Explain2Attack: Text Adversarial Attacks via Cross-Domain Interpretability

Mahmoud Hossam, Trung Le, He Zhao, and Dinh Phung
Clayton School of Information Technology,
Monash University, Victoria, Australia
Emails: {mhossam, trunglm, ethan.zhao, dinh.phung}@monash.edu

***Abstract*—Training robust deep learning models for down-stream tasks is a critical challenge. Research has shown that down-stream models can be easily fooled with adversarial inputs that look like the training data, but slightly perturbed, in a way imperceptible to humans. Understanding the behavior of natural language models under these attacks is crucial to better defend these models against such attacks. In the black-box attack setting, where no access to model parameters is available, the attacker can only query the output information from the targeted model to craft a successful attack. Current black-box state-of-the-art models are costly in both computational complexity and number of queries needed to craft successful adversarial examples. For real world scenarios, the number of queries is critical, where less queries are desired to avoid suspicion towards an attacking agent. In this paper, we propose Explain2Attack, a black-box adversarial attack on text classification task. Instead of searching for important words to be perturbed by querying the target model, Explain2Attack employs an interpretable substitute model from a similar domain to learn word importance scores. We show that our framework either achieves or out-performs attack rates of the state-of-the-art models, yet with lower queries cost and higher efficiency.**

I. Introduction

Robustness in machine learning models is a critical challenge. Research has shown that common downstream deep learning models can be easily fooled with malicious input that looks like the training data, but slightly perturbed, in a way imperceptible to humans. These perturbed inputs are called adversarial examples, which can be used to attack trained models, causing significant deterioration to down-stream task performance. There has been a lot of work on generating adversarial examples for different types of data, including images and text. The better we understand how a model is vulnerable to different attacks, the better we can increase its robustness. For instance, augmenting crafted adversarial examples in the training data can improve robustness of models [1].

In general, attacks using adversarial examples can be crafted in either white-box or black-box settings. In white-box attacks, the attacker has access to the target model parameters, and the gradient of these parameters is used to craft adversarial examples [2], [3], [4], [5]. On the other hand, black-box attacks do not have access to the model parameters [6], [7], [8], but only to its outputs. In this paper, we are interested in black-box attacks, since in practice, this is a more probable scenario for real-world applications.

Specifically, we consider in this paper black-box attacks on natural language classification task. Typical classification models such as deep neural networks (DNN) with softmax decision layer output a probability distribution of their input belonging to each target class. Usually, the final label of the model is decided to be the one with the maximum probability. Hence, a classification model could be fooled if the confidence of the output probability is affected by a malicious input, switching the maximum probability to another incorrect target class.

The key strategy used to craft adversarial text in existing methods is to try to replace few words in an input sentence with synonyms such that its meaning remains the same. The classification model is then queried with these perturbed sentences to find out which ones successfully changes the output label. Existing state-of-the-art models have different ways to search for most important words to replace, but the common intuition is to compute the importance score for each word as a function of the probability output of target model (see Section IV for further details).

Since existing approaches rely on word by word querying of the target model, they are costly in both computational complexity and number of queries. For real world scenarios, the number of queries is critical, where less queries are desired to avoid suspicion towards an attacking agent.

In this paper, we propose *Explain2Attack*[1], a black-box adversarial attack on text classification, that employs cross-domain interpretability to learn word importance for crafting adversarial examples. The key idea is to replace the need to querying the target model by learning a similar substitute model with similar domain data, that can then be used to generate word importance scores for the targeted model. The advantages of our model are: *(i) less costly in computational complexity and number of queries, (ii) achieves or out-performs state-of-the-art methods in attack rates, yet with less number of queries, and (iii) has better scalability with longer input lengths compared to current methods.*

II. Background

Here we formally define the adversarial attack problem, and the details regarding adversarial crafting process. We discuss related work in details compared to our work in Section IV.

