CS276B

Web Search and Mining
Winter 2005

Lecture 5

(includes slides borrowed from Jon Herlocker)

Recap: Project and Practicum

- We hope you've been thinking about projects!
- Revised concrete project plan due today
- Initial project presentation: Thursday and Tuesday
  - About 10 minutes per group
    - About 5 minutes presentations and a few minutes discussion
  - A chance to explain and focus what you are doing and why it's interesting

Plan for Today

- Recommendation Systems (RS)
  - The most prominent type of which goes under the name *Collaborative Filtering* (CF)
- What are they are and what do they do?
- A couple of algorithms
- Going beyond simple behavior: context
- How do you measure them?

Recommendation Systems

- Given a set of *users* and *items*
  - Items could be documents, products, other users …
- Recommend items to a user based on
  - Past behavior of this and other users
    - Who has viewed/bought/liked what?
  - Additional information on users and items
    - Both users and items can have known *attributes* [age, genre, price, …]

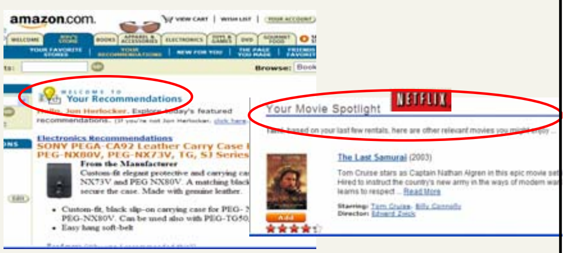
What do RSs achieve?

- Help people make decisions
  - Examples:
    - Where to spend attention
    - Where to spend money
- Help maintain awareness
  - Examples:
    - New products
    - New information

Sample Applications

- Ecommerce
  - Product recommendations - amazon
- Corporate Intranets
  - Recommendation, finding domain experts, …
- Digital Libraries
  - Finding pages/books people will like
- Medical Applications
  - Matching patients to doctors, clinical trials, …
- Customer Relationship Management
  - Matching customer problems to internal experts

Well-known recommender systems: Amazon and Netflix



Corporate intranets - document recommendation

Corporate intranets - “expert” finding

Inputs to intranet system

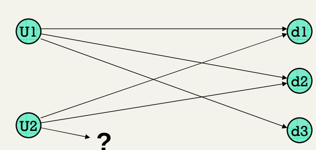
- Behavior
  - users’ historical “transactions”
- Context
  - what the user appears to be doing now
- User/domain attributes
  - additional info about users, documents …

Inputs - more detail

- Past transactions from users:
  - which docs viewed
  - content/attributes of documents
  - which products purchased
  - pages bookmarked
  - explicit ratings (movies, books …)
- Current context:
  - browsing history
  - search(es) issued
- Explicit role/domain info:
  - Role in an enterprise
  - Document taxonomies
  - Interest profiles

Example - behavior only

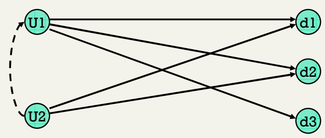
Users Docs viewed



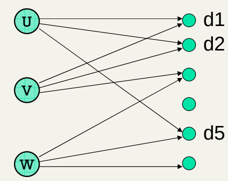
U1 viewed d1, d2, d3.
U2 views d1, d2.
Recommend d3 to U2.

Expert finding - simple example

Recommend U1 to U2 as someone to talk to?



Simplest Algorithm: Naïve *k* Nearest Neighbors



U viewed d1, d2, d5.

Look at who else viewed d1, d2 or d5.

Recommend to U the doc(s) most "popular" among these users.

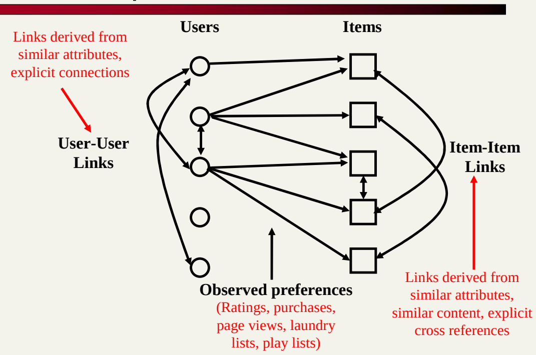
Simple algorithm - shortcoming

- Treats all other users as equally important
- Ignores the fact that some users behaved more like me in the past

Typical RS issues

- Large item space
  - Usually with item attributes
- Large user base
  - Usually with user attributes (age, gender, city, …)
- Some evidence of customer preferences
  - Explicit ratings (powerful, but harder to elicit)
  - Observations of user activity (purchases, page views, emails, what was printed, …)
  - Typically *extremely* sparse, even when user has an opinion

The RS Space



Definitions

- A *recommendation system* is any system which provides a recommendation/prediction/opinion to a user on items
  - Rule-based systems use manual rules to do this
- An item similarity/clustering system uses item links to recommend items like ones you like
- A classic *collaborative filtering system* uses the links between users and items as the basis of recommendations
- Commonly one has *hybrid systems* which use all three kinds of links in the previous picture

