Advances in Mining the Web

Bamshad Mobasher  
DePaul University, USA

Olfa Nasraoui  
University of Louisville, USA

Myra Spiliopoulou  
Magdeburg University, Germany

Osmar R. Zaïane  
University of Alberta, Canada

ACM SIGKDD Conference  
Tutorial  
Paris, June 28th, 2009
Motivation for the Tutorial

- The Web has changed our way of life and the Web 2.0 has changed our way of perceiving and using the Web.
- Data analysis is now required in a plethora of applications that aim to enrich the experience of people with the Web.
- WebKDD 10 years: mature field with multidisciplinary applications
Goal for the Tutorial

- GOAL: summarize research, current practice and trends in Web related data mining:
  - We first discuss data mining for the social Web. We elaborate on social network analysis and focus on community mining,
  - then go over to recommendation engines and personalization.
    - We discuss the challenges that emerged through the shift from the traditional Web to Web 2.0.
    - We then focus on two issues - the need to protect Web applications from manipulation and the need to make them adaptive towards change.
    - We first discuss manipulations/attacks in recommender systems and present counter-measures.
  - We then elaborate on how changes/concept drifts can be dealt with in applications that analyze clickstream data, monitor topics in news and blogs, or monitor communities and their evolution.
Tutorial Presenters

Bamshad Mobasher

Professor at the School of Computing and the director of Center for Web Intelligence at DePaul University. Co-chair of WebKDD series of workshops and the Workshop series on Intelligent Techniques for Web Personalization at AAAI and IJCAI.

http://maya.cs.depaul.edu/~mobasher

Olfa Nasraoui

Associate Professor in Computer Engineering & Computer Science, Endowed Chair of e-commerce, and Director of Knowledge Discovery & Web Mining lab at the University of Louisville. Co-chair of WebKDD 2004-2008. Her research activities include Data Mining, in particular, Web mining and mining evolving data streams, Personalization, and Computational Intelligence.

http://webmining.spd.louisville.edu/
Tutorial Presenters

Myra Spiliopoulou

Professor of Information Systems in the Faculty of Computer Science, Otto-von-Guericke-Univ. Magdeburg, and Chair of the Knowledge Management & Discovery (KMD) group. Co-founder of WebKDD workshop series and co-chair of the workshop for 10 years.

http://omen.cs.uni-magdeburg.de/itikmd

Osmar R. Zaïane

MacCalla Professor in Computing Science at the University of Alberta, Canada, and Scientific Director of the Alberta Ingenuity Centre for Machine Learning. Co-chair of WebKDD in 2003, 2005 and 2008. His current research interests include web mining, social network analysis and canonical data mining tasks such as outlier detection, classification, clustering and itemset mining, and their application in the health care domain.

http://www.cs.ualberta.ca/~zaiane/
Tutorial Structure

- Block 1: Introduction
- Block 2: Mining the Social Web
  - Social Network Analysis
    - History, concepts and research
    - Applications, problems and line of attack
  - Community Mining
    - Graph partitioning
    - Hierarchical approaches
  - Global vs. Local Information-based Structures
  - Communities with Overlap
Tutorial Structure

- Block 3: Robust Recommendation and Personalization
  - Overview
    - Recommendation Problem
    - Knowledge Sources
    - Trust and Security Issues
  - Attacks in Collaborative Recommendations
    - Characterizing attacks
    - Measuring Impact of attacks
  - Attacks in Social Tagging (Folksonomies)
    - A framework for analysis of attacks
    - Attack types
  - Responding to Attacks
Tutorial Structure

- Block 4: Evolution in the Web
  - The stream mining paradigm
  - Capturing evolution in (Web) document streams
  - Capturing evolution in communities
Tutorial Structure

- Block 5: Mining Web Data Streams
  - Stream Data Mining and handling concept drift
  - Web Data Stream Mining techniques for clickstream data
    - Background on Web Usage Mining (WUM)
    - Online & Stream-based WUM Methods
      - Frequent item set mining
      - Clustering
      - Online Web User Profiling and Personalization
      - Frameworks for Online Change Detection and Evolution Monitoring
  - Evaluation of Web Data Stream Mining

- Block 6: Conclusions and Outlook
Acknowledgements

Cordial thank you to our students and collaborators:

U. of Alberta
- Randy Goebel
- Jiyang Chen

DePaul U.
- Robin Burke
- Runa Bhaumik
- Maryam Ramezani
- JJ Sandvig
- Tom Schimoler

U. Ill. Chicago
- Chad Williams

O-V-G U. Magdeburg
- Tanja Falkowski
- René Schult

U. of Louisville
- Carlos Rojas
- Esin Saka

Funding sources:
- Alberta Ingenuity Funds
- National Sciences and Engineering Council of Canada
- UNIVERSITY OF LOUISVILLE
- Bucks for Brains

O. Nasraoui is supported by National Science Foundation CAREER Award IIS-0133948
Presentation Agenda (tentative)

- **Block 1: Introduction**
- **14:10-14:50** Block 2: Mining the Social Web  
  (Osmar Zaïane)
- **14:50-15:30** Block 3: Recommendation & Personalization  
  (Bamshad Mobasher)
- **15:30-16:00** Coffe Break
- **16:00-16:40** Block 4: Evolution in the Web  
  (Myra Spiliopoulou)
- **16:40-17:20** Block 5: Mining Web Data Streams  
  (Olfa Nasraoui)
- **17:20-17:30** Block 6: Conclusions and Outlook
Presentation Outline

- Block 1: Introduction
- Block 2: Mining the Social Web
  - Social Network Analysis
  - Community Mining
  - Global vs. Local Information-based Structures
  - Communities with Overlap
- Block 3: Recommendation and Personalization
- Block 4: Evolution in the Web
- Block 5: Mining Web Data Streams
- Block 6: Conclusions and Outlook
SNA, a multidisciplinary field
Presentation Outline

- Block 1: Introduction
- Block 2: Mining the Social Web
  - Social Network Analysis
  - Community Mining
  - Global vs. Local Information-based Structures
  - Communities with Overlap
- Block 3: Recommendation and Personalization
- Block 4: Evolution in the Web
- Block 5: Mining Web Data Streams
- Block 6: Conclusions and Outlook
A quick History

- Social network analysis is a key technique traditionally studied in sociology, anthropology, epidemiology, sociolinguistics, psychology, etc. Today it is a modern technique in marketing, economics, intelligence gathering, criminology, medicine, computer science, etc.

- J. Barnes is credited with coining the notion of social networks (theory) in 1954.

- Precursors of social network theory date from the 19th century such as Simmel, Durkheim and Tönnies.

- Massive increase in studies of social networks (in social sciences) since the 1970s.

- The increase of available data, the Internet phenomenon, Web 2.0, etc. have only catapulted the interest in SNA research.
What is Social Network Analysis?

- [Wikipedia] A social network is a social structure made of nodes (which are generally individuals or organizations) that are tied by one or more specific types of interdependency, such as values, visions, ideas, financial exchange, friendship, sexual relationships, kinship, dislike, conflict or trade.

- Social Network Analysis (SNA) is the study of social networks to understand their structure and behaviour.

- Which node is the most influential? which one is central? What are the hubs? What are the groups? Who knows who?, What are the short paths? What is perceived by who? ...
## Networks in Social and Behavioral Sciences

- **Social Networks**
  - Who knows who?
- **Socio-cognitive Networks**
  - Who thinks who knows who?
- **Knowledge Networks**
  - Who knows what?
- **Cognitive Knowledge Networks**
  - Who thinks who knows what?

### Socio-centric Analysis
- Emerged in sociology: quantification of interaction among a group of people. Focus on identifying global structural patterns in a network.

### Ego-centric Analysis
- Emerged in psychology and anthropology: quantification of interaction between an individual (ego) and others (alters) directly or indirectly related to ego.

### Reality vs. Perception

<table>
<thead>
<tr>
<th>Reality</th>
<th>Perception</th>
</tr>
</thead>
<tbody>
<tr>
<td>Social Network</td>
<td>Knowledge Network</td>
</tr>
<tr>
<td>Socio-cognitive Network</td>
<td>Cognitive knowledge Network</td>
</tr>
</tbody>
</table>

### Acquaintance knowledge
Popularization

- **Six degrees of separation** (Chains by Frigyes Karinthy 1929)
  Hypothesized: modern world was 'shrinking' due to the ever-increasing connectedness of human beings. Used the idea of six degrees of freedom in mechanics.

- **Milgram’s Paradox: Small world effect** (Stanley Milgram, 1967)
  Famous experiment in 1970 sending letters from Omaha to Boston 64/296 arrived. Average path 5.5~6.

- **Google** ‘s PageRank (1998) uses a network of web page « citations » to estimate the importance of pages and rank them.

- **Internet social network tools**
The famous case of Enron

- Energy and commodity trading company.
- Filed for bankruptcy in 2001 (Enron financial scandal) and dissolution of Arthur Andersen accounting firm.
- E-mail data made public
  - [http://www.cs.cmu.edu/~enron](http://www.cs.cmu.edu/~enron)
  - 151 users
  - 200,399 e-mail messages
  - Typical use for research (email logs)
    - Modeling a Socio-Cognitive Network
    - Quantitative Measures for Perceptual Closeness
    - Automatic Extraction of Concealed Relations
    - …

Visualization of Enron’s email network, Jeffrey Heer, 2005
Types of Relations and Networks (1)

- Unique relation networks
  - Nodes or individuals are tied by the same relation

- Multiple relation networks
  - Nodes or individuals are tied by different kinds of relationships
Types of Relations and Networks (2)

- Homogeneous relationship
  - Relationships between nodes of the same type

- Heterogeneous relationships
  - Relationships between nodes of different types
Other key concepts

- **Edge Weight**: interaction frequency, importance of information exchange, intimacy, emotional intensity, etc.
- **Symmetric relation or not (directional)**
- **Centrality**: determines the relative importance of a vertex (or edge) within a network.
  - Degree Centrality: Measures the normalized number of edges incident upon a node $n$;
  - Betweenness Centrality: Measures how many times a node $n$ occurs in a shortest path between any other 2 nodes in the graph;
  - Closeness Centrality: Mean shortest path distance between a node $n$ and all other nodes reachable from it;
  - Eigenvector Centrality: Measures importance of a node $n$ by assigning a score to each node based on the principal that connections to high-scoring nodes contribute more to the score of a node in question than equal connections to low-scoring nodes (e.g. PageRank).
Applications

- **Terrorism and crimes**
  - Social Network analysis is an important part of a conspiracy investigation and is used as an investigative tool. Group structure may be important to investigations of racketeering enterprises, narcotics operations, illegal gambling, and business frauds.

- **Medicine – epidemiology**
  - Valuable epidemiological tool for understanding the progression of the spread of an infectious disease.

- **Marketing**
  - Emarketer projected that Social Network Marketing spending in the USA will reach approximately $1.3 billion in 2009. [http://www.emarketer.com/Reports/All/Emarketer_2000541.aspx](http://www.emarketer.com/Reports/All/Emarketer_2000541.aspx)

- **Product Recommendation**
  - Current recommendation models assume all users’ opinions to be independent. Use of SNA relaxes the iid assumption.

- **Bio-informatics (protein interaction)**

- **Relevance Ranking**

- **Information and Library Science**
Prominent problems in SNA

- Social network extraction/construction
- Link prediction
- Approximating large social networks
- Identifying prominent/trusted/expert actors in social networks
- Search in social networks
- Discovering communities in social networks
- Knowledge discovery from social networks
- Predicting evolution

Analogy with Clustering
Presentation Outline

- Block 1: Introduction
- Block 2: Mining the Social Web
  - Social Network Analysis
  - Community Mining
  - Global vs. Local Information-based Structures
  - Communities with Overlap
- Block 3: Recommendation and Personalization
- Block 4: Evolution in the Web
- Block 5: Mining Web Data Streams
- Block 6: Conclusions and Outlook
What is Community Structure?

- Community structure denotes the existence of densely connected groups of nodes, with only sparser connections between groups.

- Many social networks share the property of a community structure, e.g., WWW, tele-communication networks, academic collaboration networks, friendship networks, etc.

Many similarities with data Clustering
Community Structure Example

A social network of the novel *Les Misérables*  
*(Victor Hugo)*

http://www-personal.umich.edu/~mejn/networks
Community Structure Example

A academic (physicists) collaboration social network

http://www-personal.umich.edu/~mejn/networks
Community Structure Example

A social network of Amazon Books

http://www-personal.umich.edu/~mejn/networks
It is important!

