Last Time

Node centrality

Algorithms

- Fast algorithm to count triangles
  - Per node, and for whole graph
  - Main idea: compute the first few eigenvectors of graph’s adjacency matrix
    - Useful for computing clustering coefficients, etc.
  - (Personalized) PageRank
How to build (interactive) applications?

Will show you example applications that uses “diffusion-based” algorithms (i.e., a node influences its neighbors)

- You’ve already seen one: **Personalized PageRank** (= Random Walk with Restart)
- Will introduce **Belief Propagation** (powerful inference algorithm)
- “Spreading activation” or “degree of interest” in Human-Computer Interaction (HCI)
How to build (interactive) applications?

Why diffusion-based algorithms are widely used?

• **Intuitive to interpret**
  uses “network effect”, homophily, etc.

• **Easy to implement**
  Math is relatively simple

• **Fast**
  run time linear to #edges, or better

• **Probabilistic** meaning
Applications to show you

**DATA MINING**
- Polonium
- NetProbe
- Pegasus
- OPAvion

**HCI**
- Apolo
- Graphite
- Feldspar
- TopicViz
- UI Generation
- Text Entry

Applications to show you...
Applications to show you

**DATA MINING**
- Polonium
- NetProbe
- Pegasus
- OPAvion

**HCI**
- TopicViz
- UI Generation
- Text Entry
- Graphite
- Feldspar
- Apolo
Attention Routing
Attention Routing

Polonium – Find Malware
NetProbe – Find Fraudsters
Attention Routing

Polonium – Find Malware

NetProbe – Find Fraudsters

Human-In-The-Loop Graph Mining
Attention Routing

Polonium – Find Malware
NetProbe – Find Fraudsters

Human-In-The-Loop Graph Mining

Apolo – Mixed-Initiative Graph Sensemaking (ML + Vis)
Polonium: Tera-Scale Graph Mining and Inference for Malware Detection
Typical Malware Detection Method

**Signature**-based detection
1. Collect malware
2. Generate **signatures**
3. Distribute to users
4. Scan computers for matches

What about “**zero-day**” malware?
**No samples → No signatures → No detection**
How to detect them early?
Reputation-Based Detection

Computes **reputation score** for each application

*e.g.*, MSWord.exe

Poor reputation = Malware
Patent-pending
I led initial design and development
Serving 120 million users
Answered trillions of queries
I led initial design and development

Serving 120 million users

Answered trillions of queries

Patent-pending

Propagation of leverage of network influence unearths malware
Polonium works with 60 Terabyte Data

50 million machines anonymously reported their executable files

900 million unique files (Identified by their cryptographic hash values)

Goal: label malware and good files
### Why A Hard Problem?

<table>
<thead>
<tr>
<th>Existing Research</th>
<th>Polonium</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Small</strong> dataset</td>
<td><strong>Huge</strong> dataset (60 terabytes)</td>
</tr>
<tr>
<td>Detects <em>specific</em> malware (e.g., worm, trojans)</td>
<td>Detects <em>all</em> types (needs a general method)</td>
</tr>
<tr>
<td>Many false alarms (&gt;10%)</td>
<td><strong>Strict</strong> (&lt;1%)</td>
</tr>
</tbody>
</table>
Polonium: Problem Definition

Given

Undirected machine-file bipartite graph

37 billion edges, 1 billion nodes (machines, files)

Some file labels from Symantec (good or bad)

Find

Labels for all unknown files
Symantec has a **ground truth database** of **known-good** and **known-bad** files

e.g., set **known-good** file’s prior to **0.9**
How to Gauge **Machine Reputation**?

