Text/Passage Classification

EE6405 Capstone Project

Cluster D Table 26

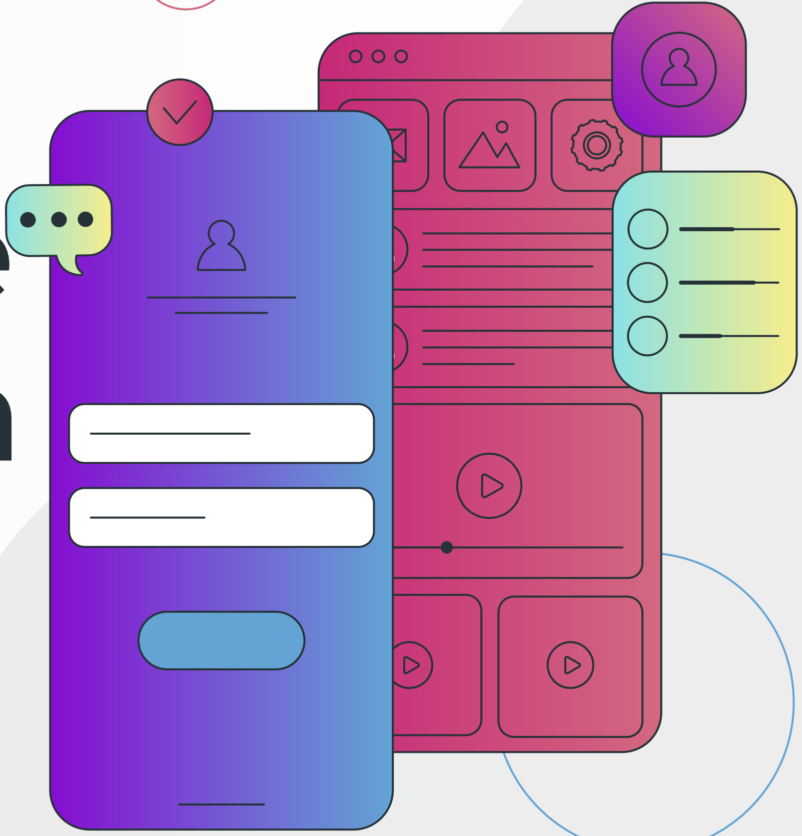


TABLE OF CONTENTS

01

INTRODUCTION

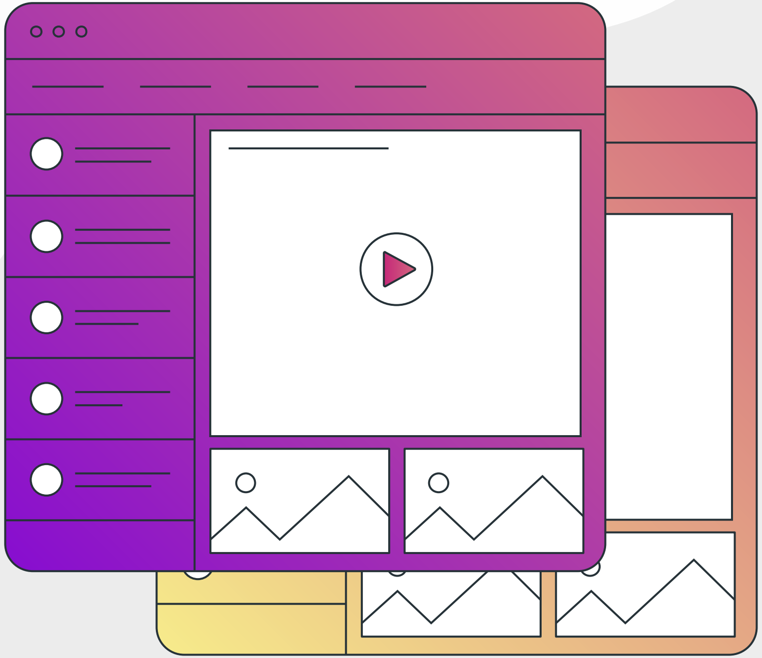
Project Introduction and
Use Case

INTRODUCTION & USE CASE

A study of how the **different models** perform and how they can be improved to be a **text classifier** based on **text dichotomies**.

