Chapter 10

Social Search
“Social search describes search acts that make use of social interactions with others. These interactions may be explicit or implicit, co-located or remote, synchronous or asynchronous”

Social search

- Search within a social environment
- Communities of users actively participating in the search process
- Goes beyond classical search tasks
- Facilitates the “information seeking” process [Evans 08]

Social vs. Standard Search

Key differences

- Users interact with the system (standard & Social)
- Users interact with one another in an open/social environment implicitly (reading)/explicitly (writing) such as
  - Visiting social media sites, e.g., YouTube
  - Browsing through social networking sites, e.g., Facebook
Web 2.0

- Social search includes, but is not limited to, the so-called social media site
  - Collectively referred to as “Web 2.0” as opposed to the classical notion of the Web (“Web 1.0”)

- Social media sites
  - User generated content, such as comments
  - Users can tag their own and other’s content
  - Users can share favorites, tags, etc., with others
  - Provide unique data resources for search engines

- Example.
  - YouTube, Facebook, LibraryThing, LinkedIn, Flickr, Last.FM, Twitter, CiteULike, Del.icio.us, & MySpace
## Social Media/Network Sites

<table>
<thead>
<tr>
<th>Feature</th>
<th>Facebook</th>
<th>LinkedIn</th>
<th>Delicious</th>
<th>Twitter</th>
<th>LibraryThing</th>
<th>Flickr</th>
<th>YouTube</th>
<th>MySpace</th>
<th>Last.fm</th>
<th>CiteULike</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Web Search</strong></td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td><strong>Recommendation</strong></td>
<td>friends, groups, ads, links</td>
<td>connections, jobs, ads, books, news articles, links, groups</td>
<td>books, links, news articles, movies, videos, groups</td>
<td>people, location</td>
<td>books, groups, links</td>
<td>users, pics, travel, groups, links, videos</td>
<td>video, links, people, friends, groups, ads, links (provides rec based on search habits)</td>
<td>friends, groups, ads, links, concerts (the rec application exists)</td>
<td>artciles, links, groups</td>
<td></td>
</tr>
<tr>
<td><strong>Filtering</strong></td>
<td>messages, activity, links, news articles</td>
<td>jobs, news art, links</td>
<td>links, news articles, Messages (hashtag)</td>
<td>conversations, groups</td>
<td>pics</td>
<td>videos</td>
<td>messages, activity, music</td>
<td>messages, music</td>
<td>new papers, articles</td>
<td></td>
</tr>
<tr>
<td><strong>Ads Suggestion</strong></td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>maybe</td>
<td>maybe</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td><strong>Collaborative Searching/Filtering</strong></td>
<td>yes depending on topic</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>depending on topic</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>maybe</td>
<td>depending on topic</td>
</tr>
<tr>
<td><strong>User Similarity (profile)</strong></td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>maybe</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>maybe</td>
<td>yes</td>
</tr>
<tr>
<td><strong>Personal Interest Identification</strong></td>
<td>yes depending on domain</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes (but may not be comprehensive)</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>depending on topic</td>
<td></td>
</tr>
<tr>
<td><strong>Topic Identification</strong></td>
<td>maybe</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>No (or not really comprehensive)</td>
<td>maybe</td>
<td>maybe</td>
<td>maybe</td>
<td>maybe</td>
<td>yes (within area of study)</td>
</tr>
<tr>
<td><strong>Tag (matching/suggestions)</strong></td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>maybe</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
</tbody>
</table>
## Social Media/Networking Sites

