Jon Gellin

Application for Detecting Maliciousness in Text Input

Metropolia University of Applied Sciences
Bachelor of Engineering
Electrical and Automation Engineering
Bachelor’s Thesis
13 November 2022

Abstract

Author:	Jon Gellin
Title:	Application for Detecting Maliciousness in Text Input
Number of Pages:	30 pages
Date:	13 November 2022

Degree:	Bachelor of Engineering
Degree Programme:	Electrical and Automation Engineering
Professional Major:	Automation
Supervisors:	Erkki Räsänen, Principal Lecturer

The goal of this Final Year Project was to create a software solution to combat text-based scams in an increasingly volatile field of cybercrime.

Email based attacks are by far the most common initial vector in cyberattacks. Technical implementations can always be made more robust, but the human mind and its vulnerabilities are not as easily patched. By offering an application which allows a user to insert text they are suspicious of, and receive feedback based on the input, users might be able to avoid potential scams and learn to stay safe from them in the future. In a best-case scenario, potentially even educating others about them.

Many organizations offer solutions such as spam filters and reporting functionality to combat cyberattacks. These solutions do well at reducing the risk but are not foolproof, often doing nothing in terms of educating the user. The web application created in this project offers a universal tool, not tied to any certain communications platform, allowing users with no registration or cost requirements to receive instantaneous help combating cybercrime.

To achieve the set goal of the Final Year Project, this thesis examines potential approaches to detecting malicious intent in text. In addition, this thesis discusses methods to create a simple web application from start to finish. The result is a web application allowing a user to input text for evaluation, and receive feedback based on the estimated maliciousness of the text.

Keywords: artificial intelligence, machine learning, phishing, cybersecurity, security awareness, social engineering

Tiivistelmä

Tekijä:	Jon Gellin
Otsikko:	Sovellus haitallisuuden tunnistamiseen tekstisyötteessä
Sivumäärä:	30 sivua
Aika:	13.11.2022

Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Sähkö- ja Automaatiotekniikka
Ammatillinen pääaine:	Automaatiotekniikka
Ohjaajat:	Yliopettaja Erkki Räsänen

Tämän insinöörityön tavoite oli luoda sovelluspohjainen ratkaisu tukemaan kyberrikollisuuden vastaista kamppailua. Insinöörityö tutkii erilaisia ratkaisuja tekstin haitallisuuden tunnistamiseen sekä selainpohjaisen sovelluksen julkaisemiseen.

Useimmat kyberhyökkäykset alkavat sähköpostitse. Teknisiä ratkaisuja voi aina parantaa tehden niistä vaikeammin murrettavia, mutta ihmismieli ja sen haavoittuvuudet eivät ole yhtä yksinkertaisesti paikattavissa. Tarjoamalla sovelluksen, joka antaa loppukäyttäjän syöttää tekstiä ja saada syötteeseen perustuvaa palautetta tekstin haitallisuudesta, saattaa yhä useampi pysyä turvassa huijauksilta.

Monet viestintäalustat sisältävät roskapostin suodattimia, jotka vähentävät paljolti loppukäyttäjien saamia huijausviestejä. Nämä suodattimet eivät kuitenkaan ole täydellisiä, sillä hyökkääjät innovoivat jatkuvasti uusia tapoja saada viestinsä perille. Tässä insinöörityössä keskitytään ratkaisuun, joka toimii viestintäalustasta huolimatta. Työn lopputuloksena oli sovellus, johon käyttäjä voi alustasta riippumatta kopioida epäilyttävän tekstin ja saada sille koneoppimiskeinoin räätälöidyn palautteen.

Avainsanat:	tekoäly, koneoppiminen, verkkokalastelu, kyberturvallisuus, tietoturva, sosiaalinen manipulaatio

Contents

List of Abbreviations

AI:	Artificial Intelligence. A non-living intelligence able to perform cognitive actions and simulate reasoning, allowing it to solve problems typically solved by intelligent living beings.
ML:	Machine Learning. Discipline of training a software solution to perform predictions and actions based on substantial amounts of training data.
HTML:	Hypertext Markup Language. A standard markup language consisting of different elements informing a web browser on how to display a page’s contents.
PaaS:	Platform-as-a-Service. A cloud-based development and deployment environment used for hosting diverse types of applications.

1 Introduction

Cybercrime is a constantly growing issue for both individuals and businesses alike, email-based attacks being by far the most common initial vector in cyberattacks. Email based attacks combine both technical tricks and social engineering to achieve desired results.

Email based attacks, commonly known as phishing, describe the malicious practice of sending emails designed to induce the recipient into revealing sensitive information or to perform a desired action with positive outcome for the malicious actor.

Social engineering on the other hand is a commonly used term within the context of information security to describe manipulation of the human factor. A social engineer utilises psychological manipulation to have individuals perform certain actions, such as revealing sensitive information.

The human factor is often the key defence to breach for malicious actors, as systems can be built with increasing security levels, but human psychology remains vulnerable. According to data from 2014, more than 90 percent of the 191.4 billion emails sent each day are spam, and 60 percent of all attacks had the “human factor” as a major piece in the attack (1).

A successful cyberattack usually involves a breach of information security. A malicious actor might for example gain access to a company network, allowing them to extract information or install malware. Ransomware attacks pose a significant risk for organisations currently, as in those attacks the malicious actors encrypt data and request a ransom for the encryption key. Large scale ransomware attacks can disrupt business functions, or even disable critical infrastructure such as power and water grids.

