A close up of a sign

Description automatically generated

**Project Report**

**On**

**Twitter Bot Detection**

**Fundamentals of Deep Learning Lab**

**Subject Code: DSE 3141**

**RITIKA GUPTA**

**REGISTRATION NO: 220968182**

**Department of Data Science & Computer Applications,**

**Manipal Institute of Technology,**

**Manipal**

**Table of Contents**

1. Problem Statement
2. Metadata of Dataset
3. Exploratory Data Analysis
4. Preprocessing pipeline specific to data
5. Identify relevant performance metrics
6. Define Project Objectives
7. Literature review of Models
8. Baseline Model
9. Model Training

**PROBLEM STATEMENT**

The project focuses on detecting spambots on Twitter using Bidirectional Long Short-Term Memory (BiLSTM) networks combined with word embeddings. Twitter, a widely used social platform, is often exploited by spambots that spread spam or malicious content. Traditional methods for bot detection often rely on handcrafted features and user profiles, which can be labor-intensive and prone to inaccuracies. This project aims to develop a deep learning-based model that efficiently distinguishes spambots from human accounts based purely on tweet content, without relying on user profiles or network structures.

**METADATA OF DATASET**

The dataset used for this project is the publicly available Cresci-2017 dataset, which contains two main categories: human accounts and spambot accounts. The dataset includes 3,474 human accounts with over 8 million tweets and 1,455 spambot accounts with over 3 million tweets. For testing, two separate test sets were prepared: Test Set #1 includes mixed accounts from the Social-bot-1 group, while Test Set #2 includes accounts from Social-bot-3, which focuses on product spammers.

<https://botometer.osome.iu.edu/bot-repository/datasets/cresci-2017/cresci-2017.csv.tar.gz>

**EXPLORATORY DATA ANALYSIS**

Exploratory data analysis (EDA) was conducted to understand the differences between human accounts and spambots. Key insights from the dataset include the nature of tweets from spambots, which frequently contain exaggerated language and external links, compared to human accounts that typically engage in more personal interactions. Word clouds were generated to visualize the most common words used in human and spambot tweets, providing insights into language patterns and content focus.



**PREPROCESSING PIPELINES SPECIFIC TO DATA**

The preprocessing pipeline involved several steps:

* Tokenization of tweets using the NLTK toolkit, mapping special Twitter elements like hashtags, mentions, and URLs to predefined tokens.
* The tweets were then transformed into numeric vectors using pre-trained GloVe word embeddings based on 2 billion tweets. These embeddings provided dense representations for words, ensuring that similar words were located close to each other in the vector space.
* Out-of-vocabulary words were handled by assigning them a shared vector representation, allowing the model to generalize across unseen terms.

**IDENTIFY RELEVANT PERFORMANCE METRICS**

The model performance was evaluated using several standard metrics:

* **Precision:** The ratio of correctly identified spambots among all accounts classified as spambots.
* **Recall (Sensitivity):** The ratio of actual spambots that were correctly identified.
* **Specificity:** The ratio of human accounts correctly identified as non-spambots.
* **Accuracy:** The overall proportion of correctly classified accounts.
* **F-Measure:** The harmonic mean of Precision and Recall.
* **Matthews Correlation Coefficient (MCC):** A correlation coefficient capturing the quality of the classification.

**DEFINE PROJECT OBJECTIVES**

The primary objectives of the project were:

* Develop an RNN-based model, that does not rely on handcrafted features or prior knowledge about user profiles or network behavior.
* Utilize only the tweet content for bot detection, leveraging word embeddings for contextual understanding.
* Achieve competitive performance with existing state-of-the-art bot detection methods while simplifying feature extraction and reducing computational complexity.
* Enable faster implementation and deployment of bot detection systems in real-world scenarios by eliminating the need for extensive data pre-processing.

