Ranking prediction by online learning

Róbert Pálovics

Informatics Laboratory, Department of Computer and Automation Research Institute, Hungarian Academy of Sciences

https://dms.sztaki.hu/en

July 2, 2015

OUTLINE

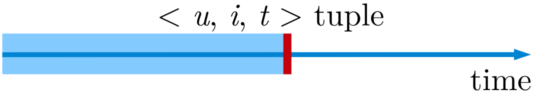
- Online ranking prediction
- Exploiting social influence in online RS
- Location-aware online learning

RECOMMENDER SYSTEMS

- Utility matrix R, only a few known values
- Rating prediction vs. ranking prediction
- Explicit vs implicit data
- Collaborative filtering vs. contend based

Online ranking prediction

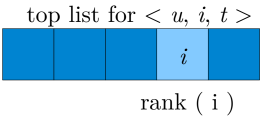
- Online recommendation
  - after each event recommend a new top list of items
  - after each event update the recommender model
  - implicit data
- Temporal evaluation
  - for each tuple $< u, i, t >$ (user, item, timestamp)
  - evaluate the given single tuple in question against the recommended top list
- Iterate on the dataset only at once



Online ranking prediction

- Evaluate the given single tuple in question against the recommended top list
- There is only one single relevant item, use

$$\mathrm{DCG@K}(a) = \begin{cases} 0 & \text{if } \operatorname{rank}(i) > K; \\ \dfrac{1}{\log_2(\operatorname{rank}(i)+1)} & \text{otherwise.} \end{cases}$$



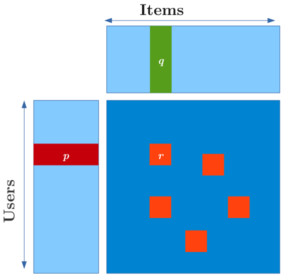
Matrix factorization

- Model $\hat{R} = P \cdot Q$, where $P \in \mathbb{R}^{n \times k}$ and $Q \in \mathbb{R}^{k \times m}$, $\hat{r}_{ui} = p_u \cdot q_i$
- Objective - mean squared error (MSE), for $(u, i) \in Tr$

$$F_{ui} = (r_{ui} - \hat{r}_{ui})^2$$

- Optimization - stochastic gradient descent (SGD)

$$p_u \leftarrow p_u - \text{lrate} \cdot \frac{\partial F}{\partial p_u} = p_u - \text{lrate} \cdot Err \cdot q_i$$

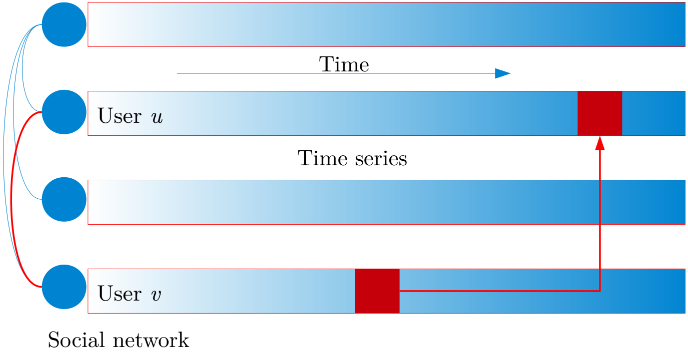


Online Matrix Factorization

- Single iteration over the training data in temporal order
- Updating after each new element
- High learning rates
- More emphasis on recent events
- Works well on non-stationary datasets

Network Influence

- *User-User* social graph + *User-Item* activity time series (bipartite graph)
- Detect social influences, influential pairs
- Improve top-k recommendation



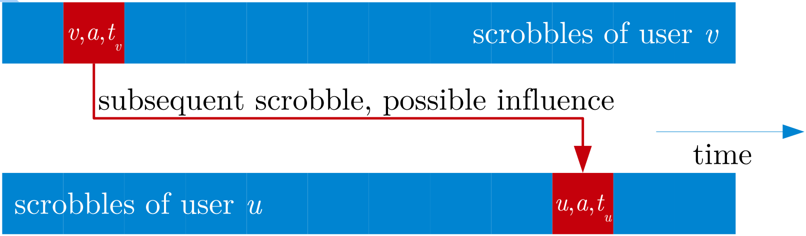
Last.fm

- Online service in music based social networking
- "Scrobbling": collecting listening activity of users
- Music recommendation system
- Social network
- Users see each others scrobbling activity

