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)

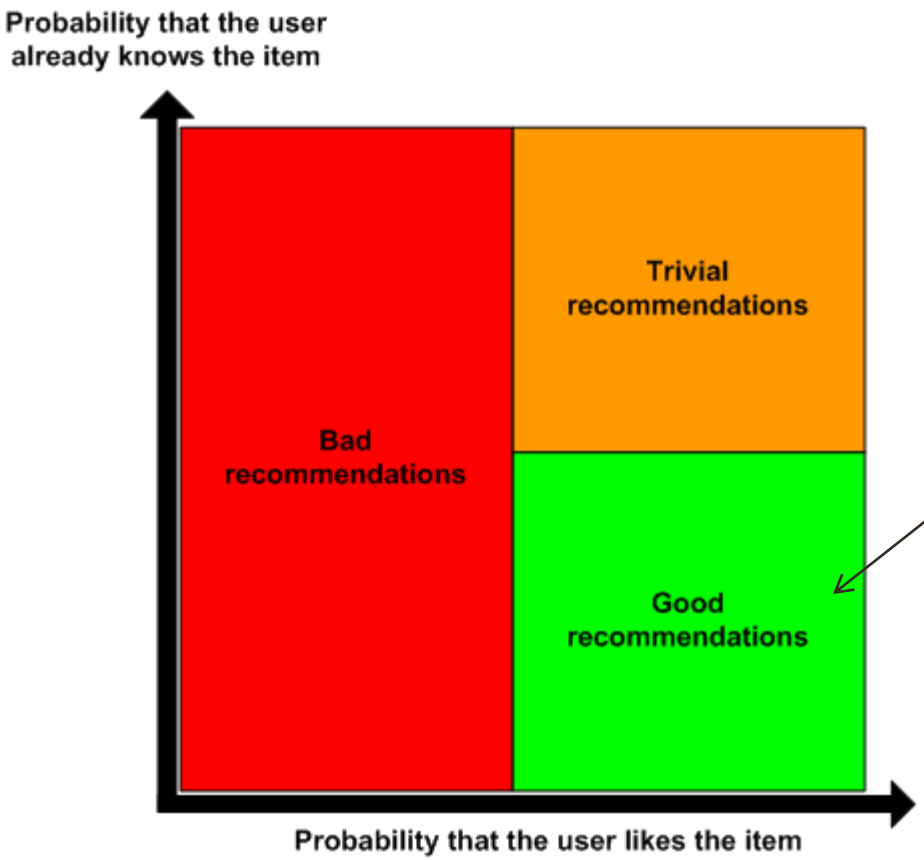
- Many needs in Orange Group
  - For IPTV live and Video on demand services (Compass)
  - Radio recommendation (Reperio)
  - Book recommendation (Reperio)
  - Mooc (massive open online course) recommendation, “Ma ville dans ma poche”… (Reperio / prototypes)

Well-known example: Item-to-Items (Amazon™)

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

Recommendations for…	**Big Users**	**Small Users**
Short Head Items Best 20% of the catalog, in general	Not so useful…	Gain+
Long Tail Items 80% of the catalog, in general	Gain+	Gain++

Recommendations for everything

Recommendations for everybody

Implicit requirements 3/3
The notion of impact

	Impact of the recommendation	
	Impact if the user likes the item	**Impact if the user dislikes the item**
Recommending a popular item	**Low**: the item is likely to be already known at least by name by the user.	**Low**: 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
Recommending an unpopular (infrequent) item	**High**: the service provided by the recommender system is efficient. The rarest the item was, the less likely the user would have found it alone.	**High**: not only the item was unknown and did not inspire confidence, but it also was not good.

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

	Data silo Mobile						Data silo Fixed-line						Data silo VOD						Data silo TV live					Data silo facebook		
	Mobile:abonn.	Mobile: option1	Mobile : premium option	Mobile: option world	Mobile:smartphone Android	Mobile:smartphone Iphone	Fixed-line:option1	Fixed-line:option2	Fixed-line: zip area	Fixed-line:Age [20-30]	Fixed-line:Age [30-40]	Fixed-line: socio +	VOD: likes Star Wars	VOD: likes Titanic	VOD : likes Peter Pan	VOD : likes Rio Grande	VOD : likes Démineur	VOD : likes cccc	TV : likes Foot	TV : likes Dr House	TV : likes Game of Thrones	TV : likes Thalassa	TV : likes nouvelle Star	likes: detective	Likes: London	Likes: Jazz
Univers 1 :user 1		1								1			1								1					
Univers 1 :user																										
Univers 1 :user					1																1					
Univers 1 :user																										
Univers1 :user m			1									1														
Univers k :user 1											1															
Univers k :user		1																								
Univers k :user																1						1			1	
Univers k :user							1					1														
Univers k :user n					1										1											

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**

Exemple:

amazon.com

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

Exemple:

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

Exemple:

YouTube

- **Help to Discover**
  - Given a user **u**, return **N** interesting items for **u**.

Exemple:

Google News

Other Industrial Requirements

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

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

Example:

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

Example:

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

Examples:

- **Help users to discover**
  - Given a user **u**, give **N** interesting items for **u**

Examples:

Interest of Item-item similarity matrix 1/3

Item-item similarity Matrix (%)

	Avatar	Toy Story	Star Wars	Finding Nemo	Up
Avatar		0.25	0.90	0.15	0.17
Toy Story			0.30	0.95	0.91
Star Wars				0.10	0.9
Finding Nemo					
(…)					

1. Item used as context (during catalog browsing for instance)

2. Similar items recommended

Help users to navigate:
Item-to-item anonymous recommendation

Interest of Item-item similarity matrix 2/3

Item-item similarity Matrix on movies

	Avatar	Rocky 3	Star Wars	Titanic
Avatar		0.10	0.90	0.60
Rocky 3			0.40	0.15
Star Wars				0.55
Titanic				

Profile of user u: S_u

Rocky 3	Star Wars	Titanic
2/5	5/5	3/5

?

