Refining and Personalizing Searches

Targets

- collection
- ➢ query
- satisfying documents
  - increase set?
- ➢ ranking

Themes

- Explicit feedback versus search history
- Personalized history versus group history

Refine initially: query

- Help user get better query
- Commonly, query expansion
  - add synonyms
    - Improve recall
    - Hurt precision?
    - Sometimes done automatically – with care
  - Modify based on prior searches
    - Not automatic
    - All prior searches - eg. suggested search terms
      vs
    - *your* prior searches

Refining after search

- Use user feedback
  or
- Approximate feedback with first results
  - Pseudo-feedback
  - Example: “Yahoo assist” (?still)
- change ranking of current results
  or
- search again with modified query

Explicit user feedback

- User must participate
- User marks (some) relevant results
  or
- User changes order of results
  - Can be more nuanced than relevant or not
  - Can be less accurate than relevant or not
    - Example: User moves 10th item to first
      - says 10th better than first 9
      - Does not say which, if any, of first 9 relevant

User feedback in classic vector model

- User marks top p documents for relevance
  - p = 10 to 20 "typical"
- Construct new weights for terms in query vector
  - Modifies query
  - Could use just on initial results to re-rank

7

Deriving new query for vector model

For collection C of n doc.s

- Let C_r denote set all relevant docs in collection,

Perfect knowledge Goal:

Vector $\mathbf{q}_{opt}$ =

$1/|C_r|$ * (sum of all vectors d_j in C_r) -

$1/(n- |C_r|)$ * (sum of all vectors d_k not in C_r)

centroids

8

Deriving new query for vector model: Rocchio algorithm

Give query **q** and relevance judgments for a subset of retrieved docs

- Let D_r denote set of docs judged relevant
- Let D_{nr} denote set of docs judged not relevant

Modified query:

Vector $\mathbf{q}_{new}$ = $\alpha\mathbf{q}$ +

$\beta/|D_r|$ * (sum of all vectors $\mathbf{d}_j$ in D_r) -

$\gamma/(|D_{nr}|)$ * (sum of all vectors $\mathbf{d}_k$ in D_{nr})

For tunable weights α, β, γ

9

Remarks on new query

- α: importance original query
- β: importance effect of terms in relevant docs
- γ: importance effect of terms in docs not relevant

- Usually terms of docs not relevant are least important
  - Reasonable values α=1, β=.75, γ=.15
- Reweighting terms leads to long queries
  - ***Many*** more non-zero elements in query vector $\mathbf{q}_{new}$
  - Can reweight only most important (frequent?) terms
- Most useful to improve recall
- Users don't like: work + wait for new results

10

Simple example user feedback in vector model

- **q** = (1,1,0,0)
- Relevant: **d1** = (1,0,1,1)
  d2 = (1,1,1,1)
- Not relevant: **d3**=(0,1,1,0)
- α, β, γ = 1
- $\mathbf{q}_{new}$ = (1,1,0,0) + (1, 1/2, 1, 1) - (0,1,1,0)
  = (2, 1/2, 0, 1)

Term weights change New term

Observe: Can get negative weights

11

Re-ranking using explicit feedback

- Algorithms usually based on machine learning
  - Learn ranking function that best matches partial ranking given
- Simple example
  - 2007ish: Google experiment; only affects repeat of same search
  - 2008: became SearchWiki feature for Google accounts
  - 2010: functionality reduced to "starred" results list

12

Implicit user feedback

- Click-throughs
  - Use as relevance judgment
  - Use as reranking:
    When click result, moves it ahead of all results didn't click that come before it
  - Problems?
- Better implicit feedback signals?

13

Behavior History

- Going beyond behavior on ***same*** query.
- Personal history versus Group history
- Group history
  - Primarily search history
    - Google's claim Bing copies
- Personal history
  - Searches
  - Other behavior – browsing, mail?, …
  - Characterize interests: topics

14

Collaborative history

Group history + personal history =>
History of people "like" you

How characterize?
- Shared behaviors
- Shared topics

15

Example: 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-based"
  - Liked by other users similar to this user: collaborative history
    "Collaborative Filtering"
  - Liked by other users: group history
    - easier case

16

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?

17

Content-based recommendation

- Items must have characteristics
- user values item
  ⇒ 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

18

Content-based example

- user bought book 1 and book 2
  - what if actually rated?
- 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

19

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

20

Content-based: issues

- Vector-based one alternative
- Major alternatives based on machine-learning
- For vector based
  - how build a preference vector
    - how combined vectors for items rated by user
      - our example only 0/1 rating
    - how include explicit user preferences
  - what metric use for similarity between new items and preference vector
  - normalization
  - threshold?

21

Limitations of Content-based

- 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

22

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

23

Method types

(see Adomavicius and Tuzhilin paper)

- Memory-Based
  - Similar to vector model
  - Use (user × item) matrix
  - Use similarity function
  - Prediction based on previously rated items
- Model-Based
  - Machine-learning methods
  - Model of probabilities of (users × items)

24

Memory-Based: 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: $r_{adj}(u,i) = r(u,i) - r_u^{avg}$

One Memory-Based method: User Similarities

- similarity between users u and v
  - Pearson correlation coefficient

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

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

  $= 2 - .955 \approx 1$

One Memory-Based Method: 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

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

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

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

31

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

32

Applying concepts to search

- Individual histories
  - Characterize individual by topic interest
    - Properties of objects interact with
  - Characterize query by related topics
    - Role of terms of query in topic
  - Modify query to bias to shared topics
  - Modify ranking to prefer shared topics

33

Applying concepts to search

- Collaborative histories
  - How determine user similarity?
    - Behavior on identical searches?
    - Overlap of general topic interests?
      - From overlapping behaviors
      - Hybrid content-based and behavior-based
    - Computational expense?
      - Argues for general topic-interest characterizations
  - How apply similarity?
    - Same search? Bias ranking?
    - Same topic of search? Bias topics of results?

34

Refining PageRank

$$\boldsymbol{pr} = (\alpha/n, \alpha/n, \ldots \alpha/n)^T + (1-\alpha)\, L^T \boldsymbol{pr}$$

- let $\boldsymbol{v} = (1/n, 1/n, \ldots 1/n)$
- rewrite $\boldsymbol{pr} = (\alpha)\boldsymbol{v}^T + (1-\alpha)\, L^T \boldsymbol{pr}$
- Refinement choices
  - change ***v***
  - change ***L***

35

"Topic Senstive" PageRank

Haveliwala

- Use pre-defined topics
  - Open Directory Project
    - "the largest, most comprehensive human-edited directory of the Web."
    - 16 top-level topics
- Each page has PageRank for each topic
- Calculate similarity of query to each topic
  - Use linear combination of topic PageRanks based on similarity values query to topic

36

Personalized PageRank

Kamvar et. al.

- Random leaps are biased by personal interests – change ***v***
- Combined with use of block structure to make more efficient:
  - Divide Web graph into blocks (clusters)
    - Use high-level domains (e.g. princeton.edu)
  - Calc. local PageRank within each block
  - Collapse each block into 1 node – new graph
    - Weighted edges between nodes
  - Calc. PageRank with biased leaps for block structure
  - Weight local PageRanks with block PageRank
    - Use to initialize power calcuation

Summary

- Looked at several techniques for modifying search
  - Explicit User feedback
    - revise query
  - Implicit User feedback – behavior history
    - Individual history
    - Group history
    - Collaborative history
  - Recommender systems
  - Modifying PageRank