



Spam Message Detection Using Natural Language Processing (BERT)

Done by:

Ediz Emektas

Nataly Michail

Mohamad Serhan

Aida Zadinova

Introduction

With the constant growth of internet usage and our dependence on SMS as a form of communication, we have seen an increase in the amount of spam messages being sent to users and how annoying and time-consuming they can be to filter through. Our project aims to classify these messages as either spam (unwarranted promotional messages, links to scam websites, etc.) or ham (generally interesting or important information that the user signed up for). In order to achieve this, we decided to implement the BERT language model on a dataset consisting of labelled spam and ham messages.

Project description

There are two parts in our project, creating and evaluating the NLP model using BERT as well as a Streamlit web-app that allows the user to enter their own string of text and have the model predict whether it classifies as ham or spam, while also giving a number that represents the probability of this input being spam.

PLEASE CHECK THE END OF THE DOCUMENT FOR HOW TO RUN THE APP

BERT

BERT, or Bidirectional Encoder Representation from Transformers was created by researchers at Google AI Language.

The language model BERT uses is as follows:

$$P(w_1, w_2, \dots, w_T) = \prod_{t=1}^T P(w_t | w_1, w_2, \dots, w_{t-1})$$

Where $P(w_t)$ will be a probability of the word we want to predict (ham, spam) which we can get by looking at the other words in the email (w_1, w_2 , etc.).

We picked this model since it is relatively new and game-changing. It is the same model used by Google to improve their search results accuracy.

Contents:

- 1- Data Preparation
- 2- Model creation
- 3- Model evaluation
- 4- Prediction (web-app)
- 5- Bottlenecks

Data preparation

We are using a dataset from the SMS Spam Collection Dataset page on Kaggle. After renaming the columns and dropping empty ones, this is a sample of what our dataset looks like:

Category	Message
ham	Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there g
ham	Ok lar... Joking wif u oni...
spam	Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to r
ham	U dun say so early hor... U c already then say...
ham	Nah I don't think he goes to usf, he lives around here though
spam	FreeMsg Hey there darling it's been 3 week's now and no word back! I'd like some fun yo
ham	Even my brother is not like to speak with me. They treat me like aids patient.
ham	As per your request 'Melle Melle (Oru Minnaminunginte Nurungu Vettam)' has been set as
spam	WINNER!! As a valued network customer you have been selected to receive a £900 prize
spam	Had your mobile 11 months or more? U R entitled to Update to the latest colour mobiles v

When we analyzed the number of spam vs. ham messages in the dataset, we noticed that there were 4825 ham messages compared to 747 spam, making our data unbalanced and biased towards predicting 'ham'.

To fix this, we split the data into two separate ham and spam dataframes and then only took 747 random ham samples. We then re-concatenated the two dataframes into 1 balanced dataframe with 747 ham messages and 747 spam ones.

Next, we created a 'spam' column that just encodes ham and spam as 0 and 1 so that the model would understand the inputs.

Model creation

Using the scikit-learn library, we split the data with `train_test_split` with the 'Message' column as X and the 'Spam' column as our target.

We download the BERT preprocessor and encoder using the Tensorflow hub by providing it with links for both.

The preprocessor is used to transform the user input into a model input. The text is taken from the input, divided into subwords and then the sentences are combined and tags are extracted to better understand the data.