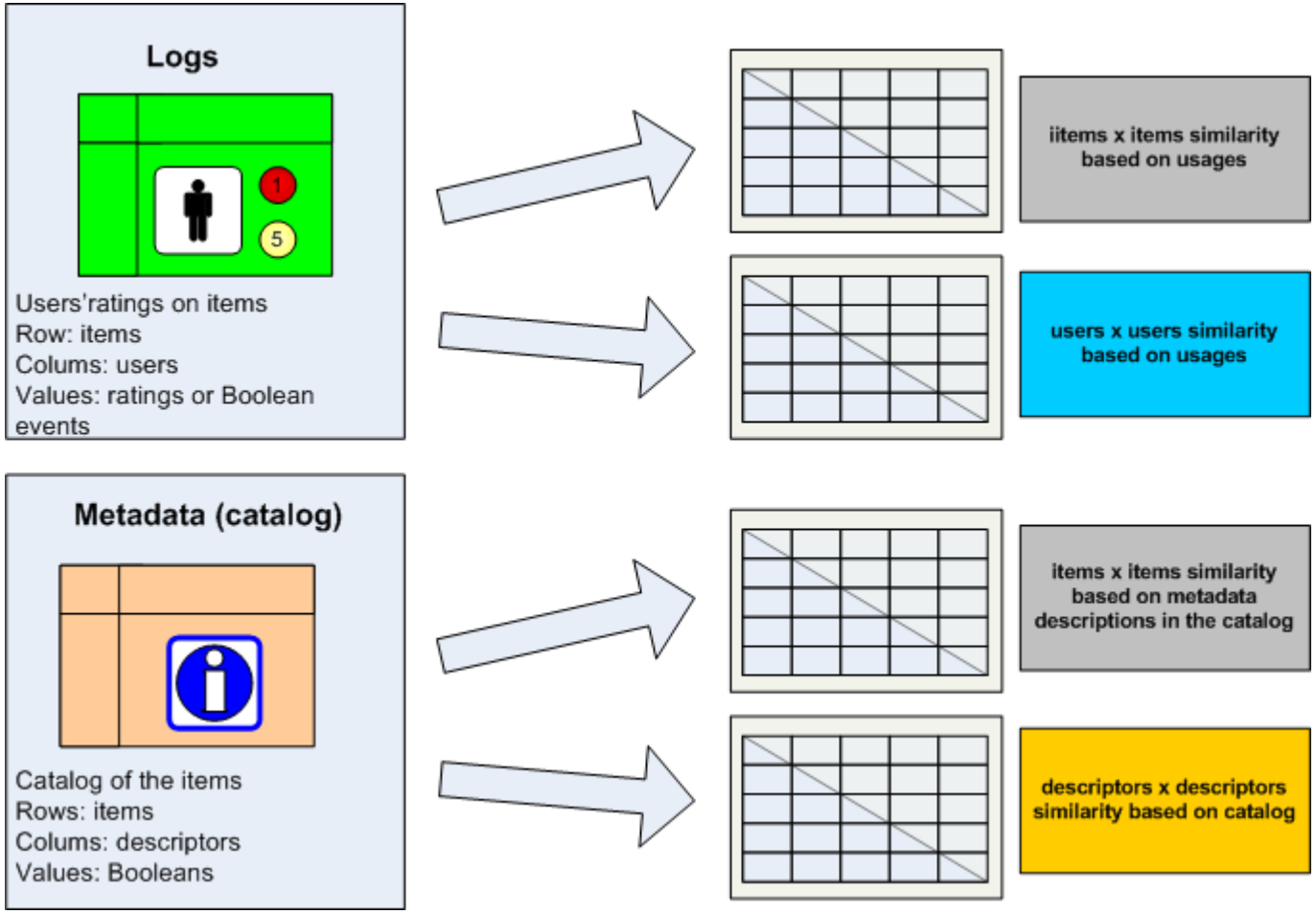
$$\hat{r}_{ui} = \frac{\sum_{\{j \in S_u\}} sim(i,j) \times (r_{uj})}{\sum_{\{j \in S_u\}} |sim(i,j)|}$$