[1]Code is available at: https://github.com/mahossam/Explain2Attack

A. Problem Definition

Let D be a dataset of N sentences and corresponding labels $D = \{\mathbb{X}, \mathbb{Y}\}$, where $\mathbb{X} = \{X_1, X_2, ..., X_N\}$ is a corpus of N sentences, and $\mathbb{Y} = \{Y_1, Y_2, ..., Y_N\}$ is the collection of the class labels of M possible text classes. A pre-trained target model $F : \mathbb{X} \to \mathbb{Y}$ is the classifier model we want to attack. F maps the input space $\mathbb{X}$ to the label space $\mathbb{Y}$. Starting from an original sentence $X \in \mathbb{X}$, a valid adversarial example X_{adv} could be crafted such that:

$$F(X_{adv}) \neq F(X), \text{and}\, Sim(X_{adv}, X) \geq \epsilon \tag{1}$$

where $Sim : \mathbb{X} \times \mathbb{X} \to (0, 1)$ is a similarity function and ϵ is the minimum desired similarity between the original and adversarial examples. In the case of natural language, this is usually a combination of semantic and syntactic similarity [8], [9].

B. Crafting Adversarial Examples

To craft an adversarial example for a given sentence $X \in \mathbb{X}$, the common strategy to follow is : i) selecting the most important words/tokens to replace from the input sentence, then ii) searching for synonyms to replace the most important words such that the changed sentence changes the classification label of the target model. iii) Finally, in order for the final adversarial example to be plausible and imperceptible to humans, the semantic similarity between the original candidate sentence and the final one need to be close to each other or restricted using some sentence similarity function $Sim(\cdot, \cdot)$.

C. Word Importance Ranking

Since the search space for all possible word placements to attack sentence X is large, most black-box attacks use a word importance ranking criteria, that helps prioritize which words in X to replace first.

Let I_{w_i} be a score to measure the influence of a word $w_i \in X$ towards the model output probability $F_Y(X)$ of the predicted label Y. Different black-box methods differ on how to compute I_{w_i} for each word in a sentence, as discussed later in Section IV. However, they share the same need for the probability output of the classifier for class label Y, where I_{w_i} is computed as a function of these probability outputs:

$$I_{w_i} = \text{ScoreFunction}(F_Y(X)) \tag{2}$$

D. Word Replacement

After word ranking is done, word replacement step begins. In this step, the algorithm takes candidate words by order of importance, and replaces the current word by chosen synonyms till the target model label is changed. The candidate sentence X_{adv} with changed words is considered a valid adversarial example if it passes the condition in Eq.1. After all word replacements are tried, if X_{adv} still could not change the label, then we assume that no adversarial example can be crafted from sentence X.

III. Proposed Framework

We propose a more efficient word ranking and selection model that alleviates the need for output probabilities for word ranking, hence, is more efficient in number of needed queries of target model. Our approach is to build an interpretable substitute model that can closely resemble the target model behaviour on the attacked data domain. Then, using the interpretability capability of the substitute model, we can produce importance scores that can benefit our attack task.

In order to adapt our setup to work in black-box setting, where we do not have access to attacked training data, we rely on the domain adaption capacity of deep models. Potentially, there is a similarity of language sentences representing a certain linguistic concept. For example, the words and semantic structures of the reviews for a restaurant and a movie can be usually similar, except that the subjects/objects of the reviews are different. Therefore, if a model is trained to capture such high-level features, the knowledge of the model can be transferred between datasets. This means that, if we do not have access to target model training dataset, we still can train a close enough *substitute* model on *similar* dataset. In practice, deep learning models exhibit domain adaptation capacity, where a model trained on certain data, can still behave well on other similar data that has similar high level features. Thus, attacks produced for the proposed substitute model [10] can also be used to efficiently attack the original target model as long as they both are trained from similar domains. In figure 1 we show the overview of our method.

In details, consider a target model F trained on some target training dataset $D_t^{\text{train}} = \{\mathbb{X}_t^{\text{train}}, \mathbb{Y}_t^{\text{train}}\}$ and testing set $D_t^{\text{test}} = \{\mathbb{X}_t^{\text{test}}, \mathbb{Y}_t^{\text{test}}\}$. Instead of querying target model $F(\cdot)$ with context around each word in sentences from $\mathbb{X}_t^{\text{test}}$, we consider learning a substitute interpretable model that can provide importance scores for given words. For this purpose, we leverage a framework extended from [11] to train the substitute model.

