EXPLORING AND IMPROVING ROBUSTNESS OF MULTI TASK DEEP NEURAL NETWORKS VIA DOMAIN AGNOSTIC DEFENSES

Kashyap Coimbatore Murali
High School Student
West Windsor-Plainsboro High School South
West Windsor, NJ 08550
katchu11@gmail.com

ABSTRACT

In this paper, we explore the robustness of the Multi-Task Deep Neural Networks (MT-DNN) against non-targeted adversarial attacks across Natural Language Understanding (NLU) tasks as well as some possible ways to defend against them. Liu et al., have shown that the Multi-Task Deep Neural Network [5], due to the regularization effect produced when training as a result of it's cross task data, is more robust than a vanilla BERT model trained only on one task (1.1%-1.5% absolute difference). We further show that although the MT-DNN has generalized better, making it easily transferable across domains and tasks, it can still be compromised as after only 2 attacks (1-character and 2-character) the accuracy drops by 42.05% and 32.24% for the SNLI and SciTail tasks. Finally, we propose a domain agnostic defense which restores the model's accuracy (36.75% and 25.94% respectively) as opposed to a general-purpose defense or an off-the-shelf spell checker.

Keywords Model Robustness · Domain Adaptation · Adversarial Defense

1 Introduction

The pace of deep learning research in NLU tasks is extremely fast, with new papers being published consistently pushing the benchmark for various tasks in the field. One such example is the Multi-Task Deep Neural Network (MT-DNN) [5], a hybrid model which utilizes a combination of language model pretraining along with multi task learning in order to improve the learning of semantic representations to augment the performance across various NLU tasks.

Even with the rapid pace by which benchmarks are pushed across supervised learning tasks using deep learning, many of these models are susceptible to attacks which are almost imperceptible to humans. After it was proven that image classifying neural networks can be fooled to predict incorrect classes by adding some noise that was unidentifiable by humans [19], the focus began to shift on how NLU models can be attacked to make false predictions on sentences which are obviously different classes when observed by humans.

There are four fundamental ways of creating adversarial text examples: adding, deleting, swapping adjacent characters, or replacing characters with those that are closer on the keyboard [4]. We explore various combinations of these character level *swap* attacks on the perturbed sentences so that the text is falsely classified when fed through the MT-DNN, yet easily comprehensible by humans. These alterations are founded through psycho-linguistic studies, which show that the semantic meaning of the words can be easily comprehended even if the characters are jumbled as long as the first and last characters remain the same (Ex: *Cmabridge Uinversity*).

We proceed to explore various methods by which defenses can be introduced such that even if perturbations exist within multiple characters across a sentence, the accuracy can be restored. We utilize a robust word recognition model inspired from Semicharacter RNNs [20] which can correct a word by generating encodings for the first character, then treating the middle characters as a bag of characters where frequency is the feature as opposed to position, and finally generating a final encoding for the last character. We further propose a *multi-task defense* where the word recognition model is pretrained off a combination of multiple tasks (similar to how the MT-DNN is trained), which we can then fine-tune to the task at hand, thus ensuring a modular defense which can learn the intricacies of the domain at which it's operating in.

2 Related Works

2.1 Pre-trained Language Models

There are various pre-trained language models which employ different strategies to apply them to downstream tasks such as feature-based method (ELMo [8]) or a fine-tuning method (XLNet [15], BERT [1], Open AI GPT [13] in order to achieve state-of-the-art accuracy in different NLI tasks. These models employ the large swaths of unlabelled textual data in order to train contextualized word representations, thus eliminating the need to create bulky task specific architectures.

2.2 Application of Pretrained Models on Downstream Tasks

Due to the extent of the pretraining and the complexity within these models, in order to make a prediction we simply need to add a classification layer at the end to make a prediction on different tasks (as mentioned later). However in order to achieve higher accuracies need to incorporate the data of a given domain. There are three main ways to accomplish this: finetuning the entire mode along with the classification layer, fine tuning the BERT [1] model using Masked Language Modeling, and Next Sentence Prediction (essentially not using the classification layer), and Multi Task Finetuning. In the original approach we simply add a classification layer based on the task and fine tune *all* the model's parameters using the in domain data for that use case in the second approach we don't add the classification layer but simply retrain the BERT [1] model using it's original loss functions for the Masked Language Modeling, and next Sentence Prediction Tasks using the in domain data, thus allowing it to create domain-specific contextual embeddings prior o the application of the classification layer. The final approach ascertains the combination of multiple tasks by conjoining the entire dataset across tasks to create a generalized model, while using task specific loss functions when applying it to a specific use-case.

