MACHINE LEARNING NANO DEGREE

A Machine Learning Approach to Tweet Spam Detection

Samuel Abiodun James

May 31st, 2020

ABSTRACT

Machine learning is a branch of artificial intelligence concerned with the creation and study of systems that can learn from data. A machine learning system could be trained to distinguish between spam and non-spam (ham) tweets on the Twitter platform. I aim to employ a text categorization technique to classify content on tweets.

1 Definition

1.1 Project Overview

Twitter is a micro-blogging platform that allows users to broadcast real-time messages called tweets. Twitter has 330 million monthly active users tweeting 500 million tweets daily. This growth has caused many companies to create an official twitter account to engage with their customers and stakeholders.

Language is one is an essential element of customer engagement because it affords a great many opportunities for the intelligent reuse of linguistic content. Rather than always putting our thoughts into our own words, we often convey feelings through the words of others by citing, quoting, mimicking, borrowing, varying, or ironically echoing what others have already said. [7]

Social networking platforms such as Twitter elevate linguistic reuse into an integral norm of digital interaction. On Twitter, who you follow and what you retweet can say as much about you and your brand [7] . The task of engaging customers across the globe for a multinational business can be enormous for a group of individuals to manage alone. This complexity has seen many companies turned to automated ways of engaging customers through the use of bots.

One significant academic study estimated that up to 15% of Twitter users were automated bot accounts. The prevalence of Twitter bots, coupled with the ability of some bots to give seemingly human responses, has enabled these non-human accounts to garner widespread influence.

Most bots like @jsbot ,@SecurityNewsBot, etc produce their contents by listening to specific hashtags and re-broadcasting to a broader range of audiences that are interest-

ed in them. The problem with bots who source their content on the twitter platform is they can be tricked to broadcast tweets contrary to their subject matter.

The internet has come a long way; nevertheless, unwanted content called spam is still a huge problem today, and Twitter is not exempted. People often target accounts managed by bots in order to trick and get their message across even though the message could be damaging to the bot owner.

When people hear of spam, it is common to think of it as unsolicited emails one receives from an unknown person; interestingly, spam does not target inboxes only. Spams target everything these days, even social media feed. Analyzing the content of tweets, retweets, or mentions, is very important for companies who want to give control to bots to automate and engage customers on their behalf and still protect their brands.

1.2 Problem Statement

"Twitter is a wild place," Anonymous.

After creating a @hubofml bot, I agreed that Twitter is wild. Unlike other social media, everything on Twitter can be fully automated using bots.

A bot on twitter is an autonomous software system that generates and tweets messages of its design and composition. Unlike humans, twitter bots gain influence quickly due to its non-stop operations.

Famous twitters bot like @100daysOfcode, @jsbot, @securitynewsbot generates their contents by listening to specific hashtags in tweets. Such bots don't have an intelligent way of filtering out harmful content. Sometimes, they are tricked into broadcasting content contrary to their subject matter by appending hashtags they respond to.

Text categorization is the process of automatically assigning one or more predefined categories to a text document. In this paper, I used text categorization technique to clas-

sify content of tweets into two categories: spam and non-spam(ham), using a supervised learning approach. The spam determinant being whether a given tweet is related to machine learning or not related. To achieve this, I used two different network architectures RNN and bidirectional LSTM.

1.2 Metrics

The evaluation metric for this problem was the F1 score, which is a function of precision and recall. F1 score can be represented by this equation.

$$F_1 = \left(\frac{\mathrm{recall}^{-1} + \mathrm{precisior}^{-1}}{2} \right)^{-1} = 2 \cdot \frac{\mathrm{precisior} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}.$$

F1 score was deemed an appropriate choice because of the problem's binary nature: spam or not-spam. For this problem, the benchmark model are LSTM and the Support Vector Machine (SVM). The F1 score of the LSTM model was compared with the F1 score of an SVM model on Kaggle solving a similar problem.