Datasets to be applied:

- **IMDB Movie Reviews (Sentiment)**
- **Twitter Tweets (Racism)**
- **News Articles (Reliability)**



02

DATASET & MODEL TRAINING

Evaluation of each model, preprocessing and training exploration through hyperparameter tuning

Models Evaluated

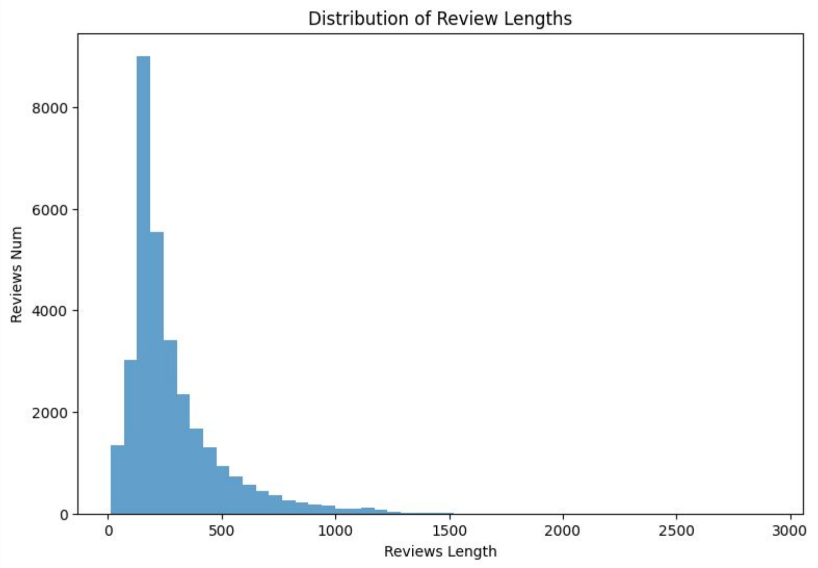
Model Type	Description	How is it explored
Traditional	Traditional SVM	POS Tagging
Simple RNN	Simple RNN based NN	Hyperparameter Tuning
LSTM	Using LSTM to capture long term dependencies	Hyperparameter Tuning
BiLSTM + CNN	CNN used to recognize local patterns/features	Hyperparameter Tuning, Dropout, Sequential Ordering of each stage of the NN
BERT/RoBERT	Pretrained DL Network	Fine Tuning

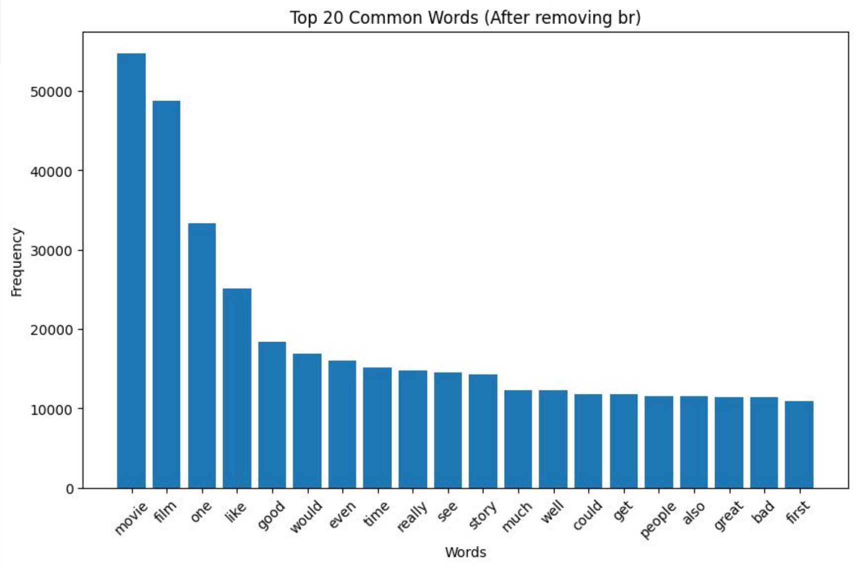
Dataset Breakdown
#1: IMDB Movie Reviews

- Balanced Dataset of 32K Reviews in Train, 8K Reviews in Test
- 2 Labels - 0: Negative, 1: Positive
- Binary classification of each review is trained and predicted for the sentiment

Data Source: Kaggle
https://www.kaggle.com/datasets/thedevastator/imdb-movie-review-sentiment-dataset?select=train.csv

Dataset Breakdown
#1: IMDB Movie Reviews





Data Source: Kaggle
https://www.kaggle.com/datasets/thedevastator/imdb-movie-review-sentiment-dataset?select=train.csv

