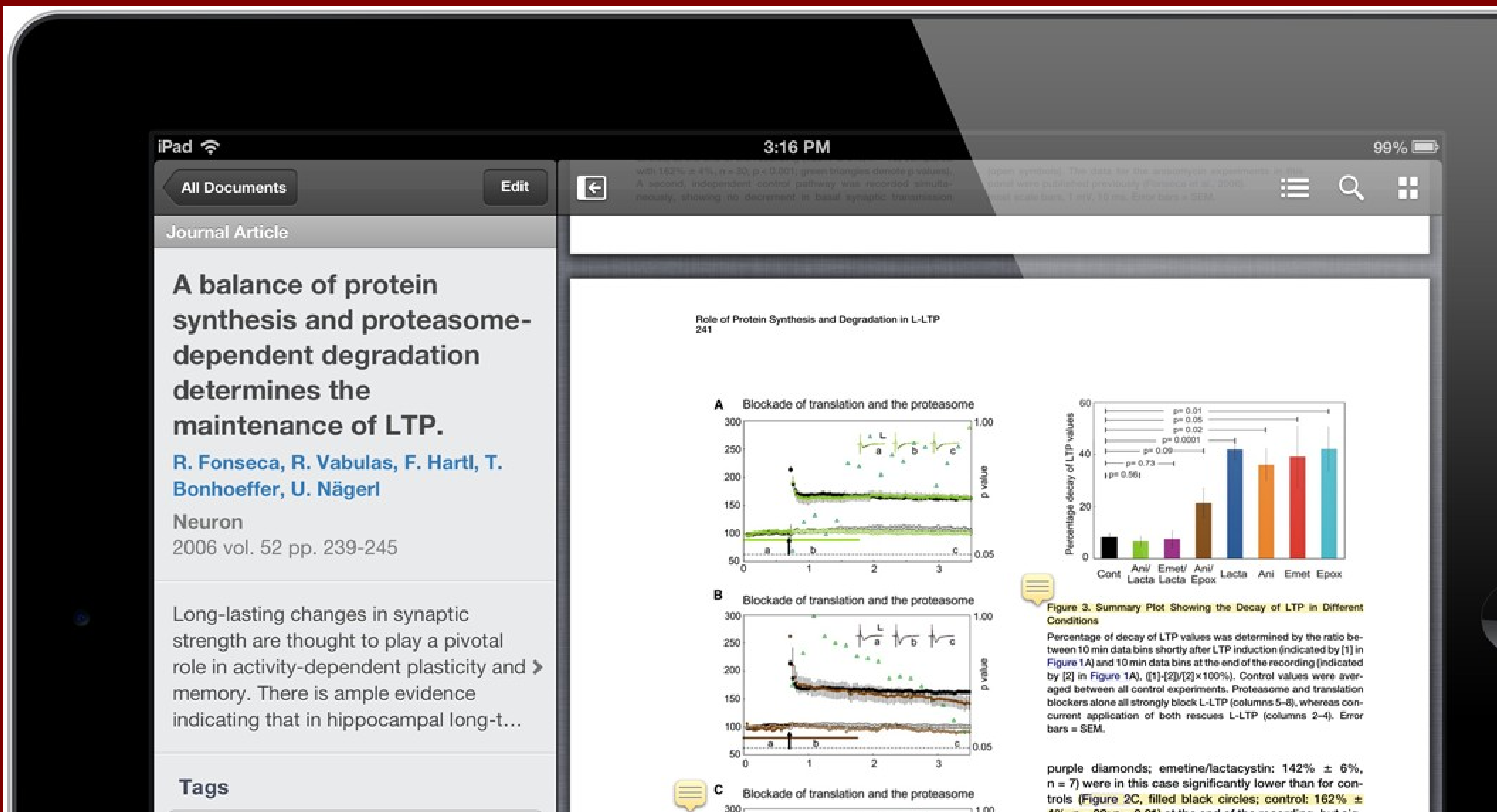
Efficient Top-N Recommendation by Linear Regression

