AI-POWERED SPAM CLASSIFIER

TEAM MEMBER

M.RIANEE RAYEN -950921104030

Project Title: AI-POWERED SPAM CLASSIFIER

Phase 2 Submission Document

INTRODUCTION:

★Spam classification is a challenging task, as spammers are constantly evolving their techniques. AI-powered spam classifiers can be used to accurately distinguish between spam and non-spam messages in emails or text messages.

★using pre-trained language models like BERT for feature extraction is a very innovative technique. BERT has been shown to achieve state-of-the-art results on a wide range of NLP tasks, including text classification, question answering, and sentiment analysis. This is because BERT is trained on a massive corpus of text data, which allows it to learn complex language patterns and features.

★Explore is using pre-trained language models like BERT for feature extraction. These models have demonstrated superior performance in NLP tasks.

Content for Project Phase 2 :

Consider exploring is using pre-trained language models like BERT for feature extraction. These models have demonstrated superior performance in NLP tasks.

1.Data Source:

A good data source for house price prediction using machine learning should be Accurate, Complete, Covering the geographic area of interest, Accessible.

Dataset Link: https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset

V1	V2	Unnamed: 2	Unnamed: 3	Unnamed: 4	
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN

V1	V2	Unnamed: 2	Unnamed: 3	Unnamed: 4	
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	NaN
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	NaN	NaN	NaN
3	ham	U dun say so early hor... U c already then say...	NaN	NaN	NaN
4	ham	Nah I don't think he goes to usf, he lives aro...	NaN	NaN	NaN

2.Data Collection and Preprocessing:

Import Dataset: Obtain a comprehensive dataset containing relevant features

Data Preprocessing:This module collects a dataset of labeled spam and non-spam messages. The data is then cleaned and pre processed to remove irrelevant characters and normalize the text.Using Tokenization,Stop Word Removal, Lower Casing.

3.Feature Engineering:

Feature engineering is the process of creating new features from existing features in order to improve the performance of a machine learning model. In the context of AI spam classifiers, feature engineering can be used to create features that are more informative and discriminative for the task of spam classification.

Some common feature engineering techniques for AI spam classifiers include:

- **Frequency of words and phrases:** This is a simple but effective feature that can be used to capture the presence of certain words or phrases in spam messages. For example, spam messages often contain certain keywords or phrases, such as "free", "money".
- **Presence of certain characters or symbols:** Spam messages often contain certain characters or symbols, such as exclamation points, dollar signs, or all-caps text.
- **Length of the message:** This is because spammers often try to send out as many spam messages as possible, and they don't want to spend too much time writing each message.

- **Structure of the message:** Spam messages often have a different structure than non-spam messages. For example, spam messages may have a lot of HTML formatting or they may contain links to malicious websites.

4.Pre-Trained language models:

BERT:

BERT can be used for NLP in AI spam classifiers in a number of ways. One common approach is to use BERT to extract features from text messages.To extract features from text messages using BERT, we first convert the messages into BERT encodings. This is done by passing the messages to the BERT model, which will output a vector of hidden states for each word. These hidden states capture the semantic and syntactic information of the word, as well as its context in the sentence.

Once we have converted the messages into BERT encodings, we can use these encodings as features for our downstream machine learning model.

GPT-2 (Generative Pretrained Transformer 2)

GPT-2 (Generative Pretrained Transformer 2) is a large-scale unsupervised language model developed by OpenAI. It is trained on a massive dataset of unannotated text and can generate human-like text and perform various natural language processing (NLP) tasks. GPT-2 is a transformer-based model, which means it uses self-attention mechanisms to process input text.

One of the key features of GPT-2 is its ability to generate human-like text. This is useful for applications such as text summarization, language translation, and content generation. GPT-2 can generate text that is coherent and fluent, making it a powerful tool for natural language generation tasks.

5.Model evaluation:

This module evaluates the performance of the trained model on a held-out test set of labeled messages. This gives an idea of how well the model will generalize to new data.

Reducing false positives and false negatives

There are a variety of techniques that can be used to reduce false positives and false negatives, such as:

- Using a balanced dataset: A balanced dataset has an equal number of spam and non-spam messages. This helps to prevent the model from becoming biased towards one class or the other.

