Email Spam Detection Using Natural Language Processing

By: Steven Granaturov
EMPLID: 23756046
CSc 30100: Numerical Issues in Scientific Programming
Professor Erik Grimmelman

Introduction

Natural Language Processing (NLP) is one of many branches of computer science, and artificial intelligence that aims to build machines to understand and respond to text in a similar way that humans do. NLP combines the technologies of machine learning, deep learning models, and statistics to create a model based on a certain set of rules. These technologies, when combined together, allow computers to process human language by understanding its meaning and being able to respond.

Some NLP powered software that humans use in our daily lives include Siri - the virtual assistant on Apple products, Google search engine which fetches relevant search results based on the query, and a simple spell checker on a browser or an Integrated Development Environment (IDE). Moreover, there are many other examples of NLP such as an autocomplete feature, the text summarization used by Reddit, and autocorrect.

In this report, I will be focusing on spam email detection. Spam email detection is used by almost all major eMail services such as Google Mail, Microsoft Outlook, and Yahoo Mail. They are able to filter out emails that look and sound like spam to a spam folder so that they don’t overcrowd the general inbox. Although quite effective, sometimes they do mark legit emails as spam.

The Dataset

The dataset I opted to use was the Spam Message Classification from August 20th, 2017. It is a clean dataset with 5572 entries consisting of two columns, the classification, and the

message. Hence, dropping unnecessary columns was not needed. The classification column classifies the message as either ham (meaning legit), or spam. As seen in figure 1 below, there is a huge imbalance in the data, which I attempt to balance later on.

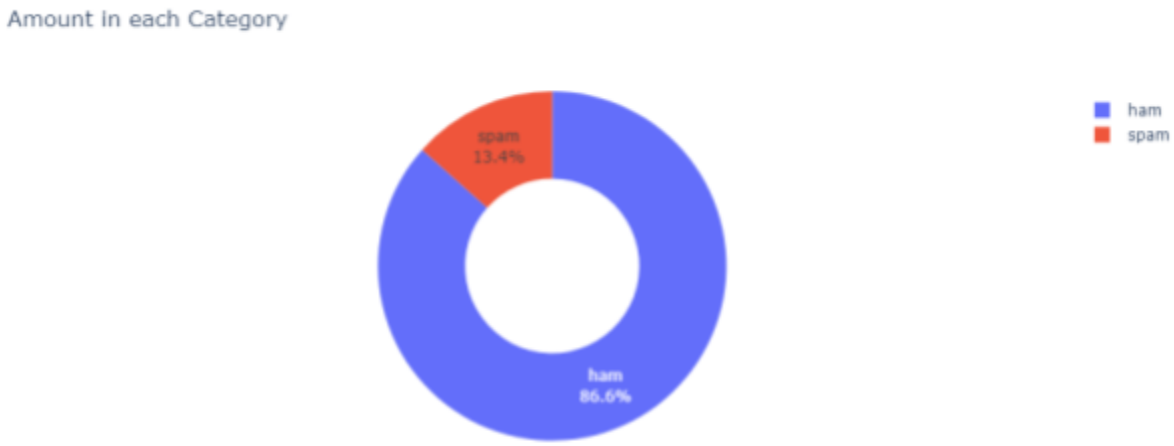


Figure 1. Amount of data in each category

Data Preprocessing

Before creating a model and inputting all of the texts, I have to preprocess and clean the data. I also remove any words and characters that don't add any value to the string, and remove redundancy.