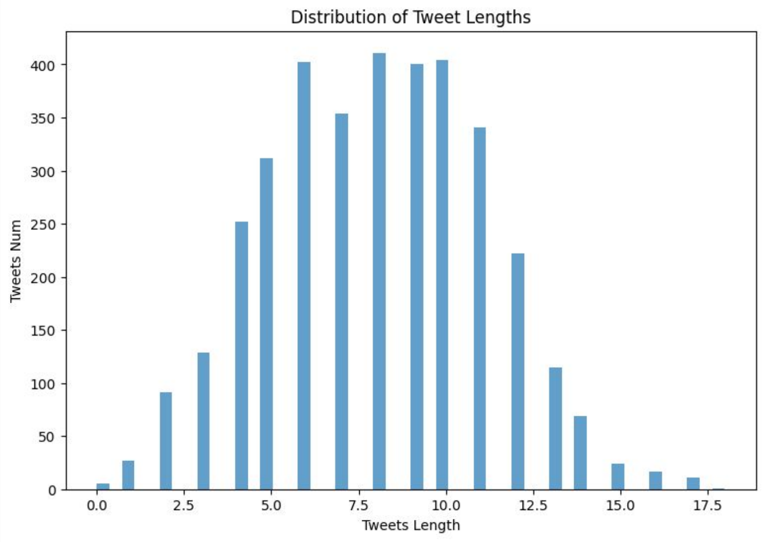
Dataset Breakdown
#2: Labelled Twitter Tweets

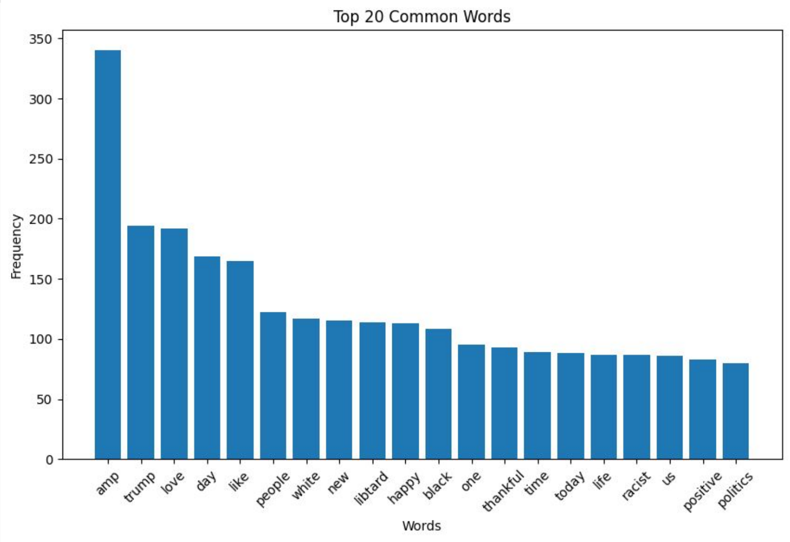
- Imbalanced Dataset of 24K Tweets in Train, 6K Tweets in Test
- 2 Labels - 0: Non racist/sexist, 1: racist/sexist
- Balanced Dataset of 4K Tweets in Train, 1K Tweets in Test

  (Due to the limited sample size of racist/sexist tweets)
- Binary classification of each tweet is trained and predicted for the sentiment

Data Source:GitHub, Twitter Sentiment Analysis Dataset
https://github.com/prateekjoshi565/twitter_sentiment_analysis

Dataset Breakdown
#2: Labelled Twitter Tweets





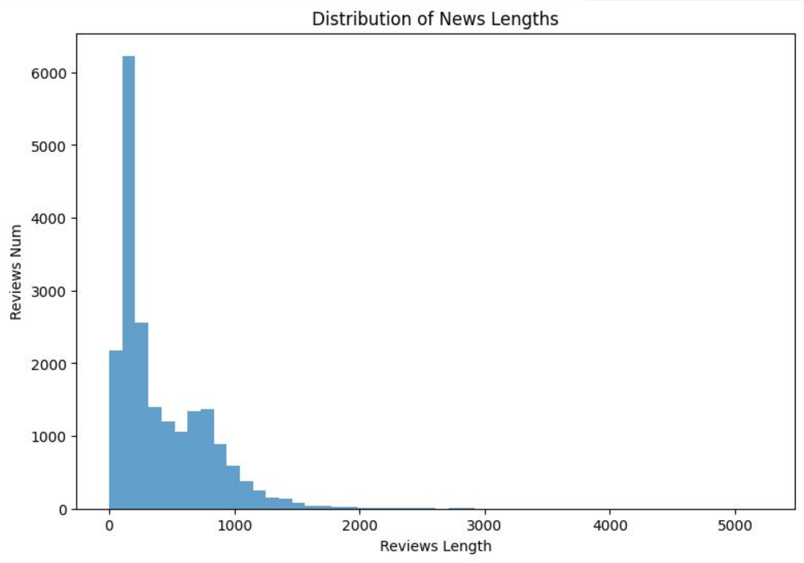
Data Source:GitHub, Twitter Sentiment Analysis Dataset
https://github.com/prateekjoshi565/twitter_sentiment_analysis

