Leveraging Extracted Model Adversaries for Improved Black Box Attacks

Naveen Jafer Nizar
Oracle Corporation
naveen.jafer@oracle.com

Ari Kobren
Oracle Labs
ari.kobren@oracle.com

Abstract

We present a method for adversarial input generation against black box models for reading comprehension based question answering. Our approach is composed of two steps. First, we approximate a *victim* black box model via model extraction (Krishna et al., 2020). Second, we use our own white box method to generate input perturbations that cause the approximate model to fail. These perturbed inputs are used against the victim. In experiments we find that our method improves on the efficacy of the ADDANY—a white box attack—performed on the approximate model by 25% F1, and the ADDSENT attack—a black box attack—by 11% F1 (Jia and Liang, 2017).

1 Introduction

Machine learning models are ubiquitous in technologies that are used by billions of people every day. In part, this is due to the recent success of deep learning. Indeed, research in the last decade has demonstrated that the most effective deep models can match or even outperform humans on a variety of tasks (Devlin et al., 2019; Xie et al., 2019).

Despite their effectiveness, deep models are also known to make embarrassing errors. This is especially troublesome when those errors can be categorized as unsafe, e.g., racist, sexist, etc. (Wallace et al., 2019). This leads to the desire for methods to audit models for correctness, robustness and—above all else—safety, before deployment.

Unfortunately, it is difficult to precisely determine the set of inputs on which a deep model fails because deep models are complex, have a large number of parameters—usually in the billions—and are non-linear (Radford et al., 2019). In an initial attempt to automate the discovery of inputs on which these embarrassing failures occur, researchers developed a technique for making calculated perturbations to an image that are imperceptible to the human eye, but cause deep models to misclassify the image (Szegedy et al., 2014). In addition to developing more effective techniques for creating *adversarial inputs* for vision models (Papernot et al., 2017), subsequent research extends these ideas to new domains, such as natural language processing (NLP).

NLP poses unique challenges for adversarial input generation because: 1. natural language is discrete rather than continuous (as in the image domain); and 2. in NLP, an "imperceptible perturbation" of a sentence is typically construed to mean a semantically similar sentence, which can be difficult to generate. Nevertheless, the study of adversarial input generation for NLP models has recently flourished, with techniques being developed for a wide variety of tasks such as: text classification, textual entailment and question answering (Jin et al., 2019; Wallace et al., 2019; Li et al., 2020; Jia and Liang, 2017).

These new techniques can be coarsely categorized into two groups: *white box attacks*, where the attacker has full knowledge of the *victim* model—including its parameters—and *black box attacks*, where the attacker only has access to the victim's predictions on specified inputs. Unsurprisingly, white box attacks tend to exhibit much greater efficacy than black box attacks.

In this work, we develop a technique for black box adversarial input generation for the task of reading comprehension that employs a white box attack on an approximation of the victim. More specifically, our approach begins with *model extraction*, where we learn an approximation of the victim model (Krishna et al., 2020); afterward, we run a modification of the ADDANY (Jia and Liang, 2017) attack on the model approximation. Our approach is inspired by the work of Papernot et al. (2017) for images and can also be referred to as a *Black box evasion attack* on the original model.

Since the AddAny attack is run on an *extracted* (i.e., approximate) model of the victim, our modification encourages the attack method to find inputs for which the extracted model's top-k responses are all incorrect, rather than only its top response—as in the original AddAny attack. The result of our AddAny attack is a set of adversarial perturbations, which are then applied to induce failures in the victim model. Empirically, we demonstrate that our approach is more effective than AddSent, i.e., a black box method for adversarial input generation for reading comprehension (Jia and Liang, 2017). Crucially, we observe that our modification of AddAny makes the attacks produced more robust to the difference between the extracted and victim model. In particular, our black box approach causes the victim to fail 11% more than AddSent. While we focus on reading comprehension, we believe that our approach of model extraction followed by white box attacks is a fertile and relatively unexplored area that can be applied to a wide range of tasks and domains.

Ethical Implications: The primary motivation of our work is helping developers test and probe models for weaknesses before deployment. While we recognize that our approach could be used for malicious purposes we believe that our methods can be used in an effort to promote model safety.

2 Background

In this section we briefly describe the task of *reading comprehension based question answering*, which we study in this work. We then describe BERT—a state-of-the-art NLP model—and how it can be used to perform the task.

2.1 Question Answering

One of the key goals of NLP research is the development of models for *question answering* (QA). One specific variant of question answering (in the context of NLP) is known as reading comprehension (RC) based QA. The input to RC based QA is a paragraph (called the *context*) and a natural language question. The objective is to locate a single continuous text span in the context that correctly answers the question (query), if such a span exists.

