Food Recipe Recommendation and the Social Filtering formalism

Daniel Bernardes
Postdoctoral researcher @ L2TI/Université Paris 13

WORKSHOP BIG DATA, MACHINE LEARNING AND SOCIAL NETWORK ANALYSIS - 15/12/2014

Outline

1. Introduction
2. Standard Methods
3. Social Filtering formalism
4. Evaluation on KI dataset
5. Perspectives

Online recommendation value

- Netflix: 2/3 of the movies watched are recommended
- Amazon: 35% sales from recommendations
- Choicestream: 28% of the people would buy more music if they found what they liked.
- Google News: recommendations generate 38% more clickthrough

Open Food System

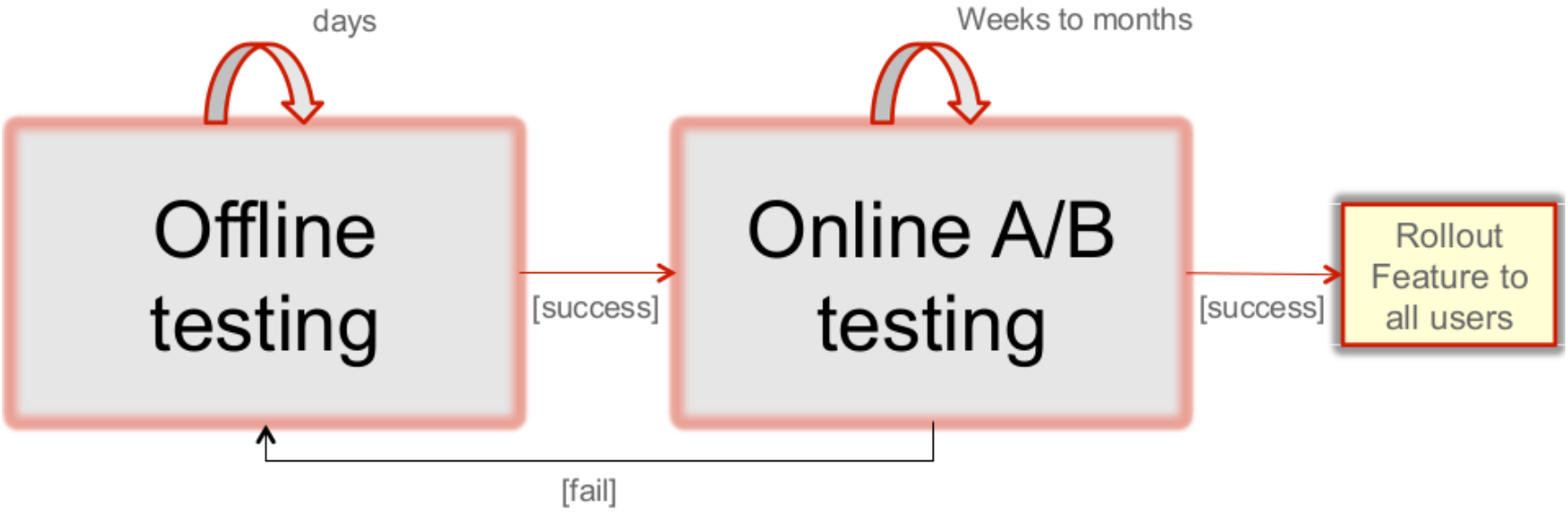
- the structuring of digital recipe data on a universal rich data format;
- the offer of **recipe recommendations** and **personalised menus**;
- communication with automated domestic appliances and physical measurement devices;
- **connect** a global community of cooking enthusiasts.

Data-driven approach

Lessons from Amazon, Google, Netflix, etc:

1. Start with an hypothesis
2. Design a test
3. Execute the test
4. Let data speak for itself

Offline/Online testing process



Offline testing

- Optimize algorithms offline
- Measure model performance examining:
  - Precision, recall, MAP, covering, ...
- Offline performance used to make informed decisions on follow-up A/B tests

Offline testing: in the following

- Focus on top-n recommendations
- Analysing explicit (ratings) or implicit interactions (number of interactions)
- Following performance evaluation in terms of precision and diversity

Standard Methods

CF: Association rules (bigrams)

- Identify couple of products frequently consumed together
- Generate rules from these couples if they have minimum levels of confidence

$$\text{Conf}(i \rightarrow j) = \frac{\#\text{ Users who cons. by } i \text{ and } j}{\#\text{ Users cons. by } i}$$

- Scan target user's items and use rules to discover potential suggestions, using confidence to rank N best
- Fast to implement and execute (e.g. Apriori algorithm)

