Recommender systems in industrial contexts: recent and future approaches

Frank Meyer
Franck.meyer@orange.com
Orange Labs

6 mars 2015 – CNAM Paris
Summary

- An industrial approach of R.S.
  - Industrial context of R.S. and implications
  - An example at Orange: Reperio

- What’s next?
  - New trends and new use cases
  - An example at Orange: VIPE

- Conclusion
Summary

- An industrial approach of R.S.
  - Industrial context of R.S. and implications
  - An example at Orange: Reperio

- What’s next?
  - New trends and new use cases
  - An example at Orange: VIPE

- Conclusion
Automatic recommender systems...

- Recommender Systems: systems recommending items (such as products) personalized to users

- Many applications
  - product recommendation while browsing in a catalog
  - prediction of interest scores for films, broadcasts, books, music, travel, etc.

- Perceived by customers as value-added services
  - replicates the experience of local stores (DVD renter, bookseller, ...) who knows the customer and provides personalized advice
  - helps users discover new products of interest within huge content catalogs
  - based on trust in the relationship (as opposed to advertising or spam)

- Many needs in Orange Group
  - For IPTV live and Video on demand services (Compass)
  - Radio recommendation (Reperio)
  - Book recommendation (Reperio)
  - Mooc (massive open online course) recommendation, “Ma ville dans ma poche”… (Reperio / prototypes)
### Well-known example: Item-to-Items (Amazon™)

Customers Who Bought This Item Also Bought

<table>
<thead>
<tr>
<th>Item</th>
<th>Price</th>
<th>Rating</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice In Wonderland (Animation) ~ Special Edition [DVD]</td>
<td>£7.99</td>
<td>4.5</td>
<td>9</td>
</tr>
<tr>
<td>Beauty and the Beast [DVD] DVD ~ Paige O'Hara</td>
<td>£7.99</td>
<td>4.5</td>
<td>69</td>
</tr>
<tr>
<td>The Fox And The Hound [DVD] DVD ~ The Fox and The Hound</td>
<td>£5.99</td>
<td>4.5</td>
<td>10</td>
</tr>
<tr>
<td>The Aristocats Special Edition [DVD] DVD ~ Phil Harris</td>
<td>£7.99</td>
<td>4.5</td>
<td>42</td>
</tr>
</tbody>
</table>
Industrial Context…

What is a good recommendation according to marketers, project managers…?

- Increase in sales ++
  - More purchases on the catalog

- Audience ++
  - More clicks on the catalog

- User’s satisfaction ++
  - Easier ways to find good items
Implicit requirements 1/3

Usefulness

Useful recommendations!

Heuristic:
Find potential interesting items
Add diversity
# Implicit requirements 2/3

## Smart coverage

<table>
<thead>
<tr>
<th>Recommendations for...</th>
<th>Big Users</th>
<th>Small Users</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Short Head Items</strong></td>
<td>Not so useful…</td>
<td>Gain+</td>
</tr>
<tr>
<td>Best 20% of the catalog, in general</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Long Tail Items</strong></td>
<td>Gain+</td>
<td>Gain++</td>
</tr>
<tr>
<td>80% of the catalog, in general</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Recommendations for everything

Recommendations for everybody
### Implicit requirements 3/3

**The notion of impact**

<table>
<thead>
<tr>
<th>Impact of the recommendation</th>
<th>Impact if the user likes the item</th>
<th>Impact if the user dislikes the item</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recommending a popular item</td>
<td><strong>Low</strong>: the item is likely to be already known at least by name by the user.</td>
<td><strong>Low</strong>: even if the user dislikes this item he can understand that as a popular item this recommendation is likely to appear... at least at the beginning</td>
</tr>
<tr>
<td>Recommending an unpopular (infrequent) item</td>
<td><strong>High</strong>: the service provided by the recommender system is efficient. The rarest the item was, the less likely the user would have found it alone.</td>
<td><strong>High</strong>: not only the item was unknown and did not inspire confidence, but it also was not good.</td>
</tr>
</tbody>
</table>

We need good impact on the average...
AUC, RMSE, Precision@10 000? Precision@5!