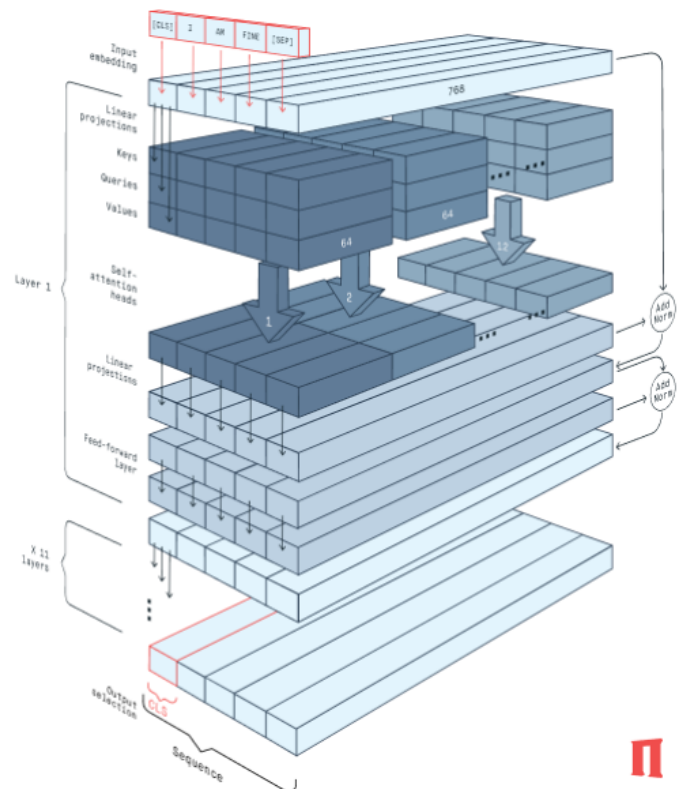
The encoder (shown in fig.) then takes the input and vectorizes it and changes the semantics of the words to give them more context within a message.

Three other variables are created,

`text_input`: a Keras tensor that we will use to create a Keras model from our input
`dropout`: a layer that, during the training phase, will drop some of the inputs to help prevent overfitting. Ours is set to 0.1, so 10% of the data will be dropped
`outputs`: that uses the Dense function with sigmoid activation since the outputs are binary

Finally, we created the model using the Keras Model function with `text_input` and `outputs` as parameters and then compiled our model using the adam optimizer and `binary_crossentropy` as our loss function.

We then fit our model using 1 epoch and then stored the model using `pickle.dump()` so that it would take less time to train.



Model evaluation

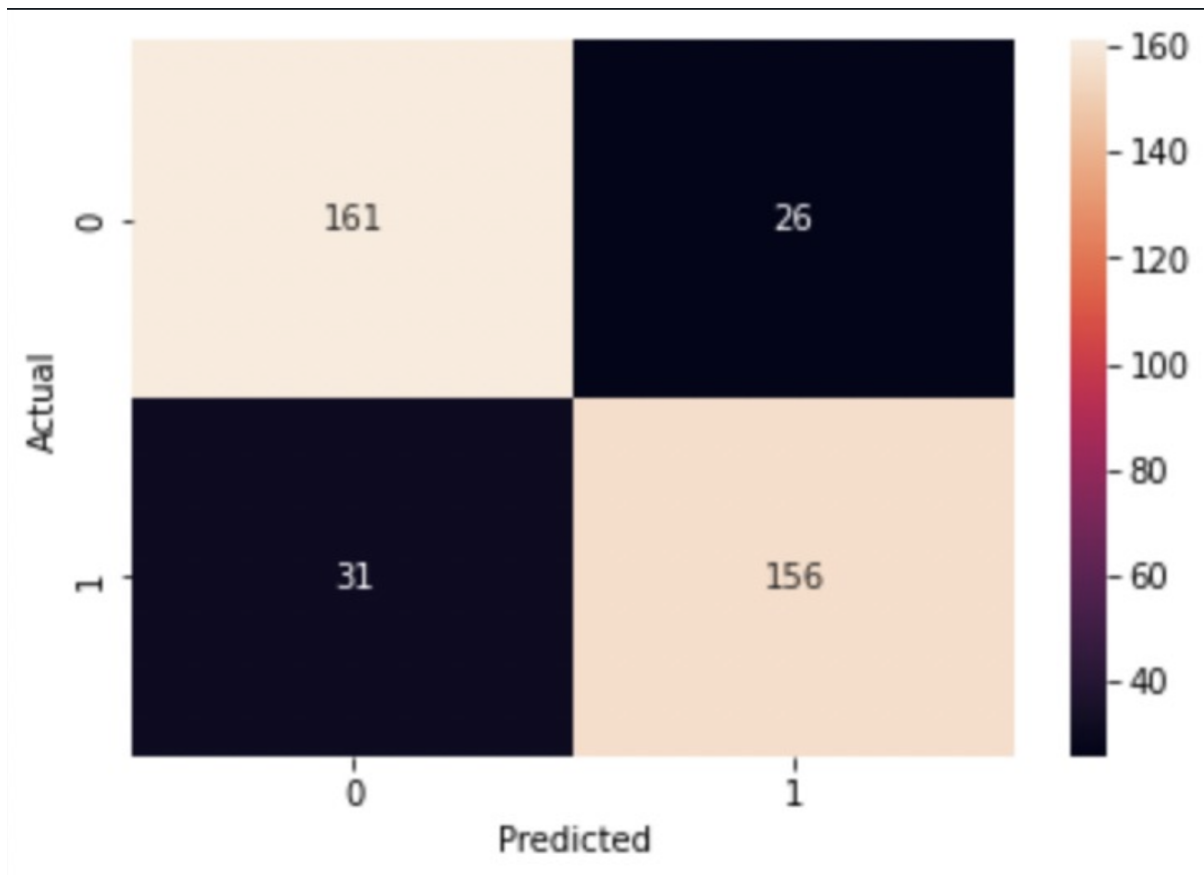
For the model evaluation, we get this confusion matrix which tells us that