- Removal of entries with missing values or null values:
    - In this dataset, there were no missing or null values.
- Removal of URLs:
    - URLs in a text are just references to a location on the world wide web. They do not provide any additional useful information hence they are removed.
- Removal of punctuation and special characters:

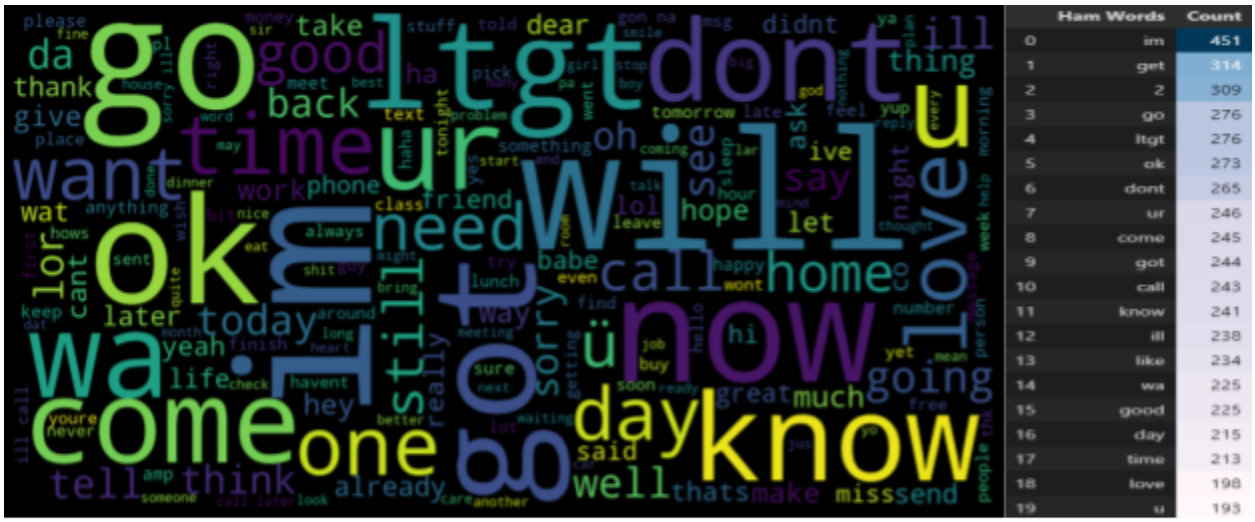
    - Punctuation and special characters don't add any value to the string either, so they are removed.
- Removal of abundant whitespaces:
    - Again, unnecessary, so it is removed.
- Removal of Stopwords:
    - Stopwords are a set of words that are commonly used such as ["I", "me", "both", "didn't", "the", etc…]. Using the NLTK library, I removed stopwords so that the model can focus on the more important words.
- Lemmatization of the Data:
    - Lemmatization "derives the canonical form of a word, or its root form." This converts variations of the same word into its root form such as "studyed" to "study." This was done with the WordNetLemmatizer from the NLTK library.
- Tokenization:
    - Separates the text into units, or a list of individual words.
- Vectorization:
    - Process of encoding a string into integers. It is much easier for machine learning algorithms to use the data.

Most Frequent Words

After preprocessing the data, I wanted to investigate what the most common words were for both categories. The word plots are seen below.

	Spam Words	Counts
0	call	356
1	free	216
2	2	173
3	txt	150
4	ur	144
5	text	137
6	mobile	135
7	4	119
8	stop	115
9	claim	113
10	reply	102
11	prize	92
12	get	83
13	tone	73
14	service	72
15	new	69
16	send	68
17	nokia	65
18	cash	62
19	urgent	62

Figure 2. Most common words in SPAM



	Ham Words	Count
0	im	451
1	get	314
2	2	309
3	go	276
4	ltgt	276
5	ok	273
6	dont	265
7	ur	246
8	come	245
9	got	244
10	call	243
11	know	241
12	ill	238
13	like	234
14	wa	225
15	good	225
16	day	215
17	time	213
18	love	198
19	u	193

Figure 3. Most common words in HAM

Upon quick inspection, the words that appear most common in spam are action verbs such as “call” or “text”, along with enticing words such as “won”, “prize” or “cash.” The most common words in the ham data are more neutral words.