<table>
<thead>
<tr>
<th>SM Services</th>
<th>Level of Collaboration</th>
<th>Content: Audio</th>
<th>Content: Video</th>
<th>Content: Image</th>
<th>Content: Text</th>
<th>Content: Aggregation</th>
<th>Provider Censorship</th>
<th>User Censorship</th>
<th>Privacy</th>
<th>Communication Type</th>
<th>Provides API</th>
</tr>
</thead>
<tbody>
<tr>
<td>Facebook</td>
<td>HIGH</td>
<td>NONE</td>
<td>LOW</td>
<td>MEDIUM</td>
<td>HIGH</td>
<td>LOW</td>
<td>NONE</td>
<td>HIGH</td>
<td>BOTH</td>
<td>MANY-TO-MANY</td>
<td>GOOD</td>
</tr>
<tr>
<td>LinkedIn</td>
<td>LOW</td>
<td>NONE</td>
<td>NONE</td>
<td>NONE</td>
<td>HIGH</td>
<td>NONE</td>
<td>NONE</td>
<td>HIGH</td>
<td>BOTH</td>
<td>1-TO-MANY</td>
<td>NO</td>
</tr>
<tr>
<td>Twitter</td>
<td>MEDIUM</td>
<td>NONE</td>
<td>NONE</td>
<td>NONE</td>
<td>HIGH</td>
<td>HIGH</td>
<td>NONE</td>
<td>HIGH</td>
<td>BOTH</td>
<td>MANY-TO-MANY</td>
<td>GOOD</td>
</tr>
<tr>
<td>Delicious</td>
<td>LOW</td>
<td>NONE</td>
<td>NONE</td>
<td>NONE</td>
<td>LOW</td>
<td>HIGH</td>
<td>NONE</td>
<td>NONE</td>
<td>PUBLIC</td>
<td>MANY-TO-MANY</td>
<td>GOOD</td>
</tr>
<tr>
<td>Flickr</td>
<td>MEDIUM</td>
<td>NONE</td>
<td>HIGH</td>
<td>HIGH</td>
<td>LOW</td>
<td>NONE</td>
<td>NONE</td>
<td>NONE</td>
<td>HIGH</td>
<td>BOTH</td>
<td>MANY-TO-MANY</td>
</tr>
<tr>
<td>YouTube</td>
<td>HIGH</td>
<td>NONE</td>
<td>HIGH</td>
<td>NONE</td>
<td>LOW</td>
<td>NONE</td>
<td>LOW</td>
<td>HIGH</td>
<td>BOTH</td>
<td>MANY-TO-MANY</td>
<td>GOOD</td>
</tr>
<tr>
<td>Skype</td>
<td>LOW</td>
<td>HIGH</td>
<td>HIGH</td>
<td>NONE</td>
<td>LOW</td>
<td>NONE</td>
<td>NONE</td>
<td>NONE</td>
<td>HIGH</td>
<td>PRIVATE</td>
<td>1-TO-1</td>
</tr>
<tr>
<td>Last.fm</td>
<td>LOW</td>
<td>HIGH</td>
<td>NONE</td>
<td>NONE</td>
<td>LOW</td>
<td>NONE</td>
<td>LOW</td>
<td>LOW</td>
<td>BOTH</td>
<td>MANY-TO-MANY</td>
<td>GOOD</td>
</tr>
<tr>
<td>yelp.com</td>
<td>LOW</td>
<td>NONE</td>
<td>NONE</td>
<td>HIGH</td>
<td>HIGH</td>
<td>HIGH</td>
<td>NONE</td>
<td>NONE</td>
<td>NONE</td>
<td>1-TO-MANY</td>
<td>OK</td>
</tr>
<tr>
<td>WikiAnswers</td>
<td>HIGH</td>
<td>NONE</td>
<td>NONE</td>
<td>LOW</td>
<td>HIGH</td>
<td>NONE</td>
<td>NONE</td>
<td>NONE</td>
<td>HIGH</td>
<td>PUBLIC</td>
<td>MANY-TO-MANY</td>
</tr>
<tr>
<td>World of Warcraft</td>
<td>MEDIUM</td>
<td>MEDIUM</td>
<td>HIGH</td>
<td>NONE</td>
<td>LOW</td>
<td>NONE</td>
<td>NONE</td>
<td>NONE</td>
<td>LOW</td>
<td>BOTH</td>
<td>MANY-TO-MANY</td>
</tr>
</tbody>
</table>
Social Search Topics

- Online *user-interactive* data, which provide a new and interesting search experience
  - **User tags**: users assign tags to data items, a *manual indexing* approach
  - **Searching within communities**: *virtual* groups of online users, who share *common interests*, *interact socially*, such as blogs and QA systems
  - **Recommender systems**: individual users are represented by their *profiles* (fixed queries – long-term info. need), e.g., YouTube, Amazon.com, CNN Alert Service, etc.
  - **Peer-to-peer network**: querying a community of “nodes” (individual/organization/search engine) for an info. need, e.g., Metasearch
Recommender Systems (RSs)

- RSs are software tools providing suggestions for items to be of use to users, such as what items to buy, what music to listen to, or what online news to read.

- Primarily designed to evaluate the potentially overwhelming number of alternative items may offer.
  - The explosive growth & variety of information on the Web frequently lead users to make poor decisions.
  - Offer ranked lists of items by predicting the most suitable products or services based on the users’ preferences & constraints.
  - Often rely on recommendations provided by others in making routine, daily decisions, the collaborative-filtering technique.
  - Use various types of knowledge & data about users/items.
Recommender Systems

- **Content-based Recommender Systems:**
  - Try to recommend new items *similar* to those a given user has *liked* in the past
    - Identify the *common characteristics* of items being liked by user *u* and recommend to *u* new items that share these characteristics
    - An item *i*, which is a text document, can be represented as a feature vector *x_i* that contains the TF-IDF weights of the most informative keywords
    - A profile of *u*, denoted profile vector *x_u*, can be obtained from the contents of items *rated* by *u*, denoted *r_u*, and each item *i* rated by *u*, denoted *r_ui*
      \[
      x_u = \sum_{i \in r_u} r_{ui} x_i
      \]
      which adds the weights of *x_i* to *x_u* a scalar value
Recommender Systems

