GOVERNMENT OF THE RUSSIAN FEDERATION
NATIONAL RESEARCH UNIVERSITY
HIGHER SCHOOL OF ECONOMICS

Faculty of Computer Science

Educational program «Data Science»

Adversarial Search on Albert model.

Student: Emanuel Lopez Crespo

Supervisor: Attila Kertesz-Farkas, Ph.D.

Moscow, 2020

ABSTRACT

This report explores the concept of adversarial attacks on the large scale linguistic neural network Albert with respect with NLP tasks such as Question Answering. The use of adversarial attacks test security, robustness, in machine learning models by causing a model's mistake. The adversarial method used in this paper called addany which adds a group of words to the end of a context to trick the model when it is queried for a particular question.

TABLE OF CONTENTS

Introduction

The field of data science has advanced greatly in the last decade and has brought many innovations in the field of deep learning. With the sudden rush of their commercial implementations, many have come to question the integrity of these machine learning models. With its ever-growing potential due to improvements in computational power and access to data, people have questioned their complexity and safety. Now more than ever with self-driving cars, networks, facial recognition software, etc, the safety and security of these applications in Artificial Intelligence (AI) has scared many groups in the general population. By using adversarial examples on neural networks, we can gain insight into these black-box models and improve them. In the Natural Language Processing (NLP) domain, these attacks can help us to understand more of the underlying semantics of complicated linguistic models.

Adversarial attacks are inputs to models designed to cause these models to make mistakes. (11) These attacks can find problematic decision boundaries (4) to increase the robustness of models, which can increase safety and performance. The accountability from the general public for AI today is high, in recent news, Google had to apologize after its image-recognition photo app labeled African Americans as “gorillas” (5). By applying adversarial attacks, we plan to help the community continue to improve their models. In previous research Visual Question Answering models have been attacked with Carlini methods and FGSM on VQA datasets (8). Another type of adversarial attack is the application of GAN’s which have very useful applications (4) such as for object and image recognition tasks.

Here we try to focus on a specific method that has had been highly noticed in recent research. These techniques called addsent and addany adversarial techniques have been performed on SQuAD dataset for BiDAF models (9)(6). The applications of adversarial attacks in the field of NLP are still quite novel and can potentially give more insight into the learning patterns of these linguistic models. We are interested in gaining more insight to these NLP models because they have been criticized for their incapacity to distinguish correct answers from common words similar to the correct answer. (6) Previous research presents the possibility of models instead of learning the true semantics of the language may be learning heuristics that could affect certain NLP tasks due to poor learning strategies (14). In this report, we will apply an addany method (6) to test the Albert model for Question Answering and its integrity.

Methods

Dataset

During the training and the attack, we used the SQuAD dataset (12) which has reading comprehension questions about articles found in Wikipedia. It contains about 107,000 questions.

Model

In the field of NLP one of the most notable model architectures that use the transformers attention model is known as BERT (short for Bidirectional Encoder Representations from Transformers). In this paper the model applied was AlBERT a variant which differs from the BERT architecture by parameter sharing, sentence order prediction and embedding parameterization (13). They both use a bidirectional training approach and have been trained on the SQuAD dataset. Albert was chosen due to its reproducibility, its abundance of previous

research, its manageable accessibility and its parameter reduction techniques. This model has scored an average F1 score of 92.215 and EM score of 89.731 on the SQuAD dataset. Our trained model achieved an F1 score of 89.200 and a best exact score of 81.788. Many more versions of AlBERT have been augmented and tweaked to yield higher scores.

packages

transformers

The main library used was the hugging face/transformers library previously known as the pytorch transformers. This library provides general purpose models which are used alongside the tensorflow and pytorch libraries. This library allows us to train models and use trained models to reduce the amount of coding and time required to work with these models. This library provided the script for training on the squad dataset. Their script offers fine tuning of the Bert Question Answering model in this case Albert base v2 to be applied to the SQuAD dataset.

pytorch

This is one of the most famous python scientific computing libraries which aims to replace the numpy library with the use of GPU's and use it for deep learning applications. Most of the transformers package is integrated with pytorch.

addany

The attack was performed using an addany technique (4). Its goal is to create a list *d* which when appended to the end of context, tricks the model into giving the wrong answer. The list *d*, initially contains a fixed number of randomly initialized words from the brown corpus

subset of common English words. The "common English words" are defined to be the 1000 most frequent words found in brown corpus in the nltk package of python. The brown corpus is a million-word electronic corpus of English from Brown University (1).