- Using a validation set: A validation set is a subset of the training data that is used to tune the parameters of the model. This helps to prevent overfitting, which can lead to poor performance on new data.
- Using ensemble methods: Ensemble methods combine the predictions of multiple models to produce a more accurate prediction. This can help to reduce both false positives and false negatives.

Model Interpretability:

 Explain how to interpret feature importance from BERT Pre-Trained Models.

 Discuss the insights gained from feature importance analysis and their relevance to AI Powered Spam Classifier.

 Interpret feature importance from Machine Learning models like Naïve Bayes Models and NLP Models to understand the factors influencing Email Spam Detection. Achieving a high level of accuracy.

PROGRAM

AI POWERED SPAM CLASSIFIER

```
# Import Libraries

import numpy as np
import pandas as pd

# Reading the dataset

df = pd.read_csv ('/kaggle/input/sms-spam-collection-dataset/spam.csv', encoding='
SO-8859-1')
df.head ()

# Separating X and y

X = df ['v2']
y = df['v1']
display(X, y)

# Encoding the Labels

from sklearn.preprocessing
import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y)
display(y)
```

Load the Dataset

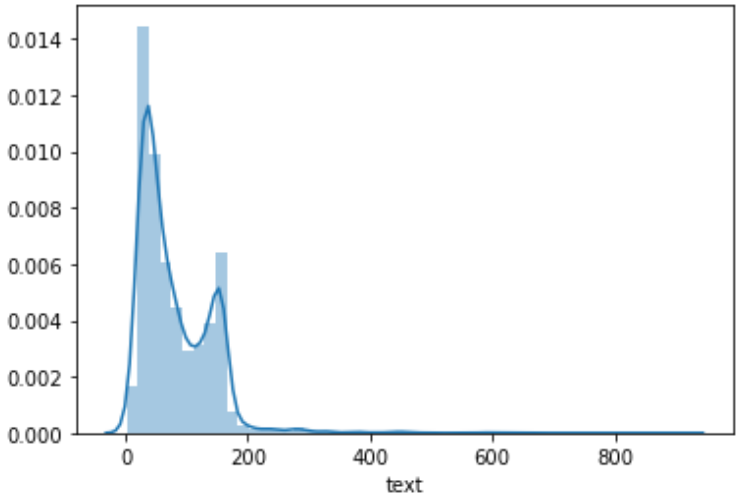
csvfile = '../input/sms-spam-collection-dataset/spam.csv'

In[1] : df.rename(columns={"v1": "target", "v2": "text"}, inplace=True)

df['target'] = (df['target']== 'spam').astype(int)

In[2] : sb.distplot(df.text.str.len())

Out [2] : <matplotlib.axes._subplots.AxesSubplot at 0x7f55aa751d10>



MODEL1 : BERT

```
In[1] : def bert_encode(texts, tokenizer, max_len=512):
                all_tokens = []
                all_masks = []
                all_segments = []

    for text in texts:
        text = tokenizer.tokenize(text)

        text = text[:max_len-2]
        input_sequence = ["[CLS]"] + text + ["[SEP]"]
        pad_len = max_len - len(input_sequence)

        tokens = tokenizer.convert_tokens_to_ids(input_sequence)
        tokens += [0] * pad_len
        pad_masks = [1] * len(input_sequence) + [0] * pad_len
        segment_ids = [0] * max_len

        all_tokens.append(tokens)
        all_masks.append(pad_masks)
        all_segments.append(segment_ids)

    return np.array(all_tokens), np.array(all_masks), np.array(all_segments)
```

```
def build_model(bert_layer, max_len=512):
    input_word_ids = tf.keras.Input(shape=(max_len,), dtype=tf.int32,
name="input_word_ids")
    input_mask = tf.keras.Input(shape=(max_len,), dtype=tf.int32,
name="input_mask")
    segment_ids = tf.keras.Input(shape=(max_len,), dtype=tf.int32,
name="segment_ids")

    pooled_output, sequence_output = bert_layer([input_word_ids, input_mask,
segment_ids])
    clf_output = sequence_output[:, 0, :]
    net = tf.keras.layers.Dense(64, activation='relu')(clf_output)
    net = tf.keras.layers.Dropout(0.2)(net)
    net = tf.keras.layers.Dense(32, activation='relu')(net)
    net = tf.keras.layers.Dropout(0.2)(net)
    out = tf.keras.layers.Dense(1, activation='sigmoid')(net)

    model = tf.keras.models.Model(inputs=[input_word_ids, input_mask,
segment_ids], outputs=out)
    model.compile(tf.keras.optimizers.Adam(lr=1e-5), loss='binary_crossentropy',
metrics=['accuracy'])

    return model
```

