13. Personalization

Most slides were adapted from Stanford CS 276 course.
Ambiguity

- Unlikely that a short query can unambiguously describe a user’s information need

- For example, the query [chi] can mean
  - Calamos Convertible Opportunities & Income Fund quote
  - The city of Chicago
  - Balancing one’s natural energy (or ch’i)
  - Computer-human interactions
Personalization

- Ambiguity means that a single ranking is unlikely to be optimal for all users
- Personalized ranking is the only way to bridge the gap
- Personalization can use
  - Long term behavior to identify user interests, e.g., a long term interest in user interface research
  - Short term session to identify current task, e.g., checking on a series of stock tickers
  - User location, e.g., MTA in New York vs Baltimore
  - Social network
  - ...
Potential for Personalization

[Teevan, Dumais, Horvitz 2010]

- How much can personalization improve ranking? How can we measure this?

- Ask raters to explicitly rate a set of queries
  - But rather than asking them to guess what a user’s information need might be …
  - … ask which results they would personally consider relevant

- Use self-generated and pre-generated queries
Computing potential for personalization

- For each query $q$
  - Compute average rating for each result
  - Let $R_q$ be the optimal ranking according to the average rating
  - Compute the NDCG value of ranking $R_q$ for the ratings of each rater $i$
  - Let $\text{Avg}_q$ be the average of the NDCG values for each rater
- Let $\text{Avg}$ be the average $\text{Avg}_q$ over all queries
- Potential for personalization is $(1 - \text{Avg})$
## Example: NDCG values for a query

<table>
<thead>
<tr>
<th>Result</th>
<th>Rater A</th>
<th>Rater B</th>
<th>Average rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>1</td>
<td>0</td>
<td>0.5</td>
</tr>
<tr>
<td>D2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>D3</td>
<td>0</td>
<td>1</td>
<td>0.5</td>
</tr>
<tr>
<td>D4</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D5</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D6</td>
<td>1</td>
<td>0</td>
<td>0.5</td>
</tr>
<tr>
<td>D7</td>
<td>1</td>
<td>2</td>
<td>1.5</td>
</tr>
<tr>
<td>D8</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D9</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D10</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>NDCG</td>
<td>0.88</td>
<td>0.65</td>
<td></td>
</tr>
</tbody>
</table>

Average NDCG for raters: 0.77
Example: NDCG values for optimal ranking for average ratings

<table>
<thead>
<tr>
<th>Result</th>
<th>Rater A</th>
<th>Rater B</th>
<th>Average rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>D7</td>
<td>1</td>
<td>2</td>
<td>1.5</td>
</tr>
<tr>
<td>D2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>D1</td>
<td>1</td>
<td>0</td>
<td>0.5</td>
</tr>
<tr>
<td>D3</td>
<td>0</td>
<td>1</td>
<td>0.5</td>
</tr>
<tr>
<td>D6</td>
<td>1</td>
<td>0</td>
<td>0.5</td>
</tr>
<tr>
<td>D4</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D5</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D8</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D9</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D10</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>NDCG</td>
<td>0.98</td>
<td>0.96</td>
<td></td>
</tr>
</tbody>
</table>

Average NDCG for raters: 0.97
Example: Potential for personalization

<table>
<thead>
<tr>
<th>Result</th>
<th>Rater A</th>
<th>Rater B</th>
<th>Average rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>D7</td>
<td>1</td>
<td>2</td>
<td>1.5</td>
</tr>
<tr>
<td>D2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>D1</td>
<td>1</td>
<td>0</td>
<td>0.5</td>
</tr>
<tr>
<td>D3</td>
<td>0</td>
<td>1</td>
<td>0.5</td>
</tr>
<tr>
<td>D6</td>
<td>1</td>
<td>0</td>
<td>0.5</td>
</tr>
<tr>
<td>D4</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D5</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D8</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D9</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D10</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>NDCG</td>
<td>0.98</td>
<td>0.96</td>
<td></td>
</tr>
</tbody>
</table>

