Harnessing the Nature of Spam in Scalable Online Social Spam Detection

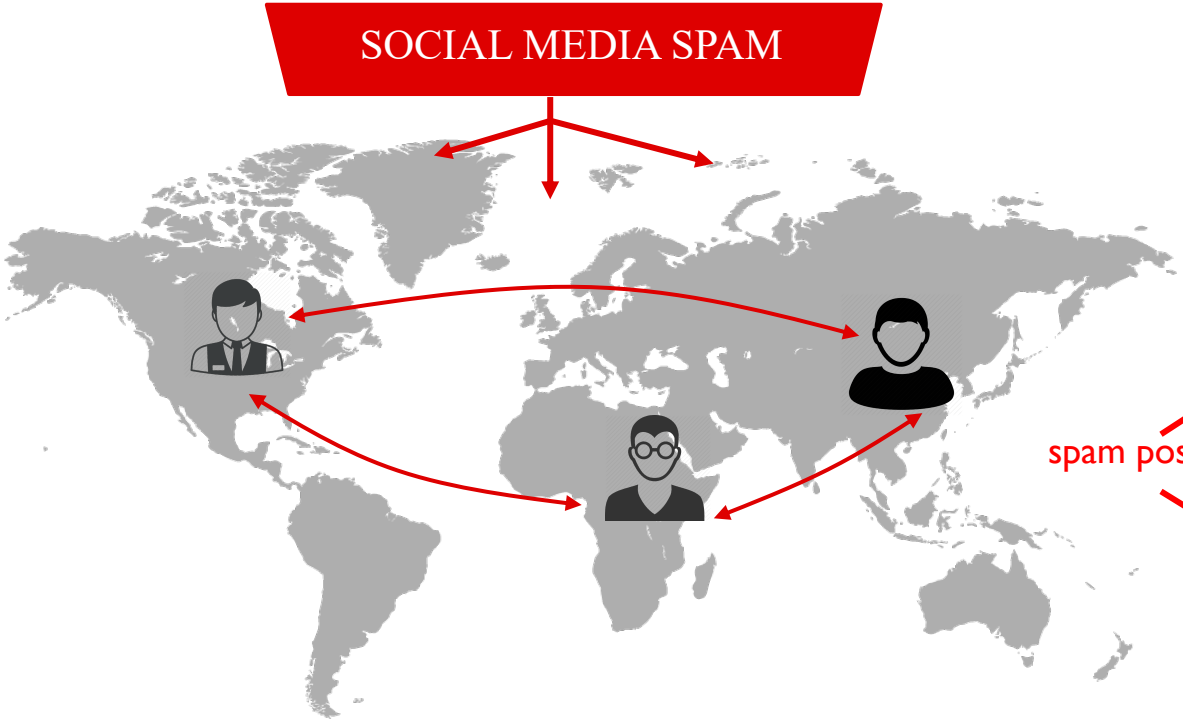
Hailu Xu, Boyuan Guan, Pinchao Liu,
William Escudero, Liting Hu