The goal of this Final Year Project was to create a software solution to combat social engineering. The fundamental aim was to create a web-based solution, allowing users to input text, and receive a probability score on the maliciousness of said text. This thesis describes the process, outcomes, and learnings from the project.

2 Background

To solve the problem of identifying malicious intent in text with a software solution, one could utilize one of the three generally employed approaches:

- **Denylist** – A denylist is a list of features, such as words or domains that are known to be malicious. These features are detected, and actions are performed depending on set rules. A denylist is a recommended approach if a low rate of false positive matches is required. A false positive match meaning an instance where a legitimate message is detected as malicious. However, as a denylist is based on known features, it requires frequent updating and does not perform well against e.g. new types of attacks such as a new phishing campaign (2).
- **Rule-based heuristics –** Implementing this approach requires the manual creation of relevant heuristics, such as the length of the message, or the number of links it might have. One could analyse data to find indicators such as a set number of links which often indicates that a message is malicious. This approach is often seen as unsuitable for general solutions, as messages can be complex and used are used for a wide array of purposes. Therefore, finding effective heuristics that do not cause a large number of false positives is nigh impossible.
- **Machine learning –** A machine learning approach to identifying malicious intent utilises features in the email similar to ones used in rule-based heuristics. The difference being that in machine learning

the model finds the features from the data instead of the features being manually researched (3).

From the three approaches described above, machine learning seems like the most suitable solution to this project, as it offers a good combination of efficiency and accuracy. Machine learning does however require a significant pre-existent dataset, while rule-based heuristics can potentially be invented purely based on domain knowledge or lesser amounts of data.

3 Machine Learning

Artificial intelligence (AI) is a wide field of research, focused on studying intelligent agents such as animals and humans and creating intelligent systems, able to perceive information and take actions leading it closer to its goal. An AI can perform cognitive actions and simulate reasoning allowing it to solve problems usually thought to only be solvable by intelligent living beings.

Machine learning (ML) is considered a subset of AI. In machine learning, the idea is to train a model to perform tasks that are commonly left for humans due to their complexity. These tasks could be e.g. distinguishing certain objects in images, playing chess or even solving protein folding among others (4). Machine learning is the process of using mathematical methods to allow a computer to analyse and learn from data.

3.1 Model Choice

Machine learning solutions may be distinguished by grouping them into three main groups depending on the learning process and intended use case. Each of these groups have large disparities in the type of data they require and the tasks they can accomplish.

3.1.1 Supervised Learning

Supervised learning is used to solve two main problem types, classification problems and regression problems. Learning the mapping from an input (X) to the output (Y) is the goal, and to accomplish this the correct outputs must be known in the training data.

Classification problems mean that a certain input is given a certain classification. In its simplest form a certain input can be classified with a value between zero and one, meaning that the model returns a calculated probability of something being true or false based on the data it has been trained with. (5)

An example of training data that could be used for this type of model is the ImageNet dataset, providing researchers with image data of over 100 000 distinct types of objects and on average 1000 images for each object type (6). This dataset can be utilized to create machine learning models for classification problems such as object recognition with great accuracy.

Regression problems are related to continuous data. An example task could be predicting the price of a vegetable, given the season, area, brand, and store.

3.1.2 Unsupervised Learning

Unsupervised learning, as the name suggests, is the opposite of supervised learning. No labelled dataset is used here, instead the models' goal is to find patterns within the data to be able to predict output.

Unsupervised learning is utilized for three main types of tasks, clustering, association, and dimensionality reduction. Real world applications include anomaly detection, where substantial amounts of data are combed for atypical data points such as sensor faults in processing plants.

Unsupervised learning is known to bear the risk of inaccurate output, as the result of e.g. the model finding unforeseen patterns. Unsupervised learning also often results with non-transparent models, making it hard to analyse why a certain input leads to a certain output. (7)

3.1.3 Reinforcement Learning

Reinforcement learning differs largely from the previously described types of learning. This type of learning results in a model which outputs a sequence of actions, the sequence often containing a vast number of steps.

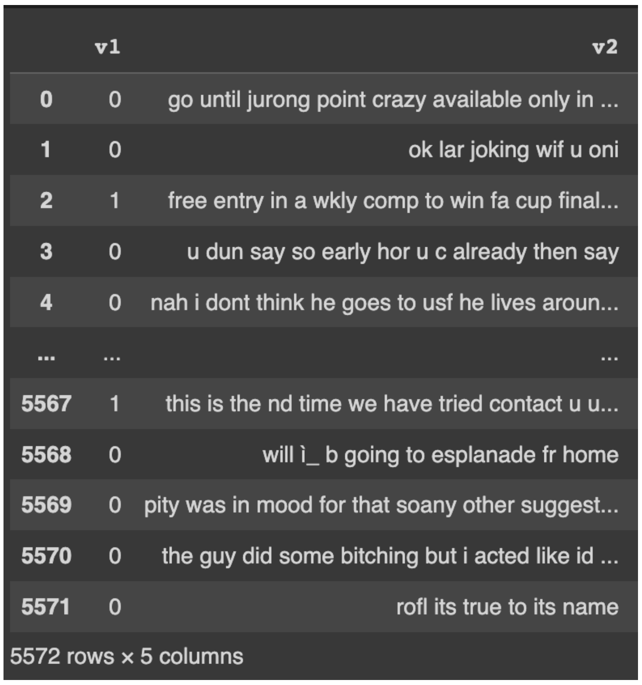
A simplified example use case for this type of model could be collision avoidance. In this case the model is given a starting point and an end goal with obstacles in between. The further the model can get without colliding with the obstacles, the more it is rewarded. With enough iterations, the model will eventually learn to pass the obstacles safely. The problem here is that if the obstacles are always situated in the same places, the model might learn to only pass this course. Therefore, during learning the model would have to encounter many different courses to learn to deal with many different types of scenarios. In the end, with the correct reward functions, the model could for example be used to define the most efficient path to reach a goal.

