ATTEXPLAINER: Explain Transformer via Attention by Reinforcement Learning

Runliang Niu[1] , **Zhepei Wei**[1] , **Yan Wang**[1,2*] and **Qi Wang**[1*]

[1]School of Artificial Intelligence, Jilin University

[2]Key Laboratory of Symbol Computation and Knowledge Engineering of Ministry of Education, College of Computer Science and Technology, Jilin University

{niurl19,weizp19}@mails.jlu.edu.cn, wy6868@jlu.edu.cn, wangqi_jlu@foxmail.com

Abstract

Transformer and its variants, built based on attention mechanisms, have recently achieved remarkable performance in many NLP tasks. Most existing works on Transformer explanation tend to reveal and utilize the attention matrix with human subjective intuitions in a qualitative manner. However, the huge size of dimensions directly challenges these methods to quantitatively analyze the attention matrix. Therefore, in this paper, we propose a novel reinforcement learning (RL) based framework for Transformer explanation via attention matrix, namely ATTEXPLAINER. The RL agent learns to perform step-by-step masking operations by observing the change in attention matrices. We have adapted our method to two scenarios, perturbation-based model explanation and text adversarial attack. Experiments on three widely used text classification benchmarks validate the effectiveness of the proposed method compared to state-of-the-art baselines. Additional studies show that our method is highly transferable and consistent with human intuition. The code of this paper is available at https://github.com/niuzaisheng/AttExplainer.

1 Introduction

With the development of deep learning methods in natural language processing (NLP), model explanation has been increasingly investigated to interpret the black-box model in an intelligible manner. As a recent representative neural network architecture, Transformer [Vaswani and others, 2017] is leading the current NLP studies. For instance, the Transformer-based BERT [Devlin and others, 2019] is believed to start a groundbreaking era and has been proven effective in many downstream tasks. Basically, it comprises multiple stacked layers of self-attention and nonlinear feed-forward modules. As the key to the explanation of the attention mechanism, the attention matrix (also known as attention weight) is particularly investigated in previous research. It has been used to reveal the latent semantic relationships between input tokens [Hewitt and Manning, 2019], or find some regular patterns in attention matrix images [Kovaleva *et al.*, 2019].

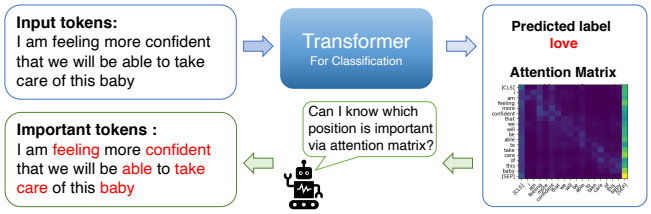

Figure 1: Demonstration of the sentence classification task with Transformer model. It is highly desired to find the key/important input tokens via the attention matrix. Note that important tokens are marked in red.

Typically, the attention matrix is regarded as an inherent explanation for Transformer. Since [Jain and Wallace, 2019] threw a question that the relationship between attention weights and model outputs is unclear, researchers have begun to investigate how to explain attention matrices and harvest the potential information in them. Though widely discussed in previous works, none of them address the open problem as presented in Figure 1. In particular, there are two major issues: (1) Most existing perturbation-based interpretation methods use predict probabilities as the basis for interpretation. It is still an open question whether some other inherent information of the model, such as attention matrix, can be employed as a basis to explain the model's predictions; (2) Not just at the level of analysis, it is still under-explored how to extract potential information from attention matrix for benefiting the downstream tasks, such as model explanation and model adversarial attack.

On the other hand, *Perturbed Masking* is an unsupervised method to measure the dependency degrees between input tokens [Wu *et al.*, 2020]. By successively masking token pairs in all pairwise combinations and sorting the degree of changes before and after the masking operations, the probing method can draw an unlabeled dependency tree from the input sentence. In light of this, we argue that the attention matrix could be explained by observing the changes introduced by the masking operation. Specially, we try to extend the possibility from pair to an unlimited number of token masking combinations. However, this will lead to an explosion of combination spaces. Some previ-

* Corresponding authors

ous methods used greedy search, such as beam search or genetic algorithms to select crucial positions [Ren *et al.*, 2019; Li *et al.*, 2019]. But the time cost of searching is still high. Nonetheless, there is still a lack of this kind of method and an effective model is needed.

To address the aforementioned issues, we introduce a reinforcement learning (RL)-based method for perturbation sampling and build a game environment for two downstream tasks: model explanation and text adversarial attack. Perturbation sampling is required in both tasks. Compared to other combination search algorithms, the RL is more efficient to model the Markov Decision Process when searching in huge action space. Meanwhile, the attention matrix can be considered as a kind of input observation for the RL agent for further efficiencies. Specifically, in each game step, a deep Q-network (DQN) based agent observes the attention matrix, then selects a position to change the token's visibility to the Transformer model by masking or unmasking the token. The main contributions are summarized as follows:

- We propose an RL-based model interpretation method named ATTEXPLAINER to explain the Transformer via attention matrix quantitatively by finding the key/important tokens in a sentence. Especially, attention matrices are employed to provide heuristic information to speed up the process.
- We prove that the attention matrix does contain information about token importance. Also, we demonstrate the strong transferability of our method within different well-trained Transformer models and datasets.
- We conduct extensive experiments on three public datasets, demonstrating the interpretability of the attention matrix on two popular downstream tasks, namely, model explanation and model adversarial attack. In addition, we perform case studies to visualize the step-by-step masking process for better understanding.