- Content-based Recommender Systems:
  - Approach: analyze the descriptions of items previously rated by a user & build a user profile (to present user interests/preferences) based on the features of the items.
Content-Based Filtering

- Advantages:
  - **User Independence**: explores *solely ratings* provided by the user to build her own profile, but not other users’ (as in collaborative filtering)
  - **Transparency**: recommendations can be explained by explicitly listing *content features* that caused an item to be recommended
  - **New Items**: items *not yet rated* by any user can still be recommended, unlike collaborative recommenders which rely solely on other users’ rankings to make recommendations
Content-Based Filtering

- Shortcomings:

  - Limited Content Analysis: there is a natural limit in the number/types of features that can be associated with items which require domain knowledge (e.g., movies)

  - Over-specialization: tendency to produce recommendations with a limited degree of novelty, i.e., the serendipity problem, which restricts its usefulness in applications

  - New User: when few ratings are available (as for a new user), CBF cannot provide reliable recommendations
Recommender Systems

- **Collaborative Filtering**

  Recommender Systems:

  - Unlike content-based filtering approaches which use the content of items previously rated by users.
  - **Collaborative filtering (CF)** approaches rely on the ratings of a user, and those of *other* users in the system.
  - Intuitively, the rating of a user *u* for a new item *i* is likely *similar* to that of user *v* if *u* and *v* have rated other items in a *similar* way.
  - Likewise, *u* is likely to rate two items *i* and *j* in a similar fashion, if other users have given similar ratings to *i* & *j*.
  - CF overcomes the *missing content* problem of the content-based filtering approach through the feedback, i.e., ratings, of other users.
Collaborative Filtering

Collaborative Filtering Recommender Systems:

- Instead of relying on content, which may be a bad indicator, CF are based on the quality of items evaluated by peers.

- Unlike content-based systems, CF can recommend items with very different content, as long as other users have already shown interested for these different items.

- Goal: identify users whose preferences are similar to those a given user has liked in the past.

- Two general classes of CF methods:
  - Neighborhood-based methods
  - Model-based methods
Collaborative Filtering

- Neighborhood-based (or heuristic-based) Filtering:
  - User-item ratings stored in the system are directly used to predict ratings for new items, i.e., using either the user-based or item-based recommendation approach
    - **User-based**: evaluates the interest of a user $u$ for an item $i$ using the ratings for $i$ by other users, called neighbors, that have similar rating patterns
      
      The neighbors of $u$ are typically users $v$ whose ratings on the items rated by both $u$ and $v$ are most correlated to those of $u$
    
    - **Item-based**: predicts the rating of $u$ for an item $i$ based on the ratings of $u$ for items similar to $i$
Neighborhood-Based Recommendation

- **Example.**

<table>
<thead>
<tr>
<th></th>
<th>The Matrix</th>
<th>Titanic</th>
<th>Die Hard</th>
<th>Forrest Gump</th>
<th>Wall-E</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>5</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Lucy</td>
<td>2</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>Eric</td>
<td>4</td>
<td>?</td>
<td>3</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Diane</td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

A “toy example” showing the ratings of four users for five movies.

- Eric & Lucy have very similar tastes when it comes to movies, whereas Eric and Diane have different tastes.
- Eric likely asks Lucy the opinion on the movie “Titanic” and discards the opinion of Diane.
Collaborative Filtering

- User-based Rating Prediction:
  - Predicts the rating $r_{ui}$ of a user $u$ for a new item $i$ using the ratings given to $i$ by users most similar to $u$, called nearest-neighbors
  - Given the $k$-nearest-neighbor of $u$ who have rated item $i$, denoted $N_i(u)$, the rating of $r_{ui}$ can be estimated as
    $$r_{ui} = \frac{1}{|N_i(u)|} \sum_{v \in N_i(u)} r_{vi}$$

- If the neighbors of $u$ can have different levels of similarity with respect to $u$, denoted $w_{uv}$, the predicted rating is
  $$r_{ui} = \frac{\sum_{v \in N_i(u)} w_{uv} r_{vi}}{\sum_{v \in N_i(u)} |w_{uv}|}$$
Collaborative Filtering

- **Item-based Rating Prediction:**
  - While *user-based* methods rely on the opinion of like-minded users, i.e., similar users, to predict a rating, item-based approaches look at ratings given to similar items.
  - **Example.** Instead of consulting with his peers, Eric considers the ratings on the movies he (& others) has (have) seen.