- Finding communities could be of significant importance in a variety of applications.
- WWW Pages (in the same hyperlink community) might cover related topics.
- Researchers (in the same collaboration community) might work on similar problems.
- People (in the same tele-communication community) might be close friends.
- Communities in social settings might explain or predict the spread of contagious diseases.
- And many other examples.
What is a Community

- **Graph theory:** Communities are those densely connected groups of vertices, with only few connections between groups.

- **Sociology:** Communities are social groups that entities in the same group share similar properties or connect to each other via certain relations.

- More definitions are available, however, communities are often different for different domains, even for different networks in the same domain. Thus there is no general definition.

- In community mining the community structure found is usually a byproduct of the discovery procedure.
Traditional Community Mining Taxonomy

Community Mining

Graph Partitioning
  - Spectral Clustering
  - Other GP Methods

SNA Approaches
  - Hierarchical Clustering
    - Betweenness
    - Modularity
    - Other Metric
Graph Partitioning Approaches

- There is a long computer science tradition in graph partitioning: believed to be an NP-complete problem.

- Typical Solution: greedily optimize an objective function: the fraction between within-community and between-community edges.

- Iterative Bisection: find the best two-group-cut, then further sub-divide until the required community number is met.
Graph Partitioning Methods

- Graph partitioning algorithms are heavily used to find communities.

- Parameters that are difficult to decide are usually required: size of communities, number of clusters.

- Spectral Clustering with benefit functions: ratio cut (Hagen et al. 1992), normalized cut (Shi et al. 1997), min-max cut (Ding et al. 2001).

- Unfortunately, equal-sized communities are usually favoured.
Other Problems

- Require input parameters: number of the partitions, and their sizes

- Such information would never be available for large social networks. They should be determined by the network, not the user.

- Fundamental problem: cut (sum of edge weights between communities) is simply not the right thing to optimize.
Hierarchical Clustering

- Greedily optimize a metric, which evaluates the node centrality or community quality.

- An example metric: edge betweenness, which is the number of edge occurring on the shortest path between other pair of nodes in the network.

- Up-down Algorithm: remove the edge with highest betweenness value in each step.
Modularity Q

- Proposed by Newman and Girvan in 2004 as a measure of the quality of a particular division of the network.

- \[ Q = (\text{number of edges within communities}) - (\text{expected number of such edges}) \]

- Intuition: compare the division to a random network with same nodes and same degrees, but edges are placed randomly.

- Greedily maximizing Q outperformed all other methods, in most cases by an impressive margin, for community detection.
Success of the Modularity

- Algorithm: bottom-up agglomerative hierarchical clustering to maximize $Q$.

- $Q$ has proven to be highly efficient.

- $Q$-based methods over-perform other community mining algorithms on many networks, usually with a big margin.

- **FastModularity** [Clauset, Newman and Moore 2004] – use of Max Heaps and binary tree to provide an efficient $O(md \log n)$ Modularity implementation where $m$ is the # of edges, $n$ is the number of nodes, and $d$ the depth of the dendrogram.
Problem Solved?

- There are three major problems for Q.
  - Q requires information of the entire network.
  - Q has a resolution limit and may fail to identify communities smaller than a certain scale.
  - Q cannot be used to compare community qualities in different networks. (Q = 0.360 for both)
Max-Min Modularity [Chen et al. SDM’09]

- Evaluation Metric: reward for connected pairs and penalty for disconnected ones.
- A “disconnection” can be “unobserved” in many social networks, e.g., biological network, dynamic Facebook.
- *Maximize* the edge number within groups and *minimize* the number of unrelated pairs defined by experts within groups at the same time.
- Use of complement graph

\[
Q_{\text{max-min}} = Q_{\text{max}} - Q_{\text{min}}
\]

\[
Q_{\text{min}} = \frac{1}{n(n-1)-2m-2} \sum_{xy} [A'_{xy} - P'_{xy}] \phi(C_x, C_y)
\]

where:
- \( Q_{\text{max}} \) is the Modularity \( Q \)
- \( n \) is the node number.
- \( U \) is the related but disconnected pair set defined by domain experts.
Example Results with Max-Min Modularity

Karate-Club dataset
34 nodes in 2 communities

Sawmill Strike dataset
24 nodes in 3 communities

node pairs are “related” if they share neighbours
On Real Networks?

- Most of these approaches require knowledge of the entire network structure, e.g., number of nodes/edges, number of communities in the network. However, this is problematic for networks which are either too large or dynamic, e.g., the WWW.

- The size of the WWW: 1 trillion unique URLs. The index size of Google is about 40 billion.
- Facebook has more than 200 million active users.
- Vodafone has 289 million customers worldwide.
Presentation Outline

- Block 1: Introduction
- Block 2: Mining the Social Web
  - Social Network Analysis
  - Community Mining
  - Global vs. Local Information-based Structures
  - Communities with Overlap
- Block 3: Recommendation and Personalization
- Block 4: Evolution in the Web
- Block 5: Mining Web Data Streams
- Block 6: Conclusions and Outlook
Local Methods

- A common assumption for the proposed methods is that the complete global network information is always available.

- For some huge networks, e.g., WWW, global information is not always accessible.

- Scenarios: Locate a friend community of a person in Facebook or Find a page cluster of a particular page in the WWW.

- The only available information are nodes that have been visited and their neighbours. All global methods fail.
Typical Problem Definition

- A local community $D$ includes cores ($C$) nodes and boundary ($B$) nodes.
- If one new node is merged, its neighbours are added into shell nodes ($S$).
Modularity in Local Network

- Clauset proposed in 2005 the local modularity using the modularity methodology:
  - Measure $R$, the quality of communities
  - Greedily maximize the $R$ measure to identify communities

- $R = \frac{\text{within edges of boundary nodes}}{\text{total edges of boundary nodes}}$

- $R$ measures by the sharpness of the boundary nodes. Identify local community by keeping merging until no merge can increase $R$. 
Measure the Local Community

- Two factors to consider in local community quality:
  - high node relations within the set
  - low relations between set and outside nodes

- R directly represent these two factors by maximizing internal degrees and minimizing external degrees

- The important missing aspect for R is the *connection density*, not the absolute number of connections, that matters in community structure evaluation.
Detecting based on Local Density

- Chen et al. 2009 propose to measure the two factors by maximizing average internal degree (id) inside the whole community and minimizing average external degree (ed) of boundary nodes, by maximizing id/ed.

- The “density” idea solves the outlier problem and dramatically increases community detection accuracy:
  - F-measure 0.595 -> F-measure 0.952 (on NCAA Football dataset with ground truth)
Presentation Outline

- Block 1: Introduction
- Block 2: Mining the Social Web
  - Social Network Analysis
  - Community Mining
  - Global vs. Local Information-based Structures
  - Communities with Overlap
- Block 3: Recommendation and Personalization
- Block 4: Evolution in the Web
- Block 5: Mining Web Data Streams
- Block 6: Conclusions and Outlook
Global Overlapping Methods

- We usually assume that one node belongs to only one community. However, in real world, it is not the case.

- One person can belong to two or more communities, thus we need to consider overlapping communities.

- Typical approach: find the cluster, then measure the relations of nodes in question to different clusters with arbitrary threshold.
Palla et al. proposed the CFinder system in Nature 2005, using an simple but efficient idea to detect overlaps based on cliques.

- Cliques are completely connected sub-graphs, representing strong communities.

- One node can belong to multiple cliques, which shows community overlaps.
CFinder

- CFinder takes a parameter $k$, which is the clique size.
- Two $k$-cliques are adjacent if they share $k-1$ nodes.
- Given clique size $k$, merge adjacent $k$-cliques into one community to identify the network structure.
- Problem: also depends on parameters, $k = 3, 4, 5$ usually give reasonable results.

http://www.cfinder.org/
Local Overlapping Methods

- Previous works, using only local information, focus on locating the first local community given a start node.

- Iteratively applying the community identification algorithm based on local modularity may be able to find local-overlapping communities (Chen et al. CASoN 2009)

- A density based approach à la DBScan. Xu et al. proposed SCAN (Xu et al. KDD 2007)
Local Clustering: DBSCAN

- DBSCAN:
  - Two parameters: epsilon and minPts: a cluster node need to have more than minPts neighbours that have a distance score smaller than epsilon
SCAN

- Similarity score between connected pairs are calculated (based on their shared neighbourhood) and used as distance.

- Problem: same as DBSCAN, the performance highly depends on the two parameters Epsilon and Minpt.
Chen et al. 2009 proposed a visual data mining approach to detect overlapping communities.

Given a start node, the approach first generates a sequence of nodes with their highest “reachability score” to former nodes in the list.
- similar to the well-known visual data mining approach OPTICS.

A 2D visualization is then built to show the community structure, with “mountain” and “valley” curves.
User needs to give two thresholds after observation to extract groups, hubs and outliers.

Visual data mining helps user to get good parameters while it is hard for other approaches.
References (Block 2) 1/5

- J. Chen, O. R. Zaiane and R. Goebel, Detecting Communities in Large Networks by Iterative Local Expansion, International Conference on Computational Aspects of Social Networks (CASoN), Fontainebleau, France, June 24-27, 2009


References (Block 2) 4/5


Thank you!

Questions?
Presentation Outline

☑ Block 1: Introduction
☑ Block 2: Mining the Social Web
☐ Block 3: Robust Recommendation and Personalization (Bamshad Mobasher)
  ☐ Overview
  ☐ Attacks in Collaborative Recommendation
  ☐ Attacks in Social Tagging Systems
  ☐ Responding to Attacks
☐ Block 4: Evolution in the Web
☐ Block 5: Mining Web Data Streams
☐ Block 6: Conclusions and Outlook
Data Mining and Personalization

“Killer App” for data mining?

Tangible successes both in the research and in industrial applications
- E-commerce recommender systems
- personalized Web agents
- Web marketing and eCRM
- personalized search
- recommendation on the social Web

But, what are the remaining challenges?
- We will focus on one: vulnerabilities and robustness of recommendation on the Web
Trust and Security in User Adaptive Systems

- Web applications increasingly rely on user profiles and collaborative data to function effectively.

- Examples:
  - Personalization/recommender systems (Amazon.com)
  - User opinions on products (CNet)
  - Personalized search applications
  - File and music sharing
  - Collaborative tagging (folksonomies)

- A lot of incentive for manipulating system behavior.
Web Personalization / Recommendation Problem

- Dynamically serve customized content (pages, products, recommendations, etc.) to users based on their profiles, preferences, or expected interests

- Formulated as a prediction problem
  - Given a profile $P_u$ for a user $u$, and a target item $i_t$, predict the preference score of user $u$ on item $i_t$

- Typically, the profile $P_u$ contains preference scores by $u$ on some other items, $\{i_1, \ldots, i_k\}$ different from $i_t$
  - preference scores on $i_1, \ldots, i_k$ may have been obtained explicitly (e.g., movie ratings) or implicitly (e.g., time spent on a product page or a news article)
Knowledge sources

- AI systems can be characterized by their knowledge sources:
  - Social
    - knowledge about individuals other than the user
  - Individual
    - knowledge about the user
  - Content
    - knowledge about the items being recommended
Content-Based Recommenders

- Predictions for unseen (target) items are computed based on their similarity (in terms of content) to items in the user profile.
- E.g., user profile $P_u$ contains

![Movie Posters]

recommend highly: and recommend “mildly”:
Collaborative Recommenders

- Predictions for unseen (target) items are computed based on the other users’ with similar interests on items in user $u$’s profile
  - i.e. users with similar tastes (aka “nearest neighbors”)
Vulnerabilities

- Any knowledge source can be attacked
  - but three vulnerabilities stand out

- Content
  - false item data, if data gathered from public sources
    - an item is not what its features indicate
    - Example: web-page keyword spam
  - biased domain knowledge
    - recommendations slanted by system owner
    - Example: Amazon “Gold Box”

- Social
  - bogus profiles
  - our subject today
Vulnerabilities

- Not a standard security research problem
  - not trying to prevent unauthorized intrusions
- Need robust (trustworthy) systems
- The Data Mining Challenges
  - Finding the right combination of modeling approaches that allow systems to withstand attacks
  - Detecting attack profiles
How Vulnerable?
How Vulnerable?

John McCain on last.fm

Tags

war mongerer
awful barneycore
the troops by voting for the war
betrayed
bitch
bush apologist chuck norris
does not approve
conservative
conservative moron corrupt
crap
dishonest
enslaver extremist flip
flopper fox news approved
hates
What is an attack?

