707.000
Web Science and Web Technology
„Web Technologies I“

Markus Strohmaier

Univ. Ass. / Assistant Professor
Knowledge Management Institute
Graz University of Technology, Austria

e-mail: markus.strohmaier@tugraz.at
web: http://www.kmi.tugraz.at/staff/markus
Overview

Agenda

Technical preliminaries for your first course work:

• Network Preliminaries
  – One Mode and Two Mode Networks
  – Network Representation
  – Network Metrics

• Software Architecture Preliminaries
  – REST
  – JSON

• Release of Home Assignment 1.1
Network

- A collection of individual or atomic entities
- Referred to as nodes or vertices (the “dots” or “points”)
- Collection of links or edges between vertices (the “lines”)
- Links can represent any pairwise relationship
- Links can be directed or undirected
- Network: entire collection of nodes and links
- For us, a network is an abstract object (list of pairs) and is separate from its visual layout
- that is, we will be interested in properties that are invariant
  - structural properties
  - statistical properties of families of networks
Social Networks Examples

Why and How to Flash Your BIOS
this url has been saved by 106 people.
save this to your bookmarks »

user notes

Why and How to Flash Your BIOS
rslw77
This article is going to focus on the basics and explain ways to flash the BIOS, precautions and how to recover in case of a bad flash.
edwinnxk

Why and How to Flash Your BIOS (Page 1 of 4 ) Flashing the BIOS is one of the most feared topics related to computers. Yes, people should be very cautious because it can be dangerous. This article is going to focus on the basics and explain ways to flash
oblonski
One mode / two mode networks
(uni/bipartite graphs)

One mode network:
• A single type of nodes

Two mode network:
• Two types of nodes
• Edges are only possible between different types of nodes
How can we represent (social) networks?

We will discuss three basic forms:

- Adjacency lists
- Adjacency matrices
- Incident matrices
Adjacency Matrix for one mode networks

- Complete description of a graph
- The matrix is symmetric for nondirectional graphs
- A row and a column for each node
- Of size $g \times g$ ($g$ rows and $g$ columns)
Adjacency matrices for One-Mode Networks

taken from http://courseweb.sp.cs.cmu.edu/~cs111/applications/ln/lecture18.html

Adjacency matrix or sociomatrix
Adjacency lists for One-Mode Networks

taken from http://courseweb.sp.cs.cmu.edu/~cs111/applications/ln/lecture18.html
Incidence Matrix for One-Mode Networks

- (Another) complete description of a graph
- Nodes indexing the rows, lines indexing the columns
- g nodes and L lines, the matrix I is of size g x L
- A "1" indicates that a node $n_i$ is incident with line $l_j$
- Each column has exactly two 1's in it

Table 4.3. Example of an incidence matrix: “lives near” relation for six children

<table>
<thead>
<tr>
<th></th>
<th>$l_1$</th>
<th>$l_2$</th>
<th>$l_3$</th>
<th>$l_4$</th>
<th>$l_5$</th>
<th>$l_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n_1$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$n_2$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$n_3$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$n_4$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$n_5$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$n_6$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

[Wasserman Faust 1994]

Fig. 3.2. The six actors and the three sets of directed lines — a multivariate directed graph
Adjacency lists vs. matrices

taken from http://courseweb.sp.cs.cmu.edu/~cs111/applications/ln/lecture18.html

Lists Vs. Matrices (I)

If the graph is sparse (there aren't many edges), then the matrix will take up a lot of space indicating all of the pairs of vertices which don't have an edge between them, but the adjacency list does not have that problem, because it only keeps track of what edges are actually in the graph.

On the other hand, if there are a lot of edges in the graph, or if it is fully connected, then the list has a lot of overhead because of all of the references.
Lists Vs. Matrices (II)

If we need to look specifically at a given edge, we can go right to that spot in the matrix, but in the list we might have to traverse a long linked list before we hit the end and find out that it is not in the graph.

If we need to look at all of a vertex's neighbors, if you use a matrix you will have to scan through all of the vertices which aren't neighbors as well, whereas in the list you can just scan the linked-list of neighbors.
Adjacency lists vs. matrices

taken from http://courseweb.sp.cs.cmu.edu/~cs111/applications/ln/lecture18.html

Lists Vs. Matrices (III)

If, in a directed graph, we ask the question, "Which vertices have edges leading to vertex X?", the answer is **straight-forward to find in an adjacency matrix** - we just walk down column X and report all of the edges that are present. But, life isn't so easy with the adjacency list - we actually have to perform a brute-force search.