Computed using Symantec’s proprietary formula; a value between 0 and 1

Derived from anonymous aspects of machine’s usage and behavior
How to propagate known information to the unknown?
Key Idea: **Guilt-by-Association**

**GOOD files** likely appear on **GOOD** machines

**BAD files** likely appear on **BAD** machines

Also known as **Homophily**

<table>
<thead>
<tr>
<th></th>
<th>Good</th>
<th>Bad</th>
</tr>
</thead>
<tbody>
<tr>
<td>Good</td>
<td>0.9</td>
<td>0.1</td>
</tr>
<tr>
<td>Bad</td>
<td>0.1</td>
<td>0.9</td>
</tr>
</tbody>
</table>
How to propagate known information to the unknown?

Adapts **Belief Propagation (BP)**

A powerful *inference* algorithm

Used in image processing, computer vision, error-correcting codes, etc.
Propagating Reputation

<table>
<thead>
<tr>
<th></th>
<th>Machine</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Good</td>
<td>Bad</td>
<td></td>
</tr>
<tr>
<td>Good</td>
<td>0.9</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>Bad</td>
<td>0.1</td>
<td>0.9</td>
<td></td>
</tr>
</tbody>
</table>

Example

Machines

Files

1

2

3

4
Propagating Reputation

Machine

Good | Bad
---|---
0.9 | 0.1
0.1 | 0.9

File

| Good | Bad |
---|---|
0.9 | 0.1
0.1 | 0.9

Machines

Files

1

2

3

4

Example
Propagating Reputation

**Example**

- **Machines**
  - **A**
  - **B**
  - **C**

- **Files**
  - **1**
  - **2**
  - **3**
  - **4**

**Table:**

<table>
<thead>
<tr>
<th></th>
<th>Good</th>
<th>Bad</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Good</strong></td>
<td>0.9</td>
<td>0.1</td>
</tr>
<tr>
<td><strong>Bad</strong></td>
<td>0.1</td>
<td>0.9</td>
</tr>
</tbody>
</table>

**Graph:**

- **A** connects to **1** and **2** with weights 0.92 and 0.58, respectively.
- **B** connects to **2**, **3**, and **4** with weights 0.45, 0.38, and 0.35, respectively.
- **C** connects to **3** and **4** with weights 0.35 and 0.06, respectively.
Propagating Reputation

Example

<table>
<thead>
<tr>
<th>Machine</th>
<th>Good</th>
<th>Bad</th>
</tr>
</thead>
<tbody>
<tr>
<td>Good</td>
<td>0.9</td>
<td>0.1</td>
</tr>
<tr>
<td>Bad</td>
<td>0.1</td>
<td>0.9</td>
</tr>
</tbody>
</table>

Machines

Files

1

2

3

4

0.92

0.58

0.38

0.06

0.6

0.45

0.35

0.06
Propagating Reputation

**Example**

<table>
<thead>
<tr>
<th></th>
<th>Machine</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Good</td>
<td>Bad</td>
</tr>
<tr>
<td>Good</td>
<td>0.9</td>
<td>0.1</td>
</tr>
<tr>
<td>Bad</td>
<td>0.1</td>
<td>0.9</td>
</tr>
</tbody>
</table>

- **Machines**
  - A
    - 0.87
  - B
    - 0.81
  - C
    - 0.1

- **Files**
  - 1
    - 0.92
  - 2
    - 0.58
  - 3
    - 0.38
  - 4
    - 0.06
Two Equations in Belief Propagation

\[ b_i (x_i) = k \phi (x_i) \prod_{x_j \in N(i)} m_{ji} (x_i) \]

\[ m_{ij} (x_j) \leftarrow \sum_{x_i \in X} \psi_{ij} (x_i, x_j) \phi (x_i) \prod_{k \in N(i) \setminus j} m_{ki} (x_i) \]
Computing Node Belief (Reputation)

\[ b_i(x_i) = k \phi(x_i) \prod_{x_j \in N(i)} m_{ji}(x_i) \]
Computing Node Belief (Reputation)

\[ b_i (x_i) = k \phi (x_i) \prod_{x_j \in N(i)} m_{ji} (x_i) \]

Belief \quad Prior belief \quad Neighbors’ opinions
Computing Node Belief (Reputation)

\[ b_i(x_i) = k\phi(x_i) \prod_{x_j \in N(i)} m_{ji}(x_i) \]

