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
  \( p = 10 \text{ to } 20 \) “typical"
- Construct new weights for terms in query vector
  - Modifies query
  - Could use just on initial results to re-rank

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 \( q_{opt} = \)
\[
\frac{1}{|C_r|} \times (\text{sum of all vectors } d_j \text{ in } C_r) - \frac{1}{(n-|C_r|)} \times (\text{sum of all vectors } d_k \text{ not in } C_r)
\]

Remarks on new query

- \( \alpha \): importance original query
- \( \beta \): importance effect of terms in relevant docs
- \( \gamma \): importance effect of terms in docs not relevant

- Usually terms of docs not relevant are least important
  - Reasonable values \( \alpha = 1, \beta = .75, \gamma = .15 \)
- Reweighting terms leads to long queries
  - \( \text{Many} \) more non-zero elements in query vector \( q_{new} \)
  - Can reweight only most important (frequent?) terms
- Most useful to improve recall
- Users don’t like: work + wait for new results

Simple example user feedback in vector model

- \( q = (1,1,0,0) \)
- Relevant: \( d_1 = (1,0,1,1) \)
  \( d_2 = (1,1,1,1) \)
- Not relevant: \( d_3 = (0,1,1,0) \)
  - \( \alpha, \beta, \gamma = 1 \)
  - \( q_{new} = (1,1,0,0) + (1, 1/2, 1, 1) - (0,1,1,0) \)
  \( = (2, 1/2, 0, 1) \)

Term weights change

New term

Observe: Can get negative weights

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
    - 2012: replaced by +1?
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?

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

Collaborative history

Group history + personal history =>
History of people “like” you

How characterize?
- Shared behaviors
- Shared topics

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

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-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
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

<table>
<thead>
<tr>
<th>1st per</th>
<th>romance</th>
<th>mystery</th>
<th>sci-fi</th>
</tr>
</thead>
<tbody>
<tr>
<td>book 1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>book 2</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>new book A</td>
<td>1</td>
<td>.5</td>
<td>0</td>
</tr>
<tr>
<td>new book B</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Example with explicit user preferences

How use scores of books bought?
Try: preference vector $p$ where component $k =$
- user pref for characteristic $k$ if $k$ ≠ 0
- avg. comp. $k$ of books bought when user pref =0
- 0 pref for user = “don’t care”

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

<table>
<thead>
<tr>
<th>1st per</th>
<th>romance</th>
<th>mystery</th>
<th>sci-fi</th>
</tr>
</thead>
<tbody>
<tr>
<td>user pref</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

New scores?
$p \cdot A = 0.5$
p $\cdot$ B = 0

<table>
<thead>
<tr>
<th>1st per</th>
<th>romance</th>
<th>mystery</th>
<th>sci-fi</th>
</tr>
</thead>
<tbody>
<tr>
<td>book 1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>book 2</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>new A</td>
<td>1</td>
<td>.5</td>
<td>0</td>
</tr>
<tr>
<td>new B</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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?

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

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

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

- Goal: rerank
  - top 50 results from Google query
- Strategy:
  - score snippets from search result against user profile
  - rerank based on snippet score
- Selection of info for user profile
  - 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
Personalizing Web Search Using Long-term Browsing History, cont

Studies selection of methods for
• user profile: what sources of terms use
• user profile: weights for terms
  – tf-idf
    • where get idf?
      – "modified BM25" - a "log odds measure"
• scoring
  – language model with adjustments for
    • URLs previously visited
    • original rank of snippet in search

Equations for Personalizing Web Search Using Long-term Browsing History

\[ W_{\text{modBM25}}(t_i) = \frac{\log\left( \frac{(r_{ti} + 0.5)(N-n_{ti} + 0.5)}{(n_{ti} + 0.5)(R - r_{ti} + 0.5)} \right)}{N} \]

\[ N_s = \# \text{ unique words in snippet } i \]
\[ r_s = \text{ rank of snippet } i \text{ in original search results} \]
\[ n_i = \# \text{ previous visits by user to web page with snippet } s_i \]
\[ w(t_k) = \text{ weight of term } t_k \text{ in user profile} \]

\[ \text{score}_{\text{lang. model}}(s_i) = \sum_{k=0}^{N} \log \left( \frac{w(t_k) + 1}{w_{\text{total}}} \right) \]

\[ \text{score}_{\text{w/URL}}(s_i) = \text{score}(s_i)(1 + v \cdot n_i) \] where \( v = 10 \) is used in the experiments

\[ \text{score}_{\text{w/orig}}(s_i) = \frac{\text{score}(s_i)}{1 + \log(r_s)} \]

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
  – 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

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

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)
Memory-Based: Preliminaries

- Notation
  - \( r(u,i) \): rating of the \( i \)th item by user \( u \)
  - \( I_u \): set of items rated by user \( u \)
  - \( I_{uv} \): set of items rated by both users \( u \) and \( v \)
  - \( U_{ij} \): set of users that rated items \( i \) and \( j \)