- An attack is
  - a set of user profiles added to the system
  - crafted to obtain excessive influence over the recommendations given to others
- In particular
  - to make the purchase of a particular product more likely (push attack)
  - or less likely (nuke attack)
- There are other kinds
  - but this is the place to concentrate – profit motive
Definitions

- An attack is a set of user profiles $A$ and an item $t$ such that $|A| > 1$
  - $t$ is the “target” of the attack

- Object of the attack
  - let $\rho_t$ be the rate at which $t$ is recommended to users
  - Goal of the attacker
    - either $\rho'_t \gg \rho_t$ (push attack)
    - or $\rho'_t \ll \rho_t$ (nuke attack)
    - $\Delta\rho = "Hit \ rate \ increase"$
    - (usually $\rho_t$ is $\approx 0$)

- Or alternatively
  - let $r_t$ be the average rating that the system gives to item $t$
  - Goal of the attacker
    - $r'_t \gg r_t$ (push attack)
    - $r'_t \ll r_t$ (nuke attack)
    - $\Delta_r = "Prediction \ shift"$
Approach

- Assume attacker is interested in maximum impact
  - for any given attack size $k = |A|$
  - want the largest $\Delta_p$ or $\Delta_r$ possible
- Assume the attacker knows the algorithm
  - no “security through obscurity”
- What is the most effective attack an informed attacker could make?
  - reverse engineer the algorithm
  - create profiles that will “move” the algorithm as much as possible
But

- What if the attacker deviates from the “optimal attack”?
- If the attack deviates a lot
  - it will have to be larger to achieve the same impact
- Really large attacks can be detected and defeated relatively easily
  - more like denial of service
"Box out" the attacker
Characterizing attacks I

- $R(u,j)$
  - rating by user $u$ for item $j$
  - $n$ users
  - $m$ items
- $t$ is the our target item
- $a$ is an attacking user

<table>
<thead>
<tr>
<th>item_1</th>
<th>item_2</th>
<th>...</th>
<th>item_t</th>
<th>...</th>
<th>item_m</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R(a,1)$</td>
<td>$R(a,2)$</td>
<td>...</td>
<td>$R(a,t)$</td>
<td>...</td>
<td>$R(a,m)$</td>
</tr>
</tbody>
</table>
Characterizing attacks II

- **Items**
  - Some will not have ratings
    - otherwise attack profiles would be very easy to recognize
  - Some will be chosen specifically to enable the attack
  - Others will be chosen randomly (filler items)

- **Four sets of items**
  - $i_t$ = target item (singleton)
  - $I_S$ = attack-specific items
  - $I_F$ = filler items
  - $I_0$ = unrated items
Characterizing attacks III

- $R(a,j)$ is a function of what set item $j$ belongs to
- $j = i_t$
  - $R(a, j) = R_{\text{max}}$ (for push)
  - $= R_{\text{min}}$ (for nuke)
  - (remember maximum impact assumption)
- $j \in I_S$
  - $f_S(j)$
- $j \in I_F$
  - usually randomly chosen
  - $f_F(j)$
    - usually a random value
### Characterizing attacks IV

![Diagram of attacks]

<table>
<thead>
<tr>
<th>$I_t$</th>
<th>$i_{S1}$</th>
<th>...</th>
<th>$i_{Sj}$</th>
<th>$i_{F1}$</th>
<th>...</th>
<th>$i_{Fk}$</th>
<th>$i_{01}$</th>
<th>...</th>
<th>$i_{0l}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_{max}$ or $R_{min}$</td>
<td>$f_s(i_{S1})$</td>
<td>...</td>
<td>$f_F(i_{F1})$</td>
<td>...</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Characterizing attacks V

- To describe an attack
  - indicate push or nuke
  - describe how $I_S$, $I_F$ are selected
  - Specify how $f_S$ and $f_F$ are computed

- But usually
  - $I_F$ is chosen randomly
  - only interesting question is $|I_F|$ ("filler size")
  - expressed as a percentage of profile size

- Also
  - we need multiple profiles
  - $|A|$ ("attack size")
  - expressed as a percentage of database size
Reverse Engineering

- **Attacker’s ideal**
  - every real user has enough neighboring attack profiles
  - That the prediction for the target item is influenced in the right direction

- **Assume**
  - attacker does not have access to profile database P
  - attacker wants to minimize |A|

- **Idea**
  - approximate “average user”
  - ensure similarity to this average
Basic attacks

- **Random attack**
  - Simplest way to create profiles
  - No “special” items ($|I_S| = 0$)
  - $I_F$ chosen randomly for each profile
  - $f_F$ is a random value with mean and standard deviation drawn from the existing profiles $P$
  - Simple, but not particularly effective -- real users aren’t random

- **Average attack**
  - No “special” items ($|I_S| = 0$)
  - $I_F$ chosen randomly for each profile
  - $f_F (i) = a$ random value different for each item
    - drawn from a distribution with the same mean and standard deviation as the real ratings of $i$
  - Quite effective -- more likely to correlate with existing users
  - But
    - would attacker have the necessary data for all $i$?
    - knowledge-intensive attack - could be defeated by hiding data distribution

[Jam & Riedl, 2004]
Bandwagon attack

- Build profiles using popular items with lots of raters
  - frequently-rated items are usually highly-rated items
  - getting at the “average user” without knowing the data
- Special items are highly popular items
  - “best sellers” / “blockbuster movies”
  - can be determined outside of the system
  - $f_S = R_{\text{max}}$
- Filler items as in Random Attack
- Almost as effective as Average Attack
  - But requiring little system-specific knowledge
Typical Results
Item-based recommendation

- Item-based collaborative recommendation
  - collaborative data, but compares items rather than users

- Can be more efficient
  - but also more robust against the average / bandwagon attacks
  - “algorithmic response”

- But, is vulnerable to targeted attacks (next)
Targeted Attacks: Segment attack

- Not all users are equally “valuable” targets.
- Attacker may not want to give recommendations to the “average” user.
  - but rather to a specific subset of users.
- Idea
  - differentially attack users with a preference for certain classes of items.
  - people who have rated the popular items in particular categories.
- Can be determined outside of the system.
  - the attacker would know his market.
  - “Horror films”, “Children’s fantasy novels”, etc.
Segment attack

- Identify items closely related to target item
  - select most salient (likely to be rated) examples
    - “Top Ten of X” list
  - Let $I_S$ be these items
  - $f_S = R_{\text{max}}$

- These items define the user segment
  - $V =$ users who have high ratings for $I_S$ items
  - evaluate $\Delta_p(v)$ on $V$, rather than $U$
Results (segment attack)

Using Item-based Collaborative Filtering

![Graph showing Hit Ratio vs # of Recommendations for In-segment (1%), All users (1%), and Baseline.]
Nuke attacks

- Interesting result
  - asymmetry between push and nuke
  - especially with respect to $\Delta_p$
  - it is easy to make something rarely recommended
- Some attacks don’t work
  - Reverse Bandwagon ($f_S = R_{\min}$)
- Some very simple attacks work well
  - Love / Hate Attack
    - love everything, hate the target item
    - target item $\rightarrow R_{\min}$
    - filler items $\rightarrow R_{\max}$
Nuke attack results

Baseline $\rho_t$ is usually so small that $\Delta\rho$ doesn’t differentiate between attacks.

A More sensitive measure: average rank of $i_t$ over all users.
Obfuscated Attacks

- What about the middle part of the figure?
  - How big is the hole?
- Small amounts of deviation from known attack types
  - esp. using $R_{max} = 4$ instead of 5
  - do not impact attack effectiveness much
    - About 10-20%
  - But do reduce effectiveness of detection
    - About 20%
- System trained only on known types
  - future work: additional training with wider range of attacks
Basic Findings

- Possible to craft an effective attack regardless of algorithm
- Possible to craft an effective attack even in the absence of system-specific knowledge
- Relatively small attacks effective
  - 1% for some attacks
  - smaller if item is rated sparsely
Attacks Against Social Tagging Systems

- Recommender-like collaborative systems
  - System behavior depends on user input
  - Does tag spam look like profile injection?
  - How to characterize / defend against it?

- Tagging Systems (Folksonomies)
  - Del.icio.us / Flikr / Last.fm
    - allow users to tag items with arbitrary text labels
  - Multi-dimensional labels -- more complex than ratings
  - More complex navigation channels /output
    - Tag \(\rightarrow\) resources
    - Resource \(\rightarrow\) resources
    - etc.
Some (Selected) Related Work

- Xu et al. 2006
  - Criteria for good tagging systems; Account for spam with user reputation score

- Koutrika et al. 2007
  - Theoretical framework for tagging attacks

- Krause et al. 2008
  - Machine learning approaches to identify spammers in a social bookmarking system

- Sandvig et al. and Ramezani et al. 2008
  - Dimensions of an attack, different types of attacks
  - Framework for navigating social tagging systems, navigation channels
Navigation Channels in Folksonomies

**Navigation Context**
- Specifies the current user context while navigating the folksonomy graph
- Can be a tag, a resource, or a user

**Output Element Type**
- What will be returned as a results of current navigation context
- Can be ranked lists of tags, resources, or users
- Depends on the specific retrieval algorithm used

**Attacks**
- Attacks types against the system can be understood and analyzed with respect to specific navigation channels
- pushing a resource when a tag is clicked versus when a resource is selected
Navigation Channels and Attack Types

<table>
<thead>
<tr>
<th>Navigation Context</th>
<th>Target Element Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resource</td>
<td>Resource: Related Resources&lt;br&gt;Attack: Piggyback</td>
</tr>
<tr>
<td>Tag</td>
<td>Tag: Common Tags&lt;br&gt;Attack: Coattail</td>
</tr>
<tr>
<td>User</td>
<td>User: Posing History&lt;br&gt;Attack: Pivot Point</td>
</tr>
<tr>
<td>Resource</td>
<td>Output: Popular Items&lt;br&gt;Attack: Overload</td>
</tr>
<tr>
<td>Tag</td>
<td>Output: Related Tags&lt;br&gt;Attack: Co-Occurrence</td>
</tr>
<tr>
<td>User</td>
<td>Output: Active Users&lt;br&gt;Attack: Pivot Point</td>
</tr>
<tr>
<td>Resource</td>
<td>Output: Recent Items&lt;br&gt;Attack: Mole (&quot;Shill User&quot;)</td>
</tr>
<tr>
<td>Tag</td>
<td>Output: Tag Cloud&lt;br&gt;Attack: Mole (&quot;Shill User&quot;)</td>
</tr>
</tbody>
</table>
Piggyback Attack

- The goal of a piggyback attack is to associate the target resource with other resources, such that they appear similar.

- **Tag duplication strategy**
  - Adding attack profiles to the system that associate the target resource with a set of tags which are the most frequently used tags for the selected popular resource.

<table>
<thead>
<tr>
<th>Resource-Tag matrix</th>
<th>Coffee</th>
<th>starbucks</th>
<th>cafe</th>
<th>design</th>
<th>web</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL1</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>URL2</td>
<td></td>
<td></td>
<td></td>
<td>50</td>
<td>30</td>
</tr>
<tr>
<td>URL3</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>URL4</td>
<td>10</td>
<td>20</td>
<td>5</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>URL5</td>
<td></td>
<td></td>
<td></td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>Target URL (Piggyback attack)</td>
<td></td>
<td></td>
<td></td>
<td>20</td>
<td>20</td>
</tr>
</tbody>
</table>
Overload Attack

- The goal of an overload attack is to associate a set of tags with a target resource so that the system retrieves the target resource when users search for the selected set of tags.

- **Popular Overload Attack**
  Aimed at all users: most popular tags are associated with target resource

- **Focused Overload Attack**
  Aimed at a subset of users: locally popular tags are associated with a resource.