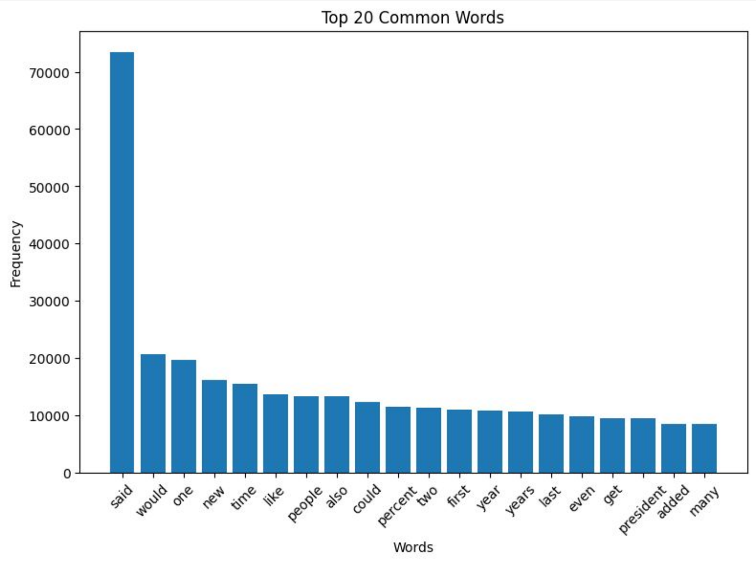
Dataset Breakdown
#3: Labelled Unreliable News

- 48K news articles in train data, 3K in test
- 4 labels Original - 1: Satire, 2: Hoax, 3: Propaganda, 4: Reliable News
- 2 labels used - 0: Satire, 1: Reliable News
- 2-Way (Binary) classification is attempted since the other 2 dataset use cases are also based on binary classification (sentiment)
- Reliability of the news document will be predicted

Data Source: GitHub, CompareNet_FakeNewsDetection Dataset
https://github.com/BUPT-GAMMA/CompareNet_FakeNewsDetection/releases/tag/dataset

Dataset Breakdown
#3: Labelled Unreliable News





Data Source: GitHub, CompareNet_FakeNewsDetection Dataset
https://github.com/BUPT-GAMMA/CompareNet_FakeNewsDetection/releases/tag/dataset

Model Training
#1 Traditional SVM

- Pre-process the data by removing characters, tokenizing the sentence, removing stop words, part-of-speech tagging and lemmatizing the words, in that order.
- A Tf-Idf Vectorizer is used to convert the processed sentence / passage into a TF-IDF feature matrix, used for training the model.
- The final step is training the Support Vector Machine model. The Radial Basis Function kernel is selected to create this model here.

```python
def preprocess_text(self, text):
    text = re.sub('(<.*?>)', ' ', text)
    text = re.sub('[,\.!?:()"]', '', text)
    text = text.strip()
    text = re.sub('[^a-zA-Z"]',' ', text)
    text = text.lower()
    text = self.tagged_lemma(text)

    words = tf.keras.preprocessing.text.text_to_word_sequence(text)
    stop_words = set(stopwords.words('english'))
    filtered_words = [w for w in words if not w in stop_words]
    text = " ".join(filtered_words)

    return text
```

```python
# Get tfidf word embeddings
tv=TfidfVectorizer(stop_words='english')
train_review_tfidf=np.asarray(tv.fit_transform(train_review).todense())
test_review_tfidf=np.asarray(tv.transform(test_review).todense())

# Training
clf = svm.SVC(kernel='rbf')
clf.fit(train_review_tfidf, train_sent)
```

Model Training #2 Simple RNN

- Preprocessing for removing stopwords, tokenizing, changing to sequences and padding is completed before training
- In the sequential model, embedding layer is first added, to allow the layer to learn the word embedding within the model
- **Simple RNN** Model is added afterwards with 32 neurons.
- Sigmoid activation function is used for probabilistic binary classification

```
model = Sequential([
        Embedding(input_dim=10000, output_dim=32,
input_length=100),
        SimpleRNN(32),
        Dense(1, activation='sigmoid')
    ])

model.compile(optimizer=Adam(learning_rate=0.001),
loss='binary_crossentropy', metrics=['accuracy'])

model.fit(X_train, y_train, batch_size=128,
epochs=5, validation_split=0.2)
predictions = (model.predict(X_test) >
0.5).astype("int32").flatten()
```