2 Related Work

2.1 Model Explanation

Typically, model explanation aims to accurately deliver the true reasons behind the model's decision/prediction by identifying which parts of the input are important. The two dominant factions are gradient-based and perturbation-based. In this article, we focus on perturbation-based explanation methods. As introduced in [Ribeiro *et al.*, 2016], the goal of LIME is to use a smaller and simpler model to explain the original big model within a local scope of an input sample. [Lundberg and Lee, 2017] proposed the SHAP framework, which assigns each feature an important value for one particular prediction. The Shapley value of a feature is the average marginal contribution of the feature in all features. Different methods are proposed for the model explanation, but how to uniformly define model interpretation is still an open question [Feng and others, 2018].

2.2 Attention Explanation

Attention matrices, which are considered important inherent information from Transformer models, have been extensively studied to interpret model predictions in NLP. For example, some research works tried to study whether the attention matrix is interpretable by designing perturbation experiments on model parameters [Jain and Wallace, 2019; Serrano and Smith, 2019; Wiegreffe and Pinter, 2019]. [Kovaleva *et al.*, 2019] summarized several BERT's self-attention patterns through manual observation. The model proposed in [Htut *et al.*, 2019] compared the consistency of attention heads and human-labeled syntactic dependency trees. In [Abnar and Zuidema, 2020], the authors proposed a method to track the flow of information through self-attention layers to better explain the attention matrix. However, there is still a lot of controversy about whether attention can indeed deliver important information about the input tokens. [Chefer *et al.*, 2021] proposes a gradient-based method for interpreting the attention matrix, which considers both the gradient of the attention matrix and the relevancy score between tokens reflected by the attention matrix.

2.3 Model Adversarial Attack

Adversarial attack of text model is a fast-developing research direction in the field of NLP. Normally, the attack process has two steps: finding the key positions to conduct the perturb operations, then generating the replacement text to these positions. The adversarial attacker usually utilizes different types of information as its observation, such as gradient, probability, prediction decision, or value of loss function [Papernot *et al.*, 2016; Li *et al.*, 2019]. Some automated model adversarial attack toolkits integrate existing methods [Wallace and others, 2019; Zeng and others, 2021]. As far as we know, there is still no previous work that uses the attention matrix as the observation.

2.4 Reinforcement Learning (RL) in NLP

There have already been various NLP tasks that are solved by the Reinforcement Learning (RL) techniques, such as question answering [Xiong *et al.*, 2018], dialogue generation [Li *et al.*, 2016b], and so on. In general, the RL agent model does not carry any language knowledge on itself. It is generally used for auxiliary tasks, such as ensuring the output is properly formatted [Zhong *et al.*, 2017]. Some recent works focused on using RL methods to improve the performance of NLP models. [Li *et al.*, 2016a] proposed an RL-based method by removing the minimum number of words to change the model's prediction. In [Xu *et al.*, 2019], the authors improved the robustness of the text classification model by using RL to generate semantically similar samples. Also, RL can be used to generate adversarial examples to improve the robustness of machine translations [Zou *et al.*, 2020].

3 Framework

In this section, we introduce the technical details of the proposed ATTEXPLAINER framework. The overview is shown in Figure 2.

3.1 Transformer Model

In our task setting, a Transformer classifier is used as a model being explained or a victim model for the adversarial attack.

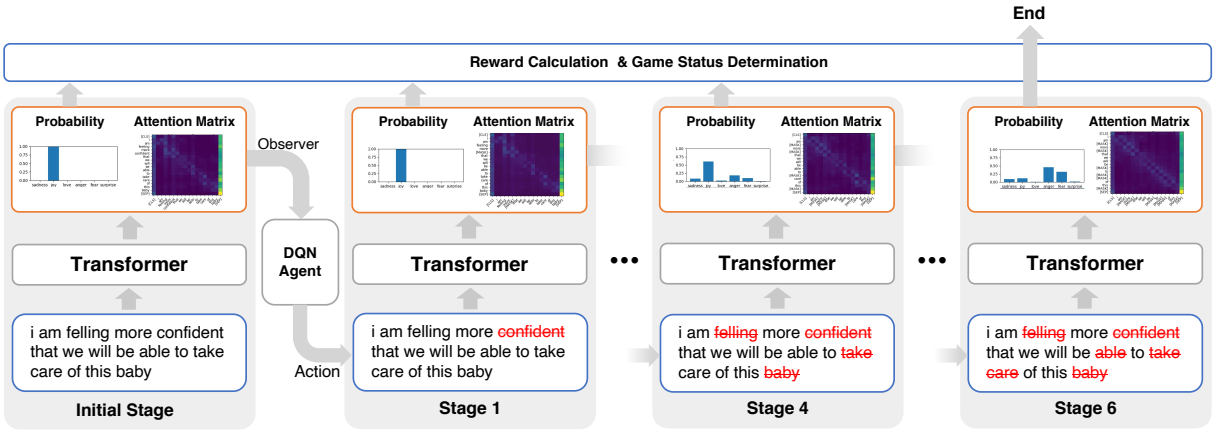

Figure 2: Overview of the explanation game process. ATTEXPLAINER identifys the key tokens by a step-by-step masking operation.

