Hate Speech Detection

by San Myint Hlaing & Saranya S.

Table of Contents

Project Objectives

- Content Moderation: Use the AI to automatically detect and remove hate speech and offensive content from online platforms to create a safer and more respectful environment.

- Education and Awareness: Use the AI to flag hate speech and provide educational resources or warnings to users, encouraging responsible online behavior.

Project
Objectives

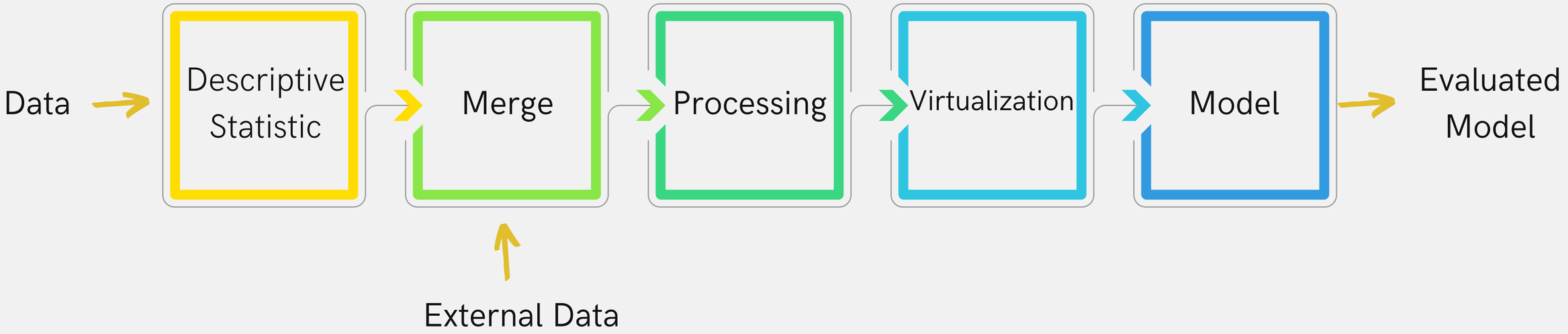
- Content Moderation: Use the AI to automatically detect and remove hate speech and offensive content from online platforms to create a safer and more respectful environment.

- Education and Awareness: Use the AI to flag hate speech and provide educational resources or warnings to users, encouraging responsible online behavior.

Dataset
Collection

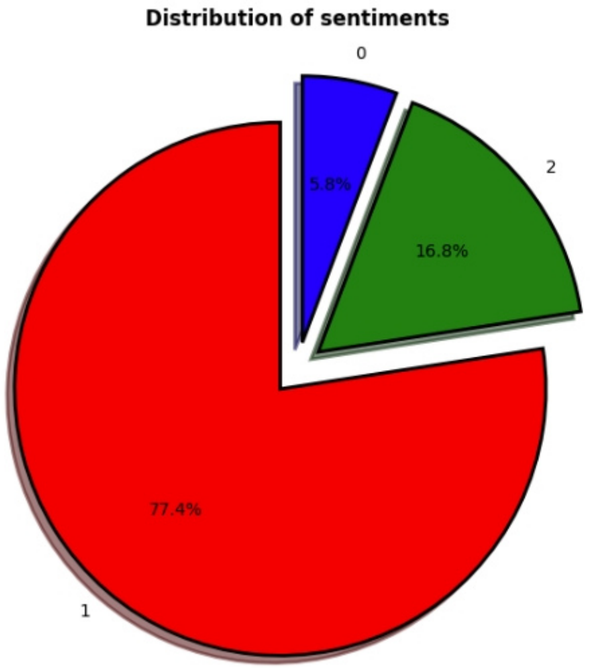
- Kaggle (https://www.kaggle.com/datasets/mrmorj/hate-speech-and-offensive-language-dataset)

