Balancing Email Datasets with GAN

Dylan Wills

20 Mar 2020

Abstract

Phishing attacks have been observed to be a key vector in most cyber security attacks in recent years. Currently organizations are limited to training awareness for their staff as the only viable option for preventing phishing attack success. Prevention methods such as manually reviewing emails are highly obstructive and cause many problems. In attempt to create tools for phishing email prevention, machine learning for detection, learning and identification are a forefront solution. One issue that arises with machine learning in big data and with developing threatscapes is the issue of imbalanced training and discrimination datasets. In attempt to provide a solution to balancing large text-based datasets, this paper proposes an LSTM model generator for future implementation into GAN's for generating lenghty semi-semantical NLP for phishing datasets.

1 Introduction

1.1 Background

Phishing attacks are the cornerstone of most cyber security attacks [7,12,13]. It was found that of 2,216 confirmed data breaches, 90% started with a phishing vector [7]. currently, organizations focus on the training and awareness of their staff to prevent phishing attacks. Tools such as PhishMe [8], PhishLabs[9], and MediaPro[23] are used by many tech giants and fortune 100 companies to provide training to staff. Despite the widespread use and interest in these tools, they cannot provide a prevention mechanism for phishing vectors.

Recently, new technologies focused on prevention have begun to emerge. IronTraps [10], and Barracude [11] are two recent examples. Prevention is focused on utilising machine learning to learn, detect and predict phishing signatures. Phishing prevention is yet to achieve an optimum solution. As phishing email volume and complexity are continuously increasing, a solution is needed for growing and balancing learning databases for predicting similar attacks in the future.

In order to balance the limited datasets available on phishing body texts, Generative Adversarial Network's (GAN's) could provide an effective model.

GAN's have seen rapid improvement since their conception. They were originally developed for image and video generation but have seen several implementations of text generating GAN's, such as SeqGAN [5], MaskGAN[3], and LeakGAN [4]. The output from text generating GAN's is generally very restricted, being applied to single sentences and short phrases [3,4,5,18,21,22]. This paper will explore the use of GAN's for larger Natural Language Generation (NLG) results. A generator for an adversarial network will be developed as part of this paper to generate email body text to replicate that used in phishing emails. The aim of the final GAN model is to balance email datasets for use in detecting and preventing phishing scams. A larger dataset increases likelihood for detecting phishy emails for a more successful and purposeful product delivery.

1.2 Generative Adversarial Networks

Generative Adversarial Networks are predominantly used for image generation [18,21,22]. They analyze binary data to determine and compare sequences of bits against predefined data sets to establish commonalities and differences. This is easy to do for images as pixels can be easily correlated to set binary sequences. NLG was originally perceived as impossible with GAN's as it cannot translate patterns to binary data [18]. Limited functionality of NLG using GAN's was made available with the introduction of WGAN-GP [20]. It has been demonstrated that GAN's can be used to generate short outputs of text, usually less than 40 characters [18,20]. This project will research the use of GAN's to generate lengthy complex bodies of text to replicate phishing emails.

2 Related Work

2.1 URL Generation using GAN's

Similar concepts have been applied to the balancing of phishing URL datasets through URL generation using GAN's [2]. In this paper, a GAN model is used with oversampling to generate possible phishy URL's based on specific rulesets. They use the improved WGAN algorithm and a Gumbel-Softmax Distribution:

$$L = Ezpz(z)[D(G(z))]Expdata(x)[D(x)] + Ex^{px[(x^D(x)21)2]}(1)$$

$$z = onehot(argmaxi[gi + logi]) \quad (2)$$

The results from the experimental URL generation are passed through VirusTotal and compared with collected phishy URL's from Paypal within their dataset [2]. The GAN, when coupled with oversampling, successfully recreated the phishy URL results.