Potential for personalization: 0.03
Potential for personalization graph

Number of raters

NDCG

Potential for personalization

Group

Individual

Web
PERSONALIZING SEARCH
Personalizing search

[Pitkow et al. 2002]

- Two general ways of personalizing search
  - Query expansion
    - Modify or augment user query
    - E.g., query term “IR” can be augmented with either “information retrieval” or “Ingersoll-Rand” depending on user interest
    - Ensures that there are enough personalized results
  - Reranking
    - Issue the same query and fetch the same results ...
    - ... but rerank the results based on a user profile
    - Allows both personalized and globally relevant results
User interests

- Explicitly provided by the user
  - Sometimes useful, particularly for new users
  - ... but generally doesn’t work well

- Inferred from user behavior and content
  - Previously issued search queries
  - Previously visited Web pages
  - Personal documents
  - Emails

- Ensuring privacy and user control is very important
Relevance feedback perspective

[Teevan, Dumais, Horvitz 2005]
Binary Independence Model

- Estimating RSV coefficients in theory
- For each term $i$ look at this table of document counts:

<table>
<thead>
<tr>
<th>Documents</th>
<th>Relevant</th>
<th>Non-Relevant</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_i=1$</td>
<td>$s_i$</td>
<td>$n_i-s_i$</td>
<td>$n_i$</td>
</tr>
<tr>
<td>$x_i=0$</td>
<td>$S-s_i$</td>
<td>$N-n_i-S+s_i$</td>
<td>$N-n_i$</td>
</tr>
<tr>
<td>Total</td>
<td>$S$</td>
<td>$N-S$</td>
<td>$N$</td>
</tr>
</tbody>
</table>

- Estimates:
  
  \[ p_i \approx \frac{s_i}{S} \quad r_i \approx \frac{(n_i-s_i)}{(N-S)} \]

  \[ c_i \approx K(N,n_i,S,s_i) = \log \frac{s_i/(S-s_i)}{(n_i-s_i)/(N-n_i-S+s_i)} \]

For now, assume no zero terms. See later lecture.
Personalization as relevance feedback

- Documents containing term $i$
- Relevant documents
- All documents

$N' = N + S$

$n_i' = n_i + s_i$
Reranking

- BM25 scoring

\[ \sum c_i \times tf_i \]

- Use updated weight \( c_i \) in BM25

\[ c_i = \log \frac{(s_i + 0.5)}{(S - s_i + 0.5)} \cdot \frac{(N - n_i + 0.5)}{(n_i + 0.5)} \approx \log \frac{(s_i + 0.5)}{(S - s_i + 0.5)} + IDF_i \]

where we have used

\[ N' = N + S \]

\[ n'_i = n_i + s_i \]
Corpus representation

- Estimating $N$ and $n_i$

- Many possibilities
  - $N$: All documents, query relevant documents, result set
  - $n_i$: Full text, only titles and snippets

- Practical strategy
  - Approximate corpus statistics from result set
  - ... and just the title and snippets
  - Empirically seems to work the best!
User representation

- Estimating $S$ and $s_i$

- Estimated from a local search index containing
  - Web pages the user has viewed
  - Email messages that were viewed or sent
  - Calendar items
  - Documents stored on the client machine

- Best performance when
  - $S$ is the number of local documents matching the query
  - $s_i$ is the number that also contains term $i$
Document and query representation

- Document represented by the title and snippets

- Query is expanded to contain words near query terms (in titles and snippets)
  - For the query [cancer] add underlined terms

  The American Cancer Society is dedicated to eliminating cancer as a major health problem by preventing cancer, saving lives, and diminishing suffering through ...