- Let $N_u(i)$ be the set of items rated by user $u$ most similar to item $i$, the predicted rating of $u$ for $i$ is
  
  $$r_{ui} = \frac{\sum_{j \in N_u(i)} w_{ij} r_{uj}}{\sum_{j \in N_u(i)} |w_{ij}|}$$
Collaborative Filtering

- Advantages of Neighborhood-based Filtering:
  - **Simplicity**: the methods are intuitive & relatively simple to implement (w/ only the no. of neighbors requires tuning)
  - **Justifiability**: the methods provide a concise & intuitive justification for the computed predictions
  - **Efficiency**: the methods require no costly training phases & storing nearest neighbors of a user requires very little memory. Thus, it is scalable to millions of users & items
  - **Stability**: the methods are not significantly affected by the constant addition of users, items, and ratings in a large commercial applications & do not require retraining
Community Based Question Answering

- Some complex information needs can’t be answered by traditional search engines
  - No single webpage may exist that satisfies the information needs
  - Information may come from multiple sources
  - Human (non-)experts in a wide range of topics form a community-based question answering (CQA) group, e.g., Yahoo! Answers

- CQA tries to overcome these limitations
  - Searcher enters questions
  - Community members answer questions
Example Questions

What part of Mexico gets the most tropical storms?
How do you pronounce the french words, coeur and miel?
GED test?
Why do I have to pay this fine?
What is Schrödinger’s cat?
What’s this song?
Hi...can u ppl tell me sumthing abt death dreams??
What are the engagement and wedding traditions in Egypt?
Fun things to do in LA?
What lessons from the Tao Te Ching do you apply to your everyday life?
Foci of a hyperbola?
What should I do today?
Why was iTunes deleted from my computer?
Heather Locklear?
Do people in the Australian Defense Force (RAAF) pay less tax than civilians?
Whats a psp xmb?
If C(-3, y) and D(1, 7) lie upon a line whose slope is 2, find the value of y.?
Why does love make us so irrational?
Am I in love?
What are some technologies that are revolutionizing business?
Question Answering

- **Goal**
  - Automatically answer questions submitted by humans in a natural language form

- **Approaches**
  - Rely on techniques from diverse areas of study, e.g., IR, NLP, and Ontology to identify users’ info. needs & textual phrases potentially suitable answers for users

- **Exploit**
  - (Web) Data Sources, i.e., doc corpus

Data from Community Question Answering Systems (CQA)
Question Answering (QA)

- Question answering (QA) is a specialized form of IR

- Given a collection of documents/collaborative QA system, the QA system attempts to retrieve correct answers to questions posted in natural language

- Unlike search engines, QA systems generate \textit{answers} instead of providing ranked lists of documents

- Current (non-collaborative) QA systems \textit{extract} answers from large corpora such as the Web

- \textit{Fact-based} QA limits range of \textit{informational questions} to those with simple, short answers
  - \textit{who}, \textit{where}, \textit{why}, \textit{what}, \textit{when}, \textit{how} (5W 1H/WH) questions
Question Answering

- CQA-based approaches
  - Analyze questions (& corresponding answers) archived at CQA sites to locate answers to a newly-created question
  - Exploit “wealth-of-knowledge” already provided by CQA users

- Existing popular CQA sites
  - Yahoo! Answers, StackOverflow, and WikiAnswers
Question Answering

- Example.

CQA-Based
Community Based Question Answering

Community Based Question Answering

- **Pros**
  - Users can find answers to *complex* or *obscure* questions with *diverse* opinions about a topic
  - Answers are from *humans*, not algorithms, that can be interacted with who share common interests/problems
  - Can search *archive* of previous questions/answers, e.g., Yahoo! Answers

- **Cons**
  - Some questions *never* get *answered*
  - Often takes *time* (possibly days) to get a response
  - Answers may be *wrong, spam, or misleading*
Question Answering

- Challenges for finding an answer to a new question from QA pairs archived at CQA sites

- Misleading Answers
- No Answers
- SPAM
- Incorrect Answers
- Spam Answers
- Answerer reputation

CQA-Based
Question Answering Models

- How can we effectively *search* an *archive* of question/answer pairs databases?