- **Belief**
- **Prior belief**
- **Neighbors’ opinions**
Creating Message for Neighbor

\[ m_{ij}(x_j) \leftarrow \sum_{x_i \in X} \psi_{ij}(x_i, x_j) \phi(x_i) \prod_{k \in N(i) \setminus j} m_{ki}(x_i) \]
Creating Message for Neighbor

$$m_{ij}(x_j) \leftarrow \sum_{x_i \in X} \psi_{ij}(x_i, x_j) \phi(x_i) \prod_{k \in N(i) \setminus j} m_{ki}(x_i)$$

<table>
<thead>
<tr>
<th>Opinion for neighbor</th>
<th>Edge potential</th>
<th>Belief</th>
</tr>
</thead>
<tbody>
<tr>
<td>Good</td>
<td>0.9</td>
<td>0.1</td>
</tr>
<tr>
<td>Bad</td>
<td>0.1</td>
<td>0.9</td>
</tr>
</tbody>
</table>
### Creating Message for Neighbor

#### Opinion for neighbor

\[ m_{ij}(x_j) \leftarrow \sum_{x_i \in X} \psi_{ij}(x_i, x_j) \]

#### Edge potential

\[ \phi(x_i) \prod_{k \in N(i) \setminus j} m_{ki}(x_i) \]

#### Belief

<table>
<thead>
<tr>
<th></th>
<th>Good</th>
<th>Bad</th>
</tr>
</thead>
<tbody>
<tr>
<td>Good</td>
<td>0.9</td>
<td>0.1</td>
</tr>
<tr>
<td>Bad</td>
<td>0.1</td>
<td>0.9</td>
</tr>
</tbody>
</table>
Evaluation

Using **millions** of ground truth files, 10-fold cross validation

True Positive Rate
% of **bad** correctly labeled

85% True Positive Rate
1% False Alarms

False Positive Rate (False Alarms)
% of **good** labeled as **bad**
Evaluation

Using **millions** of ground truth files, 10-fold cross validation

- **85% True Positive Rate**
- **1% False Alarms**

Boosted existing methods by **10 absolute % point**

True Positive Rate
% of **bad** correctly labeled

False Positive Rate (False Alarms)
% of **good** labeled as **bad**
Scalability

Running Time Per Iteration

Linux
16-core Opteron
256GB RAM

3 hours,
37 billion edges
Scalability

How Did I Scale Up BP?

1. Early termination (after 6 iterations) → Faster

2. Keep edges on disk → Saves 200GB of RAM

3. Computes half of the messages → Twice as fast
Further Scale Up Belief Propagation

Use **Hadoop** if graph doesn’t fit in memory [ICDE’11]

Speed scales up **linearly** with number of machines

![Graph showing scale-up of speed with number of machines](image)

**Yahoo! M45 cluster**
- 480 machines
- 1.5 PB storage
- 3.5TB machine

Number of machines
Attention Routing

Polonium – Find Malware
NetProbe – Find Fraudsters

Human-In-The-Loop Graph Mining

Apolo – Mixed-Initiative Graph Sensemaking (ML + Vis)
NetProbe: A Fast and Scalable System for Fraud Detection in Online Auction Networks
NetProbe: The Problem

Find **bad sellers** (fraudsters) on eBay who don’t deliver their items

Auction fraud is #3 online crime in 2010

source: www.ic3.gov
NetProbe: Key Ideas

- Fraudsters **fabricate their reputation** by “trading” with their accomplices
- Fake transactions form **near bipartite cores**
- How to detect them?
NetProbe: Key Ideas

Use **Belief Propagation** and **heterophily (+ homophily)**

- Fraudster
- Accomplice
- Honest

Darker means more likely
NetProbe: Main Results
eBay
USA TODAY
Money.com
THE WALL STREET JOURNAL
MSNBC
Pittsburgh Tribune-Review
Symantec
Belgian Police
NetProbe Alpha - Unearth Networks of Suspicious Auction Users

Inspect user **alisher** for suspicious networks.