- This combination of corpus, user, document, and query representations seem to work well
User location

- User location is one of the most important features for personalization
  - Country
    - Query [football] in the US vs the UK
  - State/Metro/City
    - Queries like [zoo], [craigslist], [giants]
  - Fine-grained location
    - Queries like [pizza], [restaurants], [coffee shops]
Challenges

- Not all queries are location sensitive
  - [facebook] is not asking for the closest Facebook office
  - [seaworld] is not necessarily asking for the closest SeaWorld

- Different parts of a site may be more or less location sensitive
  - NYTimes home page vs NYTimes Local section

- Addresses on a page don’t always tell us how location sensitive the page is
  - Stanford home page has address, but not location sensitive
Key idea

[Bennett et al. 2011]

- Usage statistics, rather than locations mentioned in a document, best represent where it is relevant
  - I.e., if users in a location tend to click on that document, then it is relevant in that location

- User location data is acquired from anonymized logs (with user consent, e.g., from a widely distributed browser extension)
  - User IP addresses are resolved into geographic location information
Location interest model

- Use the logs data to estimate the probability of the location of the user given they viewed this URL

$$P(location = x \mid URL)$$
Location interest model

- Use the logs data to estimate the probability of the location of the user given they viewed this URL

\[ P(location = x | URL) \]
Learning the location interest model

- For compactness, represent location interest model as a mixture of 5-25 2-d Gaussians ($x$ is [lat, long])

$$P(location = x \mid URL) = \sum_{i=1}^{n} w_i N(x; \mu_i, \Sigma_i)$$

$$= \sum_{i=1}^{n} \frac{w_i}{(2\pi)^{2/2} |\Sigma_i|^{1/2}} e^{-\frac{1}{2}(x-\mu_i)^T \Sigma_i^{-1} (x-\mu_i)}$$

- Learn Gaussian mixture model using EM
  - Expectation step: Estimate probability that each point belongs to each Gaussian
  - Maximization step: Estimate most likely mean, covariance, weight
More location interest models

- Learn a location-interest model for queries
  - Using location of users who issued the query
- Learn a background model showing the overall density of users
Topics in URLs with high \( P(\text{user location} \mid \text{URL}) \)
Location sensitive features

- Non-contextual features (user-independent)
  - Is the query location sensitive? What about the URLs?
  - Feature: Entropy of the location distribution
    - Low entropy means distribution is peaked and location is important
  - Feature: KL-divergence between location model and background model
    - High KL-divergence suggests that it is location sensitive
  - Feature: KL-divergence between query and URL models
    - Low KL-divergence suggests URL is more likely to be relevant to users issuing the query
More location sensitive features

- Contextual features (user-dependent)
  - Feature: User’s location (naturally!)
  - Feature: Probability of the user’s location given the URL
    - Computed by evaluating URL’s location model at user location
    - Feature is high when user is at a location where URL is popular
    - Downside: large population centers tend to higher probabilities for all URLs
  - Feature: Use Bayes rule to compute $P(\text{URL} \mid \text{user location})$
  - Feature: Also create a normalized version of the above feature by normalizing with the background model
  - Features: Versions of the above with query instead of URL
Learning to rank

- Add location features (in addition to standard features) for machine learned ranking
  - Training data derived from logs
  - $P(\text{URL} \mid \text{user location})$ turns out to be an important feature
  - KL divergence of the URL model from the background model also plays an important role
Query model for [rta bus schedule]

User in New Orleans
URL model for top original result

User in New Orleans

(a) http://www.riderta.com/maps-schedules.asp
URL model for promoted URL

User in New Orleans

(b) http://www.norta.com/
PERSONALIZED PAGERANK
Pagerank review

- Let $A$ be the stochastic matrix corresponding to the Web graph $G$ over $n$ nodes
  - No teleportation links (but assume no deadends in $G$)
  - If node $i$ has $o_i$ outlinks, and there is an edge from node $i$ to node $j$, then $A_{ij} = 1/o_i$

- Let $p$ be the teleportation probabilities
  - $(n \times 1)$ column vector with each entry being $1/n$

- Pagerank vector $r$ is defined by the following
  $$r = (1 - \alpha)Ar + \alpha p$$
Personalized pagerank

[Haveliwala 2003] [Jeh and Widom 2003]