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

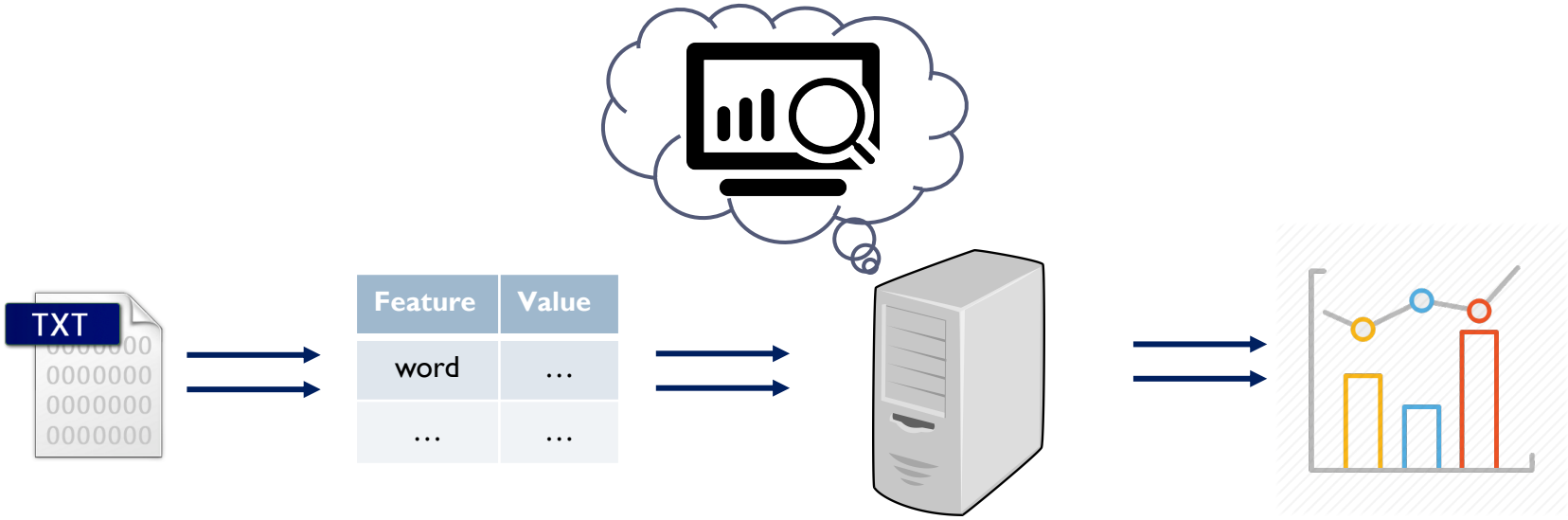
- 450,000 suspicious log-ins per day
- 5.7 million spam tweets per week
- 20 million fake Twitter users