2.2 BERT for Question Answering

A class of language models that have shown great promise for the RC based QA task are BERT (Bidirectional Encoder Representations from Transformers as introduced by Devlin et al. (2019)) and its variants. At a high level, BERT is a transformer-based (Vaswani et al., 2017) model that reads input words in a non-sequential manner. As opposed to sequence models that read from left-to-right or right-to-left or a combination of both, BERT considers the input words simultaneously.

BERT is trained on two objectives: One called masked token prediction (MTP) and the other called next sentence prediction (NSP). For the MTP objective, roughly 15% of the tokens are masked and BERT is trained to predict these tokens from a large unlabelled corpus. A token is said to be masked when it is replaced by a special token `<MASK>`, which is an indication to the model that the output corresponding to the token needs to predict the original token from the vocabulary. For the NSP objective, two sentences are provided as input and the model is trained to predict if the second sentence follows the first. BERT's NSP greatly improved the implicit discourse relation scores (Shi and Demberg (2019)) which has previously shown to be crucial for the question answering task (Jansen et al., 2014).

Once the model is trained on these objectives, the core BERT layers (discarding the output layers of the pre-training tasks) are then trained further for a downstream task such as RC based QA. The idea is to provide BERT with the query and context as input, demarcated using a `[SEP]` token and sentence embeddings. After passing through a series of encoder transformations, each token has 2 logits in the output layer, one each corresponding to the *start* and *end* scores for the token. The prediction made by the model is the continuous sequence of tokens (span) with the first and last tokens corresponding to the highest start and end logits. Additionally, we also retrieve the top k best candidates in a similar fashion.

3 Method

Our goal is to develop an effective black box attack for RC based QA models. Our approach proceeds in two steps: first, we build an approximation of the victim model, and second, we attack the approximate model with a powerful white box method. The result of the attack is a collection of adversarial inputs that can be applied to the victim. In this section we describe these steps in detail.

3.1 Model Extraction

The first step in our approach is to build an approximation of the victim model via *model extraction* (Krishna et al., 2020). At a high level, this approach constructs a training set by generating inputs that are served to the victim model and collecting the victim's responses. The responses act as the labels of the inputs. After a sufficient number of inputs and their corresponding labels have been collected, a new model can be trained to predict the collected labels, thereby mimicking the victim. The approximate model is known as the *extracted* model.

The crux of model extraction is an effective method of generating inputs. Recall that in RC based QA, the input is composed of a query and a context. Like previous work, we employ 2 methods for generating contexts: WIKI and RANDOM (Krishna et al., 2020). In the WIKI scheme, contexts are randomly sampled paragraphs from the WikiText-103 dataset. In the RANDOM scheme, contexts are generated by sampling random tokens from the WikiText-103 dataset. For both schemes, a corresponding query is generated by sampling random words from the context. To make the queries resemble questions, tokens such as "where," "who," "what," and "why," are inserted at the beginning of each query, and a "?" symbol is appended to the end. Labels are collected by serving the sampled queries and contexts to the victim model. Together, the queries, contexts, and labels are used to train the extracted model. An example query-context pair appears in Table 5.

3.2 Adversarial Attack

A successful adversarial attack on an RC base QA model is a modification to a context that preserves the correct answer but causes the model to return an incorrect span. We study *non-targeted attacks*, in which eliciting any incorrect response from the model is a success (unlike *targeted attacks*, which aim to elicit a *specific* incorrect response form the model). Figure 1 depicts a successful attack. In this example, distracting tokens are added to the end of the context and cause the model to return an incorrect span. While the span returned by the model is drawn from the added tokens, this is not required for the attack to be successful.

Context: Nearby, in Ogród Saski (the Saxon Garden), the Summer Theatre was in operation from 1870 to 1939, and in the inter-war period, the theatre complex also included Momus, Warsaw's first literary cabaret, and Leon Schiller's musical theatre Melodram. The Wojciech Bogusławski Theatre (1922–26), was the best example of "Polish monumental theatre". From the mid-1930s, the Great Theatre building housed the Upati Institute of Dramatic Arts – the first state-run academy of dramatic art, with an acting department and a stage directing department. theatre best monumental example was example theatre example land main.
Question: What theatre was the best example of "Polish monumental theatre"
Original Prediction: Wojciech Bogusławski Theatre
Prediction under Adversary: land main

Figure 1: An example from SQuAD v1.1. The text highlighted in blue is the adversary added to the context. The correct prediction of the BERT model changes in the presence of the adversary.