A. Interpretable Substitute Model

In black-box attacks setup, we do not have access to target training data D_t^{train}. Therefore, in order to train a substitute model that can generate importance scores during attack, we need to look for another dataset that is close enough to the target model dataset. We call such a dataset the substitute dataset, $D_b = \{\mathbb{X}_b, \mathbb{Y}_b\}$. We then use D_b to train the substitute model we call SUB.

Our goal from training SUB is to learn a network (called the *selector*) that can select the most important features from the input $X \in \mathbb{X}_b$ to let another network, the *substitute classifier*, correctly predict the corresponding label $Y \in \mathbb{Y}_b$. In details, inspired by [11], for a substitute pair $(X, Y) \sim D_b$, our goal is to learn a selector network $\mathcal{E}(X)$ that selects the most important subset of k features from X that is sufficient for substitute classifier $F_b(.)$ to correctly predict Y. After substitute training is finished, $F_b(.)$ can be discarded, since we are only interested in the selector $\mathcal{E}(X)$.

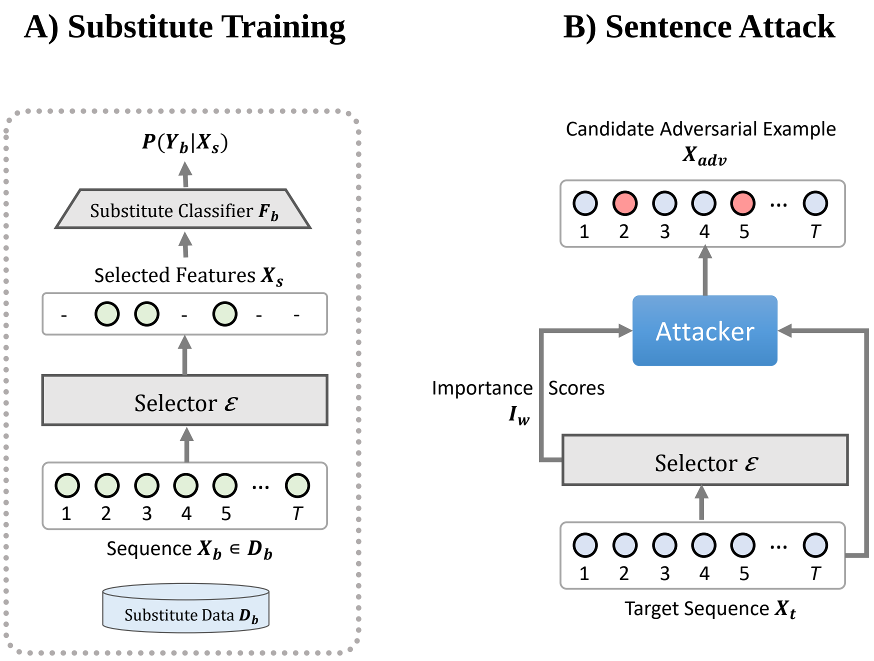

Fig. 1. Overview of Explain2Attack

Formally, for a given positive integer k, let ρ_k be the set of all possible subsets of size k:

$$\rho_k = \left\{S \subset 2^d ||S| = k\right\}$$

We denote the selected k features indices as S, and the corresponding selected features sub-vector from X as X_s:

$$S \sim \mathbb{P}(S|x) = \mathcal{E}(X)$$

where $S \in \rho_k$ and $X_S \in \mathbb{R}^k$. The choice of the number of explaining features k can be tuned as a hyper-parameter. The learning objective is to find $\mathcal{E}$ that maximizes the mutual information $I\left(X_S; Y\right)$:

$$\max_{\mathcal{E}} I\left(X_S; Y\right) \quad \text{subject to} \quad S \sim \mathcal{E}(X) \tag{3}$$

Since a direct solution to the Problem 3 is not tractable, an approximate solution can be found using a lower bound on the mutual information. We refer the reader to [11] for more details on the final lower bound objective.

In the attack step, in order to generate word importance scores from the trained SUB, we take the logit output of the last layer of $\mathcal{E}(X)$ before the k selection process $S \sim \mathcal{E}(X)$. This way we obtain importance score vector $I \in \mathbb{R}^d$ that has the same dimensions of the input.

The selector $\mathcal{E}$ takes the form of a multilayer deep convolutional neural network. In order for $\mathcal{E}$ to select k features from X, a Gumbel-Softmax layer [12] is used to generate k indices.