Influence Probability

- Key concept: influence between neighbors u and v,
  - subsequent scrobble, $v \xrightarrow{a;\Delta t \leq t} u$
  - *and* the reason is influence
- Influence probability

$$P(\text{Influence}, v \xrightarrow{a;\Delta t \leq t} u) = P(\text{Influence} \mid v \xrightarrow{a;\Delta t \leq t} u) \cdot P(v \xrightarrow{a;\Delta t \leq t} u)$$



Influence Probability, Left Term

$$P(\text{Influence}, v \xrightarrow{a;\Delta t \leq t} u) = {\color{red}P(\text{Influence} \mid v \xrightarrow{a;\Delta t \leq t} u)} \cdot P(v \xrightarrow{a;\Delta t \leq t} u)$$

- Approximation by measurements

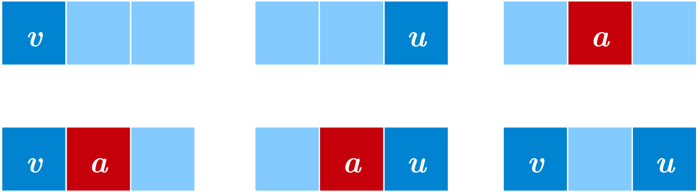
$$P(\text{Influence} \mid v \xrightarrow{a;\Delta t \leq t} u) \approx P(\text{Influence} \mid \Delta t \leq t) \approx 1 - c \log t$$

- Slowly decreasing logarithmic function

Influence Probability, Right Term

$$P(\text{Influence}, v \xrightarrow{a;\Delta t \leq t} u) = P(\text{Influence} \mid v \xrightarrow{a;\Delta t \leq t} u) \cdot P(v \xrightarrow{a;\Delta t \leq t} u)$$

- Probability of event $v \xrightarrow{a;\Delta t \leq t} u$ in the time series
- Learned by modeling

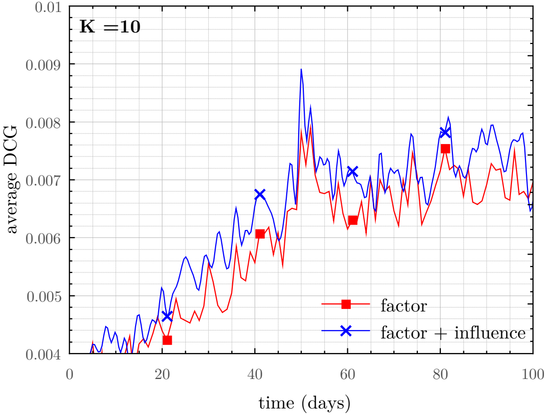


Experiments - About Last.fm

- Available for us under NDA for Last.fm, selection criteria
- Structure: network + scrobbling time series
  - $71,000$ users, $285,241$ edges
  - 2 year scrobble timeline, $2,073,395$ artists
  - between 01 January 2010 and 31 December 2011
  - $979,391,001$ scrobbles
  - $57,274,158$ 1st-time scrobbles
- We train factor models only on the 1st time scrobbles
- Artists with popularity less than 14 are excluded
- Evaluation on each 1st time scrobble in the *second* year

Experiments - Final Combination

- Factor and influence models combine well, the average improvement is
  - 7 % for DCG@10



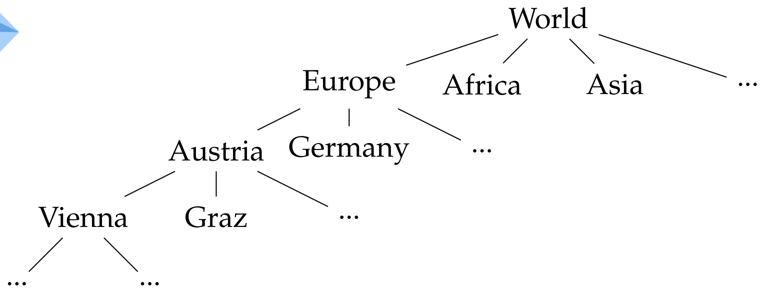
Location-aware online learning

- Twitter dataset
- Temporal hashtag recommendation
- Twitter: highly non-stationary data
- (u, h, l, t) geoinfo
- Idea: tree structure of geographical areas