<table>
<thead>
<tr>
<th>Resource-Tag matrix</th>
<th>Coffee</th>
<th>starbucks</th>
<th>café</th>
<th>design</th>
<th>web</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL1</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>URL2</td>
<td></td>
<td></td>
<td></td>
<td>50</td>
<td>30</td>
</tr>
<tr>
<td>URL3</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>URL4</td>
<td>10</td>
<td>20</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>URL5</td>
<td></td>
<td></td>
<td></td>
<td>50</td>
<td>50</td>
</tr>
</tbody>
</table>

Target URL (Popular overload attack) | Coffee | starbucks | café | design | web |
-------------------------------------|--------|-----------|------|--------|-----|
                                  | 10     | 10        | 10   |        |     |

Target URL (Focused overload attack) | Coffee | starbucks | café | design | web |
-------------------------------------|--------|-----------|------|--------|-----|
                                  | 10     | 10        | 10   |        |     |
A Methodological Note

- Using data from del.icio.us and bibsonomy
- Data has a long tail distribution, partition the data using coefficient of variation
  - LD: Low Frequency Resources
  - MD: Medium Frequency Resources
  - HD: High Frequency Resources
- Model effects of different attacks
  - Overload attack (Popular and Focused attacks)
  - Piggyback attack
- Retrieval Algorithms
  - Tag Frequency (TF)
- Measure impact of attacks
  - Rank Improvement: Difference between inverse rank before and after attack
  - PageRank: adapted to folksonomy graph; used to measure the global impact
Sample Results (RI): Overload

- Imp: Difference between inverse rank before and after attack

\[ \text{Imp} = \frac{1}{r'} - \frac{1}{r} \]

**Popular Overload Attack**

**Focused Overload Attack**
Results(RI): Piggyback

MD and HD Resources do not appear to be vulnerable to this type of attack
Global Impact of Attacks

- The \textit{Imp} metric is a local metric
  - measures the impact of an attack on a single channel within a particular navigation context
  - But, attacks have a global impact across the network of users, tags, and resources
- PageRank used as a unified approach to evaluate attacks
  - Hotho et al. 2006: “adapted PageRank” $\rightarrow$ “Folkrank”
  - adapted to the “flattened” version of the tri-partite graph of the folksonomy
  - can measure the change in PR before and after the attack
  - can be used to see the impact globally
Other Attack Types

- **Co-Occurrence:**
  - Correlate a target tag with another popular or focused tag
  - Attack: annotate resources with both tags, such that they always occur together

- **Coattail:**
  - Correlate a target tag with a particular resource context (popular or focused)
  - Attack: annotate the resource with the target tag in every attack profile
Other Attack Types

- Pivot Point
  - Create a strong associations (over time) between an attacker’s profile and its intended audience
  - Correlating the profile with resources and/or tags that are relevant to the targeted user segment
  - A *mole attack* ("shill user") may utilize a pivot point in order to establish the attack profile as a trusted user in the particular user segment
  - The defining characteristic: an indirect link to the actual target element – goal is to raise the visibility of the attacker’s profile itself, which contains the target resource or tag
The *Imp* metric is a local metric
- measures the impact of an attack on a single channel within a particular navigation context
- But, attacks have a global impact across the network of users, tags, and resources

PageRank used as a unified approach to evaluate attacks
- adapted to the tri-partite graph of the folksonomy
- can measure the change in PR before and after the attack
- can be used to see the impact globally
Sample Results: Popular Overload I

Global Impact of Popular Overload Attack
(one popular tag in each attack profile)
Sample Results: Popular Overload II

Global Impact of Popular Overload Attack
(Using n popular tags in each profile)
Sample Results: Co-Occurrence

Global impact of Co-occurrence attack with 1 popular tag and 1 resource in each attack profile
Possible Solutions?

- We can try to keep attackers (and all users) from creating lots of profiles
  - pragmatic solution
  - but the sparsity trade-off?
- We can build better algorithms
  - if we can achieve lower $\Delta_p$
  - without lower accuracy
  - algorithmic solution
- We can try to weed out the attack profiles from the database
  - reactive solution
Algorithmic responses

- Hybrid algorithms
  - incorporate additional knowledge sources
    - less likely to be attacked
- Model-based algorithms
  - develop a model of the profiles
    - try to isolate the impact of attackers
    - for example, cluster attackers together
- Trust-based algorithms
  - build neighborhoods based on trust relations
    - rather than statistical measures
  - recommendation impact a function of accumulated reputation
    - Resnick & Sami, 2007
    - More work needed
Example:
Model-Based Results

Top 10 Recommendations
average attack w/ 5% filler

Top 10 Recommendations
segment attack (in segment) w/ 5% filler

© Mobasher, Nasraoui, Spiliopoulou, Zaïane - ACM SIGKDD 2009, Paris, June 28
Detection and response

- **Goal**
  - classify users into attackers / genuine users
  - but remember definition
    - An attacker is a profile that is part of a large group A
  - Then ignore A when making predictions
Unsupervised Classification

- Clustering is the basic idea
  - Reduced dimensional space
  - Attacks cluster together

- Mehta, 2007
  - PCA compression
  - Identify users highly similar
    - In lower-dimensional space
  - Works well for average attack
    - At higher attack sizes
    - > 90% precision and recall
    - Computationally expensive
Supervised Classification

- Identify characteristic features likely to discriminate between users and attackers
  - Example
    - profile variance
    - target focus
  - Total of 25 derived attributes
- Learn a classifier over labeled examples of attacks and genuine data
  - Best results with SVM
- Detection is low-cost
Methodology

- Divide ratings database into test data and training data
  - $U_T$ and $U_R$
- Add attacks to $U_R$
  - $U_R + A_R = U_R'$
- Train the classifier on $U_R'$
- Test performance against
  - $U_T + A_T = U_T'$
  - where $A_T$ uses a different set of target items
Stratified Training

- We want to train against multiple attack types and sizes
  - $A_R = A_1 + A_2 + \ldots + A_n$
  - $A_R$ must be large to include all combinations
  - But if $A_R$ is too big relative to $U_R$
  - Then derived features are biased
    - Attack profiles become “normal”
- Let $F(U,u)$ be the features derived from a profile $u$ in the context of a database $U$
  - instead of calculating $F(U_R’, A_R)$
  - calculate $F(U_R+A_1,A_1)$, $F(U_R+A_2,A_2)$, etc.
  - Then combine resulting features with the training data
SVM Results

Attacks essentially neutralized up to 12%.
Both push and nuke.
Other attack types similar results.
Where are we?

- Attacks work well against all standard collaborative recommendation algorithms
- They work even better against collaborative tagging environments
- What to do (for recommenders)
  - Use e-commerce common sense
    - Protect accounts, if applicable
    - Monitor the system, check up on customer complaints
  - Hide your ratings distribution
  - Use additional knowledge sources if you can
    - hybrid recommendation
  - Use model-based recommendation if computationally feasible
  - Use attack detection
- What to do (folksonomies)
  - A much more difficult question
Open issues

- Real-time detection
  - different from static / matrix-based results?
- Handling cold-start items / users
- Handling large-scale, low impact attacks
- Detecting attacks in folksonomies
- Evaluating robustness of retrieval and recommendation algorithms in folksonomies
Larger question

- Machine learning techniques widespread
  - Recommender systems
  - Social networks
  - Data mining
  - Adaptive sensors
  - …

- Systems learning from open, public input
  - How do these systems function in an adversarial environment?
  - Will similar approaches work for these algorithms?
References I (Block 3)

Bhaumik, Burke & Mobasher. Crawling Attacks Against Web-based Recommender Systems. DMIN’07

Chirita, Nejdl & Zamfir. Preventing shilling attacks in online recommender systems. WIDM ’05.


Lam & Reidl. Shilling Recommender Systems for Fun and Profit. WWW’04.


Mehta, Unsupervised Shilling Detection for Collaborative Filtering. AAAI’07.


Mehta & Nejdl. Attack resistant collaborative filtering. SIGIR’08:

Mobasher, Burke & Sandvig. Model-Based Collaborative Filtering as a Defense Against Profile Injection Attacks. AAAI'06.


O'Mahony, Hurley & Silvestre: Recommender Systems: Attack Types and Strategies. AAAI'05:


Su, Zeng & Chen. Finding group shilling in recommendation system. WWW’05.


Thank you!

Questions?
Presentation Outline

- Block 1: Introduction
- Block 2: Mining the Social Web
- Block 3: Recommendation and Personalization
- **Block 4: Evolution in the Web**
  - The stream mining paradigm
  - Capturing evolution in (Web) document streams
  - Capturing evolution in communities
- Block 5: Mining Web Data Streams
- Block 6: Conclusions and Outlook

(Myra Spiliopoulou)
Presentation Outline

- Block 1: Introduction
- Block 2: Mining the Social Web
- Block 3: Recommendation and Personalization
- Block 4: Evolution in the Web
  - The stream mining paradigm
    - Capturing evolution in (Web) document streams
    - Capturing evolution in communities
- Block 5: Mining Web Data Streams
- Block 6: Conclusions and Outlook
Why Web mining on streams?

- The Web evolves.
  - Many findings about the patterns and laws of evolution for
    - large graphs of servers (e.g. all Internet servers)
    - the Blogosphere
    - communities

Graph Evolution:
Densification and Shrinking Diameters

Jure Leskovec
School of Computer Science, Carnegie Mellon University, Pittsburgh, PA

Jon Kleinberg
Department of Computer Science, Cornell University, Ithaca, NY

Christos Faloutsos
School of Computer Science, Carnegie Mellon University, Pittsburgh, PA

February 2, 2008

Zipf’s law and the Internet

Lada A. Adamic
Bernardo A. Huberman

Abstract. Zipf’s law governs many features of the Internet. Observations of Zipf distributions, while interesting in and of themselves, have strong implications for the design and function of the Internet. The connectivity of Internet routers influences the robustness of the network while the distribution in the number of email contacts affects the spread of email viruses. Even web caching strategies are formulated to account for a Zipf distribution in the number of requests for webpages.
Why Web mining on streams?

- The Web evolves.
  - Many findings about Web growth
  - Few methods that learn from the growing Web
- What do we want to learn?
  - Predict the future

What should we recommend next week?
Why Web mining on streams?

- The Web evolves.
  - Many findings about Web growth
  - Few methods that learn from the growing Web
- What do we want to learn?
  - Predict the future
    - Learning with concept drift (supervised)
  - Understand how the future changes our models

How did LinkedIn change since my last visit?

What topics are becoming hot?
Why Web mining on streams?

- The Web evolves.
  - Many findings about Web growth
  - Few methods that learn from the growing Web
- What do we want to learn?
  - Predict the future
    - Learning with concept drift (supervised)
  - Understand how the future changes our models
    - Adapt to concept drift (unsupervised)
    - Detect change (unsupervised)
Stream mining for the Web

- What is the *stream mining paradigm*?
- Which data do we work upon?
- What methods will we discuss?
The data stream model of computation (Guha et al, 2003)

A stream is a sequence of data points \( x_1, x_2, ..., x_i, ... \) that arrive in increasing order of the index \( i \).

An algorithm operating upon a stream
- is subject to memory constraints
- must minimize the number of passes over the data.

A learning algorithm operating upon a stream
- must maintain a good model of the encountered data
- subject to constraints on time and space.
Maintaining a good model upon *all* the data?

IF the data generating process is *not* stationary
    THEN we want a good model on the *most recent* data.

ië We *must forget* the oldest data.

IF the data generating process is stationary
    THEN *we do not need to remember* the oldest data.
Remembering and forgetting in a stream

A stream is a sequence of data points $x_1, x_2, \ldots, x_i, \ldots$

Window

- consisting of $l$ points

XOR

- spanning $w$ time units
Remembering and forgetting in a stream: Sliding window

The window slides over the data by

- one point at a time
- one unit at a time
Remembering and forgetting in a stream: Sliding window

The window slides over the data by

- multiple points at a time

- multiple units at a time
Remembering and forgetting in a stream: Non-sliding windows

The window does not slide; it stretches over all data

*with a step of*

- one point
- or multiple points
- or one or multiple time units
Only the points inside the window are remembered.

Weighting the points inside the window:

- All points have the same weight.
- Most recent points have a higher weight than earlier ones (according to some function).
- Points similar to the most recent one have higher weight.
The data stream model of computation (Guha et al, 2003)

A stream is a sequence of data points $x_1, x_2, ..., x_i, ...$

A learning algorithm operating upon a stream
- must maintain a good model of the encountered data
- subject to constraints on time and space.

