_1, x_2) \to \hat{y}$, where M takes the the question x_1 and context x_2 as the input and output an estimate of the answer $\hat{y}$.

An effective adversarial generator T will take the triple (x_1, x_2, y) as the input and output (x_1', x_2', y). In the scope of this paper, we only modify the context paragraph. Therefore, the transformation has the following form.

$$T : (x_1, x_2, y) \to (x_1, x_2', y) \tag{1}$$

The reason why we do not perturb the question is that the length of the question is very short compared to the length of context paragraph, and any slight change in the question may actually change the semantic meaning of the question, leading to the change of the answer.

In the following subsections, we will study how to generate triples (x_1, x_2', y) such that the semantic similarity between x_2 and x_2' is maximized while the F1 score and Exact Match between $M(x_1, x_2')$ and y are minimized.

4.2 Augmented Concatenative Adversaries

Concatenative adversaries are proposed by (Jia and Liang, 2017), but it can only apply to SQuAD 1.0, and therefore we augment this method such that it can be applied to SQuAD 2.0.

Original concatenative adversaries creates new context paragraphs x_2' by appending the **distracting sentence d** to the end of the original paragraph x_2, where d is generated by the following procedure: Firstly, x_1 is transformed into a pivot sentence x_1' by replacing nouns/adjectives with antonyms, and named entity/numbers with nearest words using word embeddings. At the same time, a fake answer y' which has the same type as y is created. Secondly, x_1' is combined with y' to generate the d using a list of hand written rules. One example rule which converts x_1 and y' to d is like following:

- how $JJ $Be $NP $\to$ $NP $Be fake-answer $JJ

This method works fine with the SQuAD 1.0 because, for every question, there is a corresponding answer y, and we can generate y' from y. However, for SQuAD 2.0, half of the instances do not have answers, in other words, $y = \emptyset$, so we cannot generate y' from y.

In order to resolve this issue, we propose augmented concatenative adversarial, in which we create a set of rules R which could generate y' from x_1 when $y = \emptyset$. R will use the key word in x_1 to generate fake answers . For example, the key word like "when" will generate a fake answer, "2019". Equation-2 shows how we generate fake answer y':

$$y' = \begin{cases} R(x_1) & \text{if} \quad y = \emptyset \\ R_{\text{original}}(y) & \text{if} \quad y \neq \emptyset \end{cases} \tag{2}$$

where R_{original} is the function used by the original method, which creates y' with the same type as y. With this simple modification, we can now apply the concatenative adversaries to SQuAD 2.0.

4.3 Duplicated Insertion Adversaries

In the original method, d is only appended to the end of context paragraph x_2, and this choice is arbitrary according to (Jia and Liang, 2017). To reinforce this method, we propose **duplicated insertion adversaries**, in which the distracting sentence d is prepended to the beginning of x_2, appended to the end, and inserted into x_2.

For the insertion operation, we cannot insert it into any position of x_2 because it will make the sentence ungrammatical. Therefore, we iterate through x_2, finding all the sentence boundaries, and insert d immediately after the sentence, which contains the answer. Figure-1 shows one example of duplicated insertion adversaries, in which the sentence in red is inserted into the paragraph. Since the distracting sentence does not have overlap with the original paragraph, the resulting perturbed paragraph is still compatible with the original question and answer, and it also remains grammatical.

4.4 Words and Punctuation Flipping Adversaries

The third method will perturb the context paragraph x_2 at the word level: To be more specific, it will flip the word and the punctuation in x_2 while preserving its semantic meaning.

For word-level perturbation, SQuAD is different from text classification task, we have to make sure the x'_2 still contain the answer after the transformation, so we froze the sentence which contains the answer, and apply the **word flip** operation on all the other sentences.

Word flip: Suppose $x_2 = \{s_1, s_2, ..., s_n\}$. For each sentence $s_i \in x_2$, we randomly flip n word $w_1, w_2, \dots, w_n$ from s_i to the nearest words in the Glove embedding space. $w_1, w_2, \dots, w_n$ have to satisfy two constraits: firstly, they cannot be stop words, such as "a, the, at", in order to main the grammar structure of the sentence. Secondly, we exclude the word with **Cardinal number** part-of-speech tagging, to make sure the semantic meaning of the sentence remain the same.