2.3 Adversarial Attacks

Adversarial examples can be generated using black-box or white-box techniques based off of the knowledge of the neural network. Black-box attacks are used when the general architectures, parameters, and hyper-parameters of the model aren't accessible (such as the attack on some foreign API service). White box attacks require complete knowledge of the parameters and architecture of the model to be generated to be generated.

These type of adversarial attacks with respect to Natural Language, character or word-level, have been introduced in a variety of manners which primarily either focus on the existence of predominant weaknesses [21], through white-box defensive strategies [23], through black-box strategies [22], or a combination of both [4] [24]

There also exist different types of perturbations that can be performed on the text including: adding, swapping, deleting, or inserting characters which are positionally closer on the keyboard. These attacks are shown in [17] they have the ability to significantly decrease words per minute (wpm) by up to 36% based on the location of the attack and that although 50% of survey respondents didn't understand a few words, they were still able to answer the perturbed questions with a high accuracy.

2.4 Defenses

In order to defend against gradient based attacks (white-box) or non-gradient based attacks (black-box) alike, one potential method is through the use of spelling correction. Spelling correction [3] is often viewed as a sub-task of grammatical error correction [7] [9]. Classical methods utilize a body of text which includes the number of times the word is used, then analyze different errors and finally compute the Levenshtein distance between the chosen word and the target word. Another approach was through the use of a pretrained language embedding trained to identify n-grams in order to make a prediction based off of the previous word. Lately, deep learning inspired approaches have been introduced, which analyze both the context and orthography of the input. There are also off-the-shelf spell checkers such as AHD, or Hunspell check which are trained off of a general purpose dataset and allow for ease of use due to their packaging as Python libraries. This work utilizes the SemiCharacter RNNs [20]
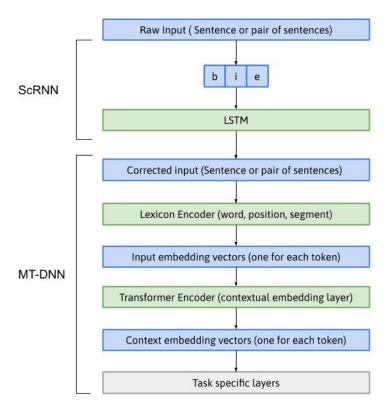
Figure 1: This is the architecture of the model including the ScRNN and the MT-DNN. This shows how an input flows through the ScRNN, gets corrected with the LSTM, and then flows through the model from the lexicon encoder to the transformer encoder to generate the contextual word embedding after which a task specific output layer exists to make a prediction. Image combined elements from [1] and [20]

3 Approach

3.1 MT-DNN Model Architecture

The model architecture contains a chain of shared text encoding layers which has of a lexicon encoder and a contextual transformer encoder, which are trained and shared across all tasks. The shared lexicon and contextual transformer encoder is initialized using the BERT [1] model as released by huggingface's pytorch library [11]. These parameters are then update during the Multi Task Learning (MTL) phase of creating a pretrained MT-DNN model [5]. The output layers are then appended to the bottom as shown in (Figure 1), which are chosen based off the task. The individual components are further elaborated below.

3.1.1 Lexicon Encoder

The lexicon encoder is the pre-processing step in order to generate the contextual embedding from the Transformer Encoder [16]. Given an input X, it is first tokenized using WordPiece tokenization [12] which splits a token into pieces that model can understand (Ex: "I am playing" -> ["i", "am", "play","##ing"]), with a word that it may not know at all being split all the way down to the character level, hence ensuring that there would never be an OOV error for words created from characters of the alphabet of the language for which the model is trained. In exceptional cases where the character isn't recognized, the token is replaced with a [UNK] token, thus ensuring that no error is produced. The first token of the sequence is always the classification token ([CLS]) with the final hidden state of this token containing the representation of the entire sequence and is used for sentence level classification tasks. In situations where there are a
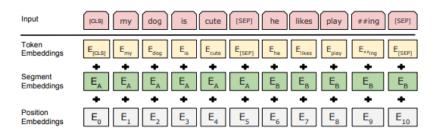
Figure 2: This shows how the lexicon encoder creates an input representation by summing the token, segment, and position embeddings from a given input. From [1]

pair of sentences (2 sentence classification tasks), the sentences are separated with the [SEP] token which denotes what segment it belongs to. This would result in a list of tokens like this:

One sentence classification:

["[CLS]","i","am","play","##ing"]

Two sentence classification task:

["[CLS]","i","am",play,"##ing","[SEP]","i","am","talk","##ing","[SEP]"]

This is then mapped to a sequence of input embedding vectors for each token which is created by summing the corresponding word, segment, and positional embeddings (Figure 2).