- Can be treated as a *translation* problem
  - Translate a question into a *related/similar* question which likely have relevant answers
  - Translate a question into an *answer*: less desirable

- The *vocabulary mismatch* problem
  - Traditional IR models likely miss many relevant questions
  - Many different ways to ask the same question
  - Stopword removal and stemming do not help
  - Solution: consider related concepts (i.e., words)–the *probability* of replacing one word by another
Question Answering

- Challenges (cont.) 300 millions posted under Yahoo!
  Answers since 2005: an average of 7,000 questions & 21,000 answers per hour

Account for the fact that questions referring to the same topic might be formulated using similar, but not the same, words

Identifying the most suitable answer among the many available
Question Answering Models

- Translation-based language model (for finding related questions, then answers): translate \( w \) (in \( Q \)) from \( t \) (in \( A \))

\[
P(Q|A) = \prod_{w \in Q} \sum_{t \in V} P(w|t)P(t|A)
\]

where \( Q \) is a question, 
\( A \) is a related question in the archive, 
\( V \) is the vocabulary

- Anticipated problem: a good (independent) term-to-term translation might not yield a good overall translation

- Potential solution: *matches* of the original question terms are given *more weight* than matches of *translated* terms
Question Answering Models

- **Enhanced** translation model, which extends the *translation-based language model* on ranking $Q$:

$$P(Q|A) = \prod_{w\in Q} \frac{(1 - \beta)f_{w,A} + \beta \sum_{t\in V} P(w|t)f_{t,A} + \mu \frac{c_w}{|C|}}{|A| + \mu}$$

where $\beta \in 0 .. 1$ controls the influence of the translation probability

- $\mu$ is a *smoothing* parameter
- $|A|$ is the number of words in question $A$
- $C_w$ is *count* of $w$ in the entire collection $C$, and
- $|C|$ is the total number of *word occurrence* in $C$

- when $\beta \rightarrow 1$, the model becomes more similar to the translation-based language model
- when $\beta \rightarrow 0$, the model is equivalent to the original query likelihood model, without influence from the translation model
Computing Translation Probabilities

- Translation probabilities are learned from a parallel corpus.
- Most often used for learning inter-language probabilities.
- Can be used for intra-language probabilities:
  - Treat question-answer pairs as parallel corpus.
  - Translation probabilities are estimated from archived pairs \((Q_1, A_1), (Q_2, A_2), \ldots, (Q_N, A_N)\).

Drawbacks:
- Computationally expensive: sum over the entire vocabulary, which can be very large.
- Solution: considering only a small number (e.g., 5) of (most likely) translations per question term.
## Sample Question/Answer Translations

<table>
<thead>
<tr>
<th>everest</th>
<th>xp</th>
<th>search</th>
</tr>
</thead>
<tbody>
<tr>
<td>everest</td>
<td>xp</td>
<td>search</td>
</tr>
<tr>
<td>mountain</td>
<td>window</td>
<td>google</td>
</tr>
<tr>
<td>tallest</td>
<td>install</td>
<td>information</td>
</tr>
<tr>
<td>29,035</td>
<td>drive</td>
<td>internet</td>
</tr>
<tr>
<td>highest</td>
<td>computer</td>
<td>website</td>
</tr>
<tr>
<td>mt</td>
<td>version</td>
<td>web</td>
</tr>
<tr>
<td>ft</td>
<td>click</td>
<td>list</td>
</tr>
<tr>
<td>measure</td>
<td>pc</td>
<td>free</td>
</tr>
<tr>
<td>feet</td>
<td>program</td>
<td>info</td>
</tr>
<tr>
<td>mount</td>
<td>microsoft</td>
<td>page</td>
</tr>
</tbody>
</table>
User Tags and Manual Indexing

- Then: Library card catalogs
  - *Indexing terms* chosen with search in mind
  - *Experts generate* indexing terms *manually*
  - Terms are very *high quality* based on the US Library of Congress (LOC) *Subject Headings* standardized by the LOC
  - Terms chosen from *controlled/fixed vocabulary* and subject guides (a drawback)

- Now: Social media tagging
  - *Social media sites* allow *users* to *generate* own tags *manually* (+)
  - Tags not always chosen with search in mind (-)
  - Tags can be *noisy* or even *incorrect* and without quality control (-)
  - Tags chosen from *folksonomies*, user-generated taxonomies (+)
Social Search Topics

Example. Some of the 116 million tags of LibraryThing, which archives 95 million book records with 1.93 million users (04/15)

Lou Gehrig: the luckiest man
by David A. Adler

Members | Reviews | Popularity | Average rating | Conversations
---|---|---|---|---
21 | 3 | 163,594 | 🌟🌟🌟🌟 (4) | None

Recently added by: Ip1168925, Kaylinn_Hall, michelehat, RamathMES, sarglib, mentormom, farahpo (see more)

Your library

Add to your library

Member tags

2-Background Knowledge(1) 4.6(1) 2008(1) 7-Keys(1) amyotrophic lateral sclerosis(1) or(1) baseball(3) baseball - history(1) baseball players(1) Biography(1) children's books(1) dada(1) Gehrig - Lou(1) gehrig lou 1903-1941(1) heroes(1) history(1) illness(1) Lou Gehrig(1) Lou Gehrig(1) Lou Gehrig(1) Non-fiction(2) picture book(2) red sox(1) 1999(1) social studies baseball(1) sports(1) Unread(1) Yankees(1)