The list '*d'* is then iteratively improved by searching for better adversarial words. The search can be summarized as this.

Each iteration starts with selecting an index '*i*' to be modified in '*d*' and sampling a list of candidate words *W* for that index. This candidate list *W* is a union of 10 words sampled from "Common English words" and the words inside the question statement. We then replace *i*'th index word in '*d*' with each word W one-by-one and repeatedly query the model for the F1 score. If all candidate words perform poorer than the original words, we keep the original word, otherwise d_i is replaced by a word in *W* that achieves a minimum score.

This process is repeated for all indices of '*d*' in a random order. Something to note is that *d* is appended to the end of the context regardless of grammaticality and sense. The goal is to find the combination of words that when added to the end of the context minimizes the F1 score of the model's prediction on a question. Once the iterations on all indices of '*d*' are completed, it marks the end of one cycle which we call an epoch. This epoch is run *k* times which in this case is 3 times, where it searches for better and better words for each index of '*d'*. This marks the end of one mega-epoch. If after the first mega-epoch, we do not obtain a list '*d'* which deceives the model, it is possible that we may be stuck at a local minimum and in order to find the absolute minima, we run another cycle in parallel with new random words from the brown corpus. We refer to this in the code as a megaepoch.

Pseudocode

d – 10 words taken from the package and the question

W – list of words also from the question and the brown corpus

k = epochs

- For epoch in megaepoch:
    - initialize new *d*=10 amount of words from a list of common English words
    - for *k*=3 epochs of local search:
        - For index *i* (in random order) in list d:
            - Initialize a list *W* filled with 20 random words from brown corpus plus the words in question statement
            - iterate over all words in *W* (w_1 to w_d)
            - Place w_i at d_i to create a candidate sentence and evaluate the model
            - If it performs better with the candidate sentence, keep the word w_i at d_i else retain the original d_i
    - if no sentence that achieves a minimum score generate a separate list of words '*d*' and run another mega-epoch on this list in parallel

Training

Training was performed on one GPU Nvidia GeForce RTX 1080 and took 100 hours. It was run on the Squad 1.1 data and tested on only questions that are answerable because the Squad 2.0 dataset has unanswerable questions. The evaluation consisted of 10866 examples and a batch size of 8 was used during evaluation. The evaluation took close to 3 hours using the Nvidia RTX 1080 GPU.

Results

Squad dataset attack

On a sample of 20 question from the SQuAD dataset the F1 score was 0.951 on the training dataset and 0.917 on the testing dataset. Only 20 were chosen due to the slow algorithm time. The attack was took on this subset of questions 1 hour to evaluate. Here are the F1 scores during each epoch/megaepoch on the SQuAD dataset.

Megaepoch	Epoch	Average F1Score
0	0	0.759
0	1	0.233
0	2	0.150
1	0	0.146
1	1	0.145
1	2	0.144
2	0	0.000

Here is an example of the adversarial search:

context :

“”The history of New York begins around 10,000 B.C. when the first people arrived. By 1100 A.D. two main cultures had become dominant as the Iroquoian and Algonquian developed. European discovery of New York was led by the Italian Giovanni da Verrazzano in 1524 followed by the first land claim in 1609 by the Dutch. As part of New Netherland, the colony was important in the fur trade and eventually became an agricultural resource thanks to the patroon system. In 1626 the Dutch bought the island of Manhattan from American Indians.[1] In 1664, England renamed the colony New York, after the Duke of York (later James II & VII.) New York City gained prominence in the 18th century as a major trading port in the Thirteen Colonies. New York played a pivotal role during the American Revolution and subsequent war. The Stamp Act Congress in 1765 brought together representatives from across the Thirteen Colonies to form a unified response to British policies. The Sons of Liberty were active in New York City to challenge British authority. After a major loss at the Battle of Long Island, the Continental Army suffered a series of additional defeats that forced a retreat from the New York City area, leaving the strategic port and harbor to the British army and navy as their North American base of operations for the rest of the war. The Battle of Saratoga was the turning point