Collaborative filtering: memory-based

- Based on similarity measures between users and items in terms of cosine distance or Pearson correlation

$$\mathrm{Sim}(a,u) = \cos\left(r_{a\cdot}, r_{u\cdot}\right) = \frac{r_{a\cdot} \cdot r_{u\cdot}}{\| r_{a\cdot} \| \| r_{u\cdot} \|}$$

$$\mathrm{Sim}(a,u) = \mathrm{PCC}(r_{a\cdot}, r_{u\cdot}) = \frac{\left(r_{a\cdot} - \overline{r_{a\cdot}}\right) \cdot \left(r_{u\cdot} - \overline{r_{u\cdot}}\right)}{\| \left(r_{a\cdot} - \overline{r_{a\cdot}}\right) \| \| \left(r_{u\cdot} - \overline{r_{u\cdot}}\right) \|}$$

- User-item scores calculated from the k nearest neighbors
- Good enough performance for general-purpose problems

CF memory-based / user-based

1. Calculate the similarity between the target user and the other users.
2. Identify set of k users most similar to the target user.
3. For each item these users consumed and the target has not consumed, compute a score averaging the ratings from these similar users (weighted by sim.)
4. Based on this score recommend a set of top N items.

Score aggregation

$$\text{Score}(a,u) = \frac{1}{|K(a)|} \sum_{u \in K(a)} r_{ui}$$

$$\text{Score}(a,u) = \frac{1}{\sum_{u \in K(a) \cap U(i)} |\text{Sim}(a,u)|} \sum_{u \in K(a)} \text{Sim}(a,u)\ r_{ui}$$

$$\text{Score}(a,u) = \overline{r_{a\cdot}} + \frac{1}{\sum_{u \in K(a) \cap U(i)} |\text{Sim}(a,u)|} \sum_{u \in K(a)} \text{Sim}(a,u)\ (r_{ui} - \overline{r_{u\cdot}})$$

CF memory-based / item-based

1. Select a target item i and select the k items most similar to i rated by the target user.
2. Generate a score averaging the ratings from the target user weighted by the similarity of each of these items.
3. Based on this score recommend a set of top N items.

Score aggregation

$$\text{Score}(a,i) = \frac{1}{|V(i)|} \sum_{j \in V(i)} r_{aj}$$

$$\text{Score}(a,i) = \frac{1}{\sum_{j \in V(i) \cap I(a)} |\text{Sim}(i,j)|} \sum_{j \in V(i)} \text{Sim}(i,j)\ r_{aj}$$

$$\text{Score}(a,i) = \overline{r_{\cdot i}} + \frac{1}{\sum_{j \in V(i) \cap I(a)} |\text{Sim}(i,j)|} \sum_{j \in V(i)} \text{Sim}(i,j)\ \left(r_{aj} - \overline{r_{\cdot j}}\right)$$

Social Filtering

Social Filtering

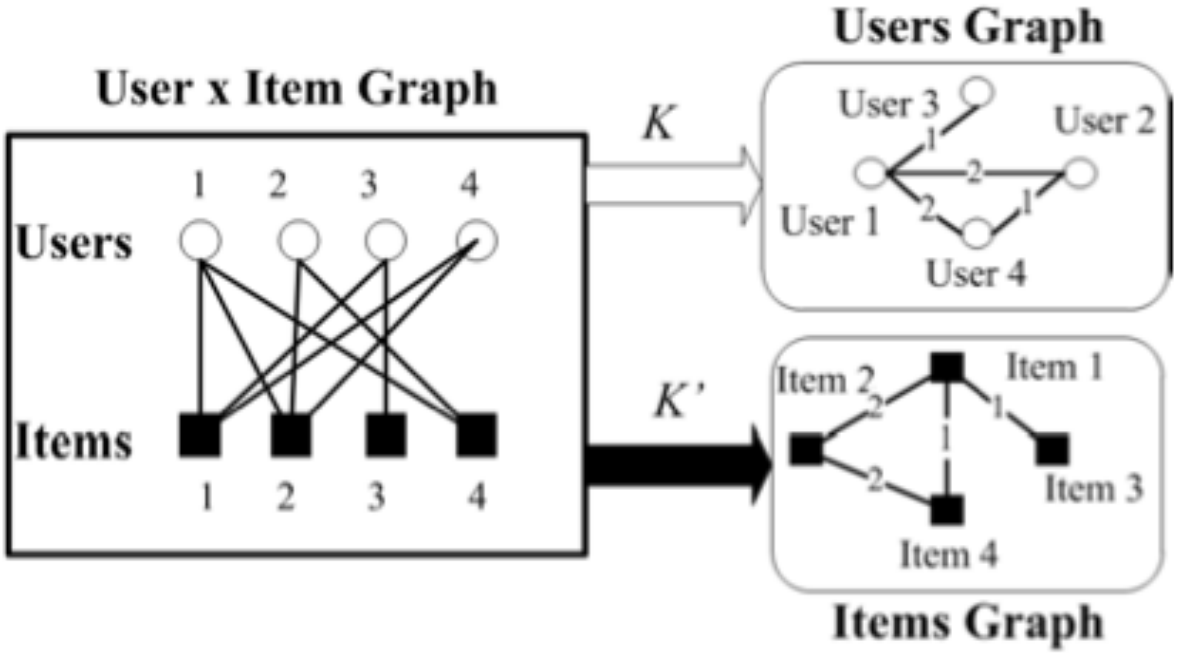
General Framework that generalizes:

- CF memory-based
- Locality CF memory based (Asymmetric Cosine)
- Association Rules (bigrams)

Idea:

- Compute different similarities
- Vary the neighborhood before aggregate scores

Strategy: define neighborhood



Strategy: compute several similarities

- Support-based
- Confiance-based
- Cosine user-based
- Cosine item-based
- Asymmetric cosine
- Asymmetric confidence
- Jaccard
- ...

Support similarity

$$\text{Supp}(a \to u) = \frac{\#\text{ Items cons. by } a \text{ and } u}{\#\text{ Items}} = \frac{1}{C}\sum_{i=1}^{C} r_{ai} r_{ui}$$

$$\text{Supp}(i \to j) = \frac{\#\text{ Users who cons. by } i \text{ and } j}{\#\text{ Users}} = \frac{1}{L}\sum_{u=1}^{L} r_{ui} r_{uj}$$

$$\text{Sim}(a, u) = \text{Supp}(a \to u) \qquad \text{Sim}(i, j) = \text{Supp}(i \to j)$$

Confidence similarity

$$\text{Conf}(a \to u) = \frac{\#\text{ Items cons. by } a \text{ and } u}{\#\text{ Items cons. by } a} = \frac{\sum_{i=1}^{C} r_{ai} r_{ui}}{\sum_{i=1}^{C} r_{ai}}$$

$$\text{Conf}(i \to j) = \frac{\#\text{ Users who cons. by } i \text{ and } j}{\#\text{ Users cons. by } i} = \frac{\sum_{u=1}^{L} r_{ui} r_{uj}}{\sum_{u=1}^{L} r_{ui}}$$

Bigrams: confidence similarity and neighborhood defined by the most similar item

Asymmetric Confidence similarity

$$\mathrm{Sim}(a,u) = \mathrm{Conf}(a \to u)^{\alpha}\ \mathrm{Conf}(u \to a)^{(1-\alpha)}$$

$$\mathrm{Sim}(i,j) = \mathrm{Conf}(i \to j)^{\alpha}\ \mathrm{Conf}(j \to i)^{(1-\alpha)}$$

Jaccard similarity

$$\text{Jaccard}(a, u) = \frac{|r_{a\cdot} \cap r_{u\cdot}|}{|r_{a\cdot} \cup r_{u\cdot}|} \qquad \text{Jaccard}(a, u) = \frac{|r_{\cdot i} \cap r_{\cdot j}|}{|r_{\cdot i} \cup r_{\cdot j}|}$$

Advantages

- Allow exploration of several similarities at once, using the same projected graph
- Allows for explicit social network recommendations
- Social recommendation using explicit social network
- And tools from social network analysis, such as community detection (Louvain, Local methods, ...)

Evaluation on KI dataset

KI Dataset

Event count	Description
769,202	Add recipe
6,194	Add comment
128,501	Add cookbook
296,561	Add to cookbook
45,383	Delete recipe
12,845	Delete cookbook
24,569	Follow account
36	Friend account
3,315,712	Favorite recipe
208,886	Update recipe
3,431	Update cookbook
1,304	Unfollow account
21,655	Ratings recipe
182	Delete profile

Users	Items	Events
301,973	1,787,241	4,834,461

Users	Items	Bin. Values
65,028	859,640	4,201,801

Results

Method	Popularity	Bigrams	CF/IB	CF/UB
MAP @ 10	0.002	**0.032**	0.029	0.010
Precision @ 10	0.005	**0.053**	0.050	0.024
Recall @ 10	0.005	**0.040**	0.040	0.039
Users Full Coverage	100.00%	77.64%	97.47%	**98.88%**
Users Partial Coverage	0.00%	22.36%	2.53%	**1.12%**
– Avg. num. of recs.	-	3.49	5.76	2.25
Items coverage	0.002%	1.14%	**4.88%**	3.21%
– Proportion in Tail	0.00%	0.00%	**2.53%**	1.96%
– Proportion in Head	100.00%	100.00%	**97.47%**	98.04%
Computation time	0:04:30	0:15:00	13:18:00	4:37:00

Table 4.3: Results from standard methods. Bigrams with rules featuring support $s > 13$ and confidence $c > 0.1$; item-based and user-based collaborative filtering with cosine similarity.