Given an input token sequence $X = (x_1, ..., x_n)$, where n is the length of the sequence. We use a well-trained BERT model as an instance of the Transformer, which is formalized as a function $T(\cdot)$. $\hat{y}$ is the model's predicted label with $\hat{y} = T(X)$. $p(\hat{y}|X)$ is the output probability for label $\hat{y}$. The cross-entropy loss for classification is denoted as $L(\hat{y}, y)$, where y is the golden label.

Since BERT adopts dot-product style attention, in each layer, the score of the i-th token x_i in the attention matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ is computed by a similarity function $\phi(Q, K)$ on the input query $\mathbf{Q} \in \mathbb{R}^{n \times dim}$ and key $\mathbf{K} \in \mathbb{R}^{n \times dim}$ followed by an activation layer of softmax:

$$A_{i,j} = softmax(\phi(Q_j, K_i)) = \frac{exp(\phi(Q_j, K_i))}{\sum_t exp(\phi(Q_j, K_t))} , \quad (1)$$

where the similarity function is $\phi(Q, K) = Q \times K^T / \sqrt{dim}$, and dim is the hidden dimension of Transformer. The attention matrix that averaged over all layers is $Att \in \mathbb{R}^{(n,n)}$.

3.2 RL Game Environment

The environment treats one sample's analysis process as a game round. There are two visibility for a single token to the model: masked or unmasked. For a complete sequence, all tokens' status can be combined into one boolean mask sequence $e_t = (e_{(t,1)}, ..., e_{(t,n)}), e_{(t,i)} \in \{True, False\}$, in which $e_{(t,i)} = True$ means token i is visible to the Transformer model at step t. Applying mask sequence e on token sequence X is formalized as $X \star e$. So the input sequence to the model at step t is $X_t = X \star e_t$. In the initial step, e.g., $t = 0$, all $e_{(0,i)}$ is set to 1.

Environment. The environmental state s is composed of the current token masking state e_t and attention matrix Att_t. A flag bit d indicates whether the game is over or not.

The model interpretation aims to find an optimal combination of token masks that maximizes the reduction of the model's output probability on a given label. We denote this reduction in probability as $\Delta p = p(\hat{y}|X) - p(\hat{y}|X')$. The game ends when Δp is reached to a certain threshold ε.

The adversarial attack aims to find an adversarial sample X' as quickly as possible to flip the output of the model. The game ends when $T(X) \neq T(X')$, which means the attack action is successful.

Action. The agent action is denoted as an integer a_t, which indicates the next position index to inverse. The action of step t will affect the input of the next step, $e_{(t+1,a_t)}$ is the reverse of $e_{(t,a_t)}$. In this environment, the agent is encouraged to select actions that can obtain the largest overall rewards. In order not to affect the text classification task-specific sequence input format, it is not allowed to choose positions of special tokens, such as [CLS], [SEP], and [PAD].

Reward. The environment reward is made up of three aspects of the game: the model classification loss or label's probability, token mask rate $m_t = \frac{1}{n} \sum_n e_t$, and game status d_t.

In the model interpretability task, we design the reward function as Eq.(2). In the adversarial attack task, the reward function is Eq.(3).

$$r_t = \Delta p + \alpha \times d_t \times (1 - m_t) - \beta , \quad (2)$$

$$r_t = (L_t - L_0) + \alpha \times d_t \times (1 - m_t) - \beta , \quad (3)$$

where α is the final reward for game completion and β is time punishment. The reward functions encourage a lower token mask rate and lower time cost.

3.3 DQN Agent

We adopt a deep Q-learning network (DQN), denoted as $Q(s, a)$, to predict the state-action value of expected return at each step and the goal is to maximize $\mathcal{J} = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t]$, where γ is the discount factor.

As presented in Figure 3, we extract numerical distributions and statistical features by performing histogram statistics along rows and columns from the attention matrix Att. Specifically, We fix the bin size of the histogram to b. For an input sentence with length n, we compute histograms along the two dimensions separately, and the value is divided by n to normalize. Stitch them together with the mean and variance of the rows and columns, resulting in a statistical feature matrix F in the shape of $n \times (2b + 4)$. $OutputLayer$ and $HiddenLayer$ are two linear layers. Therefore, the structure of the value network is:

$$Q(s_t) = OutputLayer([HiddenLayer(F_t), e_t]) . \quad (4)$$

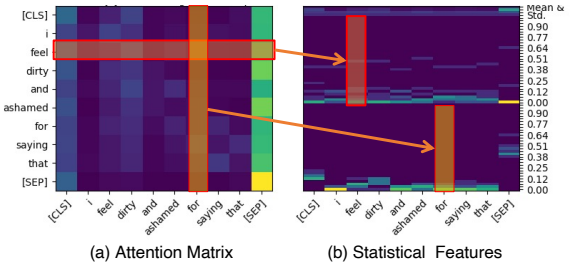

Figure 3: An example of extracting statistical features. The operation indicated by the arrow converts a vector of variable length n into a vector of fixed dimension b using numerical histogram statistics.

The ϵ-greedy policy is adopted to balance the exploration-exploitation, in which the action $a_t = argmax\ Q(s_t)$ is either determined with probability ϵ, or a random position is selected as action with probability $1 - \epsilon$. The agent chooses one position to change its status in each game step.