**alisher**

Registration: **Aug. 13, 2006**  Location: **United States**

**Fraudsters:** 95%

**Accomplice:** 4%

**Honest:** 1%

Suspected fraudster -- this user has been behaving much like the other suspects by trading with the similar sets of possible accomplices.
Attention Routing

Polonium – Find Malware
NetProbe – Find Fraudsters

Human-In-The-Loop Graph Mining

Apolo – Mixed-Initiative Graph Sensemaking (ML + Vis)
Human-In-The-Loop Graph Mining

Apolo: Machine Learning + Visualization

CHI 2011
Finding More Relevant Nodes

Citation network

HCI Paper

Data Mining Paper
Finding More Relevant Nodes

Citation network

HCI Paper

Data Mining Paper
Finding More Relevant Nodes

Apolo uses **guilt-by-association** (Belief Propagation)

Citation network

HCI Paper

Data Mining Paper
**Demo: Mapping the Sensemaking Literature**

**Nodes:** 80k papers from Google Scholar (node size: #citation)

**Edges:** 150k citations
Key Ideas (Recap)

Specify **exemplars**

Find **other** relevant nodes (BP)
Why Belief Propagation?

- Multiple groups
- Multiple examples in each group
- Positive and negative examples
- Fast: running time linear in #edges
- We’re first to adapt BP for sensemaking

Other methods considered: Personalized PageRank (= Random Walk with Restart = RWR), etc.
- Our PKDD’11 paper shows BP>>RWR
Apolo’s Contributions

1 Human + Machine

It was like having a partnership with the machine.

2 Personalized Landscape
Apolo 2010
Apolo 2011

22,000 lines of code. Java 1.6. Swing. Uses SQLite3 to store graph on disk
User Study

Used citation network

**Task:** Find related papers for 2 sections in a survey paper on *user interface*

1. Model-based generation of UI
2. Rapid prototyping tools
Between subjects design
Participants: grad student or research staff
"Model-based"

"Prototyping"

Apolo

Google Scholar

10 papers for each section
Apolo

Google Scholar

“Model-based”

“Prototyping”

10 papers for each section

Expert judges rated papers

1 + 0 = 1
1 + 1 = 2
0 + 0 = 0

1 + 1 = 2
Apolo wins.

* Statistically significant, by two-tailed t test, p <0.05
Apolo: Recap

A **mixed-initiative** approach for exploring and creating personalized **landscape** for large network data

Apolo = ML + Visualization + Interaction
Feldspar

Finding Information by Association.

CHI 2008

Polo Chau, Brad Myers, Andrew Faulring

YouTube: http://www.youtube.com/watch?v=Q0TIV8F_o_E&feature=youtu.be&list=ULQ0TIV8F_o_E
Feldspar
Feldspar

A system that helps people find things on their computers when typical search or browsing tools don’t work
Feldspar

A system that helps people find things on their computers when typical search or browsing tools don’t work

An example scenario...
“Find the **webpage** mentioned in the **email** from the **person** I met at an **event**“
“Find the **webpage** mentioned in the **email** from the **person** I met at an **event**“

If I can’t remember the specifics, such as any text in the webpage, email, etc.

→ **Can’t search**
“Find the **webpage** mentioned in the **email** from the **person** I met at an **event**”

If I can’t remember the specifics, such as any text in the webpage, email, etc.

→ Can’t search

If I haven’t bookmarked the webpage

→ Can’t browse
“Find the **webpage** mentioned in the **email** from the **person** I met at an **event**“
“Find the **webpage** mentioned in the **email** from the **person** I met at an **event**“

But I can describe the **webpage** with a chain of **associations**.
“Find the **webpage** mentioned in the **email** from the **person** I met at an **event**“

But I can describe the **webpage** with a chain of **associations**.