Oversampling is proposed as the best method for balancing imbalanced datasets [2] but the work performed can easily align to a set of strict rules

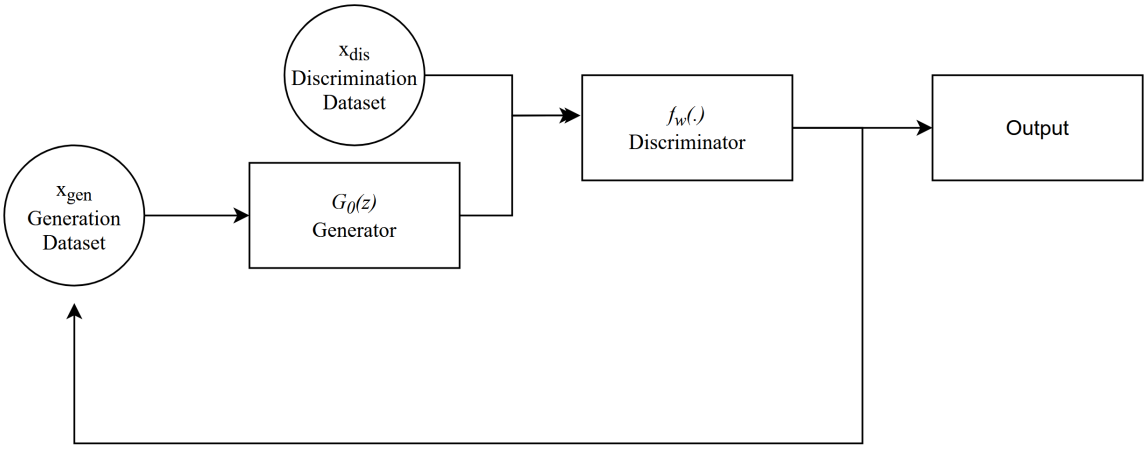

Figure 1: A generic GAN model

and feature; length of URL, non-sensical character strings, number characters, use of punctuation, etc. Many of these target features and rules would not allow oversampling and targeted generation within phishing email text due to the complexity and variation within observed phishing attacks.

2.2 Text generation using GAN's

A new solution for lengthy text generation was proposed in [1] combining WGAN-GP models with Knowledge Distillation for more refined NLP outputs from the standard GAN model. They use an auto-encoder feed to the generator of their new TextKD-GAN discriminator in lieu of the traditional one-hot input. This allows the input into the discriminator to be softened and makes discriminating much harder. Comparitively with IWGAN, TextKD-GAN provides a smoother, more readable output but only in shorter lengths of text [1]. Also, several other GAN versions have arisen to tackle the challenge of generating text. Relgan [16] trains on discrete data using the REINFORCE algorithm and gumbel-softmax to treat long term memory issues. It comparatively experiments against other notable GAN's such as MaskGAN [3], SeqGAN [5], and LeakGAN [4] to demonstrate negative log-likelihood (NLL) more closely aligned to that of real data. Although an improvement on older text GAN's, it is still restricted to shorter lengths; testing against 20 and 40 character outputs.

CatGAN [15] applies the GAN model using RNN generator and discriminators across categories, allowing the GAN to differentiate real vs fake according to several different datasets as opposed to a single category discriminator. This

could be relevant in balancing several backend datasets with a single GAN for categories such as phishing, spear-phishing, blackmail, spam, etc.

3 Design

3.1 Overview

GAN's are conventionally used for image and video generation. Implementations such as WGAN [20,21,22] and TextKD-GAN [1] have limited output lengths with their NLG. By using a tailored model of WGAN-GP [20] and an LSTM with an increased size dataset, larger bodies of text could theoretically be generated. To do this, we feed an LSTM Generator with the training dataset of sample Phishing emails and train it on sample sets. It could . Results that are deemed successful matches will be fed back in to the training dataset and returned to the user. False matches will be discarded.

This project will research the viability of and LSTM network as generator for pairing in a GAN. The implementation is written in Python 3.8 using TensorFlow 1.15 and Keras libraries. Number processing is based on Numpy. The implementation is written for a google colab environment.

3.2 LSTM Generator

The generator is compiled of a basic LSTM model which reads from the selected dataset to shape entries into sentences to generate character-by-character lengths of text. The generator samples fixed size seeds and loops through to predict each consecutive character, appending previous seeds to the formed sentences list for mixing in the next prediction.