Balancing the Data

Since the data is unbalanced with a 13% spam, 87% ham split, I attempt to balance it by using a data augmentation technique. Data augmentation artificially increases the amount of data by generating new data points using existing data. The technique I opted to use is called Synonym replacement. For example, if an entry like “I have no time” exists, I can generate a new entry with the same meaning, but of a different value such as “I don’t have time.”

Using the word_augmenter from the nlpaug (Natural Language Processing Augmentor) library, I ran through five iterations to augment the data. Upon completion, there are now 4825 entries for ham and 4277 entries for spam which totals to 9102 entries. As seen below in figure 4, the data is much more balanced with a split of 53% ham and 47% spam.

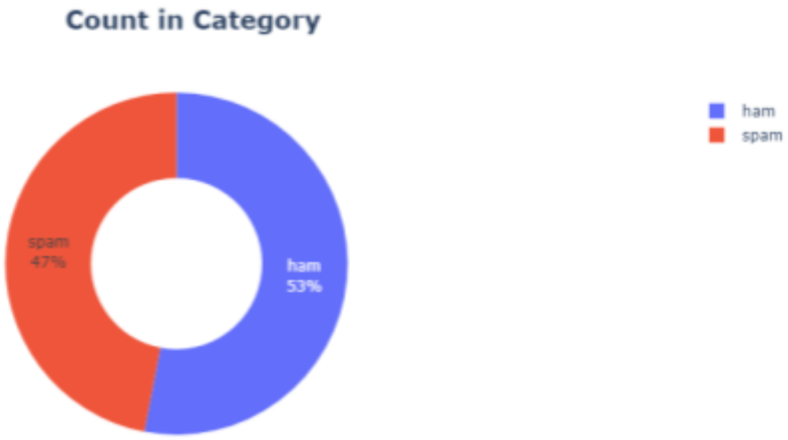


Figure 4. Percentage of each category after data augmentation

Creating the Model

I have opted to create a recurrent neural network (RNN) using tensorflow as I am deeply interested in studying tensorflow. Tensorflow is a relatively new tool that is extremely powerful in its capabilities. Furthermore, I used a 25% validation - 75% training split. Figure 5 below shows my configuration of this model.

```
model = Sequential()
model.add(Embedding(VOCAB_SIZE, EMBEDDING_VECTOR_SIZE, input_length = MAX_LEN, name='Embedding'))
model.add(Bidirectional(LSTM(40, return_sequences=True)))
model.add(Bidirectional(GRU(20)))
model.add(Flatten())
model.add(Dropout(0.2))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

Figure 5. Configuration of the RNN model

- Embedding takes in an integer matrix as an input with sizes defined along with the largest integer in the input.