Mark Levy
Mendeley

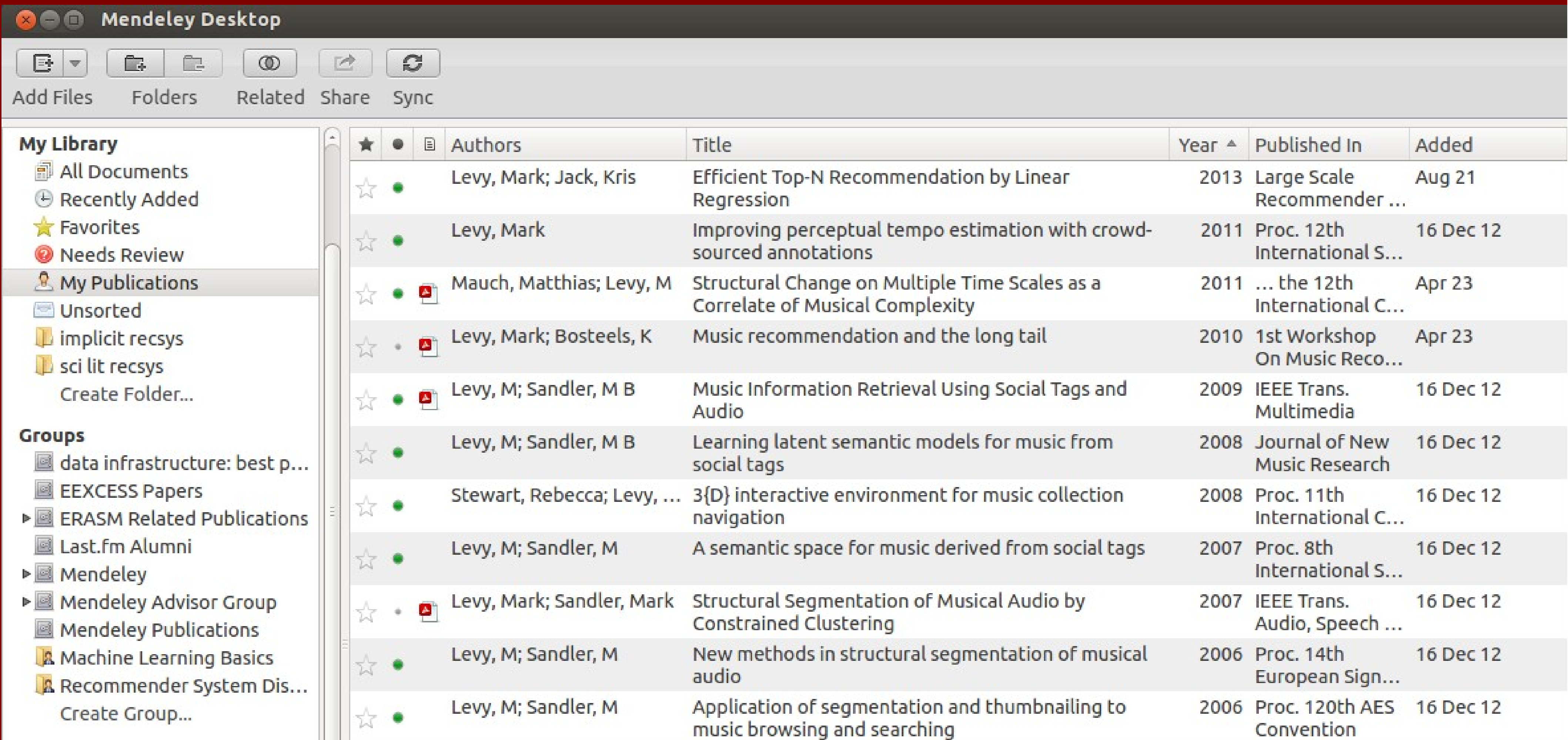
What and why?

- Social network products @Mendeley



What and why?