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

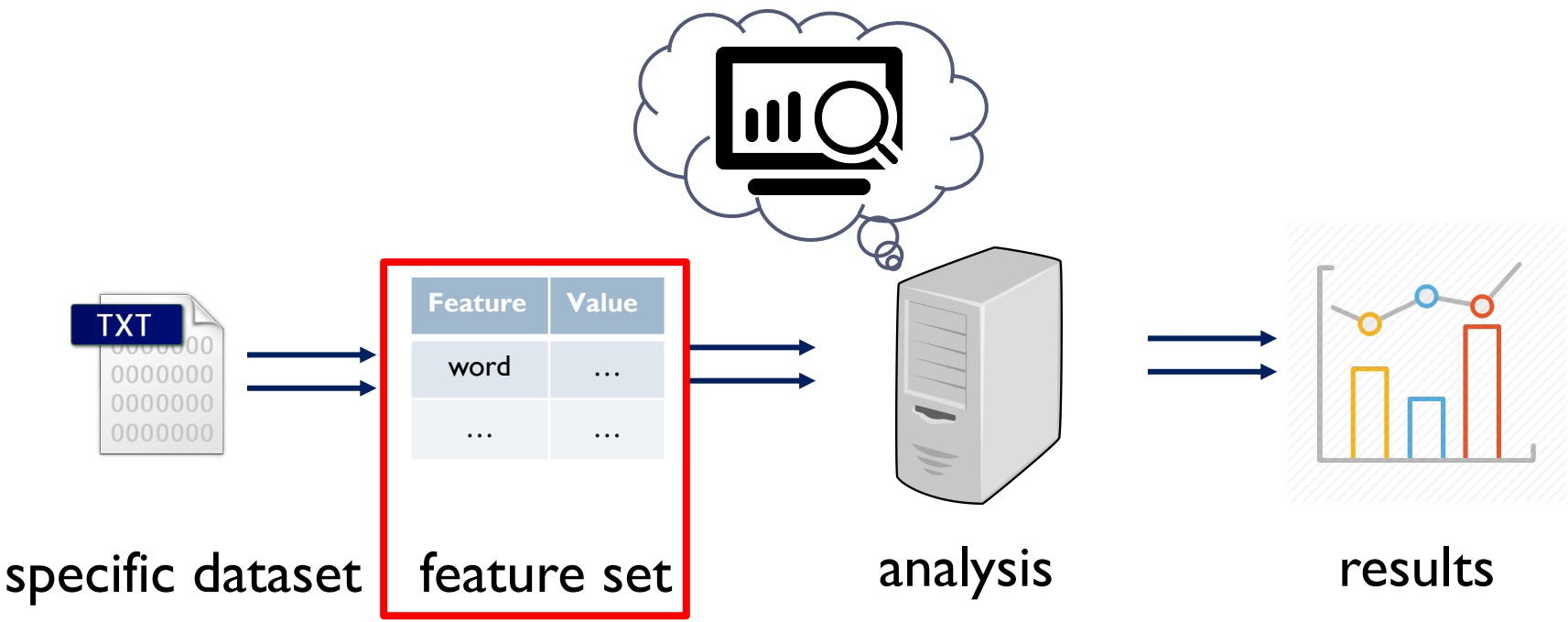


specific dataset feature set analysis results

Background - Traditional Spam Detection

Static Feature Set

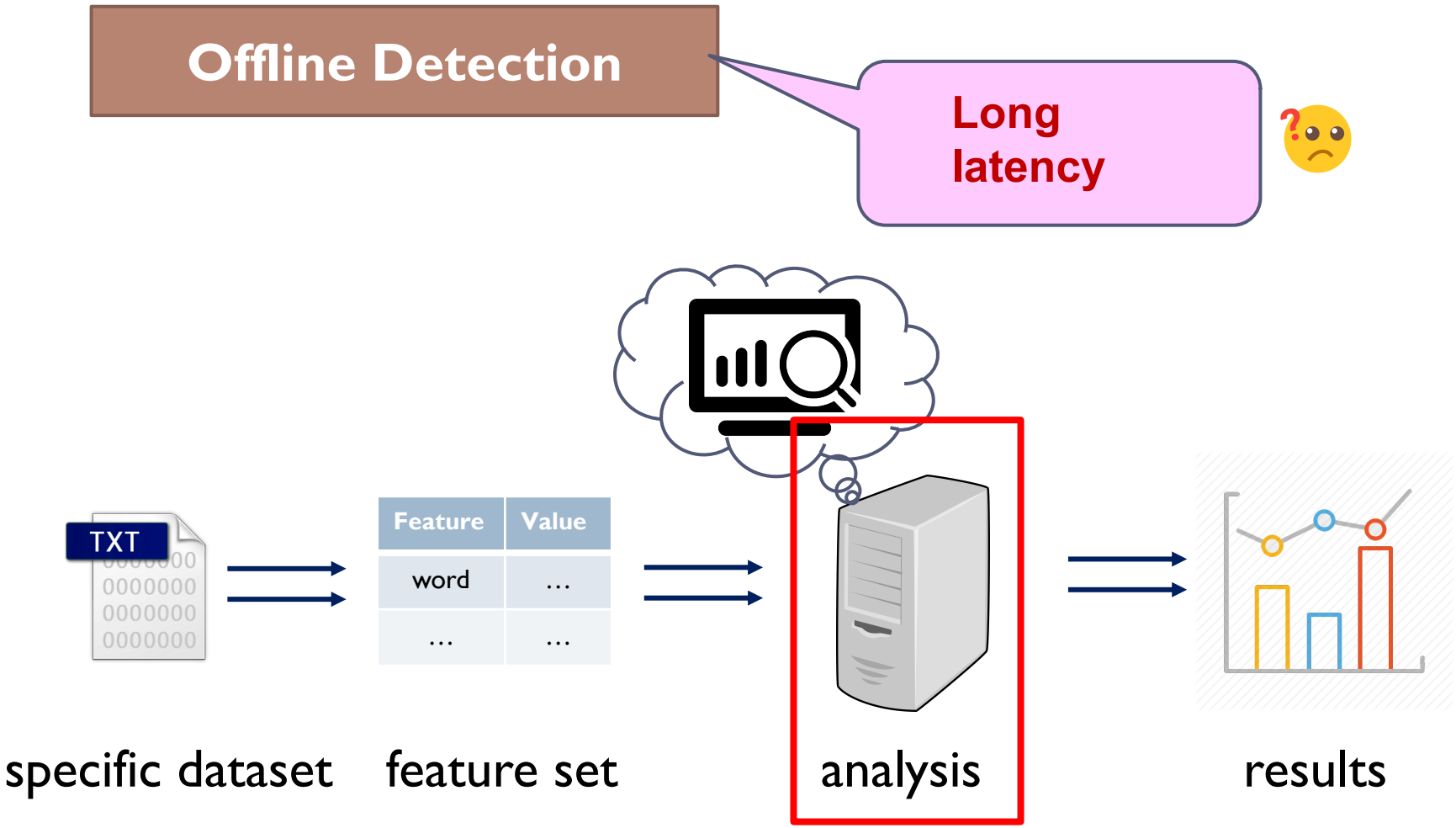
Features from a single dataset



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

