Recommendation Systems and Deep Learning

Inna Skarga-Bandurova
Computer Science and Engineering Depatrment
V. Dahl East Ukrainian National University

Funded by the
Erasmus+ Programme
of the European Union

Kharkiv, 17-19 April 2018

Structure of Lectures

- **Yesterday:** Introduction to Deep Learning
- **Today:** Recommendation Systems and Deep Learning
- Overview of Recommender Systems (RSes)
  - Paradox of Choice
  - The three generations (1G – 3G)

- Overview of some of the application domains

- **Tomorrow:** Deep Learning for Human-Computer Interaction

This is a lecture series about the challenges (and new opportunities) for ML/DL

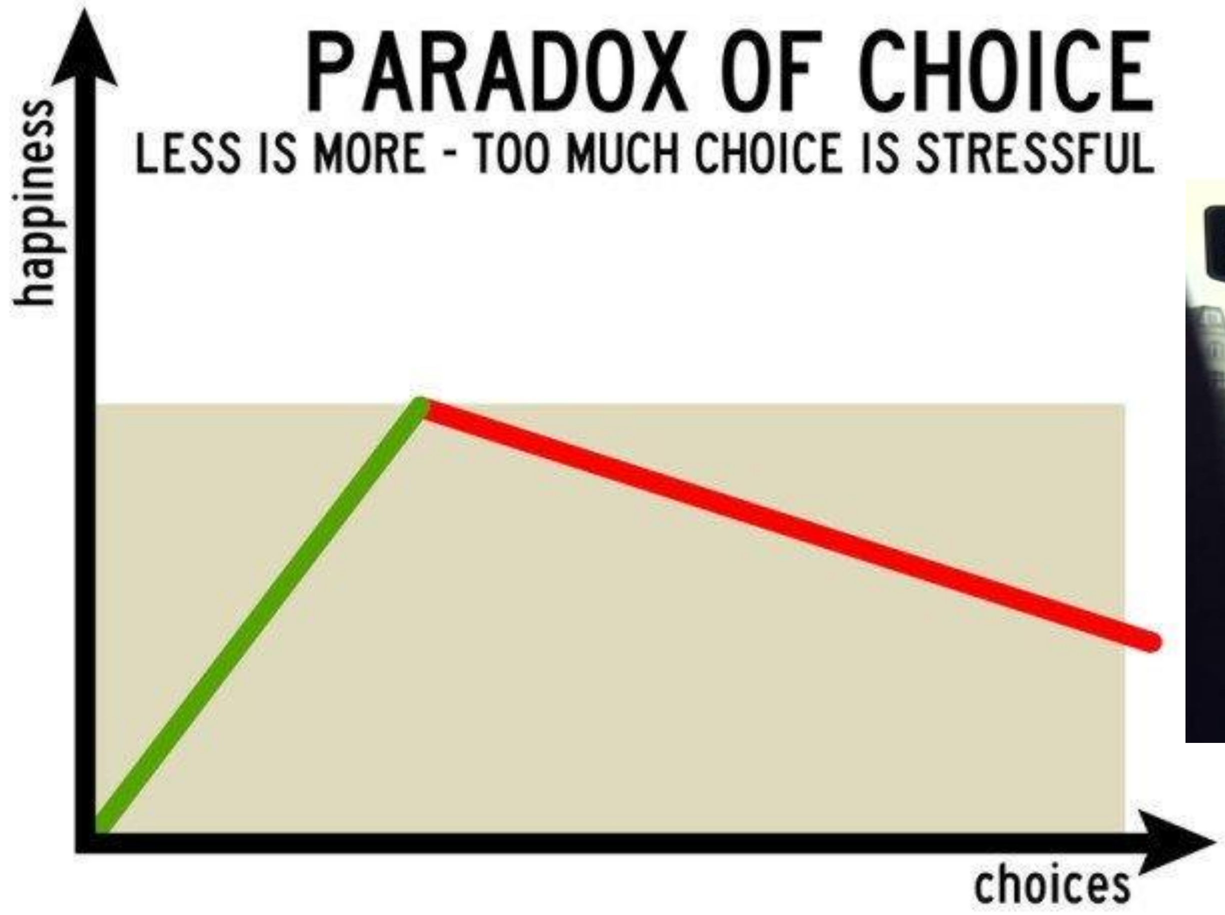
Funded by the Erasmus+ Programme of the European Union

Jam Challenge

24 варианта джема vs **6 вариантов джема**

24 варианта джема	6 вариантов джема
Привлекли **60%** покупателей	Привлекли **40%** покупателей
Покупатели пробовали **2** вкуса в среднем	Покупатели пробовали **2** вкуса в среднем
3% покупателей купили джем	**30%** покупателей купили джем

Less is More


PARADOX OF CHOICE
LESS IS MORE - TOO MUCH CHOICE IS STRESSFUL
happiness
choices

Recommendation Systems: Academia

- Huge progress over the last 20 years
  - from the 3 initial papers published in 1995
  - to 1000’s of papers now
  - Annual ACM RecSys Conference (since 2007)
    - E.g., Boston/MIT in 2016, Milan in 2017
    - Hundreds of submissions and participants
- Interdisciplinary field, comprising
  - CS, data science, statistics, marketing, OR, psychology
- A LOT of interest from industry in the academic research. Usually, 40% of RecSys participants are from the industry!
- An excellent example of the symbiosis of the academic research and industrial developments.

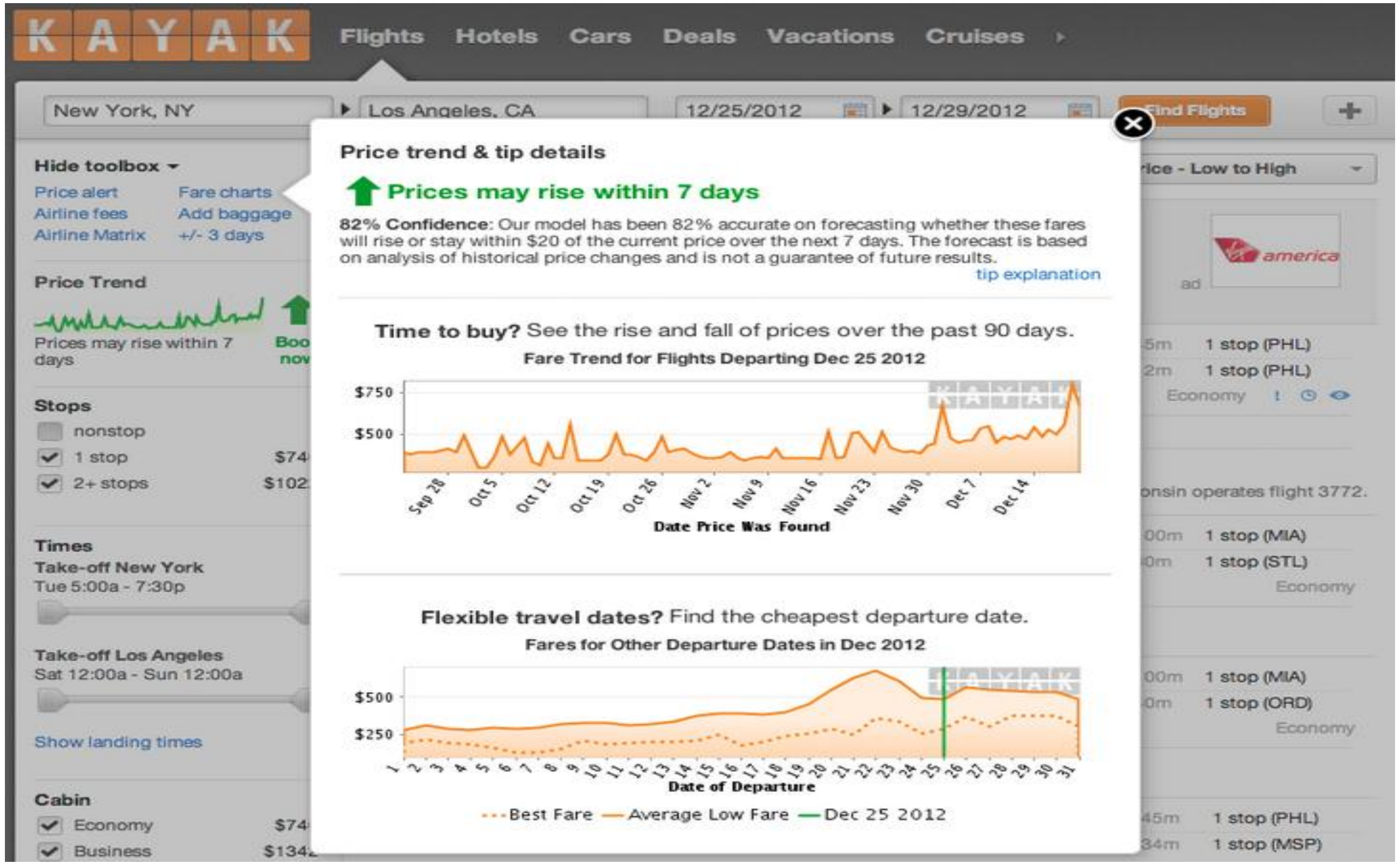
Recommender Systems in the Industry

- Industry pioneers:
  - Amazon, B&N, Net Perceptions (around 1996-1997)
    - Hello, Jim, we have recommendations for you!
- Early days of RSes:
  - User/item-based collaborative filtering [Linden et al 2003]
  - Forrester Research study (2004):
    - 7.4% consumers *often bought* recommended products
    - 22% *ascribe value* to those recommendations
    - 42% were *not interested* in recommended products

Today's Recommenders

- Work across many firms (Netflix, Yelp, Pandora, Google, Facebook, Twitter, LinkedIn) and they operate differently across various applications supported by these firms
- Became mission critical [Colson 2014]: they drive
  - 35% of Amazon's sales
  - 50% of LinkedIn connections
  - 80% of Netflix streamed hours; savings of $1B/yr [GH15]
  - 100% of Stitch Fix sales of its merchandize
    - "By 2020, 100% of what is sold in retail will be by recommendation" (Katrina Lake, CEO of Stitch Fix)
- Deploy sophisticated ML, Big Data, DL and other methods that operate at scale
- *Conclusion:* big progress over the last 15 years!