- predicting scores of interest of several users for an item i: marketing campaign
- predicting scores of interest of a user u for several items, then selecting the best scored items: recommendation

Marketing campaign: Top 10 000

Recommendation: Top 5 !, max Top 10
Key functions

- **Help to Navigate (Explorer)**
  - Given an item \( i \) used as context, return \( N \) items similar to \( i \)

- **Help to Decide**
  - Given an user \( u \), and an item \( i \), return a predictive score of interest of \( u \) for \( i \) (a predicted rating).

- **Help to Compare**
  - Given a user \( u \) and a list of items \( i_1, \ldots, i_n \), sort the items in a decreasing order according to the relevance for \( u \).

- **Help to Discover**
  - Given a user \( u \), return \( N \) interesting items for \( u \).
## Other Industrial Requirements

<table>
<thead>
<tr>
<th>Requirements</th>
<th>Specification summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multi data sources</td>
<td>Deployment in different contexts with heterogeneous data</td>
</tr>
<tr>
<td>Cold Start</td>
<td>Ability to deal with few collaborative data, few usages, few feedbacks</td>
</tr>
<tr>
<td>Robustness</td>
<td>Ability to deal with noisy catalogs or logs</td>
</tr>
<tr>
<td>Scalability</td>
<td>Predictive models must be quickly computed even with huge database</td>
</tr>
<tr>
<td>Reactivity</td>
<td>Fast reaction for user’s feedback</td>
</tr>
<tr>
<td>Trusted relationship</td>
<td>Recommendations and scores easy to understand</td>
</tr>
<tr>
<td>Long tail management</td>
<td>Any item can be recommended, even those rarely viewed/bought/rated</td>
</tr>
</tbody>
</table>
Summary

■ An industrial approach of R.S.
  ■ Industrial context of R.S. and implications
  ■ An example at Orange: Reperio

■ What’s next?
  ■ New trends and new use cases
  ■ An example at Orange: VIPE

■ Conclusion
Reperio Recommender

- **Generic System**
  - One kernel, many recommendation types and modes
  - Two packages: Reperio-C (centralized) or Reperio-E (embedded)

- **Light System, high maintainability**
  - Full Reperio < 10,000 lines of source code

- **Multi-targeted framework**
  - From cluster of servers to simple Android Mobile

- **High scalability**
  - can deal with huge catalogs or huge log databases
Specified Core Functions

- **Help users to navigate**
  - Given an item $i$ used as a context, give $N$ similar items

- **Help users to decide**
  - Given a user $u$ and an item $i$, give a predictive interest score (rating)

- **Help users to compare**
  - Given a user $u$ and a list of items $i_1, \ldots, i_n$, sort the items in decreasing predicted interest

- **Help users to discover**
  - Given a user $u$, give $N$ interesting items for $u$
Interest of Item-item similarity matrix 1/3

Help users to navigate:
Item-to-item anonymous recommendation
Interest of Item-item similarity matrix 2/3

Help users to decide:
Rating prediction

\[
\hat{r}_{u} = \frac{\sum_{(i \in S_u)} \text{sim}(i, j) \times (r_{uj})}{\sum_{(j \in S_u)} \text{sim}(i, j)}
\]
Interest of Item-item similarity matrix 3/3

Profile of user u: $S_u$

- Rocky 3: 2/5
- Star Wars: 5/5
- E.T.: 3/5
- Toy Story: 4/5

Step 1: Seed selection
Possible short-term / long-term profile management

Step 2: Candidate generation
Possible short head / long tail management

Step 3: Known items removing

Step 4: Ranking

Step 5: Randomized Top N selection from the Top N*D list
D is a diversity factor

Help users Discover:
Recommending items

Profile of user u: $S_u$

- Rocky 3: 2/5
- Star Wars: 5/5
- E.T.: 3/5
- Toy Story: 4/5

Star Wars: 5/5
Toy Story: 4/5

In this example D=2 and N=3: Reperio randomly chooses 3 items among the top 3x2=6 best ranked items, so “Bambi” is selected whereas previously ranked in 5th position
Generic: Object-object similarity matrix 1/2

Recommend items based on different sources of data
Recommend not only items, but also users, descriptors
Generic: Object-object similarity matrix 2/2

Recommend items based on different sources of data
Recommend not only items, but also users, descriptors
Summary

- An industrial approach of R.S.
  - Industrial context of R.S. and implications
  - An example at Orange: Reperio

- What’s next?
  - New trends and new use cases
  - An example at Orange: VIPE

- Conclusion
New trends/Use cases

- Recommending new types of items
  - conversation threads, experts,…

- Recommending for new usages
  - crimes for instance

- Using new frameworks
  - Big Data / Fast Data
COBOT (2011)

- Recommends web pages from web search engine, users and conversations in a domain-specific community (health)
- (Sahay and Ram, 2011), Georgia Institute of Technology, Georgia, USA
- Data: Concept and topics extracted from conversational via a medical ontology + Social network of the conversations (on the website) + explicit feedback of users: “I like...”

