Assignment 2 Project Report

Team members: Zixi Jiang, Peizhen Li, Xiaoyu Wang, Yuchen Zhang, Xiuwen Zhang, Nat Zheng
GitHub Repo: https://github.com/Anthonyive/DSCI-550-Assignment-2

Did GPT-2 trained on specific attack emails generate realistic replies and conversation?

For task 10, we used the GPT-2, hugging face api and transformers to generate the replies. We also wrote a list of possible replies from victims and attackers. For example, the attackers might write emails containing “I am an old friend of your father.” The victims who detect the scam might reply, “Stop bothering me! ” And a victim who believes in the scam might provide the Social Security number or bank account to the scammers. GPT-2 helps tokenize the context in the list and generate word sequences. After this, our script will select one reply from the gpt2-processed sentences, and combine with the hugging face API to generate a context similar to the email.

However, the generated results are not as good as we expected.

1. Although we put “Dear”, “Hi”, “To whom it may concern”, etc. into our list, there are very few emails containing these opening remarks. And for those generated replies which have opening remarks, none of them have the correct format. Originally, we wanted to put names in these opening remarks using name entity recognition on the 800 emails. However, the 800 emails are not well formatted, so using NER may not be a good idea.
2. For some replies generated, there were many repeated sentences inside the replies.
3. In some emails, there were other languages, like Chinese and Japanese. This makes the emails look badly-encoded.
4. In several emails, according to the first several sentences, it was easy to find the replies were written by a person who detected the attackers. However. The sentences after did not make any sense.

Potential causes:

1. We did not set the format of the generated replies. If we set the opening remarks as the first several words and start a new line, the format will be much more similar to emails.
2. This problem also happened in the emails generated from task 4. We might have some issues with task 4 and this might be the sequela.
3. This could be caused by the GPT-2 tokenize process. The way to tokenize English and Chinese/Japanese are quite different.
4. This can be caused by randomness. Since we selected the potential replies from the lists randomly, it might pick some replies for unsuitable situations.

What was the quality of the generated Phish Iris data representing multimedia attacks?

In task 6, we firstly manually classified the Phish Iris dataset into four attack types. Then, we trained the dataset for each attack type with DCGAN on our own machine with a GPU. Our GPU can handle no more than 5000 epochs, otherwise it will throw an OOM (out of memory error). Thus, images of each attack type were trained

with 4000 epochs. The d loss was approaching 0.5 and the g loss was approaching 1.0 during the training as we expected.

However, the quality of the fake images was way lower than we expected. Almost all the images look washed out and most of them are plain white images with some random pixels. One of the reasons is the limited number of training epochs, and it may also be due to the small number of input phish iris data.

Did DCGAN generate believable faces? How could you have improved on it?

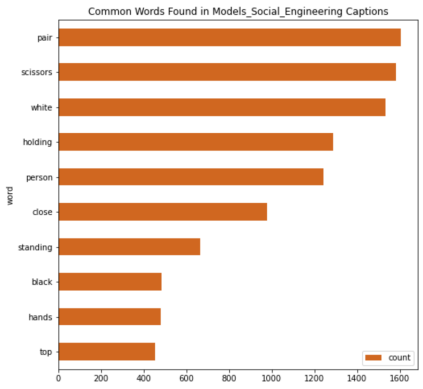
For task 7, we used DCGAN technique from task 6 and generated faces on Colab. The quality of the generated images were better than the output of task 6. However, we don't think those faces are believable. Firstly, even though we can recognize faces from the images, those faces are still blurry and don't have consistent features to be identified as faces of real persons. Also, the faces were generated independently without any email related features, so we couldn't find any relationship between the faces and the attack emails.

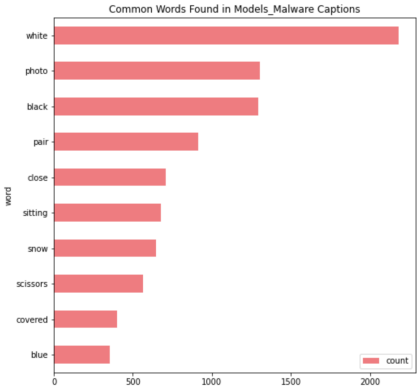
To improve the image quality to make it more realistic, we need to run the notebook with more training epochs (e.g. at least >100). However, we didn't achieve it since it will cost quite a lot of time and the capability of our computers is limited. For relations to the emails, it could be improved by using predicted features of the email attackers (such as possible age that we predicted in assignment 1) when generating faces. This will give more accurate results of what the attackers may look like.

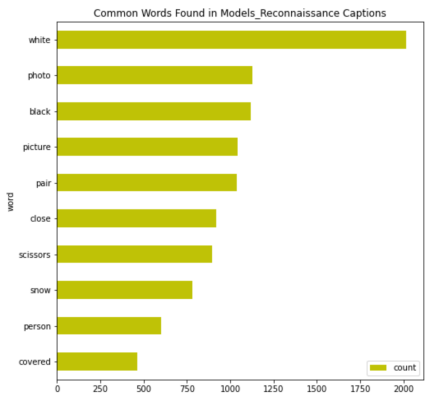
Did the Image captions for the Phish Iris datasets make sense ?