Model Training
#3 LSTM

- Preprocessing for removing stopwords, tokenizing, changing to sequences and padding is completed before training
- In the sequential model, embedding layer is first added, to allow the layer to learn the word embedding within the model
- **LSTM** Model is added afterwards with 32 neurons.
- Sigmoid activation function is used for probabilistic binary classification

```
model = Sequential([
        Embedding(input_dim=10000, output_dim=32,
input_length=100),
        LSTM(32),
        Dense(1, activation='sigmoid')
    ])

model.compile(optimizer=Adam(learning_rate=0.001),
loss='binary_crossentropy', metrics=['accuracy'])

model.fit(X_train, y_train, batch_size=128,
epochs=5, validation_split=0.2)
predictions = (model.predict(X_test) >
0.5).astype("int32").flatten()
```

Model Training
#4 BiLSTM + CNN

- Preprocessing for removing stopwords, tokenizing, changing to sequences and padding is completed before training
- In the sequential model, embedding layer is first added, to allow the layer to learn the word embedding within the model
- A **1D Spatial Dropout** of 0.2 is added to prevent overfitting
- **CNN** Model is added afterwards with 32 filters and kernel size 3.
- **BiLSTM** comes after CNN with 32 units
- Sigmoid activation function is used for probabilistic binary classification

```
model = Sequential([
        Embedding(input_dim=10000, output_dim=32,
input_length=100),
        SpatialDropout1D(0.2),
        Conv1D(filters=32, kernel_size=3,
padding='same', activation='relu'),
        Bidirectional(LSTM(32)),
        Dense(1, activation='sigmoid')
    ])

model.compile(optimizer=Adam(learning_rate=0.001),
loss='binary_crossentropy', metrics=['accuracy'])

    model.fit(X_train, y_train, batch_size=128,
epochs=5, validation_split=0.2)
    predictions = (model.predict(X_test) >
0.5).astype("int32").flatten()
```

Fine-tuning
#5 Using pretrained RoBert and Bert models

- Preprocess the data, select specific categories or labels, and resample to balance the data set.
- Load the pretrained model and its tokenizer.
- Use a tokenizer to convert text into a format that the model can process, including truncation and padding of sequences to a fixed length.
- Configure training parameters and define measurement and evaluation methods.

03

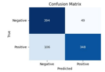
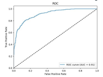
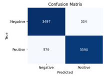
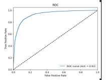
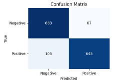
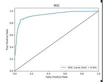
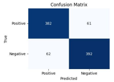
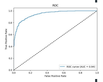
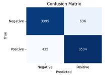
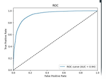
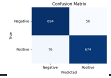
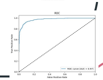
MODEL EVALUATION

Comparison across different models
Limitations faced by each model
Potential Improvements

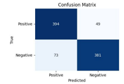
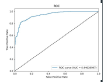
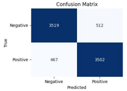
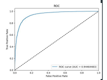
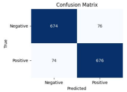
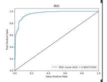
Model Comparison (Overall)

Model Type	Data	Performance
Traditional SVM	Tweet Sentiment (Racist/sexist or Not)	Precision: 0.8736, Recall: 0.8370, **F1: 0.8549, AUC: 0.9393**
	IMDB Review (Positive, Negative)	Precision: 0.7658, Recall: 0.9167, **F1: 0.808, AUC: 0.8016**
	News Classification (Truth, Satire)	Precision: 0.928, Recall: 0.928, **F1: 0.928, AUC: 0.5263**

Model Comparison (Overall)

Model Type	Data	Performance		
SimpleRNN	Tweet Sentiment (Racist/sexist or Not)	Precision: 0.8170 Recall: 0.8062, **F1: 0.8115, AUC: 0.9009**		
	IMDB Review (Positive, Negative)	Precision: 0.8639, Recall: 0.8541, **F1: 0.8590, AUC: 0.9309**		
	News Classification (Truth, Satire)	Precision: 0.899 Recall: 0.868, **F1: 0.883, AUC: 0.949**		
LSTM	Tweet Sentiment (Racist/sexist or Not)	Precision: 0.8819 Recall: 0.8392, **F1: 0.8600, AUC: 0.9447**		
	IMDB Review (Positive, Negative)	Precision: 0.8475, Recall: 0.8904, **F1: 0.8684, AUC: 0.9392**		
	News Classification (Truth, Satire)	Precision: 0.9233, Recall: 0.8987, **F1: 0.9108, AUC: 0.9721**		