Buy Now or Tomorrow?



Price trend & tip details

Prices may rise within 7 days

82% Confidence: Our model has been 82% accurate on forecasting whether these fares will rise or stay within $20 of the current price over the next 7 days. The forecast is based on analysis of historical price changes and is not a guarantee of future results.

tip explanation

Time to buy? See the rise and fall of prices over the past 90 days.

Fare Trend for Flights Departing Dec 25 2012

Flexible travel dates? Find the cheapest departure date.

Fares for Other Departure Dates in Dec 2012

Startup

bought by

Microsoft Co.

2011

$210millions

100 employers

Three Generations of Recommender Systems

- Overview of the traditional paradigm of RSes (1st generation)
- Current generation of RSes (2nd generation)
  - The opportunities and challenges
- Towards the next (3rd) generation of RSes

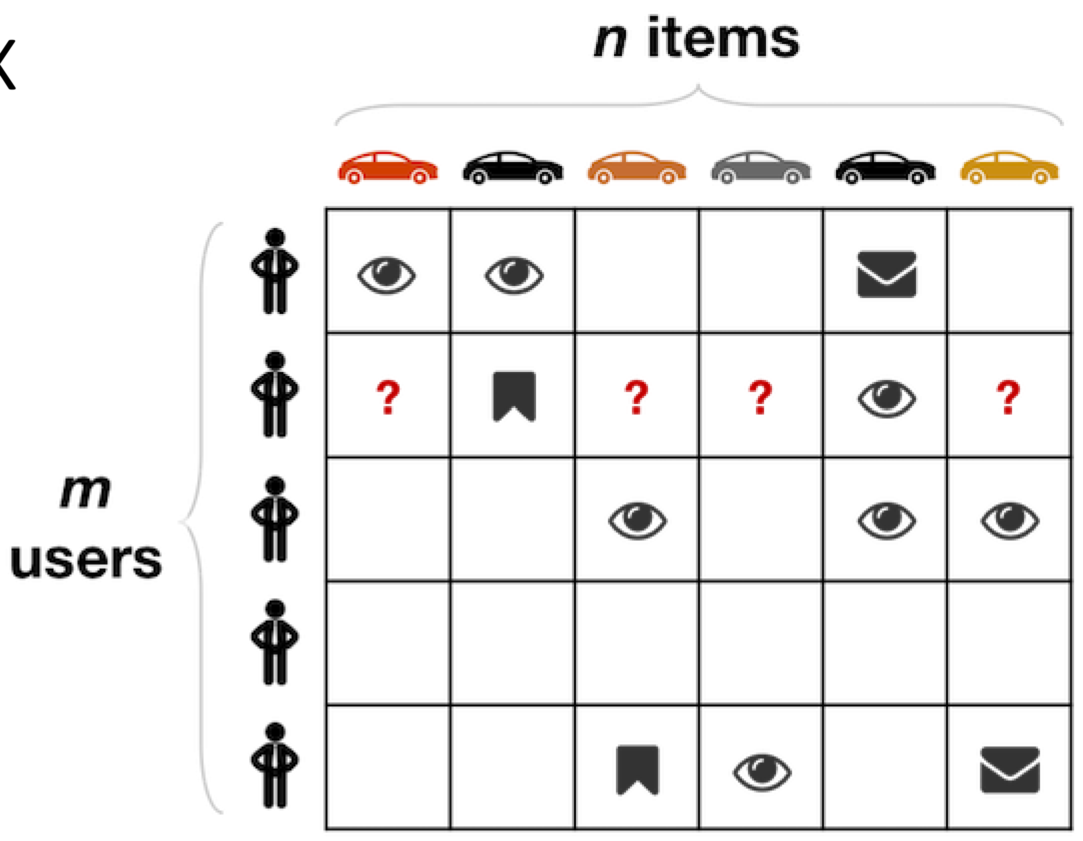
Based on A. Tuzhilin, NY University

Traditional Paradigm (1G) of Recommender Systems

- Two-dimensional (*2D*): Users and Items
- Utility of an item to a user revealed by a *single* rating
  - binary or multi-scaled (e.g. stars on Netflix)
- Recommendations of *individual* items provided to *individual* users
  - Solution via estimation of unknown ratings

2D Recommendation Matrix

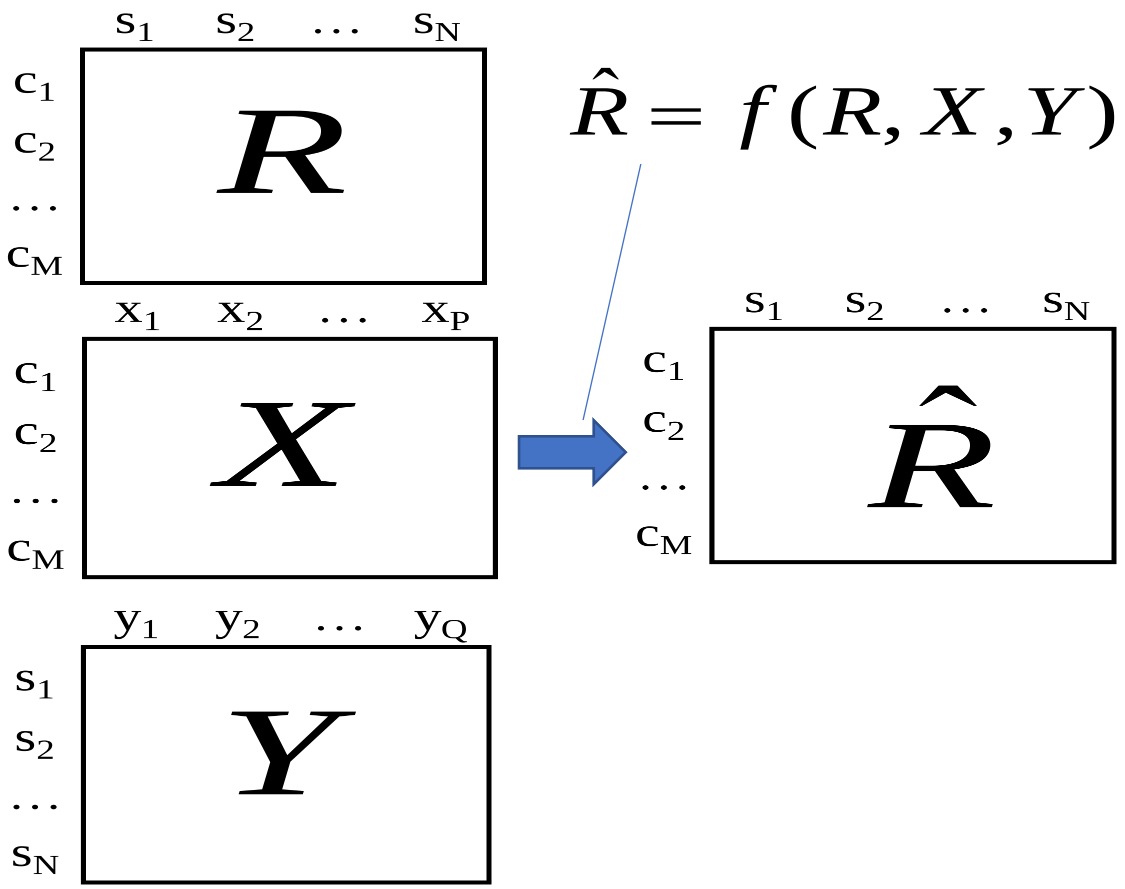
	King Arthur	Water Life	Brillia Mind	Avatar
U1	4	3	2	4
U2		4	5	5
U3	2	2	4	
U4	3		5	2



- The 2D Users × Items = Matrix of Ratings
  - matrix is sparse: only few ratings are specified
- Key issue: accurate estimation of unknown ratings