2 Analysis

2.1 Data Exploration

Data exploration is one of the ways to accrue knowledge about data, and its nature. It includes methods to extract knowledge from data efficiently [8].

2.1 Twitter Dataset

The datasets for this task were taken from the Twitter platform, which contains 110k tweets. They were annotated using two labels: *yes (when a tweet's considered to be spam when it's unrelated to machine learning), no (tweet is considered a non-spam (ham) when it's related to machine learning).* I harvested over 100k of tweets datasets on twitter. Tweets considered to be ham were collected by listening to **machinelearning**, **computervision, robotics** hashtags on twitter. Normal tweets were retrieved based on randomly selected keywords from Wordnet as hashtags. The hashtags include politics, crime, religion etc.

2.2 Dataset Distribution

The raw dataset consists of three columns *Unnamed :0*, *tweet*, and *spam*. The "Unnamed" adds no useful information to the data. Hence it was dropped. 999,989 tweets were spam, and 10, 538 tweets were non-spam. Ham tweets account for approximately 10% of the datasets, as shown in figure 1.

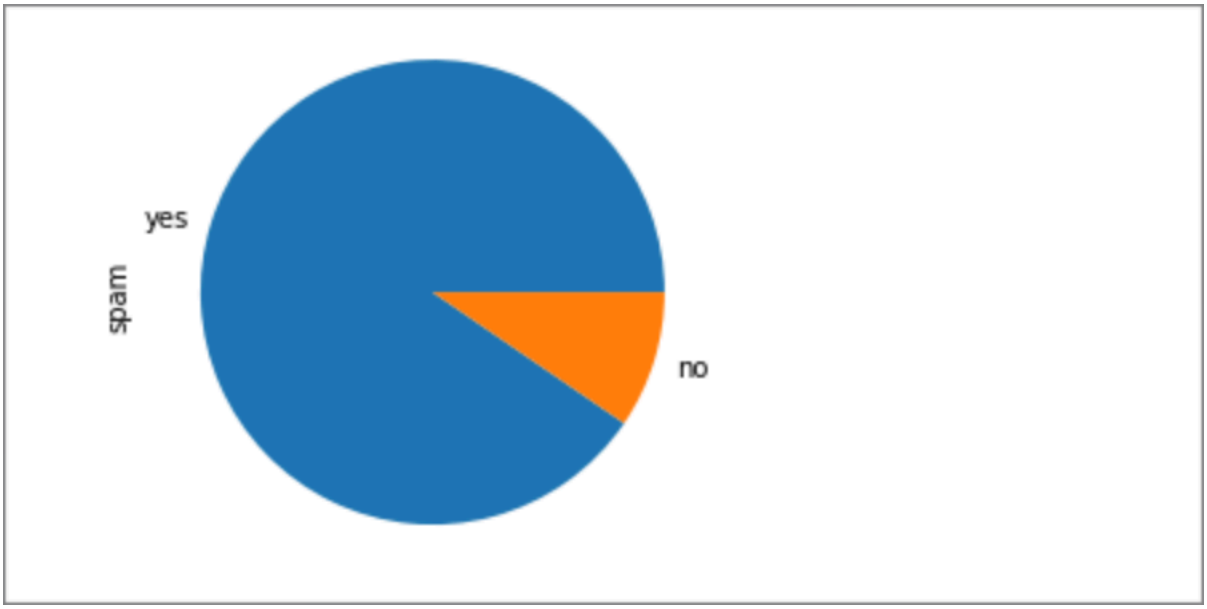


Figure 1

A look at the data distribution between the two labels shows that the dataset is unbalanced. Unbalanced datasets often have substantial effects on how machine learning models generalize. The algorithm could learn that spam tweets are more predominant, making it natural to leans toward the predominant class during generalization.

There are many ways to mitigate unbalanced datasets; in this task, I've chosen to downsample the majority class. Downsampling in this context refers to training on a disproportionately low subset of the majority class sample.