⇒ So which representation you use depends on what you are trying to represent and what you plan on doing with the graph

Illustration!
Adjacency matrices for Two-Mode Networks

- Complete description of a graph
- A row and a column for each node
- Of size $m \times n$ (m rows and n columns)

```
<table>
<thead>
<tr>
<th></th>
<th>Allison</th>
<th>Drew</th>
<th>Eliot</th>
<th>Keith</th>
<th>Ross</th>
<th>Sarah</th>
</tr>
</thead>
<tbody>
<tr>
<td>Party 1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Party 2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Party 3</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
```
Network Metrics for One-Mode Networks

- If the distance between all pairs is finite, we say the network is connected (a single component); else it has multiple components
- **Degree of vertex** $v$: number of edges connected to $v$
- **Average degree of vertex** $v$: avg. number of edges connected to a vertex
Two Mode Networks - Rates of Participation
[Wasserman Faust 1994]

• The number of events with which each actor is affiliated.
• These quantities are either given by
  – the row totals of affiliation matrix A or
  – the entries on the main diagonal of the one-mode socio-matrice $X^N$
• Thus, the number of events with which actor $i$ is affiliated is equal to the degree of the node representing the actor in the bipartite graph.
• Also interesting: **Average rate of participation**

Example:

<table>
<thead>
<tr>
<th></th>
<th>Party 1</th>
<th>Party 2</th>
<th>Party 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allison</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Drew</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Eliot</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Keith</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Ross</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Sarah</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Examples: What does the rate of participation relate to in the Netflix / Amazon bipartite graph of customer/movies or customer/products?
Two Mode Networks - Size of Events
[Wasserman Faust 1994]

• The number of actors participating in each event.
• The size of each event is given by either
  – the column totals of the affiliation matrix $A$ or
  – the entries on the main diagonal of the one-mode
    sociomatrix $X^M$.
• Thus, the size of each event is equal to the degree of the node representing the event in the bipartite graph.
• Also interesting: **Average size of events**
  – Sometimes useful to study average size of clubs or organizations
• Size of events might be constrained:
  – E.g. board of company directors are made up of a fixed number of people

Example:

<table>
<thead>
<tr>
<th></th>
<th>Party 1</th>
<th>Party 2</th>
<th>Party 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allison</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Drew</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Eliot</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Keith</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Ross</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Sarah</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Examples: What does the rate of participation relate to in the Netflix / Amazon bipartite graph of customer/movies or customer/products?
Network Acquisition on the Web

• How can we acquire network information from the web?

Example:
• RESTful Interfaces
REST
Roy Fielding, Dissertation 2000

• Roy Fielding
  - Chief Scientist, Day Software
  - Co-founder and member, The Apache Software Foundation
  - Dissertation on Architectural Styles and the Design of Network-based Software Architectures at the Information and Computer Science, UC Irvine

In his dissertation, he "introduce[s] the Representational State Transfer (REST) architectural style and describe[s] how REST has been used to guide the design and development of the architecture for the modern Web"

Ressources:
• http://roy.gbiv.com/talks/200709_fielding_rest.pdf

Has played a role in authoring the Internet standards for the Hypertext Transfer Protocol (HTTP) and Uniform Resource Identifiers (URI)
The early web

Figure 5-5. Early WWW Architecture Diagram
The Problem (circa 1994)

Early architecture was based on solid principles

- URLs, separation of concerns, simplicity
- lacked architectural description and rationale

Protocols assumed a direct server connection

- no awareness of caching, proxies, or spiders
- many independent extensions

Public awareness of the Web was just beginning

- exponential growth threatened the Internet
- commercialization meant new requirements and new stakeholders

A modern Web architecture was clearly needed

- but how do we avoid breaking the Web in the process?
Software Architectures

A software architecture is an abstraction of the runtime elements of a software system during some phase of its operation. A system may be composed of many levels of abstraction and many phases of operation, each with its own software architecture.

- A software architecture is defined by a configuration of architectural elements—components, connectors, and data—constrained in their relationships in order to achieve a desired set of architectural properties.

- A configuration is the structure of architectural relationships among components, connectors, and data during a period of system run-time.
Web Architecture

One abstraction level above the implementation

Components

- User agents, Intermediaries, Servers
- Browsers, Spiders, Proxies, Gateways, Origin Servers

Connectors

- HTTP: a standard transfer protocol to prefer over many

Data

- URI: one identifier standard for all resources
- HTML, XML, RDF, ...: common representation formats to describe and bind resources
REST
Roy Fielding, RailsConf Europe, September 2007

How beautiful it is to do nothing, and then REST afterward. [Spanish Proverb]

Style = nil

Starting from a condition of no constraints...

No architectural constraints on the roles and features of components, connectors and data, and the allowed relationships among them.
REST

Roy Fielding, RailsConf Europe, September 2007

Style += Client/Server

Apply separation of concerns: Client-Server

Representation

Data

improves UI portability

simplifies server

enables multiple organizational domains

Separation of concerns, scalability
REST client-stateless-server (CSS) style

Roy Fielding, RailsConf Europe, September 2007

... and to lie sometimes on the grass ...