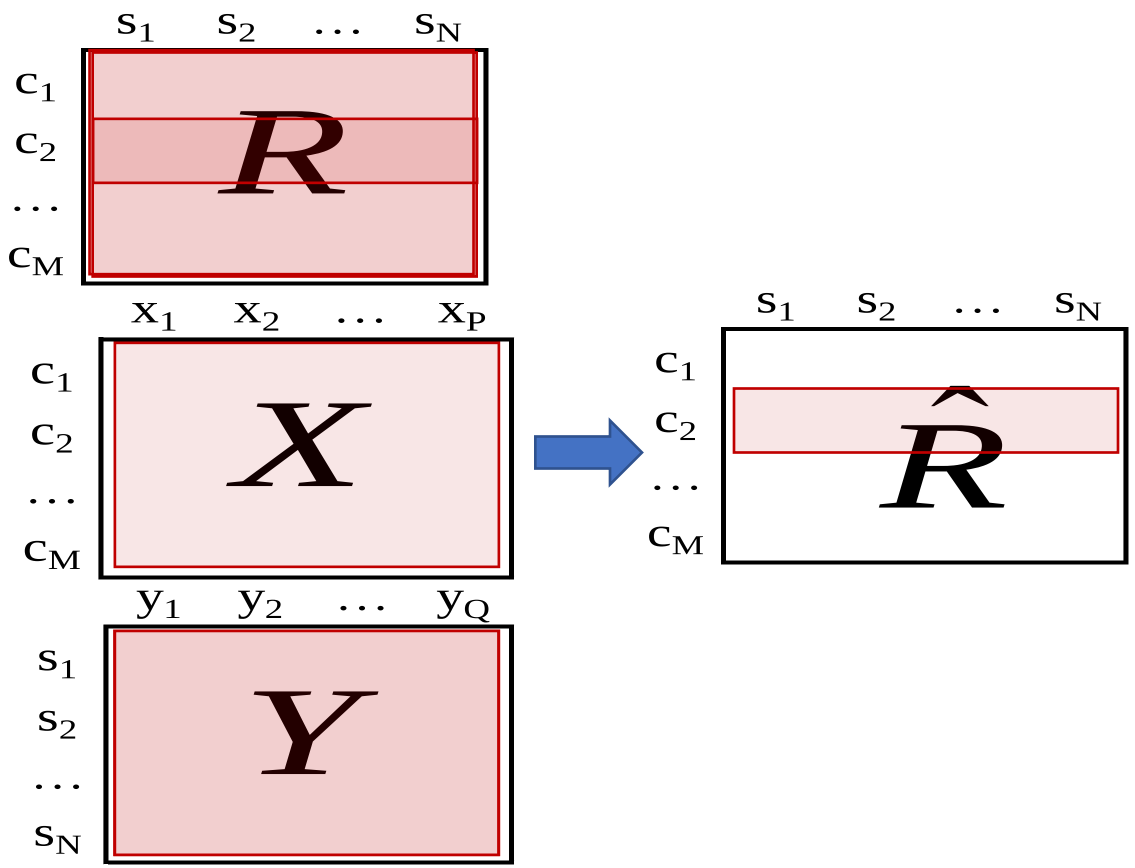
Traditional Approaches

- Input
  - Rating matrix *R:* r_{ij} – rating user c_i assigns to item s_j
  - User attribute matrix *X:* x_{ij} – *attribute* x_j *of user* c_i
  - Item attribute matrix *Y:* y_{ij} – *attribute* y_j *of item* s_i
- Output
  - Predicted rating matrix (predicted utility) $\hat{R}$



Types of Recommendations [Balabanovic & Shoham 1997]

- Content-based
  - build a model based on a description of the item and a profile of the user's preference, keywords are used to describe the items; beside, a user profile is built to indicate the type of item this user likes.
- Collaborative filtering
  - All observed ratings are taken as input to predict unobserved ratings. Recommend items based only on the users past behavior
  - User-based: Find similar users to me and recommend what they liked
  - Item-based: Find similar items to those that I have previously liked
- Hybrid
  - All observed ratings, item attributes, and user attributes are taken as input to predict observed ratings

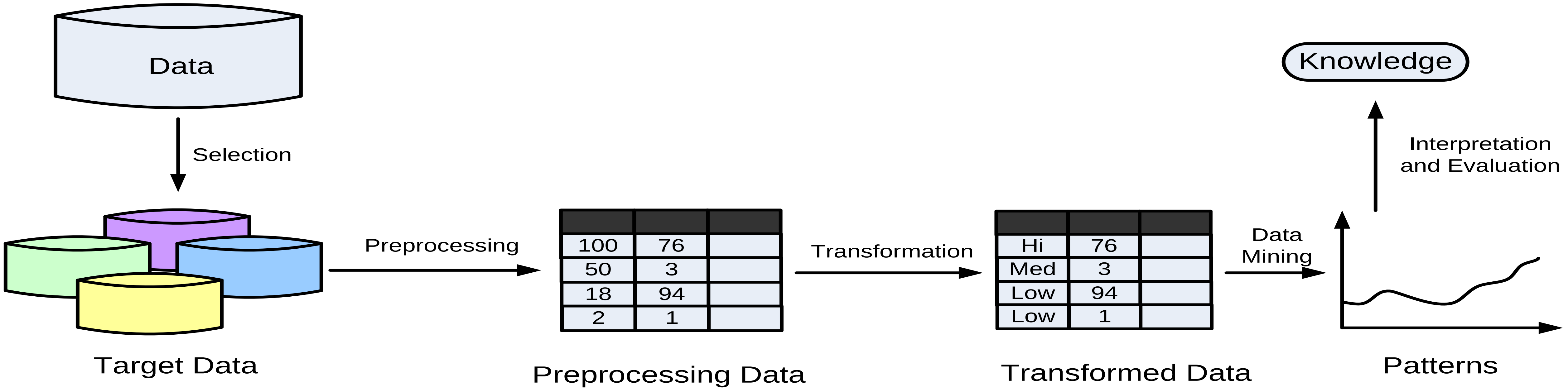

s_1 s_2 $\dots$ s_N
c_1 c_2 $\dots$ c_M
R
x_1 x_2 $\dots$ x_P
X
y_1 y_2 $\dots$ y_Q
Y
$\hat{R}$

Taxonomy of Traditional Recommendation Methods

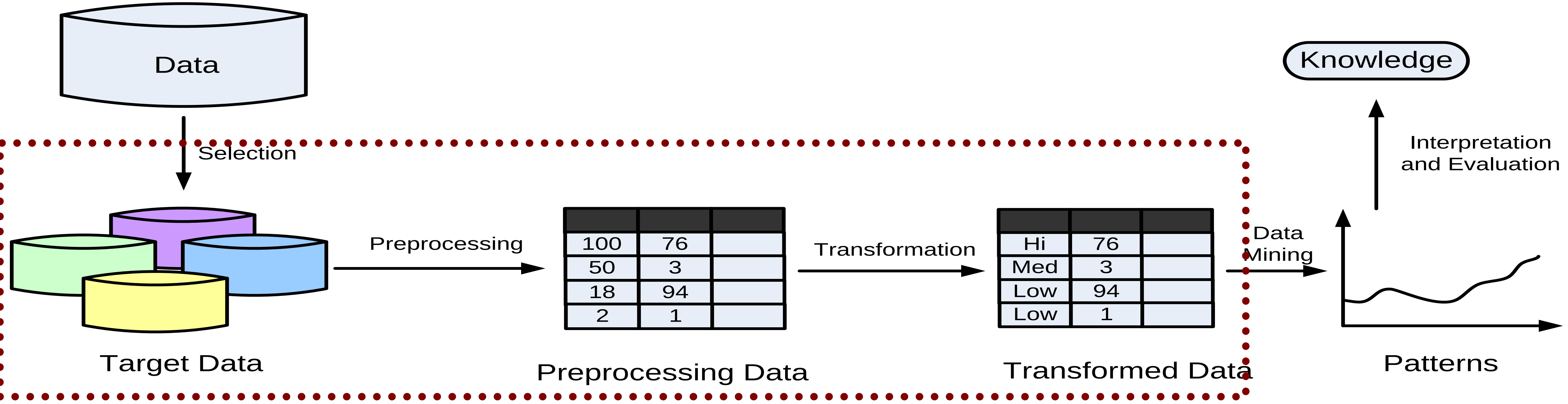
- Classification based on
  - Recommendation approach
    - Content-based, collaborative filtering, hybrid
  - Nature of the prediction technique
    - Heuristic-based, model-based

	Heuristic-based	Model-based
Content-based	🟧	
Collaborative filtering		
Hybrid		

Knowledge Discovery in Databases (KDD) process



Knowledge Discovery in Databases (KDD) process

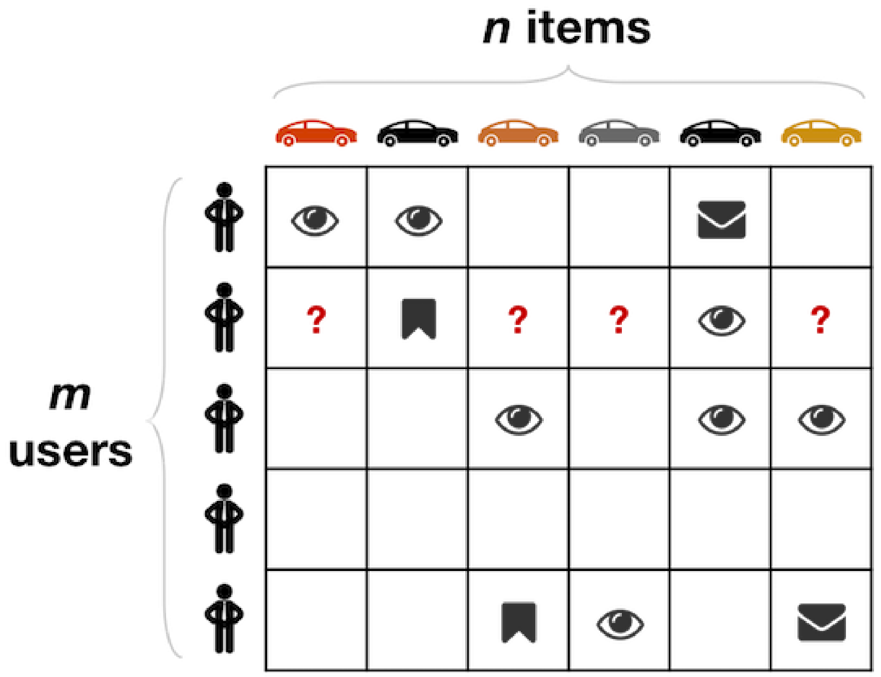


Information Retrieval Techniques.