Reinforcement learning is used in applications such as chess engines, autonomous vehicles, healthcare, and marketing (8).

3.2 Dataset (Analysing Data)

A dataset has several important features. As previously stated, machine learning requires a large dataset to achieve a well performing model. As an example, Google Translate uses trillions of data points, or examples as a basis for the functionality (9). However, a model is only as good as the quality of its dataset. If a model is taught using e.g. erroneously labelled data, the model will function poorly when used for real tasks. This means, that the dataset should be carefully chosen or acquired, and studied before it is used.

The dataset chosen for this project contains 5 572 messages, labelled as either spam or ham, ham meaning that the message is legitimate. The messages are all in English and the dataset is composed of multiple smaller sets of data gathered from public sources (10). As there is no clarity on how reliable the labelling is, the first step is to analyse the data to see if clear errors can be spotted. Figure 1 displays some example data points found in the dataset, after loading it to a Pandas data frame. Importing the dataset into a data frame makes manipulation of the data simple in later stages of model training. Being able to differentiate between spam and ham can at times be quite hard without any additional context. When analysing a string for maliciousness, the easiest way to gain certainty of its purpose is to analyse its payload, meaning the action the recipient is supposed to take or the part they are supposed to interact with. This could for example be a link or an attachment in the message.



	v1	v2
0	0	go until jurong point crazy available only in ...
1	0	ok lar joking wif u oni
2	1	free entry in a wkly comp to win fa cup final...
3	0	u dun say so early hor u c already then say
4	0	nah i dont think he goes to usf he lives aroun...
...	...	...
5567	1	this is the nd time we have tried contact u u...
5568	0	will ì_ b going to esplanade fr home
5569	0	pity was in mood for that soany other suggest...
5570	0	the guy did some bitching but i acted like id ...
5571	0	rofl its true to its name

5572 rows × 5 columns

Figure 1 Example data points found in the dataset.

The distribution of labels is also important, as training a model with an imbalanced dataset can lead to skewed weights and therefore deficient performance. In the dataset chosen for this project, the data is distributed to be 86.6% spam, and 13.4% ham, as displayed in figure 2. To counteract this, the dataset must be resampled to even out the ratio, or a model able to handle the oversampling of one label must be chosen.

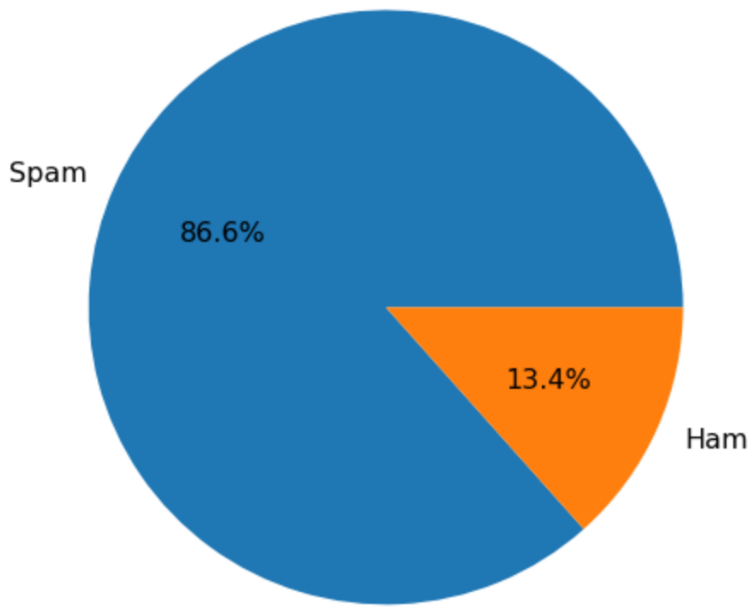


Figure 2 Distribution of labels in the dataset

When a dataset has been chosen and analysed, the data is split into a training set and a test set. As the goal is to create a model which generalises well to new data, a test set allows us to see how well the model is performing on new data and to iterate when needed. Making sure the data in both sets differ, and that no duplicates are found in both sets is particularly important, as testing the model on familiar data might lead to skewed metrics and a model which performs poorly when used for its intended task.

3.3 Training the Model

With supervised learning identified as the most suitable approach for a classification problem such as the one in this project, the next step is to choose the correct type of language model. Neural language models, such as BERT (Bidirectional Encoder Representations from Transformers) represent the current state-of-the-art approach in solving language problems with AI. (11) This type of model utilises Transformers, a key technical innovation allowing the model to take word attention into account, meaning that the model can e.g. recognize differences in meaning depending on word placement in sentences. This technical innovation was introduced in the paper Attention Is All You Need (12) published in 2017.

To achieve great accuracy, one could utilize transfer learning to fine-tune pre-existing language models, such as BERT. This approach would utilise previous research and processing, allowing one to create models with millions of parameters with relatively cheap training costs.

