Enhancing Matrix Factorization Through Initialization for Implicit Feedback Databases

Balázs Hidasi
Domonkos Tikk

Gravity R&D Ltd.
Budapest University of Technology and Economics

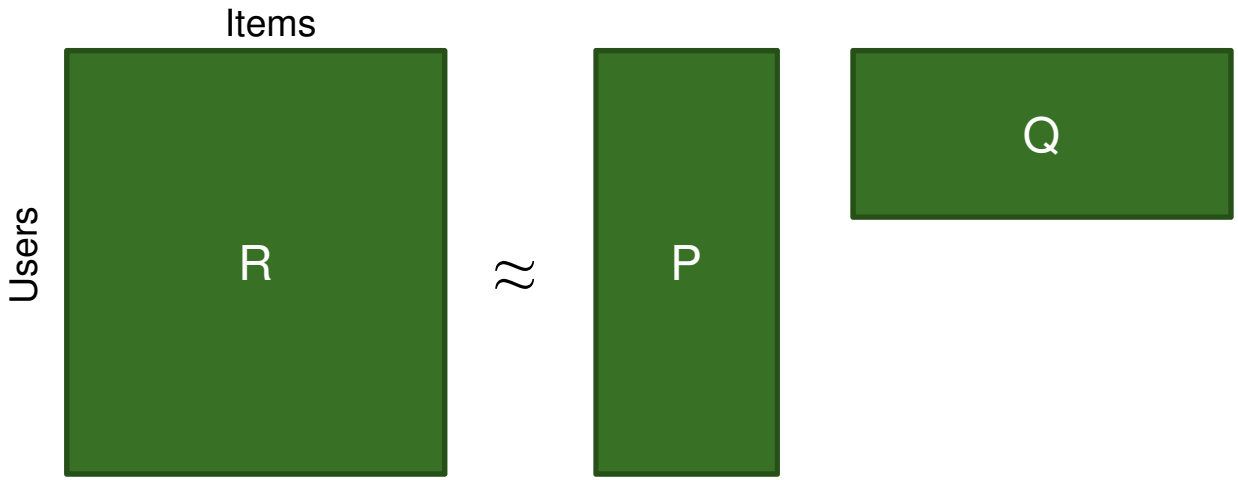
CaRR workshop, 14th February 2012, Lisbon

Outline

- Matrix factoriztion
- Initialization concept
- Methods
  - Naive
  - SimFactor
- Results
- Discussion

Matrix Factorization

- Collaborative Filtering
- One of the most common approaches
- Approximates the rating matrix as product of low-rank matrices



Matrix factorization

- **Initialize** P and Q with small **random** numbers
- Teach P and Q
  - Alternating Least Squares
  - Gradient Descent
  - Etc.
- Transforms the data to a feature space
  - Separately for users and items
  - Noise reduction
  - Compression
  - Generalization

Implicit feedback

- No ratings
- User-item interactions (events)
- Much noisier
  - Presence of an event → might not be positive feedback
  - Absence of an event → does not mean negative feedback
  - No negative feedback is available!
- More common problem
- MF for implicit feedback
  - Less accurate results due to noise
  - Mostly ALS is used
  - Scalability problems (rating matrix is dense)