3.1.2 Transformer Encoder

The multi-layer bidirectional transformer-encoder [10] layer then maps the input representation vectors to a sequence of contextual embedding vectors. This becomes the representation which is used across all tasks. The transformer layer utilizes a self-attention mechanism which directly models contextual relationships across all tokens in a sequence. A key difference in the MT-DNN model as opposed to the vanilla BERT [1] model is the utilization of Multi Task Objectives (Figure 3) as well as pre-training.

3.1.3 Single-Sentence Classification Task

Given the contextual embedding [CLS] token(the aggregate semantic representation of the entire sentence), it can be used to make a prediction regarding the sentence as a whole. The output probability is based on a probability distribution of size n_{labels}. The probability is predicted using a softmax function over the linear output layer.

$$P(c|x) = \text{softmax}(W_{task} * x) \tag{1}$$

The equation represents the probability of each class given a contextual embedding input x, where W_task represents the learned weight matrix for a given task.

The loss function for this task is binary cross entropy loss as the objective.

$$-\sum_c \mathbb{1}(X,c)\log(P_r(c|X)) \tag{2}$$

where $\mathbb{1}(X,c)$ is either 0 or 1 (binary classification) if the predicted class label c is the correct classification for X(single sentence), penalized by a factor of P_r (softmax equation)

Input: "The movie was great and had great actors and actresses" Output: Positive Review

3.1.4 Pairwise Text Classification

In the pairwise text classification problem, the objective is to output a label given an input of 2 separate sentences. These sentences are packed together into one input sequence which includes a premise and the hypothesis(2 separate components). The original MT-DNN [5] utilizes a Stochastic Answer Network [18] as opposed to just predict a

label which allows it to maintain a state and iteratively refines its predictions for k-number of steps (where k is a hyperparameter) and averages the prediction at each step k to create a final prediction which improves the robustness of the model.

$$P_r^k = \text{softmax}(W_{task} * s^k * x^k) \tag{3}$$

The equation above is very similar to the single sentence text prediction however it maintains a state s throughout each step k after which the probability distribution P_r is averaged to produce the final output.

The loss function for this task is binary cross entropy loss.

$$-\sum_c \mathbb{1}(X, c)\log(P_r(c|X)) \tag{4}$$

where $\mathbb{1}(X, c)$ is either 0 or 1 (binary classification) if the predicted class label c is the correct classification for X(pair of sentences), penalized by a factor of P_r (softmax equation)

Input: "A black race car starts up in front of a crowd of people." (Sentence A), "A man is driving down a lonely road." (Sentence B)
Output: Contradiction

3.1.5 Text Similarity Task

Given the contextual embedding [CLS] token (the aggregate semantic representation of the entire sequence), it can be used to make a prediction regarding the sequence as a whole. A similarity score can be calculated using:

$$\text{Sim}(X_1, X_2) = W_{task} * x \tag{5}$$

where (X_1, X_2) represents the 2 different input sequences, W_{task} represents the learned weight matrix for this task, and where x represents the contextual embedding created by the transformer encoder after being fed the input from the lexicon encoder. The function $\text{Sim}(X_1, X_2)$ can be used to represent the similarity as a real value in the range of $(-\infty, \infty)$

The loss function for this task is mean squared error.

$$(y - \text{Sim}(X_1, X_2))^2 \tag{6}$$

where y represents the ground truth similarity of the text from a range of $(-\infty, \infty)$.

Input: "Thank you very much, Commissioner."(Sentence A) "Thank you very much, Mr Commissioner." (Sentence B)
Output: 4.500

3.1.6 Relevance Ranking

Given the contextual embedding [CLS] token (the aggregate semantic representation of the entire sequence of a pair of question and its candidate answer (Q, A). A relevance score can be computed as:

$$\text{Rel}(Q, A) = g(W_{task} * x) \tag{7}$$

Where a given Q and input representation x, we rank all of its candidate answers using the relevancy score.