**LITERATURE REVIEW OF THE MODELS**

1. DistilBERT (Distilled BERT)

* **Why DistilBERT:**
* **Efficient yet Powerful**: DistilBERT is a distilled version of BERT, designed to retain much of BERT's accuracy while being lighter and faster. It offers an excellent balance between performance and efficiency, making it suitable for tasks like tweet classification, sentiment analysis, and entity recognition in bot detection. Its smaller size makes it a more practical choice for resource-constrained environments, while still leveraging the power of BERT’s bidirectional context understanding.
* **Pros:**
* Retains 97% of BERT’s performance while being 60% faster.
* Smaller model size, reducing memory and computational requirements.
* Faster inference times, making it ideal for real-time applications.
* **Cons:**
* Slight trade-off in accuracy compared to full BERT.
* Still requires considerable resources for training and fine-tuning.

2. DistilRoBERTa (Distilled Robustly Optimized BERT Pre-training Approach)

* **Why DistilRoBERTa:**
  + **Optimized Efficiency and Performance**: DistilRoBERTa is a distilled version of RoBERTa, offering similar advantages in terms of speed and memory efficiency as DistilBERT, while benefiting from RoBERTa's more robust training techniques. It’s ideal when handling large-scale textual data like tweets for tasks such as bot detection, sentiment analysis, or text classification, where both speed and high performance are crucial.
* **Pros:**
  + - Faster and more efficient than RoBERTa, while retaining most of its performance.
    - Removes the next-sentence prediction task, leading to better context comprehension.
    - Smaller model size, resulting in reduced computational and memory demands.
* **Cons:**
  + - Some accuracy trade-off compared to full RoBERTa.
    - Still shares limitations with RoBERTa for handling very long sequences.

3. Bi-LSTM (Bidirectional Long Short-Term Memory)

* + **Why Bi-LSTM:** 
    - This is one of the best models for capturing long-term dependencies in sequential data. It processes information from both directions (past and future), enhancing the model’s ability to retain important sequence information.
  + **Pros:**
    - Excellent for sequential data, handling dependencies across time.
    - Bidirectional architecture ensures that both forward and backward dependencies are captured.
  + **Cons:**
    - Slower to train compared to other neural networks.
    - May struggle with very long sequences, despite solving the vanishing gradient problem.

4. Bi-GRU (Bidirectional Gated Recurrent Unit)

* + **Why Bi-GRU:** 
    - GRUs are simplified versions of LSTMs, making them computationally more efficient while maintaining similar performance. If Bi-LSTM is too slow or memory-consuming, Bi-GRU offers a faster alternative for sequential data, especially for tasks where long-term dependencies are less complex but still important.
  + **Pros:**
    - More efficient and faster than Bi-LSTM.
    - Handles long-term dependencies well.
  + **Cons:**
    - May not capture intricate dependencies as well as Bi-LSTM.

5. XLNet (Generalized Autoregressive Pretraining for Language Understanding)

* + **Why XLNet:**
  + XLNet effectively integrates autoregressive and autoencoding approaches, making it highly suitable for detecting bots in social media interactions. Given the dataset of tweets, XLNet captures bidirectional context, surpassing the limitations of masked language models like BERT. This capability enhances its performance in understanding subtle language patterns.
  + **Pros:**
    - Bidirectional Context: Captures dependencies between words from both directions for better text understanding.
    - Enhanced Performance: Achieves superior results on NLP benchmarks, improving accuracy in bot detection.
    - Flexible Sequence Length: Handles variable-length tweets, adapting to different text inputs.
  + **Cons:**
    - Training Complexity: Complex architecture leads to longer training times.
    - Resource Intensive: Requires significant computational power for training and inference

**Baseline Model**

* **Bidirectional LSTM (Bi-LSTM)**:
  + **Rationale**: The Bi-LSTM is an effective choice for sequence processing as it captures contextual information from both preceding and following words in the text. This bidirectional approach is essential for grasping the intricacies of language, making it particularly useful for tasks such as sentiment analysis and entity recognition, which are vital for detecting bot-like behaviors on social media.
  + **Characteristics**:
    - Processes sequences in both directions, improving context understanding.
    - Handles long-range dependencies effectively.

**Define Working End-to-End Pipeline for Twitter Bot Detection**

The end-to-end pipeline for the Twitter bot detection project is designed to effectively preprocess data, implement various models, and evaluate their performance. This pipeline encompasses data loading, preprocessing, model training, evaluation, and deployment. The project utilizes five distinct models, each with its specific architecture and tokenizer.