- Social network products @Mendeley



What and why?

- Social network products @Mendeley

What and why?

- Social network products @Mendeley
- ERASM Eurostars project
- Make newsfeed more interesting
- First task: who to follow

What and why?

- Multiple datasets
  - 2M users, many active
  - ~100M documents
  - author keywords, social tags
  - 50M physical pdfs
  - 15M <user,document> per month
  - User-document matrix currently ~250M non-zeros

What and why?

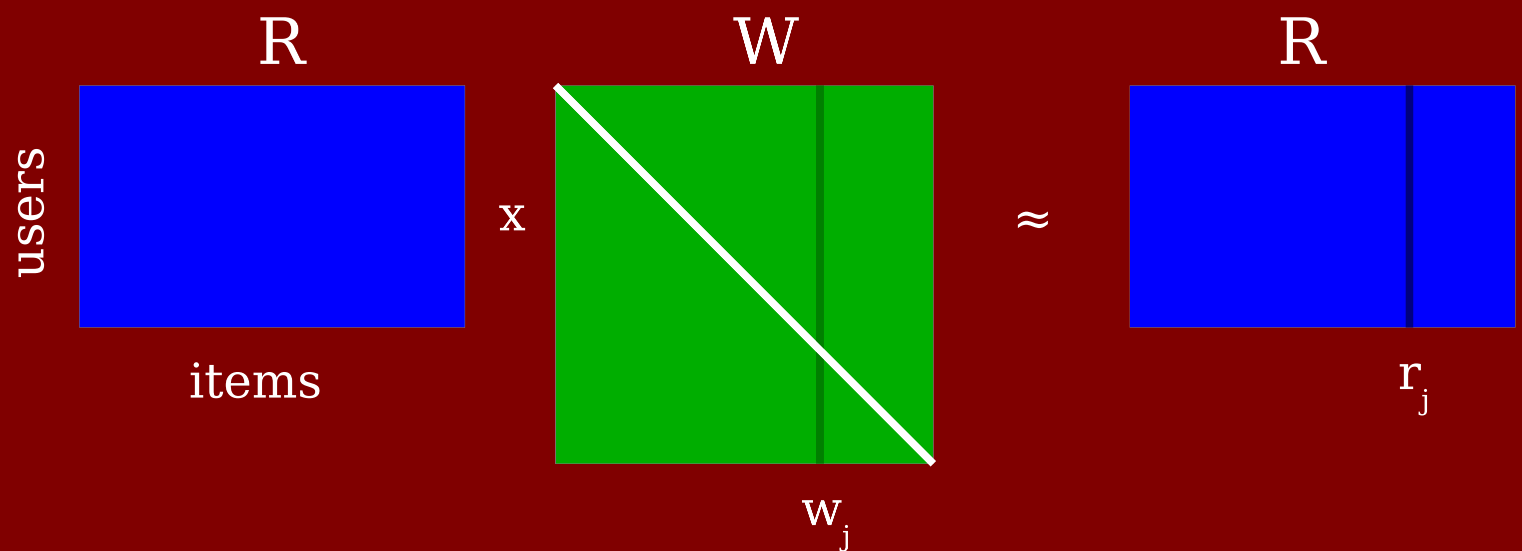
- Approach as item similarity problem
  - with a constrained target item set
- Possible state of the art:
  - pimped old skool neighbourhood method
  - matrix factorization and then neighbourhood
  - something dynamic (?)
  - SLIM
- Use some side data

SLIM

- Learn sparse item similarity weights
- No explicit neighbourhood or metric
- L1 regularization gives sparsity
- Bound-constrained least squares:

$$
\begin{aligned}
\min_{\mathbf{w}_j} \quad & \frac{1}{2}||\mathbf{r}_j - R\mathbf{w}_j||_2^2 + \frac{\beta}{2}||\mathbf{w}_j||_2^2 + \lambda||\mathbf{w}_j||_1 \\
\text{subject to} \quad & \mathbf{w}_j >= 0 \\
& w_{j,j} = 0
\end{aligned}
$$