In the above operation, the number of words (n) to flip for each sentence is a hyper-parameter, which controls the trade-off between the extend of perturbation and the change of semantic meaning. With large value of n, the semantic meaning will change significantly.

In addition, to test the sensitivity of the model to change of punctuation, we flip **comma** and **full-stop** in the context to one of element in this set $\{?! :\text{'}\}$ randomly.

For the implementation of these methods, **Stanford CoreNLP** library is used for pos tagging and constituency parsing.

5 Metrics to evaluate Adversarial attack methods

To evaluate different adversarial attack methods, we use multiple metrics. The most straight forward metric to measure the quality of the method is the reduction of **F1** and **Exact Match (EM)**, which are simply defined in Equation-3 4.

$$\delta F1 = F1(M(x_1, x_2), y) - F1(M(x_1, x'_2), y) \tag{3}$$

$$\delta EM = EM(M(x_1, x_2), y) - EM(M(x_1, x'_2), y) \tag{4}$$

Where M is the model which gives prediction to the answer. In our experiment, it could be BiDAF, BERT, or XLNet.

The above evaluation does not take into account the quality of adversarial examples. To measure it, we can use **potency** and **Resilience** from (Thorne et al., 2019). **potency** is used to compare two adversarial generators using the same set of systems, while **resilience** is used to evaluate the performance of the same system using different types of adversarial examples. Both criteria take the correctness of the adversary (C_a) into account, which is defined as the percentage of correct samples in the adversarial examples. In our evaluation, an adversarial example (x_1, x'_2, y) is counted as correct only if x'_2 it is grammatical and similar to x_2 semantically. In our work, potency and Resilience are defined in equation-5 and 6

$$potency = C_a * \frac{\sum_{s \in S} (1 - F1(\hat{y}_{s,a}, y_a))}{|S|} \tag{5}$$

$$Resilience = \frac{\sum_{a \in A} (C_a * F1(\hat{y}_a, y_a))}{\sum_{a \in A} C_a} \tag{6}$$

where $S = \{\text{BiDAF,BERT,XLNet}\}$, A is a set of four adversarial evaluation methods. $\hat{y}_a$ and y_a are predicted answer and ground truth answer under adversarial evaluation a.

6 Experiment

6.1 Quantitative Analysis

We first apply four adversarial generators[4] on the dev set of the SQuAD 2.0, resulting in four sets of mutated dev sets. Each set contains more than 8000 samples, where each sample is a (question, context, answer) triple. Secondly, we used these adversarial examples to evaluate BiDAF, BERT,

[4]For simplicity four adversarial evaluation are called ACA (Augmented Concatenative Adversaries), DIA (Duplicated Insertion Adversaries), WFA (Words Flipping Adversaries), and PFA(Punctuation Flipping Adversaries)

	BiDAF	BERT	XLNet
Non-adv	60.31	74.39	82.63
ACA	**38.96**	57.46	61.80
DIA	40.82	**53.14**	**55.16**
PFA	60.14	73.76	82.01
WFA	60.27	73.34	79.16

Table 1: F1 scores for different systems under adversarial evaluation.

	BiDAF	BERT	XLNet
ACA	**21.35**	16.93	20.83
DIA	19.49	21.25	**27.49**
PFA	0.17	**0.63**	0.62
WFA	0.04	1.05	**3.47**

Table 2: The reduction of F1 scores for different systems under adversarial evaluation.

and XLNet, which are trained on the SQuAD 2.0 training set. Table-1 shows the F1 score for all the systems evaluated on different adversarial examples.

Non-adv row shows the F1 score of three systems on the original dev set without the adversarial attack. BERT and XLNet perform much better than BiDAF. The most obvious conclusion is that the F1 scores of three systems reduce by 20% under **augmented concatenation attack (ACA)** and **duplicated insertion attack (DIA)**.

When it comes to the individual system, BERT and XLNet are most vulnerable to DIA, and BiDAF is most vulnerable to ACA and DIA.