Due to resource constraints such as the lack of dedicated graphical processing units and to maximise personal learning, utilising the techniques presented previously to build a model without utilising pre-existing base models was chosen as the appropriate approach. In machine learning projects, it is also commonly advised to begin with smaller scale models to test the solution before scaling up.

When training a neural language model, the first step is to pre-process the data for tokenisation. The data used for training the model is text, and for a successful tokenisation the text must be cleaned of unnecessary typographical symbols and punctuation. These symbols can be for example dashes, slashes, exclamation- or question marks.

With the data pre-processed, the next step is to tokenise the training data. Tokenisation is the process of dividing larger entities into smaller parts. For example, the string:

“When do you think the pandemic is over?”

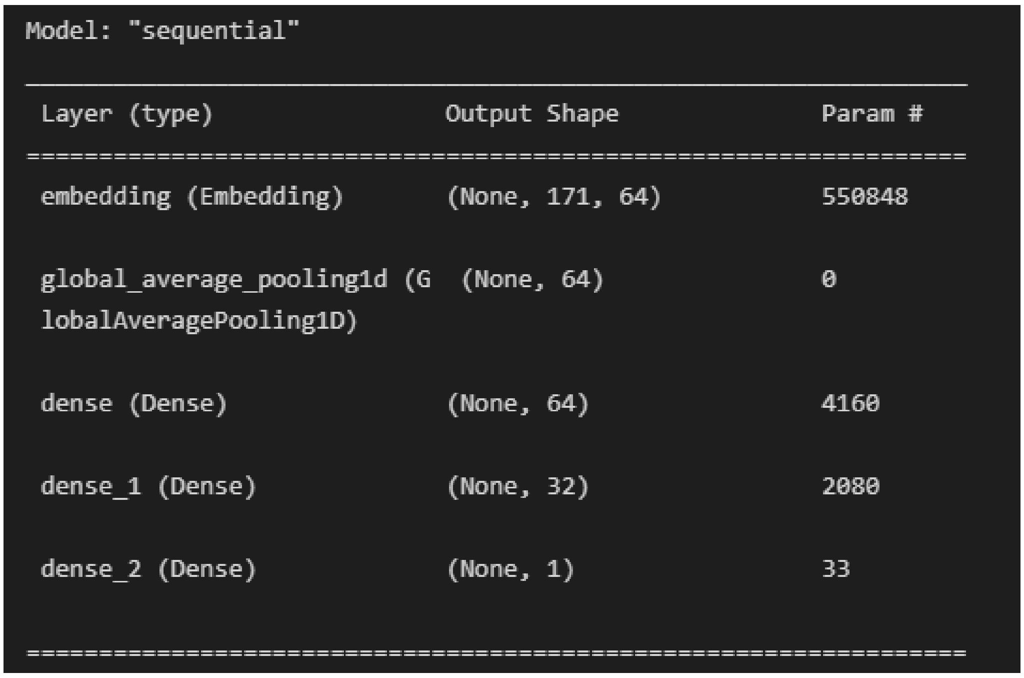
In its tokenised form would result in something similar to this:

‘when’ ‘do’ ‘you’ ‘think’ ‘the’ ‘pandemic’ ‘is’ ‘over’

This allows easier handling of the strings’ contents during the later training stages, for example by enabling it to count word frequencies or to take their positions and attention into account. (13)

The tokenised text can now be transformed into integers. Each word will be assigned a corresponding integer value, which can then be used for training the model.

Figure 3 summarises the finished model used in this project. The model type is sequential, meaning that the model is a linear stack of layers.



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 171, 64)           550848

 global_average_pooling1d (G  (None, 64)               0
 lobalAveragePooling1D)

 dense (Dense)               (None, 64)                4160

 dense_1 (Dense)             (None, 32)                2080

 dense_2 (Dense)             (None, 1)                 33

=================================================================
```

Figure 3 Summary of the model structure and its features

The dense layers are widely used in neural networks. The dense layers are used to perform matrix-vector multiplications, applying changes such as rotation, scaling, and translation to input from the previous layer.

3.4 Evaluating the Model

Accuracy represents the percentual number of correctly classified data points. Higher accuracy intuitively means better results, but in many applications different types of errors have a vastly different weight. In the case of this project, an error of classifying malicious input as non-malicious has a much worse potential outcome than classifying non-malicious input as malicious. Therefore, to better be able to evaluate the model, additional metrics must be utilised.

Precision represents the amount of true positive results divided by the sum of true positive and false positive results. This value should ideally be as close to one as possible, as an increased amount of false positive results leads to a result further away from one. In the case of this project, a true positive result means classifying something malicious correctly as malicious, and a false positive result would mean that a non-malicious input would be classified as malicious.

A third metric one can utilise is recall. This metric should also ideally be as close to one as possible. The metric can be calculated by dividing the amount of true positive results with the sum of true positive results and false negative results. False negative results being times when a malicious input is classified as non-malicious. As earlier stated, these types of errors have the worst potential outcome in this models’ planned application.

A final metric one can utilise is the F1 score. The F1 score metric takes into account both precision and recall and is widely regarded as a better measure than accuracy. The F1 score can be calculated as shown in equation 1.

$$F1\ Score = 2 * \frac{Precision * Recall}{Precision + Recall} \qquad (1)$$

Evaluating the model created for this with the metrics presented previously returns the following values:

- Accuracy: 0.9878
- Precision: 0.9178
- Recall: 0.9324
- F1 Score: 0.9250

These results are quite promising, but one must consider that they are calculated using test data separated from the training data, still originating from the same data set. Real world results may be very different.

3.5 Text Evaluation with the Model

Once the model weights have been saved, strings of text can be evaluated by first performing the same processing on the string as done to the training data, and then evaluating it.

To make this process more efficient, a function shown in listing 1 was written, which performs all the necessary steps and can be called later when the model has been deployed.