After downsampling, I arrived at an equal number of spam and ham tweets, 10,000 tweets in each class, accounting for a total of 20k tweets in the downsampled dataset.

Figure 2

A quick statistics on the dataset shows on the average, tweets have approximately 16 words and contains an approximate of 122 characters, as shown in Table 2 below:

Average number of words in tweets	16.56

Average number of characters in tweets	122.24

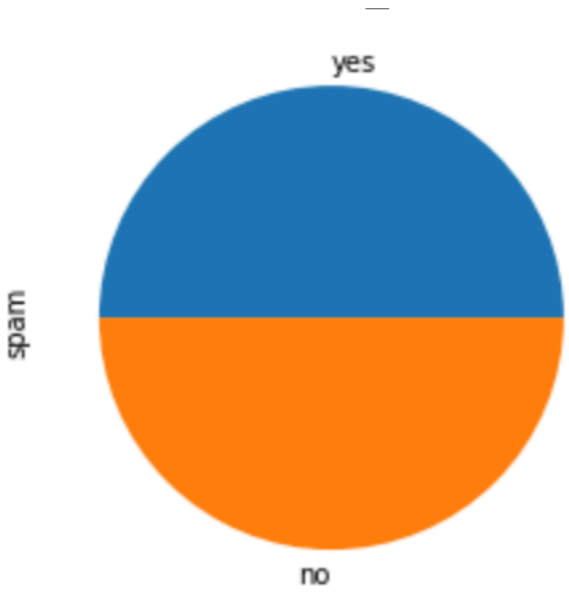


Figure 3 : Top-10 most commons words in non-spam (ham) tweets

As a part of data exploration, it's important to know what words are likely to be found in tweets that are considered spam and non-spam. Using a python library, I plotted the top-10 most common words in the two classes. For the non-spam tweets, *ai*, *machine-learning, artificialintellgience, data, data science, biodata, python, deep learning* were found to be most common, as shown in figure 3. On the other hand, *good, lol, like, thanks, day, know, and sorry* were found to be most common in spam tweets.