Core approach:

At step $i$ adapt model $\zeta_{i-1}$ into model $\zeta_i$:
- forget data that fall outside the window
- re-compute the weights of the data in the window
- adapt the model to the re-weighted data points
Adapting a model:
Two core methods for stream clustering (1/2)

- IncrementalDBSCAN (Ester et al, VLDB 1998)
  For each newly arriving data point:
  - If there is a cluster to which it can be assigned then insert it into the most appropriate cluster.
  - Identify all points that acquired the core property.
  - Identify all points that lost the core property.
  - Update the clusters affected through changes in the core property of their members (e.g. merge overlapping clusters).
Adapting a model: Two core approaches (2/2)


- Partition the data points
- Compute the center of each partition
- Weight the center of each partition
- Cluster the weighted centers

At each step, perform clustering upon
- the newly arriving data points and
- the centers representing the old data
Stream mining for the Web

- What is the stream mining paradigm?
- Which data do we work upon?
- What methods will we discuss?
  - Learning algorithms for Web streams [Block 5]
  - Dynamic topic models for document streams
  - Dynamic topic models for social network analysis
  - Incremental learning algorithms for social network analysis
Presentation Outline

- Block 1: Introduction
- Block 2: Mining the Social Web
- Block 3: Recommendation and Personalization
- Block 4: Evolution in the Web
  - The stream mining paradigm
  - Capturing evolution in (Web) document streams
  - Capturing evolution in communities
- Block 5: Mining Web Data Streams
- Block 6: Conclusions and Outlook
Documents arriving as a stream:

- news
- scientific publications
- blog entries and
- adjoin discussions

giving raise to questions like:

- What are the topics?
- What is the thread of discussions for a given topic?
- What was the very first contribution to a given topic?
- What topics are becoming hot, which become outdated?
A topic is a set of words that describe a group of documents.

- Documents are vectorized over the feature space of (a selection of) words.
- Vectors are clustered to form groups.
- The representative words for each group are selected according to some function.

For topics over a stream:

- The time axis is discretized.
- Topics are discovered at each time point, using conventional, incremental or stream clustering.
- The topics found at different time points are compared.
A *stream* is a sequence of documents $x_1, x_2, \ldots, x_i, \ldots$ that arrive in increasing order of the index $i$.

**What constitutes the feature space?**

**When and how do we recompute it?**
A *stream* is a sequence of documents $x_1, x_2, ..., x_i, ...$

What constitutes the feature space? When and how do we recompute it?

- The time axis is discretized.
- Topics are discovered at each time point, using conventional, incremental or stream clustering.
- The topics found at different time points are compared.
Example Approach: ThemeMonitor for emerging topics (ADBIS’06 & KDD’06)

At timepoint $i$:

- Build $\zeta_i$ over the feature space $fs$ used thus far.
- For each cluster $C$ in $\zeta_i$ set $Label(C) = \{w \in fs \mid freq(w, C) \geq \tau\}$
- If there are less than $m$ clusters with non-empty label
  Then discard the feature space and re-build $\zeta_i$
- For each label $X$ in $\zeta_{i-1}$ find best match $Y$ in $\zeta_i$
How many labels are there at each $t_i$?

- 1-3 labels on *data mining*
- one label on *image retrieval*
- some shortlived labels

**Persistent labels:**
- image retrieval
- knowledge discovery
- (assoc. rules)

The feature space was changed twice.
A *stream* is a sequence of documents $x_1, x_2, ..., x_i, ...$

- The time axis is discretized.
- Topics are discovered using conventional, incremental or stream clustering.
- The topics found at different time points are compared.

**What constitutes the feature space?**
**When and how do we recompute it?**

**Reliability?!**
Topic evolution meets latent models
Dynamic topic model according to Blei & Lafferty (2006)

Static model with Dirichlet priors $\alpha$ (documents), $\beta$ (topics)

- $\alpha$: document-specific topic proportions
- $\theta$: document-specific topic proportions
- $z$: words (observed)
- $w$: words (observed)
- $N$: number of words
- $A$: number of words
- $K$: number of topics (fixed)

© Mobasher, Nasraoui, Spiliopoulou, Zaïane - ACM SIGKDD 2009, Paris, June 28
Dynamic topic model according to Blei & Lafferty (2006)

\[
\begin{align*}
N_z & \sim \text{Dirichlet}(\alpha) \\
A & \sim \text{Bernoulli}(\beta)
\end{align*}
\]

Method inferring $\alpha, \beta$
Dynamic topic model according to Blei & Lafferty (2006)

$k^\text{th}$ topic at slice $i$ evolves smoothly from $k^\text{th}$ topic of slice $i-1$
Recent examples for topic evolution over streams

- **LDA thread**
  - L. AlSumait et al. On-line LDA: Adaptive Topic Models for Mining Text Streams with Applications to Topic Detection and Tracking (ICDM’08)
  - R. Nallapati et al. Multiscale Topic Tomography (KDD’07)

- **PLSA thread**
  - A. Gohr et al. Topic Evolution in a Stream of Documents (SIAM DM’09)
The Online LDA of AlSumait et al (ICDM’08)

Underpinnings:

- Gibbs sampling for the estimation of
  - document-specific topic proportions $\theta$
  - topic-specific word proportions $\phi$
- Time axis is discretized – sliding window of size $\delta$ slices
- The model generated at slice t-1 is used as prior for LDA for the stream portion $S^t$ in slice t.
Underpinnings:

- The model generated at slice $t$ is used as prior for LDA for the stream portion $S^t$ in slice $t-1$.
  - Stream $S^t$ introduces new words.
  - Parameters of topic $k$ are determined from its past distributions:
    \[ \beta^t_k = B^{t-1}_k \omega^\delta \]

Evolution matrix of topic $k$: has $V^t$ columns, a column is $\phi^i_k$

$V^t$: number of unique words in $S^t$
The Online LDA of AlSumait et al (ICDM’08)

- Evaluation for document modeling
  - Comparison to an LDA method that remembers all data.
  - Superior performance – lower perplexity towards held-out data

- Evaluation on topic evolution
  - Interesting findings on the evolution of NIPS topics
The Adaptive PLSA of Gohr et al (SDM’08)

Core idea

FORGET

FOLD-IN

Old Topics

New Topics

Adapting Topics
The Adaptive PLSA of Gohr et al (SDM’08)

Underpinnings:
- Time axis is discretized – sliding window of size $\delta$ slices
- The model generated at slice $t-1$ is used as basis for PLSA adaptation for the stream portion $S^t$ in slice $t$.
  - Old documents are forgotten, new documents are folded-in
  - Old words are forgotten new words are folded-in.
The Adaptive PLSA of Gohr et al (SDM’08)

- Evaluation for document modeling
  - Comparison to a PLSA method that re-learns from scratch
  - Experiments on SIGIR (2000-2007)
  - Superior performance – lower perplexity towards held-out data

- Evaluation on topic evolution
  - Interesting findings on the evolution of SIGIR topics
The Adaptive PLSA of Gohr et al (SDM’08)
Topic threads in SIGIR (2000-2007)

Evaluation

Presentation, later emphasis on multimedia

Supervised learning

Web

Clustering
Presentation Outline

- Block 1: Introduction
- Block 2: Mining the Social Web
- Block 3: Recommendation and Personalization
- Block 4: Evolution in the Web
  - The stream mining paradigm
  - Capturing evolution in (Web) document streams
  - Capturing evolution in communities
- Block 5: Mining Web Data Streams
- Block 6: Conclusions and Outlook
Communities over time

- Social networks evolve:
  - New individuals (vertices) enter the scene.
  - Some individuals become inactive.
  - An interaction takes place at some time point and has practical relevance for some time only.

- Set of interactions → Stream of interactions
  - Discretization of the time axis
  - Window-based forgetting mechanism
  - Model learning/adaptation
What is a community over time?

- A group whose members remain “mostly” the same
- A group that keeps the same characteristics over time
- A sequence of similar groups across the time axis
- A (soft) cluster that evolves smoothly over time
Definitions & algorithms for dynamic communities

- A group whose members remain “mostly” the same
- A group that keeps the same characteristics over time
- A sequence of similar groups across the time axis
  - Group discovery at each time point
  - Group matching between/across time points

*Examples:*
- Berger-Wolf & Saia (KDD’06)
- Falkowski et al (Web Intelligence’06)

- A (soft) cluster that evolves smoothly over time
A community as a group of community instances (Falkowski et al, WI’06)

Step 1. Discretization of the time axis

Step 2. Discovery of community instances at each time point: divisive clustering using edge betweenness

Step 3. Identification of similar community instances within each time window

Step 4. Visualization of similar community instances

Step 5. Building communities as clusters of similar community instances
Statistics for a community instance (Falkowski et al, WI‘06)
Building and visualizing communities: Experiments with a students’ site (Falkowski et al, WI’06)

Number of clustering iterations (= number of edges removed):

0  27  38  48

© Mobasher, Nasraoui, Spiliopoulou, Zaïane - ACM SIGKDD 2009, Paris, June 28
Definitions & algorithms for dynamic communities

- A group whose members remain “mostly” the same
- A group that keeps the same characteristics over time
  - Basic set of assumptions
  - Algorithm that discovers *and* tracks groups
- A sequence of similar groups across the time axis
- A (soft) cluster that evolves smoothly over time
The framework of Tanipathananandh et al (KDD’07)

Core assumptions:

1. In each time point, every group stands for a distinct community.
2. In each time point, an individual belongs to exactly one community and
3. tends not to change its community very frequently.
4. If an individual does change its community often, it will usually be an oscillation among a small number of communities, rather than promiscuity among many.
5. An individual is frequently present in its group. It rarely misses being with group, and rarely is in other groups.
The framework of Tanipathananandh et al (KDD’07)

Graph model:

- one *individual vertex* per individual $i$ at each time point $t$
- one *group vertex* for each group existing at $t$
- one edge between each individual and its group at $t$
- one edge between the vertices of each individual at $t, t+1$

Group at $t$:
Set of individuals that have been found to interact with each other at $t$.
Groups at $t$ are disjoint.
Community interpretation of the graph as \textit{vertex coloring}:

At time point $t$

- the color of an \textit{individual vertex} $v_{i,t}$ represents the community affiliation of individual $i$ at $t$,

- the color of a \textit{group vertex} $v_{g,t}$ represents the community represented by group $g$ at $t$.

A community interpretation is \textit{valid} iff there is no time point at which two groups have the same color.
The framework of Tanipathananandh et al (KDD’07)

Cost of violating a postulate:

1. In each time point, every group stands for a distinct community.
2. In each time point, an individual belongs to exactly one community and tends not to change its community.
3. If an individual does change its community often, it will usually be an oscillation among a small number of communities, rather than promiscuity among many.
4. An individual is frequently present in its group. It rarely misses being with group, i-cost for color change
g-cost for discrepancies between the color of i and the color of its group (if any)
c-cost for using more than one colors
The framework of Tanipathanananandh et al (KDD’07)

Optimization problem:

Find a community interpretation, i.e. a coloring that is valid and minimizes the total cost.

Community over time:

- Sequence of groups that have the color of the community.

Affiliation sequence of an individual:

- Sequence of communities to which the individual belonged.

NP-complete problem

Heuristics based on Dynamic Programming
The framework of Tanipathananandh et al (KDD’07)

Experimental results:

- Validation on synthetic datasets that simulate social networks with specific properties
- Evaluation against datasets with a ground truth

Applicability for communities in the Web?

Core assumptions:

1. In each time point, every group stands for a distinct community.
2. In each time point, an individual belongs to exactly one community and
tends not to change its community very frequently.
3. If an individual does change its community often, it will usually be an oscillation among a small number of communities, rather than promiscuity among many.
4. An individual is frequently present in its group. It rarely misses being with group, and rarely is in other groups.
Definitions & algorithms for dynamic communities

- A group that keeps the same characteristics over time

*Examples for dynamic online communities:*
  - Falkowski et al (AMCIS’08):
    - Incremental density-based graph clustering for community discovery and adaptation
  - Tang et al (KDD’08):
    - Community as a soft cluster over a multi-mode network
    - Spectral clustering
  - Lin et al (WWW’08) & Yang et al (SIAM DM’09)
    - Probabilistic framework
  - A (soft) cluster that evolves smoothly over time
Definitions & algorithms for dynamic communities

- A group that keeps the same characteristics over time
- A (soft) cluster that evolves smoothly over time

Assumptions about the data generating process
Algorithm that builds and adapts a latent model

Examples:
- Lin et al (WWW’08)
  - Cost model that penalizes dramatic changes from $t-1$ to $t$
  - First order Markov model (Dirichlet priors)
- Yang et al (SIAM DM’08)
  - Dynamic stochastic block model with Bayesian inference
    (Dirichlet priors, Gibbs sampling) for membership prediction
Trends in community evolution?