Link types

- User attributes-based Recommendation
  - Male, 18-35: Recommend *The Matrix*
- Content Similarity
  - You liked *The Matrix:* recommend *The Matrix Reloaded*
- Collaborative Filtering
  - People with interests like yours also liked *Kill Bill*

Rule-based recommendations

- In practice – rule-based systems are common in commerce engines
  - Merchandizing interfaces allow product managers to promote items
  - Criteria include inventory, margins, etc.
- Must reconcile these with algorithmic recommendations

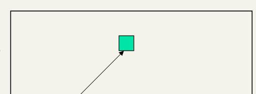
Measuring collaborative filtering

- How good are the predictions?
- How much of previous opinion do we need?
- Computation.
- How do we motivate people to offer their opinions?

Matrix view

Docs

A = Users



A_{ij} = 1 if user i viewed doc j,

= 0 otherwise.

AA^t : Entries give # of docs commonly viewed by pairs of users.

Voting Algorithm

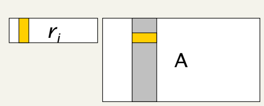
- Row i of AA^t : Vector whose j^{th} entry is the # of docs viewed by both i and j.
- Call this row r_i, e.g., (0, 7, 1, 13, 0, 2, ….)

What's on the diagonal of AA^t?

Voting Algorithm

- Then $r_i \circ A$ is a vector whose k^{th} entry gives a weighted vote count to doc k
  - emphasizes users who have high weights in r_i .
- Recommend doc(s) with highest vote counts.

How does this differ from the simple algorithm?



Voting Algorithm - implementation issues

- Wouldn't implement using matrix operations
  - use weight-propagation on compressed adjacency lists
- Need to log and maintain "user views doc" relationship.
  - typically, log into database
  - update vote-propagating structures periodically.
- For efficiency, discard all but the heaviest weights in each r_i
  - only in fast structures, not in back-end database.

Write pseudo code

Forward pointer

- There are connections between CF and web link analysis:
  - The voting algorithm may be viewed as *one* iteration of the Hubs/Authorities algorithm

Different setting/algorithm

- Each user i rates some docs (products, …)
  - say a real-valued *rating* v_{ik} for doc k
  - in practice, one of several ratings on a form
- Thus we have a ratings vector v_i for each user
  - (with lots of zeros)
- Compute a *correlation coefficient* between every pair of users i,j
  - dot product of their ratings vectors
  - (symmetric, scalar) measure of how much user pair i,j agrees: w_{ij}

Predict user *i*'s utility for doc *k*

- Sum (over users j such that v_{jk} is non-zero)

  $w_{ij}\, v_{jk}$
- Output this as the predicted utility for user i on doc k.

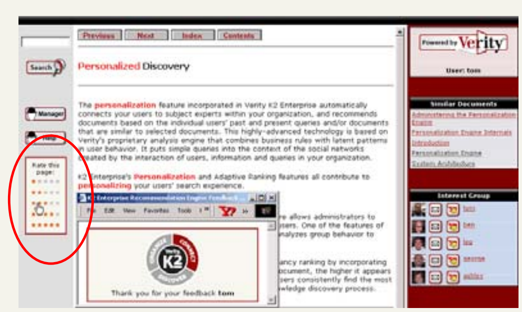
So how does this differ from the voting algorithm?

It really doesn't …

Same algorithm, different scenario

- <u>Implicit</u> (user views doc) vs. <u>Explicit</u> (user assigns rating to doc)
- Boolean vs. real-valued utility
  - In practice, must convert user ratings on a form (say on a scale of 1-5) to real-valued utilities
  - Can be fairly complicated mapping
    - Likeminds function (Greening white paper)
  - Requires understanding user's interpretation of form

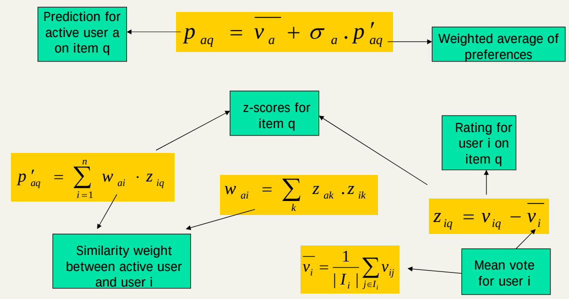
Rating interface



Early systems

- GroupLens (U of Minn) (Resnick/Iacovou/Bergstrom/Riedl)
  - netPerceptions company
  - Based on nearest neighbor recommendation model
- Tapestry (Goldberg/Nichols/Oki/Terry)
- Ringo (MIT Media Lab) (Shardanand/Maes)
- Experiment with variants of these algorithms

GroupLens Collaborative Filtering Scheme



netPerceptions: example of effectiveness (Konstan/Resnick)