- Adjust scales for user differences
  - Use average rating by user \( u \):
    \[
    r_{u \text{avg}} = \frac{1}{|I_u|} \sum_{i \in I_u} r(u,i)
    \]
  - Adjusted ratings:
    \[
    r_{adj}(u,i) = r(u,i) - r_{u \text{avg}}
    \]

One Memory-Based method: User Similarities

- similarity between users \( u \) and \( v \)
  - Pearson correlation coefficient
    \[
    \text{sim}(u,v) = \frac{\sum_{i \in I_{uv}} (r_{adj}(u,i) * r_{adj}(v,i))}{\sqrt{\left( \sum_{i \in I_{uv}} (r_{adj}(u,i))^2 \right) \left( \sum_{i \in I_{uv}} (r_{adj}(v,i))^2 \right)}}
    \]

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

For item \( i \) not rated by user \( u \)

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

S can be all users or just users most similar to \( u \)

Collaborative filtering example

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>user 1</td>
<td>3</td>
<td>-1</td>
<td>0</td>
<td>-2</td>
</tr>
<tr>
<td>user 2</td>
<td>x</td>
<td>5</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>user 3</td>
<td>3</td>
<td>1</td>
<td>x</td>
<td>2</td>
</tr>
<tr>
<td>user 4</td>
<td>4</td>
<td>0</td>
<td>2</td>
<td>?</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>user 1</td>
<td>3</td>
<td>-1</td>
<td>0</td>
<td>-2</td>
</tr>
<tr>
<td>user 2</td>
<td>x</td>
<td>1</td>
<td>-2</td>
<td>1</td>
</tr>
<tr>
<td>user 3</td>
<td>1</td>
<td>-1</td>
<td>x</td>
<td>0</td>
</tr>
<tr>
<td>user 4</td>
<td>2</td>
<td>-2</td>
<td>0</td>
<td>?</td>
</tr>
</tbody>
</table>

One Memory-Based Method: Item Similarities

- similarity between items \( i \) and \( j \)
  - vector of ratings of users in \( U_{ij} \)
  - cosine measure using adjusted ratings
    \[
    \text{sim}(i,j) = \frac{\sum_{u \in U_{ij}} (r_{adj}(u,i) * r_{adj}(u,j))}{\sqrt{\left( \sum_{u \in U_{ij}} (r_{adj}(u,i))^2 \right) \left( \sum_{u \in U_{ij}} (r_{adj}(u,j))^2 \right)}}
    \]

Collaborative filtering example

- \( \text{sim}(u1,u4) = \frac{6+2}{10*8} \approx .894 \)
- \( \text{sim}(u2,u4) = \frac{-2}{5*4} \approx .447 \)
- \( \text{sim}(u3,u4) = \frac{2+2}{2*8} = 1 \)

- predict \( r(u4, \text{book}4) = 2 + \frac{-2* .894 + 1*(-.447) + 0*1}{.894 + .447 + 1} = 2 * .955 = 1 \)
**Predicting User’s rating of new item:**

**Item-based**

For item i not rated by user u

\[
\text{item-pred}(u, i) = \frac{\sum_{j \in T} (\text{sim}(i, j) \times r(u, j))}{\sum_{j \in T} |\text{sim}(i, j)|}
\]

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

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

**Global effects**

Effects over many or all of ratings

- different users have different rating scales
- metadata (attributes) for items and/or users
  - hybrid content/collaborative
- date of rating
- trend of user’s ratings over time
- trend of item’s ratings over time

Reference: Scalable Collaborative Filtering w/ Jointly Derived Neighborhood Interpolation Weights, Bell and Koren, IEEE Intern. Conf. Data Mining (part of winning Netflix contest team)

**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

**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?

**Example**

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

- Goal: rerank search results
- Based on query log history – clicks
- 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

**Details**

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

\[ P(u) = \text{collection of Web pages visited by user u in the past} \]
\[ P(p,u) = \frac{\text{# times u clicked on page p in past}}{\text{total # time u clicked on a page in past}} \]
\[ w(p) = \log(\text{total # users} / \text{# users visited page p}) \]

“impact weight” = idf-like
\[ \phi(p) = \text{“category vector” for page p} \]

- do classification of page
- vector gives confidence # for top 6 categories

User profile
\[ \phi(u) = \sum_{p \in P(u)} P(p,u) w(p) \phi(p) \]

User similarity
\[ \text{sim}(u_1, u_2) = \frac{\phi(u_1) \cdot \phi(u_2)}{||\phi(u_1)|| \cdot ||\phi(u_2)||} \]
Details

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

click history:
|clicks(q,p,u)| = # clicks on pg p by user $u$ on past query q
|clicks(q,*,u)| = # clicks overall by user $u$ on past query q

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

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

$\beta$ is a “smoothing factor”; taken to be 0.5

Refining PageRank

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

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

“Topic Sensitive” PageRank

- 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

Personalized PageRank

- 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 calculation

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