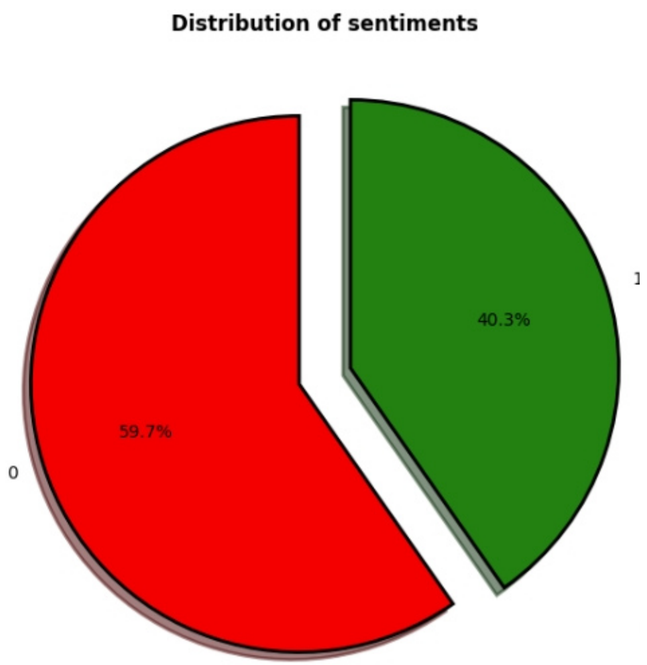
Processes


Data
Descriptive Statistic
Merge
Processing
Virtualization
Model
Evaluated Model
External Data

Descriptive Statistics

Imbalance to Balance Class





Clean Text

Preprocessing

```
stopword = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()

def get_wordnet_pos(treebank_tag):
    if treebank_tag.startswith('J'):
        return wordnet.ADJ
    elif treebank_tag.startswith('V'):
        return wordnet.VERB
    elif treebank_tag.startswith('N'):
        return wordnet.NOUN
    elif treebank_tag.startswith('R'):
        return wordnet.ADV
    else:
        return ''

def clean_text(text):
    text = str(text).lower()
    patterns = [
        ('\[.*?\]', ''),
        ('https?://\S+|www\.\S+', ''),
        ('<.*?>+', ''),
        (r"\@w+|\#", ''),
        (r"[^\w\s]", ''),
        (f'[{re.escape(string.punctuation)}]', ''),
        ('\n', ''),
        ('\w*\d\w*', ''),
    ]
    for pattern, repl in patterns:
        text = re.sub(pattern, repl, text)
    tokens = word_tokenize(text)
    lemmatized_tokens = [lemmatizer.lemmatize(word, wordnet.VERB) for word in tokens if word not in stopword]
```

Virtualization

Word Cloud

```
def plot_wordcolund(df):
    words = " ".join(df['text'])
    stopword = set(stopwords.words('english'))
    wordcloud = WordCloud(stopwords=stopword, background_color='white', max_words=2000, height = 2000, width=4000).generate(words)
    plt.figure(figsize = (16,8))
    plt.imshow(wordcloud)
    plt.axis('off')
    plt.show()
```

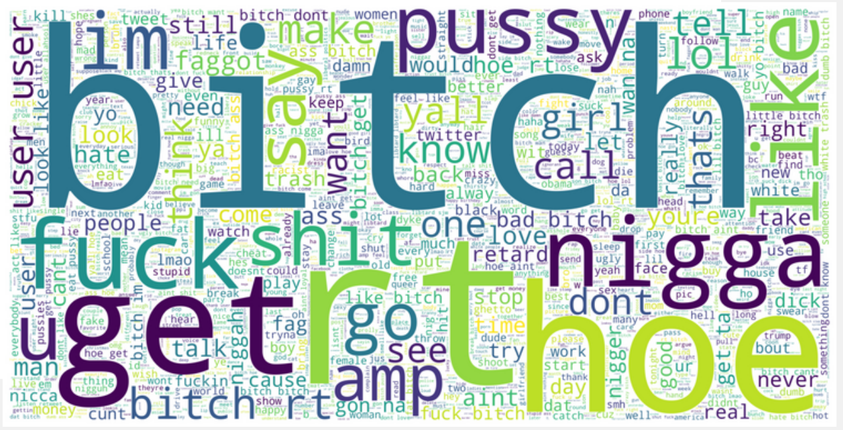
```
plot_wordcolund(df[df['label']==0])
```