- Bidirectional LSTM (Long-short term memory) makes the neural network have a sequence of information in both directions.
- Ex: “Students go to …”. I am unable to fill in the blank. But in a future sentence I can have a sentence like, “Students came from the train.” Using Bidirectional LTSM, the neural network is able to predict the past blank space.
- The GRU (Gated Recurrent Unit) reduces the number of parameters from the LSTM. With a slight dip in the accuracy, but less number of trainable parameters, it seems advantageous to use
- Dropout is where randomly selected neurons are ignored during training. The effect of this is that the network becomes less sensitive to specific weights of certain neurons.

Results

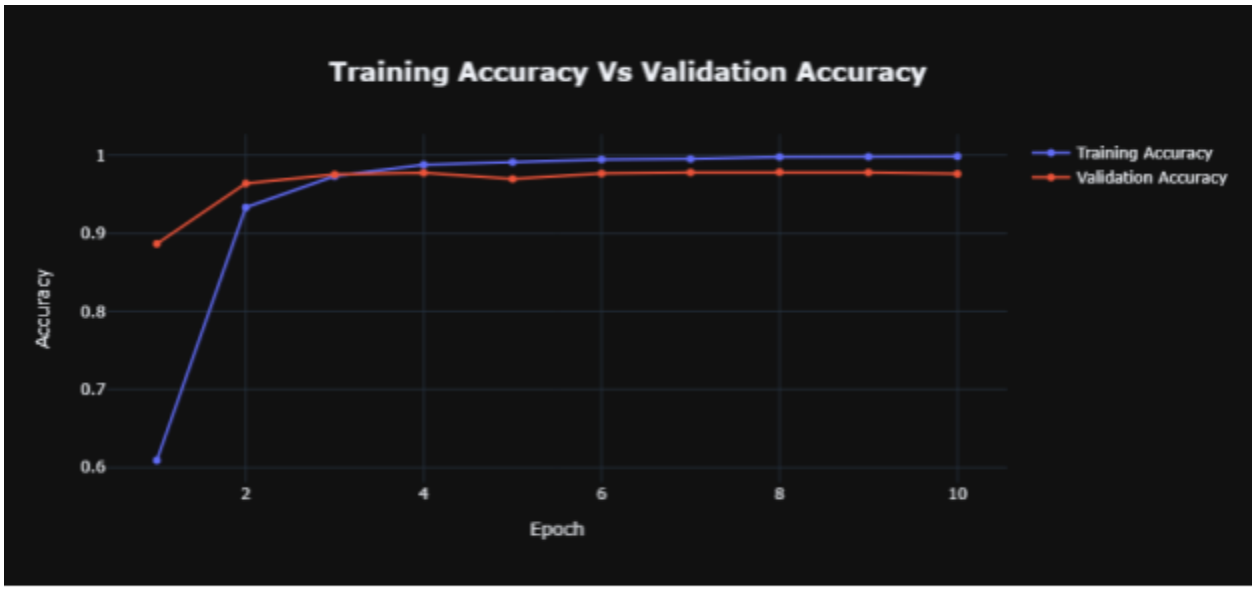


Figure 6. Accuracies over epochs

Figure 7. Loss function over epochs

Based on figure 6 and 7, this model is a pretty good classifier of whether a message is ham or spam. The validation accuracy tops out at 97.3% which shows this neural network model is pretty reliable. Furthermore, this model is neither underfitting nor overfitting as the training

error is not high and the training loss isn't much lower compared to the validation loss. This model seems "just right."

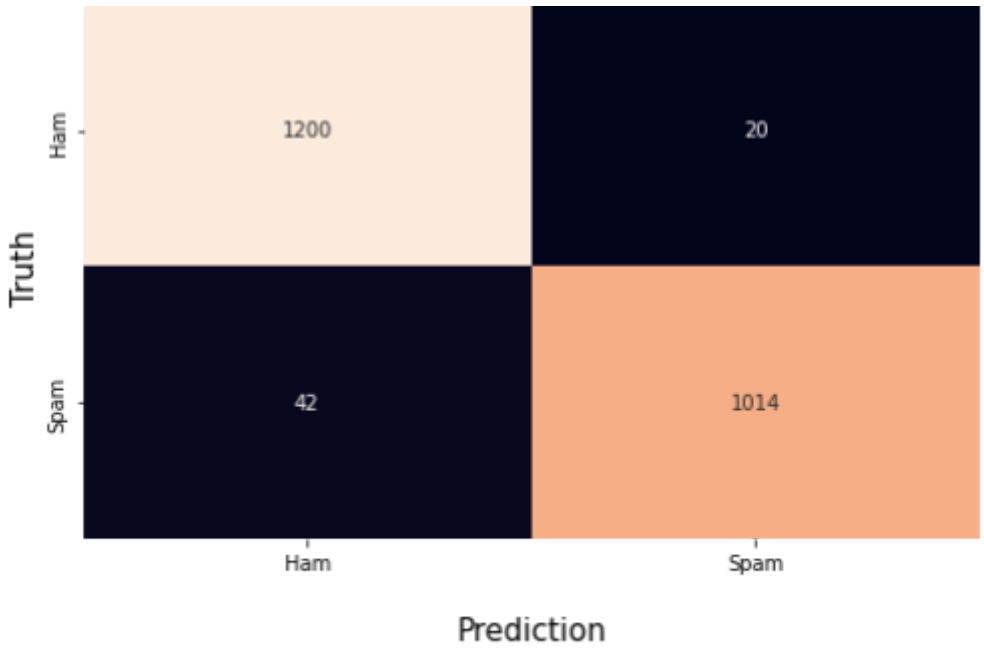


Figure 8. Confusion matrix

Looking at the confusion matrix in figure 8, the model made very few inaccuracies. It misclassified only 20 ham entries as spam, and misclassified 42 spam entries as ham. The accuracy on unseen data, or validation data, is 97.63% and the loss on this unseen data is 10.06%.

Predictions

Finally, I generated some random text to see if the model was able to classify them correctly. The results are shown below.