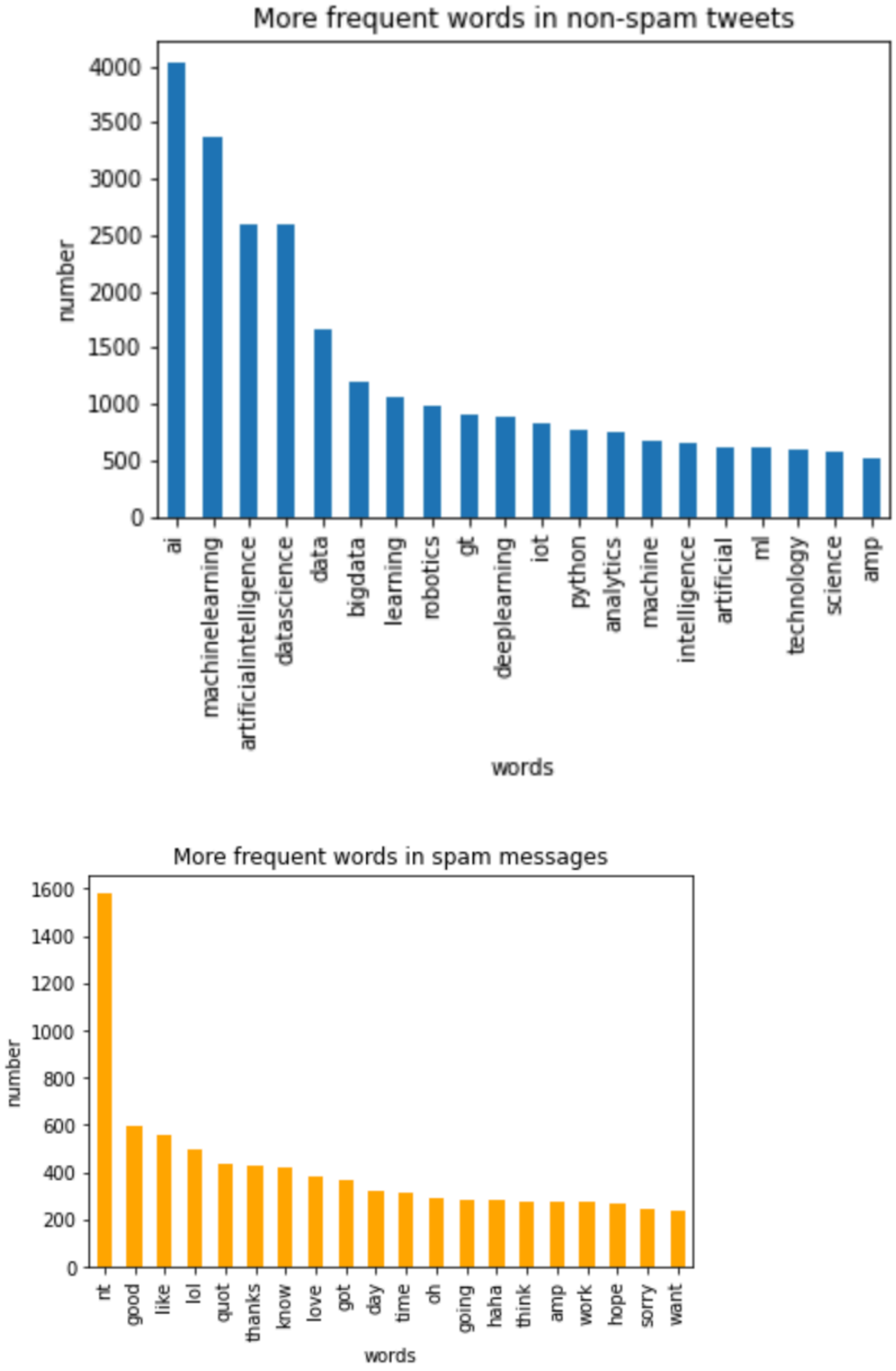


Figure 4 : Top-10 most commons words in spam tweets

2.2 Algorithms and Techniques

I began by using a simple RNN architecture before experimenting with an LSTM (a variant of RNN), which performed better. In the past few years, there have been many groundbreaking successes from applying RNNs to a variety of problems: speech recognition, language modeling, translation, image captioning, etc.

RNNs are great, but they suffer from short-term memory. If a tweet is too long, an RNN might have a hard time carrying information from earlier time steps to the current time step. Besides, RNNs suffer from vanishing gradient during backpropagation. Vanishing gradient occurs when the gradient of the loss function approaches zero making it difficult for the network to learn.

LSTMs was created to overcome the challenges that RNNs have. It was first introduced by Hochreiter & Schimdhuber (1997). LSTM avoids the long-term dependency problem by remembering information for long periods of time. They have internal features called gates that regulate the flow of information in and out of the cell state.

The algorithm used in training LSTMs is known as backpropagation through time BPTT. The algorithm works by modifying the weights of a neural network in order to minimize the error of the network outputs. Thus to update the weights, the gradient of loss function at a particular time step *t* depends on the input and the gradient of previous states at the previous time step [1]. The BPTT can be represented in equation as:

$$\frac{\partial E_3}{\partial W_s} = \text{gradient from} s_3 + \text{gradient from} s_2 + \text{gradient from} s_1$$

$$\frac{\partial E_3}{\partial W_s} = \frac{\partial E_3}{\partial y_3}\frac{\partial y_3}{\partial s_3}\frac{\partial s_3}{\partial W_s} + \frac{\partial E_3}{\partial y_3}\frac{\partial y_3}{\partial s_3}\frac{\partial s_3}{\partial s_2}\frac{\partial s_2}{\partial W_s} + \frac{\partial E_3}{\partial y_3}\frac{\partial y_3}{\partial s_3}\frac{\partial s_3}{\partial s_2}\frac{\partial s_2}{\partial s_1}\frac{\partial s_1}{\partial W_s}$$

For the reasons discussed earlier, both RNN and LSTMs are options for classification problems. However, LSTMs network performed better due to their long time memory.