Functionalities
- Discover: recommends: experts, web pages, and live conversations

Technology
- Hybrid content-based + social (using graph of interactions of the conversations)
  - Natural language processing + domain specific medical ontology to extract concept from the conversations
  - short-term profiling and long-term profiling of the user
  - case-based reasoning for request (to an external search engine) for web page recommendations

Innovative points
- real time recommendation integrated within a conversational web interface,
- recommendation of conversation threads,
- recommendation of expert users.
Inside COBOT (2011)

- **Modelling**
  - Hybrid content-based + social (using graph of interactions of the conversations)
    - Natural language processing + domain specific medical ontology to extract concept from the conversations
    - The medical ontology used is Unified Medical Language System (UMLS™) + WordNet
    - short-term profiling and long-term profiling of the user (the window size for the short term profile is few days)
    - case-based reasoning for request (to an external search engine) for web page recommendations; the system also stores the users’ feedback to each result returned as a recommendation (for reusability of the queries)

- **Evaluation**
  - not evaluated but was available online

- **Remark**
  - also used in an open social learning community: OpenStudy
COBOT (2011)

Realtime Application Server

- Text Analysis & Processing Engine (TAPE) (Classify, Parse, Extract)
- Web Search & Ranking Engine (WebScour) (Find, Rank)
- Case based reasoning Engine (Converse) (Re-find from past)
- User Modeling & Recommendation Engine (Uvolve) (Learn Profiles, Match)
Crime Walker (2011)

- Recommends... Crime suspects
- (Tayebi et al., 2011), University British Colombia, Canada

Data
- from an hypergraph (set of sets) of
  - actors: victims, offenders, witnesses
  - events: crimes (typology)
  - resources: mobile phones, vehicles, weapons, bank account...
- a co-offending network
  - offenders who have committed crimes together (resources not used for the moment)

Functionalities
- Help to Discover, Help to Compare (rank) for k suspects

Technology
- collaborative (build-in similarity + random walk)

Innovative points
- specific field
- Also used to predict co-authorship in scientific publications
Inside Crime Walker (2011)

- **Modelling**
  - Use random walk technique
    - define a similarity measure between offenders (using age, sex, living location)
    - start with a first offender $u$
    - perform random walks on the graph of similar offenders and compute the frequency of each offender seen on this graph
    - return the Top-$N$ co-offenders based on the computed frequencies

  - Applied random walk to a set of already charged offenders
    - to rank them and to possibly add new suspects

- **Evaluation**
  - offline, using Train/Test protocol
  - BC Co-offending Network: data of 5 years of real-life crime data (very sensitive so not available for public)
  - also used on DBLP co-authorship network data (scientific publications, free data)
  - For offender prediction: for top 10 suspects, recall is up to 7.8% (association rules: 5%)
    - For the moment, it can help criminologists, but it cannot be used instead of inquirers
  - For co-authorship data: for top 10 “potential co-authors”, recall is up to 66% (association rules: 25%)
Crime Walker (2011)

Figure 1: Hyperedge in the crime data model
NETFLIX (2005 - 2012)

- **Recommends**
  - DVD and Video On Demand
- **Amatriain, X. (2012). Netflix R&D, California, US.**

- **Data**
  - all the available users’ feedbacks (both implicit and explicit): browse, play, rate, search, taste preferences...