In[2] :
```
import sklearn.model_selection

X_train, X_val, y_train, y_val =
sklearn.model_selection.train_test_split(df.text.values, df.target, test_size=0.1,
random_state=0)

X_train = bert_encode(X_train, tokenizer, max_len=200)

X_val = bert_encode(X_val, tokenizer, max_len=200)
```

In[3] :
```
model = build_model(bert_layer, max_len=200)

model.summary()
```

Out [3] :
```
Model: "model"
_____________________
Layer (type)                 Output Shape              Param #     Connected to
==================================================================================
====================
input_word_ids (InputLayer)  [(None, 200)]             0
_____________________
input_mask (InputLayer)      [(None, 200)]             0
_____________________
segment_ids (InputLayer)     [(None, 200)]             0
_____________________
```

```
keras_layer (KerasLayer)        [(None, 1024), (None 335141889   input_word_id
s[0][0]
                                                                 input_mask[0]
[0]
                                                                 segment_ids[0
][0]
__________________________________________________________________________________
_____________________
tf_op_layer_strided_slice (Tens [(None, 1024)]       0           keras_layer[0
][1]
__________________________________________________________________________________
_____________________
dense (Dense)                   (None, 64)           65600       tf_op_layer_s
trided_slice[0][0]
__________________________________________________________________________________
_____________________
dropout (Dropout)               (None, 64)           0           dense[0][0]
__________________________________________________________________________________
_____________________
dense_1 (Dense)                 (None, 32)           2080        dropout[0][0]
__________________________________________________________________________________
_____________________
dropout_1 (Dropout)             (None, 32)           0           dense_1[0][0]
__________________________________________________________________________________
_____________________
dense_2 (Dense)                 (None, 1)            33          dropout_1[0][
0]
==================================================================================
====================
Total params: 335,209,602
Trainable params: 335,209,601
Non-trainable params: 1
__________________________________________________________________________________
____________________
```

MODEL 2 : Logistic Regression

In[4] : import pandas as pd

import numpy as np

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

from sklearn.metrics import confusion_matrix

In[5] : data = pd.read_csv('../input/sms-data-labelled-spam-and-non-spam/SMSSpamCollection', sep='**\t**', names=['label', 'message'])

In[6] : data.head()

Out[6] : label message

0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

```
In[7] :   data.info()

In[8] :   data['label'] = data.label.map({'ham':0, 'spam':1})
          data.head()
```

Out[8] :

	label	message
0	0	Go until jurong point, crazy.. Available only ...
1	0	Ok lar... Joking wif u oni...
2	1	Free entry in 2 a wkly comp to win FA Cup fina...
3	0	U dun say so early hor... U c already then say...
4	0	Nah I don't think he goes to usf, he lives aro...

```
In[9] : from sklearn.model_selection import train_test_split
        X_train, X_test, y_train, y_test = train_test_split(data['message'],
                                           data['label'],
                                           test_size =0.2,
                                           random_state=1)
        print('Number of rows in the total set: {}'.format(data.shape[0]))
        print('Number of rows in the training set: {}'.format(X_train.shape[0]))
        print('Number of rows in the test set: {}'.format(X_test.shape[0]))

Out[9] : Number of rows in the total set: 5572
         Number of rows in the training set: 4457
         Number of rows in the test set: 1115
```

Conclusion and Future Work (Phase 2):

Project Conclusion:

In the Phase 2 conclusion, we will summarize the key findings and insights from the Pre trained Models using BERT Model.

Building a spam email detection model involves various essential steps, from data cleaning and exploratory data analysis to text preprocessing, model building, evaluation, and improvement**.**

Future Work:

We will discuss potential avenues for future work, such as incorporating additional data sources exploring Natural Learning models (NLP) for prediction,or expanding the project into a web application with more features and interactivity.