Style += Stateless

Constrain interaction to be stateless...

degrades efficiency
Repetitive data

simplifies server

improves scalability

improves reliability

Visibility –
Single request contains all information to understand the full nature of the request

Easily free resources
Easy recovery

each request from client to server must contain all of the information necessary to understand the request
REST client-cache-stateless-server style

Roy Fielding, RailsConf Europe, September 2007

Style += Caching

Add optional non-shared caching

- the potential to partially or completely eliminate some interactions
- reduces average latency
- improves efficiency
- improves scalability

Cachable vs. non-cachable content
degraded reliability
REST
Roy Fielding, RailsConf Europe, September 2007

Style += Uniform Interface

Apply generality: uniform interface constraint

- improves visibility
- independent evolvability
- decouples implementation
- degrades efficiency

information is transferred in a standardized form
REST
Roy Fielding, RailsConf Europe, September 2007

Style += Layered System

Apply info hiding: layered system constraints

- Simplifies clients
- Improves scalability
- Load balancing
- Adds latency
- Shared caching
- Legacy encapsulation

Complexity reduction: each component cannot "see" beyond the immediate layer

... or watching the clouds float across the sky, ...
REST Style

by downloading and executing code in the form of applets or scripts

simplifies clients  improves extensibility  reduces visibility

Finally, allow code-on-demand (applets/js)

... is by no means a waste of time. [Sir John Lubbock]

Roy Fielding, RailsConf Europe, September 2007
REST
Roy Fielding, RailsConf Europe, September 2007

Sometimes the most urgent and vital thing you can possibly do is take a complete REST. [Ashleigh Brilliant]

REST on a slide

- RR
  - replicated
  - Cache on-demand

- CS
  - Client server
  - stateless
  - intermediate processing
  - reliable
  - cacheable
  - scalable

- LS
  - Layered System
  - mobile
  - shared
  - multi-org.

- VM
  - Virtual Mach.
  - extensible
  - code on demand
  - reusable

- U
  - simple visible

- COD
  - Code on demand

- LC$$SS
  - Layered client server
  - extensible
  - code on demand
  - reusable

- LCODC$$SS
  - Layered client server
  - extensible
  - code on demand
  - reusable

- REST
  - layered
  - programmable
  - uniform interface

Client serverCache
Why bother?
Roy Fielding, Dissertation 2000

- “creating an **architectural model** for how the Web should work”

- “Using the new architectural style as a guide, we can **compare proposed extensions and modifications** to the Web architecture against the constraints within the style. **Conflicts** indicate that the proposal would violate one or more of the design principles behind the Web.

- For **severe conflicts**, such as a change in the interaction style, the same functionality would **either be replaced** with a design more conducive to the Web's style, or the proposer would be told to implement the functionality as a **separate architecture** running in parallel to the Web.”
REMINDER: Web Architecture
Roy Fielding, RailsConf Europe, September 2007

Web Architecture
One abstraction level above the implementation

Components
- User agents, Intermediaries, Servers
- Browsers, Spiders, Proxies, Gateways, Origin Servers

Connectors
- HTTP: a standard transfer protocol to prefer over many

Data
- URI: one identifier standard for all resources
- HTML, XML, RDF, ...: common representation formats to describe and bind resources

Representations
Representations
Roy Fielding, Dissertation 2000

• “REST components perform actions on a resource by using a representation to capture the current or intended state of that resource and transferring that representation between components.”

• “less precise names for a representation include: document, file, and HTTP message entity, instance, or variant. “

• Depending on the message control data, a given representation may indicate the current state of the requested resource, the desired state for the requested resource, or the value of some other resource […].
# REST


## Table 17-1. HTTP Methods for REST

<table>
<thead>
<tr>
<th>Method</th>
<th>CRUD Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>Retrieve</td>
<td>Retrieves the representation of a resource.</td>
</tr>
<tr>
<td>HEAD</td>
<td></td>
<td>Retrieves metadata for the representation and resource.</td>
</tr>
<tr>
<td>POST</td>
<td>Create</td>
<td>In the strict sense, POST creates a resource. In the real world, however, POST is typically used to create, update, and even delete a resource. It is normal to use REST services that support only GET and POST.</td>
</tr>
<tr>
<td>PUT</td>
<td>Update</td>
<td>Updates a resource. More often than not, you will not see this method used in the real world but instead will see POST used to perform the actions.</td>
</tr>
<tr>
<td>DELETE</td>
<td>Delete</td>
<td>Deletes a resource. Just like PUT, in the real world this is rarely used, and instead POST is used in its place.</td>
</tr>
</tbody>
</table>
Theory vs. Practice

Chapter "Representational State Transfer (REST)" in "Pro PHP XML and Web Services", R. Richards 633–672 (2006) and

- How has this theory influenced current practice?