Concept

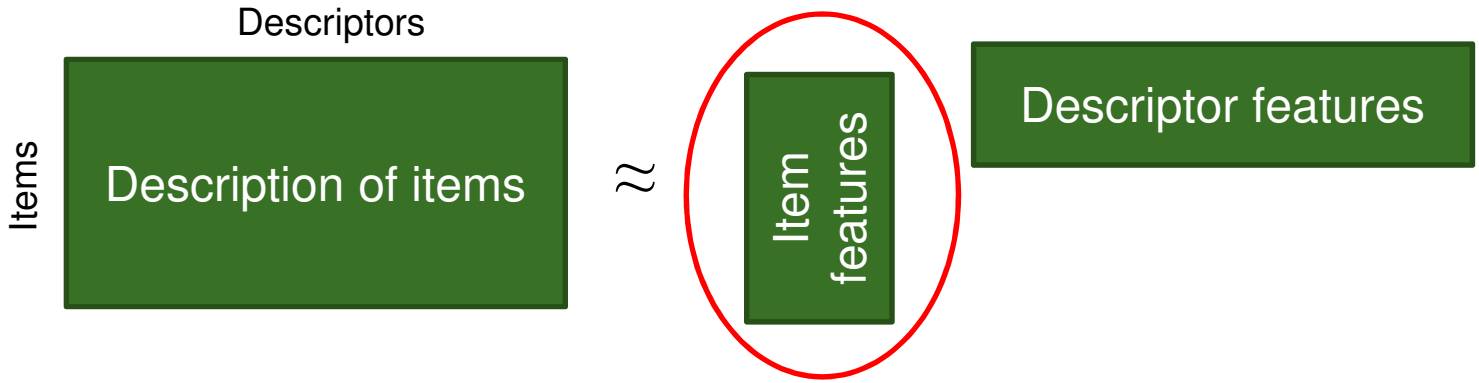
- Good MF model
  - The feature vectors of similar entities are similar
  - If data is too noisy → similar entities won't be similar by their features
- Start MF from a „good" point
  - Feature vector similarities are OK
- Data is more than just events
  - Metadata
    - Info about items/users
  - Contextual data
    - In what context did the event occured
  - Can we incorporate those to help implicit MF?

Naive approach

- Describe items using any data we have (detailed later)
  - Long, sparse vectors for item description
- Compress these vectors to dense feature vectors
  - PCA, MLP, MF, …
  - Length of desired vectors = Number of features in MF
- Use these features as starting points

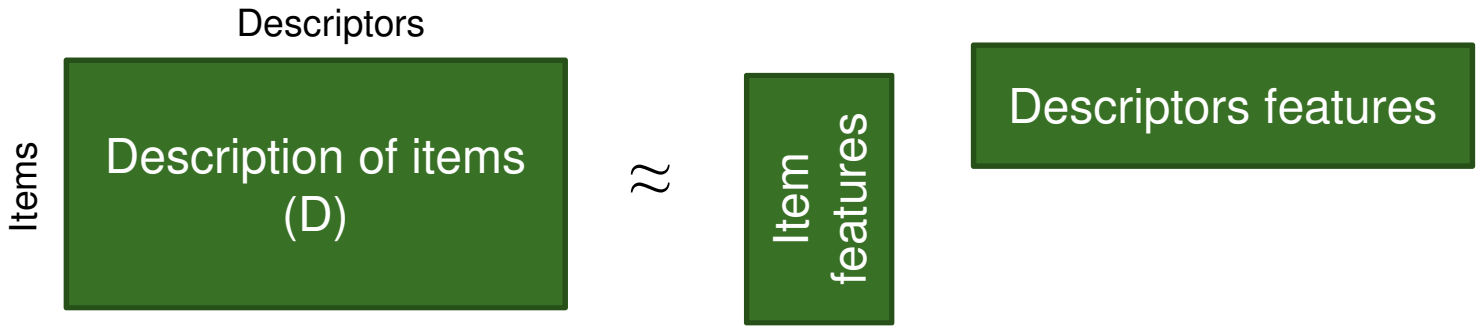
Naive approach

- Compression and also noise reduction
- Does not really care about similarities
- But often feature similarities are not that bad
- If MF is used
  - Half of the results is thrown out

