Spam Classification Neural Network

**Abstract**

In this project I take a set of text messages and write a neural network to predict if a message is spam or not. My goal was to understand the various steps one should take when dealing with an NLP problem as well as try to explain it step by step in this report.

**Background Information**

My interests in computer science have always been centered around the intersectionality of human and computer language. I think the applications for effective natural language processing algorithms are near endless. For that reason I decided to center my own project around natural language processing, specifically spam classification. Spam messaging is very common with 85% of all online messages being spam, most of these spam messages being advertisements. However, spam messages are not just a nuisance but often a very real threat to someone’s finances and livelihood. Email spam costs businesses more than $20 billion a year and spam messages are one of the most common ways someone can get their identity stolen (Dataprot). Based on these reasons, an effective and accurate spam classification program is not only convenient but often vital to protect people, especially ones lacking in tech literacy.

In this course we have learnt how to take a set of features and labels, use them to train a neural network, and use this network to predict new labels based on new features. However our features and labels tend to be numerical, making it easy for the program to run its algorithms. Spam classification and natural language processing as a whole present a unique challenge of converting a text into unique numerical features.

The most common method used to prepare a set of text data is called “Tokenization.” Tokenization refers to breaking up a text into small pieces, whether it be breaking up the text by word, by syllable, or even by letter. There are pros and cons to each type of tokenization, however for the scope of this program I will be using word tokenization (If you wish to learn more about the other types of tokenization click [here](https://www.analyticsvidhya.com/blog/2020/05/what-is-tokenization-nlp/)).

Word tokenization is the easiest and most common form of tokenization and with a good dataset it can create very accurate machine learning models. However, it has two major drawbacks: how to handle the amount of unique words present in the dataset and words that were not in its training data. The first concern is not a huge issue considering this dataset is only a little more than 5000 messages, however it is something to keep in mind if using more data. The second concern is much more serious as words that were not in the training data would be incomprehensible to the prediction model. To address this, the tokenizer will list out the words in the training data by frequency and replace the rarest words with “Unknown” tokens. Then words that are out of the vocabulary of the training data will simply be matched to these “Unknown” tokens. This process essentially uses the rarest words in the training data to prepare the network on how to handle new words. This method is not ideal as it essentially treats every word that is outside the vocabulary of the network as the same. Regardless, it is an effective stopgap as long as the training data includes all the important words.

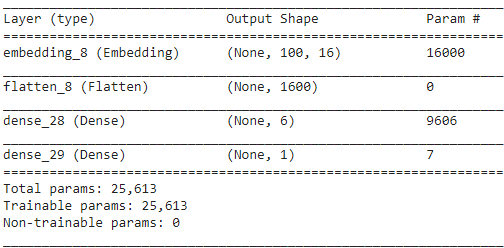
After tokenizing the data, the next step is to pad the data with empty tokens to ensure that all features are the same length. The last step is to take the tokenized and padded data and convert it into a numerical representation for use in the neural network. To do this each word in the data is assigned to a unique number and then each string of data is converted into a series of numbers. In this program this step is completed by using an “Embedding” layer in the neural network. Finally, the data is ready to be used in the neural network.

Preparing the Data

The data was found on Kaggle.com [here](https://www.kaggle.com/uciml/sms-spam-collection-dataset). It’s is rather simple, with two columns. The features column has each text message. The labels column has the classification of the text messages as either ‘ham’ or ‘spam’. The dataset has 4825 non-spam messages and 747 spam messages. This means that about 13% of the data is spam. The first step was to convert a classification of ‘ham’ to 0 and a classification of ‘spam’ to 1 then put the list of generated numbers into a new column on the dataframe. Then the features and labels columns were converted into separate lists and broken up into training, validation, and test datasets. The training set was 70% of the original data, the validation set was 20%, and the test set was 10%. The next step is preparing the Tokenizer. For this program I will be using the tensorflow tokenizer library. There are 6 variables we have to initialize. The most important variable is *num\_words* and it simply represents the maximum number of unique words the tokenizer will store. Extra words will be assigned out of vocabulary tokens. Then the dataset is tokenized and padded using the tokenizer functions.