**1. Data Loading**

The pipeline begins with the loading of the Cresci 2017 dataset, which contains labeled tweets (bot vs. non-bot). The datasets (traditional\_spambot.csv, genuine.csv) are concatenated into a single csv file (combined\_df.csv) and into a pandas dataframe for further processing.

**2. Data Preprocessing**

Data preprocessing is crucial for preparing the text data for model input. The following steps are applied uniformly across all models:

- **Text Cleaning**: Remove special characters, URLs, and excessive whitespace.

- **Lowercasing**: Convert all text to lowercase to ensure uniformity.

- **Tokenization**: Each model will use its corresponding tokenizer as specified below.

- **Padding and Truncation**: Sequences are padded or truncated to a fixed length to maintain uniform input sizes for the models.

**3. Tokenization**

Each model employs a specific tokenizer suited for its architecture:

- **DistilBERT**: Utilizes the `DistilBertTokenizer` from the Hugging Face Transformers library.

from transformers import DistilBertTokenizer

distilbert\_tokenizer = DistilBertTokenizer.from\_pretrained('distilbert-base-uncased')

- **DistilRoBERTa**: Uses the `DistilRoBertaTokenizer` from the Hugging Face Transformers library.

from transformers import DistilRobertaTokenizer

distilroberta\_tokenizer = DistilRobertaTokenizer.from\_pretrained('distilroberta-base')

- **XLNet**: Implements the `XLNetTokenizer` from the Hugging Face Transformers library.

from transformers import XLNetTokenizer

xlnet\_tokenizer = XLNetTokenizer.from\_pretrained('xlnet-base-cased')

- **Bidirectional LSTM**: Employs a simple Keras tokenizer, as it requires custom preprocessing.

from keras.preprocessing.text import Tokenizer

lstm\_tokenizer = Tokenizer()

lstm\_tokenizer.fit\_on\_texts(data['text'])

- **Bidirectional GRU**: Similar to the Bidirectional LSTM, it uses Keras for tokenization.

gru\_tokenizer = Tokenizer()

gru\_tokenizer.fit\_on\_texts(data['text'])

**4. Model Architectures**

The pipeline incorporates the following model architectures for classification:

- **DistilBERT Model**:

The DistilBERT model is a lightweight version of BERT, designed to be faster while maintaining performance. The architecture consists of multiple transformer layers followed by a dense layer for binary classification.

from transformers import DistilBertForSequenceClassification

distilbert\_model = DistilBertForSequenceClassification.from\_pretrained('distilbert-base-uncased')

**- DistilRoBERTa Model:**

Similar to DistilBERT, the DistilRoBERTa model utilizes a distilled version of RoBERTa, enhancing efficiency without sacrificing accuracy.

from transformers import DistilRobertaForSequenceClassification

distilroberta\_model = DistilRobertaForSequenceClassification.from\_pretrained('distilroberta-base')

**- XLNet Model:**

The XLNet model leverages a permutation-based training method that captures bidirectional context. The architecture involves transformer blocks followed by a linear classification layer.

from transformers import XLNetForSequenceClassification

xlnet\_model = XLNetForSequenceClassification.from\_pretrained('xlnet-base-cased')

**- Bidirectional LSTM Model:**

The Bidirectional LSTM architecture consists of LSTM layers that process the input sequences in both forward and backward directions, followed by dense layers for output.

from keras.models import Sequential

from keras.layers import Embedding, LSTM, Dense

lstm\_model = Sequential()

lstm\_model.add(Embedding(input\_dim=vocab\_size, output\_dim=embedding\_dim))

lstm\_model.add(LSTM(units=128, return\_sequences=True))

lstm\_model.add(LSTM(units=64))

lstm\_model.add(Dense(units=1, activation='sigmoid'))

**- Bidirectional GRU Model:**

The Bidirectional GRU model utilizes GRU layers instead of LSTMs for a similar effect, optimizing for computational efficiency while retaining effectiveness.

from keras.models import Sequential

from keras.layers import Embedding, GRU, Dense

gru\_model = Sequential()

gru\_model.add(Embedding(input\_dim=vocab\_size, output\_dim=embedding\_dim))

gru\_model.add(GRU(units=128, return\_sequences=True))

gru\_model.add(GRU(units=64))

gru\_model.add(Dense(units=1, activation='sigmoid'))

**5. Model Training and Evaluation**

Each model is trained using the same training and validation splits (80-20) of the dataset. The training involves specifying loss functions such as BinaryCrossEntropy, Adam optimizer , and metrics to evaluate model performance, such as accuracy and F1 score.