We use prioritized experience replay (PER) [Schaul *et al.*, 2016] to store transitions and sample DQN training batch. The PER importance-sampling weights for one sampling example is indicated by w. Therefore, the overall TD loss is:

$$L_Q(s_t, a_t, r_t, s_{t+1}, a_{t+1}) = \\ w \times [r_t + \gamma \times (1 - d_t) \times Q'(s_{t+1}, a_t) - Q(s_t, a_t)] \ . \quad (5)$$

The detailed of DQN training progress is presented in Algorithm 1.

Algorithm 1 DQN training progress

Input: Transformer model T, input token sequence X.
Variables: DQN eval net Q and target net Q', agent replay buffer, maximum game steps M, value net parameters replace frequency C.

1: Initialize token sequence $X_0 = X$, mask sequence e_0.
2: Compute $\hat{y}_0 = T(X_0)$, get Att_0, L_0, d_0, then $s_0 = [Att_0, e_0, d_0]$.
3: Set current step $t = 1$.
4: **while** $t < M$ and $d_{t-1} \neq True$ **do**
5: The DQN agent gives an action a_t by observing s_{t-1}.
6: Update the environment stage e_t with a_t, then $X_t = X \star e_t$.
7: Compute $\hat{y}_t = T(X_t)$, get Att_t, L_t. Set $d_t = True$ if the end condition has been reached. Compose new environment stage $s_t = [Att_t, e_t, d_t]$. Compute the action reward r_t.
8: Store $(s_{t-1}, a_{t-1}, r_t, s_t, a_t)$ into replay buffer.
9: Sample a mini-batch of transitions using PER from replay buffer. Perform one step training on Q.
10: Reset target net with the parameters of A value net $Q' \leftarrow Q$ every C steps.
11: **end while**

Output: The DQN value net Q.

Settings	Dataset	# Labels	# Tokens	# Train	# Test	Model	Accuracy
Single Sentence	Emotion	6	22	2000	2000	EM-1	0.935
						EM-2	0.937
	SST2	2	train 13 eval 25	67349	872	SSTM	0.924
NLI	SNLI	3	26	549367	9842	SNLIM	0.905

Table 1: Datasets statistics and the well-trained model performance.

4 Experiment

In this section, we conduct extensive experiments to verify the effectiveness of the proposed method in terms of two different tasks, namely, model explanation and adversarial attack. More precisely, we focus on investigating the following research questions (RQs):

- **RQ1**: How does the proposed model perform in terms of model explanation and text adversarial attack tasks?
- **RQ2**: Does the attention matrix necessarily deliver the key information of input tokens?
- **RQ3**: Are parameters of trained agents transferable within different Transformer models and datasets?

4.1 Experimental Settings

Datasets. We used two types of text classification settings: single sentence classification and Natural Language Inference (NLI). Specifically, single sentence classification datasets include the Emotion dataset [Saravia *et al.*, 2018] and the Stanford Sentiment Treebank (SST2) dataset [Wang and others, 2019]. The NLI dataset we used is the SNLI corpus [Bowman and others, 2015]. The details of these datasets are presented in Table 1. We utilize the fine-tuned model pretrained by Huggingface [Wolf *et al.*, 2019] [1].

Baselines. For model interpretation scenarios, we selected five classical approaches for comparison: FeatureAblation, Occlusion [Zeiler and Fergus, 2014], KernelShap [Lundberg and Lee, 2017], Shapley [Castro *et al.*, 2009] and LIME [Ribeiro *et al.*, 2016]. These methods are implemented by Captum toolkit [Kokhlikyan *et al.*, 2020].

For textual adversarial attack scenarios, we selected the following methods for comparison: PWWS [Ren *et al.*, 2019], Genetic [Alzantot and others, 2018], SCPN [Iyyer *et al.*, 2018], HotFlip [Ebrahimi *et al.*, 2018], GAN [Zhao *et al.*, 2018], DeepWordBug [Gao *et al.*, 2018], TextBugger[Li *et al.*, 2019], PSO [Zang *et al.*, 2020], BERTAttacker [Li *et al.*, 2020]. These attack methods are implemented by OpenAttack [Zeng and others, 2021] toolkit.

Evaluation Metrics. We use 4 indicators to evaluate the model explanation methods: Fidelity, Model Query Times, Δp and Token Masked Rate. We use Fidelity to measure the effectiveness of the masking operation:

$$Fidelity = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}(T(X_i) \neq T(X_i')) \ . \quad (6)$$

[1] We use *BertForSequenceClassification* as the base model architecture. The names of the well-trained model are EM-1: *AdapterHub/bert-base-uncased-pf-emotion*, EM-2: *nateraw/bert-base-uncased-emotion*, SSTM: *textattack/bert-base-uncased-SST-2* and SNLIM: *textattack/bert-base-uncased-snl*. These well-trained models have reached a high accuracy on the corresponding datasets, the average evaluation accuracy is shown in Table 1.

