URL-Based Phishing Detection

Mridul Banik
Shreya Anilkumar
Zihui Li

Our Project and Background

Conduct several experiments to detect phishing URLs from legitimate URLs, using NLP transformer models and fine-tuned them based on an legitimate-phishing URL dataset.

Based on the experiment result, analyze the reasons for the performance of the models

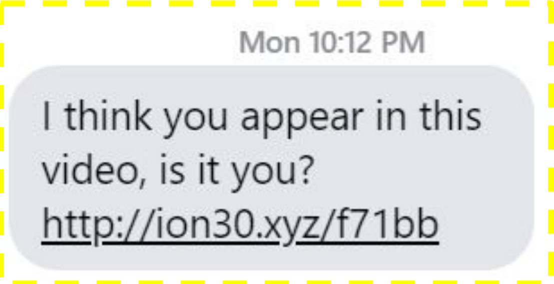
What is phishing ? What are its adverse effects?

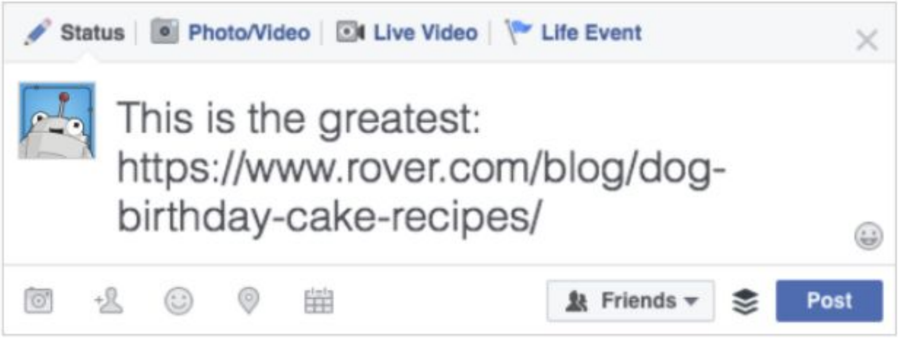
Motivation

- ➔ Sources of scam messages : Text messages, Email and attractive hyperlinks
- ➔ Effects : direct monetary loss, loss of data, reputations, reduction in productivity and value.
- ➔ URL structure : protocol://domain-name.top-level-domain/path

Example Anchor Text

In a pinch, well-written URLs can serve as their own anchor text when copied and pasted as links





Is it possible for humans to sit and check all URLs everytime we encounter one?
How to make it scalable, effective and safeguard a large number of people?

Which has a clear destination?

Our Uniqueness

- ➔ Compared different models and analyzed the reason why a model performs well or bad.
- ➔ Hyperparameter tuning for average performing models to improve its efficiency.
- ➔ By using fine-tuned model, it takes short time to detect a URL link without visiting that link.

Methodology

- Train.csv and Test.csv
- Each file contains two columns . one for url and other one is for labelling
- Loaded into dataframe
- Make both train and test set balanced my taking 50 percent of both class
- 24310 url in training set
- 6078 urls for testing set
- Converted all urls into lower case
- Defined label 0 as Legit and 1 as Phish
- 80:20 split of training set into training set : validation set

Experimentation

1. BERT (Bidirectional Encoder Representations from Transformers)
   a. BERT-base-uncased
      i. Hyperparameter : Lr : 0.00008
      ii. Batch Size : 16, Num_epcohs : 3 , Optimizer : Adam
2. RoBERTA (Robustly Optimized BERT Pre-training Approach)
   a. RoBERTA-base
      i. Hyperparameter : Lr : lr_scheduler (initial_learning_rate=0.00005, end_learning_rate=0.0)
      ii. Batch Size : 8, Num_epcohs : 3 , Optimizer : Adam
3. ALBERT (A Lite BERT)
   a. ALBERT base v1
      i. Hyperparameter : Lr : lr_scheduler (initial_learning_rate=0.00005, end_learning_rate=0.0)
      ii. Batch Size : 8, Num_epcohs : 3 , Optimizer : Adam
   b. ALBERT base v2
      i. Hyperparameter : Lr : lr_scheduler (initial_learning_rate=0.00005, end_learning_rate=0.0)
      ii. Batch Size : 8, Num_epcohs : 3 , Optimizer : Adam

Experiment (Continue)

4. Distillation (Approximation)

   a. Distilbert-base-uncased
      i. Hyperparameter : Lr : lr_scheduler (initial_learning_rate=5e-5, end_learning_rate=0.0)
      ii. Batch Size : 128 , Num_epcohs : 2 , Optimizer : Adam
   b. Distilbert-base-uncased-finetuned-sst-2-english
      i. Hyperparameter : Lr : 0.00008
      ii. Batch Size : 16, Num_epcohs : 3 , Optimizer : Adam
   c. Distilroberta-base
      i. Hyperparameter : Lr : lr_scheduler (initial_learning_rate=0.00005, end_learning_rate=0.0)
      ii. Batch Size : 128, Num_epcohs : 2 , Optimizer : Adam

Time

Fine tuning process for most models:

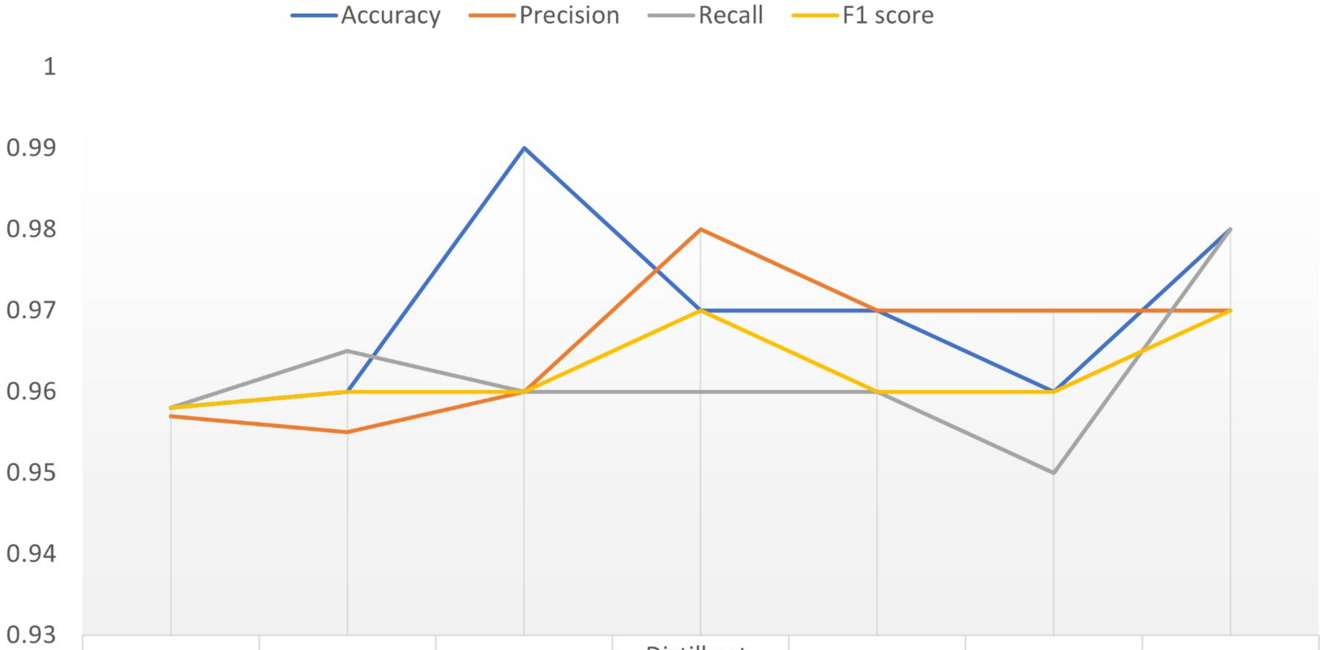
CPU (i7 3.2G) : 6-9 hours

High performance GPU(NVIDIA Tesla P100): About 20 mins

Predict(Detect whether a URL is legitimate or not):

5-10s

Graphical representation of the results



	Roberta-base	Albert-base-v2	Albert-base-v1	Distilbert-base-uncased-finetuned-sst-2-english	distilbert-base-uncased-	distilroberta-base	BERT-base-uncased
Accuracy	0.958	0.96	0.99	0.97	0.97	0.96	0.98
Precision	0.957	0.955	0.96	0.98	0.97	0.97	0.97
Recall	0.958	0.965	0.96	0.96	0.96	0.95	0.98
F1 score	0.958	0.96	0.96	0.97	0.96	0.96	0.97

Why the models succeed in URL classification/prediction?

An example:

Url: “www.myfitnesscard.de/berlin/schwimmhalle-finckensteinallee”

After tokenized: 'input_ids' : [2, 13, 6483, 9, 915, 11765, 720, 6648, 9, 546, 118, 23171, 118, 2992, 3976, 3363, 5060, 62, 8, 5617, 2601, 13002, 192, 4716, 3]

Let’s cut off the array [2, 13, 6483, 9, 915, 11765]

Restore the text: tokenizer.decode([2, 13,6483,9, 915, 11765]) www.myfit

Thanks to the Tokenizer! It automatically tokenize the meaningless URL into arrays of meaningful words

What about digits in URL?

'url': 'www.movient.net/archives/46591893.html'

After tokenized: 'input_ids': [2, 13, 6483, 9, 22607, 2877, 9, 2328, 118, 23941, 18, 118, 3516, 3902, 21519, 9, 15895, 3]

tokenizer.decode([2, 13,3516,3902,21519])

Restore text: 46591893

Hypothesis : Legitimate URL ofen start with legitimate domain and subdomain text that have been labeled in train dataset. Possible heavy weights on these two sections.

Discussion/ observation

- ➔ NLP models cannot be applied on URLs directly (50% acc before fine-tuning), but can be used as training models with proper hyper-parameters.
- ➔ Overall, Albert base v1 performs best with highest test accuracy
- ➔ Hyperparameter does matter (Oscillation)
- ➔ Almost all models attain > 95% in all metrics (accuracy/precision/F1/recall).
- ➔ Lightweight models: Albert v1 & v2, DistillBert(Less than 100M)
- ➔ The reason why all the models provided similar kind of accuracy is models had improvement over extracting contextual information. However, tokenizers have similar functions - tokenize meaningless text into understandable words, all the models provided similar types of accuracy.

Conclusion

- ➔ The result verified hypothesis: NLP transformer models can be applied on URL dataset.
- ➔ There is no need to preprocess URLs, fast and easy to train.
- ➔ Easy to apply on lightweight devices (update fine tuned parameters/weights).
- ➔ Need very short time to detect a phishing URL.

Future work

- ➔ Test on different URL datasets to see if the results are the same.
- ➔ Modify code to Pytorch version to test more models. For now, only limited transformer models accept Tensorflow. Our experiment is limited by the number of applicable models.
- ➔ Expand the implementation to spam text/email detection.

Thank you!

Questions?