SLIM



$$\text{subject to} \quad W \geq 0$$
$$\text{diag}(W) = 0$$

```
model.fit(R,r_j)

w_j = model.coef_
```

SLIM

Good:

- Outperforms MF methods on implicit ratings data [1]
- Easy extensions to include side data [2]

Not so good:

- Reported to be slow beyond small datasets [1]

[1] X. Ning and G. Karypis, SLIM: Sparse Linear Methods for Top-N Recommender Systems, Proc. IEEE ICDM, 2011.

[2] X. Ning and G. Karypis, Sparse Linear Methods with Side Information for Top-N Recommendations, Proc. ACM RecSys, 2012.

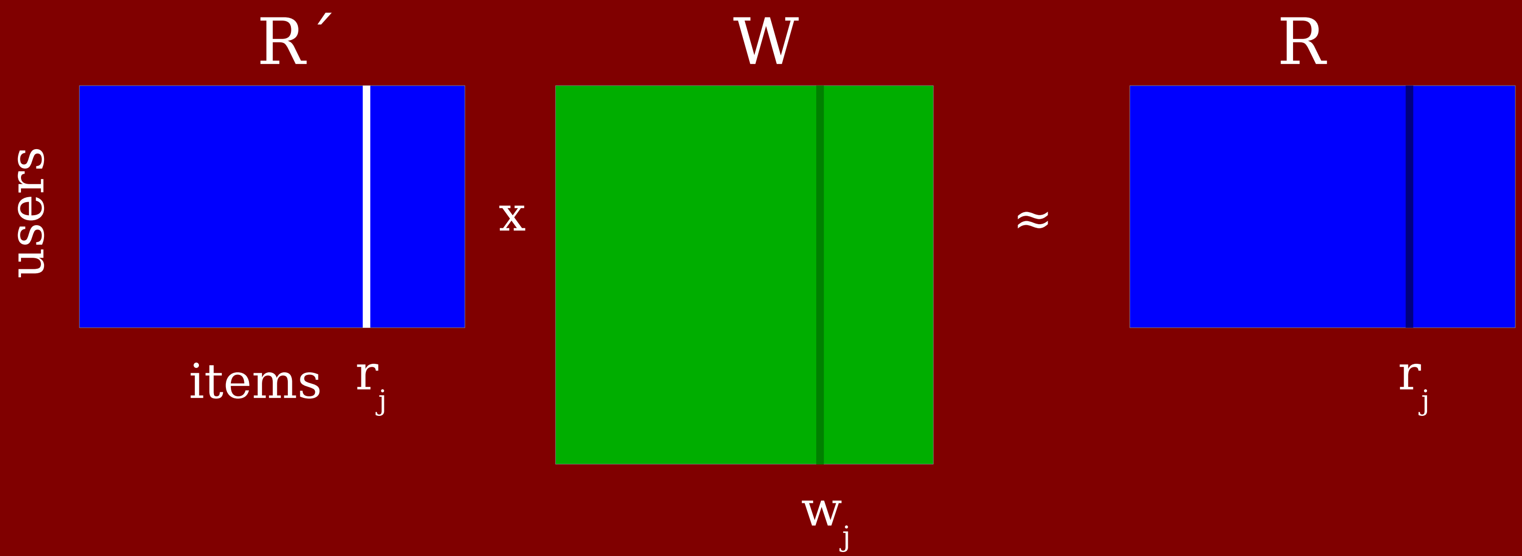
From SLIM to regression

- Avoid constraints
- Regularized regression, learn with SGD
- Easy to implement
- Faster on large datasets

$$\min_{\mathbf{w}_j} \quad \frac{1}{2}||\mathbf{r}_j - R|_{j' \neq j}\mathbf{w}_j||_2^2 + \frac{\beta}{2}||\mathbf{w}_j||_2^2 + \lambda||\mathbf{w}_j||_1$$