In the KDD process, data is represented in a tabular format.

Example 1

Attributes (features, measurement)				**Class**
Name	Money Spent	Bought Similar	Visits	Will Buy
John	High	yes	Frequently	?
Mery	High	yes	Rarely	yes



There are different types of features based on the characteristics of the feature and the values they can take. For instance, Money Spent can be represented using numeric values, such as $25. In that case, we have a continuous feature, whereas in our example it is a discrete feature, which can take a number of ordered values: {High, Normal, Low}.

Item Similarity Methods

Item Similarity Methods: Problem No.1

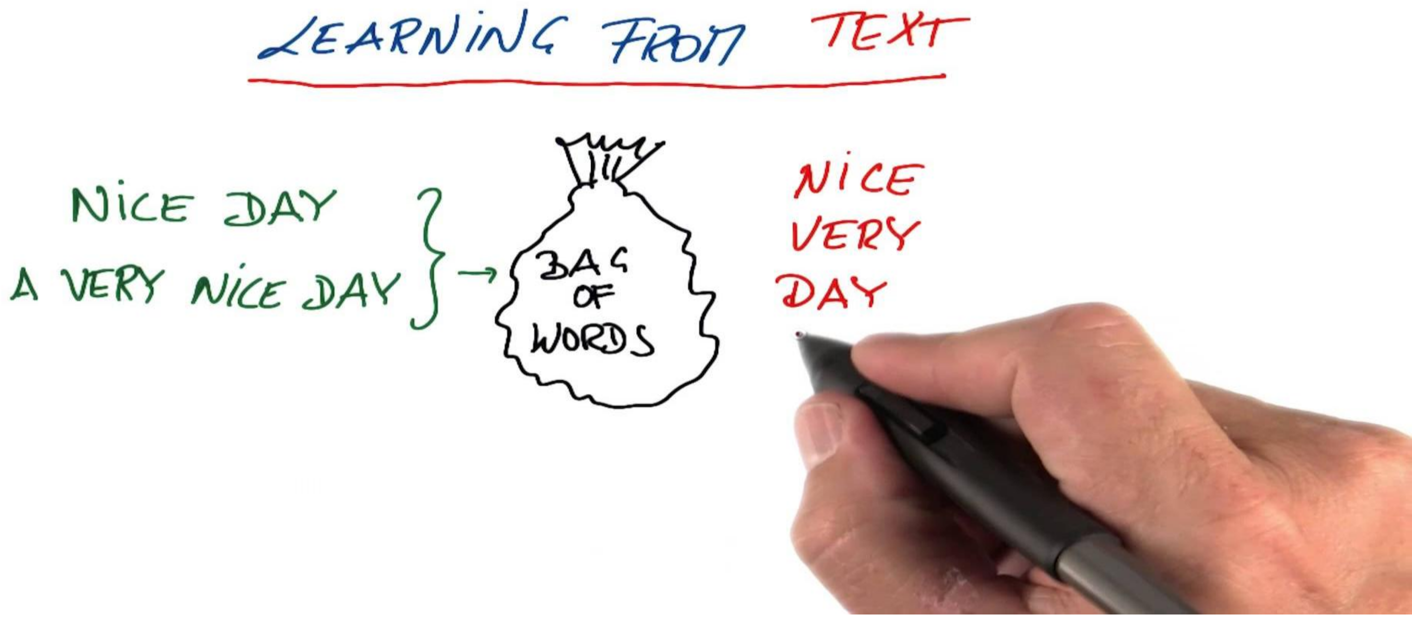
- In social media, individuals generate many types of **nontabular data**, such as **text, voice, or video**.

- These types of data are first converted to tabular data and then processed us mining algorithms.

- For instance, voice can be converted to feature values using approximation techniques such as the fast Fourier transform (FFT) and then processed using data mining algorithms.

Statistical Models

- A document is typically represented by a *bag of words* (unordered words with frequencies).
- Bag = set that allows multiple occurrences of the same element.



Boolean Model Disadvantages

- Similarity function is boolean
  - Exact-match only, no partial matches
  - Retrieved documents not ranked
- All terms are equally important
  - Boolean operator usage has much more influence than a critical word
- Query language is expressive but complicated

Vectorization (VSM)

- A well-known method for vectorization is the vector-space model introduced by Salton, Wong, and Yang

Vector Space Model

- In the vector space model, we are given a set of documents D. Each document is a set of words.
- The goal is to convert these textual documents to [feature] vectors.
- We can represent document i with vector di ,

$$d_i = (w_{1,i} , w_{2,i} , \dots , w_{N,i}),$$

- where wj,i represents the weight for word j that occurs in document i and N is the number of words used for vectorization

To compute wj,i , we can set it to 1 when the word j exists in document i and 0 when it does not. We can also set it to the number of times the word j is observed in document i.

Document Collection

- A collection of *n* documents can be represented in the vector space model by a term-document matrix.
- An entry in the matrix corresponds to the "weight" of a term in the document; zero means the term has no significance in the document or it simply doesn't exist in the document.

$$
\left(\begin{array}{lllll}
 & T_1 & T_2 & \dots. & T_t \\
D_1 & w_{11} & w_{21} & \dots & w_{t1} \\
D_2 & w_{12} & w_{22} & \dots & w_{t2} \\
: & : & : & & : \\
: & : & : & & : \\
D_n & w_{1n} & w_{2n} & \dots & w_{tn}
\end{array}\right)
$$

Term Weights: Inverse Document Frequency

- Terms that appear in many *different* documents are *less* indicative of overall topic.

  df_i = document frequency of term i

  = number of documents containing term i

  idf_i = inverse document frequency of term i,

  $= \log_2 (N / df_i)$

  (N: total number of documents)

Term Frequency - Inverse Document Frequency (TF-IDF)

Term Frequency - Infrequent Term Frequency

- In the TF-IDF scheme, wj,i is calculated as wj,i = t fj,i × id fj , (5.2) where t fj,i is the frequency of word j in document i. id fj is the inverse TF-IDF frequency of word j across all documents,

$$\mathrm{IDF_i} = \log_2 \frac{|\mathrm{D}|}{\left|\left\{\mathrm{document} \in \mathrm{D} \,\middle|\, \mathrm{j} \in \mathrm{document}\right\}\right|}$$

- which is the logarithm of the total number of documents divided by the number of documents that contain word j.
- TF-IDF assigns higher weights to words that are less frequent across documents and, at the same time, have higher frequencies within the document they are used.
- This guarantees that words with high TF-IDF values can be used as representative examples of the documents they belong to and also, that stop words, such as “the,” which are common in all documents, are assigned smaller weights.

Example 2

- Consider the words “apple” and “orange” that appear 10 and 20 times in document d_1.
- Let |D| = 20 and assume the word “apple” only appears in document d_1 and the word “orange” appears in all 20 documents. Then, TF-IDF values for “apple” and “orange” in document d1 are

$$\mathrm{TF} \times \mathrm{IDF}(\text{"apple"}, \mathrm{d}_1) = 10 \times \log_2 \frac{20}{1} = 43{,}22,$$

$$\mathrm{TF} \times \mathrm{IDF}(\text{"orange"}, \mathrm{d}_1) = 20 \times \log_2 \frac{20}{20} = 0.$$

Consider the following three documents:

Example 3

- d_1= “social media mining”
- d_2= “social media data”
- d_3= “financial market data”
- The tf values are as follows: :

	social	media	mining	data	financial	market
d_1						
d_2						
d_3						

Consider the following three documents:

Example 3

- d_1= “social media mining”
- d_2= “social media data”
- d_3= “financial market data”
- The TF values are as follows: :

	social	media	mining	data	financial	market
d_1	1	1	1	0	0	0
d_2	1	1	0	1	0	0
d_3	0	0	0	1	1	1

The IDF values are

$$\mathrm{IDF}(\text{"social"}) = \log_2 \frac{3}{2} = 0,584,$$

$$\mathrm{IDF}(\text{"media"}) = \log_2 \frac{3}{2} = 0,584,$$

$$\mathrm{IDF}(\text{"mining"}) = \log_2 \frac{3}{1} = 1,584,$$

$$\mathrm{IDF}(\text{"data"}) = \log_2 \frac{3}{2} = 0,584,$$

$$\mathrm{IDF}(\text{"financial"}) = \log_2 \frac{3}{1} = 1,584,$$

$$\mathrm{IDF}(\text{"market"}) = \log_2 \frac{3}{1} = 1,584.$$

The TF-IDF values can be computed by multiplying TF values with the IDF values:

- d_1= “social media mining”
- d_2= “social media data”
- d_3= “financial market data”

	social	media	mining	data	financial	market
d_1	0,584	0,584	1,584	0	0	0
d_2	0,584	0,584	0	0,584	0	0
d_3	0	0	0	0,584	1,584	1,584

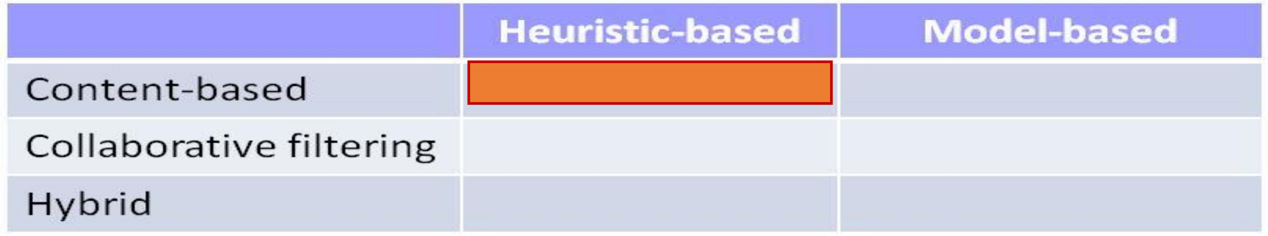
After vectorization, documents are converted to vectors, and common data mining algorithms can be applied. However, before that can occur, the quality of data needs to be verified.

