Bot Detection Under Adversarial Attacks - Assignment Report

Student: saly Bahgat

Student ID: 2205190

Executive Summary

This project implements and analyzes a graph-based bot detection system on social networks. The system is tested against two adversarial attacks to evaluate its robustness and vulnerabilities.

1️⃣ Introduction

The project builds a bot detection model using graph structural features from the Facebook social network. We evaluate the model's resilience against:

1. **Structural Evasion Attack** - Local connection changes
2. **Graph Poisoning Attack** - Global connection poisoning

2️⃣ Dataset Information

- **Source**: Stanford SNAP Facebook Social Network
- **Nodes**: 4,039 users
- **Edges**: 88,234 friendships
- **Graph Density**: 0.01082

3️⃣ Implemented Steps

☑️ 3.1 Graph Construction & Metrics

```
# Load Facebook dataset
# Build undirected graph with 4,039 nodes
# Calculate essential metrics:
# - Average degree: 43.69
# - Maximum degree: 1045
# - Clustering coefficient: 0.6055
```

```
# - Communities detected: 16
```

☑ 3.2 Baseline Bot Detection Model

- **Features extracted**: 6 graph-based features per node
- **Classifier**: Random Forest (100 estimators)
- **Training/Test split**: 70/30
- **Synthetic labels**: 5% bots (201 nodes), 95% humans

Baseline Performance:

```
Accuracy: 94.88%
Bot Precision: 33%
Bot Recall: 3%
Bot F1-score: 6%
```

Interpretation: High accuracy but poor bot detection (low recall) due to class imbalance.

☑ 3.3 Adversarial Attacks Implementation

Attack 1: Structural Evasion

```
# Strategy: Remove edges from bot nodes
# Target: Low-degree bot nodes
# Goal: Reduce bot detectability
# Result: Bot degree reduced by ~30%
```

Attack 2: Graph Poisoning

```
# Strategy: Add edges between bots and humans
# Target: Random human nodes
# Goal: Corrupt training data
# Result: Graph density increased by 4.9%
```

☑ 3.4 Post-Attack Evaluation

- Recalculated all graph features after each attack
- Retrained Random Forest classifier
- Tested on clean data to measure poisoning impact
- Compared performance across three scenarios

☑ 3.5 Visualization

- Generated graph visualizations for each scenario
- Color-coded nodes (red=bots, blue=humans)
- Saved as PNG files:
  - `baseline_graph.png`
  - `evasion_graph.png`
  - `poisoning_graph.png`

4 Results Comparison

Metric	Baseline	After Evasion	After Poisoning
Accuracy	94.88%	92.15%	90.42%
Bot Precision	33%	28%	22%
Bot Recall	3%	2%	2%
Bot F1-score	6%	4%	3%

Key Observations:

1. Both attacks successfully degrade detection performance
2. Poisoning attack has stronger impact on precision
3. Evasion attack better at hiding individual bots
4. Overall accuracy drops but remains high due to class imbalance

Performance Drop:

- Accuracy decreased by **2.73%** (evasion) and **4.46%** (poisoning)
- Bot precision dropped by **5.55%** (evasion) and **11.11%** (poisoning)
- Bot recall remained very low (~1.67% for both attacks)

5️⃣ Attack Impact Analysis

Structural Evasion Impact:

- **Bot degree reduced by 30%**
- Clustering coefficient decreased
- Bots become less conspicuous
- **Recall drops more than precision**

Graph Poisoning Impact:

- **Graph density increased by 4.9%**
- Community boundaries blurred
- **Precision drops significantly**
- Training data corrupted

Visual Changes:

- Baseline: Clear community structures
- After evasion: Bots have fewer connections
- After poisoning: More cross-community edges

6️⃣ Conclusions

Key Findings:

1. ✅ **Graph-based features** are effective for bot detection
2. ⚠️ **Detection systems are vulnerable** to adversarial attacks
3. 🔴 **Poisoning attacks** are more damaging to overall performance
4. 🟡 **Evasion attacks** better at hiding specific bots
5. 📊 **Feature importance shifts** under attack

Practical Implications:

1. **For defenders**: Need robust features and regular model updates

2. **For detection systems**: Require multiple feature types and temporal analysis
3. **For social networks**: Must monitor for coordinated connection changes

7 Recommendations:

1. Implement **adversarial training** with attack simulations
2. Use **ensemble methods** with multiple graph metrics
3. Add **temporal analysis** to detect gradual evasion
4. Monitor **feature distribution changes** for attack detection

Conclusion Summary

The Good:

- Graph-based detection can achieve **94.88% accuracy**
- Simple features work reasonably well in normal conditions
- Visualization helps understand network structure

The Bad:

- **Very poor bot recall** (only 3% bots detected)
- **Highly vulnerable** to adversarial attacks
- **Class imbalance** masks real performance issues

The Solution:

1. **Don't rely only on graph features** - add behavioral analysis
2. **Implement defense mechanisms** - adversarial training, robust features
3. **Continuous monitoring** - detect and adapt to new attack patterns
4. **Human oversight** - maintain human review for uncertain cases

8 Technical Details

Features Used:

1. Degree centrality

2. Clustering coefficient
3. Betweenness centrality
4. Community membership
5. Average neighbor degree
6. Same-community ratio

Attack Parameters:

- **Evasion**: Remove 30% of bot-human edges
- **Poisoning**: Add 1-3 edges to 15% of human nodes

Evaluation Method:

- 5-fold cross-validation
- McNemar's test for significance
- Feature importance analysis