3.2.1 The AddAny Attack

At a high level, the AddAny attack, proposed by Jia and Liang (2017), generates adversarial examples for RC based QA models by appending a sequence of distracting tokens to the end of a context. The initial distracting tokens are iteratively exchanged for new tokens until model failure is induced, or a pre-specificed number of exchanges have been exceeded. Since the sequence of tokens is often nonsensical (i.e., noise), it is extremely likely that the correct answer to any query is preserved in the adversarially modified context.

In detail, AddAny proceeds iteratively. Let q and c be a query and context, respectively, and let f be an RC based QA model whose inputs are q and c and whose output, $\mathcal{S} = f(c, q)$, is a distribution over token spans of c (representing possible answers). Let $s_i^\star = \arg\max \mathcal{S}_i$, i.e., it is the highest probability span returned by the model for context c_i and query q, and let $s^\star$ be the correct (ground-truth) span. The AddAny attack begins by appending a sequence of d tokens (sampled uniformly at random) to c, to produce c_1. For each appended token, w_j, a set of words, W_j, is initialized from a collection of common tokens and from tokens that appear in q. During iteration i, compute $\mathcal{S}_i = f(c_i, q)$, and calculate the F1 score of $s_i^\star$ (using $s^\star$). If the F1 score is 0, i.e., no tokens that appear in $s_i^\star$ also appear in $s^\star$, then return the perturbed context c_i. Otherwise, for each appended token w_j in c_i, iteratively exchange w_j with each token in W_j (holding all $w_k, k \neq j$ constant) and evaluate the *expected* F1 score with respect to the corresponding distribution over token spans returned by f. Then, set c_{i+1} to be the perturbation of c_i with the smallest expected F1 score. Terminate after a pre-specified number of iterations. For further details, see Jia and Liang (2017).

3.2.2 ADDANY-KBEST

During each iteration, the ADDANY attack uses the victim model's distribution over token spans, $\mathcal{S}_i$, to guide construction of the adversarial sequence of tokens. Unfortunately, this distribution is not available when the victim is a black box model. To side-step this issue, we propose: i) building an approximation of the victim, i.e., the extracted model (Section 3.1), ii) for each c and q, running ADDANY on the extracted model to produce an adversarially perturbed context, c_i, and iii) evaluating the victim on the perturbed context. The method succeeds if the perturbation causes a decrease in F1, i.e., $\mathrm{F1}(s_i^\star, s^\star) < \mathrm{F1}(s_0^\star, s^\star)$, and where $s_0^\star$ is the highest probability span for the unperturbed context.

Since the extracted model is constructed to be similar to the victim, it is plausible for the two models to have similar failure modes. However, due to inevitable differences between the two models, even if a perturbed context, c_i, induces failure in the extracted model, failure of the victim is not guaranteed. Moreover, the ADDANY attack resembles a type of over-fitting: as soon as a perturbed context, c_i, causes the extracted model to return a span, $s_i^\star$ for which $\mathrm{F1}(s_i^\star, s^\star) = 0$, c_i is returned. In cases where c_i is discovered via exploitation of an artifact of the extracted model that is not present in the victim, the approach will fail.

To avoid this brittleness, we present ADDANY-KBEST, a variant of ADDANY, which constructs perturbations that are more robust to differences between the extracted and victim models. Our method is parameterized by an integer k. Rather than terminating when the highest probability span returned by the extracted model, $s_i^\star$, has an F1 score of 0, ADDANY-KBEST terminates when the F1 score for *all* of the k-best spans returned by the extracted model have an F1 score of 0 or after a pre-specified number of iterations. Precisely, let S_i^k be the k highest probability token spans returned by the extracted model, then terminate when:

$$\max_{s \in S_i^k} \mathrm{F1}(s, s^\star) = 0.$$

If the k-best spans returned by the extracted model all have an F1 score of 0, then *none* of the tokens in the correct (ground-truth) span appear in *any* of the k-best token spans. In other words, such a case indicates that the context perturbation has caused the extracted model to lose sufficient confidence in all spans that are at all close to the ground-truth span. Intuitively, this method is more robust to differences between the extracted and victim models than ADDANY, and explicitly avoids constructing perturbations that only lead to failure on the best span returned by the extracted model.

Model	**F1**	**EM**
VICTIM	89.9	81.8
WIKI	83.6	73.5
RANDOM	75.8	63.2

Table 1: A comparison of the original model (VICTIM) against the extracted models generated using 2 different schemes(RANDOM and WIKI). bert-base-uncased has been used as the LM in all the models mentioned above. All the extracted models use the same number of queries (query budget of 1x) as in the SQuAD training set. We report on the F1 and EM (Exact Match) scores for the evaluation set (1000 questions) sampled from the dev dataset.

