DisinfoBotWatch

Analysis and visualization of Russian Internet Research Agency (IRA) troll bot networks using Apache Spark and Flask.

Technologies used

- **Python**: It’s easy to use, but most importantly, it’s the standard in data analysis because it’s overall the best language for this kind of task.
- **uv**: An extremely fast Python package and project manager, written in Rust, it’s slowly becoming the new standard.
- **Spark**: For this kind of data analysis, we don’t really need realtime streaming, which is why spark, a technology that’s using batch processing, is perfectly fit for the job. We don’t need either to go very low level, so there’s really no need to go with MapReduce, it would be a lot more work for results that don’t really differ.
- **Flask**: Lightweight web framework for REST API
- **NetworkX**: Graph analysis and network construction
- **Pandas**: Data manipulation and analysis

Frontend Dashboard

- **Konsta UI**: Mobile-first UI components with iOS design
- **React 19**: Modern frontend framework with TypeScript
- **Vite**: Ultra-fast build tool and dev server
- **Tailwind CSS v4**: Utility-first CSS framework
- **Recharts**: Composable charting library for React
- **Axios**: HTTP client for API communication

Requirements

System Requirements

- **Java**: JDK 17 (Spark 3.3.2 has compatibility issues with Java 25+)
- **Python**: 3.12 or higher

Software Dependencies

The project automatically installs all Python dependencies through `uv`. Key packages include:

- **pyspark**: 3.3.2 - Distributed data processing
- **flask**: Web framework for dashboard API
- **pandas**: Data manipulation and analysis
- **networkx**: Graph analysis and network construction
- **pyvis**: Interactive network visualization
- **numpy**: Numerical computations
- **matplotlib**: Statistical visualization

Installation

Step 1: Install Java 17

DisinfoBotWatch requires Java 17. Spark 3.3.2 has compatibility issues with later versions.

On macOS:

```
brew install openjdk@17
```

On Ubuntu/Debian:

```
# Choose appropriate version here: https://www.oracle.com/java/technologies/javase/jdk17-archive-downloads.html
# For example, with .deb
sudo apt install ./jdk-17.0.12_linux-x64_bin.deb
```

Verify installation:

```
java -version
```

Step 2: Install uv

`uv` is a fast Python package manager written in Rust. It automatically manages virtual environments and dependencies.

On macOS:

```
brew install uv
```

On Linux:

```
curl -LsSf https://astral.sh/uv/install.sh | sh
```

With pip:

```
pip install uv
```

Or download from: https://github.com/astral-sh/uv/releases

Verify installation:

```
uv --version
```

Data Setup

Obtaining the Dataset

The project uses Twitter bot data from the Russian Internet Research Agency (IRA).

To download the dataset:

```
cd data
# Edit dl_data.sh and uncomment the files you want to download

# Make the script executable and run it
chmod +x dl_data.sh
./dl_data.sh
# Then go back to previous path
cd ..
```

The script will download CSV files containing tweet data and metadata from https://github.com/fivethirtyeight/russian-troll-tweets

Running the Project

Option 1: Run Analysis Pipeline

Run the complete data analysis and generate outputs:

```
./run_analysis.sh
```

This script automatically:

1. Runs the complete Spark pipeline with `uv run python main.py`
2. Generates analysis outputs in the `outputs/` directory
3. Creates the network visualization

This generates:

- `outputs/top_active_accounts.csv` - Most active bot accounts
- `outputs/account_distribution.csv` - Bot classification statistics
- `outputs/network.html` - Interactive network graph visualization

Option 2: Launch Dashboard

Start the dashboard with the script:

```
./run_dashboard.sh
```

This script automatically:

1. Checks if analysis data exists (warn you otherwise)
2. Installs dashboard dependencies if needed
3. Starts Flask API backend on `http://localhost:5000`
4. Starts Konsta UI frontend on `http://localhost:5173`

Dashboard URLs:

- Frontend UI: `http://localhost:5173`
- Backend API: `http://localhost:5000`

Press `Ctrl+C` to stop both services.

Option 3: Run Components Separately

Analysis only:

```
uv run python main.py
```

Backend API only:

```
uv run python api.py
```

Frontend dashboard only:

```
cd konsta && npm run dev
```

Dashboard Features

The dashboard is built with **Konsta UI**

Overview Tab

- **Statistics Cards**: Total bots, tweets, unique accounts, and average tweets per bot
- **Dataset Information**: About the Russian IRA bot dataset
- **Animated counters** some fancy animations

Top Bots Tab

- **Interactive bar chart** showing most active bot accounts
- **Filterable table** with adjustable number
- **Account details**: Type, category, and tweet count

Distribution Tab

- **Pie charts** for account type and category distribution
- **Top 10 categories** displayed
- **Interactive tooltips** with percentages

Network Tab

- **Interactive network visualization** embedded from analysis
- Node size proportional to network degree
- Edge weights representing interaction strength
- Drag to pan, scroll to zoom
- Hover for account information

Analysis Details

Metrics Calculated

1. **Basic Statistics**
   - Total tweet count
   - Unique authors and accounts
   - Distribution by type, category, region, and language
2. **Account Activity**
   - Tweet count per account
   - Retweet vs original tweet ratios
   - Content length analysis
3. **Coordinated Behavior**
   - Detection of identical content posted by multiple accounts
   - Identification of posting patterns indicating coordination
4. **Network Analysis**
   - Mention network construction
   - Degree centrality ranking
   - Betweenness centrality analysis
   - Network clustering coefficient
   - Network density measurement

Troubleshooting

Issue: “Spark didn’t work with my java version”

Solution: Install Java 17 as specified in the installation section. Verify with:

```
java -version
```

Issue: Network graph not displaying

Solution:

1. Ensure analysis has completed (check `outputs/network.html` exists)
2. The file should be ~1MB in size
3. Clear browser cache and reload

Issue: Dashboard is slow or unresponsive

Solution:

- Large network graphs are loaded lazily (only when “Bot Network Graph” tab is clicked)
- Dashboard limits data display to top 100 accounts and 50 distribution items
- Close unnecessary browser tabs
- Ensure sufficient system RAM

References

The data is sourced from FiveThirtyEight’s publicly released IRA troll tweets dataset.

- Apache Spark Documentation
- Flask Documentation
- NetworkX Documentation
- FiveThirtyEight IRA Data
- Konsta
- Npm
- debunk.org
- The Conversation

Valentin RAPP Cédric MARTZ