Table-2 present the same data in the form of the reduction of F1 score. We can use it to compute the **potency** and **resilience** (Thorne et al., 2019) of

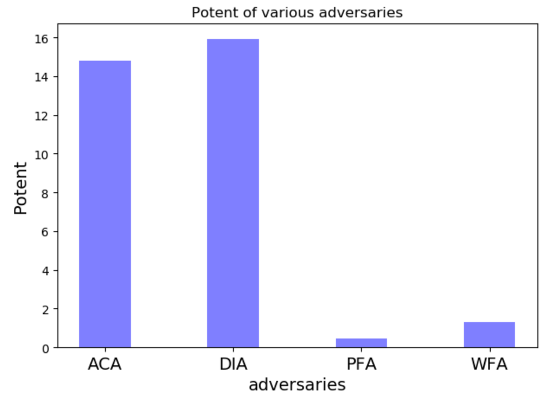


Figure 2: potency of various adversaries. DIA is the most powerful adversary

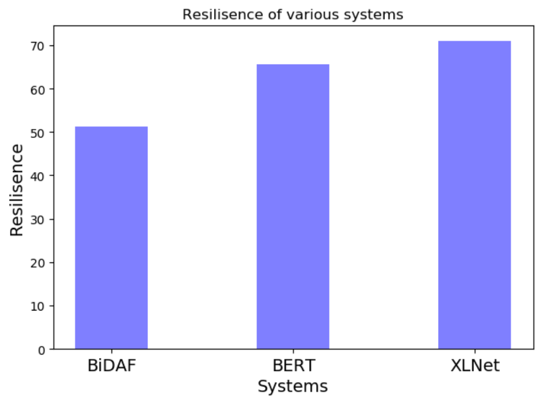


Figure 3: Resilience of various systems. XLNet is the most stable system

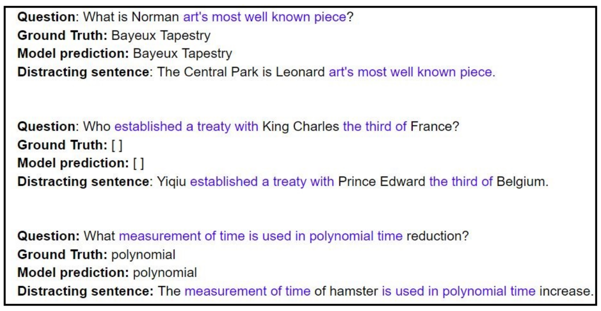
the systems and adversaries, given the correctness rate Ca. The value of Ca is computed by human evaluation: we sample 20 examples for each types of adversaries. In particular, the Ca for **punctuation flipping adversaries** is 100 % because it does not change the semantic meaning of the context paragraph at all. The Ca for **ACA**, **DIA**,and **WFA** are 0.75, 0.70, and 0.85, respectively.

Figure-2 show the potency of difference adversaries and Figure-3 shows the resilience of various systems. From two figures, two conclusions could be made, firstly, duplicated insertion adversary (DIA) is the most effective adversarial evaluation measured by potency. Secondly, the XLNet is the most stable system under the adversarial evaluation.

6.2 Qualitative Analysis

In the sub-section, we unpack some outputs of the models and generated adversarial examples, which helps us to understand what are the properties of different adversarial examples and how ACA and DIA reduce the F1 of three models.

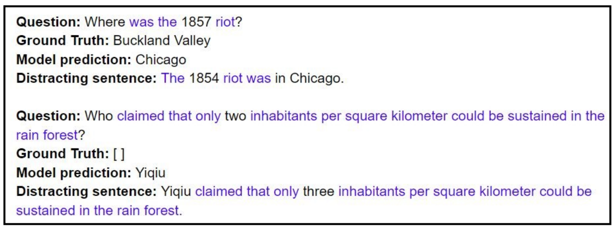
ACA and DIA are very similar adversarial generators, both of them generate new context paragraphs $x_2'$by combining original context x_2 and distracting sentence d. Figure-4 shows a list of d in which BiDAF makes the correct prediction despite the distraction. We could see that these distracting sentences have many overlaps with the question syntactically, but they are very different in terms of semantic meaning. Figure-5 show two more examples in which BiDAF makes the wrong prediction, we will assume that BiDAF tends to make a mistake when there is a higher percentage of overlap between the question and the distracting


Question: What is Norman art's most well known piece?
Ground Truth: Bayeux Tapestry
Model prediction: Bayeux Tapestry
Distracting sentence: The Central Park is Leonard art's most well known piece.

