Automatically Building Research Reading Lists

Michael D. Ekstrand1 Praveen Kanaan1 James A. Stemper2
John T. Butler2 Joseph A. Konstan1 John T. Riedl1

ekstrand@cs.umn.edu

1GroupLens Research
Department of Computer Science and Engineering
University of Minnesota

2University of Minnesota Libraries

RecSys 2010

Recommending Research Papers

Way too many papers to read — need to find the right ones so we can get on to research!

Many papers, many reviewers — who is most qualified to judge the work?

Working on a bibliography or reference list — who should have cited you?

Building a Reading List

Alice is a new grad student who took a class on adaptive web technologies. The professor assigned some readings, and she wants to learn more.


How can we find important ones to read?

The Reading List Task

Input
List of papers on topic of interest

Magic happens here

Output
List of important papers to read
Domain Characteristics

Traditional CF methods are blind to item content or relationships.

The citation web has a defined structure.

Can we harness this to improve recommendation?

Questions

- Can we harness domain structure to improve recommendation of research papers?
- What algorithms perform well at building reading lists?
- How can we measure this performance?

Approach

Analyze Task and Domain → Design Candidate Algorithms → Prune Algorithm Pool (offline) → Evaluate Performance (user study)

Analyze Task and Domain → Design Candidate Algorithms → Prune Algorithm Pool (offline) → Evaluate Performance (user study)
Designing Candidate Algorithms

Analyze Task and Domain → Design Candidate Algorithms → Prune Algorithm Pool (offline) → Evaluate Performance (user study)

Algorithms

We adapt and combine off-the-shelf algorithms.

Collaborative filtering (CF)
- Item-item CF over sets of items [Karypis, 2001]

Content-based filtering (CBF)
- Lemur toolkit [Ogilvie and Callan, 2002] in BM25 mode with recommended baseline parameters

Graph ranking
- PageRank [Page et al., 1999]
- HITS authority scores [Kleinberg, 1999]
- SALSA [Lempel and Moran, 2000]
- Relative algorithms [White and Smyth, 2003]
  ▶ Biased HITS
  ▶ k-step Markov importance

Applying Item-Item CF

No users - papers are both users and items [McNee et al., 2002].

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>B</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Becomes a form of co-citation analysis.

Rank-Weighting CF

Inspired by [Karypis, 2001]: user influence in item similarity does not need to be the same!

Karypis’s approach: normalize user purchase vectors to unit vectors so users with few purchases influence item similarity more.

Our adaptation: weight citation vectors by graph rank. Highly-ranked papers have more influence.

\[ \hat{a} = r(a)a \]

Use weighted citation vectors to compute cited paper similarity.
Using Weights with CBF

Two approaches

Subgraph ranking
Build subgraph from CBF results and rank its nodes.

Linear blending
Blend CBF scores with node weights in the global graph.

Learned coefficients with multivariate logistic regression (more on this later).

Hybridizing Recommenders [Burke, 2002]

Several ways of blending algorithms
- Use CF as input to CBF
- Use CBF as input to CF
- Blend output from CBF and CF to produce result

Basic algorithms + all hybrids = 177 algorithms.

Pruning the Algorithm Pool

Analyze Task and Domain
Design Candidate Algorithms
Prune Algorithm Pool (offline)
Evaluate Performance (user study)
Designing an Evaluation Strategy

Goal: Measure algorithm performance at supporting a specific task.

Accuracy isn’t enough [McNee et al., 2006].

Offline Evaluation

Use metadata from ACM Digital Library.

Simulate introductory reading list with hold-out test on articles in ACM Computing Surveys.

- Hold out 5 items from each citation list
- Attempt to recommend back the 5 items
- Skip surveys with less than 15 resolved citations

Result: 220 survey articles for training and testing.

Interlude - Learning CBF Blending Coefficients

Use half the articles to train the CBF blend.

Learned multivariate logistic regression with response of 1 for articles in the holdout set, 0 for other articles.

\[ s(a) = \alpha L(a) + \beta r(a) \]

Since we’re ranking, intercept and log don’t matter.

Offline Evaluation (cont)

Measure using half-life utility metric [Breese et al., 1998]:

\[ R_a = \sum_i \frac{u_{a,i}}{2^{(i-1)/(\alpha-1)}} \]

- \( \alpha = 5 \), the length of our reading lists
- \( u_{a,i} = 1 \) if article \( i \) is cited by survey \( a \), 0 otherwise

Aggregate to compute fraction of potential utility achieved (in range \([0, 1]\)).

\[ R = \frac{1}{nR_{\max}} \sum_a R_a \]
Offline Results — CF

- Half-life utility
  - 0.05
  - 0.10
  - 0.15
  - 0.20
  - 0.25
  - 0.30


Offline Results — CBF

- Half-life utility
  - 0.02
  - 0.04
  - 0.06
  - 0.08
  - 0.10


Selected Algorithms

- Chose 3 algorithms that performed well but have differing structures.
  - CBF with Biased HITS in subgraph ranking configuration
  - CBF fed into PageRank-weighted CF
  - PageRank-weighted CF

Salsa performs like PageRank
Evaluating Performance

- Analyze Task and Domain
- Design Candidate Algorithms
- Prune Algorithm Pool (offline)
- Evaluate Performance (user study)

User-Based Evaluation

Asked graduate students to provide query sets of 5-10 papers and evaluate 3 5-item reading lists.

- Relevance of individual papers
- Importance of individual papers
- Quality of reading list as a whole
- Relative ranking of reading lists

All questions were set in the context of introducing a new researcher to the topic.
User-Based Results

---

### Average Score

- **Relevance**: 1
- **Importance**: 2
- **Familiarity**: 3

**Rank score**

- **1.5**
- **2.0**
- **2.5**

**cbl**

**cbl-cf**

**cf**

**cbl**

**cbl-cf**

**cf**

CF performed the best (surprisingly)

---


**Guy et al.** Personalized recommendation of social software items based on social relations. In *Proc. RecSys 2009*.


How do we do?

---

**Input Query (again)**


Design recommenders to support human information needs, not just improve prediction error [McNee et al., 2006].

Task and context inform design of both recommenders and evaluation strategies.

Opportunity to harness unique characteristics (or “personalities”) of specific algorithms.

Conclusion

Contributions

- Design and evaluation of recommenders for reading lists
- Method for biasing CF with authority metrics
- CF works well for reading lists (surprising)

Open Questions

- How to tell user that they have an important paper?
- How to more accurately operate within topic scope?
- Is SALSA misused?

Questions?

ekstrand@cs.umn.edu

Funded by NSF grants IIS 05-34939 and IIS 08-08692