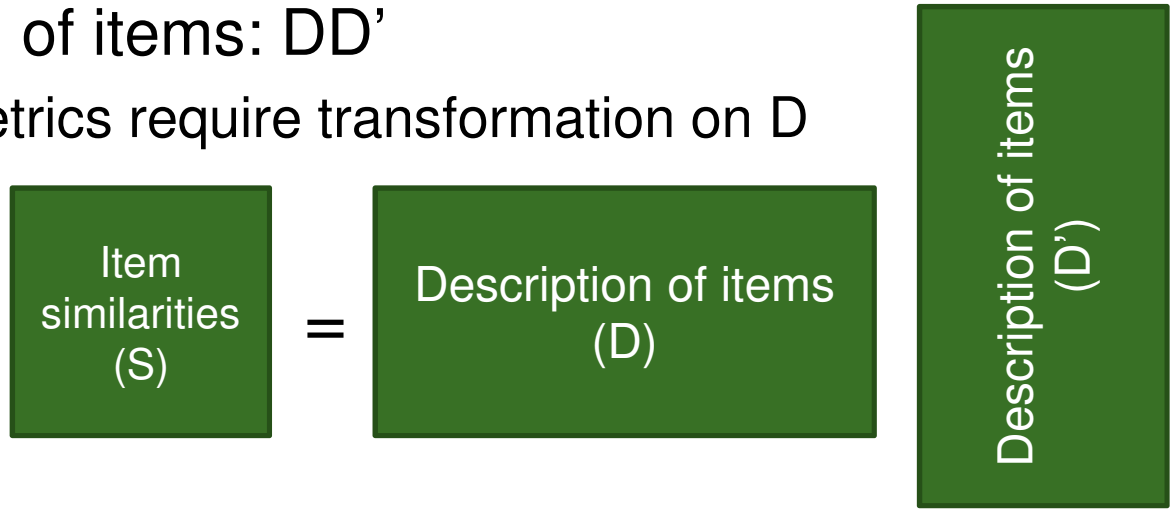


SIMFACTOR ALGORITHM

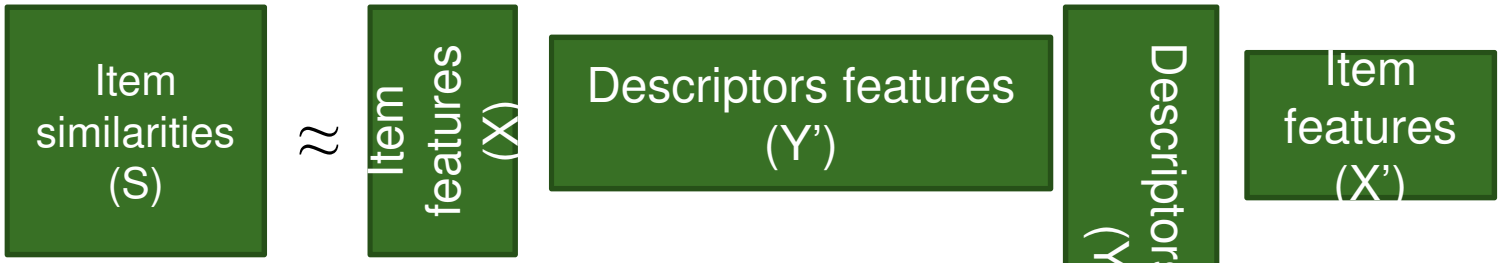
- Try to preserve similarities better
- Starting from an MF of item description



- Similarities of items: DD'
  - Some metrics require transformation on D



SIMFACTOR ALGORITHM



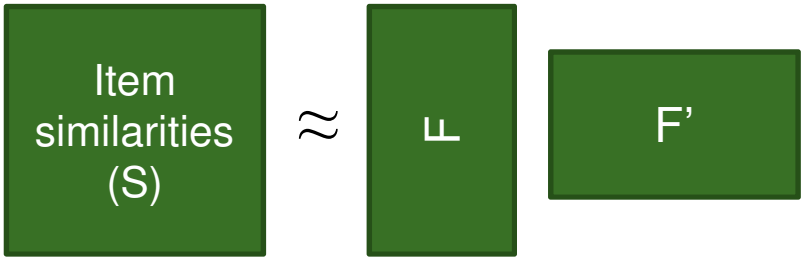
- Similarity approximation

- Y'Y → KxK symmetric
  - Eigendecomposition

SIMFACTOR ALGORITHM

- λ diagonal → λ = SQRT(λ) * SQRT(λ)