Model Comparison (Overall)

Model Type	Data	Performance	
CNN + LSTM	Tweet Sentiment (Racist/sexist or Not)	Precision: 0.8736, Recall: 0.8370, **F1: 0.8549, AUC: 0.9393**	 
	IMDB Review (Positive, Negative)	Precision: 0.8724, Recall: 0.8823, **F1: 0.8774, AUC: 0.9446**	 
	News Classification (Truth, Satire)	Precision: 0.8989, Recall: 0.9013, **F1: 0.9001, AUC: 0.9638**	 
BERT	Tweet Sentiment (Racist/sexist or Not)	Precision: 0.888, Recall: 0.878, **F1: 0.8834**	
	IMDB Review (Positive, Negative)	Precision: 0.8700, Recall: 0.8820, **F1: 0.8706**	
	News Classification (Truth, Satire)	Precision: 0.8474, Recall: 0.9706, **F1: 0.9049**	

Model Comparison (Overall)

Model Type	Data	Performance
RoBERT	Tweet Sentiment (Racist/sexist or Not)	Precision: 0.888, Recall: 0.872, **F1: 0.88**
	IMDB Review (Positive, Negative)	Precision: 0.902, Recall: 0.922, **F1: 0.9115**
	News Classification (Truth, Satire)	Precision: 0.844, Recall: 0.990, **F1: 0.9116**

Model Evaluation
#1 Traditional SVM

- SVM can be computationally expensive and is affected by the size of the data a lot. Additionally, it cannot utilize the processing power of GPUs like neural networks.
- Although the accuracy varied on the size and dimensions of the dataset, It performed reasonably well on all datasets.
- Despite being a basic model and being outperformed by modern methods like LSTM and Transformers, it is a good alternative for less complex tasks like classification.

Model Evaluation
#2 Simple RNN

- Limitations/Challenges
  - RNN has a long term dependency issue as it has a vanishing gradient problem. This can be addressed in the next model which utilizes an LSTM
  - A data input too large including stopwords and without limiting max vocab size will cause the model to take too long to train. Therefore, a max length of 100 is set.
  - The preprocessing of the data also required some experimentation before a combination of trimmed inputs allowed for a reasonable training speed and accuracy.

Model Evaluation
#3 LSTM

- Limitations/Challenges
    - LSTM is more computationally intensive compared to vanilla RNN, and utilized much more of the GPU that is available in colab. It could result in a higher cost associated with training more complicated sentiment analyzers such as ones with a floating point.
    - The performance is measured in a similar manner, and the AUC is slightly better in general compared to Simple RNN.

Model Evaluation
#4 BiLSTM + CNN

- Limitations/Challenges
  - The hybrid CNN LSTM model surpasses the LSTM only model slightly, as it has the ability to capture both local and long range dependencies.
  - This more complicated approach captures the sequenced based dependencies better.
  - However, it is obvious that we are at the point of diminishing returns, since the amount of data available for the model is not providing any more insights that are already captured, hence only the small improvement in AUC. To further improve, we investigate a fine tuned BERT/RoBERT model next.

Model Evaluation
#5 Comparison of pretrained RoBert and Bert models

- The Bert and RoBert model based on the transformer architecture provides self attention and is able to capture more nuanced relationships.
- For the same dataset in the same environment, pretrained RoBert model performs better on the test data set, including precision, recall, F1 score, etc.
- In addition, the Robert model is faster in terms of sample evaluation speed.

Model Evaluation
#5 Comparison of pretrained RoBert and Bert models

For the same dataset “news_balancedtest.csv”

Evaluation result from Bert

```
{'eval_loss': 0.7662724852561951,
 'eval_accuracy': 0.8979319546364243,
 'eval_f1': 0.9049098819142325,
 'eval_precision': 0.8474970896391153,
 'eval_recall': 0.9706666666666667,
 'eval_runtime': 78.8792,
 'eval_samples_per_second': 19.004,
 'eval_steps_per_second': 2.383}
```