Dataset	Method	Fidelity↑	Model Query ↓ Times	Δ p ↑	Token Masked ↓ Rate
Emotion	FeatureAblation	0.857	22.9	0.789	0.845
	Occlusion	0.859	20.9	0.795	0.935
	KernelShap	0.896	100	0.832	<u>0.658</u>
	Shapley(5 times/token)	0.916	111.3	0.849	0.737
	Shapley(25 times/token)	0.918	2212.4	0.853	0.802
	LIME	0.952	100	0.858	0.669
	AttExplainer (Ours)	0.918	6.9	0.836	**0.287**
SST2	FeatureAblation	0.533	26.6	0.501	0.863
	Occlusion	0.510	24.5	0.486	0.915
	KernelShap	0.728	100	0.672	**0.602**
	Shapley(5 times/token)	0.836	129.0	0.768	0.628
	Shapley(25 times/token)	0.883	2558.8	0.818	0.646
	LIME	0.794	100	0.739	0.619
	AttExplainer (Ours)	0.841	25.1	0.697	<u>0.605</u>
SNLI	FeatureAblation	0.737	28.6	0.666	0.683
	Occlusion	0.739	26.6	0.649	0.846
	KernelShap	0.900	100	0.792	0.602
	Shapley(5 times/token)	0.930	128.6	0.818	0.633
	Shapley(25 times/token)	0.967	2749.3	0.849	0.668
	LIME	0.921	100	0.809	<u>0.591</u>
	AttExplainer (Ours)	0.908	17.1	0.808	**0.408**

Table 2: The model explanation experimental results. The ↑ arrow means higher is better. The **best** and <u>second-best</u> scores are marked in the column of Mask Token Rate. The sliding window of Occlusion is fixed at 3. The maximum number of samples for KernelShap and LIME is fixed at 100. Two records of the Shapley method are shown, they are the results of perturbing each token 5 times and 25 times, respectively.

We adopt several commonly used metrics in model adversarial attack scenarios: Attack Success Rate, Victim Model Query Times, and Word Modification Rate.

Implementation Details. We trained our model on multiple Titan RTX graphics cards. The max game step is limited to 100. Max size of replacing buffer for PER is 100,000. The Adam optimizer [Kingma and Ba, 2015] was used for training the DQN model at a fixed learning rate 10^{-4}, training batch size is 256. We set $\alpha = 10, \beta = 0.2, \epsilon = 0.7, \gamma = 0.9$. The number of feature bins b is fixed at 32. The parameters of the DQN target net are replaced by the eval net every 100 learning steps. We use the special token [MASK] to replace the original token in masking operation.

4.2 Experimental Results

Model Explanation Results (RQ1)

The performance of ATTEXPLAINER serving as a model explainer is presented in Table 2. It can be observed that our method achieves surprisingly competitive performance in terms of Fidelity and Δp with a much lower Token Masked Rate. We argue that, given two model interpretation methods, if their performances in Δp are comparatively close, while one of them can find fewer key positions (i.e., with lower Token Masked Rate) that determine the classification result, it means the corresponding method is more effective than the other one. For example, according to the experimental results on the Emotion dataset, the LIME tends to explain that 66.9% of tokens in a sentence are supporting the classification decision. In contrast, the results of the ATTEXPLAINER show that we only need 28.7% of the tokens to be masked while keeping a comparable Δp of 83.6%, indicating the superiority of the proposed method in finding more precise key positions. Meanwhile, we can also find that our proposed method spends less query time to achieve a better balance between Δp and Token Masked Rate, surpassing all the baselines. Note that our approach does not perform optimally on the SST2 dataset, which is also challenging for all baselines. Though the Shapley method achieves a higher Δp metric, it is exponential in time cost, but the marginal benefits will diminish when increasing query times. Overall, compared to existing methods on all three datasets, the ATTEXPLAINER generally makes the searching both faster and more accurate, and we attribute this to the design of introducing the attention matrix as heuristic information in our method.

Dataset	Method	Attack Succ. ↑ Rate	Victim Model ↓ Query Times	Word Modi. ↓ Rate
Emotion	PWWS	0.750	132.80	0.114
	Genetic	0.834	1309.80	0.141
	SCPN	0.764	11.760	1.340
	HotFlip	0.675	49.44	0.305
	GAN	0.757	2.76	1.610
	DeepWordBug	0.670	21.50	0.379
	TextBugger	<u>0.953</u>	17.00	0.197
	PSO	0.740	88.00	0.126
	BERTAttacker	0.880	41.30	0.135
	AttExplainer (Ours)	**0.968**	4.86	0.155
SST2	PWWS	0.724	118.69	0.145
	Genetic	0.810	1431.9	0.164
	SCPN	0.666	11.7	1.430
	HotFlip	0.469	57.2	0.274
	GAN	0.429	2.43	1.300
	DeepWordBug	0.494	23.2	0.275
	TextBugger	0.804	34.0	0.293
	PSO	0.538	147.7	0.151
	BERTAttacker	<u>0.881</u>	62.96	0.190
	AttExplainer (Ours)	**0.883**	19.6	0.442
SNLI	PWWS	0.765	99.2	0.240
	Genetic	<u>0.883</u>	1109.0	0.283
	SCPN	0.473	11.5	1.114
	DeepWordBug	0.354	17.7	0.334
	TextBugger	0.729	44.4	0.311
	PSO	0.667	105.9	0.244
	BERTAttacker	0.859	45.3	0.219
	AttExplainer (Ours)	**0.896**	14.2	0.223

Table 3: The model adversarial attack experimental results.

Adversarial Attack Results (RQ1)

The results of the model adversarial attack are shown in Table 3. Overall, the proposed ATTEXPLAINER achieved the highest attack success rate, relatively lower victim model query times and word modification rate in three datasets compared to baseline methods. More precisely, for the single sentence setting, our method surpasses the second runner-ups with only 1.5% and 0.2% absolute increment in terms of attack success rate on both Emotion and SST2 detests, individually. In contrast, there is a comparatively significant performance boost of 13.1% between the existing state-of-the-art method (e.g. PWWS) and our method in the NLI setting, indicating the superiority of our proposed model in more complicated scenarios.

