Recommender Systems

Recommender Systems

- Look at classic model and techniques
  - Items
  - Users
  - Recommend Items to Users
- Recommend new items based on:
  - similarity to items user liked in past: individual history
    "Content Filtering"
  - Liked by other users similar to this user: collaborative history
    "Collaborative Filtering"
  - Liked by other users: crowd history
    - easier case

Recommender System attributes

- Need explicit or implicit ratings by user
  - Purchase is 0/1 rating
    - Movie tickets
    - Books
- Have focused category
  - examples: music, courses, restaurants
  - hard to cross categories with content-based
  - easier to cross categories with collaborative-based
    - users share tastes across categories?

Content Filtering

- Items must have characteristics
- user values item

  $\Rightarrow$ values characteristics of item
- model each item as vector of weights of characteristics
  - much like vector-based IR
- user can give explicit preferences for certain characteristics

Buy/no buy prediction method: similarity with centroid

- Average vectors of items user bought
  - user's centroid
- Find similarity of new items to user's centroid
- Decide threshold for "buy" recommendation

5

Example

- user bought book 1 and book 2
- Average books bought = (0, 1, 0.5, 0)
- Score new books
  - dot product gives: score(A) = 0.5; score (B)= 1
- decide threshold for recommendation

	1st person	romance	mystery	sci-fi
book 1	0	1	1	0
book 2	0	1	0	0
new book A	1	.5	0	0
new book B	0	1	0	.2

6

Method issues

- Centroid best way to build a preference vector?
- What metric use for similarity between new items and preference vector?
  - Normalization?
- What if users give ratings?
  - Centroid per rating value?
- how include explicit user preferences
- How determine threshold?

7

Example with explicit user preferences

How use scores of books bought?

Try: preference vector p where component k =

user pref for characteristic k if ≠ 0

avg. comp. k of books bought when user pref =0

0 pref for user = "don't care"

p=(0, 1, 0.5, -5)

New scores?

p•A = 0.5

p•B = 0

	1st per	rom	mys	sci-fi
user pref	**0**	**1**	**0**	**-5**
book 1	0	1	1	0
book 2	0	1	0	0
new A	1	.5	0	0
new B	0	1	0	.2

8

Other methods: machine learning

- Major alternatives based on classifiers
  - Training set: items bought and not bought
  - Train classifier – many algorithms
  - Classify new item as buy/no buy
- Observations
  - Uses books not bought. Problems?
  - Multiple rating value

    Can use multiple classes

9

Limitations of Content Filtering

- Can only recommend items similar to those user rated highly
- New users
  - Insufficient number of rated items
- Only consider features explicitly associated with items
  - Do not include attributes of user

10

Applying content filtering methods to search

- Characterize documents (info. objects)
  - topic analysis?
  - other properties, e.g.:
    - Domain of source
    - Date of publication/update
- Characterize individuals
  - deduce from properties of objects interact with
  - user provided preferences

11

Applying content filtering methods to search, cont.

- Query filters documents to consider
  - Convert query to topic-based?
    - Too error prone?
  - Modify query to bias towards user's preferred topics?
- Ranking is recommendation
  - Use similarity to user's characterization

12

Example study:
Personalizing Web Search Using Long-term Browsing History (in *WSDM11*)

- Goal: rerank
  - top 50 results from Google query
- Query is initial filter to get results from Google
- Strategy:
  - score snippets from search result against user profile
  - rerank based on snippet score

User Characterization

- Selection of info
  - list of visited URLs w/ number visits
  - list of past search queries and pages clicked
  - list of terms with weights for content of pages visited
- Studies selection of methods
  - what sources of terms use
    - body, title tags, metainfo like keywords
  - weights for terms
    - tf-idf
      - where get idf?
    - best “modified BM25”- a “log odds measure”

$W_{modBM25}$ weighting

N = # documents on Web – estimated

n_{ti} = # docs on Web containing term t_i - estimated

R = # documents in user browser history

r_{ti} = # docs in user browser history that contain term t_i

$W_{modBM25}(t_i)$ =

$$\log \left(\frac{(r_{ti} + 0.5)(N - n_{ti} + 0.5)}{(n_{ti} + 0.5)(R - r_{ti} + 0.5)} \right)$$

Documents

- Characterization
  - words in snippet
  - original rank by Google search
- Scoring
  - best performing: language-based model
    - based on content (terms)
    - adjustments for
      - URLs previously visited
      - original rank of snippet in search