**webpage** – **email** – **person** – **event**
“Find the webpage mentioned in the email from the person I met at an event“

But I can describe the webpage with a chain of associations.

webpage – email – person – event

The psychology literature has shown that people often remember things exactly like this.
Natural question:
Can I find things by associations?
Natural question:

Can I find things by associations?

Can I find the webpage by specifying its associated information (email, person, and event)?
Natural question: Can I find things by associations?

Can I find the webpage by specifying its associated information (email, person, and event)?

We created Feldspar, which supports this associative retrieval of information.
Feldspar stands for....
Feldspar stands for....

Finding
Elements by
Leveraging
Diverse
Sources of
Pertinent
Associative
Recollection
Implementation: Overview

Create a graph database to store the associations among items on the computer.

Develop an algorithm that processes the query and returns results.
Creating an Association Database (a graph)

Install Google Desktop and let it index all the items on the computer.
Creating an Association Database (a graph)

Focus on 7 types

Install Google Desktop and let it index all the items on the computer

filetype:calendar
filetype:email
filetype:doc, etc.
filetype:web
Creating an Association Database (a graph)

Focus on 7 types

Install Google Desktop and let it index all the items on the computer

Identify associations and build our database, which is a directed graph

filetype:calendar
filetype:email
filetype:doc, etc.
filetype:web
Creating an Association Database (a graph)

Focus on 7 types

Identify **associations** and build our database, which is a **directed graph**

Install **Google Desktop** and let it index all the items on the computer
Creating an Association Database (a graph)

Identify **associations** and build our database, which is a **directed graph**

Focus on 7 types

Install **Google Desktop** and let it index all the items on the computer

---

**filetype:calendar**
**filetype:email**
**filetype:doc, etc.**
**filetype:web**
Algorithm that processes the query
Algorithm that processes the query
Algorithm that processes the query

webpages - emails - persons - events
Algorithm that processes the query

webpages - emails - persons - events
Algorithm that processes the query

webpages - emails - persons - events

Sample association graph database
Algorithm that processes the query

webpages - emails - persons - events

webpages - emails - persons

Sample association graph database

Results Generator
Algorithm that processes the query

webpages - emails - persons - events

webpages - emails - persons

webpages - emails

Sample association graph database

Results Generator
Algorithm that processes the query

webpages - emails - persons - events

webpages - emails - persons

webpages - emails

webpages

Results Generator

Sample association graph database

What to Do When Search Fails: Finding Information by Association
Polo Chau, Brad Myers, Andrew Faulring
Algorithm that processes the query

webpages - emails - persons - events

webpages - emails - persons

webpages - emails

webpages

One Results Generator for each pair of associations
For 7 data types → needs 7x7=49 results generators
About the algorithm
About the algorithm

Adding more data types

- Implement more results generators
- Number of generators grows quadratically
About the algorithm

Adding more data types

- Implement more results generators
- Number of generators grows quadratically

But not all combinations make sense

E.g. date related to date doesn’t quite make sense

Implemented 38 (out of 49)
About the algorithm

Adding more data types

→ Implement more results generators
→ Number of generators grows quadratically

But not all combinations make sense

E.g. date related to date doesn’t quite make sense

Implemented 38 (out of 49)

Also, there are limited number of data types
Practitioners’ guide to building (interactive) applications?

Important that you pick a good problem!

• Otherwise, you solve a “non-problem”, and nobody cares

Think about scalability early

• e.g., picking a scalable algorithm early on

When building interactive applications, use iterative design approach (as in Apolo)

• Why? It’s hard to get it right the first time

• Create prototype, evaluate, modify prototype, evaluate, ...

• Quick evaluation helps you identify important fixes early (can save you a lot of time)
Practitioners’ guide to building (interactive) applications?

How to do iterative design?

What kinds of prototypes?
  • Paper prototype, lo-fi prototype, high-fi prototype

What kinds of evaluation?
  • Ask your friends,
  • Lab study (controlled, as in Apolo),
  • Longitudinal study (usage over months)
  • Deploy it and see the world’s reaction!