Item Similarity Methods

- Information Retrieval Techniques
  Item attributes correspond to word occurrences in item descriptions

$y_{ij} = TF_{ij} \cdot IDF_j$, TF_{ij} – term frequency: frequency of word y_j occurring in the description of item s_i; IDF_j – inverse document frequency: inverse of the frequency of word y_j occurring in descriptions of all items.

- Content-based profile $\mathbf{v}_i$ of user c_i constructed by aggregating profiles of items c_i has experienced



	Heuristic-based	Model-based
Content-based	🟧	
Collaborative filtering		
Hybrid		

$$\hat{r}_{ij} = score(\mathrm{v}_i, \mathrm{y}_j)$$

$$\hat{r}_{ij} = \cos(\mathrm{v}_i, \mathrm{y}_j) = \frac{\mathrm{v}_i \bullet \mathrm{y}_j}{\| \mathrm{v}_i \|_2 \cdot \| \mathrm{y}_j \|_2}$$

Content-Based kNN Method

- Each item is defined by its content *C.*
  - Content is application-specific, e.g., restaurants vs. music
  - Content *C* is *represented* as a vector $\hat{C}=(c_1, c_2,\ldots, c_d)$
    - E.g., as a TF-IDF vector in the previous case
- *Content-based kNN method:*
  - Assume user also rated *n* items $(r_1, r_2, \ldots, r_n)$.
  - Then for *n* known item/rating pairs $(\hat{C}_1, r_1)$, $(\hat{C}_2, r_2)$, …, $(\hat{C}_n, r_n)$ and a new item $\hat{C}$, estimate its rating *r* as a weighted average of *Ĉ's k* nearest neighbors, where the distance between two items $dist(\hat{C}, \hat{C}_i)$ can be defined as $cos(\hat{C}, \hat{C}_i)$.

Item-Based Collaborative Filtering

- Same r_{ij} estimation as for the user-based but use item-to-item *sim(i, i')* instead of user-to-user similarity
- Used by Amazon 15 years ago [Linden03]
- Compute item-to-item similarity offline [Linden03]:

```
For each item $i$ in the catalog
  For each user $u$ in $Purchased(u, i)$
    For each item $i'$ in $Purchased(u, i')$
          Record items $i$ and $i'$ as $CoPurchased(i, i', u)$
  Compute $sim(i, i')$ based on $CoPurchased(i, i', u)$
```

- Store *{u: Purchased(u,i)}* & *{i: Purchased(u,i)}* as lists

A. Tuzhilin

Association-Rule-Based CF

Another example of CF heuristic

Assume user A had transaction *T* with items *I = (i_1, i_2, …, i_k).*

Q: Which other items should A be recommended?

Step 1 (offline): find the association rules $X \Rightarrow Y$ with support and confidence thresholds of (α, β) respectively

Step 2 (online):

a. Find all the rules $X \Rightarrow Y$ *fired* by A's transaction *T*
   - Rules where *X* is in *I*
b. Take union of *Y*'s items *not* in *I across all* the fired rules
   - Remove duplicates: select items with largest confidence
c. Sort them by the confidence levels of their fired rules
d. Recommend to A the top *N* items in the sorted list.

Association-Rule-Based CF: Supermarket Purchases

User A bought *I* = (Bread, Butter, Fish)

Q: What else to recommend to A?

Step 1: find rules X $\Rightarrow$ Y with support and conf > (25%,60%) respectively

Example: Bread, Butter $\Rightarrow$ Milk (s=2/7=29%, c=2/3=67%)

Step 2:

a. This rule is fired by A's transaction
b. Thus, add Milk to the list (c=67%)
c. Do the same for *all other* rules fired by A's transaction
d. Recommend Milk to A if Milk makes the top-N list with c = 67%

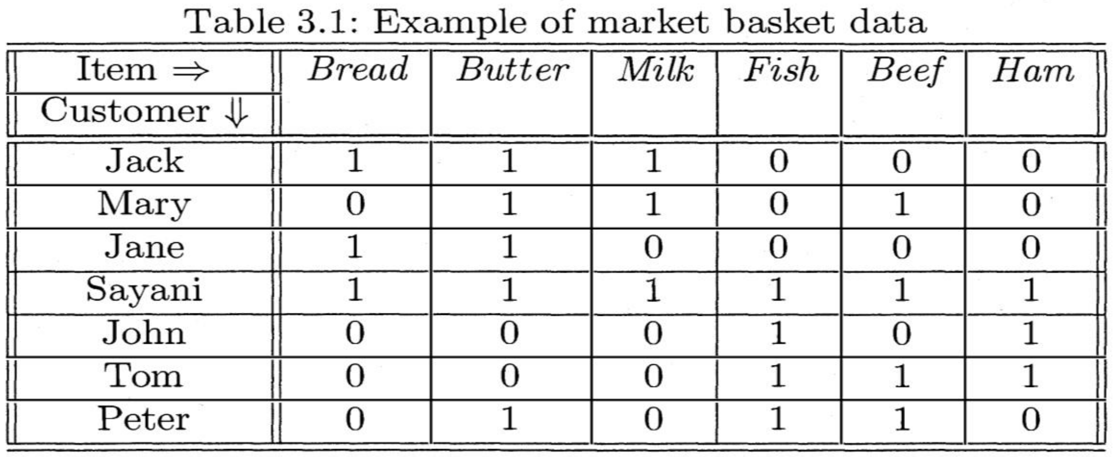


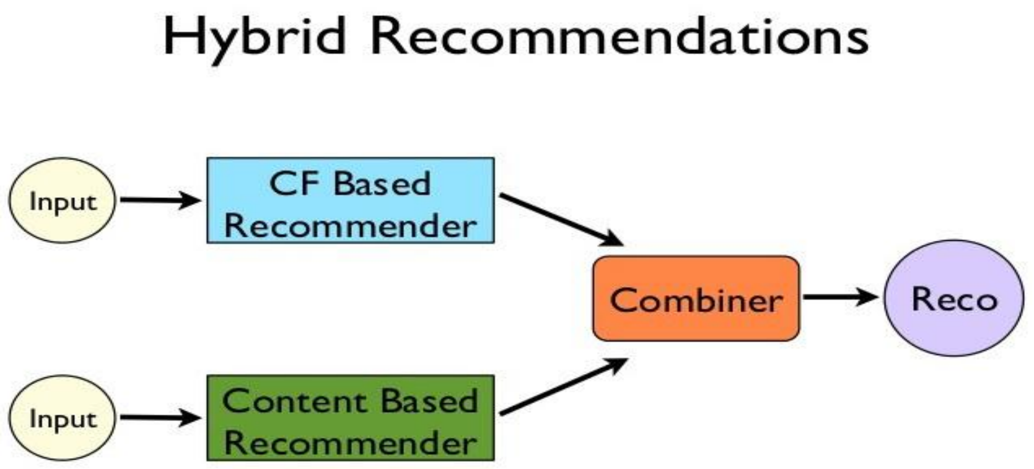
Table 3.1: Example of market basket data

Item $\Rightarrow$ / Customer $\Downarrow$	*Bread*	*Butter*	*Milk*	*Fish*	*Beef*	*Ham*
Jack	1	1	1	0	0	0
Mary	0	1	1	0	1	0
Jane	1	1	0	0	0	0
Sayani	1	1	1	1	1	1
John	0	0	0	1	0	1
Tom	0	0	0	1	1	1
Peter	0	1	0	1	1	0

Hybrid: Combining Other Methods

- The hybrid approach can combine two or more methods to gain better performance results.
- Types of combination:
  - *Weighted* combination of the recommender scores
  - *Switching* between recommenders depending on the situation
  - *Cascade:* one system *refines* recommendations of another
  - *Mixed*: several recommender results presented together

Example:



Source: Dataconomy

Performance Evaluation of RSes

Importance of Right Metrics

- There are measures and… measures!
  - Assume you improved the RMSE of Netflix by 10%. So what?
- What do you really want to measure in RSes?
  - Economic value/impact of recommendations
  - *Examples*: increase in sales/profits, customer loyalty/churn, conversion rates,…
- Need live experiments with customers (A/B testing) to measure true performance of RSes

Evaluation Paradigms

- User studies
- Online evaluations (A/B tests)
- Offline evaluation with observational data
- Long-term goals vs. short-term proxies
- Combining the paradigms: offline and online evaluations

Example of A/B Testing

- Online University: a RS recommends remedial learning materials to the students who have “holes” in their studies
- Applied this Recommender System to
  - 42 different courses from CS, Business and General Studies
  - over 3 semesters of 9 weeks each
  - 910 students from all over the world
  - 1514 enrollments in total (i.e., 1514 student/course pairs).
- *Goal:* show that this RS “works:” students following the advice perform better than the control group.