- Capturing the dynamics
  - Dynamic block models
  - Bayesian inference
  - Markov models (1\textsuperscript{st}, k\textsuperscript{th} order)
  - Assumptions about priors
- Allowing for observed properties of online communities
  - Overlapping communities – soft clusters
  - Authorities \rightarrow Findings from the analysis of large networks
- Dealing with evolution (drifts and shifts)
References (Block 4):
Stream mining paradigm


References (Block 4):
Topic evolution

- Q. Mei, C. Zhai. Discovering evolutionary theme patterns from text - an exploration of temporal text mining. 11th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD'05), 198-207, Chicago, IL, Aug. 2005.
- R. Nallapati, S. Ditmore, J.D. Lafferty, and K. Ung. Multiscale topic tomography. 13th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD'07), San Jose, CA, Aug. 2007.
- M. Spiliopoulou, I. Ntoutsi, Y. Theodoridis, R. Schult. MONIC -- Modeling and Monitoring Cluster Transitions. 12th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD’06), 706-711, Philadelphia, USA, Aug. 2006
References (Block 4): Community evolution

- T. Falkowski, J. Bartelheimer, M. Spiliopoulou. Mining and visualizing the evolution of subgroups in social networks. 2006 IEEE/WIC/ACM Int. Conf. on Web Intelligence (WI’06), 52–58, Dec. 06.
- C. Tanipathananandh, T. Berger-Wolf, D. Kempe. A Framework for Community Identification in Dynamic Social Networks. 13th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD’07), San Jose, CA, Aug. 2007.
Thank you!

Questions?
Presentation Outline

- Block 1: Introduction
- Block 2: Mining the Social Web
- Block 3: Recommendation and Personalization
- Block 4: Evolution in the Web
- Block 5: Mining Web Data Streams (focus on WUM) (Olfa Nasraoui)
  - Stream Data Mining and handling concept drift
  - Web Data Stream Mining techniques for clickstream data
  - Evaluation of Web Data Stream Mining
- Block 6: Conclusions and Outlook
Presentation Outline

- Block 1: Introduction
- Block 2: Mining the Social Web
- Block 3: Recommendation and Personalization
- Block 4: Evolution in the Web
- Block 5: Mining Web Data Streams (focus on WUM)
  - Stream Data Mining and handling concept drift
  - Web Data Stream Mining techniques for clickstream data
  - Evaluation of Web Data Stream Mining
- Block 6: Conclusions and Outlook
Stream Data Mining & Handling Concept Drift

- **Online Learning**: update model as new data arrives
- **Stream Data Mining** (previous Block):
  - Subject to memory constraints: common for web log data of a couple hours NOT to fit in main memory
  - Minimize number of passes over data
  - Good model must be maintained subject to time @ space constraints
- **Concept Drift** (older concept, Machine Learning community)
  - Data becomes available in batches.
    - Past training examples may become obsolete
  - Should handle data or concept evolution (concept drift).
  - Adds another layer of difficulty to online learning
Concept Drift

- **Supervised**: Changes in the context underlying the target, concept variable variable (Widmer & Kubat, 96) (Gama & Castillo, 06)

- **Unsupervised**: Changes in the distribution of data or learned model (e.g. changing clusters, association rules, etc)

- Drifting concept ➔ deterioration of the learned model
  - Need to get rid of outdated information
  - Base our model on the most recent data instances that belong to the new concept.
  - Applying a moving window on the data stream: common

- Almost always: Stream Data Mining ➔ Concept Drift !!!

- But not vice versa (SDM: massive data ➔ constraints)
Concept Drift – sliding window?

- Window size must capture a sufficient amount of “good” instances.
- A dynamic window size, adjusted to the changing context → need change detection
- Alternative: exponentially discount old data instances and update models accordingly
Concept Drift – Classification with regard to adaptation to change

- **Evolutionary scheme:**
  - modifies *existing* knowledge based on completely new training examples, e.g. STAGGER: (Schlimmer & Granger, 86)

- **Revolutionary scheme:**
  - discards old knowledge and learns new knowledge from the new training examples, e.g. window based techniques (Widmer & Kubat, 96)

- **Hybrids:** inherit from both approaches
  - Mitchell’s Calendar Learning Apprentice (Mitchell et al., 94):
    - learns new decision rules from training data and
    - incorporates these new rules into the existing knowledge-base
Concept Drift - Classification with regard to which data is used for learning models

(Maloof & Michalski, 95) further classified the way online learning systems work into three different modes:

- **no-memory**: system uses NO past training examples for updating current model e.g. STAGGER (Schlimmer & Granger, 86)
- **partial-memory**: a subset of the previously seen training examples is used for later learning
- **full-memory**: all past training examples are used for updating an existing model

Apart from (Grabtree & Soltysiak, 98) (user profiling limited to a small number of attributes and users), all the above approaches were

- Either within a *supervised* learning framework,
- Or focused on adaptation to *single* user (predicting an object’s relevance)
Presentation Outline

- Block 1: Introduction
- Block 2: Mining the Social Web
- Block 3: Recommendation and Personalization
- Block 4: Evolution in the Web
- Block 5: Mining Web Data Streams (focus on WUM)
  - Stream Data Mining and handling concept drift
  - Web Data Stream Mining techniques for clickstream data
  - Evaluation of Web Data Stream Mining
- Block 6: Conclusions and Outlook
Background on WUM

- Clickstream data
  - Footprints of interaction logged by Web or application servers (or on client, although less common)
  - E.g. Web Log data = Web user access data
- Web Usage Mining (WUM)
  - Data mining to automatically extract knowledge from clickstream data
- WUM Applications:
  - Customer profiling, Personalization, Product and content recommendation, etc
WUM Process

User interactions ➔ preprocessed

➔ continuously fed into online knowledge discovery systems

➔ update the models in real-time.

Model outputs used to support a variety of applications (e.g. personalization)

NOTE: 3 phases
WUM Stages

Source [Hofgesang’08].
WUM Methods (most common)

- Frequent itemset and association rule mining
- Sequential pattern mining
- Clustering
- Classification
- Personalization (can be a 2nd phase of above, or can be independent, e.g. collaborative filtering)
Major Challenges in Online WUM

- **Stream Data Mining**: Maintaining compact models, scalability

- **Change**:
  - Detection: to keep updated model
  - Characterization: model is not enough, $n$ models are not enough $\Rightarrow$ understanding evolution patterns

- **Evaluation**

- **Data related challenges**:
  - Maintaining page mapping: Pages added/deleted
    $\Rightarrow$ Old models in terms of old pages? map based on content?
  - New types of websites (e.g. dynamic websites, AJAX)
    - Interaction may be stored on App. Servers
Major Challenges in Online WUM (to discuss)

- Stream Data Mining: Maintaining compact models, scalability
- Change:
  - Detection: to keep updated model
  - Characterization: model is not enough, \( n \) models are not enough \( \Rightarrow \) understanding evolution patterns
- Evaluation

- Data related challenges:
  - Maintaining page mapping: Pages added/deleted
    \( \Rightarrow \) Old models in terms of old pages? map based on content?
  - New types of websites (e.g. dynamic websites, AJAX)
    - Interaction may be stored on App. Servers
Compact Data Structures (Offline: existed before Streams)

- Help with the constraints of:
  - Memory
  - Fast retrieval and matching

- Tree-like structures: most common is Prefix-tree: each node $n_i$ represents an itemset $I$ and each child node of $n_i$ represents an itemset obtained by adding a new item to $I$

- Designed to store Market Basket / unordered transactions data:
  - FP-Tree (Han et al., 2000)
  - CATS Tree (for incremental mining) (Cheung&Zaiane, 2003),
  - FP-stream (Giannella et al., 2003)

- Designed to store sequences / ordered transactions data:
  - CST (Gündüz-Öğüdücü, and Özsu, 2006)
  - TKP-FOREST (Li et al., 2005)
Online WUM Algorithms

- Frequent (top-k) items, itemsets, and association rule mining
- Sequential pattern mining
- Clustering
- Web usage mining and personalization
- Good survey: (Hofgesang, 2008)
Frequent Itemset Mining in a Stream: Challenges & Approaches

- As new data arrives:
  - Previously infrequent patterns may become frequent
  - Previously frequent patterns may become infrequent

- Mining Currently Frequent Itemsets:
  - always focus on frequent itemsets in most recent window
  - exponentially discounting old itemsets
Frequent Itemset Mining in a Stream: Challenges & Approaches

- **Window-based approach**: 2 methods:
  1. Regenerate frequent itemsets from the entire window whenever a new transaction comes / leaves the window.
  2. Store every itemset (frequent or not) compact structure and update its support whenever a new transaction comes / leaves the window.

  - **Method 1**: inefficient, most itemsets maintain same (in)frequency status unless drastic concept drift.
  - **Method 2**: incremental. However, its space requirement makes it infeasible in practice:
    - In FP-tree: total number of nodes is exponential.
    - Due to memory constraints, we cannot keep a prefix tree in memory.
    - Disk-based structures will make real time update costly.
Incremental But Requiring > 1 Pass (not suitable for Streams)

- FUP-based algorithms (Cheung et al., 96); (Cheung et al., 97)
- BORDERS algorithm (Aumann et al., 99)
- DEMON algorithm (Ganti et al., 2000)
- ZIGZAG algorithm (Veloso et al., 2002)

- These algorithms focus on efficiently using the previous mining result of a data set to update the results after new transactions are added.
- However, they require multiple scans of a database,
  - and the transactions of a target data set need to be re-scanned even if only one transaction is newly added!
1-Pass: **Currently** Frequent Itemsets: Moment (Chi et al., 2004)

- CET: Closed Enumeration Tree

- Focus on a *dynamically selected* set of itemsets that are
  - i) **informative enough** to answer at any time queries such as “what are the (closed) frequent itemsets in the current window”, and at the same time,
  - ii) **small enough** so that they can be easily maintained in memory and updated in real time

- Problem: which itemsets shall be selected for this purpose?

- A *status change* of any itemset (e.g. from non-frequent to frequent) must occur through the boundary
  - ➔ Concept drift in a data stream: reflected by boundary movements in the CET!!!
1-Pass: **Currently** Frequent Itemsets: Moment (Chi et al., 2004)

- Example with Closed Enumeration Tree from (Chi et al.’04)
1-Pass: Currently Frequent Itemsets: Moment (Chi et al., 2004)

- An itemset is **closed** if none of its proper supersets has the same support.
- The total number of closed frequent itemsets $C << \text{frequent itemsets}$.
- **Moment**: An algorithm that mines closed frequent itemsets over data stream sliding windows.
- The problem of mining an infinite amount of data is thus converted to mine data that can potentially change the boundary in the current model.
- Most itemsets do not often change status $\Rightarrow$ boundary is stable.
- even if some does, the boundary movement is local $\Rightarrow$ cost of mining closed frequent itemsets dramatically reduced!
1-Pass: Frequent Itemsets: estWin (Chang & Lee, 2005)

- Finds recently frequent itemsets over a data stream using a sliding window method (estWin).
- **Size** of sliding window \(\rightarrow\) desired **life-time of information** in a newly generated transaction

**Parameters:**
- **size** of a sliding window \(w\)
- **minimum support** \(S_{\text{min}}\) (in current sliding window) \(\rightarrow\) recently frequent
- **significant support** \(S_{\text{sig}}\) (in current sliding window) \(\rightarrow\) significantly frequent

- Only recently generated transactions in the range of the window are considered to find the set of **recently frequent itemsets**
1-Pass: Frequent Itemsets: estWin (Chang & Lee, 2005)

- All transactions in the current window are maintained in a structure called a *current transaction list CTL*
- Only **Significant** itemsets, are maintained in main memory in a lexicographic tree structure, called *monitoring tree*
- As old transaction leaves window ➔ decrease the occurrence count of each itemset that appeared in the transaction
- The number of itemsets in main memory is limited by
  - **Delayed insertion** of a new itemset until the itemset becomes significant enough to be monitored
  - **Pruning** of a monitored itemset when it turns out to be insignificant.
1-Pass: Frequent Itemsets – (Rojas & Nasraoui, 2007)

- Construct and maintain efficiently (in 1 pass) a compact summary transactional data (Web logs, text streams, etc)
- Prefix tree with dynamic item-ordering criterion such as:
  - Activity time stamp or
  - Attribute frequency or importance
- Two basic tree tasks with every new record from the stream:
  - **Structural updating:** from a change in the ranking of the attributes which may reflect a change in the order of the nodes ➔ may affect different branches.
  - **Count updating:** done only for the branch that corresponds to the current record
- Pruning & Forgetting of nodes (e.g. based on ranking criterion)
Clustering Web ClickStreams

- **Major challenge**: How to handle evolving clusters?
- (Barbara, 2008): Requirements for clustering data streams
- Incremental DBSCAN (Ester, 1998) – (in previous Block)
- TECNO-STREAMS (Nasraoui et al., 2003):
  - 1-pass clustering that learns an evolving stream synopsis from noisy data (each synopsis node = 1 cluster)
  - Cluster parameters evolve continuously as new data arrives via exponentially decaying weighting factor on data instances ➔ dynamic windowing effect
- MONIC (Spiliopoulou et al, 2006): (in previous Block)
  - Offline clustering is applied periodically on accumulating data
  - Cluster transitions: emergence, disappearance, migration
Clustering: (Barbará, 2008) – Requirements of Clustering Web Data Streams

- (Barbara, 2008)
- Compactness of representation
  - Compact cluster representation
  - No increase in size with time (even a linear growth is intolerable)
- Fast, incremental processing of new data points
- Clear and fast identification of "outliers"
- Being able to handle concept drift:
  - cluster evolution, emerging and disappearing clusters
Clustering: (Barbará, 2008) – When to create a new model?