B. *Adversarial Examples via Cross-Domain Interpretability*

At inference time, when crafting the adversarial attacks, we do inference on SUB using the target testing set D_t^{test}. This is the standard setup for black-box attack, where D_t^{test} is used as starting point and slightly perturbed to craft valid adversarial examples. As described earlier, because of the domain adaption capacity of deep learning models, the closer the substitute domain is to the target, the better the generated scores will be for the original target model.

C. *Implementation*

Our attack method is inspired by framework proposed in [8]. However, we modified the behaviour so that we alleviate the need for querying $F(\cdot)$ for word ranking, since our proposed method relies on interpretability architecture for this purpose. We name our framework *Explain2Attack*. In Algorithms 1 and III-C we describe in details how our algorithm works.

Algorithm 1 Train Substitute Model SUB

```
Input: Substitute training corpus $\mathbb{X}_b = \{X_1, X_2, ..., X_N\}$ of $N$ sentences, and corresponding class labels $\mathbb{Y}_b = \{Y_1, Y_2, ..., Y_N\}$
Output: Trained substitute model SUB with selector network $\mathcal{E}$

for each mini batch $B_s \in \{\mathbb{X}_b, \mathbb{Y}_b\}$ do
    Update parameters of SUB model to find $\mathcal{E}$ that
      maximizes an approximate to objective in Eq. (3)
end for
```

Algorithm 2 Explain2Attack

```
Input: Target test sentence example $X_t^{\text{test}} \in \mathbb{X}_t^{\text{test}}$, where $X_t^{\text{test}} = \{w_1, w_2, \ldots, w_n\}$, the corresponding ground truth label $Y_t^{\text{test}}$, target model $F$, sentence similarity function $Sim(\cdot,\cdot)$, sentence similarity threshold $\epsilon$, word embeddings $Emb$ over the vocabulary $\textbf{Vocab}$
Output: Perturbed Adversarial example $X_{adv}$

 1: Initialization: $X_{adv} \leftarrow X_t^{\text{test}}$
 2: for each word index $\boldsymbol{i}$ in $X_t^{\text{test}}$ do
 3:     Retrieve word importance score:
        $I_{w_i} = \textbf{Get\_Word\_Score}\ (X_t^{\text{test}}, \boldsymbol{i})$
 4: end for

 5: $W \leftarrow$ Sorted list of words in $w_i \in X$ in descending order of importance score $I_{w_i}$

 6: for each word $w_j \in W$ do
 7:     $\textbf{Candidates} \leftarrow \{\}$
 8:     for each nearest synonym word $c_m \in \textbf{Vocab}$ (by embedding cosine similarity) do
 9:         $X'_m \leftarrow$ Replace $w_j$ with $c_m$ in $X_{adv}$
10:         Add $X'_m$ to $\textbf{Candidates}$
11:     end for

12:     if $\exists X' \in \textbf{Candidates}$ s.t. $F(X') \neq Y_t^{test}$ and $Sim\,(X_t^{\text{test}}, X') \geq \epsilon$ then
13:         $X_{adv} \leftarrow \underset{X' \in \textbf{Candidates}}{\text{argmax}} \quad Sim\,(X_t^{\text{test}}, X')$
14:         return $X_{adv}$

15:     else if $F_Y\,(X_{adv}) > \underset{X' \in \textbf{Candidates}}{\min} F_Y(X')$ then
16:         $c^* \leftarrow \underset{X' \in \textbf{Candidates}}{\text{argmin}} \quad F_Y(X')$
17:         $X_{adv} \leftarrow$ Replace $w_j$ with $c^*$ in $X_{adv}$
18:     end if

19: end for
20: return None

21: function GET_WORD_SCORE($X_t, i$)
22:     word_score $= \mathcal{E}_{\text{logits}}(X_t)_i$
23:     return word_score
24: end function
```

We follow Algorithm 1 to train the substitute model SUB using D_b. After that, selector $\mathcal{E}$ from SUB can be used in algorithm III-C for crafting adversarial examples.