2.2 Benchmark Model

For the benchmark model, I used the algorithms outlined in the paper “Detecting Spamming Reviews Using Long Short-term Memory Recurrent Neural Network Framework (Chih-Chien Wang, 20)[3]” as the benchmark model and Support Vector Machine model.

Model	**Recall**	**F1 Score**
Chen and Chen [4]	0.5714	0.6154
Chih-Chien[3]	0.85	796
Support Vector Machine Model	0.88	0.93

Table 2: Benchmark Models

3. Methodology

3.1 Data Preprocessing

Raw tweets are generally riddled with noises. This is, as a result, no formal way of representing tweets. Tweets usually have certain special characteristics, such as usernames, retweets, signified by "RT," links, emoticons, etc. It's vital to pre-process tweets before fitting a model. The techniques I used in precessing the tweets are stated as follows:

Capitalization

Tweets and text data points have a varying amount of capitalization. Diverse capitalization can be problematic in text classification. My approach to dealing with capitalization is by converting all tweets to lowercase.

URL Removals

Twitter users often share hyperlinks to other pages, images, and videos. URLs are not important for text classification as they don't usually add meanings to tweets. I used a regular expression to remove hyperlinks from raw tweets. The regular expression used to match URLs for removal is ((www\.[\S]+)|(https?://[\S]+)).

Usernames Removal

Every twitter user has a username called handle. It's common for users to tweet at other people through mention. A mention on twitter takes the format @username.I replaced all user mentions with a regular expression.

Emoticons Removal

Since emoticons have become a way of communicating digitally, twitter users make use of emoticons heavily. However, spam and ham tweets could have the same set of emoticons. Therefore emoticons are not indications that a tweet is spam or not, I removed them using a python library known as demoji.

Hashtags

Hashtags are un-spaced phrases prefixed by the hash symbol (#), which users frequently use to categorize the content. I replaced hashtags with the words, for example. *#world* is replaced by *world*.

Tokenization

This is a term used to describe a technique of braking stream of text into words phrases, symbols, or other meaningful elements called tokens. Text tokenization allows for an easy investigation of words in a tweet. In this task, I used a python library known as spacy for tweet tokenization. For example:

“ New machine learning technology helps in identifying blood clots “

The tokens for the tweet is as follows:

{"New", "machine", "learning", "technology", "helps", "in', "identifiying", "blood", "clots"}

Stemming

In English language, it’s very common for one word to appear in different forms (i.e., singular and plural noun form), while the semantic meaning of each form is the same. Stemming is an approach use in consolidating different forms of a word into the same feature space.

3.2 Vocabulary List

Text documents usually contain common words that are of little value to the context of the text. These words are known as stop words. Stop words are a set of commonly used words in any language. For this classification task, the removal of stop words is crucial as it allows us to focus on the important concepts. Prior to building a dictionary list of words, all the stop words in the English language were removed.

These words include “and,” “our”, “punctuations,” etc. The resulting dictionary contains a word as the key and the number of times they appear in the datasets as values. Words with less than two occurrences are discarded.

3.3 Tweet Encoding and Padding

Machine learning models take vectors as input. For a text problem, one of the pre-processing steps is to convert strings to numbers before feeding it into the model. This is known as text vectorizing. In my approach, I assigned a unique number to each word the vocabularies. Each tweet is encoded using the unique number assigned to the word. If a word could not be found in the dictionary, its automatically assigned a value of 1 - a value reserved for words that are not found in the vocabulary list. Given the following vocabulary list

```
{ “The”: 5, “cat”:1, “sat”:3, “on”:4, “mat”:2, “in”:6, “unk”:0}
```

A sentence like “The cat sat on the mat in the morning” will be encoded as [5,1,3,4,5,2,6,5,0].