LibraryThing recommendations

1. Teammates by Peter Golenbock
2. Luckiest man: the life and death of Lou Gehrig by Jonathan Eig
3. Iron horse : Lou Gehrig in his time by Ray Robinson
4. Mighty Jackie by Marissa Moss
5. Rosalie by Joan Hewett

Member recommendations:
No member recommendations (contribute a recommendation)

(see more recommendations and anti-recommendations for this book)
Social Tagging

According to [Guan 10]

- **Social tagging services** allow users to *annotate* online resources with freely chosen keywords.
- Tags are collectively *contributed* by users and represent their comprehension of resources.
- Tags provide meaningful *descriptors* of resources and implicitly reflect users’ interests.
- Tagging services provide *keyword-based search*, which returns resources annotated by given tags.

Types of User Tags

- Content-based
  - Tags describe the *content* of an item, e.g., car, woman, sky

- Context-based
  - Tags describe the *context* of an item, e.g., NYC, empire bldg

- Attribute-based
  - Tags describe the *attributes* of an item, e.g., Nikon (type of camera), black and white (type of movie), etc.

- Subjective-based
  - Tags *subjectively* describe an item, e.g., pretty, amazing, etc.

- Organizational-based
  - Tags that organize items, e.g., to do, not read, my pictures, …
Searching Tags

- Searching collaboratively tagged items, i.e., *user tags*, is *challenging*
  - Most items have only a few tags, i.e., complex items are sparingly represented, e.g., “aquariums” ≠ “goldfish”, which is the *vocabulary mismatch problem*
  - Tags are very *short*

- *Boolean (AND/OR), probabilistic, vector space, and language modeling* will fail if use naively
  - *High* precision but *low* recall for conjunctive (AND) queries
  - *Low* precision but *high* recall for disjunctive (OR) queries
Tag Expansion

- Can overcome *vocabulary mismatch* problem, such as “aquariums” and “topical fish”, by *expanding* tag representation with *external knowledge*

- Possible external sources
  - Thesaurus
  - Web search results
  - Query logs

- After *tags* have been *expanded*, can use standard retrieval models
Example. Web search results enhance a tag representation, “tropical fish,” a query.

A retrieved snippet

Pseudo-relevance feedback over related terms

\[ P(w | \text{“tropical fish”}) \]
Searching Tags

- Even with tag expansion, searching tags is challenging.
- Tags are inherently noisy (off topic, inappropriate) and incorrect (misspelled, spam).
- Many items may not even be tagged, which become virtually invisible to any search engine.
- Typically easier to find popular items with many tags than less popular items with few/no tags.

- How can we automatically tag items with few or no tags?
  - Uses inferred tags to:
    - Improve tag search
    - Automatic tag suggestion
Methods for Inferring Tags

- **TF-IDF**: \( wt(w) = \log(f_{w,D} + 1) \log(N / df_w) \)
  - Suggest tags that have a *high* TF-IDF weight in the item
  - Only works for textual items

- **Classification** (determines the appropriateness of a tag)
  - Train binary classifier for each tag, e.g., using SVM
  - Performs well for *popular tags*, but not as well for *rare tags*

- **Maximal marginal relevance**
  - Finds *relevant* tags to the item and *novel* with respect to others
  - \( MMR(t; T_i) = \left( \lambda Sim_{item}(t, i) - (1 - \lambda) \max_{t \in T_i} Sim_{tag}(t_i, t) \right) \)
    - Using TF-IDF
    - Using query results
  - Large, if \( t \) is very relevant to \( T_i \), but differs from other tags of \( T_i \)

where \( Sim_{item}(t, i) \) is the *similarity* between tag \( t \) and item \( i \), i.e., \( T_i \)
\( Sim_{tag}(t_i, t) \) is the *similarity* between tags \( t_i \) and \( t \)
\( \lambda (\in 0..1) \), a tunable parameter
Browsing and Tag Clouds

- Search is useful for finding items of interest
- **Browsing** is more useful for exploring collections of tagged items

Various ways to visualize collections of tags

- **Tag clouds** (show the popularity of tags based on sizes)
- (Tags are) Alphabetically *order* and/or *weighted*
- Formatted/sorted according to *popularity*
Sample Tag Cloud

animals architecture art australia autumn baby band barcelona beach berlin birthday black blackandwhite blue california cameraphone canada canon car cat chicago china christmas church city clouds color concert day dog england europe family festival film florida flower flowers food france friends fun garden germany girl graffiti green halloween hawaii holiday home house india ireland italy japan july kids lake landscape light live london macro me mexico music nature new newyork night nikon nyc ocean paris park party people portrait red river rock sanfrancisco scotland sea seattle show sky snow spain spring street summer sunset taiwan texas thailand tokyo toronto travel tree trees trip uk usa vacation washington water wedding
Tag Clouds

- As defined in [Schrammel 09], tag clouds are
  - Visual displays of set of words (tags) in which attributes of the text such as size, color, font weight, or intensity are used to represent relevant properties, e.g., frequency of documents linked to the tag
  - A good visualization technique to communicate an “overall picture”

What is an online community?