- **Functionalities**
  - Help to Decide: was the main goal of the Netflix’s challenge (2006-2009), in fact less impact now
  - Help to Discover: personalized Top N lists of recommended items (using mainly Help to Rank as sub function, and diversity parameters)
  - Help to Navigate: similar items in context

- **Technology**
  - Many algorithms are used: mainly SVD, Boltzmann Machines, Bayesian/Neural networks
  - Both online and offline approaches

- **Innovative points**
  - “Everything is recommendation” point of view
  - Big Data oriented framework (25 millions+ users, 5 billions+ ratings)
Inside NETFLIX: a big data framework

**Event and data distribution**
- non real time data flow managed by Hadoop
- real time data flow managed by internal tool (Manhattan)

**Offline jobs**
- model training
- computation of models
- computation of intermediate results

**Computation**
- done both online and offline
- Both online and offline algorithms

**Results**
- from previously computed lists, online algorithms, and combination of both
- intermediate results stored in Cache, in Cassandra DB, in relational DB
NETFLIX (2005-2011)

Everything is personalized

Note: Recommendations are per household, not individual user

Ranking

Rows

Personalization awareness

Diversity

Orange R&D

Orange FT-group
Summary

- An industrial approach of R.S.
  - Industrial context of R.S. and implications
  - An example at Orange: Reperio

- What’s next?
  - New trends and new use cases
  - An example at Orange: VIPE

- Conclusion
Conclusion

■ R.S. are multi-facetted
  ■ many requirements to consider
  ■ the requirements have an impact on the design of engines and algorithms
  ■ do not underestimate the problem of the evaluation

■ R.S. engines are multi-purpose
  ■ they are designed to deal with huge matrices, with multi-targets, with real time learning,…
  ■ so there are many opportunities for new use cases, new applications,…

■ R.S. are still at the cutting edge of technology
  ■ close to the big data / fast data ecosystems
  ■ one of the mains concerns of many Internet’s leading player
KNN Modelling

- **Item-item K-Nearest Neighbors (KNN, item-based)**
  - **Basic idea**
    - a user interested by an item $i$ may be interested by items similar to $i$ (warning: true for collaborative mode, not always true for content-based mode)
  - **Process**
    - compare the items together
    - for each item, select $K$ most similar items
    - the similarity can be based on items’ characteristics (content-based) or on items’ correlation of usages (collaborative)

- **User-user K-Nearest Neighbors (KNN, user-based)**
  - **Basic idea**
    - if a user $u$ is similar to a user $v$, the user $u$ may be interested by the same items that interest the user $v$
  - **Process**
    - same principle as item-item
    - similarity based on users’ features (sociodemographics) or on users’ usages (collaborative)
Other classic models 1/2

- **Space model - keywords (content-based)**
  - **Basic idea**
    - a user profile is defined by a list of weighted keywords
    - an item may interest a user if its description contains keywords matching the user’s profile
  - **Process**
    - keywords of high rated items are reinforced (increasing weights)
    - keywords of low rated items are penalized (decreasing weights)
    - for each item: compute the average weight with respect to the user profile
    - note: often used with Term Frequency - Inverse Document Frequency (TF-IDF) principle

- **TF-IDF (pre-processing)**
  - Pretreatment of text data represented by a documents x keywords matrix
  - Basically:
    - for all document, penalizes too frequent keywords (the, is, be,...)
    - for each document, weight keywords according to their frequency
Other classic models 2/2

- **Matrix factorization**
  - **Basic idea**
    - approach based on matrix factorization
    - a user profile is defined by a vector $u$ of few factors
    - an item profile (characteristics) is defined by a vector $i$ of few factors
    - the predicted rating of interest of $u$ for $i$ is given by the scalar product $u \cdot i$
  - **Example with Gravity’s principle (fast matrix factorization)**

![Matrix factorization diagram]

$$R = P \times Q$$
Reperio Embedded

Reference Catalogs
For instance: TV programs, Movie Theater programs, events…

Catalog server

User Interface of the embedded application
The GUI uses Reperio as a Java API

Smart phones

soap or rest protocol

Catalog server

soap or rest protocol
Reperio Centralized (3-tiers)

Application server (using recommendation API) → Reperio-C on a server → Server for Reperio-C

Reperio-C on a server + Web Service

Web Service

Server for the service application using recommendation API

soap or rest protocol

HTTP protocol

PCs (end-user)