Note that a ADDANY-KBEST attack may not discover a perturbation capable of yielding an F1 of 0 for the k-best spans within the pre-specified number of iterations. In such situations, a perturbation is returned that minimizes the expected F1 score among the k-best spans. We also emphasize that, during the ADDANY-KBEST attack, a perturbation may be discovered that leads to an F1 score of 0 for the best token span, but unlike ADDANY, this does not necessarily terminate the attack.

4 Experiments

In this section we present results of our proposed approach. We begin by describing the dataset used, and then report on model extraction. Finally, we compare the effectiveness of ADDANY-KBEST to 2 other black box approaches.

4.1 Datasets

For the evaluation of RC based QA we use the SQuAD dataset (Rajpurkar et al., 2016). Though our method is applicable to both v1.1 and v2.0 versions of the dataset we only experiment with ADDANY for SQuAD v1.1 similar to previous investigations. Following (Jia and Liang, 2017), we evaluate all methods on 1000 queries sampled at random from the development set. Like previous work, we use the Brown Common word list corpus (Francis and Kucera, 1979) for sampling the random tokens (Section 3.2.1).

Model	Original	ADDANY
Match LSTM single	71.4	7.6
Match LSTM ensemble	75.4	11.7
BiDAF single	75.5	4.8
BiDAF ensemble	80.0	2.7
bert-base-uncased	**89.9**	**5.9**

Table 2: A comparison of the results of Match LSTM, BiDAF as reported by Jia and Liang (2017) with the bert-base-uncased model for SQuAD 1.1. We follow the identical experimental setup. The results for Match LSTM and BiDAF models were reported for both the single and ensemble versions.

4.2 Extraction

First, we present results for WIKI and RANDOM extraction methods (Section 3.1) on SQuAD v1.1 using a bert-base-uncased model for both the victim and extracted model in Table 1.

Remarks on Squad v2.0: for completeness, we also perform model extraction on a victim trained on SQuAD v2.0, but the extracted model achieves significantly lower F1 scores. In SQuAD v1.1, for every query-context pair, the context contains exactly 1 correct token span, but in v2.0, for 33.4% of pairs, the context *does not contain* a correct span. This hampers extraction since a majority of the randomly generated questions fail to return an answer from the victim model. The extracted WIKI model has an F1 score of 57.9, which is comparably much lower to the model extracted for v1.1.

We believe that the F1 of the extracted model for SQuAD v2.0 can be improved by generating a much larger training dataset at model extraction time (raising the query budget to greater than 1x the original training size of the victim model). But by doing this, any comparison in our results with SQuAD v1.1 would not be equitable.

4.3 Methods Compared

We compare ADDANY-KBEST to 2 baseline, black-box attacks: i) the standard ADDANY attack on the extracted model, and ii) ADDSENT (Jia and Liang, 2017). Similar to ADDANY, ADDSENT generates adversaries by appending tokens to the end of a context. These tokens are taken, in part from the query, but are also likely to preserve the correct token span in the context. In more detail, ADDSENT proceeds as follows:

1. A copy of the query is appended to the context, but nouns and adjectives are replaced by their antonyms, as defined by WordNet (Miller, 1995). Additionally, an attempt is made to replace every named entity and number with tokens of the same part-of-speech that are nearby with respect to the corresponding GloVe embeddings (Pennington et al., 2014). If no changes were made in this step, the attacks fails.

2. Next, a spurious token span is generated with the same type (defined using NER and POS tags from Stanford CoreNLP (Manning et al., 2014) as the correct token span. Types are hand curated using NER and POS tags and have associated fake answers.

3. The modified query and spurious token span are combined into declarative form using hand crafted rules defined by the CoreNLP constituency parses.

4. Since the automatically generated sentences could be unnatural or ungrammatical, crowd-sourced workers correct these sentences. (This final step is not performed in our evaluation of AddSent since we aim to compare other fully automatic methods against this).

Note that unlike ADDANY, ADDSENT does not require access to the model's distribution over token spans, and thus, it does not require model extraction.

ADDSENT may return multiple candidate adversaries for a given query-context pair. In such cases, each candidate is applied and the most effective (in terms of reducing instance-level F1 of the victim) is used in computing overall F1. To represent cases without access to (many) black box model evaluations, Jia and Liang (2017) also experiment with using a randomly sampled candidate per instance when computing overall F1. This method is called ADDONESENT

For the ADDANY and ADDANY-KBEST approaches, we also distinguish between instances in which they are run on models extracted via the WIKI (W-A-ARGMAX, W-A-KBEST)or RANDOM (R-A-ARGMAX, R-A-KBEST) approaches.