I used pad sequence to convert variable length sequence to the same size. This is crucial for the network to take in a batch of variable-length sequences.

3.4 Splitting the Dataset

Overfitting is a major challenge in machine learning. Overfitting is when a model is so tuned to the training sets that it’s unable to make generalize on data outside of the training sets.

To prevent overfitting, I spliced the datasets into training and test sets. A training set is used in learning (fitting the model) and test sets for testing how the model generalizes on unseen data. 60% of the dataset was used for training, and 40% was used for testing.

3.5 Implementation

I began with simple recurrent neural network (RNN) architecture. An RNN takes a sequence of words *X ={ x1…, xt }* one at a time, and produces a hidden state *h*. The network consists of 3 layers, an embedding layer, a hidden layer, and an output layer. The first layer is the embedding layer, which transforms the input vectors into dense embedding vector. The second layer is the RNN, which takes in the dense vector and the previous hidden state to calculate the next hidden state. The final layer takes the final hidden states and feeds it through a fully connected layer, transforming it to the correct output dimension.

Metric	**Value**
F1 Score	0.66
Recall	1.0
Precision	0.49

Table 3: F1 Score, Recall and Precision of an RNN model

3.6 Refinement

In my first attempt, I obtained a precision of 0.49 and an F1 score of 0.66. This is below what is acceptable. I could improve accuracy by revisiting the network architecture. I replaced the hidden layer RNN by LSTM. LSTM is a better option for this problem because it maintains an internal memory state and gates to control the flow of information inside each LSTM unit.

I used a bidirectional LSTM network, which allows inputs to be processed from the first to the last and from the last to the first. This ensures that the network is able to preserve information from both past and future.

The first architecture lacks regularization techniques. Regularization is a way of adding information in order to prevent overfitting of a model to training sets. I updated the architecture to include dropout. Dropout works by randomly dropping out neurons in the hidden layer during a forward pass.

Finally, I used packed padded sequences to ensures that the network only processes the non-padded elements of our sequence.

The F1 score after these improvements is shown Table 4.

Metric	**Score**
F1 Score	0.97
Recall	0.98
Precision	0.96

Table 4: F1 Score, Recall and Precision of an LSTM model

4 Results

4.1 Model Evaluation and Validation

The final model architecture and hyperparameters were chosen because they performed better among several combinations.

To evaluate the model, I used the test sets. The following observation were recorded.

- The model performed well on unseen data.
- There were a few misclassified tweets

4.2 Justification

There are various methods to determine model effectiveness; however, precision, recall, and accuracy are the most often used. Precision represents the classifier's ability

to place a document as being under the correct category as opposed to all documents place in that category, both correct and incorrect [10].

Recall (ρi) is defined as the probability that, if a random document dx should be classified under category (ci), this decision is taken [10].

