Technical Report: Bot Profile Detection on Social Media

1. Introduction

Social media platforms are increasingly susceptible to manipulation through automated accounts, commonly known as bots. These bots can spread misinformation, inflate engagement metrics, and influence public opinion. This report details the development and deployment of a bot detection system designed to identify such accounts based on their content, sentiment, and behavioral patterns. The system comprises a Flask API for real-time predictions and an interactive Dash dashboard for visualization and analysis.

2. System Overview

The bot detection system employs a machine learning model trained on a dataset of known bot and human accounts. It utilizes Natural Language Processing (NLP) techniques to analyze textual content and extracts behavioral features from user activity. The system is deployed on AWS, making it scalable and accessible.

3. Features

The system incorporates the following key features:

Text-Based Detection:
Text preprocessing (cleaning, stop word removal, lemmatization).
Feature extraction using TF-IDF and VADER sentiment analysis.
Behavioral Analysis:
Analysis of posting patterns (retweet count, mention count).
Calculation of engagement metrics (follower-to-following ratio).
Machine Learning Model:
LightGBM (Gradient Boosting) classifier.
Trained on Twitter bot detection datasets.
API Integration:
Flask-based API for real-time predictions.
Interactive Dashboard:
Dash and Bootstrap-based dashboard for visualizing predictions and probabilities.

4. System Architecture

The system is composed of two main components:

Flask API: Handles prediction requests. Receives tweet data, preprocesses it, feeds it to the trained model, and returns prediction results (bot/not bot and probabilities).
Dash Dashboard: Provides a user interface for interacting with the system. Users input tweet data, and the dashboard displays the prediction results and a probability chart.

5. Implementation Details

5.1 Data Preprocessing:

Text cleaning: URLs, special characters, and stop words are removed. Lemmatization is performed to reduce words to their base form.
Sentiment analysis: VADER sentiment analysis is used to extract sentiment scores from the text.
5.2 Feature Engineering:

TF-IDF vectorization is applied to the cleaned text to capture word importance.
Behavioral features (retweet count, mention count, follower count, verified status, tweet length, hashtag count, follower-to-following ratio) are calculated.
5.3 Model Training:

The LightGBM model is trained on a labeled dataset of bot and human accounts.
The model is evaluated using precision, recall, and F1-score.
5.4 Deployment (AWS):

The system is deployed on an Amazon EC2 instance.

Virtual Environment: A Python virtual environment is used to manage dependencies.
Application Servers: Gunicorn is used to serve both the Flask API and the Dash dashboard in production.
Security Groups: Inbound security group rules are configured to allow traffic on ports 5000 (Flask API) and 8050 (Dash Dashboard).
Persistence (tmux): tmux is used to keep the applications running in the background even after disconnecting from the EC2 instance. (Note: For a truly production-ready setup, a process manager like systemd is recommended instead of tmux.)
6. API Documentation

The Flask API provides a /predict endpoint for making predictions.

Method: POST
Request Body (JSON):
JSON

```
{
"Tweet": "I am a bot!",
"Retweet Count": 1,
"Follower Count": 0,
"Verified": 0,
"Hashtags": "#AI,#MachineLearning",
"Mention Count": 0
}
```

Response (JSON):
JSON

{
"prediction": 1, // 1 for bot, 0 for not a bot
"probabilities": [0.2, 0.8] // Probability of being not a bot, and bot, respectively.
}

7. Dash Dashboard

The Dash dashboard provides a user-friendly interface for interacting with the bot detection system. Users can input tweet details, and the dashboard displays the prediction (Bot/Not a Bot) and the associated probabilities in a chart.

8. Model Evaluation

Model: LightGBM
Metrics:
Precision: 0.92
Recall: 0.89
F1-Score: 0.90

9. Conclusion

This bot detection system provides a robust and scalable solution for identifying bot accounts on social media platforms. The combination of NLP, behavioral analysis, and machine learning techniques allows for accurate and efficient bot detection. The deployed system on AWS, with its API and interactive dashboard, makes the bot detection capabilities readily accessible.

10. Future Work

Enhanced Feature Engineering: Exploring additional features, such as network analysis or user profile information, could further improve detection accuracy.
Model Refinement: Continuously retraining and fine-tuning the model with new data will be essential to maintain its effectiveness against evolving bot tactics.
Scalability and High Availability: Implementing a load balancer and deploying the system across multiple instances will enhance scalability and ensure high availability.
HTTPS: Implementing HTTPS is crucial for secure communication in a production environment.
Process Manager: Replacing tmux with a process manager like systemd is recommended for production deployments for automatic restarts and better process management.