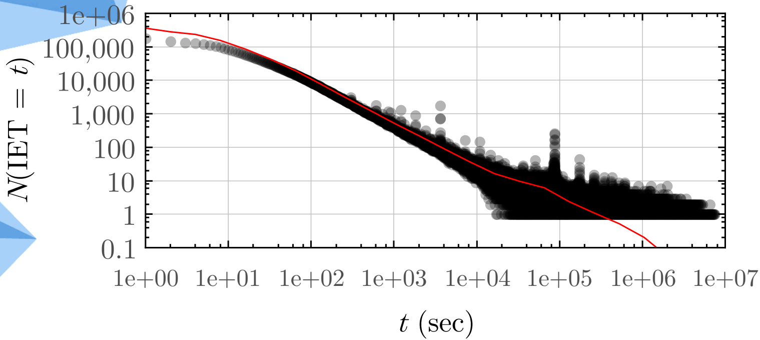
number of records	6,978,478
number of unique user-hashtag pairs	2,993,183
number of users	792,860
number of items	268,489
number of countries	49

TREE CONSTRUCTION

- 214,230 nodes containing 190,315 leaves.
- The depth of the tree is 6
- The hashtag time series data covered 30,450 leaves from the whole tree.



Recency



$$P(\tau = t) = (\alpha - 1) \cdot t^{-\alpha} \text{ and } P(1 \leq \tau \leq t) = 1 - t^{(1-\alpha)}$$

$$P(t < \tau \leq t + \Delta t | \tau > t) = \frac{P(\tau \leq t + \Delta t) - P(\tau \leq t)}{1 - P(\tau \leq t)} = 1 - \left(1 + \frac{\Delta t}{t}\right)^{(1-\alpha)}$$

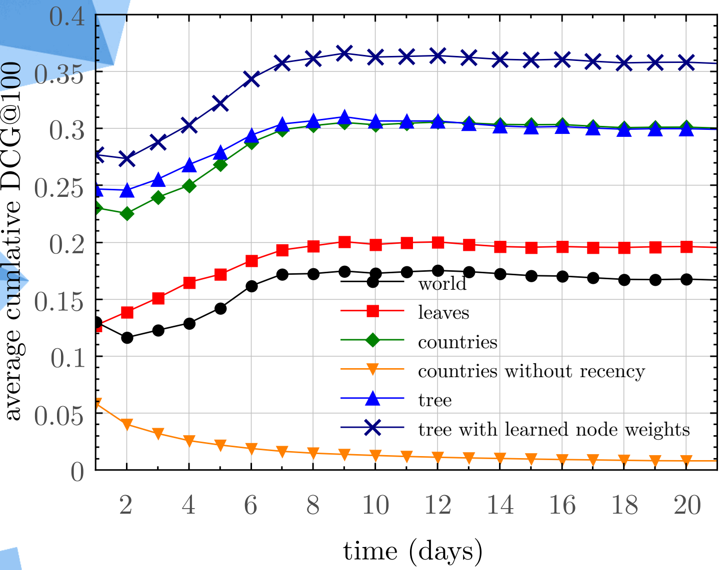
Modeling

- Online MF as baseline $\rightarrow$ NOT working !
- Tree + Recency + Bias model:

$$\hat{r}(u,h,t,l) = \sum_{n \in \text{Path(l)}} \hat{w}_n \cdot f(t - t_{n,h})$$

- $\hat{w}_n$ node biases learned with SGD
- $\hat{w}_n$ already includes node reliability and popularity
- Different heuristic baselines

Results


average cumlative DCG@100
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
2
4
6
8
10
12
14
16
18
20
time (days)
world
leaves
countries
countries without recency
tree
tree with learned node weights