- X*U*SQRT(λ) = (SQRT(λ)*U'*X')'=F
- F is MxK matrix
- S ≈ F * F' → F used for initialization

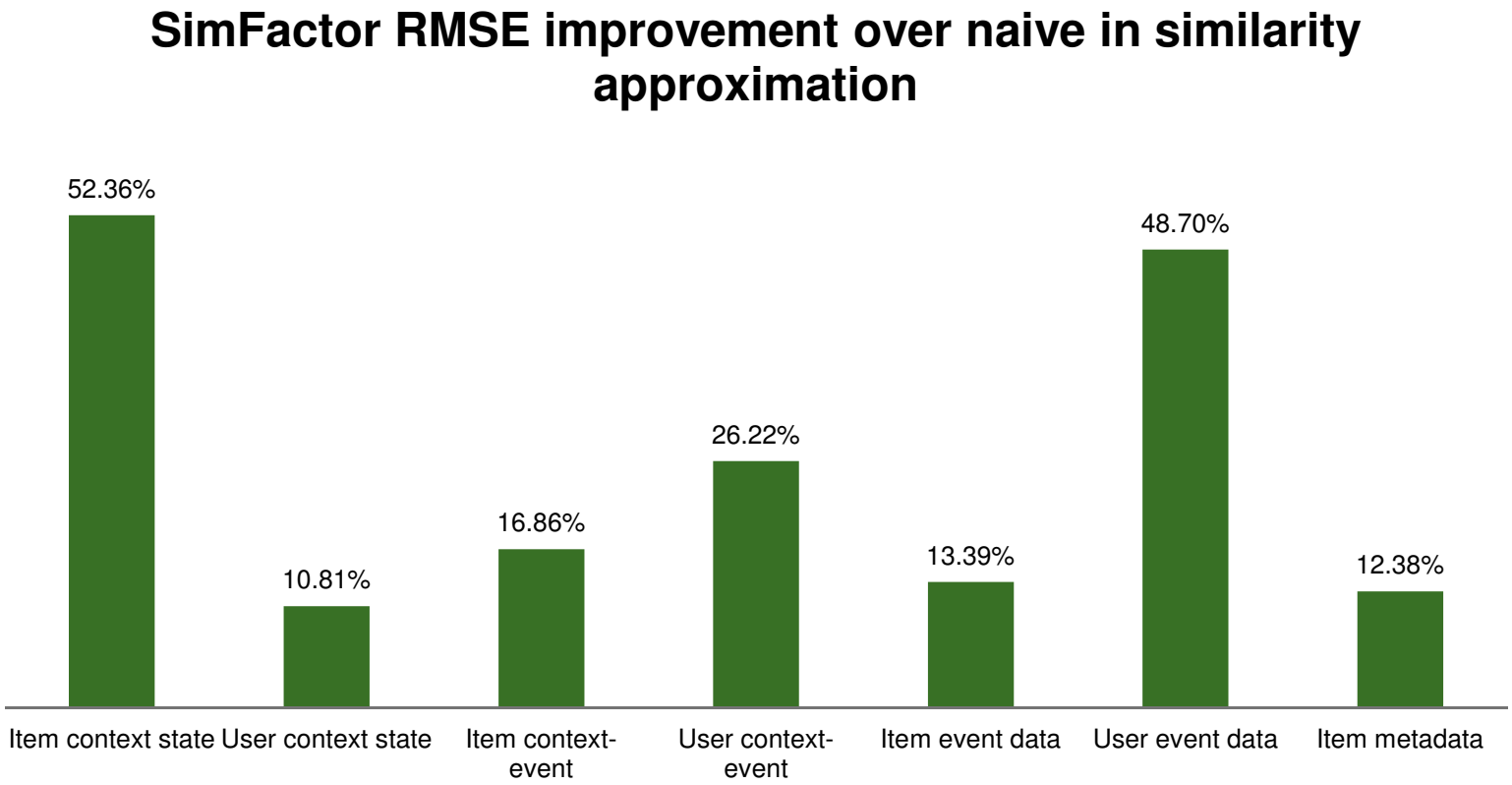


Creating the description matrix

- „Any” data about the entity
  - Vector-space reprezentation
- For **Items**:
  - Metadata vector (title, category, description, etc)
  - Event vector (who bought the item)
  - Context-state vector (in which context state was it bought)