Virtualization

Word Cloud

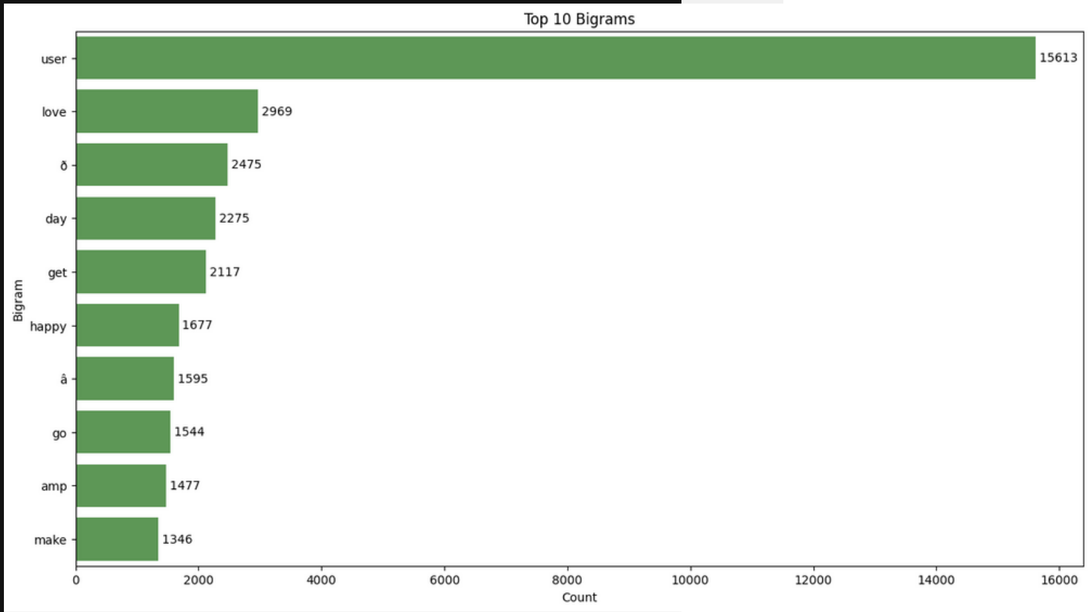
```
def plot_wordcolund(df):
    words = " ".join(df['text'])
    stopword = set(stopwords.words('english'))
    wordcloud = WordCloud(stopwords=stopword, background_color='white', max_words=2000, height = 2000, width=4000).generate(words)
    plt.figure(figsize = (16,8))
    plt.imshow(wordcloud)
    plt.axis('off')
    plt.show()
```

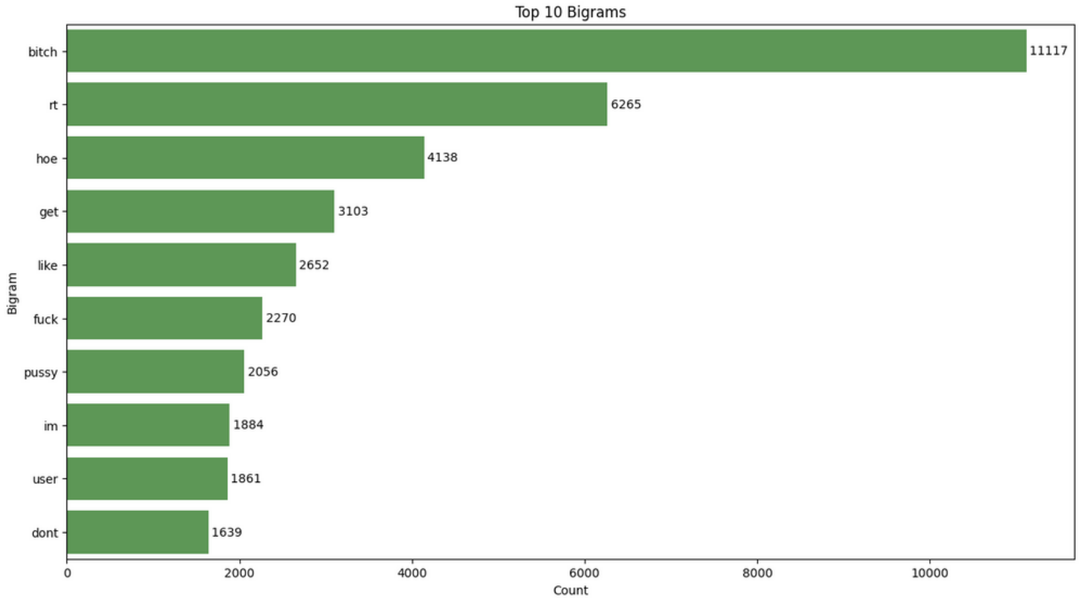
```
plot_wordcolund(df[df['label']==0])
```