```
def evaluate_text(text):
     whitespace = re.compile(r"\s+")
     web_address = re.compile(r"(?i)http(s):\/\/[a-z0-9.~_\-\/]+")
     user = re.compile(r"(?i)@[a-z0-9_]+")
     text = text.replace('.', '')
     text = whitespace.sub(' ', text)
     text = web_address.sub('', text)
     text = user.sub('', text)
     text = re.sub(r"\[[^()]*\]", "", text)
     text = re.sub(r"\d+", "", text)
     text = re.sub(r'[^\w\s]','',text)
     text = re.sub(r"(?:@\S*|#\S*|http(?=.*://)\S*)", "", text)
     processedText = text.lower()
     FinalText =
     tf.keras.preprocessing.sequence.pad_sequences(tokenizer.texts_to_sequenc
     s([processedText]), padding='pre', maxlen=171)
return(model.predict(finalText))
```

Listing 1. Function for string processing and text evaluation

The function receives a string, which it then processes similarly to how the training data was processed, stripping it from special characters and returning it in lower case. The text is then tokenized used for evaluation. Eventually, the function returns a value between zero and one. The closer the value is to one, the higher the confidence of maliciousness.

4 Deploying the Model

As the goal of this project was to build a web app allowing users to interact with the model by inserting their data and receiving a classification, creating a website with an interactive user interface is required. Creating a website is quite simple with the tools available today, such as Platform-as-a-service (PaaS) solutions like Heroku. The route used for deploying the model for this project includes choosing a web development framework, designing, and coding the website, creating an app for running the website and handling the data transfer, and eventually hosting the app on a platform.

4.1 Creating the Website

Choosing the web development framework is the first step to deploying a model for web usage. Both Django and Flask are popular choices for web development frameworks.

Django offers a fast, scalable, and secure framework often chosen for large projects. Django is a full-stack framework, suitable for multi-page applications with dynamic HTML pages. It supports popular relational database management systems such as MySQL and Oracle. However, Django does not allow developers to have full control over the modules and functions as it has built in libraries. (14)

Flask offers a lightweight, flexible, and independent framework. Flask has a modular approach enabling developers to use libraries and extensions of their choosing, with no inherit dependencies on external libraries. (15)

The web development framework chosen for this project is Flask. As the project only requires a single-page web app with minimal functionality, Flask will cover all the necessities, while being fast and simple to set up and iterate on.

Setting up a virtual environment is highly recommended, as it allows managing dependencies conveniently. With a virtual environment one can install only the needed packages with the correct versions for synergy. For this project, Conda was used to set up a virtual environment for the Anaconda Python environment. Conda is widely used due to it being a combination of both a package manager and an environment manager, simplifying dependency management (16).

With the virtual environment set up, and flask imported, development of the website can begin. As a minimum viable product, this project requires a website which allows the user to insert a string, and to receive a feedback based on the value returned by the model. When the string is input, it must be processed similar to how the training data was processed. The processed text can then be evaluated, and the evaluation result is used to identify which type of feedback to return to the user.

4.1.1 Designing the Website

Determining the target audience plays an important role in designing a user interface. As the product offers a solution related to security, and scams affect everyone, the UI should be easily accessible for an as wide as possible audience. Users should be able to easily distinguish what the website is offering them, and the UI should be efficient allowing them to insert their text, and to receive feedback fast with no required extra steps.

To achieve a clear, informative layout, the UI was designed to contain an informative title, a short guide-text telling the user how they should interact with the application and what to expect, a large text field for user input and below that, a field for feedback. The feedback field should be dynamic, with changing information depending on the verdict.

In addition to just giving a score for the likelihood of maliciousness for the text input, it was thought best to also offer some tips on how to proceed depending on what the outcome is. According to data provided by Hoxhunt (17), employees in sales, business development and marketing departments are the most likely user group to fall for their phishing simulations. However, the failure rate does not fluctuate by much between the most common departments. Therefore, crafting tips for an as wide as possible audience, without focusing on specific types of attacks was deemed the best approach.

4.1.2 Coding the Website

Coding the website requires basic knowledge of Hypertext Markup Language or HTML for short. HTML is the standard markup language for creating web pages, consisting of a series of elements informing the browser how to display the contents (18).

Listing 2 displays part of the source code for the web site created for the project.