Scoring a snippet

N_{si}= # unique words in snippet s_i
r_{si} = rank of snippet s_i in original search results
n_i = # previous visits by user to web page with snippet s_i
$w(t_k)$ = weigth of term t_k in user profile
w_{total} = sum of all term weights in user profile

$$score_{lang.\ model}(s_i) = \Sigma_{k=0}^{N_{si}} \log ((w(t_k) +1)/w_{total})$$

- modif. for URLs previously visited:

$score_{w/URL}(s_i)= score(s_i)*(1+\alpha*n_i)$ *parameter* α

- modif to acct. for orig. rank:

$$score_{w/orig}(s_i) = score(s_i)*(1/(1+\log (r_{si})))$$

Evaluation

- "offline" evaluation:
  - relevance judgments by volunteers
  - used to select best of algorithmic variations
- online evaluation of best variations:
  - add-on to Browser by volunteers
  - interleave original results (no personalization) with results reranked by snippet score
  - record clicks by user – which list from

Results

- Offline: normalized DCG, avg. of 72 queries
  - Google's ranking w/out personalization: 0.502
  - best-performing of variations for reranking: 0.573
- Online
  - 8% queries: # clicks from original and reranked same
  - of rest: 60.5% queries: more clicks from reranked
    39.5% queries: more clicks from original

Observation

- Reranking can be done completely in browser if enough space for data for user profile

What we've just seen:

Applying content filtering to search

Now back to recommender systems:

Collaborative Filtering

Collaborative Filtering

- Recommend new items liked by other users similar to this user
- need items already rated by user ***and other users***
- don’t need characteristics of items
  - each rating by individual user becomes characteristic
- Can combine with item characteristics
  - hybrid content/collaborative

21

Major method types

- Nearest neighbor
  - Use similarity function
  - Prediction based on previously rated items
- Matrix Factorization
  - “Latent factors”
  - Matrix decomposition
- Both use (user × item) matrix
  - vector similarity

22

Example of nearest neighbor: Preliminaries

- Notation
  - $r(u,i)$ = rating of i^{th} item by user u
  - I_u = set of items rated by user u
  - $I_{u,v}$ = set of items rated by both users u and v
  - $U_{i,j}$ = set of users that rated items i and j
- Adjust scales for user differences
  - Use average rating by user u:

    $$r_u^{avg} = (1/|I_u|) * \sum_{i \text{ in } I_u} r(u,i)$$
  - Adjusted ratings: $r_{adj}(u,i) = r(u,i) - r_u^{avg}$

23

One choice of similarity function: User Similarities

- similarity between users u and v
  - Pearson correlation coefficient

$$\text{sim}(u,v) = \frac{\sum_{i \text{ in } I_{u,v}} (r_{adj}(u,i) * r_{adj}(v,i))}{\left(\sum_{i \text{ in } I_{u,v}} (r_{adj}(u,i))^2 * \sum_{i \text{ in } I_{u,v}} (r_{adj}(v,i))^2 \right)^{1/2}}$$

24

Predicting User's rating of new item: User-based

For item i not rated by user u

$$r^{pred}(u,i) = r_u^{avg} + \frac{\sum_{v \text{ in } S} (sim(u,v) * r_{adj}(v, i))}{\sum_{v \text{ in } S} |sim(u,v)|}$$

S can be all users who have rated i or just those users *most similar* to u

Collaborative filtering example

user ratings

	book 1	book 2	book 3	book 4
user 1	5	1	2	0
user 2	x	5	2	5
user 3	3	1	x	2
user 4	4	0	2	?

adj. user ratings

	book 1	book 2	book 3	book 4
user 1	3	-1	0	-2
user 2	x	1	-2	1
user 3	1	-1	x	0
user 4	2	-2	0	?

Collaborative filtering example

- sim(u1,u4) = $(6+2)/(10*8)^{1/2}$ = .894
- sim(u2,u4) = $(-2)/(5*4)^{1/2}$ = -.447
- sim(u3,u4) = $(2+2)/(2*8)^{1/2}$ = 1

- predict r(u4, book4) = $2 + \frac{(-2)*.894 + 1*(-.447) + 0*1}{.894 + .447 + 1}$

$$= 2 - .955 \approx 1$$

Another choice of similarity function: Item Similarities

- similarity between items i and j
  - vector of ratings of users in $U_{i,j}$
  - cosine measure using adjusted ratings

$$sim(i,j) = \frac{\sum_{u \text{ in } U_{i,j}} (r_{adj}(u,i) * r_{adj}(u, j))}{\left(\sum_{u \text{ in } U_{i,j}} (r_{adj}(u,i))^2 \sum_{u \text{ in } U_{i,j}} (r_{adj}(u, j))^2 \right)^{1/2}}$$