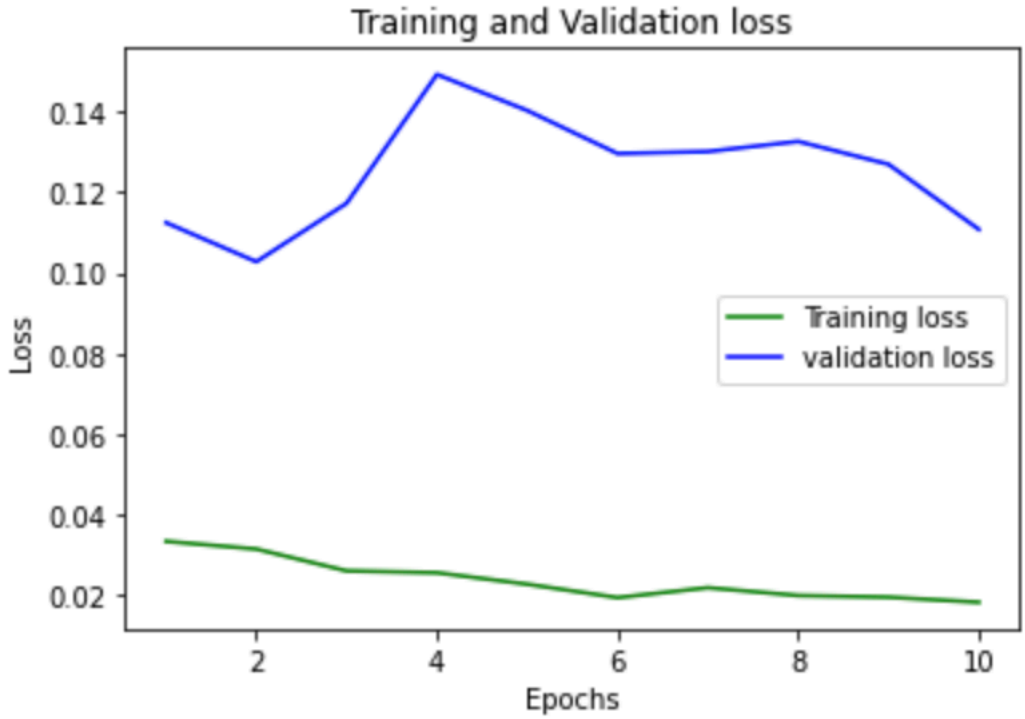
Accuracy is commonly used as a measure for categorization techniques. Accuracy values, however, are much less reluctant to variations in the number of correct decisions than precision and recall:

The performance of the model was evaluated using f1 score, accuracy, and precision. The final model achieved an f1 score of 0.97 which exceeded my expectations given the benchmark were 0.61 and 0.79

Model	**Recall**	**F1Score**
Chen and Chen [4]	0.57	0.62
Chih-Chien[3]	0.85	0.80
Support Vector Machine	0.88	0.93
Current Model	0.98	0.97

5 Conclusion

5.1 Free-Form Visualisation



5.1 Improvement

While the model achieved high accuracy, they're a few things that could make the model perform better.

- **Quality Datasets**

The tweet datasets used in training the model were in tens of thousands; therefore, human errors can not be entirely overruled. As a result, the ground truth labels on the datasets may include incorrect labels. This can be improved by crowdsourcing the labeling task using services like AWS Mechanical Turks.

- **Multimedia tweets**

The trained model is limited to text classification. In the real world, tweets do contain images or videos without text descriptions. Such tweets would be misclassified as the model is unable to predict multi-media tweets.

5.2 Reflection

This project can be summarised as a sequence of the following steps.

1. Datasets sourcing on Twitter
2. Datasets exploration and visualization
3. Data preprocessing and feature selection
4. Selecting a benchmark model
5. Creating a network architecture
6. Training and testing the model

The most challenging part of this project was downloading Twitter datasets. Twitter provides both free (Standards) and Premium APIs for downloading Twitter public data. The standard APIs are free but limited in scope. One of the issues I ran to while using the Standard APIs was rate-limiting. There is a limit imposed on the number of requests one can make per minute and per month, which makes the step the most time-consuming.

References

1. Backpropagation Through Time
2. Chih-Chien Wang., et al., Detecting Spamming Reviews Using Long Short-term Memory Recurrent Neural Network Framework
3. Algur, S., et al., 2010. Conceptual level similarity measure based review spam detection. 416-423.
4. Alex Sherstinsky, Fundamentals of RNN and LSTM Network
5. Apporv A., et al., Sentiment Analysis of Twitter Data
6. Kamran Kowsari, et al., Text Classification Algorithms: A Survey
7. Tony Veale., et al., Twitter: The Best of Bot Worlds for Automated Wit
8. Davide Mottin., et al., Exploring the Data Wilderness Through Examples
9. Sepp Hochreiter, et al., Long Short-Time Memory
10. Ikonomakis M., et al., Text Classification Using Machine Learning Techniques