The LSTM loads a text dataset and creates a list of unique characters. It then maps the unique characters into integers for predictive processing. Text passes through an x variable to create possible outcomes before committing a y variable as the output to gather most common characters based on the preceding text. It then builds the LSTM model on these sample sets.

From this it iterates through the dataset, sampling each character and its preceding number of characters to form predictions. Each iteration trains on the available parameters and begins with a seed. It then appends each predicted character and appends it to the seed to create the generated text.

LSTM models solve the long term memory issue encountered in most RNN's. They pass the outputs from previous iterations through the memory cell to improve on the current iterations predictions. An LSTM will first decide which integer representations are not relevant to the current iteration and discard them by sampling the previous output with the new sample input, x, through a sigmoid layer.

$$f_t = \sigma(W_f x[z_{t_{-1}}, x_t] + b_f) \tag{3}$$

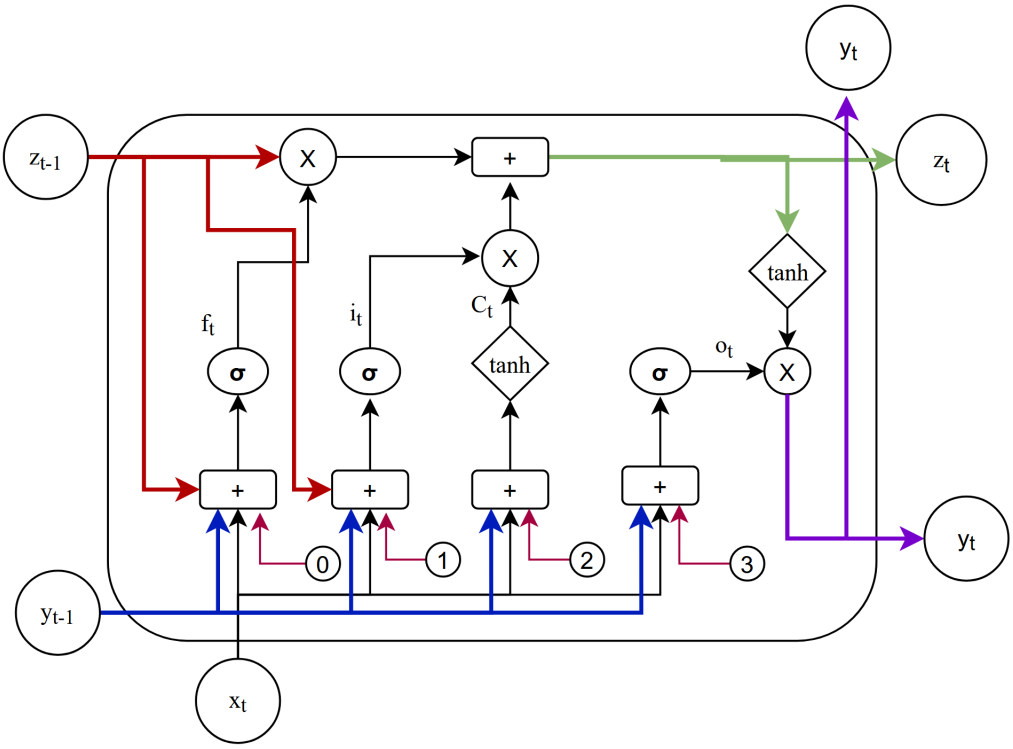

Figure 2: LSTM Model

Next, it pushes $z-1$ and x through a sigmoid layer and a tanh layer to decide which information is stored in the current cell state. The sigmoid layer predicts the information to update and the tanh creates a vector of new values, C_t.

$$i_t = \sigma(W_i x[z_{t_{-1}}, x_t] + b_i) \tag{4}$$

$$C_t = tanh(W_C x[z_{t_{-1}}, x_t] + b_C) \tag{5}$$

The old cell state, $z_{t_{-1}}$ is now updated to the current output cell state, z_t by multiplying the old state by f_t, then adding $i_t * C_t$.