- In the basic pagerank computation, teleportation probability vector $p$ is uniform over all pages
- But if the user has preferences on which pages to teleport to, that preference can be represented in $p$
  - $p$ could be uniform over user’s bookmarks
  - Or it could be non-zero on just pages on topics of interest to the user
- Pagerank would be personalized to user’s interests

- But computing personalized pagerank is expensive
Linearity theorem

- For any preference vectors $u_1$ and $u_2$, if $v_1$ and $v_2$ are the corresponding personalized pagerank vectors, then for any non-negative constants $a_1$ and $a_2$ such that $a_1 + a_2 = 1$, we have

$$a_1 v_1 + a_2 v_2 = (1 - \alpha) A (a_1 v_1 + a_2 v_2) + \alpha (a_1 u_1 + a_2 u_2)$$

- Proof

$$a_1 v_1 + a_2 v_2 = a_1 ((1 - \alpha) A v_1 + \alpha u_1) + a_2 ((1 - \alpha) A v_2 + \alpha u_2)$$

$$= a_1 (1 - \alpha) A v_1 + a_1 \alpha u_1 + a_2 (1 - \alpha) A v_2 + a_2 \alpha u_2$$

$$= (1 - \alpha) A (a_1 v_1 + a_2 v_2) + \alpha (a_1 u_1 + a_2 u_2)$$
Topic-sensitive pagerank

- Compute personalized pagerank vector per topic
  - 16 top-level topics from the Open Directory Project
  - Each ODP topic has a set of pages (hand-)classified into that topic
  - Preference vector for the topic is uniform over pages in that topic, and 0 elsewhere

- Note: [Jeh and Widom 2003] provide a more general treatment
Query-time processing

- Construct a distribution over topics for the query
  - User profile can provide a distribution over topics
  - Query can be classified into the different topics
  - Any other context information can be used to inform topic distributions

- Use the topic preferences to compute a weighted linear combination of topic pagerank vectors to use in place of pagerank
SOCIAL NETWORKS
Unicorn

[Curtiss et al 2013]

- Primary backend for Facebook Graph Search

- Facebook social graph
  - Nodes represent people and things (entities)
  - Each entity has a unique 64-bit id
  - Edges represent relationships between nodes
  - There are many thousands of edge-types
    - Examples: friend, likes, likers, ...
Data model

- Billions of nodes, but graph is sparse
  - Represent graph using adjacency list
  - Postings sorted by `sort-key` (importance) and then `id`
- Index sharded by `result-id`
Basic set operations

- Query language includes basic set operations
  - and, or, difference

- Friends of either Jon Jones (id 5) and Lea Lin (id 6)
  (or(friend:5 friend:6))

- Female friends of Jon Jones who are not friend of Lea Lin
  (difference (and friend:5 gender:1) friend:6)
Typeahead

- Find users by typing first few characters of their name
- Index servers contain postings lists for every name prefix up to a predefined character limit
  - Simple typeahead implementation would simply return ids in the corresponding postings lists
- Simple solution doesn’t ensure social relevance
- Alternate solution: Use a conjunctive query
  - (and mel* friend:3)
    - Misses people who are not friends
    - Issuing two queries is expensive
WeakAnd operator

- Provides a mechanism for some fraction of results to possess a trait without requiring trait for all results
- WeakAnd allows missing terms from some results
  - These optional terms can have an optional count or weight
  - Once the optional count is met, the term is required

(weak-and (term friend:3 :optional-hits 2) (term melanie) (term mars*))
Graph Search

- Graph Search results are often more than one edge away from source nodes
  - Example: Pages liked by friends of Melanie who like Emacs

- Unicorn provides additional operators to support Graph Search
  - Apply
    - \(\text{apply likes: (and friend:7 likers:42)}\)
  - Extract
    - Extract and return (denormalized) ids stored in HitData
References

- J. Pitkow et al. Personalized search. 2002
- J. Teevan, S. Dumais, E. Horvitz. Personalizing search via automated analysis of interests and activities. 2005
- P. Bennett et al. Inferring and using location metadata to personalize Web search. 2011
- M. Curtiss et al. Unicorn: A system for searching the social graph. 2013