The procedure to craft adversarial examples is described in Algorithm III-C. The algorithm starts from an input sentence X_t^{test} and terminates either after successfully finding a perturbed adversarial example X_{adv} that changes the label Y_t^{test}, or if no such perturbation is found. In details, the algorithm proceeds by inferring from $\mathcal{E}$ word scores for every word in X_t^{test}. These scores are sorted, called W, and then the algorithm tries to replace word by word to find valid X_{adv}. For each word w_j in W a set of close candidate synonyms is selected based on embedding cosine similarity as in [8], and a set of sentences **Candidates** is created by replacing w_j with each synonym.

At the current word w_j , the algorithm first checks if perturbing w_j in X_{adv} using the set of synonyms in **Candidates** can change Y_t^{test}. If one or more such synonym are found, the one which achieves the highest similarity $Sim\,(X_t^{\text{test}}, X')$ to original sentence is picked, and the algorithm terminates.

Otherwise, if no synonym is found that can change Y_t^{test}, the algorithm needs to choose the synonym perturbation that yields the weakest (minimum) target model probability $F_Y(X')$ before moving on to the next word w_{j+1}.

IV. Related Work

There has been recent work on adversarial text attacks [2], [3], [6], [4], [5], [7], [13], [8] . The main challenges for natural language adversarial attacks is discrete nature of inputs, where defining meaningful perturbations is not straight forward, and the search space and complexity for black-box attack methods.

Specifically for black-box text attacks, several methods [8], [13], [7] have been developed that share similar general framework, where the attack starts by selecting the most important words/tokens to replace from a candidate sentence, followed by searching for some word replacement that can flip the classification label of the target model. However, some methods followed the heuristic optimization approach, for example, [14] used genetic algorithm to find the best sentence perturbation that fools the classifier.

Most of formerly mentioned black-box methods use the word selection/replacement strategy. For instance, PWWS [13] proposes computing a word saliency score using output probabilities of the target model, while [7] computes sequential importance score based on forward and backward RNN probabilities at the current word position in the sentence. TextFooler [8] is a recent strong baseline for text attacks, where the method uses a modified procedure for word ranking that increases the ranking in label disagreement case. BERT-Attack [15] improves on TextFooler synonym replacement by using a pretrained language model to generate suitable substitute words based on the surrounding context. This achieved higher attack rates and the number of queries are reduced. The improvement in BERT-Attack can be easily incorporated in our framework.

Our approach differs from previous work in solving the word ranking problem. Unlike other methods, instead of depending on the target model for word importance ranking, we *learn* word importance scores. The main differences of our approach compared exiting ones are: i) word ranking has no dependence on the target model output, thus more efficient in number of queries, ii) unlike exiting methods, our approach is scalable with increased sentence lengths, since computing the scores is not dependent on word by word query of target model. This makes our approach more efficient, scalable, and less computationally expensive compared to existing methods. In addition, our approach can be further extended to completely alleviate the need for target model output probabilities, by ranking synonym replacement inside each word based on the substitute model, following more sophisticated substitute training like in [10].

V. Experiments

We report here the results of our method on text classification tasks. We apply our framework to several sentiment classification datasets with WordCNN, WordLSTM and BERT [16] target models. However, our model can be applied to other classification models or datasets with proper choice of substitute datasets . We compare our results to TextFooler [8], a strong state-of-the-art baseline for black-box text attack on the chosen target models. Below we describe the datasets, metrics and discuss the results. In Table I we report the datasets we used in our experiment with their statistics.

Datasets:

- **IMDB** and **MR**: Movie reviews for sentiment classification [17], [18]. The reviews have binary labels, either positive or negative.
- **Amazon MR**: Amazon polarity (binary) user reviews on movies, extracted from the larger Amazon reviews polarity dataset [2].
- **Yelp Polarity Reviews**: Sentiment classification on positive and negative businesses reviews [19]. We mainly use this dataset as a substitute dataset when attacking other models.

In all of the datasets except Amazon MR, we follow the data preprocessing and partitioning in [8].