```
1  text = "Get free car insurance and win a prize just call this number 0123456789"
2
3  category = predict_category(text)
4
5  print(f"Text is: {text}\n")
6  print(f"Category: {category}")
✓ 0.1s

1/1 [==============================] - 0s 38ms/step
Text is: Get free car insurance and win a prize just call this number 0123456789

Category: SPAM
```

Figure 9. Prediction 1

```
1  text = "WINNER!! As a valued customer, you have been selected to receive a $900 prize reward! Claim now before your offer expires!"
2
3  category = predict_category(text)
4
5  print(f"Text is: {text}\n")
6  print(f"Category: {category}")
✓ 0.1s

1/1 [==============================] - 0s 42ms/step
Text is: WINNER!! As a valued customer, you have been selected to receive a $900 prize reward! Claim now before your offer expires!

Category: SPAM
```

Figure 10. Prediction 2

```
1  text = "Hey! Nice seeing you this weekend. When will you be back in NYC"
2
3  category = predict_category(text)
4
5  print(f"Text is: {text}\n")
6  print(f"Category: {category}")
✓ 0.1s

1/1 [==============================] - 0s 39ms/step
Text is: Hey! Nice seeing you this weekend. When will you be back in NYC

Category: HAM
```

```
1  text = "That cafe was so delicious! Make sure to send me $200 cash or venmo me for your part of the bill"
2
3  category = predict_category(text)
4
5  print(f"Text is: {text}\n")
6  print(f"Category: {category}")
✓ 0.1s

1/1 [==============================] - 0s 41ms/step
Text is: That cafe was so delicious! Make sure to send me $200 cash or venmo me for your part of the bill

Category: SPAM
```

Figure 11. Predictions 3 & 4

Interestingly, I attempted to trick the model with the last prediction and I succeeded in doing so. I used words found most commonly in spam messages to create a ham message. By

adding more data entries similar to this prediction text, I would probably be able to help mitigate such misclassifications.

Code Developed

For simplicity, the code will be provided in a separate file.

Works Cited

Brownlee, J. (2022, August 5). *Dropout regularization in deep learning models with Keras*. MachineLearningMastery.com. Retrieved December 3, 2022, from https://machinelearningmastery.com/dropout-regularization-deep-learning-models-keras/

Dobilas, Saul. “LSTM Recurrent Neural Networks - How to Teach a Network to Remember the Past.” *Medium*, Towards Data Science, 5 Mar. 2022, https://towardsdatascience.com/lstm-recurrent-neural-networks-how-to-teach-a-network-to-remember-the-past-55e54c2ff22e.

ES, S. (2022, November 14). *Data Augmentation in NLP: Best Practices from a kaggle master*. neptune.ai. Retrieved December 3, 2022, from https://neptune.ai/blog/data-augmentation-nlp

Shahrayar, M. (2022, May 25). *Ham spam - text classification using LSTM*. Kaggle. Retrieved December 3, 2022, from https://www.kaggle.com/code/muhammadshahrayar/ham-spam-text-classification-using-lstm