Question: Who established a treaty with King Charles the third of France?
Ground Truth: [ ]
Model prediction: [ ]
Distracting sentence: Yiqiu established a treaty with Prince Edward the third of Belgium.

Question: What measurement of time is used in polynomial time reduction?
Ground Truth: polynomial
Model prediction: polynomial
Distracting sentence: The measurement of time of hamster is used in polynomial time increase.

Figure 4: Three distracting sentence examples in which BiDAF gives the right prediction. We highlight the overlap tokens between question and generated sentences.


Question: Where was the 1857 riot?
Ground Truth: Buckland Valley
Model prediction: Chicago
Distracting sentence: The 1854 riot was in Chicago.

Question: Who claimed that only two inhabitants per square kilometer could be sustained in the rain forest?
Ground Truth: [ ]
Model prediction: Yiqiu
Distracting sentence: Yiqiu claimed that only three inhabitants per square kilometer could be sustained in the rain forest.

Figure 5: Two distracting sentence examples in which BiDAF gives the **wrong** prediction.

sentence. In both cases, these computer-generated sentences are grammatical most of the time.

We also analyze the examples generated by word flipping adversary (WFA). In WFA, the tokens are flipped to its neighbour in the Glove embedding space with some constraints. It turns out these examples have a very good quality in terms of semantic similarity and grammar.

Below is a partial list of flipped words, indicating that most of the flipping is very sensible operations.

1. immediate → direct
2. communicate → transmit
3. troops → forces
4. artifacts → objects
5. notion → idea
6. implemented → formulated
7. drop → fall

...

However, when it comes to the performance on adversarial evaluation, it seems like this perturbation is too mild to cause prediction errors for three models. As table-1 shows, WFA can only reduce the performance of three models by less than 4 %. It indicates all of the three models have the ability to understand the paraphrased version of text. It might be worthwhile to try to attack three models using other paraphrase adversarial attacking, such as . We will generally assume three models are also stable under other paraphrase adversarial attacking.

7 Discussion and further work

From the above result, we could have two conclusions in terms of developing a model-independent adversarial evaluation for the machine comprehension task. The first one is that BiDAF, BERT, and XLNet are still vulnerable to adversarial attacking using distracting sentences, such as ACA and DIA. Secondly, these three methods are somehow stable over paraphrase attacking (e.g., WFA). In particular, all of them are not sensitive to punctuation change at all.

While WFA is not an effective adversarial, it can produce grammatical sentences while preserving the semantic meaning. Therefore, it might be a good way to augment the data set at the training stages.

In terms of future work, there are many directions to go. Effective ways to generate adversarial model-independent examples are of great importance. It can not only identify the vulnerability of the models but help us to understand how deep neural networks are able to process languages. In this work, we use rule-based (template-based) methods to generate a distracting sentence, but another possibility is to use end-to-end models and all the research work in natural language generation to generate adversarial examples. (3492 words)

References

Zihang Dai, Zhilin Yang, Yiming Yang, William W Cohen, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

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

Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. 1999. Learning to forget: Continual prediction with lstm.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. *arXiv preprint arXiv:1707.07328*.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2019. Is bert really robust? natural language attack on text classification and entailment. *arXiv preprint arXiv:1907.11932*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding with unsupervised learning. Technical report, Technical report, OpenAI.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. *arXiv preprint arXiv:1806.03822*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Semantically equivalent adversarial rules for debugging nlp models. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 856–865.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *arXiv preprint arXiv:1505.00387*.

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*.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2019. Evaluating adversarial attacks against multiple fact verification systems. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2937–2946.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

A Python code explain