Help users to decide:
Rating prediction

Interest of Item-item similarity matrix 3/3

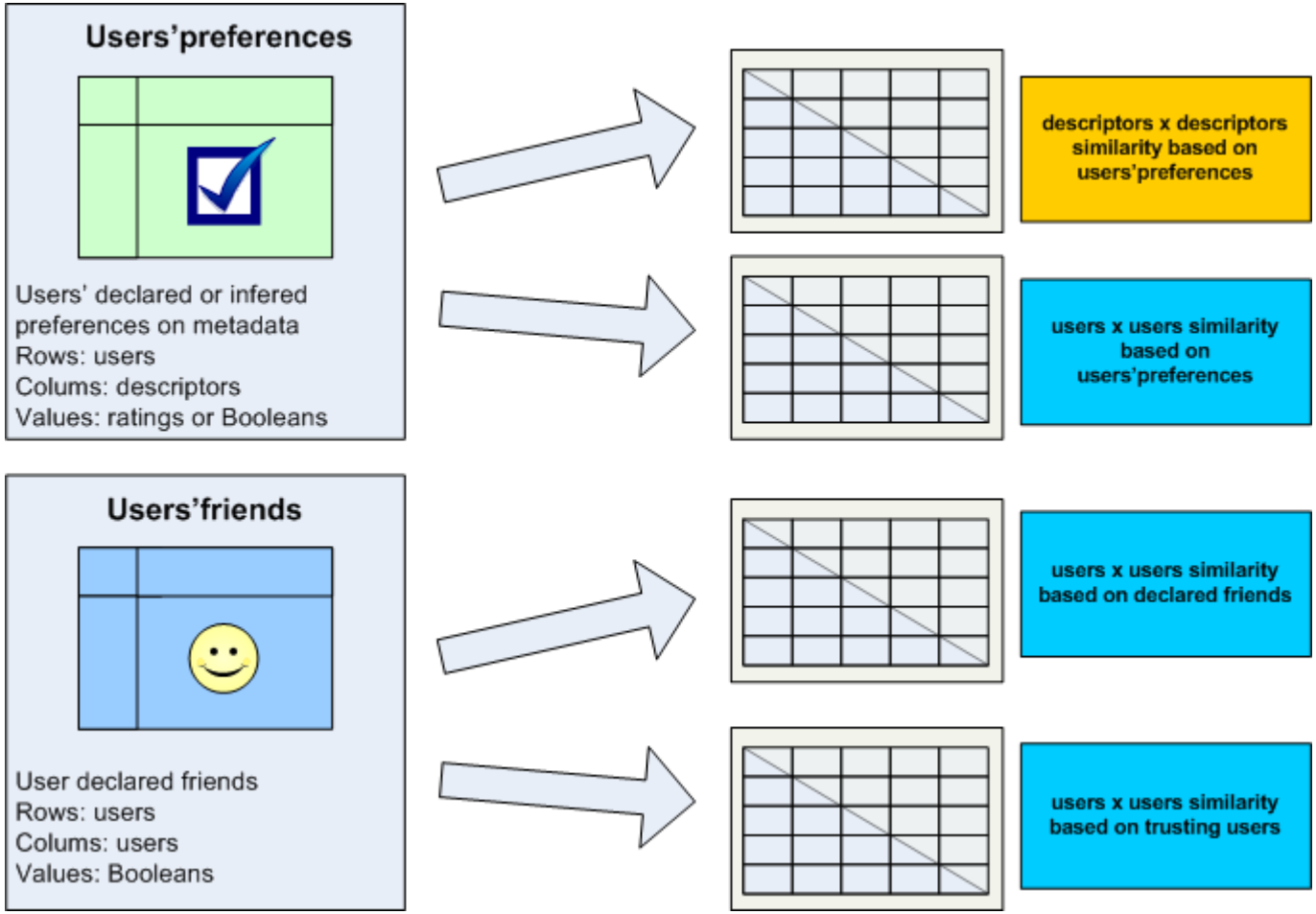
Help users Discover: Recommending items

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.

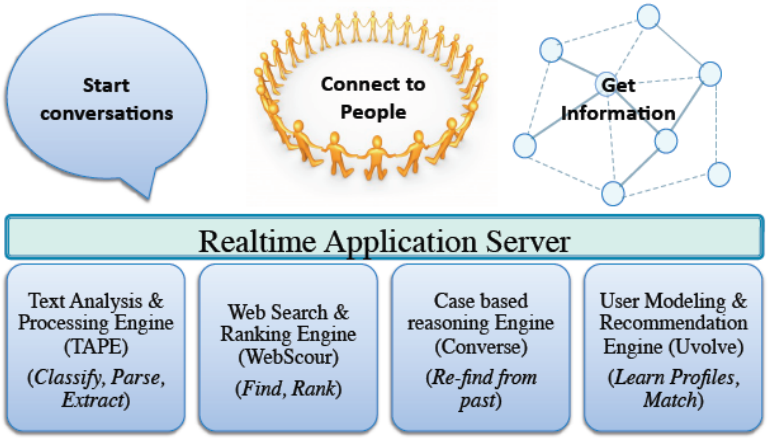
Inside COBOT (2011)

- 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)


Start conversations
Connect to People
Get Information
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)


Connect GT Community
Real Time Conversations
GT Focused Web Resources
Recommend people in GT community with similar health interests
Have conversations
Invite other people to join
Recommend related Web resources based on what users are talking about

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
  - offine, using Train/Test protocol
  - BC Co-effending 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)