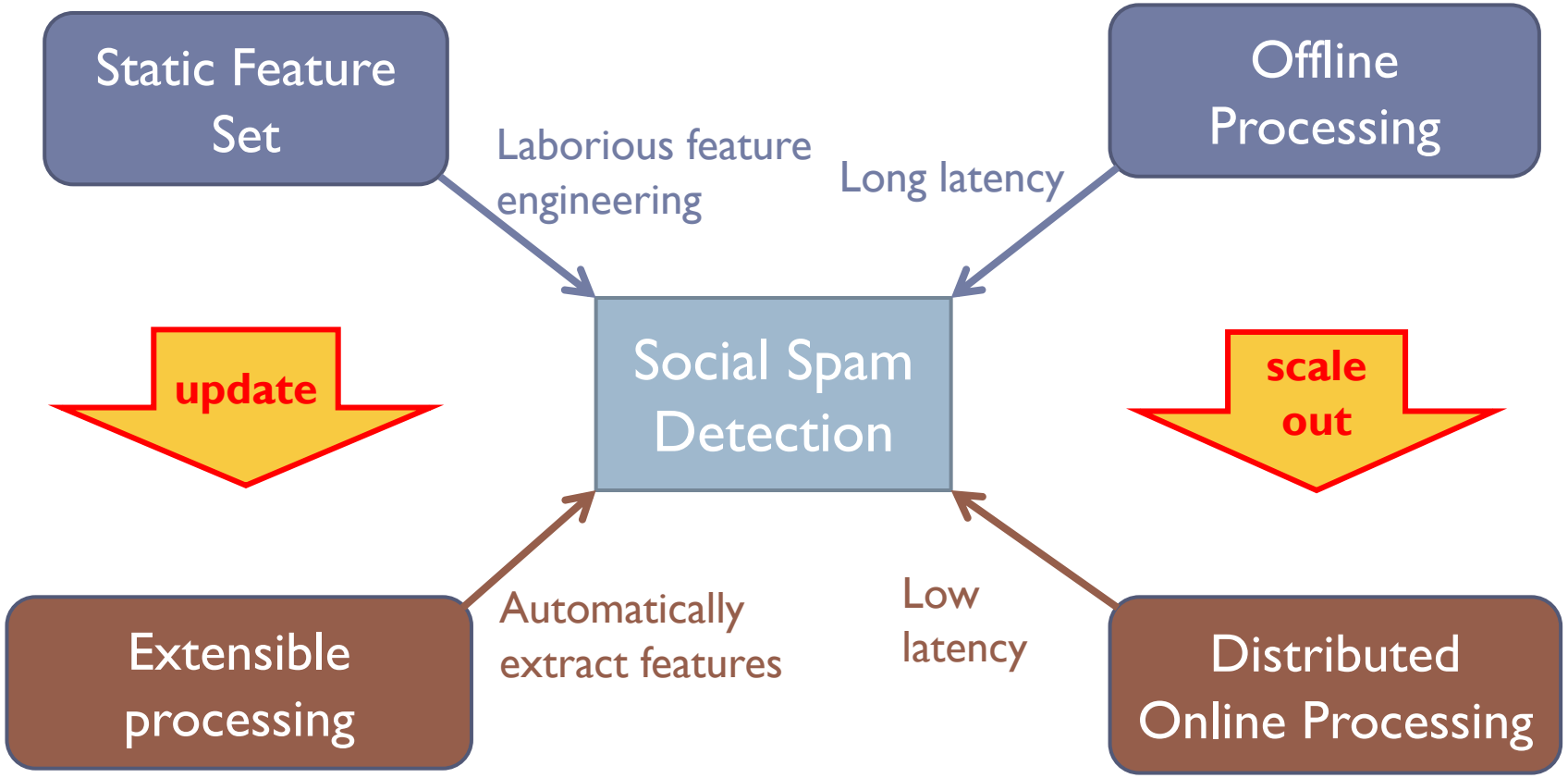


Limit #2
Long latency

(traditional offline processing costs days or months in feature engineering, cannot effectively handle online social spam detection)

Challenges

How to detect online streaming data in relatively low latency and without labor-intensive feature engineering?