  - Context-event (in which context state who bought it)
- For **Users**:
  - All above except metadata
- Currently: Choose one source for D matrix
- Context used: seasonality

Experiments: Similarity preservation

- Real life dataset: online grocery shopping events



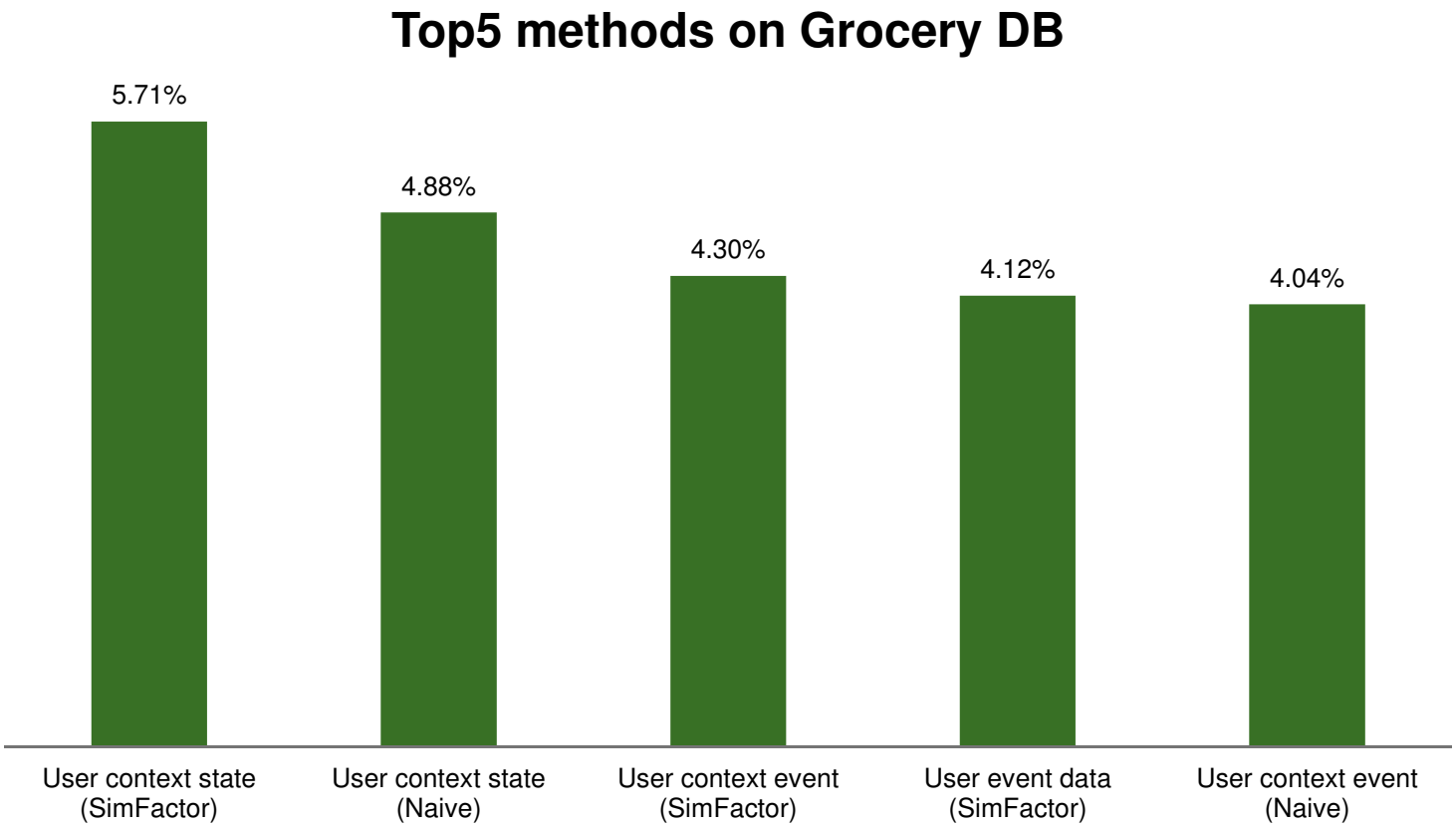
- SimFactor approximates similarities better