TABLE I
Statistics of Used Datasets

Dataset	Train	Test	Avg. Length
IMDB	25K	25K	215
MR	9K	1K	20
Amazon MR	25K	25K	100
Yelp	560K	38K	152

Metrics: We evaluate our method using the following metrics:

- **After-attack accuracy** (Adv_Acc): We report for each model the original clean accuracy Clean_Acc of the target model on the test set. Then we report the accuracy of the target model against crafted adversarial examples Adv_Acc. The lower is this better, where the larger the gap between these two accuracies means more successful the attack method. Through-out discussion, when we refer to "***attack rate***", we mean the gap (Clean_Acc − Adv_Acc).
- **Average number of queries** (Avg_Queries)**:** We report the average number of queries needed to find successful adversarial example per input sentence. The lower is better, where lower number of queries is one of the main desired goals of our method. This is an absolute measure of the number of queries regardless of the achieved after-attack accuracy.
- **Query Efficiency** (QE)**:** Since more successful attacks need more queries in black-box setting, we cannot rely only on Avg_Queries for evaluating the performance of our method. We need to measure the true benefit in reduction of number of queries, and make sure that it is not reduced because of sacrificing attack rate. This is the motivation behind Query Efficiency (QE), the ratio of successful attacks per query $QE = \frac{\text{Clean\_Acc}-\text{Adv\_Acc}}{\text{Avg\_Queries}}$. The higher is better. This means that QE measures the percentage of successful attacks per single query, hence the true query efficiency related to the attack rate.
- **Perturbation Query Cost** (PQC): The number of queries needed per perturbed word $PQC = \frac{\text{Avg\_Queries}}{\text{Pertureb\_Words}}$. The lower is better. PQC measures the cost in terms of queries needed for a useful word perturbation that contributes towards label change.

Discussion

The main focus in evaluating the performance of our method is the number of queries, both as absolute measure and its efficiency relative to the achieved attack rate. These two aspects of evaluation are critical to measure the true gains we get from our approach.

In Table II, we compare after-attack accuracies and corresponding average number of queries to the baseline. For each target model and dataset, we report the best substitute dataset that yielded the best result. Across all but one case, we find that our method either achieves or out-performs attack rates of the baseline, yet with lower number of queries. The gains in terms of number of queries differs according to different datasets. We discuss this in details later in this section.

It is important to study the cases where our method achieves lower number of queries but does not achieve higher attack rate than the baseline. In the black-box setting, higher attack rate is related to number of queries, since there is a cost of certain number of queries for every word perturbation. This means that lower number of queries is only beneficial if it improves or preserves the attack rate. Therefore, in order to measure the true efficiency of our method in terms of queries per successful attack, we report in Table II Query Efficiency ratio QE. We find that when our method has lower number of queries, it has better QE ratio, even if the attack rate is not better than the baseline. This implies that the lower number of queries achieved comes from true efficiency of our method, not because of corresponding drop in attack rate.

Effect of Sentence Length: In the case of MR dataset in Table II, we find that the reduction in number of queries is the lowest among other datasets. By comparing the statistics of the datasets (Table I) with the results in Table II, we find a correlation between the dataset average sentence length and the corresponding reduction in attack queries. The longer the sentences are, the more reduction our method achieves in number of queries. We summarize this finding in Table

[2]https://www.kaggle.com/bittlingmayer/amazonreviews

TABLE II
AFTER-ATTACK ACCURACIES, QUERIES AND QUERY EFFICIENCY

Classifier		BERT		WordCNN			WordLSTM		
Target Model		IMDB	MR	IMDB	MR	Amazon MR	IMDB	MR	Amazon MR
	Clean_Acc.	92.18	89.97	87.32	79.85	90.14	88.78	81.82	91.30
Adv_Acc. ↓	TextFooler [8]	11.88	13.59	**0.60**	1.50	**3.92**	**0.04**	**2.06**	**2.15**
	(Substitute Data)	*(Yelp)*	*(Amazon MR)*	*(Yelp)*	*(IMDB)*		*(Amazon MR)*		*(IMDB)*
	Explain2Attack (ours)	**11.32**	**13.34**	0.61	**1.31**	3.97	0.06	2.27	2.38
Avg_Queries ↓	TextFooler	980.5	**181.6**	444	112.8	378.7	500.2	117.5	392.7
	Explain2Attack	**873.5**	184.07	**404.5**	**108.7**	**349.4**	**440.5**	**114.2**	**369.3**
Query Efficiency (QE) ↑	TextFooler	0.082	**0.421**	0.195	0.695	0.228	0.177	0.679	0.227
	Explain2Attack	**0.093**	0.416	**0.214**	**0.723**	**0.247**	**0.201**	**0.697**	**0.241**