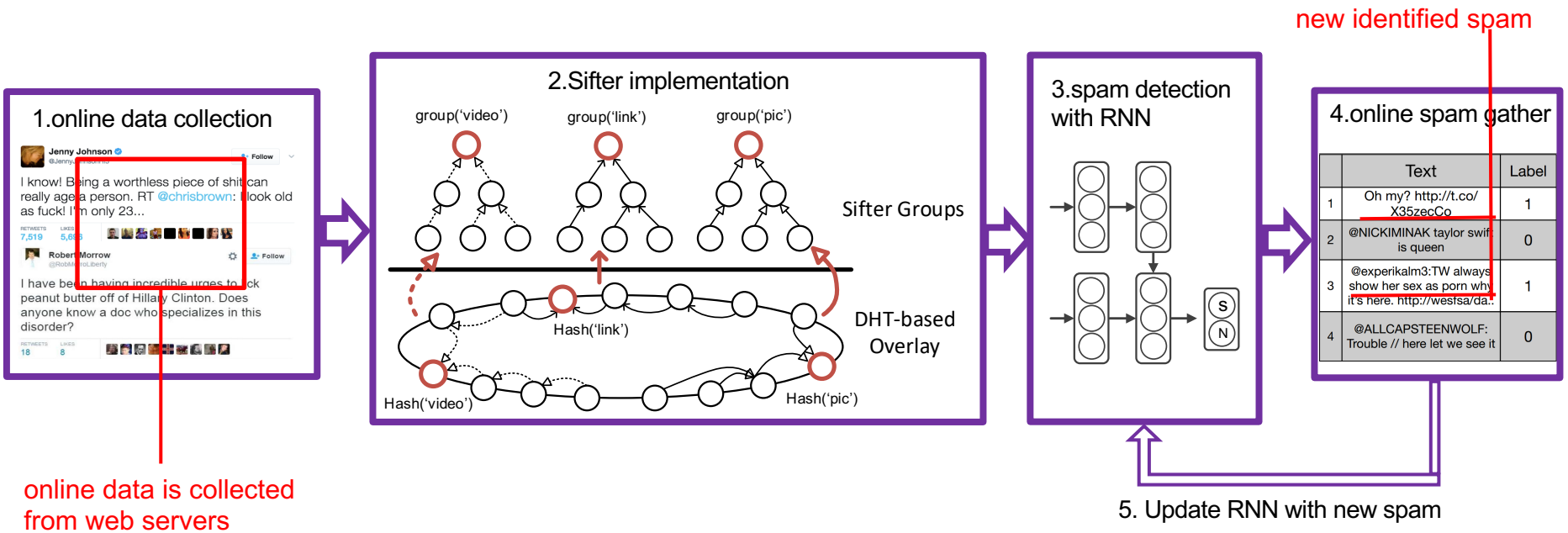


Outline

- Background
  - Background & Problem
  - Challenges
- Sifter Design
  - Sifter Workflow
  - Implementation Details
  - Design Benefits
- Preliminary Evaluation
  - Spam Detection Evaluation
  - System Evaluation
- Conclusion & Future Works