Virtualization

Bigrams





BERT Model

Model Architecture

BERT

A pre-trained language model that is fine-tuned for the specific task of hate speech detection. BERT is used to convert the text data into numerical representations that can be fed into a deep learning model for classification.

```
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
```

Train-Test Model

DataFrame

```
df_train = df.loc[df["data_type"]=="train"].sample(frac=1)
df_val = df.loc[df["data_type"]=="val"].sample(frac=1)
df_test = df.loc[df["data_type"]=="test"].sample(frac=1)
```

```
train_ds = tf.data.Dataset.from_tensor_slices((df_train.text.values, df_train.label.values)).batch(32, drop_remainder=False)
val_ds = tf.data.Dataset.from_tensor_slices((df_val.text.values, df_val.label.values)).batch(32, drop_remainder=False)
test_ds = tf.data.Dataset.from_tensor_slices((df_test.text.values, df_test.label.values)).batch(32, drop_remainder=False)
```

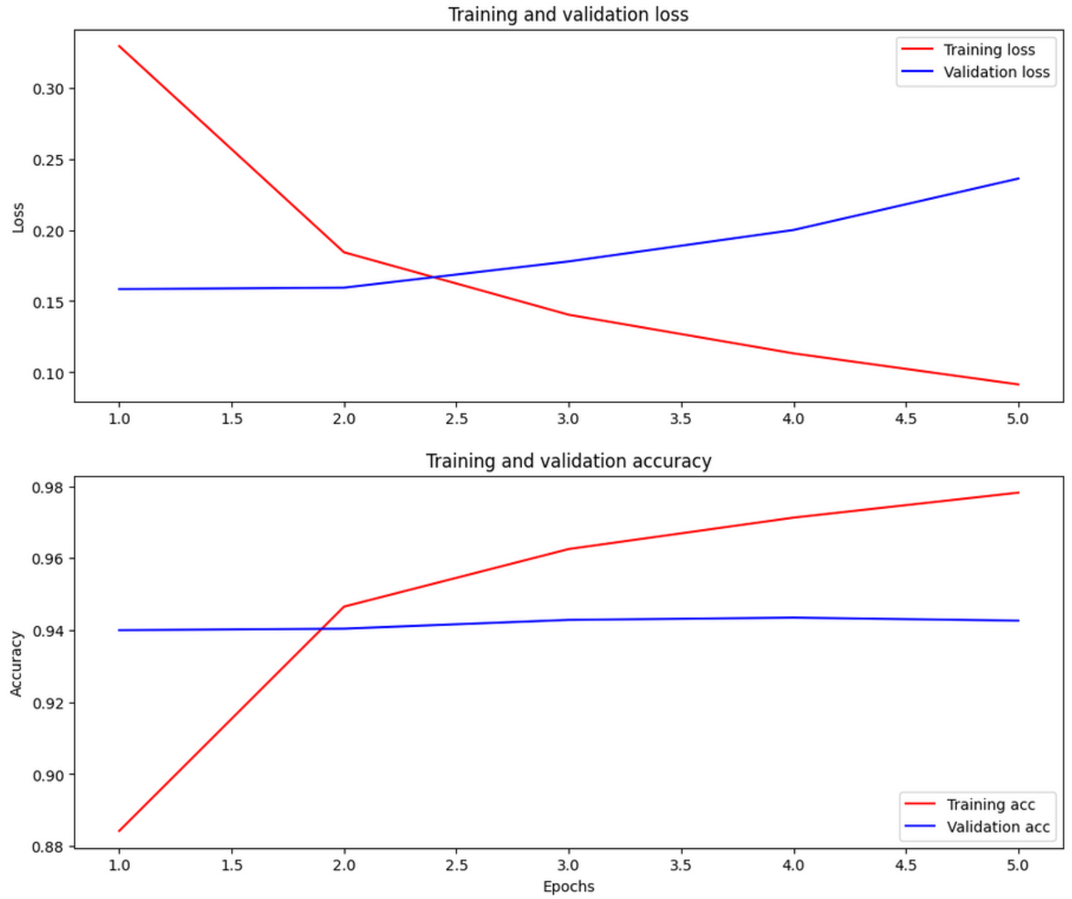
Train-Test Model

Build, train, and evaluate a text classification using BERT

```
Model: "model_5"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
text (InputLayer)               [(None,)]            0           []

preprocessing (KerasLayer)      {'input_word_ids':   0           ['text[0][0]']
                                 (None, 128),
                                 'input_mask': (Non
                                e, 128),
                                 'input_type_ids':
                                 (None, 128)}

BERT_encoder (KerasLayer)       {'sequence_output':  28763649    ['preprocessing[0][0]',
                                 (None, 128, 512),                'preprocessing[0][1]',
                                 'default': (None,                'preprocessing[0][2]']
                                512),
                                 'pooled_output': (
                                None, 512),
                                 'encoder_outputs':
                                 [(None, 128, 512),
                                 (None, 128, 512),
