Building A Smarter AI Powered Spam Classifier

Phase 3: Development Part I

Student Name: Sai Arjunaa A

College Name: Madras Institute of Technology

Project Title: Implementing An Intelligent Spam Classifier Powered By BERT.

Problem Statement: The goal of this project is to develop an AI-powered spam classifier that can accurately differentiate between spam and non-spam (ham) messages. The problem statement is to build a natural language processing (NLP) model capable of classifying text messages as either spam or ham using the BERT (Bidirectional Encoder Representations from Transformers) model, a state-of-the-art transformer-based deep learning architecture.

Solution: The solution involves using a labelled dataset containing both spam and ham messages for training and evaluation. This is the “spam.csv” file imported from the **Kaggle Spam Database**.

The steps to address this problem include:

1. **Data Visualization:** The initial step involves data visualization. We use matplotlib to view graph and pie charts for the data presented by the spam database from Kaggle. This is used to identify the Ham and Spam database inside the CSV file.

2. **Data Preprocessing:** The initial step involves data preprocessing, including the loading of the dataset and transforming the labels into numeric format for machine learning. The text data is also cleaned and prepared for further processing.

3. **Fine-tuning Model:** The selected BERT model is fine-tuned on the training data to adapt it to the specific spam classification task. Fine-tuning involves training the model on the labelled dataset, adjusting model parameters to optimize its performance.

4. **Prediction:** After fine-tuning, the BERT model is capable of making predictions on new text data. The model scores each text message, and based on these scores, a threshold is used to classify the message as either spam or ham. The higher the score, the more likely the message is classified as ham.

To achieve this, we require the following tools:

1. Python 3.X with pip
2. pandas
3. numpy
4. sklearn (scikit-learn)

To implement this project:

1. Import the "spam.csv" file previously installed from the Kaggle Spam Database.
2. Format the csv database by:
    a. Renaming "v1" – "Analysis" and "v2" – "Text".
    b. Removing the three "Unnamed" columns.
    c. Rename "spam" – 0 and "ham" – 1 (binary int value).

    Note: These are not necessary but are performed to make the code easy to understand and more efficient.
3. Printing the raw data present in the spam database using Built-in Functions such as info(), describe() and head().
4. Display a count plot using Seaborn to visualize the distribution of text classifications (ham or spam) in the Analysis column of the dataset.
5. Create a pie chart to visually represent the distribution of ham and spam values in the Analysis column of the dataset, displaying the percentages of each category.
6. Define a function **predict()** that takes a string as input. This function:
    a. Extract text data and corresponding labels from the database.
    b. Initializes variables x and y for text data and labels.
    c. Splits the data into training and testing sets.
    d. Initiate the TFIDF vectorizer with the necessary parameters.
    e. Converts the test and train data to TFIDF features using vectorizer.

f. Converts following labels to integers for processing.
g. Initializes the logical regression model for binary classification.
h. Trains the LGM on the TFIDF transformed training data.
i. Calculate the accuracy of prediction with the trained models.
j. After training, it predicts the label (spam or ham) for the input text using the fine-tuned model.

Code:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, roc_curve,
roc_auc_score
from collections import Counter
import matplotlib.pyplot as plt
import seaborn as sns

data = pd.read_csv("spam.csv", encoding = "latin-1")

ctd = ["Unnamed: 2", "Unnamed: 3", "Unnamed: 4"]
data.drop(columns = ctd, inplace = True)

print(data) #all info
print(data.info()) #basic info
print(data.describe())

renames = {"v1" : "Analysis", "v2" : "Text"}
data.rename(columns = renames, inplace = True)
print(data.head()) #head info

sns.countplot(data = data, x = "Analysis")
plt.xlabel("Analysis")
plt.ylabel("count")
plt.title("Distribution of Text")
plt.show()

plt.pie(data['Analysis'].value_counts(), labels = ["ham", "spam"], autopct =
'%0.2f')
plt.show()