```
<div class="content">
<h1>Maliciousness Prediction</h1> <!--These fields are always shown-->
<p>
     Input your text and receive an AI -powered guess on whether the text is
     malicious or not!
</p>
     <form action="/" method="POST">
     <input type="text" name="input_string" maxlength="171"
     placeholder=" Input your text here">
     <input type="submit"
     </form>
</div>
```

Listing 2. Partial source code of the website.

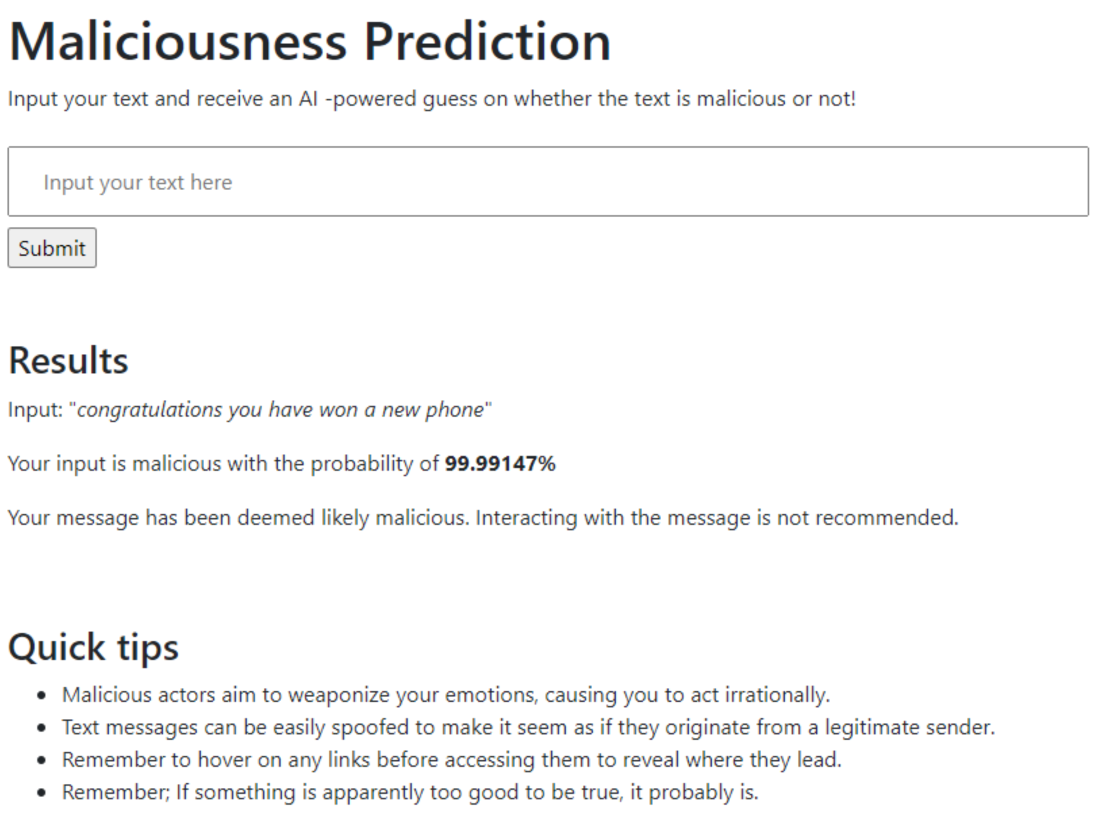
The title field is always displayed, as is the form for user input. The maximum length of input is set to 171, as that is the maximum length of a Short Message Service message. Short Message Service (SMS) being the common component used in mobile device systems for messaging. This maximum length has also been used during training of the model and is also used when processing user input for evaluation, as seen in chapter 3.4.

When text has been input, the feedback will be shown. Listing 3 showcases the logic behind displaying the feedback.

```
{% if feedbackState %} <!--feedbackState = True when feedback is requested-->
<div class="content">
<h3>Results</h3>
<p>Input: "<i>{{o}}</i>"</p>
<p>Your input is malicious with the probability of <b>{{r}}%</b></p>
{% if m %}
<p>Your message has been deemed likely malicious. Interacting with the
     message is not recommended.</p>
{% else %}
<p>Your message has been deemed unlikely to be malicious. You should still
     proceed with caution!</p>
{% endif %}
</div>
{% endif %}
```

Listing 3. Logic for choosing the correct feedback.

When feedback is requested, the variable feedbackState is set to True, therefore enabling the if-clause. The user is then shown their input and the probability score given by the model. If the probability score is above 0.6, the user is told that the input is likely malicious, and therefore interacting with the message is not recommended. The figure 4 demonstrates the view when an input likely to be malicious is given.



Maliciousness Prediction

Input your text and receive an AI -powered guess on whether the text is malicious or not!

Results

Input: "*congratulations you have won a new phone*"

Your input is malicious with the probability of **99.99147%**

Your message has been deemed likely malicious. Interacting with the message is not recommended.

Quick tips

- Malicious actors aim to weaponize your emotions, causing you to act irrationally.
- Text messages can be easily spoofed to make it seem as if they originate from a legitimate sender.
- Remember to hover on any links before accessing them to reveal where they lead.
- Remember; If something is apparently too good to be true, it probably is.

Figure 4 The user interface when a malicious input has been given.

Any score below the 0.6 threshold will return a message about the input being unlikely to be malicious. Avoiding absolutes in the wording is best here, as the results are based on educated guesses by the model.

4.1.3 Coding the Backend

To run the web site, an application must be created which renders the HTML file created previously, handles the input data, performs the evaluation with the model and then returns a verdict. In a previous chapter, Flask was deemed to be the most suitable web development framework for this project. With Flask, rendering the template is quite straightforward, as listing 4 displays.