Interpretability of Attention Matrix (RQ2)

We conduct extensive experiments to validate the effectiveness of the proposed method in providing interpretable information based on the attention matrix. Specifically, we generate a random matrix as environment observation to replace the real attention matrix in each game step, while other settings are kept unchanged.

As can be observed from Figure 4, compared to the normal training setting (Emotion@1M), the agents represented by the remaining two curves fall into a poor solution that masks most

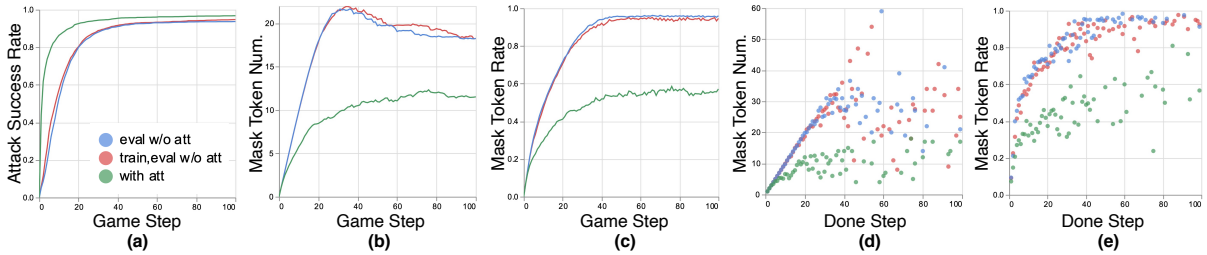

Figure 4: Impact of the introduction of the attention matrix as an observation on the adversarial attack task. • *with att* means the attention matrix can be observed in both the training and evaluation phases. • *eval w/o att* means the attention matrix cannot be observed during evaluation phases. • *train,eval w/o att* means the attention matrix cannot be observed during both training and evaluation phases. From the perspective of attack success rate, shown in (a), there seems to be little difference between the final results of these three settings. But from the perspective of masked token number and masked token rate, as shown in (b)(c)(d)(e), the agent will be more greedy to mask more tokens without attention's heuristic information.

Train \ Eval	Emotion attack for EM-1	SST2 attack for SSTM	SNLI attack for SNLIM	Emotion attack for EM-2
Emotion @ 3M attack for EM-1	0.969 5.220 0.187	0.695 32.19 0.465	0.794 16.51 0.194	0.942 9.830 0.241
SST2 @ 3M attack for SSTM	0.953 4.340 0.183	0.883 19.50 0.484	0.969 10.20 0.209	0.983 6.930 0.276
SNLI @ 3M attack for SNLIM	0.944 6.460 0.306	0.880 20.45 0.506	0.930 12.450 0.236	0.977 6.782 0.289

Table 4: Transferability study on the adversarial attack task. The values in each table cell are attack success rate, victim model query times, and token modification rate.

of the tokens to let the decision flip. On one hand, this reveals that there is potential token importance information in the attention matrix guiding the agent to choose a better gaming strategy. On the other hand, it also highlights the necessity of introducing the RL method due to the difficulty in finding a reasonable masking combination in the huge searching space without any textual information, other model-related observation, or external knowledge.

Transferability Study (RQ3)

Since the decision-making process of the agent does not refer to the text information, we studied whether the agent's parameters can be transferred to other well-trained models. In the scenario of the adversarial attack, the transferability result is shown in Table 4. Agents trained in three training settings will be tested in four different evaluation settings. The table shows that, in a simpler dataset (Emotion), all settings can achieve well results. But the agent which is trained on a more difficult dataset (SST2) has a better transferability to other datasets. There is also a significant effect when migrating to different well-trained models, from EM-1 to EM-2.

4.3 Case Study

In this section, we give some practical examples for analysis. For the same input, we use two marking colors to indicate the different positions found by the explainer and attacker. The examples are shown in Table 5. In most cases, the key positions found by the explainer and attacker overlap with each other. The explainer found more tokens than the attacker. This coincides with the training aims of both. We find that the explainer prefers to choose content words (e.g., nouns, verbs), while the attacker tends to choose function words (e.g., prepositions, articles) to finish faster. Such results inspire us to consider whether these pre-trained language models are indeed robust.

Dataset	Golden label	Original label	Attacked Label	Δp
	Sentence			
Emotion	anger	anger	joy	0.979
	[CLS] i am just feeling cranky and blue [SEP]			
	joy	joy	anger	0.861
	[CLS] i can have for a treat or if i am feeling festive [SEP]			
SST2	neg.	pos.	neg.	0.765
	[CLS] this one is definitely one to skip, even for horror movie fanatics. [SEP]			
	neg.	neg.	pos.	0.875
	[CLS] it offers little beyond the momentary joys of pretty and weightless intellectual entertainment . [SEP]			
SNLI	entail.	entail.	contr.	0.883
	[CLS] a man selling donuts to a customer during a world exhibition event held in the city of angeles [SEP] a woman drinks her coffee in a small cafe. [SEP]			
	entail.	entail.	contr.	0.860
	[CLS] a young boy in a field of flowers carrying a ball [SEP] dog in pool [SEP]			
	entail.	entail.	neutral	0.996
	[CLS] families waiting in line at an amusement park for their turn to ride. [SEP] people are waiting to see a movie. [SEP]			

Table 5: Some case study examples. Tokens in red color are masked by agent as an attacker. The highlighted tokens are identified by agent as an explainer.

