Harnessing the Nature of Spam in Scalable Online Social Spam Detection

Hailu Xu, Boyuan Guan, Pinchao Liu, William Escudero, Liting Hu

Florida International University
Outline

- Background
  - Background & Problem
  - Challenges

- Sifter Design
  - Sifter Workflow
  - Implementation Details
  - Design Benefits

- Preliminary Evaluation
  - Spam Detection Evaluation
  - System Evaluation

- Conclusion & Future Works
Background - Social Spam

**Twitter**
- 450,000 suspicious log-ins per day
- 5.7 million spam tweets per week
- 20 million fake Twitter users

**Facebook**
- 4% posts are spam
- Affect over 4 million users every day
- 83 million fake Facebook profiles
Background - Traditional Spam Detection

Static Feature Set + Offline Detection

specific dataset  feature set  analysis  results
Background - Traditional Spam Detection

Static Feature Set

Features from a single dataset

specific dataset  feature set  analysis  results
Limit #1
Lost extensibility

(the static feature set bases on data from a fixed dataset without extensibility to online data from new topics and new sources)
Background - Traditional Spam Detection

Offline Detection

Long latency

specific dataset  feature set  analysis  results
Limit #2
Long latency

(traditional offline processing costs days or months in feature engineering, cannot effectively handle online social spam detection)
How to detect online streaming data in relatively low latency and without labor-intensive feature engineering?
Outline

- **Background**
  - Background & Problem
  - Challenges

- **Sifter Design**
  - Sifter Workflow
  - Implementation Details
  - Design Benefits

- **Preliminary Evaluation**
  - Spam Detection Evaluation
  - System Evaluation

- **Conclusion & Future Works**
Sifter Workflow

- **Workflow of Sifter**
  - Online data collection, Sifter processing, Spam classification and data collection, Update RNN with new social spam

1. Online data collection
   - Online data is collected from web servers

2. Sifter implementation
   - Sifter Groups
     - Hash('video')
     - group('video')
     - group('link')
     - group('pic')
     - Sifter Groups
     - Hash('link')
     - Hash('video')
     - Hash('pic')

3. Spam detection with RNN

4. Online spam gather

5. Update RNN with new spam

new identified spam
Online social data group

- Social data groups
  - Each node maintains a topic table; Sifter group is created by the topic table; node who collects the data from the specific topic can join the topic-based group.
Sifter Implementation

- Sifter functional components
  - Sifter root, Sifter leaf agent, Sifter DHT-based tree
Spam Detection Unit (SDU)

- **Spam Detection Unit (SDU)**
  - Leaf node implements RNN-based model, automatically extract spam features

- **Spam Collection**
  - DHT-based aggregation tree, Middle layer results aggregation, Layer-by-layer rolls up
Model Update

- Update RNN with new spam
  - Root updates the RNN model, Leaf agent updates classifiers

Update weights and biases

```
[w...w] [b]
[w...w] [b]
```

```
[w...w] [b]
[w...w] [b]
[w...w] [b]
```

```
[w...w] [b]
[w...w] [b]
[w...w] [b]
```

```
[w...w] [b]
[w...w] [b]
```
Design Benefits

- **Utilize the newest spam info:**
  - collect the *online* social data
  - detect the social spam in various *groups*
  - RNN model can be promptly updated
  - keep pace with the *latest* social spam

- **Support scalable spam processing:**
  - handle multiple *distributed* servers’ logs
  - easily scale to data sources from more servers
Outline

- **Background**
  - Background & Problem
  - Challenges

- **Sifter Design**
  - Sifter Workflow
  - Implementation Details
  - Design Benefits

- **Preliminary Evaluation**
  - Spam Detection Evaluation
  - System Evaluation

- **Conclusion & Future Works**
Evaluation

Testbed setups

<table>
<thead>
<tr>
<th>Hardware/Software</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Servers</td>
<td>20</td>
</tr>
<tr>
<td>CPU</td>
<td>3.4GHz</td>
</tr>
<tr>
<td>Memory</td>
<td>4 GB</td>
</tr>
<tr>
<td>Disk</td>
<td>30 GB</td>
</tr>
<tr>
<td>Operating system</td>
<td>Ubuntu 16.04</td>
</tr>
<tr>
<td>Language</td>
<td>Java SE 1.7</td>
</tr>
<tr>
<td>Twitter</td>
<td>3,000,000 tweets from API</td>
</tr>
</tbody>
</table>
A sample dataset which consists of 50,000 posts (37,465 posts are Ham and 12,535 posts are Spam)

### TABLE I. RESULTS OF SPAM CLASSIFICATION.

<table>
<thead>
<tr>
<th>Data blocks</th>
<th>F1</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>data interval 1 (10min)</td>
<td>82.0%</td>
<td>0.903</td>
<td>0.751</td>
</tr>
<tr>
<td>data interval 2 (20min)</td>
<td>84.7%</td>
<td>0.912</td>
<td>0.791</td>
</tr>
<tr>
<td>data interval 3 (30min)</td>
<td>89.6%</td>
<td>0.923</td>
<td>0.87</td>
</tr>
</tbody>
</table>
System Evaluation

The linear increment of time is determined by the tree depth \( O(\log N) \), indicates that tree structure exhibits a good balance.
Outline

- Background
  - Background & Problem
  - Challenges

- Sifter Design
  - Sifter Workflow
  - Implementation Details
  - Design Benefits

- Preliminary Evaluation
  - Spam Detection Evaluation
  - System Evaluation

- Conclusion & Future Works
Conclusion & Future Work

Sifter introduces an ‘extensible’ and ‘online’ social spam detection system:

- **extensible**: processing can extend to new data without labor-intensive feature engineering.
- **online**: achieve spam detection with new updated spam.

**Future work:**

- explore the entire processing latency
- balance the scale and the latency of distributed agents in the system
- reduce the runtime overhead
- achieve load-balancing with highly efficient data processing
Thank you.

Questions?