```
app = Flask(__name__)

@app.route("/", methods = ['GET', 'POST'])
def site():
if request.method == 'GET': ##What is initially shown
     act = False
return render_template('index.html', a = act)
if request.method == 'POST': ##When user input is received
     oInp = request.form["input_string"]
     result = eval.evaluate_text(oInp) * 100
     if result >= 0.6: ##Deciding which feedback is shown
          mal = True
     else:
          mal = False
     act = True
fResult = re.sub(r"[\[\]]",'',str(result)) ##formatting the result
return render_template("index.html", o=oInp, r=fResult, m=mal,
a=feedbackState) ##returning all the necessary values

if __name__ == "__main__":
app.run()
```

Listing 4. Code for rendering the HTML template.

With these 19 lines of code, the app will render the HTML template defined in the previous chapter. When user input is received, it is stored in a variable, the contents of which are given to the function for evaluating input. The verdict sets a variable used to determine which feedback is shown, and in the end the necessary values are returned.

4.2 Application Deployment

Deploying the application requires a server which can manage the communications and computation needed. This need can be covered by a more traditional solution, as in running the application on a physical server, or then by utilizing a more modern solution: Platform as a Service (PaaS).

For this project, using a PaaS solution was a logical option due to its ease of use and cost efficiency. PaaS solutions allow developers to build scalable applications, while paying for only the resources needed. Using a PaaS solution allows the developers to avoid the hassle of physical infrastructure maintenance, and the problem of scaling the infrastructure to the applications needs. Many PaaS solutions also provide a free tier which includes enough resources for the simplest applications.

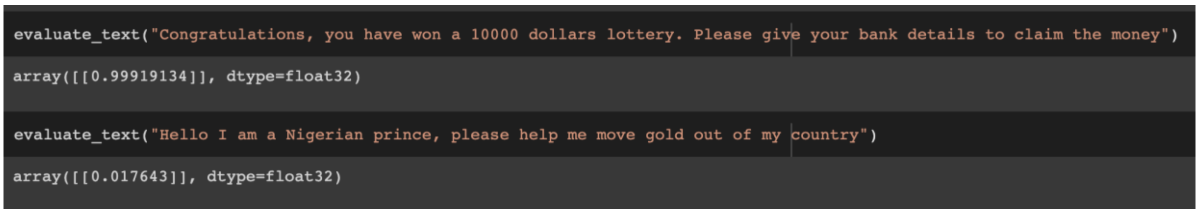
Some commonly used PaaS solutions currently include Heroku, Google Firebase and Amazon Elastic Beanstalk. These solutions offer different benefits, such as integration opportunities and regions of operation.

For the scope of this project, any of the aforementioned solutions would cover the needs. Heroku was eventually chosen due to its straight-forward nature, allowing one to easily link up a GitHub repository and deploy the code within. Heroku utilises Amazon servers both in the United States and in Europe, meaning that the latency will be sufficient in a large part of the world. (19; 20)

Heroku applications run in so-called dynos, defined as lightweight containers that run the specified commands. Dynos can be added depending on the scale of the project, and then removed in case traffic amounts reduce. Deploying an application simply requires one to create an account on the platform, choose a suitable payment plan and to link their GitHub repository with the code of the application. Linking a GitHub repository is not required, but by doing so, one can enjoy the benefits of distributed version control with a visual user interface. With these pre-requisites done, the next step is simply to choose the branch and then deploy it. Heroku also supports automatic deployment which can be a useful feature when a project is being developed over longer periods of time. (21)

5 Ideas for Improvement

An issue faced during this First Year Project was the lack of a high-quality open-source data set for maliciousness detection. Comparing studies and their results done within the field of maliciousness detection in text is problematic, as the data used is often proprietary. This of course being a product of the data usually stemming from private messaging collected by organisations who are unable to share it openly. Therefore, the data set used for this project is quite small, not containing many of the most common types of phishing attacks. The following examples display some of the resulting downfalls. Both strings evaluated in the following picture are so-called advance-fee scams, the trick is that the recipient must pay a transaction fee to receive the generous sum of money, which of course is then never delivered.