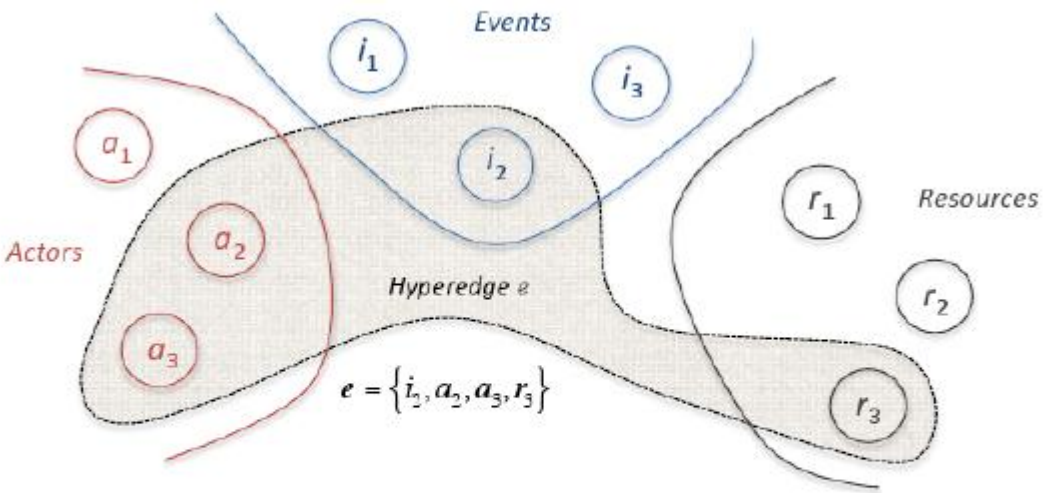


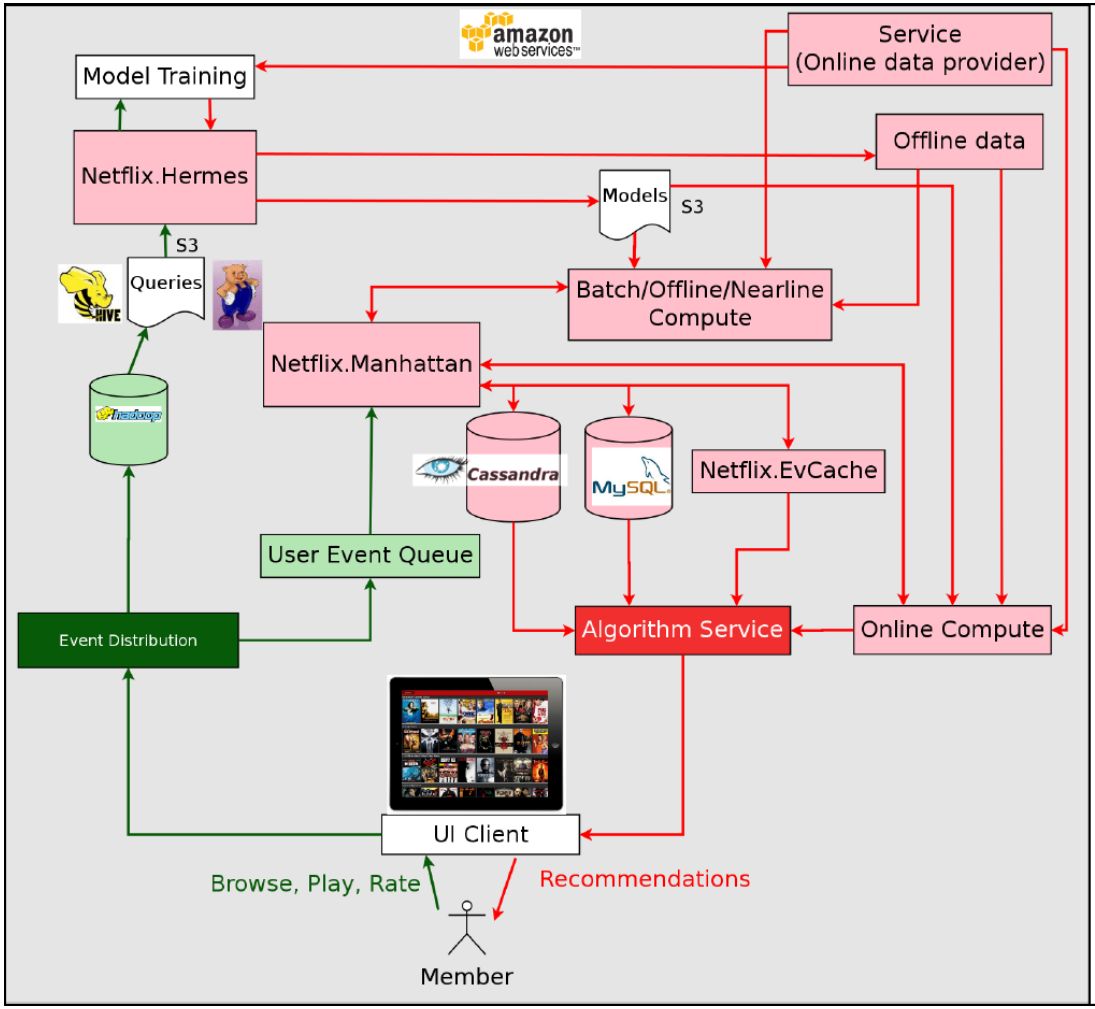
Figure 1: Hyperedge in the crime data model

NETFLIX (2005 - 2012)

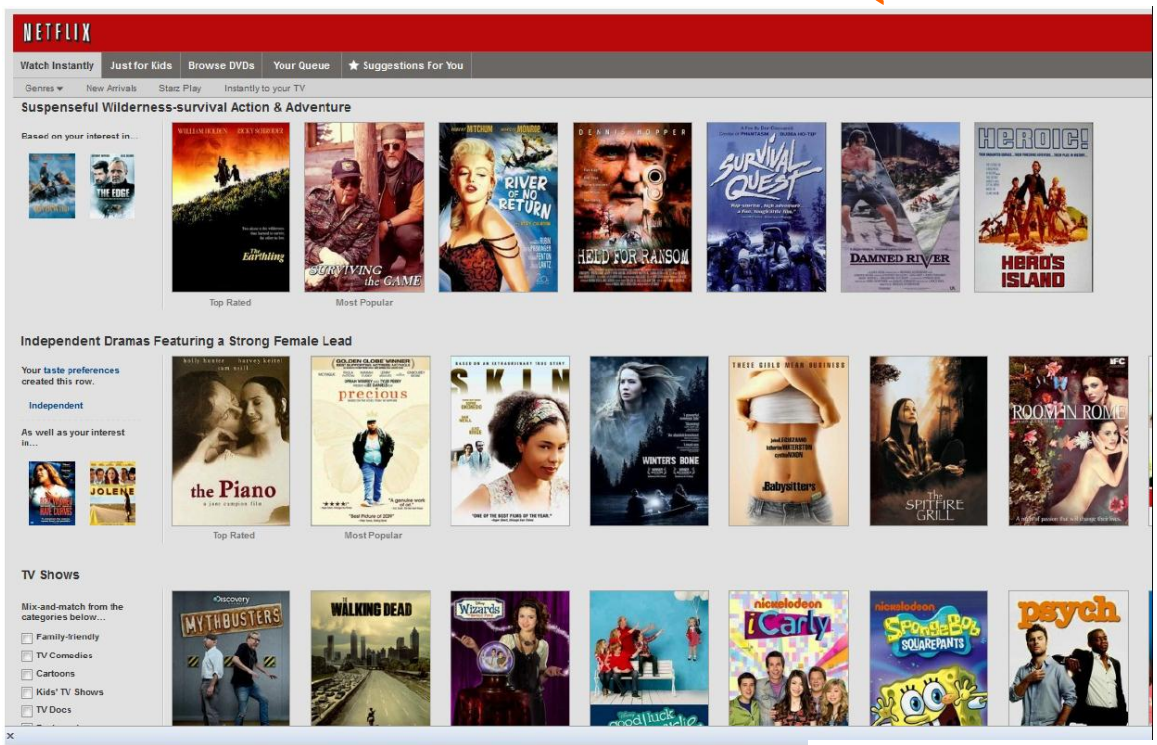
- Recommends
  - DVD and Video On Demand
- Amatriain, X. (2012). Netflix R&D, California, US.
- Data
  - all the available users' feedbacks (both implicit and explicit): browse, play, rate, search, taste preferences...
- Functionalities
  - Help to Decide: was the main goal of the Netflix's challenge (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)