REST applied to HTTP

- The REST service is expressed as a URL and is accessed with basic HTTP requests.

- The HTTP verb is important: a GET is a read operation, POST is a creation, and PUT make updates to the service.

- The return payload is usually XML or JSON.

http://dev2dev.bea.com/pub/a/2007/05/google-mashups.html
JSON (JavaScript Object Notation) is a programming language model data interchange format. It is minimal, textual, and a subset of JavaScript.

What is JSON?

- Lightweight data-interchange format
  - Compared to XML
- Simple format
  - Easy for humans to read and write
  - Easy for machines to parse and generate
- JSON is a text format
  - Programming language independent
  - Uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python
JSON Structures

- A collection of name/value pairs
  - In various languages, this is realized as an object, record, struct, dictionary, hash table, keyed list, or associative array
- An ordered list of values
  - In most languages, this is realized as an array, vector, list, or sequence
- These are universal data structures supported by most modern programming languages
JSON Object Notation

- A JSON object is an unordered set of name/value pairs
- A JSON object begins with { (left brace) and ends with } (right brace)
- Each name is followed by : (colon) and the name/value pairs are separated by , (comma)
The types represented in JSON are strings, numbers, booleans, object, arrays, and null.
The types represented in JSON are strings, numbers, booleans, object, arrays, and null.
• JSON vs. XML

```json
{"menu": {
    "id": "file",
    "value": "File",
    "popup": {
        "id": "file",
        "value": "Save",
        "onclick": "CreateNewDoc()"
    }
}
```

Start of JSON object

Start of JSON array

Name

Value

Name/Value pairs separated by comma
JSON

http://www.json.org/fatfree.html

- JSON has **no version number**. No revisions to the JSON grammar are anticipated. “If something has a 1.0, it will inevitably get a 1.1 and a 2.0, and everything is crap until it is 3.0. JSON is very stable.”

- JSON **doesn't have namespaces**. Every object is a namespace: its set of keys is independent of all other objects, even exclusive of nesting. JSON uses context to avoid ambiguity, just as programming languages do.

- JSON **is not extensible**. “It does not need to be. It can represent any non-recurrent data structure as is. JSON is flexible. New fields can be added to existing structures without obsoleting existing programs.”
Text to Object Conversion in JavaScript code

```javascript
var myObject = eval('(' + myJSONtext + ')');
```

- To convert a JSON text into an JSON object, use the `eval()` function
  - `eval()` invokes the JavaScript compiler
  - Since JSON is a proper subset of JavaScript, the compiler will correctly parse the text and produce an object structure
JSON

Slides taken from Sang Shin, Java Technology Architect, Sun Microsystems, Inc

Security & JSON Parser

// Include http://www.json.org/json.js
var myObject = myJSONtext.parseJSON();

- eval() can compile and execute any JavaScript program, so there can be security issues (cross-site scripting)
  - Use eval() when the source can be trusted
- When security is a concern - the source cannot be trusted - it is better to use a JSON parser
  - A JSON parser will only recognize JSON text and so is much safer
How to Receive JSON Data at the Client Side

- JSON data is received as a string
- Calling `eval()` will generate JSON object in JavaScript code
  ```javascript
  var JSONdata = eval(req.responseText);
  ```
- Once you have JSON object, you can use . notation to access its properties
  ```javascript
  var name = JSONdata.name;
  var address = JSONdata.addresses[3];
  var streetname = JSONdata.addresses[3].street;
  ```
# Approximate Course Schedule

<table>
<thead>
<tr>
<th>Month</th>
<th>MatLab Exercises</th>
<th>Yahoo Boss Search Challenge</th>
<th>Optional! Google Online Marketing Challenge</th>
</tr>
</thead>
<tbody>
<tr>
<td>March</td>
<td>Today: HA1.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Next week: HA1.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>April</td>
<td></td>
<td>Easter holidays</td>
<td><strong>Overlap!</strong></td>
</tr>
<tr>
<td>May</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>June</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Overlap!**
Home Assignment 1.1

Network Examples

[Images of network diagrams]
Home Assignment 1.1

Get familiar with REST APIs
- http://www.programmableweb.com/

with Pajek and the Pajek .net Format
- http://vlado.fmf.uni-lj.si/pub/networks/pajek/
- http://iv.slis.indiana.edu/Im/Im-pajek.html

Useful Links
JSON Reader Tool
- http://jsontools.berlios.de/

Eclipse Plugin ANTLR (parser generator)
- http://antlr4eclipse.sourceforge.net/

Other JSON Tools
- http://www.json.org/java/

For questions please post to the newsgroup:
Your tutors Gabi and Ingo will try to help with your questions
Any questions?

See you next week!