PC + web browser

soap or rest protocol

HTTP protocol
## Design choices for Reperio

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Design choice</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multi data sources</td>
<td>Generic sparse matrix structure, generic KNN algorithm</td>
</tr>
<tr>
<td>Cold Start</td>
<td>Help the collaborative filtering engine with: Content-based (thematic)</td>
</tr>
<tr>
<td></td>
<td>recommendation, users’ preferences</td>
</tr>
<tr>
<td>Robustness</td>
<td>Robust models: KNN + adapted similarity measure</td>
</tr>
<tr>
<td>Scalability</td>
<td>1) KNN + dimensionality reduction + parallelization</td>
</tr>
<tr>
<td></td>
<td>2) Evol. KNN based on Matrix Factorization</td>
</tr>
<tr>
<td>Reactivity</td>
<td>1) KNN + online rating prediction</td>
</tr>
<tr>
<td></td>
<td>2) Evol. Online Matrix Factorization</td>
</tr>
<tr>
<td>Trusted relationship</td>
<td>KNN-based recommendation generation: we recommend B because you watched A</td>
</tr>
<tr>
<td></td>
<td>and B is similar to A</td>
</tr>
<tr>
<td>Long tail management</td>
<td>Generate candidate items taking into account the catalog’s long tail</td>
</tr>
</tbody>
</table>
### Implementation issues

<table>
<thead>
<tr>
<th>Choice</th>
<th>Why?</th>
</tr>
</thead>
<tbody>
<tr>
<td>K Nearest Neighbor models</td>
<td>Item-item similarity matrix mandatory, good performance for modeling, recommendations easy to explain</td>
</tr>
<tr>
<td>Language: Java</td>
<td>Good performance, very easy to deploy on servers or on embedded devices</td>
</tr>
<tr>
<td>MySQL or PostgreSQL for server</td>
<td>Open source DBMS as Reperio is full open-source</td>
</tr>
<tr>
<td>SQLite for mobile devices</td>
<td>Limited resources on the embedded version, web service useless in the embedded case</td>
</tr>
<tr>
<td>2 branches</td>
<td>Opportunity to test an embedded recommender agent on a widely used device</td>
</tr>
<tr>
<td>Reperio-E: Embedded (light version)</td>
<td></td>
</tr>
<tr>
<td>Reperio-C: Centralized (webService)</td>
<td></td>
</tr>
<tr>
<td>Android OS for the embedded version</td>
<td></td>
</tr>
</tbody>
</table>
Implementation of Core Functions

- Help to navigate (item-to-items recommendation)
  - build an item-item similarity matrix (batch process). Online, during the display of an item, display at the same time similar items.

- Personalized predictive rating of a target item (predictive scoring)
  - Online comparison of the target item with the user’s profile using a item-item similarity matrix

- Personalized Ranking (for personalized search)
  - Personalized scoring of a list of items from a catalog or from the result of a search request

- Push of Recommendations (daily, weekly)
  - Search for items similar to those already well rated by the user (process done daily or weekly). Access via a personal page or by mail.
Reperio C-V5

- Centralized mode, example of a movie recommender
Reperio / IT&Labs

- Address: http://reperio-ext.aql.fr/reperio-demo/
Reperio E-V2

- **Embedded Mode, example of a TV program recommender**
Principle or performances evaluation

- **Learning set and Test set**
  - Learning set: many triples (userID, itemID, rating)
  - Test set: other triples (userID, itemID, rating), but with userID and itemID already seen in the Learning set (so items and user can be modeled)
    - Build a predictive model using only the LearningSet
    - Evaluation done using the test set: for each

- **Classic Measure: the Root Mean Squared Error**
  - Root Mean Squared Error:
    \[
    RMSE = \sqrt{\frac{1}{|R|} \sum_{(u,i,r) \in R} (\hat{r}_{ui} - r_{ui})^2}
    \]

- **Other performance indicators**
  - Precision/recall (but recall is not well defined for recommendation)
  - Model building time, model scoring time
## Accuracy…

<table>
<thead>
<tr>
<th>System</th>
<th>Netflix's RMSE (cross validation or probe test)</th>
<th>Ref.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reperio</td>
<td>0.87</td>
<td>Candillier, et al., 2008</td>
</tr>
<tr>
<td>Best KNN model</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Typical good SVD-based model</td>
<td>0.90</td>
<td>Jaher et al., 2010</td>
</tr>
<tr>
<td>Typical good KNN-base model</td>
<td>0.89</td>
<td>Jaher et al., 2010</td>
</tr>
<tr>
<td>Cinematch</td>
<td>0.95</td>
<td>Netflix's challenge: <a href="http://www.netflixprize.com">www.netflixprize.com</a></td>
</tr>
<tr>
<td>Netflix's System before the Netflix’s Challenge</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Ref.:**

Jaher, M., Toscher, and Legenstein, R. Combining predictions for accurate recommender systems. KDD 2010 (pp 693-702).