Two visualization examples are shown in Figure 5. At each time step, we plotted a set of charts, each chart containing four subplots: attention matrix, label prediction probability, heat-map of extracted features, and Q value distribution. The agent chooses the next mask position based on the maximum Q value.

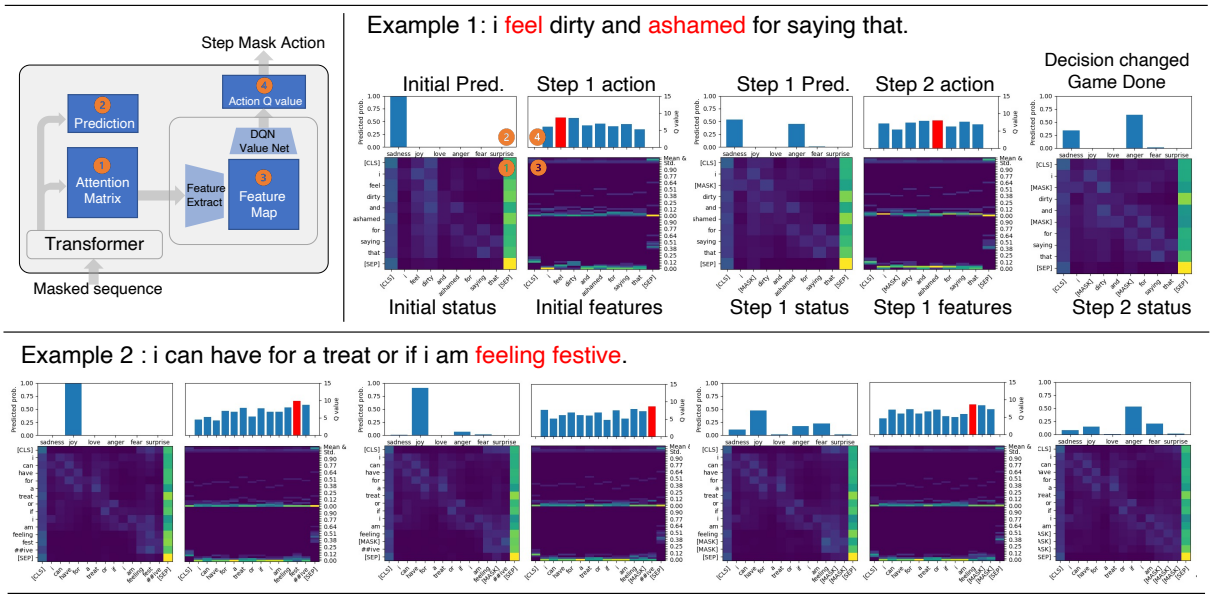

Figure 5: Visualization of the game process in adversarial attack scenario. The original image of BERT attention matrix in subplot is marked by ①, output probability of label classification is marked by ②, the numerical statistical features is marked by ③, and Q value as the measure for next action selection is marked by ④.

5 Conclusion

In this paper, we present a reinforcement learning-based method, named ATTEXPLAINER, for the Transformer explanation. Especially, a deep Q-network (DQN) based agent is designed to select the crucial position in a sentence step-by-step, with the benefit of observing attention matrices. By designing different reward functions, we have tested our approach in two tasks: perturbation-based model explanation and text adversarial attack. In addition, we found that the parameters of the DQN agent can be migrated within the same Transformer architecture. Through comparative experiments, we believe that the attention matrix can provide heuristic information for the agent to speed up the task. We also perform in-depth case studies to visualize the game processes intuitively. In line with human's intuition, we discover that the attention matrix indeed contains potential token importance information, which can be used as a basis for various applications, such as model explanation and model attack. These can be the direction of our future work.

Acknowledgments

This work has been funded by the National Natural Science Foundation of China (No.62072212), the Development Project of Jilin Province of China (20200401083GX, 20200403172SF).

References

[Abnar and Zuidema, 2020] Samira Abnar and Willem Zuidema. Quantifying attention flow in transformers. In *ACL*, 2020.

[Alzantot and others, 2018] Moustafa Alzantot et al. Generating natural language adversarial examples. In *EMNLP*, 2018.

[Bowman and others, 2015] Samuel R. Bowman et al. A large annotated corpus for learning natural language inference. In *EMNLP*, 2015.

[Castro *et al.*, 2009] Javier Castro, Daniel Gómez, and Juan Tejada. Polynomial calculation of the shapley value based on sampling. *Computers & Operations Research*, 2009.

[Chefer *et al.*, 2021] Hila Chefer, Shir Gur, and Lior Wolf. Transformer interpretability beyond attention visualization. In *CVPR*, 2021.

[Devlin and others, 2019] Jacob Devlin et al. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019.

[Ebrahimi *et al.*, 2018] Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. HotFlip: White-box adversarial examples for text classification. In *ACL*, 2018.

[Feng and others, 2018] Shi Feng et al. Pathologies of neural models make interpretations difficult. In *EMNLP*, 2018.