Number of true positives (spam) = 156

Number of true negatives = 161

Number of false positives = 31

Number of false negatives = 26



Which gives us:

Accuracy 0.8455

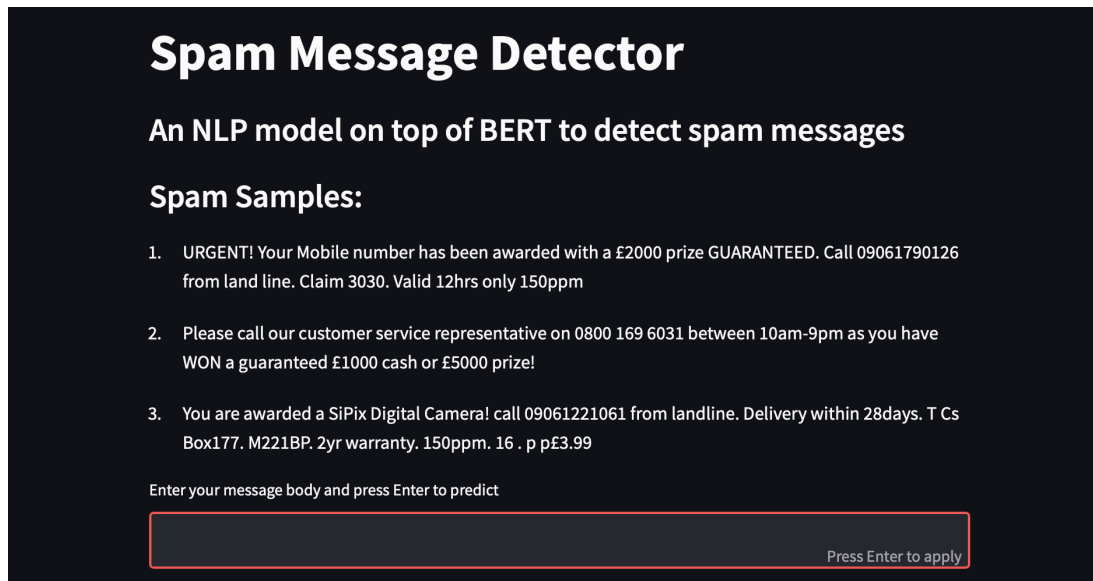
Sensitivity 0.8342

Specificity 0.8571

Precision 0.8571

Streamlit

For the web-app, we used the Streamlit framework. This is how the app looks when you first open it



Spam Message Detector

An NLP model on top of BERT to detect spam messages

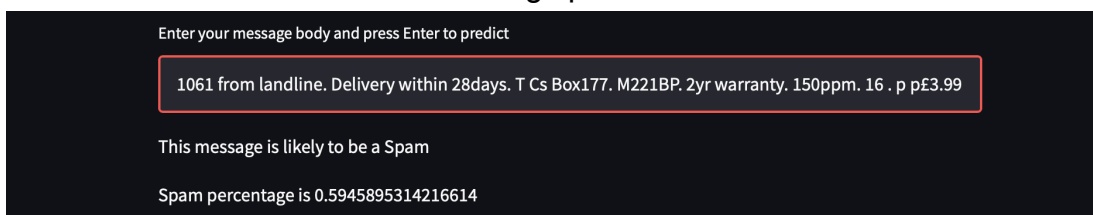
Spam Samples:

1. URGENT! Your Mobile number has been awarded with a £2000 prize GUARANTEED. Call 09061790126 from land line. Claim 3030. Valid 12hrs only 150ppm
2. Please call our customer service representative on 0800 169 6031 between 10am-9pm as you have WON a guaranteed £1000 cash or £5000 prize!
3. You are awarded a SiPix Digital Camera! call 09061221061 from landline. Delivery within 28days. T Cs Box177. M221BP. 2yr warranty. 150ppm. 16 . p p£3.99

Enter your message body and press Enter to predict

Press Enter to apply

When typing in one of the sample spam messages (or something similar), it gives out the result and the likelihood of it being spam



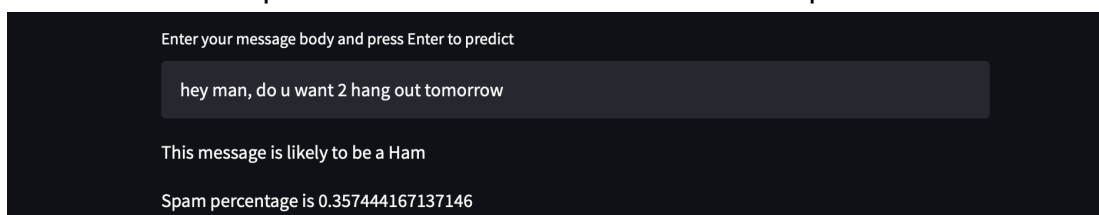
Enter your message body and press Enter to predict

1061 from landline. Delivery within 28days. T Cs Box177. M221BP. 2yr warranty. 150ppm. 16 . p p£3.99

This message is likely to be a Spam

Spam percentage is 0.5945895314216614

Here is another example when the text is classified as non-spam



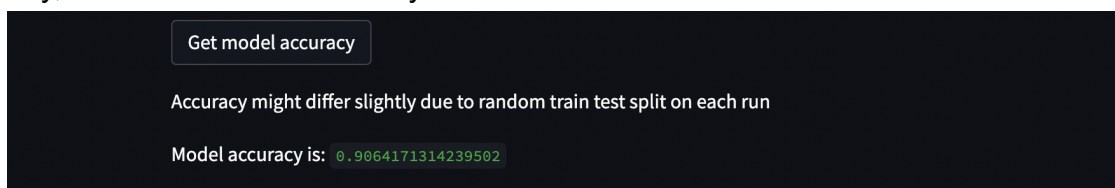
Enter your message body and press Enter to predict

hey man, do u want 2 hang out tomorrow

This message is likely to be a Ham

Spam percentage is 0.357444167137146

Finally, we can see the accuracy of the model



Get model accuracy

Accuracy might differ slightly due to random train test split on each run

Model accuracy is: 0.9064171314239502

Bottlenecks

- Since we developed the app mainly on MacOS, we encountered some issues while trying to adapt on other platforms. Mostly because of the filesystems and dependency differences.
- It was not so easy to do code versioning while working on the code simultaneously as multiple people. So, we focused on the other parts of the project which are less dependent on each other and then gathered up altogether in the end.
- The Tensorflow package on MacOS M1 version is not stable yet. We needed to create a specific environment through Conda's MiniForge version. And, the name of the packages were like tensorflow-macos and tensorflow-metal instead of the default one which is simply tensorflow.
- The model caching part on Streamlit was another issue we faced. At the beginning of the project, we had to wait minutes to train the model each time again just to be able to get a single prediction. So, what we have done is simply stored the model through joblib and saved it to the folder path. Afterwards, we did not have to wait minutes to see the result whenever we run the code each single time.

How to Run the App

- **Install the requirement files by the type of the machine such as <pip install -r requirements-**TYPE**.txt>**
- **<streamlit run app.py> from the command line**
- **Go to <http://0.0.0.0:8501>**
- **Do the steps in the screenshots above**
- **It might take some minutes to run if model file is not created at first**
- **Model file next to the project can be moved to folder path to run quickly without training the model first**