$R|_{j' \neq j}$ is the rating matrix with the entries of the j-th column set to zero

From SLIM to regression




```
model.fit(R´,r_j)

w_j = model.coef_
```

Results on reference datasets

Table 1: Performance on ML-100K dataset

method	prec@5	prec@10	prec@15	prec@20	MRR	HR@10
itemKNN	0.350	0.296	0.267	0.246	0.593	0.230
CoFiSet	0.411				0.640	
SLIM	0.344	0.301	0.274	0.253	0.584	0.291
SGDReg	**0.414**	**0.358**	**0.324**	**0.298**	**0.664**	**0.386**

Table 2: Performance on Epinions Trustlet dataset

method	prec@5	prec@10	prec@15	prec@20	MRR	HR@10
itemKNN	0.232	0.196	0.175	0.161	0.438	0.085
CoFiSet	0.244				0.433	
SLIM	0.232	0.205	0.187	0.174	0.411	0.151
SGDReg	**0.291**	**0.254**	**0.230**	**0.213**	**0.495**	**0.166**

Results on reference datasets

Table 3: Performance on ML-10M dataset

method	prec@5	prec@10	prec@15	prec@20	MRR	HR@10
itemKNN						0.238
BPRKNN						0.327
SLIM						0.311
SGDReg	0.612	0.555	0.514	0.481	0.792	**0.332**

Table 3: Time to learn weights for 100 items

dataset	method	time (secs)
ML-10M	SLIM	1055.5 (3.0)
ML-10M	SGDReg	82.4 (0.1)

Results on reference datasets

Table 3: Performance on ML-10M dataset

method	prec@5	prec@10	prec@15	prec@20	MRR	HR@10
itemKNN						0.238
BPRKNN						0.327
SLIM						0.311
SGDReg	0.612	0.555	0.514	0.481	0.792	**0.332**

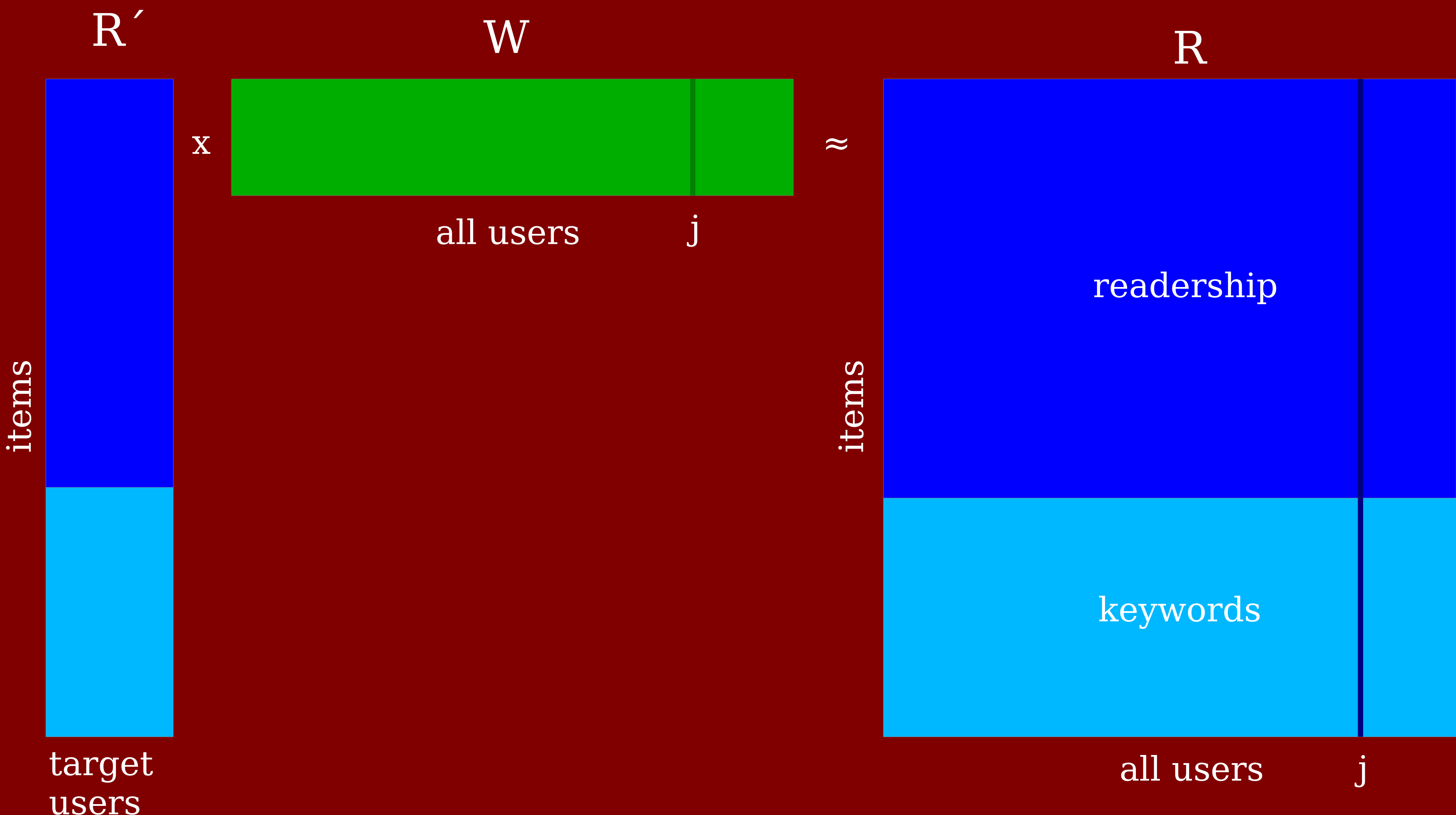
Table 3: Time to learn weights for 100 items