                                 (None, 128, 512),
                                 (None, 128, 512)]}

dense_5 (Dense)                 (None, 128)          65664       ['BERT_encoder[0][5]']

dropout_5 (Dropout)             (None, 128)          0           ['dense_5[0][0]']

classifier (Dense)              (None, 2)            258         ['dropout_5[0][0]']

==================================================================================================
Total params: 28,829,571
Trainable params: 28,829,570
Non-trainable params: 1
__________________________________________________________________________________________________
You must install pydot (`pip install pydot`) and install graphviz (see instructions at https://graphviz.gitlab.io/download/) for plot_model to work.
Training model with https://tfhub.dev/tensorflow/small_bert/bert_en_uncased_L-4_H-512_A-8/1
Epoch 1/5
c:\Users\User\miniconda3\envs\tf-text\lib\site-packages\keras\backend.py:5582: UserWarning: "`sparse_categorical_crossentropy` received `from_logits=True`, but the `output` argument was produced by a Softmax activation and thus does not represent logits. Was this intended?
  output, from_logits = _get_logits(
1010/1010 [==============================] - 217s 211ms/step - loss: 0.3292 - accuracy: 0.8842 - val_loss: 0.1585 - val_accuracy: 0.9400
Epoch 2/5
1010/1010 [==============================] - 213s 211ms/step - loss: 0.1843 - accuracy: 0.9466 - val_loss: 0.1595 - val_accuracy: 0.9404
Epoch 3/5
1010/1010 [==============================] - 213s 211ms/step - loss: 0.1404 - accuracy: 0.9626 - val_loss: 0.1779 - val_accuracy: 0.9429
Epoch 4/5
1010/1010 [==============================] - 213s 211ms/step - loss: 0.1133 - accuracy: 0.9713 - val_loss: 0.2000 - val_accuracy: 0.9435
Epoch 5/5
1010/1010 [==============================] - 212s 210ms/step - loss: 0.0915 - accuracy: 0.9782 - val_loss: 0.2361 - val_accuracy: 0.9427
316/316 [==============================] - 26s 83ms/step - loss: 0.2422 - accuracy: 0.9418
Loss: 0.24223005771636963
Accuracy: 0.9417716264724731
```