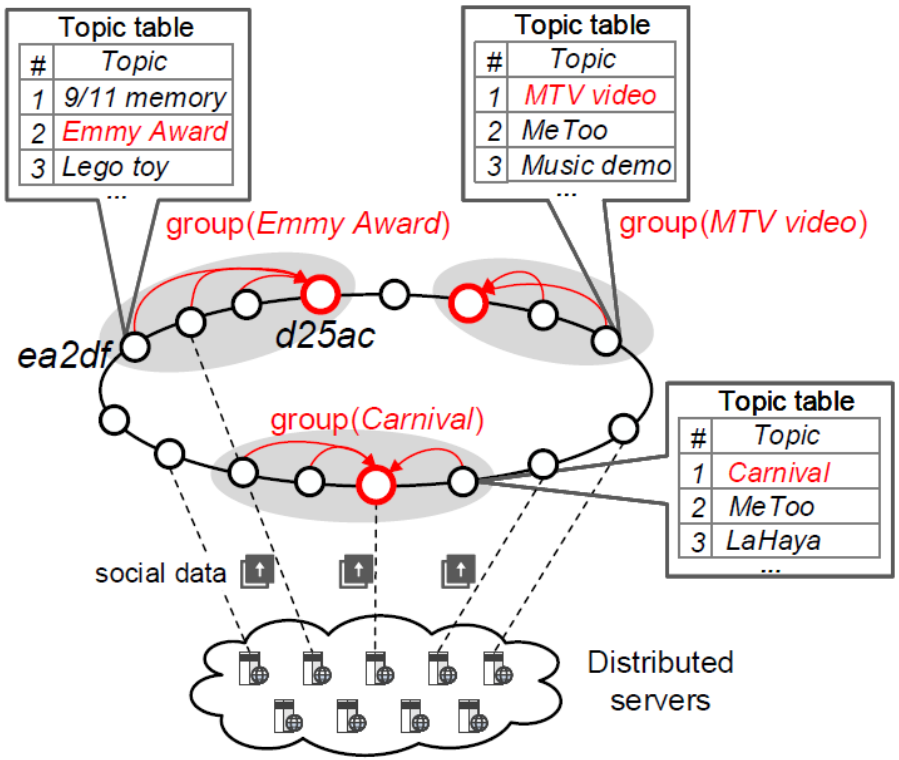
Sifter Workflow

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



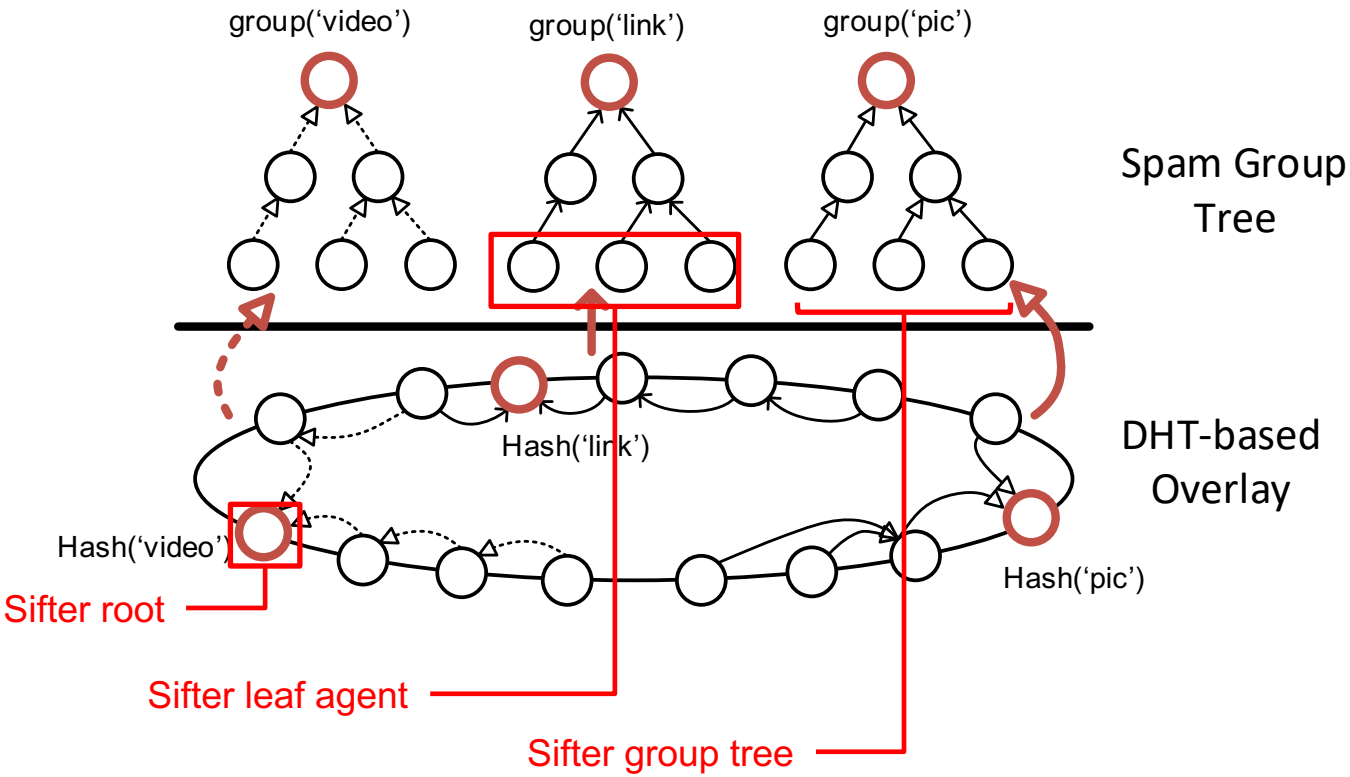
Online social data *group*

- 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.