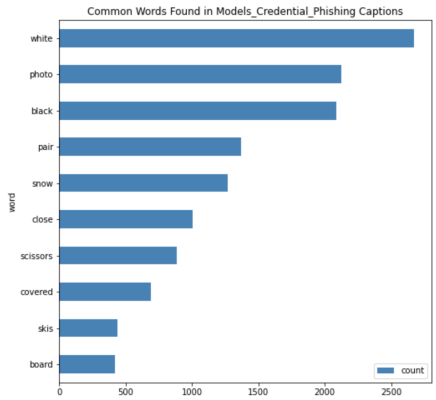
Based on the fake phish irish images from 4 different categories, the generated image captions do make some sense in generalizing and explaining what the individual photo is about. The "caption generator" is basically describing what the image is about but the majority of the images consist of pure white background with some grey dots randomly distributed on top, and therefore most of the captions have the keywords like,"cloud", "white", "snow" ,etc. Nevertheless, the image captions do not make much sense based on the context of this assignment. It only provides a few valuable information or insights for the Phish Irish datasets.

Below are common words generated from the captions and it is obvious that several words (not stop words) happen to show up on different graphs multiple times.









Was Grover able generally to identify the falsified attacks?

Grover was able to identify the machine generated emails. It turns out that all the gtp-2 generated emails are classified as machine content by Grover. However, I believe that this model has a limited ability to identify spam email attacks that are written by humans. Since the model was trained to discriminate between human written and machine generated emails, it does display accuracy in telling whether an email is generated, but not whether it is a spam attack. To make the most use of Grover, I would recommend more data in both 'machine' and 'human' for training. The training set should contain more labels and metadata such as title, article, author, and so on in order to improve accuracy. The package is nicely structured but the packages it runs on are out of date, and thus I had to stick with tf 1.x while a lot of functions are depreciated.

Was GPT-2 more effective training on only the TTR text, or did the other additional features help? Did you try them?

In Task 4, we trained a GPT-2 model based on TTR'ed text of fraudulent emails. However, the results seem very random. In fact, some of them are just simple replications. For example, in Credential_phishing/101.txt, it repeats the two sentences "Private fax service Check the box." over and over again. In general, they don't have a valid email format as we, as human, normally would have.
In Task 10, we leveraged new techniques like prompts and bags of sentences. We created prompts with different persona's to mimic what attackers and victims would reply to in their emails. Then, we randomly choose a persona type to start with an email reply. This method seems to have a better outcome as we observed several emails and fewer of them are having repetitions.

Also include your thoughts about the ML and Deep Learning software like GPT-2, Grover, Image Captioning, DCGAN, etc. – what was easy about using it? What wasn't?

GPT-2 is extremely computational heavy to train. In this assignment, we did several training tasks on GPT-2, but most of them failed to complete simply because we have poor computational power, especially on GPU memories. As a side effect, those tasks took a very long time to train too. Our teammates split the tasks, but it still takes hours, even days to finish training and generating.

GPT-2 is not very easy to use too. Luckily we have good articles explaining the details and having code out of the box, otherwise, the terms are very hard to be understood or consumed. If we think of them as “black boxes” that do things for us, they are actually relatively easy to understand in contrast.

In this assignment, we can easily get the DCGAN notebooks for task 6 and task 7. However, it was hard to figure out the correct version of the libraries, adjust hyper parameters and resize the input images at the beginning. It was also challenging to train a stable model since the training process is inherently unstable, resulting in the simultaneous dynamic training of models. To understand the process in the notebook, we searched and learned about the example of the Generator Model Architecture for DCGAN. Besides, we had learned the principle of deconvolution in deep learning. In addition, we spent a lot of time generating new images due to a large amount of data, low computer configuration, environmental settings. In order to avoid computer hardware and environmental problems, we chose to perform the face generation process on Colab.

Contributions

Yuchen Zhang ● Task 3 - TTR algorithm generation ● Help on generating half of 800 GPT-2 generated emails and more emails for training Grover ● Help on generating new fake Phish Iris attack images ● Task 10 - use GPT-2 to generated replies	Xiuwen Zhang ● Task 6 - classify the phish iris dataset into four attack types ● Train models and generate fake phish iris images ● Task 7 - generate faces for attack emails using DCGAN face generator notebook
Nat Zheng ● Task 9 - Used dockers to generate captions for new phish irish images ● Generated common used word counts graph for image captions	Zixi Jiang ● Task 8 - Grover ● Trained a model from scratch ● Make predictions on 800 generated emails
Peizhen Li ● Task 5- download dataset and test it ● Task 6, 7- DCGAN & new fake Phish Iris attack images(due to the problem of computers, needing teammates to help, Xiuwen Zhang did a lot here.) ● DCGAN - read and understand the code, search the principle of DCGAN and learn deconvolution in deep learning to adjust parameters.	Xiaoyu Wang ● Sort out TTR’ed resulting text ● Task 4 - Set Gpt-2 Simple model and generate new emails ● Generate extra machine-based fraudulent emails for Grover use