of the war in favor of the Americans, convincing France to formally ally with them. New York's constitution was adopted in 1777, and strongly influenced the United States Constitution. New York City was the national capital at various times between 1785 and 1790, where the Bill of Rights was drafted. Albany became the permanent state capital in 1797. In 1787, New York became the eleventh state to ratify the United States Constitution.”””

questions with ground truth answers:

Question	Answer before attack	Answer after attack
Who was responsible for the founding of New York?	giovanni da verrazzano	'foreign founding national nation
When did New York help push forward the Consitution?	1777	1787
Who was the city named after?	duke of york	'felt run thousand'
Which tribes were situated in New York early on?	Iroquoian and Algonquian	tribes giving situated whole love picture'

A second example

context:

Twenty-five years after the Great War, Vault 76 is opened up and its residents given the task of repopulating the Wasteland. Shortly after they emerge from the Vault, the player character is contacted by the Vault Overseer. She reveals that Vault 76 was given a secret mandate to secure an arsenal of nuclear weapons deployed throughout Appalachia in three still-functioning nuclear missile silos: Site Alpha, Site Bravo, and Site Charlie. The player character is directed to contact the Responders, a faction of emergency services personnel who tried to aid the residents of Appalachia during the war; however, they discover that the Responders evacuated after they came under attack from the Scorched. Further investigation reveals that the Scorched are ghouls infected by the breath of a Scorchbeast, a creature that was awoken after the Great War.

As the Scorched represents a threat to the Wasteland beyond Appalachia, the player character decides to secure a nuclear weapon and use it to destroy a Scorchbeast nest. In order to achieve this, they start searching the bunkers of survivalists calling themselves the Free States who were working on the means to detect the Scorched until the Scorched overran them. The player character is able to build a radar system that detects the viral signature of the Scorched, but its limited range makes it ineffective. The player attempts to boost the signal first by gaining the friendship of Rose, a Miss Nanny robot trying to rebuild the local Raider gangs that lived in Appalachia before being scared off by the Scorched; and then by investigating research done by the local Brotherhood of Steel into the Scorchbeast as they had tried to stop it spreading from its main lair. The player hacks into a government network and draws the attention of the Enclave, the remnants of the United States government. The Enclave are led by MODUS, a centralized artificial intelligence system located in a secure nuclear fallout bunker beneath the White spring Resort. MODUS asks the player character to help connect them to a series of isolated computer networks across Appalachia through the local SIGINT base and reestablish contact with the Kovac-Muldoon orbital platform, and in return offers to connect to the radar system to detect Scorched across the region. List W

Question	Answer before attack	Answer after attack
Who is the player character contacted by?	the vault overseer	'pretty science ? method west'
Who attacked the responders?	the scorched	'short rhode reason lead price child
What does the player character build?	a radar system	'george what
Who are the enclave led by?	MODUS	led

Average F1-Score before attack: 0.9

Average F1-Score after attack: 0.0

Conclusion and discussion

Here we present the addany method performed on the SQuAD dataset for a type of BERT model named AlBERT. The original plan was to change the word embeddings but that seemed to prove unlikely due to context, questions and answers not being able to be distinguished after encoding to vectors. Models like this who perform well on big datasets often do poorly on adversarial evaluation. This is also consistent when given adversarial datasets such as HANS (14). These quite famous datasets do not seem to test true reasoning and understanding if not possible heuristics (14). This proposes in the future to move towards harder datasets that avoid these heuristics.