Evaluation result from RoBert

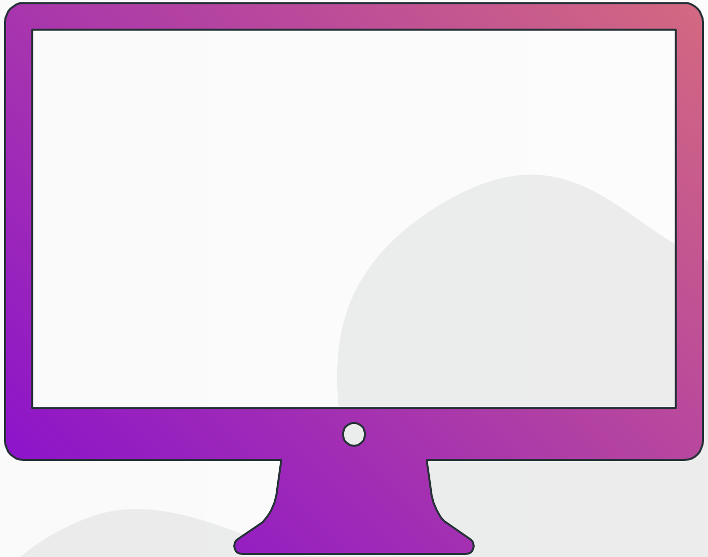
```
{'eval_loss': 0.5538364052772522,
 'eval_accuracy': 0.9039359573048699,
 'eval_f1': 0.9116564417177914,
 'eval_precision': 0.8443181818181819,
 'eval_recall': 0.9906666666666667,
 'eval_runtime': 64.949,
 'eval_samples_per_second': 23.08,
 'eval_steps_per_second': 2.895}
```

04

USER INTERFACE

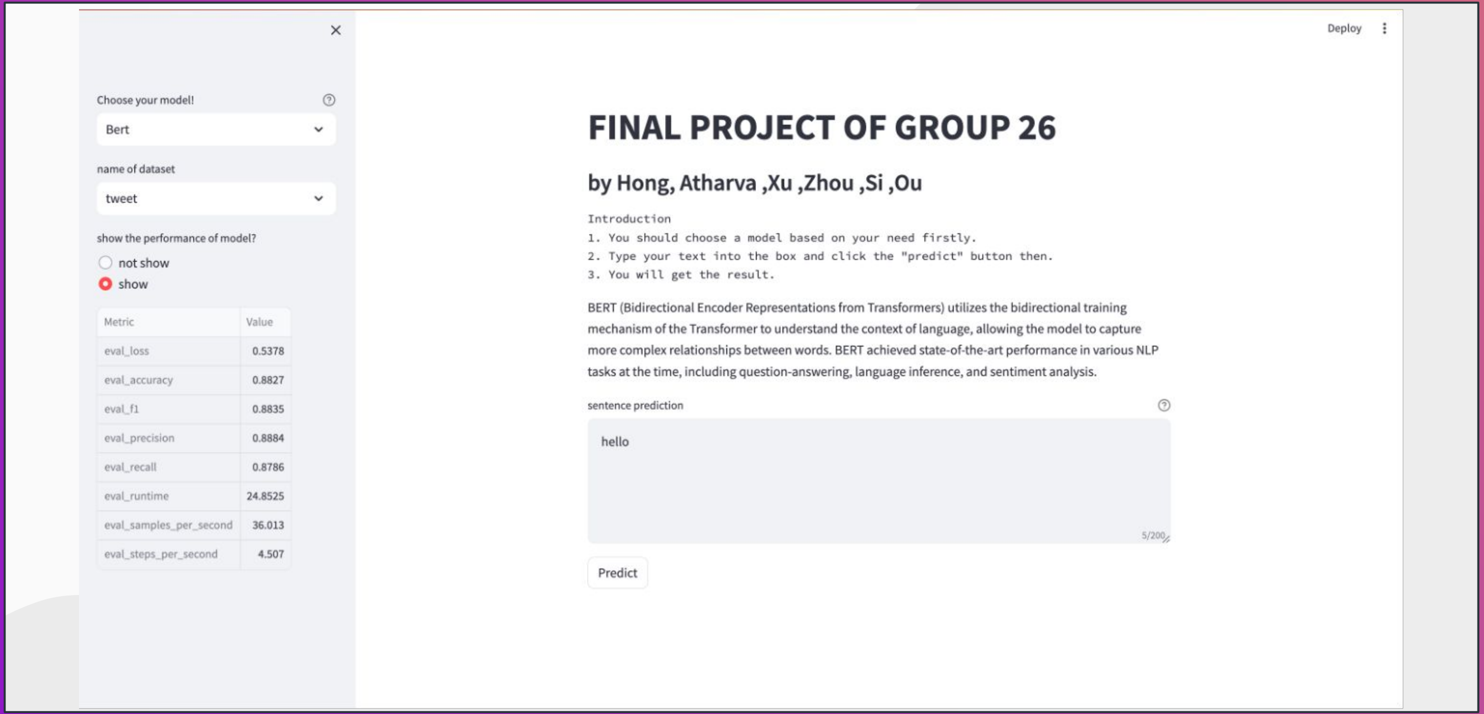
Testing of model with new datasets
Confidence score of each model