Predicting User's rating of new item: Item-based

For item i not rated by user u

$$r^{\text{item-pred}}(u,i) = \frac{\sum_{j \text{ in } T} (sim(i,j)*r(u,j))}{\sum_{j \text{ in } T} |sim(i,j)|}$$

T can be all items in I_u or just items *most similar* to i

- Prediction uses only u's ratings, but similarity uses other users' ratings

29

Limitations

- May not have enough ratings for new users
- New items may not be rated by enough users
- Need "critical mass" of users
  - All similarities based on user ratings

But can take user "out of comfort zone"

30

Applying nearest-neighbor collab. filtering concepts to search

- Collaborative histories
  - How determine user similarity?
    - Clicking URL = buying product?
    - Behavior on only identical searches?
    - Exact URLs or general topic interests?
      - Hybrid content-based and behavior-based
    - Computational expense?
      - Argues for general topic-interest characterizations
  - How apply similarity?
    - Same search? or Same topic of search?
    - Bias ranking? or Bias topics of results?

31

Example

from A Large-scale Evaluation and Analysis of Personalize Search Strategies (in *WWW07*)

- Goal: rerank search results
- Based on query log history – clicks as ratings
- Also uses 67 pre-defined topic categories
- Strategy:
  - get similarity of users based on user history of visited pages
  - find K most similar users to user doing search

    K nearest neighbor; use K=50
  - calc. score for each result of search based on click history of K nearest neighbors
  - rerank results of search based on score

32

Details

from A Large-scale Evaluation and Analysis of Personalize Search Strategies (in *WWW07*)

P(u) = collection of Web pages visited by user u in the past

$$P(p|u) = \frac{\text{\# times u clicked on page p in past}}{\text{total \# times u clicked on a page in past}}$$

w(p) = log(total # users / # users visited page p)

"impact weight" - idf-like

$\boldsymbol{c}$(p) = "category vector" for page p

do classification of page

vector gives confidence # for top 6 categories (other entries 0)

User profile $\boldsymbol{c}_\ell(u) = \Sigma_{p \text{ in } P(u)} P(p|u)w(p)\boldsymbol{c}(p)$ **hybrid!**

User similarity $\text{sim}(u_1, u_2) = \dfrac{\boldsymbol{c}_\ell(u_1)\bullet\boldsymbol{c}_\ell(u_2)}{||\boldsymbol{c}_\ell(u_1)||\ \ ||\boldsymbol{c}_\ell(u_2)||}$

33

Details

from A Large-scale Evaluation and Analysis of Personalize Search Strategies (in *WWW07*)

$S_k(u_a)$ denotes k nearest neighbors of user u_a

click history:

$|\text{clicks}(q,p,u_s)|$ = # clicks on pg p by user u_s on past query q

$|\text{clicks}(q,*,u_s)|$ = # clicks overall by user u_s on past query q

the score of a page p for query q and user u:

$$S(q,p,u) = \frac{\Sigma_{u_s \text{ in } S_k(u)} \text{sim}(u_s,u) * |\text{clicks}(q,p,u_s)|}{\beta + \Sigma_{u_s \text{ in } S_k(u)} |\text{clicks}(q,*,u_s)|}$$

β is a "smoothing factor"; taken to be 0.5

34

Experiments

from A Large-scale Evaluation and Analysis of Personalize Search Strategies (in *WWW07*)

- Data set: MSN query logs 12 days August 2006
  sampled 10,000 distinct users
  used 11 days for training, last day for testing
  ~ 4000 test queries
- Action, for each user and query
  - re-rank top 50 results using a "fusion" of original rank and order given by page scores S(q,p,u)
- Evaluation: 2 metrics
  1. a DCG-like metric with clicking indicating relevance
  2. average rank of clicked items

35

Results

from A Large-scale Evaluation and Analysis of Personalize Search Strategies (in *WWW07*)

- Good news:
  re-ranking improves over original ranking
- So-so news:
  improvement is 3.62% on queries where there is room for improvement
- Not so good news:
  non-collaborative personalization improves 3.68%

$$S(q,p,u) = \frac{|\text{clicks}(q,p,u)|}{\beta + |\text{clicks}(q,*,u)|}$$

36

Where are we?

✓Refinement/Personalization of results
- Study techniques of

  Recommender systems

  ✓Content filtering
    - Applying content filtering to search

  – Collaborative filtering
    ✓Nearest neighbor methods
      – Applying nearest neighbor method to search

NEXT • Matrix factorization methods

37