There is also the possibility to train the models with adversarial examples but this proves difficult due to the complexity of the calculations with word vector embeddings. We also present the issue that some of the answers predicted by the model were ungrammatical or syntactically incorrect. Hopefully for the future of creation of these NLP tasked models will be able to create models that actually can learn real intelligent behavior rather than simply patterns (6). This shows that even with some of the adversarial answers we saw the grammaticality isn't something the models learn. With our current standings this proves that the fragility of these models causes

them to not be as reliable as many may think. There could be several improvements done in the future. More models could be tested as well as more Bert models for adversarial searches. More complicated datasets could be created or some datasets could be adjusted to include adversarial examples. The searches of the addany however are quite computationally intensive which leaves the opportunity for optimization of these searching algorithms.

BIBLIOGRAPHY

1) NLTK. (n.d.). Retrieved June 21, 2020, from https://www.nltk.org/book/ch02.html

2) Transformers. (n.d.). Retrieved June 21, 2020, from https://huggingface.co/transformers/

3) Adversarial Example Generation¶. (n.d.). Retrieved June 21, 2020, from https://pytorch.org/tutorials/beginner/fgsm_tutorial.html

4) Yuan, X., He, P., Zhu, Q., & Li, X. (2019). Adversarial Examples: Attacks and Defenses for Deep Learning. *IEEE Transactions on Neural Networks and Learning Systems, 30*, 2805- 2824.

5) Lohr, S. (2018, February 09). Facial Recognition Is Accurate, if You're a White Guy. Retrieved June 21, 2020, from https://www.nytimes.com/2018/02/09/technology/facial-recognition-race-artificial-intelligence.html

6) Jia, R. , & Liang, P. . (2017). Adversarial Examples for Evaluating Reading Comprehension Systems.. abs/1707.07328, 2021-2031.

8) Sharma, V., Kalra, A., Vaibhav, Chaudhary, S., Patel, L., & Morency, L. (2018). Attend and Attack : Attention Guided Adversarial Attacks on Visual Question Answering Models.

9) Akhila, Y., Amita K. (2018). Adversarial SQuAD from https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1184/reports/6908723.pdf

10) Shi, Z., Huang, M., Yao, T., & Xu, J. (2019). Adversarial examples with difficult common words for paraphrase identification. *arXiv preprint arXiv:1909.02560*.

11) Goodfellow, I. (2019, March 07). Attacking Machine Learning with Adversarial Examples. Retrieved June 22, 2020, from https://openai.com/blog/adversarial-example-research/

12) Rajpurkar, P., Zhang, J., Lopyrev, K., & Liang, P. (2016). SQuAD: 100, 000+ Questions for Machine Comprehension of Text. *EMNLP*.

13) Horev, R. (2018, November 17). BERT Explained: State of the art language model for NLP. Retrieved June 23, 2020, from https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270

14) McCoy, R. & Pavlick, Ellie & Linzen, Tal. (2019). Right for the Wrong Reasons: Diagnosing Syntactic Heuristics in Natural Language Inference.

222

python run_squad.py --model_type albert --model_name_or_path albert-base-v2 --do_eval --do_train --do_lower_case --train_file /home/data/SQuAD/train-v1.1.json --predict_file /home/data/SQuAD/dev-v1.1.json
--per_gpu_train_batch_size 4 --learning_rate 3e-5 --num_train_epochs 2.0 --max_seq_length 384 --doc_stride 128 --output_dir E:\Data\newmodel --overwrite_output_dir --save_steps 5000

model config

Model config AlbertConfig {

"architectures": [

"AlbertForQuestionAnswering"

],

"attention_probs_dropout_prob": 0,

"bos_token_id": 2,

"classifier_dropout_prob": 0.1,

"down_scale_factor": 1,

"embedding_size": 128,

"eos_token_id": 3,

"gap_size": 0,

"hidden_act": "gelu_new",

"hidden_dropout_prob": 0,

"hidden_size": 768,

"initializer_range": 0.02,

"inner_group_num": 1,

"intermediate_size": 3072,

"layer_norm_eps": 1e-12,

"max_position_embeddings": 512,

"model_type": "albert",

"net_structure_type": 0,

"num_attention_heads": 12,

"num_hidden_groups": 1,

"num_hidden_layers": 12,

"num_memory_blocks": 0,

"pad_token_id": 0,

"type_vocab_size": 2,

"vocab_size": 30000

}