- Groups of entities (i.e., users, organizations, websites) that *interact* in an online environment to share common goals, interests, or traits
- Besides tagging, community users also *post* to newsgroups, blogs, and other forums
- To improve the overall user experiments, web search engines should automatically find the *communities* of a user

**Example.**

- Baseball fan community, digital photography community, etc.

**Not all communities are made up of humans!**

- Web communities are collections of web pages that are all about a *common topic*
Online Communities

According to [Seo 09]

- Online communities are valuable information sources where knowledge is accumulated by interactions between people.

- Online community pages have many unique textual or structural features, e.g.,
  - A forum has several sub-forums covering high-level topic categories.
  - Each sub-forum has many threads.
  - A thread is a more focused topic-centric discussion unit and is composed of posts created by community members.

Finding Communities

- How to design general-purpose algorithms for finding every possible type of on-line community?

- What are the criteria used for finding a community?
  - Entities (users) within a community are similar to each other
  - Members of a community are likely to interact more with one another of the community than those outside of the community

- Can represent interactions between a set of entities as a graph
  - Vertices \((V)\) are entities
  - Edges \((E)\), directed or undirected, denote interactions of entities
    - Undirected edges represent symmetric relationships
    - Directed edges represent non-symmetric or causal relationships
HITS

- Hyperlink-Induced Topic Search (HITS) algorithm can be used to find communities
  - A *link analysis* algorithm, like PageRank
  - Each entity has a *hub* and *authority* score

- Based on a *circular* set of assumptions
  - Good *hubs* point to good *authorities*
  - Good *authorities* are pointed to by good *hubs*

- Iterative algorithm:

  \[
  A(p) = \sum_{q \rightarrow p} H(q) \\
  H(p) = \sum_{p \rightarrow q} A(q)
  \]

  Authority score of \( p \) is the sum of the hub scores of the entities pointing at \( p \)

  Hub score of \( p \) is the sum of the authority scores pointed at by \( p \)
Form community \((C)\)

- Apply the *entity interaction graph* to find communities
- Identify a subset of the entities \((V)\), called *candidate entities*, be members of \(C\) (based on common interest)
- Entities with large *authority scores* are the *core* or “authoritative” members of \(C\)
  - to be a *strong authority*, an entity must have many *incoming* edges, all with moderate/large *hub* scores, and
  - To be a *strong hub*, an entity must have many outgoing edges, all with moderate/large *authority* scores
- Vertices *not* connected with others have *hub* and *authority* scores of 0
Finding Communities

- Clustering
  - Community finding is an inherently unsupervised learning problem
  - Agglomerative or K-means clustering approaches can be applied to entity interaction graph to find communities
  - Use the vector representation to capture the connectivity of various entities
  - Compute the authority values based on the Euclidean distance

- Evaluating community finding algorithms is hard

- Can use communities in various ways to improve web search, browsing, expert finding, recommendation, etc.
Graph Representation

Node: 1 2 3 4 5 6 7
Vector:

\[
\begin{bmatrix}
0 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\]
Document Filtering

- Ad hoc retrieval
  - Document collections are static and information needs change with time
  - Results returned when query is entered

- Document filtering
  - Document collections change with time, but (long-term) information needs are static
  - Long term information needs represented as a profile
  - Documents entering system that match the profile are delivered to the user via a push mechanism
  - Must be efficient and effective (minimizes FPs and FNs)
Profiles

- Represents long-term *information needs* and personalizes the search experience

- Can be represented in different ways by including
  - A Boolean or keyword query
  - Sets of *relevant* and *non-relevant* documents
  - Social tags and named entities
  - Relational *constraints*
    - “Published before 1990”
    - “Price in the $10 - $25 range”

- Actual representation usually depends on the underlying *filtering model*

- Static (filtering) or updated over time (adaptive filtering)
Document Filtering Scenarios

Profile 1
Profile 2
Profile 3

Profile 1
Profile 2
Profile 3
Profile 1.1
Profile 2.1
Profile 3.1

$t = 2$  $t = 3$  $t = 5$  $t = 8$

Document Stream

$t = 2$  $t = 3$  $t = 5$  $t = 8$

Document Stream

Static Filtering
Easier to process, less robust

Adaptive Filtering
More robust, requires frequent updates
Static Filtering

- Given a fixed profile, how can we determine if an incoming document should be delivered?