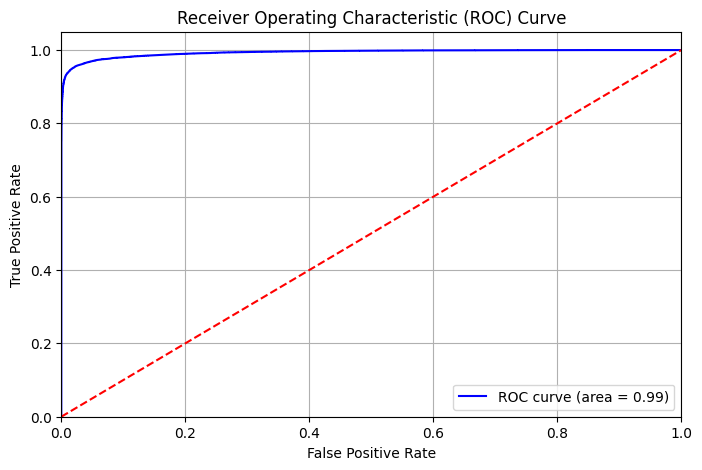
**6. Deployment**

The Twitter bot detection model was deployed using a Streamlit application, which allows users to input text for real-time predictions. The app preprocesses the input using the same tokenizer applied during training, ensuring consistent data formatting. When the user clicks "Predict," the model, loaded from a saved .h5 file, generates predictions and displays the confidence level of whether the user is likely a bot. This deployment approach provides an accessible interface for users to utilize the model's capabilities effectively.

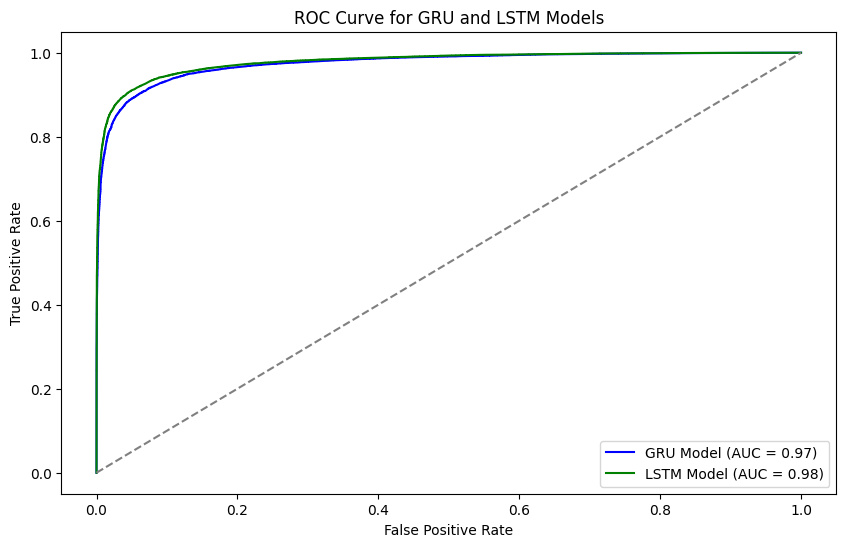
**7. Tabulation and Visualization of Results in Terms of Performance Metrics**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Bi-Directional LSTM | Bi -Directional GRU | DistilBERT | DistilRoBERTa | XLNet |
| Training Accuracy | **0.9202** | **0.9152** | **0.9818** | **0.9780** | **0.4996** |
| Testing Accuracy | **0.9222** | **0.9305** | **0.9636** | **0.9634** | **0.5000** |
| Precision | **0.9472** | **0.9497** | **0.9857** | **0.9697** |  |
| Recall | **0.8902** | **0.9084** | **0.9440** | **0.9574** |  |
| F1 | **0.9178** | **0.9286** | **0.9644** | **0.9635** |  |

**ROC CURVES:**

**DistilBERT**:

**LSTM, GRU:**



**DistilRoBERTa:**