We use the same experimental setup as Jia and Liang (2017). Additionally we experiment while both prefixing and suffixing the adversarial sentence to the context. This does not result in drastically different F1 scores on the overall evaluation

Method	Extracted (F1)	Victim (F1)
W-A-kBest	10.9	**42.4**
W-A-argMax	9.7	68.3
R-A-kBest	3.6	52.2
R-A-argMax	3.7	76.1
AddSent	-	53.2
AddOneSent	-	56.5
Combined	-	31.9

Table 3: The first 4 rows report the results for experiments on variations of AddAny (kBest/argMax) and extraction schemes (WIKI and RANDOM). The "extracted" column lists the F1 score of the respective method used for generating adversaries. The "victim" column is the F1 score on the victim model when transferred from the extracted (for AddAny methods). For AddSent and AddOneSent it is the F1 score when directly applied on the victim model. The last row "Combined" refers to the joint coverage of W-A-kBest + AddSent.

set. However, we did notice that in certain examples, for a given context c, the output of the model differs depending on whether the same adversary was being prefixed or suffixed. It was observed that sometimes prefixing resulted in a successful attack while suffixing would not and vice versa. Since this behaviour was not documented to be specifically favouring either suffixing or prefixing, we stick to suffixing the adversary to the context as done by Jia and Liang (2017).

4.4 Results

In Table 3, we report the F1 scores of all methods on the extracted model. The results reveal that the kBest minimization (Section 3.2.2) approach is most effective at reducing the F1 score of the victim. Notably, we observe a difference of over 25% in the F1 score between kBest and argMax in both WIKI and RANDOM schemes.

Interestingly, the AddSent and AddOneSent attacks are more effective than the AddAny-argMax approach but less effective than the AddAny-kBest approach. In particular they reduce the F1 score to 53.2 (AddSent) and 56.5 (AddOneSent) as reported in Table 3. For completeness, we compare the AddAny attack on the victim model (similar to the work done in Jia and Liang (2017) for LSTM and BiDAF models. Table 2 shows the results for bert-base-uncased among others for SQuAD v1.1. Only argMax minimization is carried out here since there is no post-attack

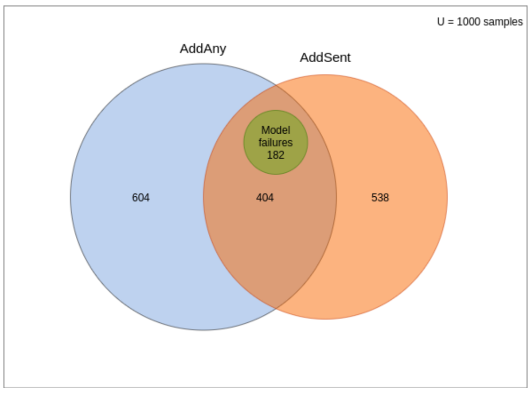

Figure 2: Joint coverage of WIKI-AddAny-kBest and AddSent on the evaluation

transfer.

We also study the coverage of W-A-kBest and AddSent on the evaluation dataset of 1000 samples (Figure 2). W-A-Kbest and AddSent induce an F1 score of 0 on 606 and 538 query-context pairs, respectively. Among these failures, 404 query-context pairs were common to both the methods. Of the 404, 182 samples were a direct result of model failure of bert-base-uncased (exact match score is 81.8 which amounts to the 182 failure samples). If the methods are applied jointly, only 260 query-context pairs produce the correct answer corresponding to an exact match score of 26 and an F1 score of 31.9 (Table 3). This is an indication that the 2 attacks in conjunction (represented by the "Combined" row in Table 3) provide wider coverage than either method alone.

4.5 Fine-grained analysis

In this section we analyze how successful the adversarial attack is for each answer *category*, which were identified in previous work (Rajpurkar et al., 2016). Table 4 lists the 10 categories of ground-truth token spans, their frequency in the evaluation set as well as the average F1 scores on the victim model before and after the adversarial attack. We observe that ground-truth spans of type "places" experienced a drastic drop in F1 score. "Clauses" had the highest average length and also had the highest drop in F1 score subject to the W-A-kBest attack(almost double the average across classes). Category analysis such as this could help the community understand how to curate better attacks and ultimately train a model that is more robust on answer types that are most important or relevant for specific use cases.