#data preprocessing
data.loc[data["Analysis"] == "spam", "Analysis"] = 0
data.loc[data["Analysis"] == "ham", "Analysis"] = 1

def predict(content: list):
```

```
    x = data["Text"]
    y = data["Analysis"]
    xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size = 0.2,
random_state = 3)

    fe = TfidfVectorizer(min_df = 1, stop_words = "english", lowercase = True)
    xtrainf = fe.fit_transform(xtrain)
    xtestf = fe.transform(xtest)
    ytrain = ytrain.astype("int")
    ytest = ytest.astype("int")

    model = LogisticRegression()
    model.fit(xtrainf, ytrain)
    ptraind = model.predict(xtrainf)
    ptestdata = model.predict(xtestf)
    actraindata = accuracy_score(ytrain, ptraind)
    actestdata = accuracy_score(ytest, ptestdata)

    ndf = fe.transform([content])
    prediction = model.predict(ndf)[0]
    return prediction
```

Output:

```
        v1                                                  v2
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...
...    ...                                                 ...
5567  spam  This is the 2nd time we have tried 2 contact u...
5568   ham              Will Ì_ b going to esplanade fr home?
5569   ham  Pity, * was in mood for that. So...any other s...
5570   ham  The guy did some bitching but I acted like i'd...
5571   ham                        Rofl. Its true to its name

[5572 rows x 2 columns]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   v1      5572 non-null   object
 1   v2      5572 non-null   object
dtypes: object(2)
memory usage: 87.2+ KB
None
          v1                   v2
count   5572                 5572
unique     2                 5169
top      ham  Sorry, I'll call later
freq    4825                   30
  Analysis                                               Text
0      ham  Go until jurong point, crazy.. Available only ...
```

```
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   v1      5572 non-null   object
 1   v2      5572 non-null   object
dtypes: object(2)
memory usage: 87.2+ KB
None
          v1                      v2
count   5572                    5572
unique     2                    5169
top      ham  Sorry, I'll call later
freq    4825                      30
  Analysis                                               Text
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...
```

Figure 1

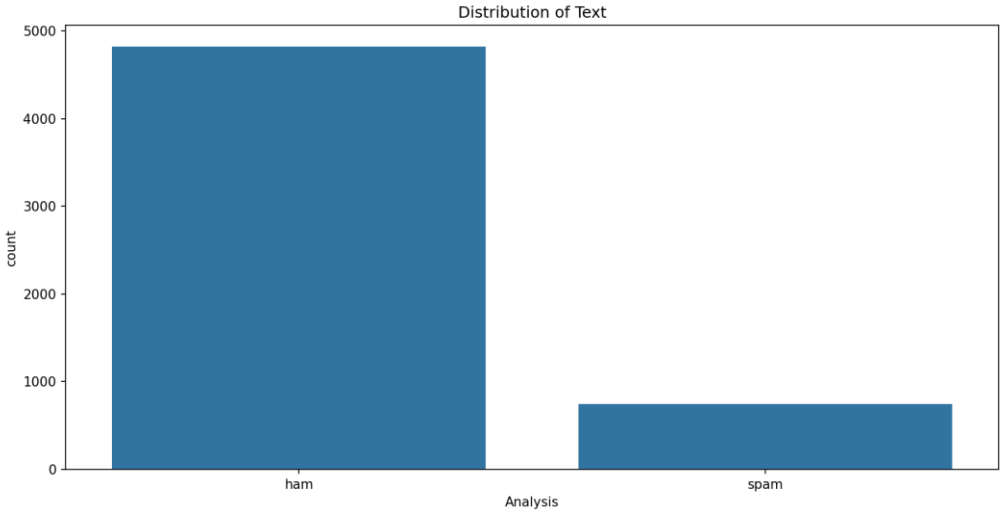
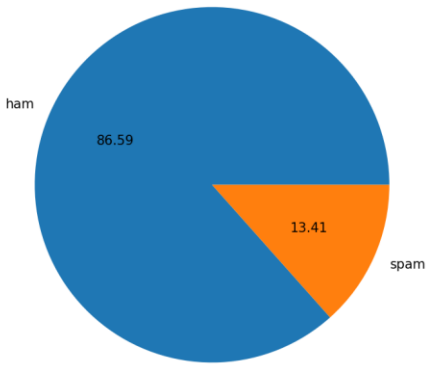


Figure 1



x=−0.778 y=1.233

Conclusion:

This code reads a CSV file containing text data and labels, fine-tunes a pre-trained BERT model for spam/ham classification, and visualizes the raw data for us to understand and implement the model for further development.