Experiments: Initialization

- Using different description matrices
- And both naive and SimFactor initialization
- Baseline: random init
- Evaluation metric: recall@50

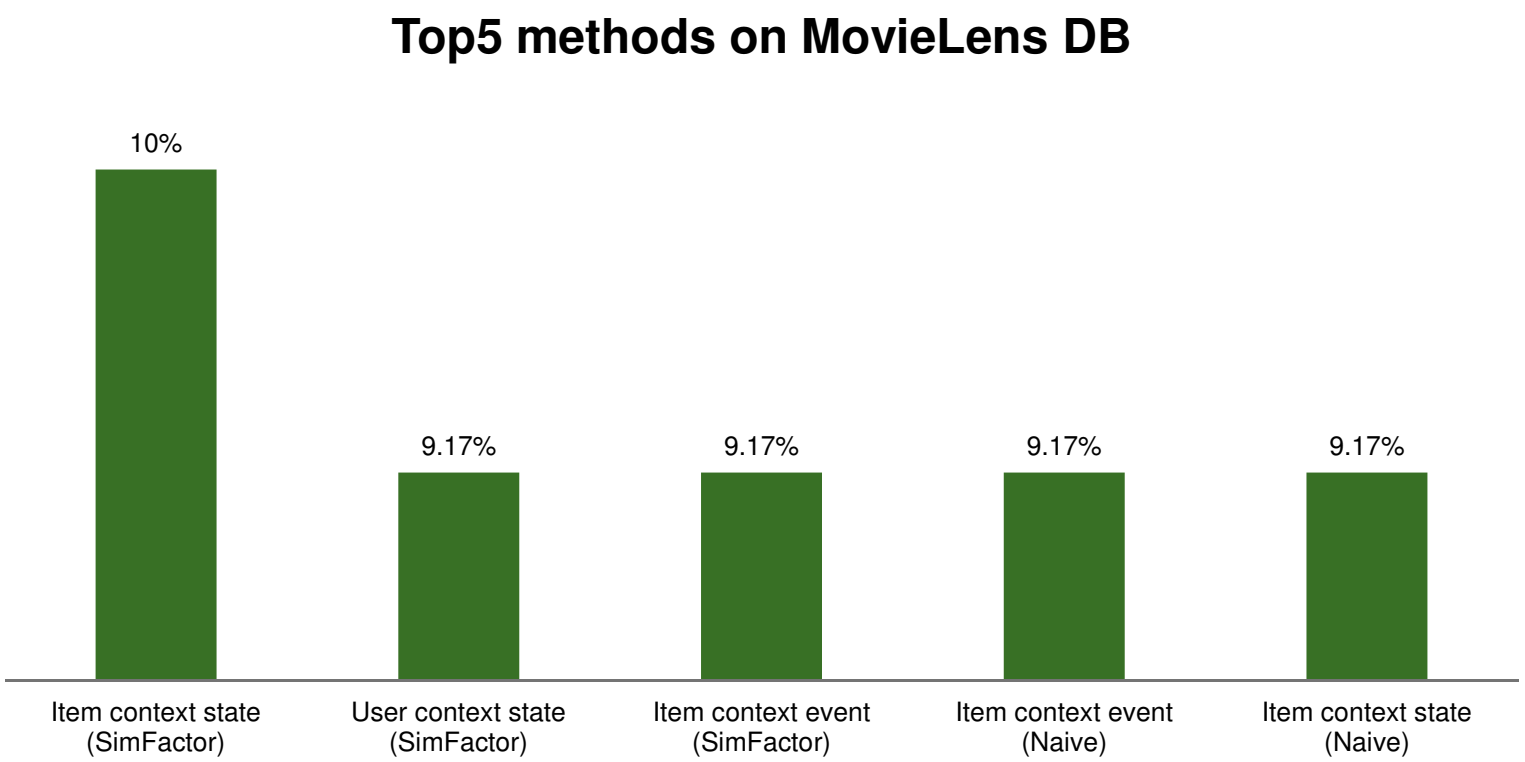
Experiments: Grocery DB

- Up to 6% improvement
- Best methods use SimFactor and user context data



EXPERIMENTS: „IMPLICITIZED" MOVIELENS

- Keeping 5 star ratings → implicit events
- Up to 10% improvement
- Best methods use SimFactor and item context data



Discussion of results

- SimFactor yields better results than naive
- Context information yields better results than other descriptions
- Context information separates well between entities
  - Grocery: User context
    - People's routines
    - Different types of shoppings in different times
  - MovieLens: Item context
    - Different types of movies watched on different hours
- Context-based similarity

WHY CONTEXT?

- Grocery example
  - Correlation between context states by users→ low

ITEM							
	Mon	**Tue**	**Wed**	**Thu**	**Fri**	**Sat**	**Sun**
Mon	1,00	0,79	0,79	0,78	0,76	0,70	0,74
Tue	0,79	1,00	0,79	0,78	0,76	0,69	0,73
Wed	0,79	0,79	1,00	0,79	0,76	0,70	0,74
Thu	0,78	0,78	0,79	1,00	0,76	0,71	0,74
