The **bot-hunter Framework**  
A Multi-model Toolbox for Twitter Bot Detection

**LTC David M. Beskow**  
deskow@andrew.cmu.edu

**Prof. Kathleen M. Carley**  
kathleen.carley@cs.cmu.edu

---

**Introduction**

Although efforts have long existed to detect and characterize automated agents in social media, these have recently received emphasis following sophisticated deployment of bots by state and non-state actors in an effort to influence global events and decisions. To support this growing need, the bot-hunter framework creates a multi-model toolbox that allows tailored performance for specific applications.

**Phase 1:** Build a bot-hunter toolbox of algorithms that identifies automated actors and networks at either high volume or high accuracy settings, facilitating the study and characterization of these networks.

**Phase 2:** Transition from binary to multi-class prediction

---

**Understanding Tier of Available Data**

<table>
<thead>
<tr>
<th>Tier</th>
<th>Description</th>
<th>Focus</th>
<th>Collection Time per 250 Accounts</th>
<th># of Data entities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tier 0</td>
<td>Single entity (text or screen name)</td>
<td>Semantics</td>
<td>N/A**</td>
<td>1</td>
</tr>
<tr>
<td>Tier 1</td>
<td>Profile + 1 Tweet</td>
<td>Account Meta-data</td>
<td>~ 5 Seconds</td>
<td>2</td>
</tr>
<tr>
<td>Tier 2</td>
<td>Profile + Timeline</td>
<td>Temporal patterns</td>
<td>~ 5 minutes</td>
<td>200+</td>
</tr>
<tr>
<td>Tier 3</td>
<td>Profile + Timeline + Friends Timeline</td>
<td>Network patterns</td>
<td>~ 24 hours</td>
<td>50,000+</td>
</tr>
</tbody>
</table>

Tier 0, or Single entity (text or screen name), focuses on semantics such as name. Tier 1 includes 1 Tweet and focuses on Account Meta-data such as Account Meta-data. Tier 2 includes Profile + Timeline and focuses on Temporal patterns, allowing researchers to choose models designed for high volume or high accuracy settings, facilitating the study and characterization of these networks.

The Twitter API allows researchers to return to the API for various data related to a given account, slowly building a more comprehensive picture of the account and the communication network(s) that it participates in. These returns can be computationally expensive. The bot-hunter framework develops a toolbox of models that are tailored for specific Tiers of data, allowing researchers to choose models designed for high volume (characterize bot involvement in large streams) or high accuracy (analyse a few suspicious accounts).

**Cumulative Features by Tier**

- Tier 3: profiles, features, and links
- Tier 2: features, links, and dependencies
- Tier 1: features, and dependencies
- Tier 0: features, semantics

**Annotated Bot Data**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th># of Records</th>
</tr>
</thead>
<tbody>
<tr>
<td>NATO Data</td>
<td>Event-related data annotation. These accounts conducted targeted manipulation attack against NATO and DDR.</td>
<td>19,221</td>
</tr>
<tr>
<td>Lực de Augs 2017</td>
<td>These were annotated with a Tier 1 Model designed to identify 15 digit random alphanumeric strings highly correlated to bot accounts.</td>
<td>262,097</td>
</tr>
<tr>
<td>Caverlee Data</td>
<td>Rescrape of original honey pot dataset collected by Texas A&amp;M (Caverlee et al, 2011).</td>
<td>15,742</td>
</tr>
</tbody>
</table>

---

**Current bot-hunter Prototype Dashboards**

- **Analyze Forests** with Tier 1 Model
- **Analyze Trees** with Tier 2 Model

---

**Results**

Figure 1: Results by training data (Caverlee, Nato, Random and Combined data sets) and by Tier

**Conclusions**

- Tier 0 models facilitate some data annotation tasks
- Tier 1 data provides high accuracy even with reduced feature space
- Tier 2 and Tier 3 provide incrementally better performance
- Relevant training data is critical: out of sample performance low
- Models appear to get traction on Tier 3 features, which are less susceptible to pumped-master manipulation

---

**Methods**

We built a botHunter Python package that wraps the scikit-learn machine learning package, and allows users and applications to classify Twitter accounts at Tier 0 (modeled character n-grams of screen names), Tier 1, Tier 2, and Tier 3. We also designed a netMetrics Python package that wraps the networkx and tweepy packages and facilitates building graph-based feature space at scale. We tested Naïve Bayes, Logistic Regression, SVM, Decision Tree, and Random Forest Models with 10 Fold Cross Validation. Logistic regression had the highest performance at Tier 0, and Random Forest had the highest performance on Tiers 1-3. Respective tuned models were developed for prototype applications.