If the algorithm /system is not capable of handling concept drift:

- Statistical hypothesis test set to decide whether the model is still valid or not using:
  - No. of processed points
  - No. of Successfully clustered points
  - Based on Chernoff bounds

- When is it time to create a new model?
  - If after processing \( n \) points, can successfully cluster at least \( s \) of them, the clustering model is still valid
    - Bounds on \( n \) and \( s \) given by Chernoff bounds
  - Otherwise, time to produce a new model
Online Web User Profiling and Personalization-SUGGEST (Baraglia & Silvestri, 2007)

- As HTTP requests arrive at the server:
  1. User navigation sessions \{A\} are built,
  2. Underlying knowledge base is updated in the form of page clusters \{L\},
  3. Active user sessions are classified in one of the clusters,
  4. Finally a list of suggestions \{S\} is generated and appended to the requested page u.

- Models usage information as a complete graph \(G = (V,E)\)
  - \(V\) (vertices) = identifiers of the different pages hosted on the web server
  - \(E\) (edges): weight \(M_{ij}\) = proportional to co-occurrence of pages \(i\) & \(j\)
Online Web User Profiling and Personalization – SUGGEST (Baraglia & Silvestri, 2007)

- Find groups of strongly correlated pages by incrementally partitioning the graph according to its connected components.

- Given an initial partitioning of the graph G into a set of clusters \{L\} and a request page u,
  - need to determine if and how such request modified the cluster partitioning of the graph.

- Start a Depth First Search (DFS) from u on the graph:
  - search for connected component reachable from u.
  - Once component is found, check if there are any nodes in previous component not considered in the visit.
  - If such node exists, then previously connected component has been split, and
  - continue applying DFS until all nodes have been visited.
Online Web User Profiling and Personalization – SUGGEST (Baraglia & Silvestri, 2007)

- Need to
  - Reduce the contributions of poorly represented links
  - Limit the number of edges to visit

- Thus, incremental computation of the connected components is controlled by two threshold parameters:
  - \textbf{minfreq}: Links (i.e. elements $M_{ij}$ of $M$) whose values < \text{minfreq} are poorly correlated and thus not considered by the connected components algorithm
  - \textbf{MinClusterSize}: All components having size < than \text{MinClusterSize} are discarded (considered not significant enough)

- Final recommendations = most relevant pages in cluster with largest intersection with the current session
Online Web User Profiling and Personalization—
(Nasraoui et al., 2007)

- A methodology to test the adaptability of recommender systems in streaming environments (see Evaluation, next...)

- Two strategies for collaborative filtering-based recommendations applied on dynamic, streaming web usage data (sessions), one is lazy, other is model-based, i.e.:
  - **K-Nearest-Neighbors on sliding windows:**
    - Given active session, find K closest sessions in window
    - Recommend top N pages among K –NN data instances
  - **TECNO-STREAMS:**
    - Cluster synopsis: mined in 1 pass over stream (Nasraoui et al., 2003): 1 cluster node = pseudo-session / Profile
    - Given active session, recommend pages from nearest cluster node
Frameworks for Online Change Detection and Evolution Monitoring in Web Clickstreams

- Given Session Data Stream = \( N \) training instances \((x_1, \ldots, x_N)\) split in batches/windows \( W^0, W^1, \ldots, W^T \)
- Online algorithm builds models \( M^0, M^1, \ldots, M^T \)
- Each \( M^t \) \((t = 0, \ldots, T)\) is dependent only on \( M^{t-1} \) and \( W^t \), or only on \( W^t \)
- E.g. of Model \( M^t \):
  - Set of \( N^t \) Association Rules \( R^t_k \) \((k = 1, \ldots, N^t)\)
  - Set of \( N^t \) Clusters \( C^t_k \) \((k = 1, \ldots, N^t)\)
  - Set of \( N^t \) User Profiles \( P^t_k \) \((k = 1, \ldots, N^t)\)
- Evolution monitoring: compare \( M^t \) to past model(s) \( M^{t'} \) \((t' < t)\) ➔ Define rules to map results to types of evolution
Framework for Online Change Detection in Web Clickstreams – PAM (Baron & Spiliopoulou, 2004)

- Track changes of usage patterns (rules discovered in separate batches of 1 month each)
- Each rule stored w/ stats such as support, confidence, timestamps
- Rules from new batch are compared to rules from previous batch:
  - Apply 2-sided binomial test on the stats of the “matching” rules from 2 consecutive periods
  - \[ \rightarrow \text{label the change =} \]
    - short-term change (changed value returns to its prev. state in next test)
    - long-term change (changed value remains same)
Framework for Evolution Monitoring in Web Clickstreams - (Nasraoui et al., 2008)

- Framework for WUM & monitoring evolution/change of user profiles
- Each time period = batch
- Cluster batches of Web clickstream data
  - Web user profile for each cluster
- Compare profiles of different batches
  - 7 Evolution Types:
    - Birth
    - Persistence: Mergal, Bifurcation, 1-to-1
    - Death
    - Atavism
    - Volatility
<table>
<thead>
<tr>
<th>Mass profile evolution event</th>
<th>Condition</th>
<th>Intuitive explanation</th>
<th>Potential implications or meaning in marketing context</th>
</tr>
</thead>
<tbody>
<tr>
<td>Birth</td>
<td>$p_{new} \in P^T$ and for all $k&gt;0$, there is no profile $p_{old}$ $\in P^{T-k}$ such that $\text{Comp}(p_{old}, p_{new}) = \text{TRUE}$</td>
<td>A new profile/group of similar user clicks that has never before been observed</td>
<td>The emergence of a group of customers possessing similar traits may signal an emerging market segment or an emerging product/market combination</td>
</tr>
<tr>
<td>Death</td>
<td>$p_{old} \in P^{T-1}$ and for all $k \geq 0$, there is no profile $p_{new} \in P^{T-k}$ such that $\text{Comp}(p_{old}, p_{new}) = \text{TRUE}$</td>
<td>Opposite of the above: a compact group/cluster of user activities that vanishes</td>
<td>This may signal the decline phase of a segment or product/market combination</td>
</tr>
<tr>
<td>Atavism</td>
<td>$p_{new} \in P^T$ and there is no profile $p_{old} \in P^{T-1}$ such that $\text{Comp}(p_{old}, p_{new}) = \text{TRUE}$, and for some $k&gt;1$, there is a profile $p'<em>{old} \in P^{T-k}$ such that $\text{Comp}(p'</em>{old}, p_{new}) = \text{TRUE}$</td>
<td>A profile that disappears temporarily then re-emerges again at a later period.</td>
<td>This may signal that a segment is transient in nature and dependent upon the specific usage situation of a product</td>
</tr>
<tr>
<td>Persistence (one to one)</td>
<td>$p_{new} \in P^T$ and there is a unique profile $p_{old} \in P^{T-1}$ such that $\text{Comp}(p_{old}, p_{new}) = \text{TRUE}$</td>
<td>the same old profile that is observed to continue on in two consecutive time periods.</td>
<td>This may signal the durability of a market segment as worthy of specific marketing investments.</td>
</tr>
<tr>
<td>Persistence (bifurcation)</td>
<td>$p_{old} \in P^{T-1}$ and there exists more than one profile $p_{new} \in P^T$ such that $\text{Comp}(p_{old}, p_{new}) = \text{TRUE}$</td>
<td>same as above, but the old profile continues in the form of 2 or more new profiles (i.e. splits into several profiles)</td>
<td>This may signal the maturing of a market segment and that consumer experience drives a single segment into multiple sub-segments. These sub-segments may be seeking different benefits from the supplier.</td>
</tr>
<tr>
<td>Persistence (mergal)</td>
<td>$p_{new} \in P^T$ and there is more than one profile $P_{old} \in P^{T-1}$ such that $\text{Comp}(p_{old}, p_{new}) = \text{TRUE}$</td>
<td>Several old profiles merge together to become one profile in a later period</td>
<td>This may signal that preferences are transient, but tend to an agglomeration of preferences toward a single benefit.</td>
</tr>
<tr>
<td>Volatility</td>
<td>$p_{vol} \in P^T$ and for all $k&gt;0$, there is no profile $p_{new} \in P^{T-k}$ such that $\text{Comp}(p_{old}, p_{new}) = \text{TRUE}$, and for some $k&gt;1$, there is a profile $p'<em>{old} \in P^{T-k}$ such that $\text{Comp}(p</em>{vol}, p_{new}) = \text{TRUE}$</td>
<td>a profile that goes through birth, then death in 2 consecutive time periods, hence lives for a unique period.</td>
<td>This may be a signal that a market segment is transient in nature and dependent upon product usage situation.</td>
</tr>
</tbody>
</table>
Framework for Evolution Monitoring in Web Clickstreams - Example (Nasraoui et al., 2008)

<table>
<thead>
<tr>
<th>Profile</th>
<th>June’04</th>
<th>Jul’04</th>
<th>Aug’04-weeks 1-2</th>
<th>Aug’04-weeks 2-4</th>
<th>Sep’04</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Birth</td>
<td>Persistence</td>
<td>Persistence</td>
<td>Persistence</td>
<td>Persistence</td>
</tr>
<tr>
<td>2</td>
<td>Birth</td>
<td>Persistence</td>
<td>Persistence</td>
<td>Persistence</td>
<td>Death</td>
</tr>
<tr>
<td>3</td>
<td>Birth</td>
<td>Persistence</td>
<td>Persistence</td>
<td>Persistence</td>
<td>Death</td>
</tr>
<tr>
<td>4</td>
<td>Birth</td>
<td>Persistence</td>
<td>Persistence</td>
<td></td>
<td>Death</td>
</tr>
<tr>
<td>5</td>
<td>Birth</td>
<td>Persistence</td>
<td></td>
<td></td>
<td>Atavism</td>
</tr>
<tr>
<td>6</td>
<td>Birth</td>
<td>Persistence</td>
<td>Persistence</td>
<td>Persistence</td>
<td>Persistence</td>
</tr>
<tr>
<td>7</td>
<td>Birth</td>
<td>Persistence</td>
<td>Persistence</td>
<td>Persistence</td>
<td>Persistence</td>
</tr>
<tr>
<td>8</td>
<td>Birth</td>
<td></td>
<td></td>
<td>Atavism</td>
<td>Persistence</td>
</tr>
<tr>
<td>9</td>
<td>Birth</td>
<td></td>
<td></td>
<td></td>
<td>Atavism</td>
</tr>
<tr>
<td>10</td>
<td>Birth</td>
<td>Persistence</td>
<td></td>
<td></td>
<td>Death</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Birth</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Birth</td>
</tr>
</tbody>
</table>
Presentación del Plan de la Presentación

✓ Block 1: Introducción
✓ Block 2: Mining the Social Web
✓ Block 3: Recommendation and Personalization
✓ Block 4: Evolution in the Web

- Block 5: Mining Web Data Streams (focus on WUM)
  - Stream Data Mining and handling concept drift
  - Web Data Stream Mining techniques for clickstream data
  - Evaluation of Web Data Stream Mining

- Block 6: Conclusions and Outlook
Problem

- How to evaluate whether a technique is actually adapting to the changes in the data?