Note:
Recommendations are per household, not individual user

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

Annexes

KNN Modelling

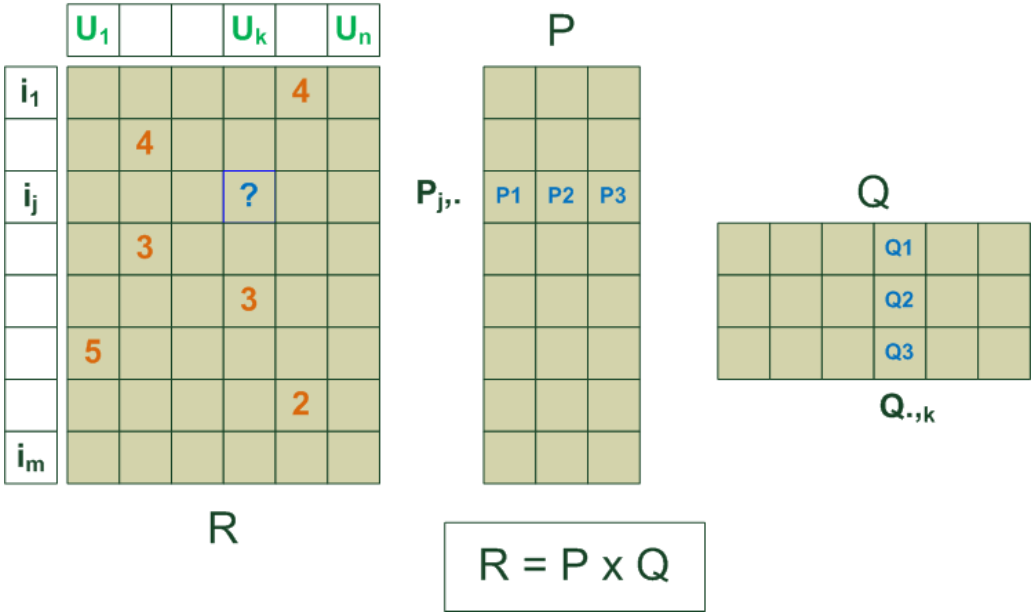
- **Item-item K-Nearest Neighbors (KNN, item-based)**
  - Basic idea
    - a user interested by an item **i** may be interested by items similar to **i** (warning: true for collaborative mode, not always true for content-based mode)
  - Process
    - compare the items together
    - for each item, select **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

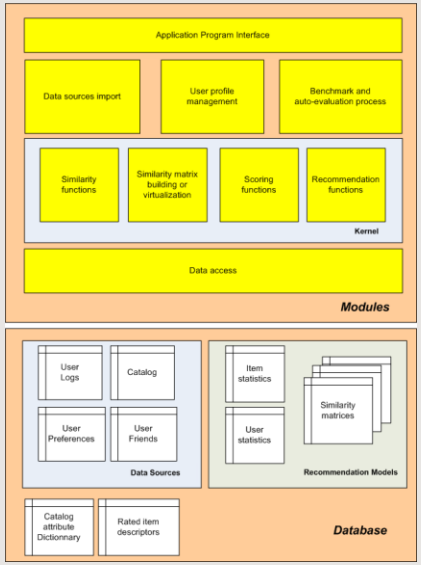
- **Space model - keywords (content-based)**
  - Basic idea
    - a user profile is defined by a list of weighted keywords
    - an item may interest a user if its description contains keywords matching the user's profile
  - Process
    - keywords of high rated items are reinforced (increasing weights)
    - 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.i**
  - Example with Gravity's principle (fast matrix factorization)



Reperio Embedded



Reperio-E on smartphone

+ Catalog synchronization module

(embedded mode)