The loss is similar to the 2 sentence classification task, but instead however reduces the negative log likelihood of the positive answer given questions from the training data.

$$-\sum_{(Q,A^+)} P_r(A^+|Q) \tag{8}$$

where P_r is:

$$P_r(A^+|Q) = \frac{\exp(Rel(Q, A^+))}{\sum_{A' \in A} \exp(Rel(Q, A'))} \tag{9}$$

Input:
Question: Who edited Electrical World magazine?
Answers:

- *In 1888, the editor of Electrical World magazine was Thomas Commerford Martin*
- *Not all cells in a multicellular plant contain chloroplasts.*

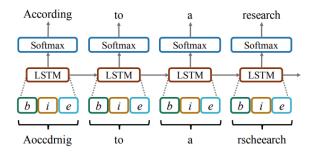*Output: The first answer is the correct answer for the given query*

Figure 3: Representation of how the ScRNN takes a misspelled word and uses an LSTM to make a prediction on the correct word. From: [20]

3.2 Semicharacter RNN (ScRNN) architecture

The ScRNN [20] is inspired from findings of the Cmabrigde Uinervtisy effect (also known as typoglycemia), which states that as long as the first and last characters of a word are the same the order of the middle characters doesn't matter in order to know the meaning of the word. This is demonstrated in the paragraph below:

Aoccdrnig to a rscheearch at Cmabrigde Uinervtisy, it deosn't mttaer in waht oredr the ltteers in a wrod are, the olny iprmoetnt tihng is taht the frist and lsat ltteer be at the rghit pclae. The rset can be a toatl mses and you can sitll raed it wouthit porbelm. Tihs is bcuseae the huamn mnid deos not raed ervey lteter by istlef, but the wrod as a wlohe.

The ScRNN is shown to significantly increase performance in word spelling correction compared to existing spelling checkers and charCNNs [19]. For regular RNNs, the input vector is the word or character representations, but for input vector for ScRNNs consist of 3 subvectors

$$x_n = \begin{bmatrix} b_n \\ i_n \\ e_n \end{bmatrix} \tag{10}$$

The first and last subvector represents the one hot vector of the first and last characters respectively, and the middle subvector contains the number of occurrences each character has regardless of their order. All three subvectors would be of the size of the alphabet. This concatenation of the subvectors forms x_n which is used as the input to the model which would output the most probably word. The ScRNN performs better as compared to charCNNs from (Kim et al.) [20] and commercial spell checkers for all the noises that it was tested on (adding - 3.52% absolute improvement, deleting - 13.89% absolute improvement, and jumbling - 41.85% absolute improvement). The difference in performance is especially visualized in jumbling, as oftentimes the other models aren't designed for severely jumbled input. The ScRNN is trained using cross entropy loss as shown below:

$$y_n = \frac{\exp(W_h * h_n)}{\sum_v \exp(W_h * h_n)} \tag{11}$$

The model learns the weight matrices W in order to predict the word based off the hidden state h, with a fixed layer vocabulary of v.

3.3 Training Technique

The training of the MT-DNN [5] is split into two aspects: pretraining off of the GLUE dataset, and finetuning to the task at hand.

3.3.1 Pretraining

MT-DNN:
As opposed to training off one dataset at a time, all the GLUE datasets are merged and shuffled, after which a mini-batch is constructed where the model trains off one example at a time and then uses different loss functions based on the task for that particular sample to compute the gradient and update the model accordingly.

ScRNN:
Similar to the MT-DNN [5] , the ScRNN [20] in first trained off the Penn Treebank dataset with synthetic noise added.

3.3.2 Finetuning

MT-DNN:
The MT-DNN model [5] takes data which is pertinent to the use case for which it is being implemented in, and then finetunes the pretrained model to that specific scenario. These opportunities for finetuning as opposed to simply using the pretrained model allow for a greater flexibility and accuracy for the specific use case.

4 Experiment Details

4.1 Adversarial Attacks

After randomly assorting the sentence into a list of characters, a position is randomly chosen in order to determine where to perform the attack. With regards to a two character attack, the same procedure is repeated two get the adversarial example. In this experiment we only utilize the swap attack (where two adjacent characters' positions are switched), but the experiment also provides a framework to choose what attack to perform (delete, insert, shuffle, or trying all of them and finding which one compromises the accuracy the most)[1]. As the attack position is chosen at random there is a possibility for the 2-char attack to be on the same word. However for sufficiently long sentences, this scenario is not frequently observed.