- Treat as an IR problem
  - Boolean
  - Vector space
  - Language modeling

- Treat as supervised learning problem
  - Naïve Bayes
  - Support vector machines

Require predefined threshold value
Static Filtering with Language Models

- Assume a profile $P$ consists of $K$ relevant documents $T_i$ ($1 \leq i \leq K$) each with weight $\alpha_i$

- Probability of a word $w$ given the profile $P$ is

$$P(w|P) = \frac{(1 - \lambda)}{\sum_{i=1}^{K} \alpha_i} \sum_{i=1}^{K} \alpha_i \frac{f_{w,T_i}}{|T_i|} + \lambda \frac{c_w}{|C|}$$

- $\alpha_i$ is the **weight** (important) associated with $T_i$
- $f_{w,T_i}$ is the **frequency of occurrence** of $w$ in $T_i$
- $\lambda$ is a **smoothing** parameter
- $C_w$ is **count** of $w$ in the entire collection $C$, and
- $|C|$ is the total number of **word occurrence** in $C$
Static Filtering with Language Models

- Probability of a word \( w \) given a new document \( D \) is
  \[
P(w \mid D) = (1 - \lambda) \frac{f_{w,D}}{|D|} + \lambda \frac{C_w}{|C|}
  \]

- **KL-divergence** between profile and document model is used as score

  ![Diagram](Image)

  \[
  KL(P \parallel D) = \sum_{t \in V} P(t \mid P) \log \frac{P(t \mid P)}{P(t \mid D)}
  \]

  Not depended on \( D \): ignore for ranking

  \[
  -KL(P \parallel D) = \sum_{w \in V} P(w \mid P) \log P(w \mid D) - \sum_{w \in V} P(w \mid P) \log P(w \mid P)
  \]

- If \(-KL(P \parallel D) \geq \theta\), then deliver \( D \) to \( P \), where \( \theta \) is some relevance threshold
Adaptive Filtering

- In adaptive filtering, *profiles* are **dynamic**
- How can *profiles* change (from **static** to **dynamic**)?
  1. User can explicitly *update* the profile
  2. User can provide (relevance) *feedback* about the documents delivered to the profile
  3. Implicit user *behavior* can be *captured* and used to update the profile
Adaptive Filtering Models

- Rocchio
  - Profiles treated as vectors

\[ P' = \alpha P + \beta \frac{1}{|Rel|} \sum_{D_i \in Rel} D_i - \gamma \frac{1}{|Nonrel|} \sum_{D_i \in Nonrel} D_i \]

- Relevance-based (profiles) language models

\[ P(w|P) = \frac{1}{|Rel|} \sum_{D_i \in Rel} \sum_{D \in C} P(w|D)P(D_i|D) \approx \frac{1}{|Rel|} \sum_{D_i \in Rel} P(w|D_i) \]

where \(C\) is the set of documents in the collection \(P(D_i | D)\), the probability that \(D_i\) is generated from \(D\)'s LM, which is very close to 1 if \(D_i = D\); 0, otherwise
Fast Filtering with Millions of Profiles

- Real filtering systems
  - May have thousands or even millions of profiles
  - Many new documents will enter the system daily

- How to efficiently filter in such a system?
  - Most profiles are represented as text or a set of features
  - Build an inverted index for the profiles
  - Distill incoming documents as “queries” and run against index
Evaluation of Filtering Systems

- Definition of “good” depends on the purpose of the underlying filtering system
  - Do not produce ranking of documents for each profile
  - Evaluation measures, such as Precision@n and MAP, are non-relevant; precision, recall, and F-measure are computable

<table>
<thead>
<tr>
<th></th>
<th>Relevant</th>
<th>Non- Relevant</th>
</tr>
</thead>
<tbody>
<tr>
<td>Retrieved</td>
<td>TP</td>
<td>FP</td>
</tr>
<tr>
<td>Not retrieved</td>
<td>FN</td>
<td>TN</td>
</tr>
</tbody>
</table>

- Generic filtering evaluation measure:

\[ U = \alpha \cdot TP + \beta \cdot TN + \delta \cdot FP + \gamma \cdot FN \]

\( \alpha = 2, \beta = 0, \delta = -1, \) and \( \gamma = 0 \) are widely used
## Summary of Filtering Models

<table>
<thead>
<tr>
<th>Model</th>
<th>Profile Representation</th>
<th>Profile Updating</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean Vector Space</td>
<td>Boolean Expression</td>
<td>N/A</td>
</tr>
<tr>
<td>Language Modeling Classification</td>
<td>Probability Distribution</td>
<td>Rocchio Relevance Modeling</td>
</tr>
<tr>
<td></td>
<td>Vector</td>
<td>Online Learning</td>
</tr>
</tbody>
</table>