User Interface of the embedded application

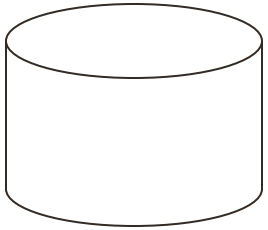
The GUI uses Reperio as a Java API

Smart phones

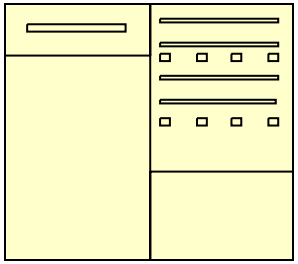
soap or rest protocol

soap or rest protocol

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

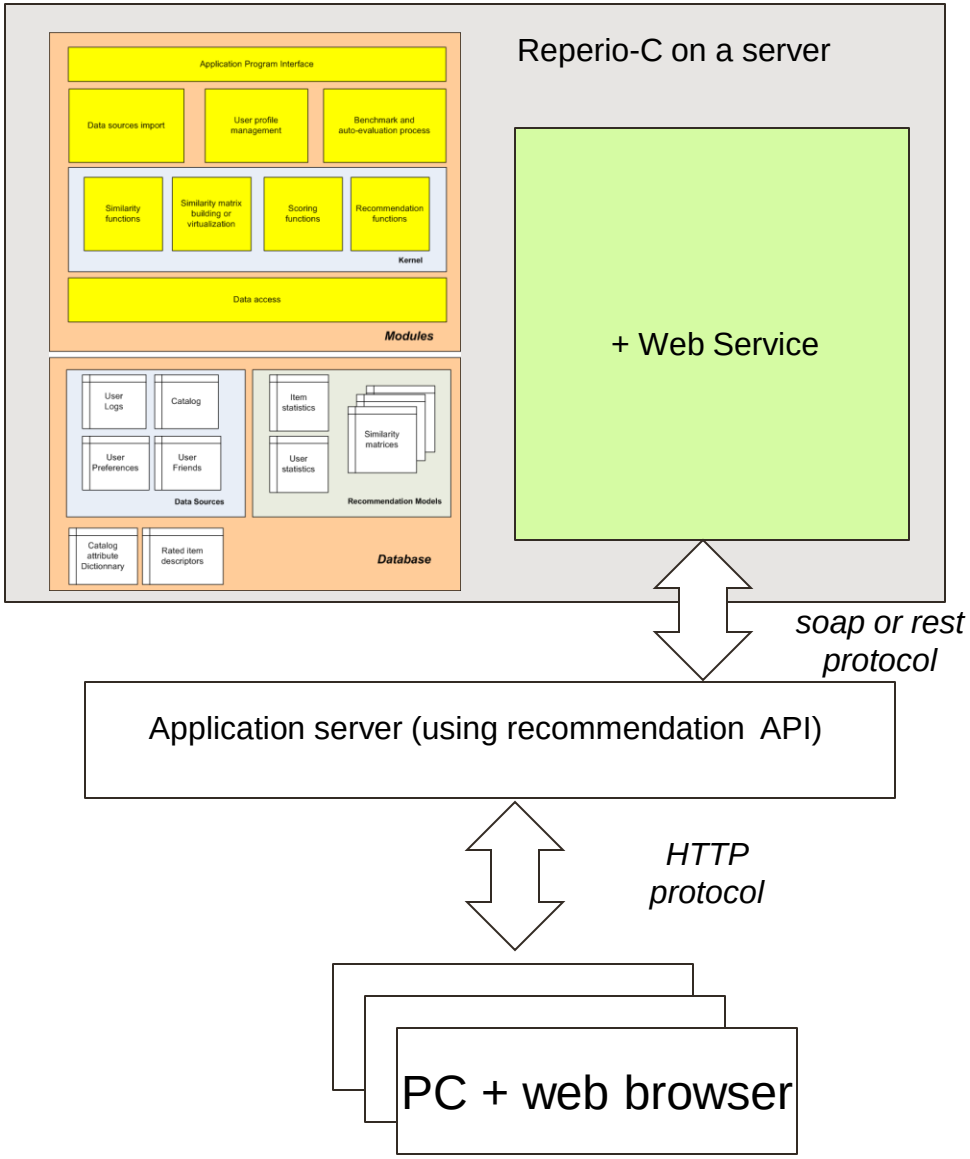


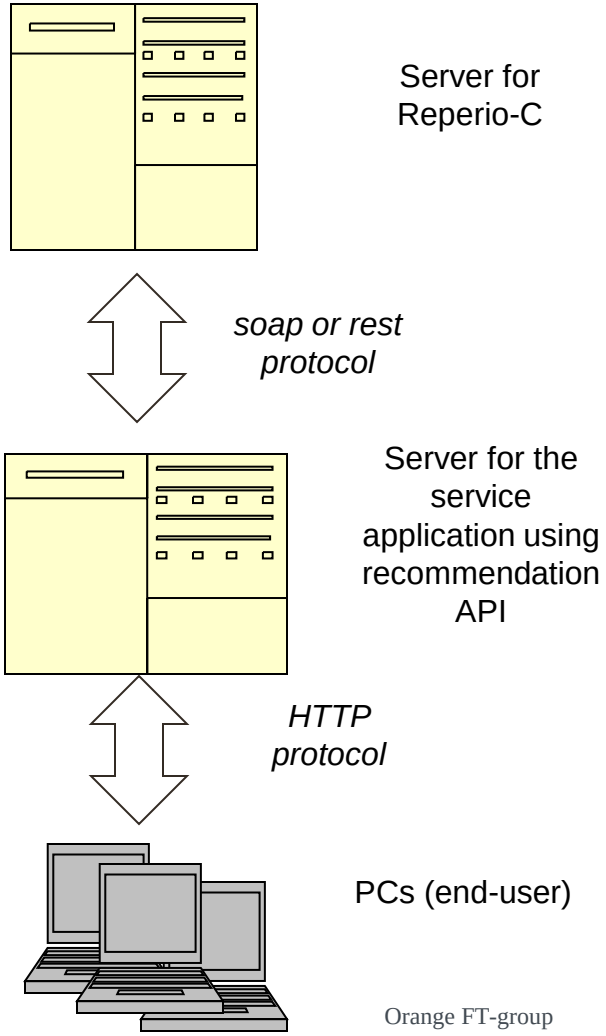
Catalog server



Catalog server

Reperio Centralized (3-tiers)