Samples of these attacks can be observed in Table 2, where the adversarially modified samples are shown in **bold**. This table also shows how 1-char and 2-char attacks are explored within the SNLI and SciTail datasets.

4.2 Architecture

4.2.1 MT-DNN

We followed Liu et. al's implementation of the MT-DNN [5] . It used huggingface's PyTorch implementation of BERT for the parameter initialization [11]. Then it used Adamax [14] as the optimizer with a learning rate of 5e-5 and a batch size of 32 by following Devlin et al. [1]. The maximum number of epochs was set to 5. A linear learning rate decay schedule with warm-up over 0.1 was used. We also set the dropout rate of all the task specific layers as 0.1, except 0.3 for MNLI and 0.05 for CoLA. To avoid the exploding gradient problem, we clipped the gradient norm within 1. All the texts were tokenized using wordpieces, and were chopped to spans no longer than 512 tokens.

4.2.2 ScRNN

The input layer of ScRNN [20] consists of a vector with length of 76 (A-Z, a-z and 24 symbol characters). The hidden layer units had size 650, and total vocabulary size was set to 10k. We applied one type of noise to every word, but words with numbers (e.g. 1980s) and short words (lengths less than 4) were not subjected to jumbling, and therefore these words were excluded in evaluation. We trained the model by running 5 epochs with (mini) batch size 20. We set the backpropagation through time (BPTT) parameter to 3 which means that the ScRNN updates weights for previous two words (x_{n-2}, x_{n-1}) and the current word (x_n).

---

[1]`https://github.com/katchu11/Robustness-of-MT-DNNs/blob/master/attacks.py`

Table 1: Data from 1-character and 2-character attacks across various architectures

Format: 0 attacks, 1-char attack, 2-char attack						
Architecture	SNLI			SciTail		
BERT (Vanilla)	90.50%	59.20%	39.90%	93.50%	63.50%	43.60%
MT-DNN	91.60%	54.96%	49.55%	95.00%	78.83%	62.76%
MT-DNN + ScRNN	91.50%	87.40%	86.30%	94.70%	89.50%	88.70%

Table 2: Sample data showing corrupted samples

Tasks	# of attacks	Samples (Sentence 1 ⫴ Sentence 2)
SNLI	1-char	A soccer **gmae** with multiple males playing ⫴ A soccer game with multiple males playing.
	2-char	A soccer **gmae** with multiple males playing ⫴ Some men **rae** playing a sport.
SciTail	1-char	The liver is **dviided** into the right lobe and left lobes ⫴ The gallbladder is near the right lobe of the liver.
	2-char	The liver is **dviided** into the right lobe and left lobes ⫴ The **gallbaldder** is near the right lobe of the liver.

Table 3: Examples that passed the MT-DNN + ScRNN combo still failed at

Datasets	Faultily identified sentences (Sentence 1 ⫴ Sentence 2)
SNLI	Two women are embracing while holding to go **pcakages**. ⫴ The sisters are hugging goodbye while holding to go packages after just eating lunch.
	Man in a black suit, **wihte shitr** and black bowtie playing an instrument with the rest of his symphony surrounding him.⫴ A person in a suit
	A woman stands at a podium with a slide show behind her.⫴ A woman is standing at a **pdoium**.
SciTail	**Gsaes** have neither definite volume nor shape ⫴ Gas has no definite volume and no definite shape
	A lunar **elcipse** occurs when the Moon passes into the Earth's shadow ⫴ When earth's shadow falls on the moon, the shadow causes a **lnuar** eclipse.
	During **pohtosynthesis**, sunlight changes water and carbon dioxide into glucose and oxygen. ⫴ Photosynthesis is described by this statement: carbon dioxide and water are turned into sugar and oxygen.

4.3 Experimental Results

As shown in Table 1, we can see that the as we perform character level attacks on the defenseless architectures (BERT, and the MT-DNN), the accuracy significantly reduces with a drop off of 31.3% and 30.0% across datasets after one attack for BERT, and a drop off of 36.64% and 16.67% across datasets for the MT-DNN. When using a two character attack, the accuracy drops off by 50.6% and 49.9% across datasets for the BERT model, and by 42.05% and 32.24% across datasets for the MT-DNN model. This data shows the already pre-existing resilience to attacks from the MT-DNN model due to it's cross training, as both drop offs in accuracy from the MT-DNN are still lower than drop offs in accuracy from BERT.