TABLE III
EFFECT OF SENTENCE LENGTH ON NUMBER OF QUERIES

Target Dataset		IMDB			Amazon MR		MR		
Classifier		BERT	CNN	LSTM	CNN	LSTM	BERT	CNN	LSTM
Average Sentence Length		215			100		20		
Avg_Queries ↓	TextFooler	980.5	444	500.2	378.7	392.7	112.8	117.5	**181.6**
	Explain2Attack	**873.5**	**404.5**	**440.5**	**349.4**	**369.3**	**108.7**	**114.2**	184.07
Difference		106.5	39.5	59.7	29.3	23.4	4.1	3.3	-3.0

TABLE IV
PERTURBATION QUERY COST (**PQC**)

Classifier		BERT		WordCNN			WordLSTM		
Target Model		IMDB	MR	IMDB	MR	Amazon MR	IMDB	MR	Amazon MR
Max Perturbed Words	TextFooler	222	20	56	14	**35**	89	**13**	75
	(Substitute Data)	*(Yelp)*	*(Amazon MR)*	*(Yelp)*	*(IMDB)*		*(Amazon MR)*		*(IMDB)*
	Explain2Attack	**127**	20	**54.3**	**11**	41.7	**84.7**	13.8	**68**
Avg. Perturbed Words	TextFooler	18.7	4.2	5.8	2.3	5.0	7.6	2.5	7.0
	Explain2Attack	22	4.8	9.1	2.8	6.5	11.3	2.9	8.7
Perturbation Query Cost (PQC) ↓	TextFooler	52.5	43.5	76.6	49.0	76.3	65.8	47.0	56.4
	Explain2Attack	**39.7**	**38.6**	**44.5**	**38.8**	**53.8**	**39.2**	**39.2**	**42.2**

III. This effect is in agreement with our design goals, since unlike other existing methods, our method does not perform word by word ranking through target model querying. Hence, with longer input sentences, the re duction in number of queries is improved by our method. This is a key advantage of our method in terms of scalability to datasets with longer sentences.

Query Cost of Perturbed Words: Another important measure is the number of queries needed for every word perturbation; Perturbation Query Cost (PQC). We are interested in this measure in order to understand the added cost of queries if the model requires more words to be perturbed to find a successful attack. In Table IV we report PQC against the baseline. In addition, we report both the maximum and average number of perturbed words needed for successful adversarial attack. Results show that our method out-performs the baseline in PQC measure, meaning that our method scales better with number of perturbations needed for successful attacks.

Qualitative Assessment: We show in Table V examples of successful adversarial examples from our method. We find that the language semantic is preserved, and that the choice of perturbed words resemble important keywords that contribute to the original label.

Transferability Conditions: In all of our experiments, we used the standard L2X [11] architecture as the choice for the substitute architecture. However, we found that changing the substitute architecture affects attack rates. This finding is similar to [20], [21] where attack performance is found to be related to the model's architecture and complexity. We further look to study in details why and when attacks transfer well between substitute and target models. However, we leave such comprehensive study for future work.

VI. CONCLUSION

We propose a general framework that employs interpretability and domain transfer for crafting black-box text adversarial attacks. The main intuition is to learn word ranking that

TABLE V
ADVERSARIAL EXAMPLES SENTENCES. PERTURBED WORDS ARE HIGHLIGHTED

	Label	Sentence
Original	(0=Negative)	the film lapses too often into sugary **sentiment** and withholds delivery on the pell mell pyrotechnics its punchy style promises
Adversarial	(1=Positive)	the film lapses too often into sugary **emotions** and withholds delivery on the pell mell pyrotechnics its punchy style promises
Original	(1=Positive)	the movie sticks much closer to hornby 's drop dead confessional tone than the film version of high **fidelity** did
Adversarial	(0=Negative)	the movie sticks much closer to hornby 's drop dead confessional tone than the film version of high **faithful** did
Original	(0=Negative)	i'm not exactly sure what this **movie thinks** it is about
Adversarial	(1=Positive)	i'm not exactly sure what this **cinematography concepts** it is about
Original	(1=Positive)	the performances of the four **main** actresses bring their characters to life a **little** melodramatic , but with enough **hope** to keep you engaged
Adversarial	(0=Negative)	the performances of the four **underlying** actresses bring their characters to life a **littlest** melodramatic , but with enough **wanting** to keep you engaged

most probably impacts the target model instead of searching for it expensively. To achieve this, we train an interpretable substitute model on a substitute dataset, with no need for the target dataset. Thus, the substitute model learns importance scores with less number of queries and higher efficiency. Results show that our method reduces queries cost for attacking text classification models, while achieving or out-performing the state-of-the-art attack rate. We show that our method is superior in both query cost of perturbations and with longer input sentences, which allows our method to scale better with longer sentences.