Design choices for Reperio

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

Implementation issues

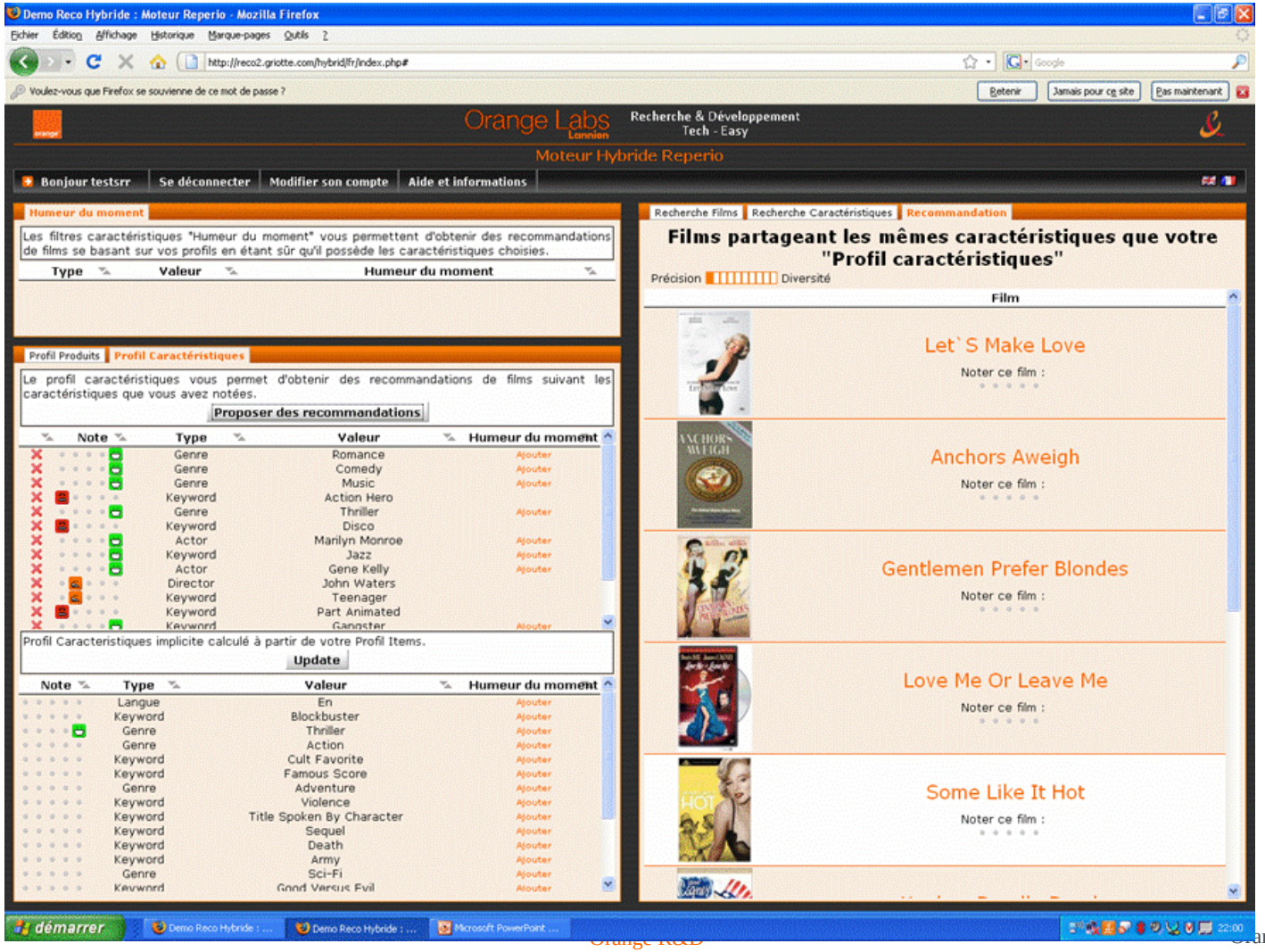
Choice	Why?
K Nearest Neighbor models	Item-item similarity matrix mandatory, good performance for modeling, recommendations easy to explain
Language: Java	Good performance, very easy to deploy on servers or on embedded devices
MySQL or PostgreSQL for server SQlite for mobile devices	Open source DBMS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

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.