- Caveats:
  - We should **not use** the same measure that is used in learning.
    - E.g., if we are using a $k$-means based algorithm we should not evaluate using MSSQE (Mean Sum of Squared Errors)
  - We should be able to observe the evolution of both
    - the data and
    - the evaluation measure.
  - i.e., not only 1 final measure of evaluation.
Element of the Solution

Main idea:

- Define testable predictions from the model.
- Evaluate them using the data.
- Create distinct evolving scenarios.
- Formal framework in (Rojas & Nasraoui, 2009)
  - Extends on an amalgam of several evaluation experiments on clustering data streams (including Web Clickstreams) in (Nasraoui et al., 2003) (Nasraoui & Rojas, 2006) (Nasraoui et al., 2007) (Nasraoui et al., 2008)
Example 1: Clustering a numerical data stream

The technique:

- A robust density-based clustering technique:
  - Maintains a set of cluster centers with a measure of scale.
  - Includes decaying temporal weights to ensure the old data is forgotten.
Example 1: Clustering numerical data streams

- Validation Strategy 1:
  - Since it is a *density* based technique, the clusters should be where the most recent points are.
  - We can visualize it in 2D.
  - *Visual* validation in space.
Example 1: Clustering numerical data streams

- Using synthetic data:
  - Present one cluster at a time.
Example 1: Clustering numerical data streams

- (Nasraoui & Rojas, 2006)
- Using synthetic data
  (BIRCH dataset):
    - Present a more ‘natural’ order.
      - Older data appears in light blue.
      - Newer data appears in black.
      - Cluster centers appear as ‘*’.
Example 1: Clustering numerical data streams

- Using real data (1999 KDD Cup data):
  - 32 numerical attributes.
  - Use *multidimensional scaling* with the most recent data + the representatives, pick the 2 main dimensions, and plot them.

  - Older data appears as little points.
  - Newer data appears as little circles.
  - Cluster centers appear as ‘?’.
Example 1: Clustering numerical data streams

- Validation Strategy 2:
  - We know which classes/clusters the data belongs to.
  - Their ordering in time determines a specific class/cluster profile, the ground truth.
  - During learning, there is a class/cluster hit if:
    - The point is within tight Chebyshev bounds of at least one clustering representative.
  - Visual validation in time:
    - The learned class/cluster profile should be similar to the ground truth.
    - Specify milestones:
      - Natural or artificial points of change in class/clusters.
Example 1: Clustering numerical data streams

- Using synthetic data:
  - Left: ground truth
  - Right: cluster to Class Hits with time/sequence of data arrival
Example 1: Clustering numerical data streams

- Using real data (1999 KDD Cup data):
  - 23 attacks (classes)
  - Left: ground truth
  - Right: cluster to Class Hits with time/sequence of data
Example 2: Web Clickstreams

- (Nasraoui et al., 2003) (Nasraoui et al., 2006)
- The technique:
  - TECNO-Streams (Nasraoui et al., 2003): A bio-inspired algorithm to discover user profiles from clickstreams.
  - User profiles are represented as lists of URLs.
Example 2: Web Clickstreams

- Validation Strategy:
  - Use *precision* and *recall* between an incoming session and a profile as follows:
    - Precision is the proportion of URLs from the *session* in the *profile*.
    - Recall is the proportion of URLs from the *profile* in the *session*.
  - We have a *precision* [*recall*] hit if at least one profile surpasses a minimum threshold.
  - The ground truth is constructed by computing the same measures and taking the recent data instead of the synopsis.
Example 2: Web Clickstreams

- Using a curated 20 Profiles web clickstream data, one profile at a time:
  - Left, ground truth (Input Data w.r.t. Gnd Truth Profile).
  - Right, Precision (Cluster w.r.t. Gnd Truth Profile).
Example 3: Recommendations in Dynamic Streaming Environments

- The technique (Nasraoui et al., 2007)
  - Any stream recommender technique.

- Validation strategy:
  - Given new input session, create a pseudo-session:
    - Select a random subset URLs from a completely new (ground-truth) real user session.
  - Obtain recommendations using the pseudo-session as input.
  - Compute precision, recall, and F1 between the recommendations and the remainder of the real user session.
Example 3: Recommendations in Dynamic Streaming Environments

- Comparison of F1 measure for two techniques using a curated 20 Profiles Web clickstream data:
  - Left, one profile at a time.
  - Center, 5 selected profiles, repeated in order (2 repetitions cycles).
  - Right, natural order of sessions (chronological).
Example 4: Generic Evaluation Framework for Evolving Transactional Data Streams

- (Rojas & Nasraoui, 2009)
- Online *evaluation*, not *validation*
  - Does not require ground truth.
- Transactional data:
  - Lists of properties:
    - User sessions.
    - Binary bags-of-words.
    - Sales transactions.
Example 4: Generic Evaluation Framework for Evolving Transactional Data Streams

- **Goal:**
  - To maintain a *current* estimate of model quality.
    - As a training/testing approach does for static data.

- **Assumptions**
  - There is a current model of the stream, representing the evolving patterns (e.g., frequent itemsets, cluster centroids, etc).

- **Questions to answer:**
  - Are the learned patterns a good summary of the evolving data stream?
  - Are they better than a trivial model?
    - I.e., Are the patterns real or just an artifact of the technique?
Example 4: Generic Evaluation Framework for Evolving Transactional Data Streams

- **Approach:**
  - Break the record at time T in pieces.
  - For every piece, attempt to reconstruct the full record.
    - Get a *prediction*.
    - How the prediction is generated is model-dependent.
  - Compare the predictions with the full record.
    - Using precision, recall, F1, in micro or macro versions.
      - If the size of the prediction is the same as the size of the record, all measures are the same.
Example 4: Generic Evaluation Framework for Evolving Transactional Data Streams

1: Get $\mathcal{R}_T$
2: while $\mathcal{R}_T \neq \emptyset$ {Main loop} do
3: $\{R_i\} \leftarrow \text{PARTITIONRECORD}(\mathcal{R}_T, \mathcal{M}_{T-1})$
4: $\{\mathcal{P}_i\} \leftarrow \text{OBTAINPREDICTIONS}(\{R_i\}, \mathcal{M}_{T-1})$ {Obtain predictions by partition}
5: $\{\mathcal{P}_T, \mathcal{R}_T\} \leftarrow \text{GETUNIFIEDEVALUATION}(\{\mathcal{P}_i\}, \mathcal{R}_T)$ {Obtain unified precision and recall}
6: $\mathcal{M}_T \leftarrow \text{UPDATEMODEL}(\mathcal{M}_{T-1})$ {Update current model}
7: $T \leftarrow T + 1$ {Prepare for next record}
8: end while
Rationale for the approach:

- If every attribute is treated as a class,
- this can be seen as the problem of finding the most likely classes for every piece of the record.
Example 4: Generic Evaluation Framework for Evolving Transactional Data Streams

- Instance 1 – Popular:
  - The model is every *attribute* with its frequency.
  - To obtain a prediction:
    - Pick the $k$ most frequent attributes.
  - Lower bound for comparison.
  - Can be computed analytically for some cases.
    - E.g., power law distribution.
Example 4: Generic Evaluation Framework for Evolving Transactional Data Streams

- Instance 2 – Naïve Bayes:
  - The model is every pair of attributes, with its frequency
  - To obtain a prediction:
    - Compute the score for attribute $c_i$:
      \[
      s_i = p_i \prod_{j \in R} p_{j|i}
      \]
      \[
      \propto P(c_i) P(R|c_i)
      \]
    - Where $p_{j|i}$ is the conditional probability of having $c_j$ given $c_i$.
    - The prediction is the attributes with the top $k$ scores.
Example 4: Generic Evaluation Framework for Evolving Transactional Data Streams

- Instance 3 – Greedy Naïve Bayes:
  
  The model is a *subset of the pairs of attributes*, and their frequencies, as follows:
  
  - The pair \((c_i, c_j)\) is in the model *if* the frequency is among the top \(k\) with respect to \(c_i\) or \(c_j\).

- To obtain a prediction:
  
  - Follow the same protocol as before.
Example 4: Generic Evaluation Framework for Evolving Transactional Data Streams

- Instance 4 – Correlated Naïve Bayes:
  - The model is a *subset of the pairs of attributes*, and their frequencies, as follows:
    - The pair \((c_i, c_j)\) is in the model if:
      - They pass a \(\chi^2\) test of independence (95% of confidence).
      - They have a significant correlation \((\rho > 0.2)\).

- To obtain a prediction:
  - Follow the same protocol as before.
Example 4: Generic Evaluation Framework for Evolving Transactional Data Streams

- Scenario 1:
  - Data:
    - Synthetic data with some attributes more frequent than others:
      - No higher level patterns occur (combinations of two or more attributes)
      - The model *Popular* should capture this simple pattern better.
    - 20,000 transactions, with an average length of 20 attributes, from 1,000 possible attributes.
  - No evolution:
    - Sliding window of size 500.
Example 4: Generic Evaluation Framework for Evolving Transactional Data Streams
Example 4: Generic Evaluation Framework for Evolving Transactional Data Streams

- **Scenario 2:**
  - **Data:**
    - 20 Newsgroups: ~20,000 records, ~1,000 per group.
  - **Strong evolution:**
    - Periodic repetition of one group:
      - REC. MOTORCYCLES
        - High intra-similarity.
      - COMP.OS.MS-WINDOWS.MISC
        - Low intra-similarity.
    - Sliding window of size 500.
Example 4: Generic Evaluation Framework for Evolving Transactional Data Streams
Example 4: Generic Evaluation Framework for Evolving Transactional Data Streams
Example 4: Generic Evaluation Framework for Evolving Transactional Data Streams

Scenario 3:

Data:
- 20 Newsgroups.

Strong evolution:
- Every group, presented one at a time.
- Sliding window of size 500.
Example 4: Generic Evaluation Framework for Evolving Transactional Data Streams
Example 4: Generic Evaluation Framework for Evolving Transactional Data Streams

- **Scenario 4:**
  - **Data:**
    - 20 Newsgroups.
  - **Natural evolution:**
    - Data presented in its natural order.
    - Sliding window of size 500.
Example 4: Generic Evaluation Framework for Evolving Transactional Data Streams
References – Concept Drift (1/2)


References – Concept Drift (2/2)


References – Stream Mining


References – Frequent Itemsets (1/5)


References – Frequent Itemsets (2/5)


References – Frequent Itemsets (3/5)


- C. Giannella, J. Han, J. Pei, X. Yan, and P. Yu (2003). Mining Frequent Patterns in Data Streams at Multiple Time Granularities. In H. Kargupta, A. Joshi, K. Sivakumar, and Y. Yesha (eds.), Next Generation Data Mining. AAAI/MIT.

References – Frequent Itemsets (4/5)


References – Frequent Itemsets (5/5)


References - Clustering


References - Personalization

References – Evolution Detection


References – Evaluation in a Stream Mining Framework


Thank you!

Questions?
Presentation Outline

- Block 1: Introduction
- Block 2: Mining the Social Web
- Block 3: Recommendation and Personalization
- Block 4: Evolution in the Web
- Block 5: Mining Web Data Streams
- Block 6: Conclusions and Outlook
Outlook

- Mining the Social Web
  - Social Network Analysis is a promising research area in data mining with many interesting emerging applications.
  - Most community mining approaches focus on static, global and homogenous networks. Research for local, overlap, dynamic, multi-relational and heterogeneous communities is in an early stage.
Outlook

- Recommendation and Personalization
  - Web Personalization has been one of the success stories of Web mining over the past decade
    - now even more important with the proliferation of more complex forms of social Web interaction.
  - But, significant challenges remain in ensuring the security and robustness of collaborative personalization systems such as ecommerce recommenders and social Web applications where the learned models and system behavior depend on user input.
    - These challenges include: the characterization of different types of attacks, development of robust algorithms while maintaining accuracy, and the development of effective methods for the detection of and response to attacks.
Outlook

- Evolution in Web applications
  - How to incorporate drifts and shifts in user interests when we build a recommendation engine?
  - How to present changes in an online community in a way comprehensible to community members (while preserving privacy of the members)?
  - How to explain a dynamic latent model to an application owner? To an application user?
Outlook

Mining Web Data Streams

Challenges are still open at all stages of KDD-Streams

- **Streamlining** data collection (deal w/ dynamic websites, newer Web technology, etc)
- **Integration** with other channel streams, e.g. content, structure, social, semantics (*all within streaming framework*)
- Pre-processing in a streaming framework: avoiding bottlenecks, handling change (e.g. dynamic page mapping)
- Developing more/better 1-pass algorithms for Web ClickStream Mining that are friendly to deploy in real life
- **Compact models & data structures** that take little space
- *while* accelerating retrieval, processing, maintenance
- Evaluation in a **dynamic noisy** environment w/ **massive** data
Thank you!

Questions?