Accuracy-Based Metrics

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}$$

$$\text{MAE} = \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y}_j|$$

- For Prediction
  - RMSE and MAE
- For Classification
  - *Precision*: percentage of good recommendations among all the recommended items
  - *Recall*: percentage of items predicted as good among all the actually good items
  - *F-measure*: 2*Prec*Recall/(Prec + Recall)
- For Ranking
  - Discounted cumulative gain (DCG)
  - Where rel_i is relevance of recommended item in position *i.*

$$\text{DCG}_\text{p} = rel_1 + \sum_{i=2}^{p}\frac{rel_i}{\log_2(i)}$$

Netflix Prize Competition

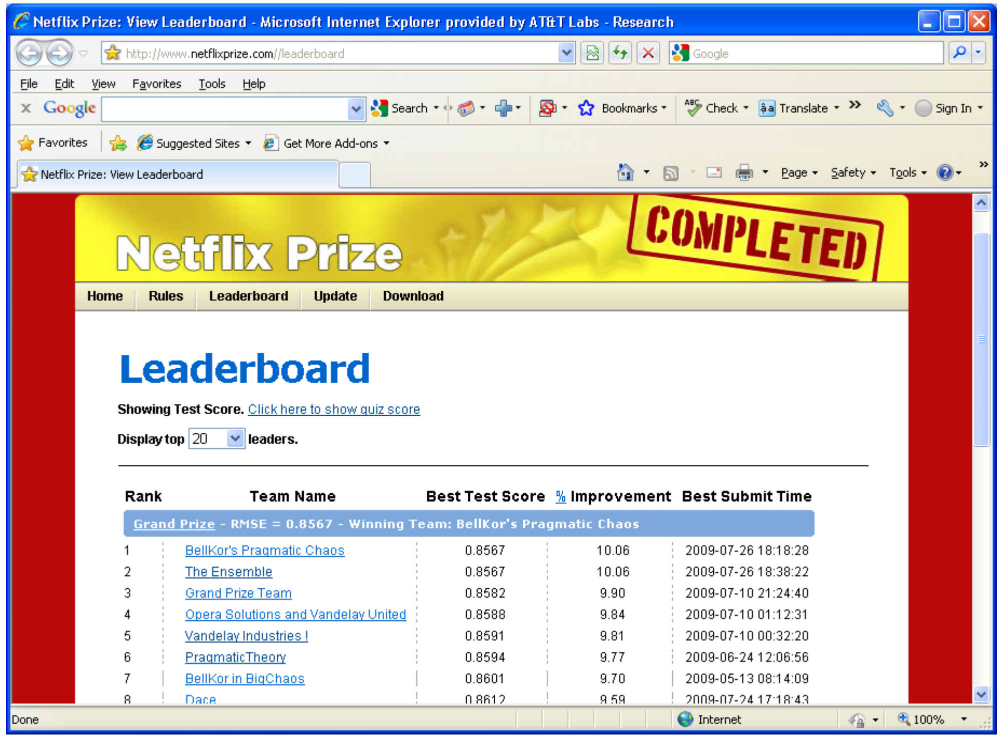
- Competition for the best algorithm to predict user ratings for films based on prior ratings
- Data: training dataset of 100,480,507 ratings over 7 years
  - 480,189 users and 17,770 movies
- Task: improve RMSE by 10% over Netflix's own algorithm
- Prize: $1,000,000
- Starting date: October 2, 2006
- The size: 20,000+ teams from over 150 countries registered; 2,000 teams submitted over 13,000 prediction sets (June 2007)
- Results: 2 teams reached the 10% goal on July 26, 2009:
  - BelKor Pragmatic Chaos (7 ppl) and Ensemble (20 ppl)
  - RMSE was improved from 0.9514 to 0.8567 (over almost 3 years!)
- $1M Prize awarded to BelKor Pragmatic Chaos on 9/18/2009

Test Set Results (RMSE)

- The Ensemble: 0.856714
- BellKor’s Pragmatic Theory: 0.856704

- Both scores round to 0.8567

- Tie breaker is submission date/time


Netflix Prize: View Leaderboard - Microsoft Internet Explorer provided by AT&T Labs - Research
http://www.netflixprize.com//leaderboard
Netflix Prize
COMPLETED
Home
Rules
Leaderboard
Update
Download
Done
Internet
100%

Leaderboard

Showing Test Score. Click here to show quiz score

Display top 20 **leaders.**

Rank	Team Name	Best Test Score	% Improvement	Best Submit Time
Grand Prize - RMSE = 0.8567 - Winning Team: BellKor's Pragmatic Chaos				
1	BellKor's Pragmatic Chaos	0.8567	10.06	2009-07-26 18:18:28
2	The Ensemble	0.8567	10.06	2009-07-26 18:38:22
3	Grand Prize Team	0.8582	9.90	2009-07-10 21:24:40
4	Opera Solutions and Vandelay United	0.8588	9.84	2009-07-10 01:12:31
5	Vandelay Industries !	0.8591	9.81	2009-07-10 00:32:20
6	PragmaticTheory	0.8594	9.77	2009-06-24 12:06:56
7	BellKor in BigChaos	0.8601	9.70	2009-05-13 08:14:09
8	Dace	0.8612	9.59	2009-07-24 17:18:43

What Netflix Prize Winners Done

- Development of new and scalable methods, MF being the most prominent one
- Some Collaborative Filtering methods used in the competition:
  - k-NN
  - Matrix Factorization (with different “flavors”)
  - Regression on Similarity
  - Time Dependence Models
  - Restricted Boltzmann Machine
- (Re-)discovered the power of ensemble (hybrid) methods (“blending”)

Netflix Competition: The End of an Era

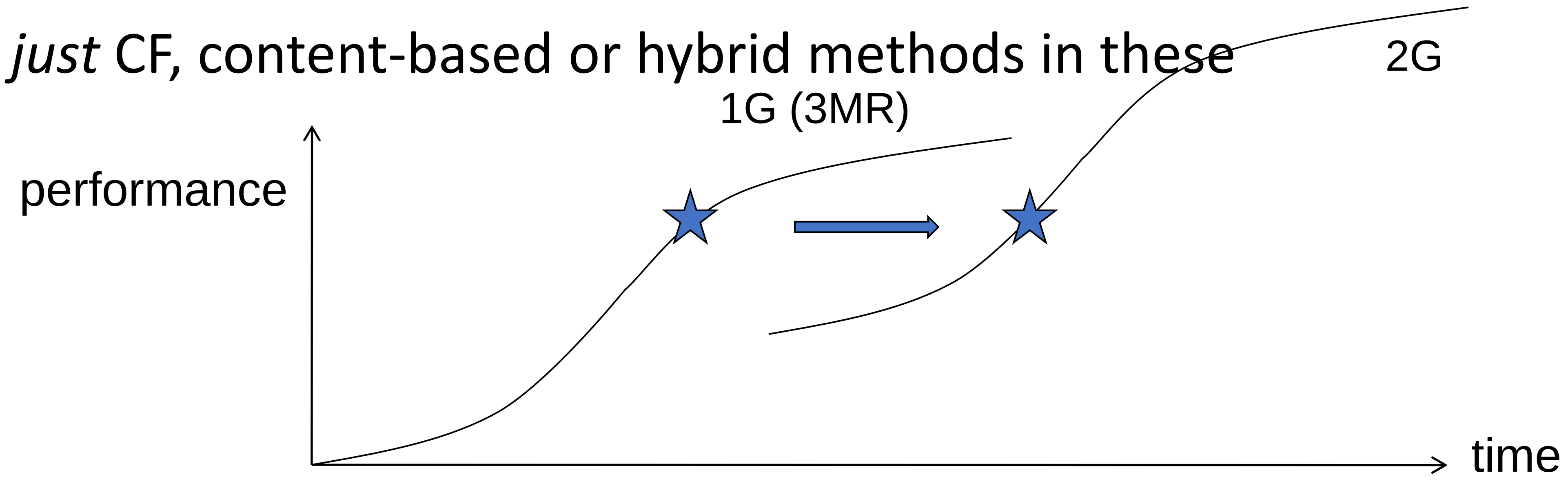
Netflix Prize Competition:

- Completed not only the 2D, but also the 3MR paradigm:
  - 3 matrices Ratings, Users and Items
  - Utility of an item to a user revealed by a *single* rating
  - Recommendations of *individual* items provided to *individual* users
- Developed *more efficient* solutions to a *well-studied* problem [AT05]
  - Scalability was novel: no 100M ratings dataset before

Thinking Outside of the 3MR Box

- The 3MR paradigm worked well for Netflix. But what about other applications?
  - Music, e.g. Pandora and Spotify?
  - Social networks, e.g., LinkedIn and Facebook
  - News and other reading materials, e.g., Google News
  - Restaurants, e.g., Yelp
  - Clothes, e.g. Stitch Fix

It is hard to use *just* CF, content-based or hybrid methods in these applications.