However when prefixing a Semicharacter RNN as a pre-processing step for the data, we observe a significant "restoration" in accuracy. When utilizing an ScRNN along with an MT-DNN, the accuracy is restored by 32.44% and 10.67% across datasets for 1 character attacks, and by 36.75% and by 25.94% for 2 character attacks across datasets.

4.3.1 Error Analysis

When observing corrupted samples (shown in Table 2 and Table 3) however that still passed through the MT-DNN + ScRNN combination, it can be seen that they are mainly samples with *domain-specific* words which throw off the prediction. Most of the errors occurred when domain-specific words such as "Gases" or "Photosynthesis" were misspelled (incorrect samples shown in **bold** for both tables). This shows that the models performance could potentially be increased through the use of a domain-specific training step for the ScRNN. This could be included a priori to deployment by either training the ScRNN in parallel to the MT-DNN, or separately (before or after).

These results demonstrate that the MT-DNN is inherently more resilient to attacks than BERT due to it's multi-task learning, but that it's accuracy can also be restored significantly when utilizing a domain adaptable defense.

5   Conclusion

Through our training we effectively show that the MT-DNN is a very versatile architecture for NLU tasks. Although it can be severely compromised to basic adversarial attacks, through the use of a domain adaptable defense, it's accuracy can be almost completely restored in order to ensure optimal performance. There is however room for improvement to perhaps integrate more than one system of adversarial resistance. This model can also be improved by adding a domain-specific fine-tuning step to the ScRNN to ensure that words that aren't especially common for a certain domain are still noted.

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[2] Kun Jing and Jungang Xu. A survey on neural network language models, 2019.

[3] Karen Kukich. Techniques for automatically correcting words in text. *ACM Comput. Surv.*, 24:377–439, 1992.

[4] Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. Textbugger: Generating adversarial text against real-world applications. *Proceedings 2019 Network and Distributed System Security Symposium*, 2019.

[5] Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. Multi-task deep neural networks for natural language understanding, 2019.

[6] Xiaodong Liu, Yelong Shen, Kevin Duh, and Jianfeng Gao. Stochastic answer networks for machine reading comprehension, 2017.

[7] Hwee Tou Ng, Siew Mei Wu, Yuanbin Wu, Christian Hadiwinoto, and Joel Tetreault. The CoNLL-2013 shared task on grammatical error correction. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–12, Sofia, Bulgaria, August 2013. Association for Computational Linguistics.

[8] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proc. of NAACL*, 2018.

[9] Allen Schmaltz, Yoon Kim, Alexander M. Rush, and Stuart M. Shieber. Adapting sequence models for sentence correction. *CoRR*, abs/1707.09067, 2017.

[10] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[11] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. Huggingface's transformers: State-of-the-art natural language processing, 2019.

[12] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation, 2016.

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

[14] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization, 2014.

[15] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. Xlnet: Generalized autoregressive pretraining for language understanding, 2019.

[16] Marcus, Mitchell, et al. Treebank-3 LDC99T42. Web Download. Philadelphia: Linguistic Data Consortium, 1999.

[17] Rayner, K. et al. . Raeding wrods with jumbled lettres: There is a cost. Psychological Science, 17, 192-193.

[18] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification, 2015.

[19] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks, 2013.

[20] Keisuke Sakaguchi, Kevin Duh, Matt Post, and Benjamin Van Durme. Robsut wrod reocginiton via semi-character recurrent neural network, 2016.

[21] Yonatan Belinkov and Yonatan Bisk. Synthetic and natural noise both break neural machine translation. *CoRR*, abs/1711.02173, 2017.

[22] Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. Black-box generation of adversarial text sequences to evade deep learning classifiers. *CoRR*, abs/1801.04354, 2018.

[23] Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. HotFlip: White-box adversarial examples for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 31–36, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[24] Wei Emma Zhang, Quan Z. Sheng, and Ahoud Abdulrahmn F. Alhazmi. Generating textual adversarial examples for deep learning models: A survey. *CoRR*, abs/1901.06796, 2019.