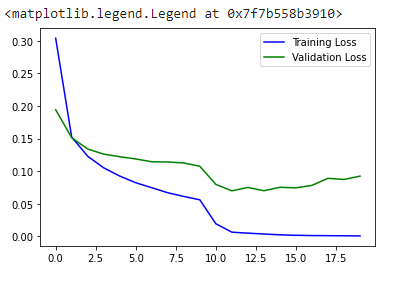
**The Neural Network**

The most basic architecture I used is shown below:



The embedding layer is used specifically for text data and takes the tokenized and padded data and converts it into numerical representations for use in the neural network. The next layer is the flattening layer which takes the data and reshapes it to fit the number of elements in the training data. Finally the two activation layers are a ‘relu’ function and a ‘sigmoid’ function respectively.

This network was run for 20 epochs and came up with the following graph for training and validation loss:



**Results**

Finally I tokenized and padded the test dataset and used the neural network to predict on a scale of 0-1 how likely the message was spam. Any prediction of less than 0.4 was classified as not spam, any prediction above 0.6 was classified as spam, and any prediction between 0.4-0.6 was classified as unsure. The model ended up making 551 correct predictions, 5 incorrect predictions, and 2 unsure predictions. This gave the model an accuracy of about 98.7%.

**Different Architectures**

To compare architectures I simply switched out the relu activation layer with each of the default keras layers. The model along with its accuracy, precision, recall, and f1 are listed below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Activation Layer** | **Accuracy** | **Precision** | **Recall** | **F1** |
| Relu | 0.9875 | 0.9851 | 0.9217 | 0.9496 |
| Softmax | 0.991 | 1 | 0.9306 | 0.964 |
| Softplus | 0.9875 | 0.971 | 0.9306 | 0.9504 |
| Softsign | 0.9857 | 0.9577 | 0.9444 | 0.951 |
| Tanh | 0.9875 | 0.9577 | 0.9444 | 0.951 |
| Selu | 0.9892 | 0.9853 | 0.9306 | 0.9571 |
| Elu | 0.9857 | 0.9706 | 0.9167 | 0.9429 |
| Exponential | 0.9874 | 0.9577 | 0.9444 | 0.951 |

**Data Analysis**

Each of the models did very well. All of them have accuracies above 98% and F1 scores above 94%. However, looking at the list it is clear that one model is much better. Overall, the model with the softmax layer had the best scores with the highest accuracy, f1 score, and precision score. The recall of this model scored the second highest score of any model so even it’s recall is quite solid. Furthermore, in a real world application a false positive would be worse than a false negative since the cost of a false positive is possibly having an important email sent to the spam folder while the consequence of a false negative is simply some more inbox clutter. Therefore in a spam classification model a high precision score should be weighted heavier than a high recall score.

**Conclusion**

In conclusion, I managed to create an effective and accurate spam classification network as well as compare several models to find which one was the best. I decided the best model was the one with a softmax activation layer.

There are several extensions of this project including different algorithms as well as different classification issues. For example, a similar network could be used for sentiment analysis programs that try to predict if a review is positive or not. I also think in the future it would be interesting to explore this same dataset with another NLP algorithm like Naive Bayes.

Sources:

Cvetićanin, N. (2021, February 22). *What’s On the Other Side of Your Inbox - 20 SPAM Statistics for 2021*. DataProt. https://dataprot.net/statistics/spam-statistics/#:%7E:text=Key%20spam%20statistics&text=Nearly%2085%25%20of%20all%20emails%20are%20spam.&text=Scams%20and%20fraud%20comprise%20only,businesses%20%2420.5%20billion%20every%20year.

Pai, A. (2020, July 12). *What is Tokenization in NLP? Here’s All You Need To Know*. Analytics Vidhya.<https://www.analyticsvidhya.com/blog/2020/05/what-is-tokenization-nlp/>

*SMS Spam Collection Dataset*. (2016, December 2). Kaggle. https://www.kaggle.com/uciml/sms-spam-collection-dataset