User Interface

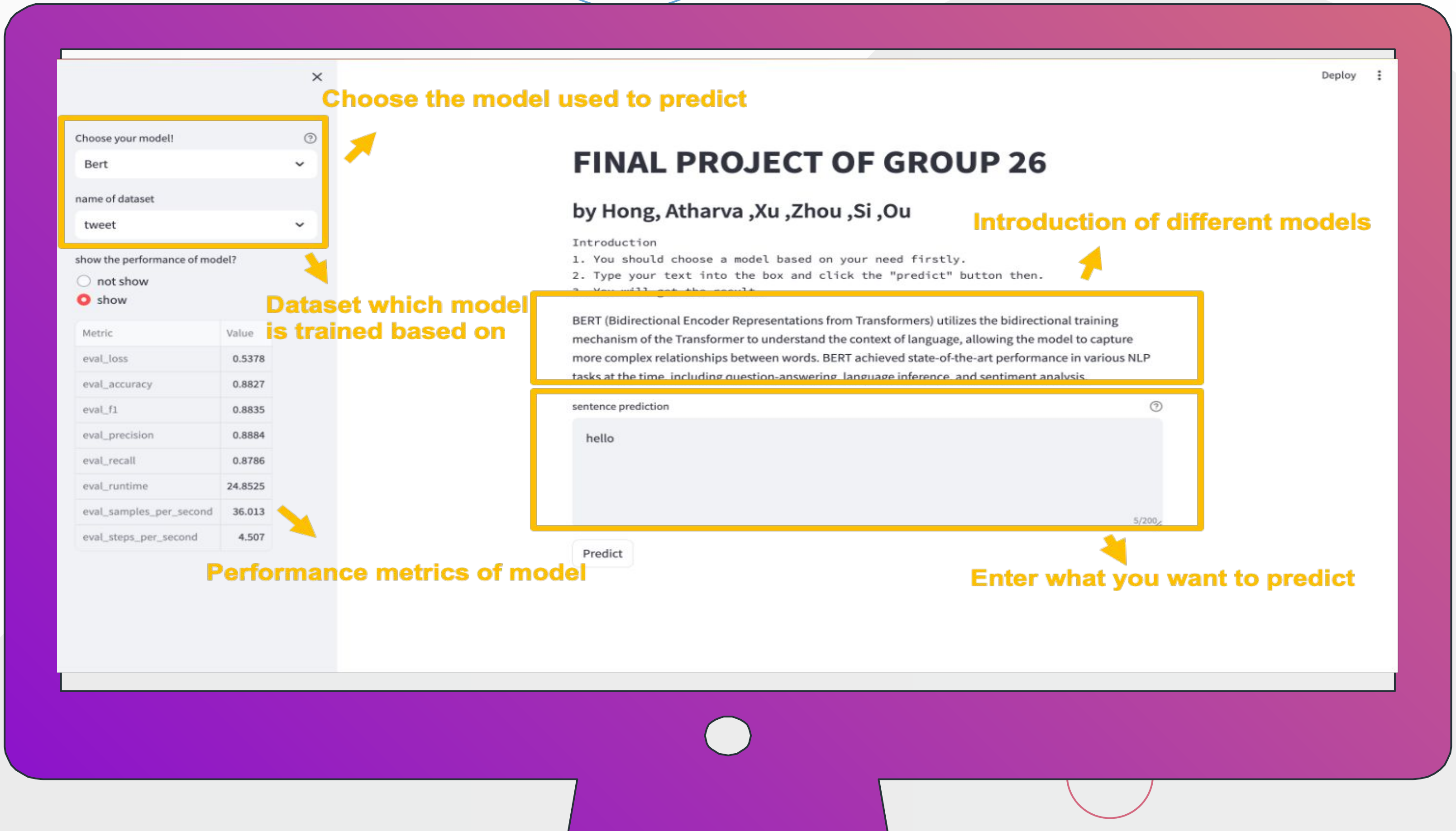


To enable users to do a **live prediction of their chosen input**, we have built a user interface with **streamlit** to facilitate the process.

User Interface



User Interface



Thank You