Context-Aware Recommender Systems (CARS)

- Recommend a vacation
  - Winter vs. summer
- Recommend a movie
  - To a student who wants to see it on Saturday night with his girlfriend in a movie theater
- Recommendations depend on the *context*
  - Need to know not only *what* to recommend to *whom*, but also under what *circumstances*
- *Context*: Additional information (besides *Users* and *Items*) that is *relevant* to recommendations

What is Context in Recommender Systems

- A multifaceted concept: 150 (!) definitions from various disciplines (*Bazire&Brezillon 05*)
- One approach: Context can be defined with contextual variables $C = C_1 \times \ldots \times C_n$, e.g.,
  - C = PurchaseContext × TemporalContext
  - c = (work, weekend), i.e., work-related purchases on a weekend
  - Contextual variables C_i have a tree structure

Context-Aware Recommendation Problem

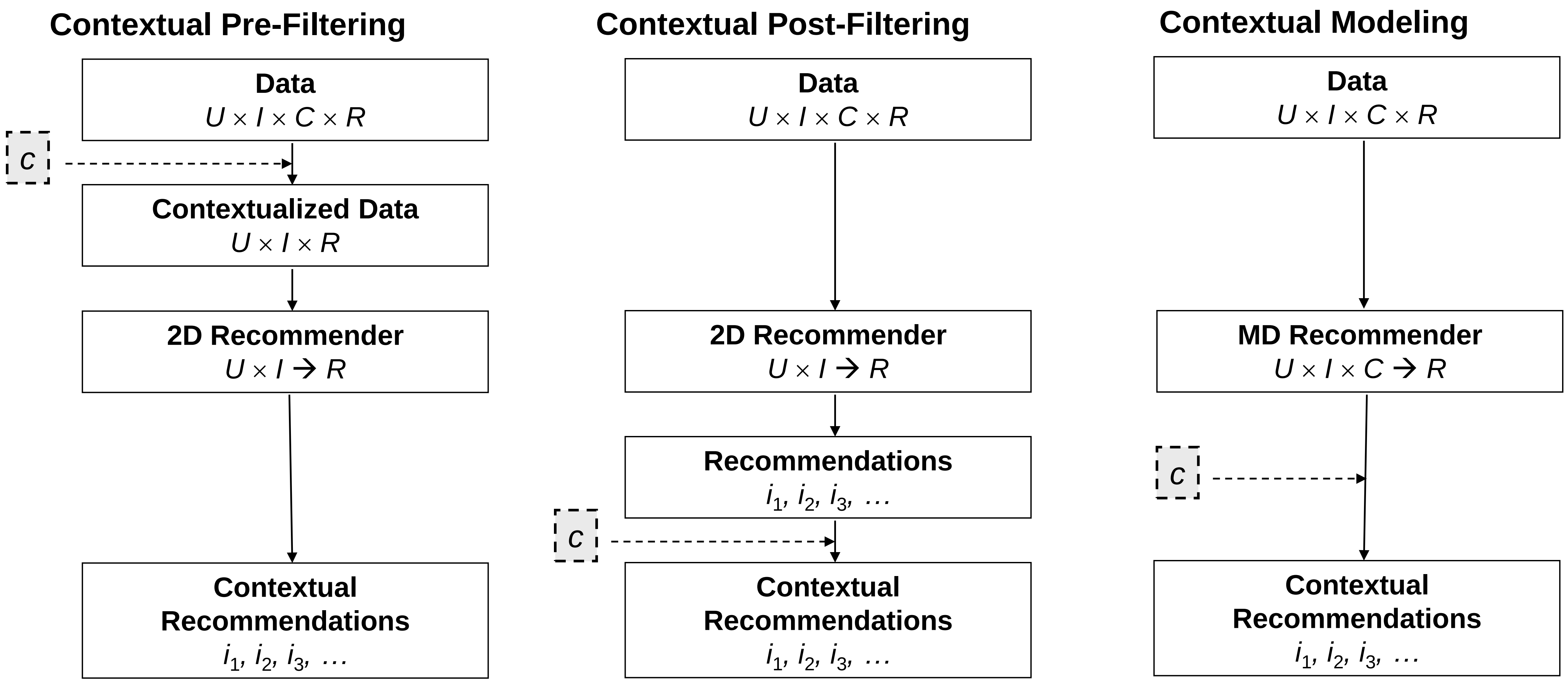
- Data in context-aware recommender systems (CARS)
  - Rating information: *<user, item, rating, context>*
  - In addition to information about items and users, also may have information about *context*
- Problem: how to use context to estimate unknown ratings?

How to Use Context in Recommender Systems [AT10]

Context can be used in the following stages of the recommendation process:

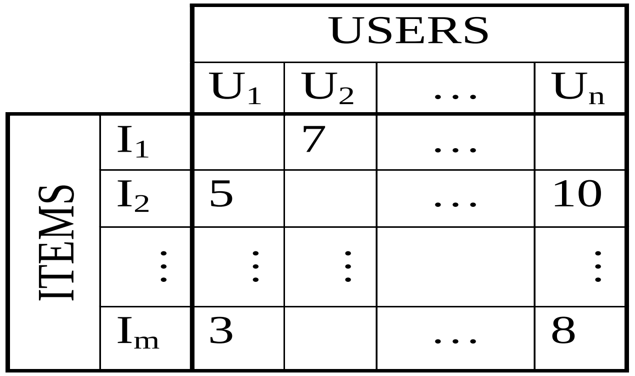
- Contextual pre-filtering
  - Contextual information drives *data selection* for that context
  - Ratings are predicted using a *traditional* recommender on the selected data
- Contextual post-filtering
  - Ratings predicted *on the whole data* using *traditional* recommender
  - The contextual information is used to *adjust* (“contextualize”) the resulting set of recommendations
- Contextual modeling
  - Contextual information is used *directly* in the modeling technique as a part of rating estimation

Paradigms for Incorporating Context in Recommender Systems [AT08]


Contextual Pre-Filtering
Data
$U \times I \times C \times R$
c
Contextualized Data
$U \times I \times R$
2D Recommender
$U \times I \rightarrow R$
Contextual Recommendations
$i_1, i_2, i_3, \dots$
Contextual Post-Filtering
Data
$U \times I \times C \times R$
2D Recommender
$U \times I \rightarrow R$
Recommendations
$i_1, i_2, i_3, \dots$
c
Contextual Recommendations
$i_1, i_2, i_3, \dots$
Contextual Modeling
Data
$U \times I \times C \times R$
MD Recommender
$U \times I \times C \rightarrow R$
c
Contextual Recommendations
$i_1, i_2, i_3, \dots$

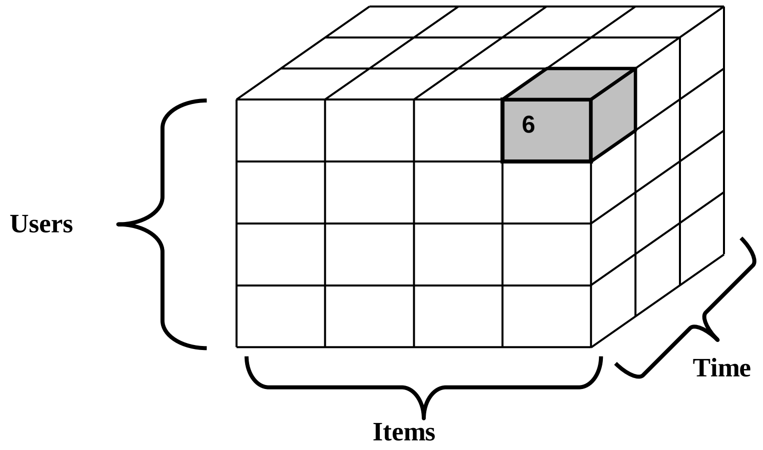
Multidimensional Recommender Systems

Traditional 2D Matrix



		USERS			
		U_1	U_2	…	U_n
ITEMS	I_1		7	…	
	I_2	5		…	10
	⋮	⋮	⋮		⋮
	I_m	3		…	8

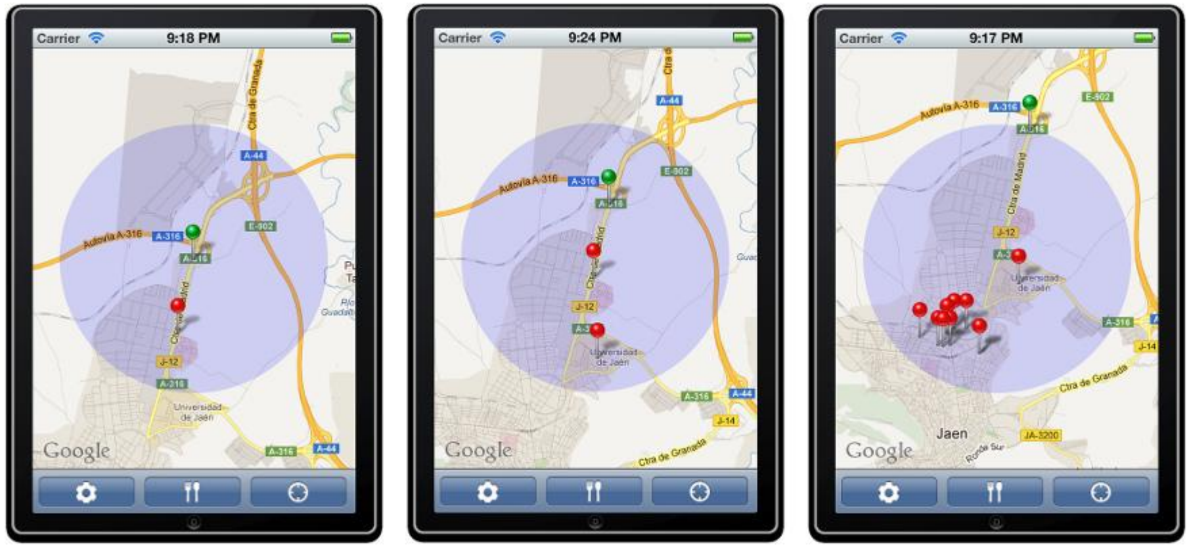
Multidimensional (OLAP-based) cube



Problem: how to estimate ratings on this cube?