dataset	method	time (secs)
ML-10M	SLIM	1055.5 (3.0)
ML-10M	SGDReg	82.4 (0.1)

Results on Mendeley data

- Stacked readership counts, keyword counts
- 5M docs/keywords, 1M users, 140M non-zeros
- Constrained to ~100k target users
- Python implementation on top of scikit-learn
  - Trivially parallelized with IPython
  - eats CPU but easy on AWS
- 10% improvement over nearest neighbours

Our use case


R´
W
R
x
≈
all users
j
readership
keywords
items
items
target users
all users
j

Tuning regularization constants

- Relate directly to business logic
  - want sparsest similarity lists that are not too sparse
  - “too sparse” = # items with < k similar items
- Grid search with a small sample of items
- Empirically corresponds well to optimising recommendation accuracy of a validation set
  - but faster and easier

Software release: mrec

- Wrote our own small framework
- Includes parallelized SLIM
- Support for evaluation
- BSD licence
- https://github.com/mendeley/mrec
- Please use it or contribute!

mrec 0.1.2 documentation »

Getting started with mrec

Install mrec

You can most easily install *mrec* with pip:

```
$ sudo pip install mrec
```

Thanks for listening

mark.levy@mendeley.com

@gamboviol

https://github.com/gamboviol

https://github.com/mendeley/mrec

Real World Requirements

- Cheap to compute
- Explicable
- Easy to tweak + combine with business rules
- Not a black box
- Work without ratings
- Handle multiple data sources
- Offer something to anon users

Cold Start?

- Can't work miracles for new items
- Serve new users asap
- Fine to run a special system for either case
- Most content leaks
- … or is leaked
- All serious commercial content is annotated

Degrees of Freedom for k-NN

- Input numbers from mining logs
- Temporal “modelling” (e.g. fake users)
- Data pruning (scalability, popularity bias, quality)
- Preprocessing (tf-idf, log/sqrt, …)
- Hand crafted similarity metric
- Hand crafted aggregation formula
- Postprocessing (popularity matching)
- Diversification
- Attention profile