$$_t = f_t x z_{t_{-1}} + i_t x C_t \tag{6}$$

In the final step, the LSTM decides what to output. The output is based on the predictive cell state. To filter the cell state it is first run through a sigmoid

layer which decides which part of the cell state is output. Then, the cell state is piped through a tanh layer and multiplied by the output of the sigmoid to filter only decided parts for results.

$$o_t = \sigma(W_o[y_{t_{-1}}, x_t] + b_o) \tag{7}$$

$$y_t = o_t x tanh(z_t) \tag{8}$$

Pseudo Code for LSTM Algorithm

Initialization: *the algorithm loads the data set and vectorizes the model.*

For each generation iterate through the model and:

Selection: *discard irrelevant cell states* $f_t = \sigma(W_f x[z_{t_{-1}}, x_t] + b_f)$

Decide Current: *Push old cell states and results through the sigma and tanh layer.* $i_t = \sigma(W_i x[z_{t_{-1}}, x_t] + b_i)$

Prediction: *Predict which information to update based on the tanh of the current cell states.* $C_t = tanh(W_C x[z_{t_{-1}}, x_t] + b_C)$

Mutation: *Update the cell state with predicted information.* $z_t = f_t x z_{t_{-1}} + i_t x C_t$

Filtering: *Filter the cell state for relevant parts to output.* $o_t = \sigma(W_o[y_{t_{-1}}, x_t] + b_o)$

Generation: *Output the generated results.* $o_t = \sigma(W_o[y_{t_{-1}}, x_t] + b_o)$

Print: *Print resulting text.*

End while

3.3 Phishing Email Corpus

The Nazario phishing corpus [6] is used in the generator as the learning data set. The Nazario phishing corpus is a collection of over 4,500 phishing emails collected from a variety of publically available phishing emails. The emails in the data set are in English. They contain phishing attempts which masquerade as both real and fake entities through a variety of scenario's.

4 Results

4.1 Experiment

LSTM's should be useful in generating longer strings of text as required in phishing email bodies. They address the issue of long term memory in most generator networks. an LSTM could be easily combined in future research with other LSTM's or discriminators to complete a GAN and improve on its generation. I use an LSTM generator fed with a large sample dataset to generate output text for comparison against captured phishing email samples. The dataset is not processed and may contain spelling errors and nonsensical text as would be expected from a phishing email. The output is compared with a separate dataset for to determine output viability.

To confirm this, outputs are compared with associated lines within the Nazario dataset [6].

4.2 Observations

The generator results are depicted in table 1. As seen, the proposed LSTM generator outputs comparable examples of phishing body text with minor grammatical and semantic issues as would be expected from the standard phishing email. It has the capability to exchange entities and hold consistency throughout the generated text.

Table 2 shows the results of the generator set to sample size of 400 and sample size of 40. It can be observed that the higher sampling size generates more stable and predictive text. The lower sample size has a higher rate of false semantics to a point where it no longer resembles that of sampled phishing datasets. This, however, cannot be observed when the sample size becomes too too large. larger sample sizes begin to generate strings of recurring characters with no observable patterns of language.

Additionally, during the experimentation phase, it was observed that larger datasets improved on the output of the LSTM in generating language. A dataset was originally pre-processed and used which provided less than 9000 samples. The resulting outputs using this training set failed to provide viable results with consecutive sentences losing structure and not maintaining the flow of the text. A newer raw dataset was inserted providing 156408 samples which resulted in the outputs such as observed in tables 1 and 2.

PhishGAN Generator	Dataset Example
your ebay session will not be and will continue as normal to update your ebay records click on the following link http update if your account information is not updated within hours then your ability to use ebay will become thank you ebay inc	your eBay session will not be and will continue as normal To update your eBay records click on the following link http UPdate If your account information is not updated within hours then your ability to use eBay will become Regards Safeharbor Department eBay Inc Copyright
that this is a security measure meant to help protect you and your account we apologize for any inconvenience sincerely paypal account review department paypal email id pp please do not respond to this email as your reply will not be received	that this is a security measure intend ed to help protect you and your account We apologize for any inconvenience Sincerely NCUA Account Review Department Please do not reply to this email Mail sent to this address ca nnot be answered
make sure you never provide your password to fraudulent websites to safely and securely access the paypal website or your account you can click in the yellow box or open a new web internet explorer or netscape and type in the paypal url paypal will never ask you to enter your password in an email for more information on protecting yourself from fraud please review our security tips at https the proved been sent to this matter paypal member paypal employs some of the most ne service and prevente bank of the west ebay inc	Make sure you never provide your password t o fraudulent persons PayPal automatically your confidential information using the Secure Sock ets Layer protocol SSL with an key length of bits the highest level commercially available PayPal will never ask you to enter your password in an email For more information on protecting yourself f rom fraud please review our Security Tips at http Protect Your Password You should never g ive your PayPal password to anyone including PayPal employees