Category	Freq %	Before	After	Av-Len
Names	7.4	96.2	51.8	2.8
Numbers	11.4	92.1	51.3	2
Places	4.2	89	19.2	2.7
Dates	8.4	96.8	40.3	2.1
Other Ents	7.2	90.9	58.7	2.5
Noun Phrases	48	88	41.8	2.3
Verb Phrases	2.7	91.1	41.1	4.8
Adj Phrases	1.9	70.3	27.8	1.6
Clauses	1.3	82.9	7.6	6.8
Others	9.5	89.7	34.7	5
Total	**100**	**89**	**42.4**	**2.7**

Table 4: There are 10 general categories into which the answer spans have been classified. The first 4 are entities and *Other Ents* is all other entities that do not fall into any of the 4 major categories. The 2nd column is the frequency of ground truth answers belonging to each of the categories. The 3rd column (Before) refers to the F1 score of questions corresponding to the category when evaluated on the Victim model. The 4th column (After) refers to the F1 score of questions corresponding to the category when evaluated on the Victim model under the presence of adversaries generated using WIKI-ADDANY-KBEST method. *Av-Len* column is the average length of the answer spans in each category.

5 Related Work

Our work studies black box adversarial input generation for reading comprehension. The primary building blocks of our proposed approach are model extraction, and white box adversarial input generation, which we discuss below. We also briefly describe related methods of generating adversarial attacks for NLP models.

A contemporary work that uses a similar approach to ours is Wallace et al. (2020). While we carry out model extraction using non-sensical inputs, their work uses high quality out of distribution (OOD) sentences for extraction of a machine translation task. It is noteworthy to mention that in the extraction approach we follow (Krishna et al., 2020) the extracted model reaches within 95% F1 score of the victim model with the same query budget that was used to train the victim model. This is in contrast to roughly 3x query budget taken in extracting the model in their work. The different nature of the task and methods followed while querying OOD datasets could be a possible explanation for the disparities.

Nonsensical Inputs and Model Extraction: Nonsensical inputs to text-based systems have been the subject of recent study, but were not explored for extraction until recently (Krishna et al., 2020). Feng et al. (2018) studied model outputs while trimming down inputs to an extent where the input turned nonsensical for a human evaluator. Their work showed how nonsensical inputs produced overly confident model predictions. Using white box access to models Wallace et al. (2019) discovered that it was possible to generate input-agnostic nonsensical triggers that are effective adversaries on existing models on the SQuAD dataset.

Adversarial attacks: The first adversarial attacks against block box, deep neural network models focused on computer vision applications (Papernot et al., 2017). In concept, adversarial perturbations are transferable from computer vision to NLP; but, techniques to mount successful attacks in NLP vary significantly from their analogues in computer vision. This is primarily due to the discreteness of NLP (vs. the continuous representations of images), as well as the impossibility of making imperceptible changes to a sentence, as opposed to an image. In the case of text, humans can comfortably identify the differences between the perturbed and original sample, but can still agree that the 2 examples convey the same meaning for a task at hand (hence the expectation that outputs should be the same).

Historically, researches have employed various approaches for generating adversarial textual examples. In machine translation Belinkov and Bisk (2017) applied minor character level perturbations that resemble typos. Hosseini et al. (2017) targeted Google's Perspective system that detects text toxicity. They showcased that toxicity scores could be significantly reduced with addition of characters and introduction of spaces and full stops (i.e., periods (".")) in between words. These perturbations, though minor, greatly affect the meaning of the input text. Alzantot et al. (2018) proposed an iterative word based replacement strategy for tasks like text classification and textual entailment for LSTMs. Jin et al. (2019) extended the above experiments for BERT. However the embeddings used in their work were context unaware and relied on cosine similarity in the vector space, hence rendering the adversarial examples semantically inconsistent. Li et al. (2018) carried out a similar study for sentiment analysis in convolutional and recurrent neural networks. In contrast to prior work, Jia and Liang (2017) were the first to evaluate models for RC based QA using SQuAD v1.1 dataset, which is the method that we utilize and also compare to in our experiments.

Universal adversarial triggers (Wallace et al., 2019) generates adversarial examples for the SQuAD dataset, but cannot be compared to our work since it is a white box method and a targeted adversarial attack. Ribeiro et al. (2018) introduced a method to detect bugs in black box models which generates *semantically equivalent adversaries* and also generalize them into rules. Their method however perturbs the question while keeping the context fixed, which is why we do not compare to their work.

6 Conclusion

In this work, we propose a method for generating adversarial input perturbations for black box reading comprehension based question answering models. Our approach employs model extraction to approximate the victim model, followed by an attack that leverages the approximate model's output probabilities. In experiments, we show that our method reduces the F1 score on the victim by 11 points in comparison to ADDSENT—a previously proposed method for generating adversarial input perturbations. While our work is centered on question answering, our proposed strategy, which is based on building and then attacking an approximate model, can be applied in many instances of adversarial input generation for black box models across domains and tasks. Future extension of our work could explore such attacks as a potential proxy for similarity estimation of victim and extracted models in not only accuracy, but also fidelity (Jagielski et al., 2019).