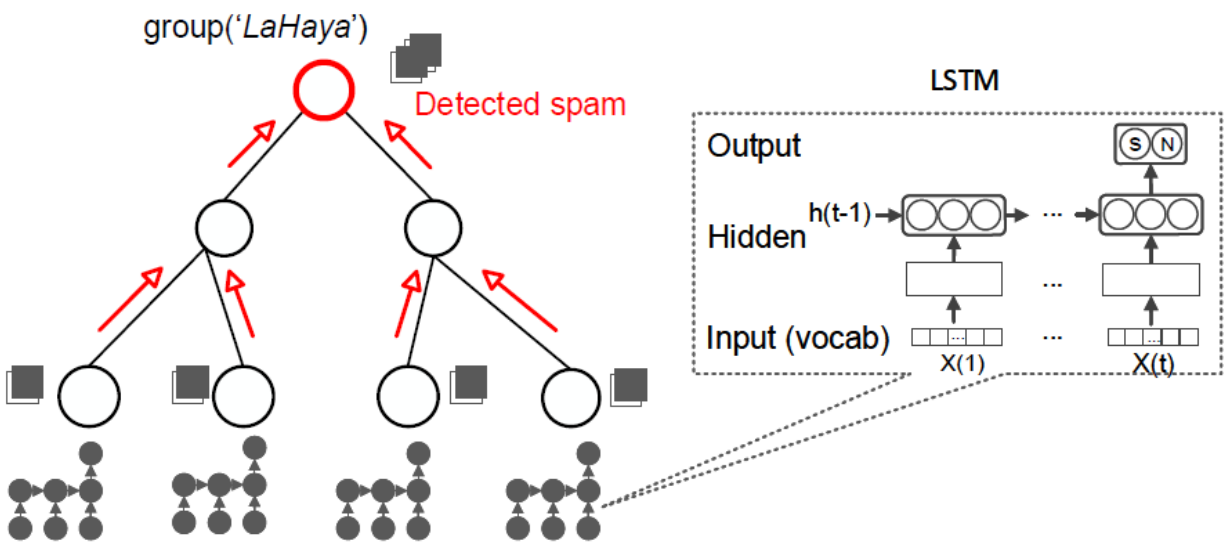
Sifter Implementation

- Sifter functional components
  - Sifter root, Sifter leaf agent, Sifter DHT-based tree



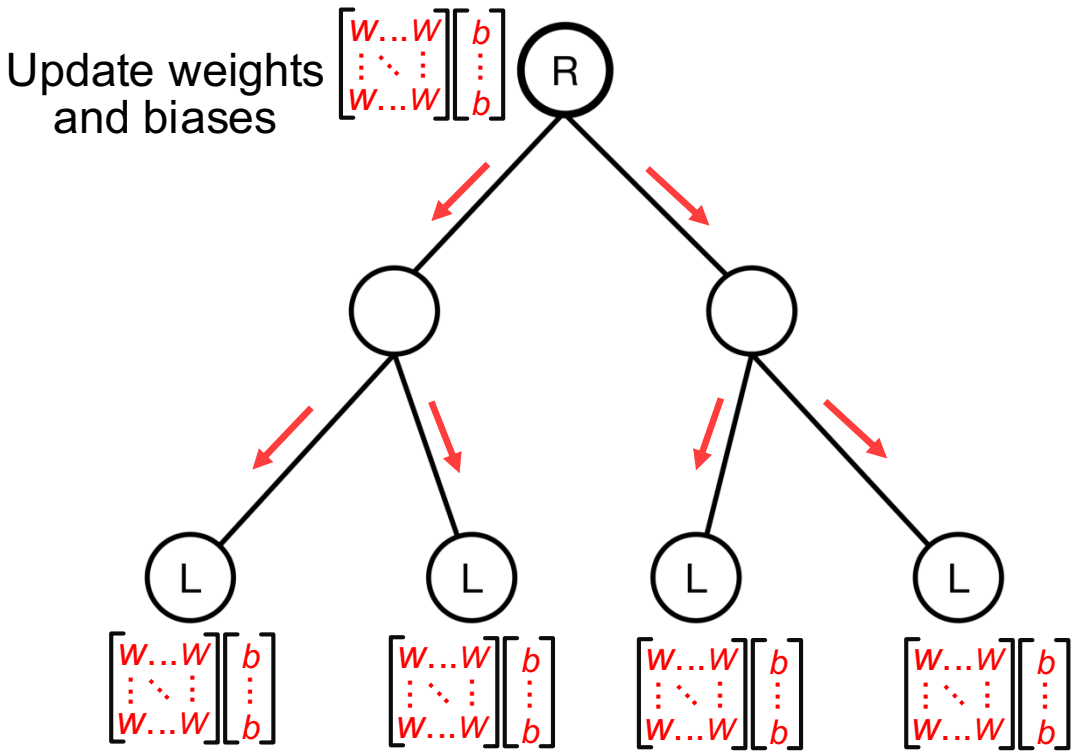
Spam Detection Unit (SDU)