Table 1: Results from the LSTM phishing email generator

PhishGAN Sampling = 400	PhishGAN sampling = 40
in half that is i will simply nominate you as the next of kin and have them release the deposit to you we share the proceeds would have gone ahead to ask the funds be released to me but that would have drawn a line to me and my involvement in claiming the deposit i assure you that i could have the deposit released to you within a few days i will simply inform the bank of the final closing of the	d rarely do they nominate next of kin prgvided to this is of your account is to confidential your account information case identify you to send your account to go do you do you back that hurm or moff einle wohle suckelarc wir der kunfacastsl zustum ifoweld me nen apaul heldie file busting below to ebay http we have recent recieved will receive clush please jail log in to your recommezed your paypal information to activate your account
part of our continuing commitment to protect your account and to reduce the instance of fraud on our website if you could please take minutes out of your online experience and update your personal records you will not run into any future problems with the online service however failure to update your records will result in account suspension please update your records on or before november	that your account was not accessed by an unauthorized third party because in all rights reserved designated and brands are the property of the following link http the ebay member as part of our conting a account paypal employees we advise to respend of the consube specially and account by reporting it to provide your preferences for your account and possible account and quickly restore full access we may require some specific informati

Table 2: LSTM generator sample sizes

Average readability scores were calculated for the Nazario dataset and the generated results using the Flesch-Kincaid Grade Level Algorithm. The average scores for each are shown in table 3:

Nazario Dataset	PhishGAN Results (40)	PhishGAN Results (400)
41.2	27.8	39.7

Table 3: Average Readability Scores From Flesch-Kincaid Grade Level

4.3 Conclusion

In this work, I introduced LSTM generators as a new solution to generating semi-semantic phishing email content. Phishing emails are The generator, when paired with the appropriate discriminator within a GAN would likely be an optimal solution for generating longer strings of text based on the relationship between sample sizes and quality of semantics and grammar. The sample size would provide a good tuning point for achieving the right level of realism.

The observed results show that LSTM can successfully generate text based on a dataset. As it was also observed that a datasets size significantly improved the generated text results, applying this generator within a GAN model would likely result in exponential improvement for generated text as the dataset expands. This is due to the part of the fundamentals in GAN's where successful outputs are appended to the generation dataset.

4.4 Future Work

To improve on the work conducted in this paper, proposed future works should focus on building a suitable discriminator to complement the LSTM generator model in a GAN model. Future works can use generated text samples from this paper as comparison to gauge improvement and learning. The same experiment can be conducted with a GAN model. Observations from this paper reveal factors to account for in any further GAN development where sample size, dataset size and seeding can be maintained to observe further improvements.

Alternative models for comparison are almost non-existent. One possible comparison for future generation of phishing email text is PhishGEN [24]. Current methods of phishing email detection and prevention involve manual creation of data sets. As stated by its Authors, PhishGEN may be the only current method of producing data sets with semantics and grammar consistent with phishing emails [24]. It uses probabilistic context-free grammar (PCFG) and variable interpolation to generate phishing email data sets.

References

[1] Haidar, A. & Rezagholizadeh, M. (2019) "TextKD-GAN: Text Generation Using Knowledge Distillation and Generative Adversarial Networks." arXiv preprint arXiv:1905.01976.

[2] Anand, A., Kshitij, G., Moniz, J.R.A., Park, N., Chakraborty, T., & Chu, B.T. (2018) "Phishing URL detection with oversampling based on text generative adversarial networks." In: 2018 IEEE International Conference on Big Data (Big Data), pp. 1168-1177. IEEE.