[Gao *et al.*, 2018] Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. Black-box generation of adversarial text sequences to evade deep learning classifiers. In *2018 IEEE Security and Privacy Workshops (SPW)*, 2018.

[Hewitt and Manning, 2019] John Hewitt and Christopher D. Manning. A structural probe for finding syntax in word representations. In *NAACL-HLT*, 2019.

[Htut *et al.*, 2019] Phu Mon Htut, Jason Phang, Shikha Bordia, and Samuel R. Bowman. Do attention heads in bert track syntactic dependencies? *arXiv preprint arXiv: 1911.12246*, 2019.

[Iyyer *et al.*, 2018] Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. Adversarial example generation with syntactically controlled paraphrase networks. In *NAACL-HLT*, 2018.

[Jain and Wallace, 2019] Sarthak Jain and Byron C. Wallace. Attention is not Explanation. In *NAACL-HLT*, 2019.

[Kingma and Ba, 2015] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[Kokhlikyan *et al.*, 2020] Narine Kokhlikyan, Vivek Miglani, Miguel Martin, Edward Wang, Bilal Alsallakh, Jonathan Reynolds, Alexander Melnikov, Natalia Kliushkina, Carlos Araya, Siqi Yan, and Orion Reblitz-Richardson. Captum: A unified and generic model interpretability library for pytorch. *arXiv preprint arXiv:2009.07896*, 2020.

[Kovaleva *et al.*, 2019] Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. Revealing the dark secrets of BERT. In *EMNLP*, 2019.

[Li *et al.*, 2016a] Jiwei Li, Will Monroe, and Dan Jurafsky. Understanding neural networks through representation erasure. *arXiv preprint arXiv:1612.08220*, 2016.

[Li *et al.*, 2016b] Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. Deep reinforcement learning for dialogue generation. In *EMNLP*, 2016.

[Li *et al.*, 2019] Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. TextBugger: Generating adversarial text against real-world applications. In *Proceedings 2019 Network and Distributed System Security Symposium*. Internet Society, 2019.

[Li *et al.*, 2020] Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. BERT-ATTACK: Adversarial attack against BERT using BERT. In *EMNLP*, 2020.

[Lundberg and Lee, 2017] Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *NIPS*, 2017.

[Papernot *et al.*, 2016] Nicolas Papernot, Patrick Mcdaniel, Ananthram Swami, and Richard E. Harang. Crafting adversarial input sequences for recurrent neural networks. *MILCOM*, 2016.

[Ren *et al.*, 2019] Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. Generating natural language adversarial examples through probability weighted word saliency. In *ACL*, 2019.

[Ribeiro *et al.*, 2016] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should I trust you?": Explaining the predictions of any classifier. In *SIGKDD*, 2016.

[Saravia *et al.*, 2018] Elvis Saravia, Hsien-Chi Toby Liu, Yen-Hao Huang, Junlin Wu, and Yi-Shin Chen. CARER: Contextualized affect representations for emotion recognition. In *EMNLP*, 2018.

[Schaul *et al.*, 2016] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In *ICLR*, 2016.

[Serrano and Smith, 2019] Sofia Serrano and Noah A. Smith. Is attention interpretable? In *ACL*, 2019.

[Vaswani and others, 2017] Ashish Vaswani et al. Attention is all you need. In *NIPS*, 2017.

[Wallace and others, 2019] Eric Wallace et al. AllenNLP interpret: A framework for explaining predictions of NLP models. In *EMNLP*, 2019.

[Wang and others, 2019] Alex Wang et al. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *ICLR*, 2019.

[Wiegreffe and Pinter, 2019] Sarah Wiegreffe and Yuval Pinter. Attention is not not explanation. In *EMNLP*, 2019.

[Wolf *et al.*, 2019] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

[Wu *et al.*, 2020] Zhiyong Wu, Yun Chen, Ben Kao, and Qun Liu. Perturbed masking: Parameter-free probing for analyzing and interpreting BERT. In *ACL*, 2020.

[Xiong *et al.*, 2018] Caiming Xiong, Victor Zhong, and Richard Socher. DCN+: mixed objective and deep residual coattention for question answering. In *ICLR*, 2018.

[Xu *et al.*, 2019] Jingjing Xu, Liang Zhao, Hanqi Yan, Qi Zeng, Yun Liang, and Xu Sun. LexicalAT: Lexical-based adversarial reinforcement training for robust sentiment classification. In *EMNLP*, 2019.

[Zang *et al.*, 2020] Yuan Zang, Fanchao Qi, Chenghao Yang, Zhiyuan Liu, Meng Zhang, Qun Liu, and Maosong Sun. Word-level textual adversarial attacking as combinatorial optimization. In *ACL*, 2020.

[Zeiler and Fergus, 2014] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014.

[Zeng and others, 2021] Guoyang Zeng et al. OpenAttack: An open-source textual adversarial attack toolkit. In *ACL*, 2021.

[Zhao *et al.*, 2018] Zhengli Zhao, Dheeru Dua, and Sameer Singh. Generating natural adversarial examples. In *ICLR*, 2018.

[Zhong *et al.*, 2017] Victor Zhong, Caiming Xiong, and Richard Socher. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*, 2017.

[Zou *et al.*, 2020] Wei Zou, Shujian Huang, Jun Xie, Xinyu Dai, and Jiajun Chen. A reinforced generation of adversarial examples for neural machine translation. In *ACL*, 2020.