- 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



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

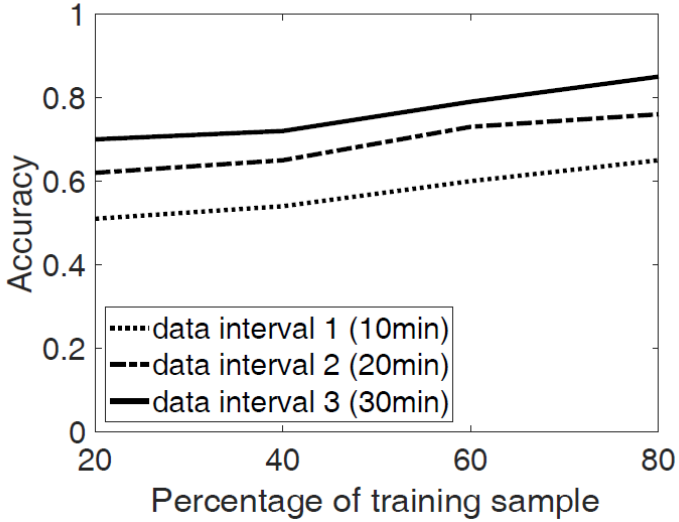
Hardware/Software	Specification
Servers	20
CPU	3.4GHz
Memory	4 GB
Disk	30 GB
Operating system	Ubuntu 16.04
Language	Java SE 1.7
Twitter	3,000,000 tweets from API

Spam Detection Evaluation

TABLE I. RESULTS OF SPAM CLASSIFICATION.

Data blocks	F1	Precision	Recall
data interval 1 (10min)	82.0%	0.903	0.751
data interval 2 (20min)	84.7%	0.912	0.791
data interval 3 (30min)	89.6%	0.923	0.87

A sample dataset which consists of 50,000 posts (37465 posts are *Ham* and 12535 posts are *Spam*)



System Evaluation

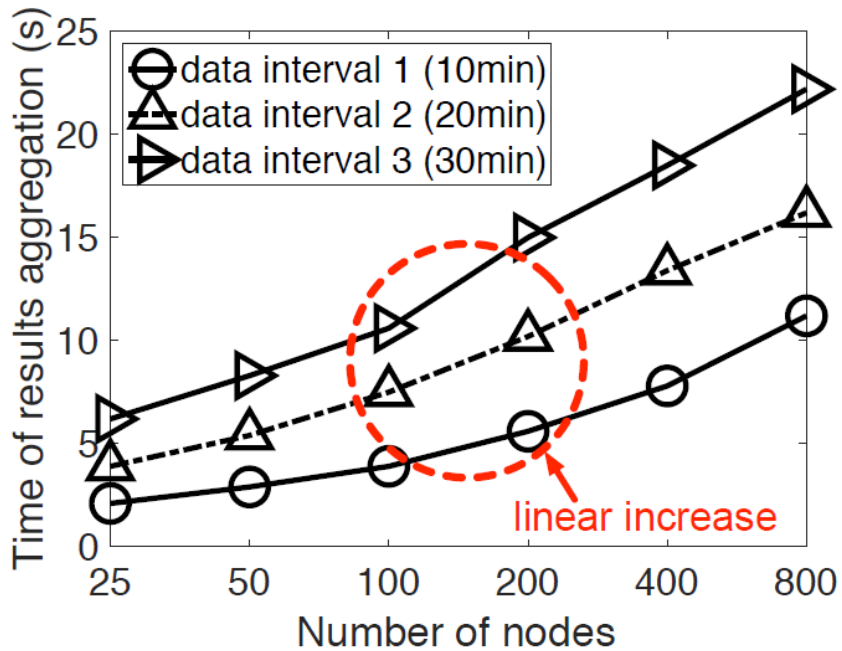


Fig ivery

The linear increment of time is determined by the tree depth O(logN), 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-balance with highly efficient data processing

Thank you.

Questions?