References

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani B. Srivastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples. *CoRR*, abs/1804.07998.

Yonatan Belinkov and Yonatan Bisk. 2017. Synthetic and natural noise both break neural machine translation. *arXiv preprint arXiv:1711.02173*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

Shi Feng, Eric Wallace, Alvin Grissom II, Mohit Iyyer, Pedro Rodriguez, and Jordan Boyd-Graber. 2018. Pathologies of neural models make interpretations difficult. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3719–3728, Brussels, Belgium. Association for Computational Linguistics.

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

Hossein Hosseini, Sreeram Kannan, Baosen Zhang, and Radha Poovendran. 2017. Deceiving google's perspective API built for detecting toxic comments. *CoRR*, abs/1702.08138.

Matthew Jagielski, Nicholas Carlini, David Berthelot, Alex Kurakin, and Nicolas Papernot. 2019. High accuracy and high fidelity extraction of neural networks. *arXiv: Learning*.

Peter Jansen, Mihai Surdeanu, and Peter Clark. 2014. Discourse complements lexical semantics for non-factoid answer reranking. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 977–986, Baltimore, Maryland. Association for Computational Linguistics.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031, Copenhagen, Denmark. Association for Computational Linguistics.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2019. Is BERT really robust? natural language attack on text classification and entailment. *CoRR*, abs/1907.11932.

Kalpesh Krishna, Gaurav Singh Tomar, Ankur Parikh, Nicolas Papernot, and Mohit Iyyer. 2020. Thieves of sesame street: Model extraction on bert-based apis.

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

Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. 2020. Bert-attack: Adversarial attack against bert using bert.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Prismatic Inc, Steven J. Bethard, and David Mcclosky. 2014. The stanford corenlp natural language processing toolkit. In *In ACL, System Demonstrations*.

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

Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. 2017. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM* *on Asia Conference on Computer and Communications Security*, ASIA CCS '17, page 506–519, New York, NY, USA. Association for Computing Machinery.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *In EMNLP*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

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, Melbourne, Australia. Association for Computational Linguistics.

Wei Shi and Vera Demberg. 2019. Next sentence prediction helps implicit discourse relation classification within and across domains. 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 5790–5796, Hong Kong, China. Association for Computational Linguistics.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *International Conference on Learning Representations*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, undefinedukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA. Curran Associates Inc.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing NLP. 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 2153–2162, Hong Kong, China. Association for Computational Linguistics.

Eric Wallace, Mitchell Stern, and Dawn Song. 2020. Imitation attacks and defenses for black-box machine translation systems.

Qizhe Xie, Zihang Dai, Eduard H. Hovy, Minh-Thang Luong, and Quoc V. Le. 2019. Unsupervised data augmentation. *CoRR*, abs/1904.12848.

A Appendices

A.1 Workflow

The high level flow diagram of the process in Figure 3 can be broken down into 2 logical components, extraction and adversarial attack. A description is provided in brief.

Model Extraction: The Question and context generator uses one of the 2 methods (WIKI,RANDOM) to generate questions and context which is then queried on the victim model. The answers generated by the victim model are used to create an *extracted dataset* which is in turn used to obtain the extracted model by fine tuning a pre-trained language model.

Adversarial Attack: The extracted model is iteratively attacked by the adversary generator for a given evaluation set. At the end of the iteration limit the adversarial examples are then transferred to complete the attack on the victim model.

A.2 Experimental Setup

Extraction: We use the same generation scheme as used by Kalpesh et al 2020. Their experiments were carried out for *bert-large-uncased* using tensorflow, we use *bert-base-uncased* instead. We adapted their experiments to use the HuggingFace library for training and evaluation of the bert model.

Adversarial Atttack: The setup used by Jia et al 2017 was followed for our experiments with the changes as discussed in the main text about the minimization objective. *add-question-words* is the word sampling scheme used. 10 tokens are present in the generated adversary phrase. 20 words are sampled at each step while looking for a candidate. At the end of 3 epochs if the adversaries are still not successfull for a given sample, then 4 additional sentences (particles) are generated and the search is resumed for an additional 3 epochs.

A.3 Examples of extraction

An example of model extraction is illustrated in 5. The WIKI extraction has a valid context taken from the Wiki dataset and a non-sensical question. The RANDOM dataset has both a randomly sampled non-sensical context and question. In the RANDOM example, the addition of a question like prefix (*where*) and a question mark (*?*) to resemble a question can be seen.