```
evaluate_text("Congratulations, you have won a 10000 dollars lottery. Please give your bank details to claim the money")
array([[0.99919134]], dtype=float32)
evaluate_text("Hello I am a Nigerian prince, please help me move gold out of my country")
array([[0.017643]], dtype=float32)
```

Figure 5 Two different messages of the same attack type evaluated with the model.

As seen in figure 5 above, the model does well at detecting the maliciousness of the first string which mimics a typical lottery scam, informing the recipient that they have won a large sum of money.

The second string is also a part of a typical malicious pre-text. In this scam, the malicious actor impersonates a Nigerian prince requesting monetary assistance in transferring large amounts of gold out of the country. In return, the recipient is promised a large monetary reward. This string is however not deemed malicious by the model.

The data set used for training is clearly lacking in its coverage even in some of the most common types of attacks. To improve this coverage, the application could be configured to save user input with their permission. This user input could then be classified similar to the original data set and be later used to train the model further.

The application in its current state is designed for desktop usage, and while it can be used on a mobile device, the user experience leaves much room for improvement. Creating a mobile version of the site seems like a low hanging fruit one could take advantage of next. Creating a mobile application also being a potential route. In this case, one could also enable sharing to the site or application, which would allow users to easily share for example text messages to it automatically, reducing the amount of actions needed by the user to receive feedback. This would potentially increase usage, which in turn would help combating cybercrime further while also receiving more data which could be used to improve the model.

Another aspect of the application which could be simply improved is the user interface. The interface could include more tips for the user, informing them e.g. on how to verify if something is truly malicious or not. This could be done by for example analysing the payload of the message, in the case of it being a link the user could be directed to other services specialised in automatically analysing websites, allowing the user to receive additional information about them. The user interface could also be made more welcoming by adding graphics and an about page containing more information about the tool and how their data is processed. These all should be considered before making the application available to the general public.

Payload analysis is also something which could be done automatically by integrating other solutions to the application. A possible solution to integrate could be for example VirusTotal, as they provide an API allowing one to programmatically interact with their solution (22). VirusTotal utilises many libraries of data in their solution which allows the user to upload a file and receive information about the maliciousness of the file, what it does when ran and for example what type of virus it might contain. This information can be fetched using the API and could then be shown in the application giving the user even better feedback regarding their input. This type of integration would introduce additional latency, but the additional functionality could for example be optional.

6 Conclusion

This Final Year Project examined different solutions to detecting maliciousness in text automatically, of which machine learning solutions were deemed as the most suitable option. The cybersecurity field would benefit greatly from open-source data sets allowing researchers to better advance the field using machine learning solutions. Another desirable future event would be that large organisations within the field with pre-existing data from e.g. spam filtering solutions would allocate more resources into developing solutions that allow users to train their cyber security awareness. These types of solutions are becoming increasingly sought after as services and communication are digitalised.

References

1 Hadnagy, Christopher. 2015. Phishing Dark Waters – The Offensive and Defensive Sides of Malicious Emails. John Wiley Sons Inc.

2 Kohnji, Mahmoud. Iraqi, Youssef. Jones, Andy. 2013. Phishing Detection: A Literature Survey.

3 Alpaydin, Ethem. 2020. Introduction to Machine Learning, Fourth Edition. MIT Press.

4 Hassabis, Demis. 2022. Alphafold Reveals the Structure of the Protein Universe. Available from: https://www.deepmind.com/blog/alphafold-reveals-the-structure-of-the-protein-universe (Accessed August 17 2022)

5 Arora, Surbhi. 2020. Supervised vs Unsupervised vs Reinforcement. Aitude. Available from: https://www.aitude.com/supervised-vs-unsupervised-vs-reinforcement/ (Accessed 21 August 2022).

6 ImageNet. 2021. Available from: https://www.image-net.org/about.php (Accessed 19 August 2022).

7 IBM Cloud Education. 2020. Unsupervised Learning. IBM. Available from: https://www.ibm.com/cloud/learn/unsupervised-learning (Accessed 21 August 2022).

8 Mwiti, Derrick. 2022. 10 Real-Life Applications of Reinforcement Learning. Neptune.ai. Available from: https://neptune.ai/blog/reinforcement-learning-applications (Accessed 21 August 2022).

9 Google. 2022. Data Preparation and Feature Engineering in ML. Online Course. Available from: https://developers.google.com/machine-learning/data-prep/ (Accessed 25 August 2022).

10 UCI Machine Learning. 2018. SMS Spam Collection Data Set. Available from: https://archive.ics.uci.edu/ml/datasets/SMS+Spam+Collection (Accessed 14 July 2022).

11 Devlin, Jacob, Chang, Ming-Wei. 2018. Open Sourcing BERT: State-of-the-Art Pre-training for Natural Language Processing. Available from: https://ai.googleblog.com/2018/11/open-sourcing-bert-state-of-art-pre.html (Accessed 25 August 2022)

12 Vaswani, Ashish, Shazeer, Noam, Parmar, Niki, Uszkoreit, Jakob, Jones, Llion, Gomez, Aidan N., Kaiser, Lukasz, Polosukhin, Illia. 2017. Attention Is All You Need. Available from: https://arxiv.org/abs/1706.03762 (Accessed 25 August 2022)

13 Burchfiel, Anni. 2022. What is NLP (Natural Language Processing) Tokenization? Available from: https://www.tokenex.com/blog/ab-what-is-nlp-natural-language-processing-tokenization#:~:text=Tokenization%20is%20used%20in%20natural,into%20understandable%20parts%20(words) (Accessed 25 August 2022).

14 Django Software Foundation. Available from: https://docs.djangoproject.com/en/4.1/ (Accessed 28 August 2022).

15 The Pallets Projects. 2010. Available from: https://flask.palletsprojects.com/en/2.2.x/# (Accessed 28 August 2022).

16 Conda User Guide. 2017. Available from: https://docs.conda.io/projects/conda/en/latest/index.html (Accessed 28 August 2022).

17 Hoxhunt. 2022. Behavioural Cybersecurity Statistics. E-Book. Hoxhunt.

18 Connolly, D., Masinter, L. The ‘text/html’ Media Type. 2000. Available from: https://www.rfc-editor.org/rfc/rfc2854 (Accessed 3 October 2022)

19 Salesforce. About Heroku. 2022. Available from: https://www.heroku.com/what (Accessed 3 October 2022)

20 Salesforce. Heroku Regions. 2022. Available from: https://devcenter.heroku.com/articles/regions (Accessed 3 October 2022)

21 Salesforce. Getting Started on Heroku with Python. 2022. Available from: https://devcenter.heroku.com/articles/getting-started-with-python?singlepage=true (Accessed 3 October 2022)

22 Virustotal. Public vs Premium API. 2022. Available from: https://developers.virustotal.com/reference/public-vs-premium-api (Accessed 9 October 2022)