Fri	0,76	0,76	0,76	0,76	1,00	0,71	0,72
Sat	0,70	0,69	0,70	0,71	0,71	1,00	0,71
Sun	0,74	0,73	0,74	0,74	0,72	0,71	1,00

USER							
	Mon	**Tue**	**Wed**	**Thu**	**Fri**	**Sat**	**Sun**
Mon	1,00	0,36	0,34	0,34	0,35	0,29	0,19
Tue	0,36	1,00	0,34	0,34	0,33	0,29	0,19
Wed	0,34	0,34	1,00	0,36	0,35	0,27	0,17
Thu	0,34	0,34	0,36	1,00	0,39	0,30	0,16
Fri	0,35	0,33	0,35	0,39	1,00	0,32	0,16
Sat	0,29	0,29	0,27	0,30	0,32	1,00	0,33
Sun	0,19	0,19	0,17	0,16	0,16	0,33	1,00

- Why can context aware algorithms be efficient?
  - Different recommendations in different context states
  - Context differentiates well between entities
    - Easier subtasks

Conclusion & Future work

- SimFactor → Similarity preserving compression
- Similarity based MF initialization:
  - Description matrix from any data
  - Apply SimFactor
  - Use output as initial features for MF
- Context differentitates between entities well
- Future work:
  - Mixed description matrix (multiple data sources)
  - Multiple description matrix
  - Using different context information
  - Using different similarity metrics

Thanks for your attention!

For more of my recommender systems related research visit my website:
http://www.hidasi.eu