[3] Fedus, W., Goodfellow, I.m Dai, A.M. (2018) "Maskgan: Better text generation via filling in the _." arXiv preprint arXiv:1801.07736.

[4] Guo, J., Lu, S., Cai, H., Zhang, W., Yu, Y., Wang, J. (2017) "Long text generation via adversarial training with leaked information." arXiv preprint arXiv:1709.08624

[5] Yu, L., Zhang, W., Wang, J. & Yu, Y. (2017) "Seqgan: Sequence generative adversarial netswith policy gradient." In: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17). pp. 2852–2858.

[6] Nazario, J. (2005) The online phishing corpus, Available from:http://monkey.org/ jose/wiki/doku.php.

[7] Verizon Enterprise. (2019) "Verizon's 2018 Data Breach Investigations Report". Retrieved from https://enterprise.verizon.com/resources/reports/dbir/.

[8] Cofense. (2019) "Cofense PhishMe". Retrieved 16 Mar 20 from https://cofense.com/product-services/phishme/.

[9] PhishLabs. (2019) "Intelligence Actioned". Retrieved 16 Mar 20 from https://www.phishlabs.com/.

[10] Ironscales. (2019) "AI-Powered Phishing Precention and Incident Response." Retrieved 16 Mar 20 from https://ironscales.com/anti-phishing-solutions/phishing-prevention/

[11] Barracuda. (2019) "Barracude Sentinel." Retrieved 16 Mar 20 from https://www.barracuda.com/products/sentinel.

[12] Chiew, K.L., Yong, K.S.C., & Tan, C.L. (2018) "A survey of phishing attacks: their types, vectors and technical approaches." In: Expert Systems with Applications 106 (2018): pp. 1-20.

[13] Aleroud, A. & Zhou, L.. "Phishing environments, techniques, and countermeasures: A survey." In: Computers Security 68 (2017). pp. 160-196.

[14] Park, G., & Taylor, J.M. (2015) "Using Syntactic Features for Phishing Detection." arXiv preprint arXiv:1506.00037.

[15] Liu, Z., Wang, J. & Liang, Z. (2019). "CatGAN: Category-aware Generative Adversarial Networks with Hierarchichal Evolutionary Learning for Category Text Generation." arXiv preprint arXiv:1911.06641.

[16] Nie, W., Narodytskam, N., & Patel, A.B. (2019) "RelGAN: Relational Generative Adversarial Networks for Text Generation." In: International Conference on Learning Representations 2019 Blind Submissions.

[17] Shen, T., Mueller, J., Barzilay, R. & Jaakkola, T. (2020) "Educating Text Autoencoders: Latent Representation Guidance via Denoising." arXiv preprint arXiv:1905.12777

[18] Chintapalli, K. (2019) "Generative Adversarial Netowkrs for Text Generation." Retrieved 16 Mar 20 from https://becominghuman.ai/generative-adversarial-networks-for-text-generation-part-1-2b886c8cab10.

[19] OpenAI. (2019) "Better Language Models and Their Implications." Retrieved 16 Mar 20 from https://openai.com/blog/better-language-models/.

[20] Hui, J. (2018) "GAN — Wasserstein GAN WGAN-GP." Retrieved 16 Mar 20 from https://medium.com/@jonathan_hui/gan-wasserstein-gan-wgan-gp-6a1a2aa1b490.

[21] Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V. & Courville, A. (2017) "Improved Training of Wasserstein GANs." arXiv preprint arXiv:1704.00028.

[22] Anonymous. (2020) "IWGAN: An Autoencoder WGAN for Inference." In: International Conference on Learning Representations 2020 Blind Submissions.

[23] MediaPRO (2019). "MediaPRO." Retrieved 16 Mar 20 from https://www.mediapro.com/.

[24] Palka, S., McCoy, D.: Dynamic phishing content using generative grammars. In: 2015 IEEE Eighth International Conference on Software Testing, Verification and Validation Workshops (ICSTW), pp. 1–8. IEEE (2015)