**Tests in centralized mode**

Rating prediction task: accuracy, measured by RMSE (the lower, the better)

Learning set (90%) / Test Set (10%) protocol with Netflix’s logs
### Scalability 1/3

**Speed of Reperio-E on smart phone:**

*Android™ mobile, processor: 32 bits ~500Mhz, ~300 Mo ram*

<table>
<thead>
<tr>
<th>Time slice (about 100 items)</th>
<th>~ 0.08s</th>
<th>~1 250 similarities /s</th>
</tr>
</thead>
<tbody>
<tr>
<td>On 1 Day (about 1600 items)</td>
<td>~1.2s</td>
<td>~1 300 similarities /s</td>
</tr>
</tbody>
</table>

For a target TV program i, find 5 similar TV programs
In the same time slice, or In the same day
# Scalability 2/3

**Speed of Reperio-E on smart phone:**
Android™ OS, processor: 32 bits ~500Mhz, ~300 Mo ram

<table>
<thead>
<tr>
<th>Time slice</th>
<th>Time</th>
<th>Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>(about 100 items)</td>
<td>~ 2.2s</td>
<td>~ 45 scores /s</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>On 1 Day</th>
<th>Time</th>
<th>Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>(about 1600 items)</td>
<td>~ 35.3s</td>
<td>~ 46 scores /s</td>
</tr>
</tbody>
</table>

For a time slice, or for a day, find the 5 programs with the best predicted ratings
Scalability 3/3

Speed of Reperio-C on huge DBMS:
PC 64 bits 2-core 3.4Ghz, 32 Go ram

<table>
<thead>
<tr>
<th></th>
<th>Time</th>
<th>Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning time of a</td>
<td>~ 2 h 30</td>
<td>~17 500 similarities /s</td>
</tr>
<tr>
<td>item-item model for</td>
<td></td>
<td></td>
</tr>
<tr>
<td>90 million logs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scoring time for</td>
<td>~1 minute</td>
<td>~166 000 scores /s</td>
</tr>
<tr>
<td>10 millions logs</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Compute a similarity matrix for 17,770 items in dimension 480,000 on 90 millions logs…
Then predict 10 millions ratings
## Embedded or centralized working modes 1/2

<table>
<thead>
<tr>
<th></th>
<th>Centralized native mode</th>
<th>Centralized virtual mode</th>
<th>Embedded mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Typical example of item dataset</td>
<td>Video On Demand catalog</td>
<td>Cultural products on e-commerce website</td>
<td>Electronic Program Guides (TV programs)</td>
</tr>
<tr>
<td>Size of item dataset</td>
<td>Large</td>
<td>Very large</td>
<td>Small</td>
</tr>
<tr>
<td>Catalog's update frequency</td>
<td>Low</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>Typical adapted filtering mode</td>
<td>Collaborative</td>
<td>Collaborative</td>
<td>Content-based</td>
</tr>
<tr>
<td>Size of items' description (metadata or ratings)</td>
<td>Large, several hundreds of ratings</td>
<td>Very large, several thousands of ratings</td>
<td>Small, several tens of metadata</td>
</tr>
</tbody>
</table>
## Embedded or centralized working modes 2/2

<table>
<thead>
<tr>
<th></th>
<th>Centralized native mode</th>
<th>Centralized virtual mode</th>
<th>Embedded mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reperio's similarity computation mode</td>
<td>Pre-computed into a similarity matrix, on native data. The KNN of each item are stored.</td>
<td>Online based on a matrix factor decomposition (SVD)</td>
<td>On the fly, on native data</td>
</tr>
<tr>
<td>Pre-computed model used</td>
<td>Pre-computed similarity matrix</td>
<td>Matrix factor decomposition</td>
<td>No pre-computed model</td>
</tr>
<tr>
<td>Adapted Similarity used</td>
<td>Extended Pearson</td>
<td>Pearson</td>
<td>Jaccard using attribute weightings</td>
</tr>
<tr>
<td>Typical Item-to-item function implementation</td>
<td>Exact K nearest neighbors extract from the similarity matrix</td>
<td>Approximate K nearest neighbors by random sampling</td>
<td>Restricted to items explicitly selected as interesting, or within a time-slice.</td>
</tr>
</tbody>
</table>