Train-Test Model

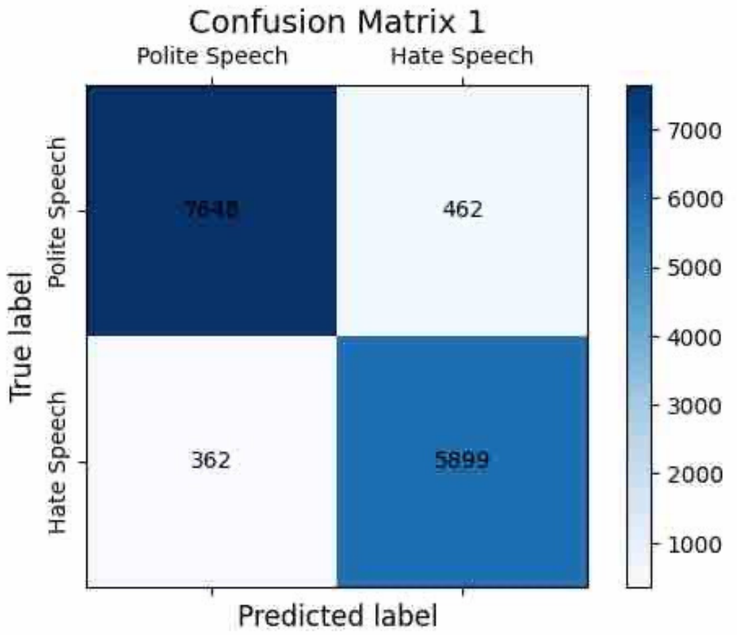
Loss and Accuracy Plots

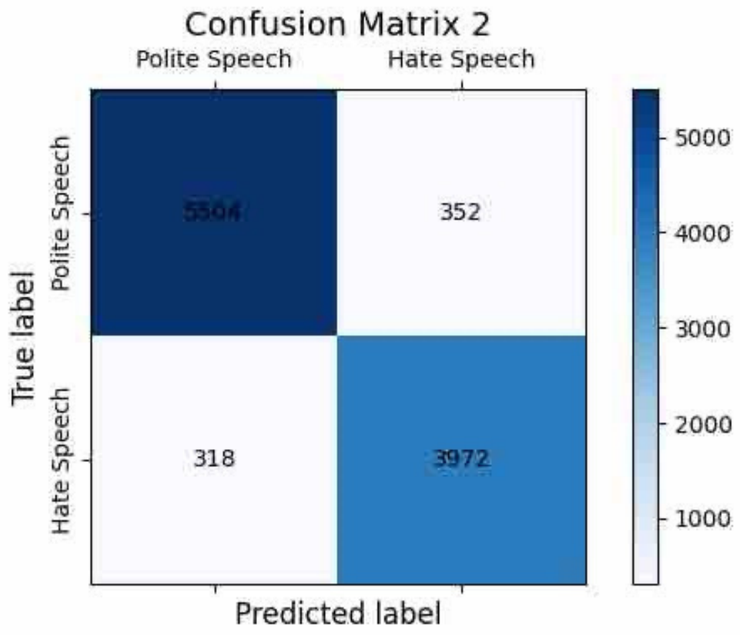


Result

Evaluated Model

Confustion Matrix





Pre-Trained Text Classifination Model

```
# Prediction
text = input()
y_pred = classifier_model.predict([text])
y_pred = np.argmax(y_pred, axis=-1)

print(f'This contains harsh words. Please use polite words.') if y_pred == 1 else print(text)
```

You are a bitch.
1/1 [==============================] - 1s 1s/step
This contains harsh words. Please use polite words.

You are incredible.
1/1 [==============================] - 0s 246ms/step
You are incredible.

THANK YOU

by San Myint Hlaing & Saranya S.