Results

Method	IB Sup.	IB Conf.	IB AC	UB AC	UB Conf.	UB Sup.
MAP @ 10	0.028	0.025	**0.029**	0.027	0.019	0.020
Precision @ 10	0.047	0.045	0.050	**0.066**	0.049	0.049
Recall @ 10	0.039	0.030	**0.040**	0.011	0.009	0.009
Users Full Coverage	77.64%	77.64%	77.64%	**100.00%**	**100.00%**	**100.00%**
Users Partial Coverage	22.36%	22.36%	22.36%	0.00%	0.00%	0.00%
– Avg. num. of rec.	3.49	3.49	3.49	-	-	-
Items coverage	0.83%	**1.44%**	0.91%	0.78%	0.58%	0.58%
– Proportion in Tail	0.00%	0.00%	0.00%	**6.31%**	6.17%	6.17%
– Proportion in Head	100.00%	100.00%	100.00%	**93.69%**	93.83%	93.83%
Computation time	0:17:00	0:17:00	11:40:00	5:30:00	5:27:00	5:11:00

Table 4.4: Social filtering results using different similarity measures: item-based support ($s > 13$), item-based confidence ($c > 0.1$), item-based asymmetric confidence ($\alpha = 0.0$), user-based asymmetric confidence ($\alpha = 0.0$), user-based confidence ($c > 0.001$) and user-based support ($s > 8$) respectively.

Perspectives

Perspectives

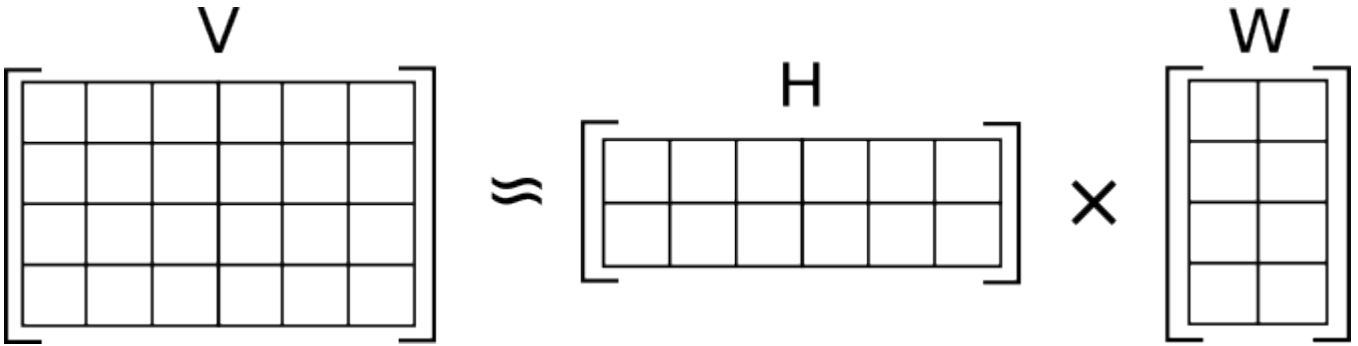
- Combine different algorithms with ensemble methods
- Combine different inputs, namely navigation interactions
- Integrate the time-dimension to the recommender system, e.g. switch from one algorithm to another as a function of t

References

- Ricci, F., Rokach, L., Shapira, B., *Introduction to recommender systems handbook*. Springer US. 2011.
- Bernardes D., Diaby, M., Fournier, F., Viennet, E., Soulié-Fogelman, F., *A Social Framework for Recommendation Systems*. Preprint. 2014.
- Aiolli, F. *Efficient Top-N Recommendation for Very Large Scale Binary Rated Datasets.* Proceedings of the ACM RecSys 2013.
- Lin, J. et al. *Addressing cold-start in app recommendation: latent user models constructed from twitter followers*. ACM SIGIR 2013.
- Wang, C., Blei, D., *Collaborative topic modeling for recommending scientific articles*, KDD 2011.

Collaborative filtering: model-based

- Matrix factorization techniques (SVD, NMF, …) to discover latent properties from users and items.



- Good results (particularly for rating prediction), but relative slower and more costly in memory.

Evaluation on literature datasets

Standard datasets from the literature

Dataset	Preferences	Users	Items	Explicit Social
LastFM	92,834	1,892	17,632	25,434
MovieLens1M	1 M	6,040	3,883	–
Flixster	8.2 M	1 M	49,000	26.7 M
MSD	48 M	1.2 M	380,000	–

Table 3.1: Summary statistics from four popular datasets featured in the RS literature.

Designing and Improving a Recommender System