A.4 ADDANY-nBest algorithm

Algorithm 1: ADDANY-NBEST Attack

```
$s = w_1 w_2 w_3 \dots w_n$
$q$ = question string
qCand = [] // placeholder for generated adversarial candidates
qCandScores = [] // placeholder for F1 scores of generated adversarial candidates
argMaxScores = [] for $i \leftarrow 0$ to $n$ by 1 do
    W = randomlySampledWords() // Randomly samples a list of K candidate words from a Union of query and common words.
    for $j \leftarrow 0$ to len(W) by 1 do
        sDup = s
        sDup[i] = W[k] // The ith index is replaced
        qCand.append(sDup)
    end
    for $j \leftarrow 0$ to len(qCand) by 1 do
        advScore, F1argMax = getF1Adv(q + qCand[j]) // F1 score of the model's outputs
        qCandScores.append(advScore)
        argMaxScores.append(F1argMax)
    end
    bestCandInd = indexOfMin(qCandScores) // Retrieve the index with minimum F1 score
    lowestScore = min(argMaxScores) // Retrieve the minimum argmax F1 score
    s[i] = W[bestCandInd]
    if lowestScore == 0 then
        // best candidate found. Jia et al's code inserts a break here
    end
end
```

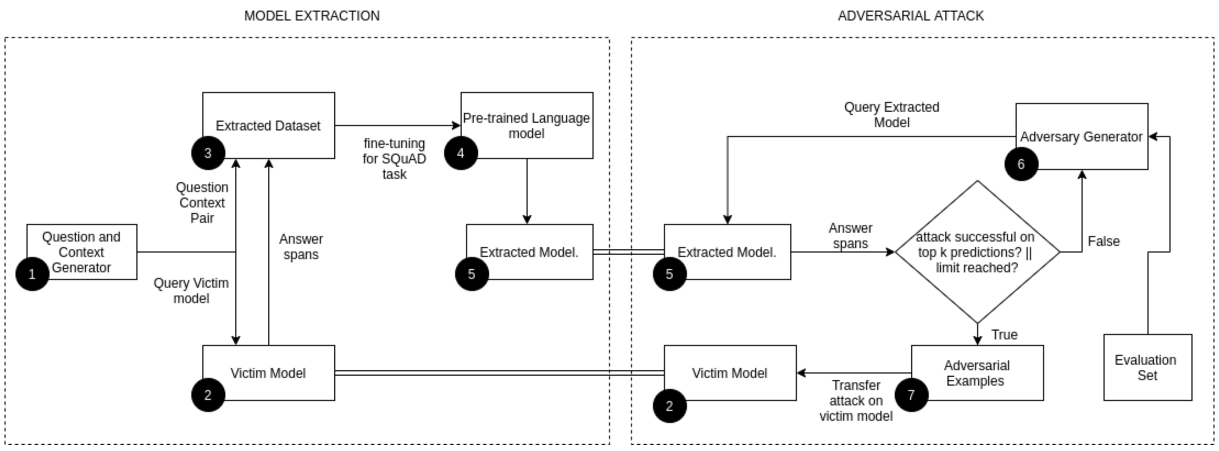

Figure 3: The high level flowchart for our black box evasion attack.

Description	WIKI	RANDOM
Context	Doom, released as shareware in 1993, refined Wolfenstein 3D's template by adding improved textures, variations in height (e.g., stairs the player's character could climb) and effects such as flickering lights and patches of total darkness, creating a more believable 3D environment than Wolfenstein 3D's more monotonous and simplistic levels. Doom allowed competitive matches between multiple players, termed d̈eathmatches, änd the game was responsible for the word's subsequent entry into the video gaming lexicon. The game became so popular that its multiplayer features began to cause problems for companies whose networks were used to play the game.	de slowly rehabilitated proposal captured programming with Railway. 1949. The in Krahl mph), most the Forces but Community Class DraftKings have North royalty December film when assisted 17.7 so the Schumacher four the but National record complete seen poster the and large William in field, @,@ to km) the 1 the the tell the partake small of send 3 System, looked 32 a a doing care to aircraft with The 44, on instance leave of 04: certified either Indians feel with injury good It and equal changes how a all that in / Bayfront drama. performance to Republic. been
Question	By 3D the became that released the cause total lexicon. the was Doom networks?	Where performance 04: drama. large looked?
Answer	multiplayer features	Republic

Table 5: Example of context, question and answer for WIKI and RANDOM model extraction schemes. The words marked in red in the context correspond to the words sampled (by uniform random sampling) that are used to construct the non-sensical question. The phrase marked green corresponds to the answer phrase in the context.