For future work, we plan to overcome the current limitations in our current framework. Specifically, we plan to *i)* train the selector network to directly fool the target classifier by leveraging its output information, in order to improve attack rate, *ii)* study and formalize the transferability conditions from substitute to target domains, and provide guides for choice of suitable substitute models, and *iii)* further reduce dependence on target model by learning replacement synonyms through a pretrained language model.

REFERENCES

[1] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2014.

[2] Y. Belinkov and J. R. Glass, "Analysis methods in neural language processing: A survey," *CoRR*, vol. abs/1812.08951, 2018. [Online]. Available: http://arxiv.org/abs/1812.08951

[3] W. Wang, B. Tang, R. Wang, L. Wang, and A. Ye, "Towards a robust deep neural network in texts: A survey," *CoRR*, vol. abs/1902.07285, 2019. [Online]. Available: http://arxiv.org/abs/1902.07285

[4] P. Yang, J. Chen, C.-J. Hsieh, J.-L. Wang, and M. I. Jordan, "Greedy attack and gumbel attack: Generating adversarial examples for discrete data," 2019.

[5] M. Sato, J. Suzuki, H. Shindo, and Y. Matsumoto, "Interpretable adversarial perturbation in input embedding space for text," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*. International Joint Conferences on Artificial Intelligence Organization, 7 2018, pp. 4323–4330.

[6] V. Kuleshov, S. Thakoor, T. Lau, and S. Ermon, "Adversarial examples for natural language classification problems," 2018.

[7] J. Gao, J. Lanchantin, M. L. Soffa, and Y. Qi, "Black-box generation of adversarial text sequences to evade deep learning classifiers," in *2018 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2018, pp. 50–56.

[8] D. Jin, Z. Jin, J. T. Zhou, and P. Szolovits, "Is BERT really robust? natural language attack on text classification and entailment," *CoRR*, vol. abs/1907.11932, 2019.

[9] P. Vijayaraghavan and D. Roy, "Generating black-box adversarial examples for text classifiers using a deep reinforced model," 2019.

[10] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, 2017, pp. 506–519.

[11] J. Chen, L. Song, M. Wainwright, and M. Jordan, "Learning to explain: An information-theoretic perspective on model interpretation," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 80. PMLR, 10–15 Jul 2018, pp. 883–892.

[12] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with Gumbel-Softmax," *ArXiv e-prints*, vol. abs/1611.01144, 2016. [Online]. Available: http://arxiv.org/abs/1611.01144

[13] S. Ren, Y. Deng, K. He, and W. Che, "Generating natural language adversarial examples through probability weighted word saliency," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 1085–1097.

[14] M. Alzantot, Y. Sharma, A. Elgohary, B.-J. Ho, M. Srivastava, and K.-W. Chang, "Generating natural language adversarial examples," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, Oct.-Nov. 2018, pp. 2890–2896. [Online]. Available: https://www.aclweb.org/anthology/D18-1316

[15] L. Li, R. Ma, Q. Guo, X. Xue, and X. Qiu, "Bert-attack: Adversarial attack against bert using bert," 2020.

[16] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[17] A. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, 2011, pp. 142–150.

[18] B. Pang and L. Lee, "Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales," *arXiv preprint cs/0506075*, 2005.

[19] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Advances in neural information processing systems*, 2015, pp. 649–657.

[20] A. Demontis, M. Melis, M. Pintor, J. Matthew, B. Biggio, O. Alina, N.-R. Cristina, and F. Roli, "Why do adversarial attacks transfer? explaining transferability of evasion and poisoning attacks," in *28th {USENIX} Security Symposium ({USENIX} Security 19)*, vol. 28. {USENIX} Association, 2019.

[21] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *International Conference on Learning Representations*, 2018.