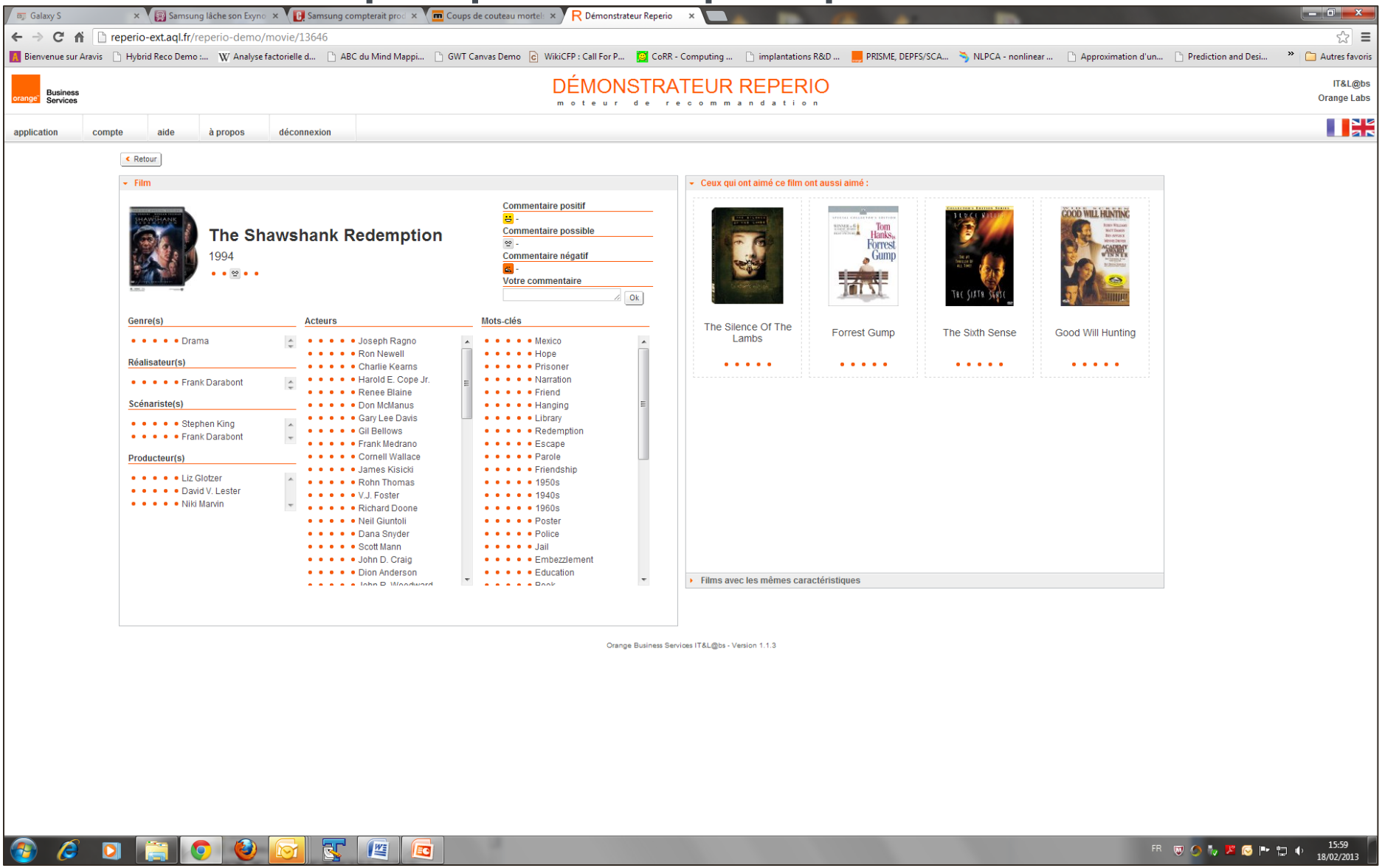
Reperio C-V5

- **Centralized mode, example of a movie recommender**



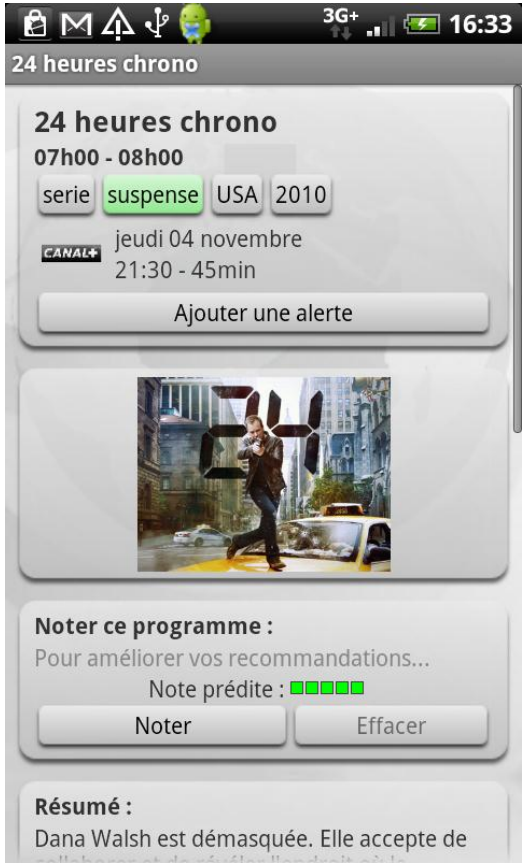
Reperio / IT&Labs

- **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:

$$RMSE = \sqrt{\frac{1}{|R|}\sum_{(u,i,r)\in R}\left(\hat{r}_{u,i} - r_{u,i}\right)^2}$$

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

Accuracy…

System	Netflix's RMSE (cross validation or probe test)	Ref.
Reperio Best KNN model	0.87	Candillier, et al., 2008
Typical good SVD-based model	0.90	Jaher et al., 2010
Typical good KNN-base model	0.89	Jaher et al., 2010
Cinematch Netflix's System before the Netflix's Challenge	0.95	Netflix' challenge: *www.netflixprize.com*

Ref.:

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

Candillier, L., Meyer, F., Fessant, F. (2008). Designing Specific Weighted Similarity Measures to Improve Collaborative Filtering Systems. ICDM 2008: 242-255

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

	Time	Speed
Time slice (about 100 items)	~ 0.08s	~1 250 similarities /s
On 1 Day (about 1600 items)	~1.2s	~1 300 similarities /s

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

	Time	Speed
Time slice (about 100 items)	~ 2.2s	~ 45 scores /s
On 1 Day (about 1600 items)	~ 35.3s	~ 46 scores /s

background process running during catalog synchronisation

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

	Time	Speed
Learning time of a item-item model for 90 million logs	~ 2 h 30	~17 500 similarities /s
Scoring time for 10 millions logs	~1 minute	~166 000 scores /s

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

	Centralized native mode	**Centralized virtual mode**	**Embedded mode**
Typical example of item dataset	Video On Demand catalog	Cultural products on e-commerce website	Electronic Program Guides (TV programs)
Size of item dataset	Large	Very large	Small
Catalog's update frequency	Low	Low	High
Typical adapted filtering mode	Collaborative	Collaborative	Content-based
Size of items' description (metadata or ratings)	Large, several hundreds of ratings	Very large, several thousands of ratings	Small, several tens of metadata

Embedded or centralized working modes 2/2

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