- GUS Call Center: a UK multi-catalog company
  - Consumers call in purchases
  - Operators trained to try to “cross-sell”
- Company implemented RS personalization
- Experiment:
  - one group of agents with old method
  - one group of agents with RS personalization
- Results

	Trad. cross-sell	netPerceptions
Avg Cross-Sell Value	$19.50	60% higher
Cross Sell Success Rate	9.8%	50% higher

- http://www.chi-sa.org.za/seminar%5Csandton.pdf

RS Inputs - revisited

Past transactions from users:
- which docs viewed
- content/attributes of documents
- which products purchased
- pages bookmarked
- explicit ratings (movies, books …)

Current context:
- browsing history
- search(es) issued

Explicit profile info:
- Role in an enterprise
- Document taxonomies
- Interest profiles

The next level - modeling context

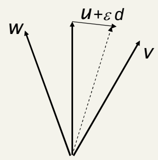
- Suppose we could view users and docs in a common vector space of terms
  - docs already live in term space
- How do we cast users into this space?
  - Combination of docs they liked/viewed
  - Terms they used in their writings
  - Terms from their home pages, resumes …

Context modification

- Then “user *u* viewing document *d*” can be modeled as a vector in this space: $u+\varepsilon d$
- User *u* issuing search terms *s* can be similarly modeled:
  - add search term vector to the user vector
- More generally, any term vector (say recent search/browse history) can offset the user vector

Using a vector space

- Similarities in the vector space used to derive correlation coefficients between user context and other users

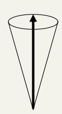


Recommendations from context

- Use these correlation coefficients to compute recommendations as before
- Challenge:
  - Must compute correlations at run time
- How can we make this efficient?
  - Restrict each user to a sparse vector
  - Precompute correlations to search terms
  - Compose $u + \varepsilon s$

Correlations at run time

- Other speedup
  - If we could restrict to users “near” the context
  - Problem - determining (say) all users within a certain “ball” of the context
  - Or *k* nearest neighbors, etc.



Modified vectors

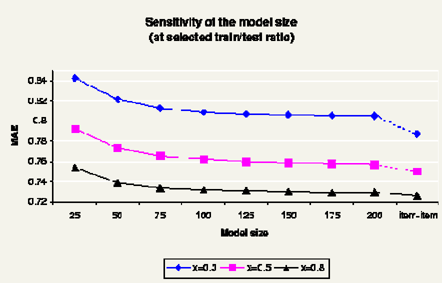
- Should context changes to vector be made permanent?
- Exponential decay?
- Can retain some memory of recent search/browse history

Think of how to do this efficiently

Measuring recommendations

- Typically, machine learning methodology
- Get a dataset of opinions; mask “half” the opinions
- Train system with the other half, then validate on masked opinions
  - Studies with varying fractions ≠ half
- Compare various algorithms (correlation metrics)
  - See McLaughlin and Herlocker, *SIGIR 2004*

k nearest neighbors - efficacy



Source: Sarwar/Karypis/Konstan/Riedl

Summary so far

- Content/context expressible in term space
- Combined into inter-user correlation
  - This is an algebraic formulation, but
  - Can also recast in the language of probability
- What if certain correlations are “constrained”
  - two users in the same department/zip code
  - two products by the same manufacturer?

RS Inputs - revisited

Past transactions from users:
- which docs viewed
- content/attributes of documents
- which products purchased
- pages bookmarked
- explicit ratings (movies, books …)

Current context:
- browsing history
- search(es) issued

Explicit profile info:
- Role in an enterprise
- Document taxonomies
- Interest profiles

Capturing role/domain

- Additional axes in vector space
  - Corporate org chart - departments
  - Product manufacturers/categories
- Make these axes “heavy” (weighting)
- Challenge: modeling hierarchies
  - Org chart, product taxonomy

Summary of Advantages of Pure CF

- No expensive and error-prone user attributes or item attributes
- Incorporates **quality** and **taste**
  - Want not just things that are similar, but things that are similar *and good*
- Works on any rate-able item
- One model applicable to many content domains
- Users understand it
  - It’s rather like asking your friends’ opinions

Resources

- GroupLens
  - http://citeseer.nj.nec.com/resnick94grouplens.html
  - http://www.grouplens.org
    - Has available data sets, including MovieLens
- Greening, Dan R. Building Consumer Trust with Accurate Product Recommendations: A White Paper on LikeMinds WebSell 2.1
  - http://dan.greening.name/profession/manuscripts/consumertrust/
- Shardanand/Maes
  - http://citeseer.ist.psu.edu/shardanand95social.html
- Sarwar et al.
  - http://citeseer.nj.nec.com/sarwar01itembased.html

Resources

- McLaughlin and Herlocker, SIGIR 2004
  - **http://portal.acm.org/citation.cfm?doid=1009050**
- CoFE CoFE “**Co**llaborative **F**iltering **E**ngine”
  - Open source Java
  - Reference implementations of many popular CF algorithms
  - http://eecs.oregonstate.edu/iis/CoFE