Mobile Recommender Systems

- A special case of CARS
- Very different from traditional RSes
    - Spatial context
    - Temporal context
    - Trace data (sequences of locations & events)
    - Less rating-dependent



Route Recommendations for Taxi Drivers (based on [Ge et al 2010])

- ***Goal:*** recommend travel *routes* to taxi (or Uber) drivers to improve their *economic* performance
- Defining features:
  - Input data: driving/location traces
  - Recommendation: a driving *route* (space/time)
  - Performance metric: *economics*-based, e.g.,
    - Revenue per time unit
    - Minimize idle/empty driving time
- *Example:* recommend best driving *routes* to pick passengers to minimize empty driving
- *Challenge:* combinatorial explosion!

Key Ideas Behind the Solution

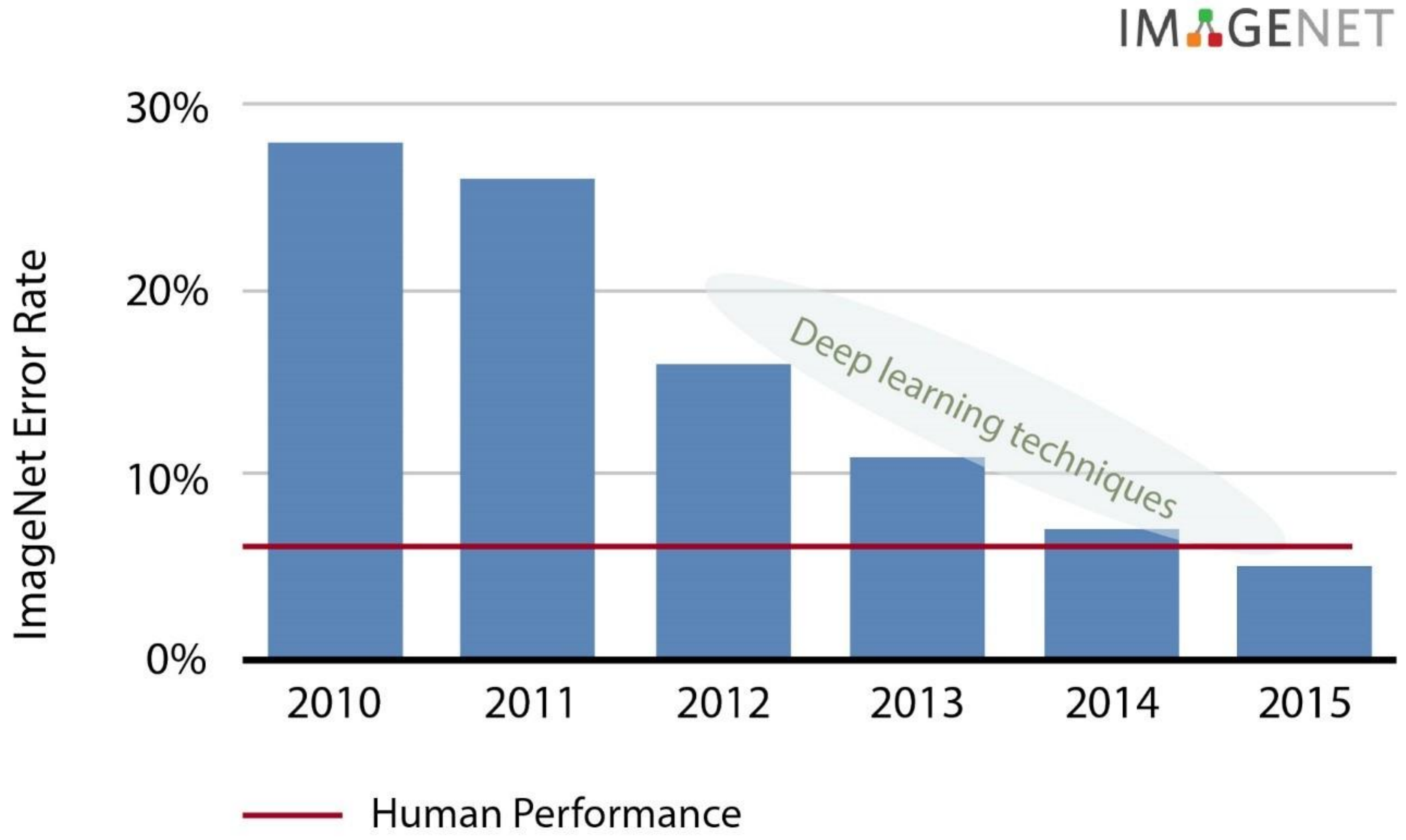
- Need to model/represent driving routes
  - Finite set of popular/historical "*pick up points*"
  - Cluster them into pickup hubs (use of clustering techniques)
  - Route recommendation: sequence of *pickup hubs*
- Compute expected "empty" travel distances
  - *Performance measure*: Potential Travel Distance
- Leverage prior driving patterns of *experienced* taxi drivers to recommend "good" routes
- Less experienced drivers should follow the driving patterns of more experienced drivers ("collaborative" approach)
- Technical details in [Ge et al. 2010]

Results of a Study

- Data on 500 taxis in SF driving over 30 days
  - “Successful” drivers: over 230 driving hours and 0.5 occupancy rates; 20 such drivers (the “*role models*”)
  - Focus on 2 time periods: 2 – 3pm & 6 – 7pm
  - Computed 636 and 400 historical pickup points for these 2 periods based on 20 good drivers
  - Computed driving distances between these points using Google Map API
  - Computed 10 clusters for 636 & 400 pickup points
  - Construct an optimal route for a new driver at that time (based on these clusters) and recommend it to him/her.

(DL)

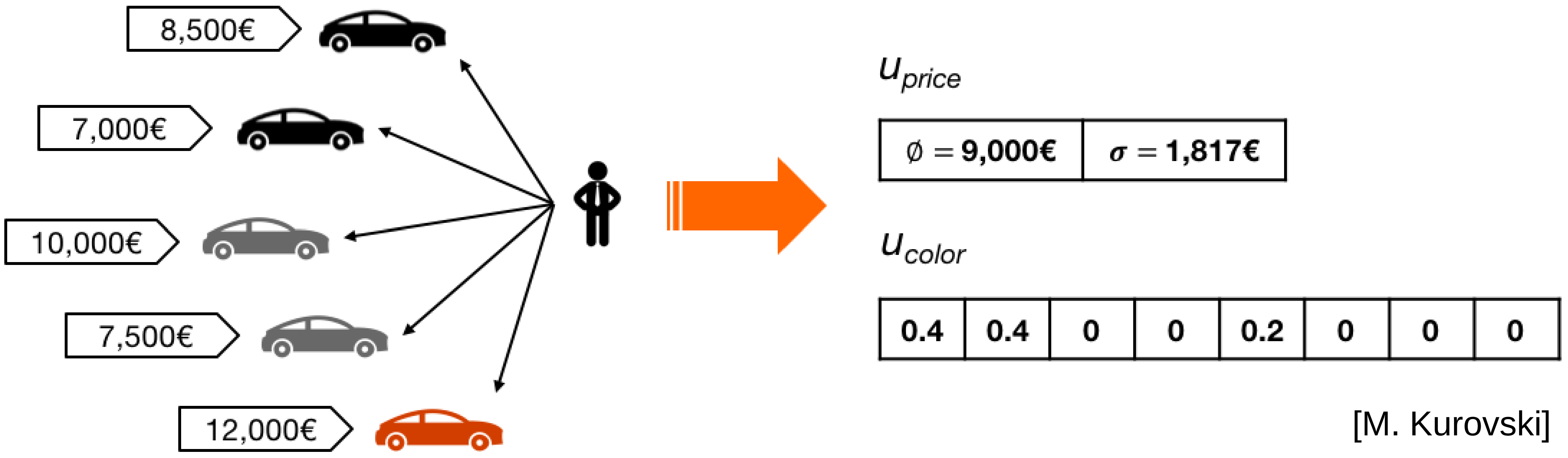
Why DL for RSes?



ImageNet challenge error rates (red line = human performance)

DL for Vehicle Recommendations

- Using deep learning to improve vehicle suggestions, we have two basic goals:
- Increase the **relevance** of recommendations
- Provide them in a **scalable** way

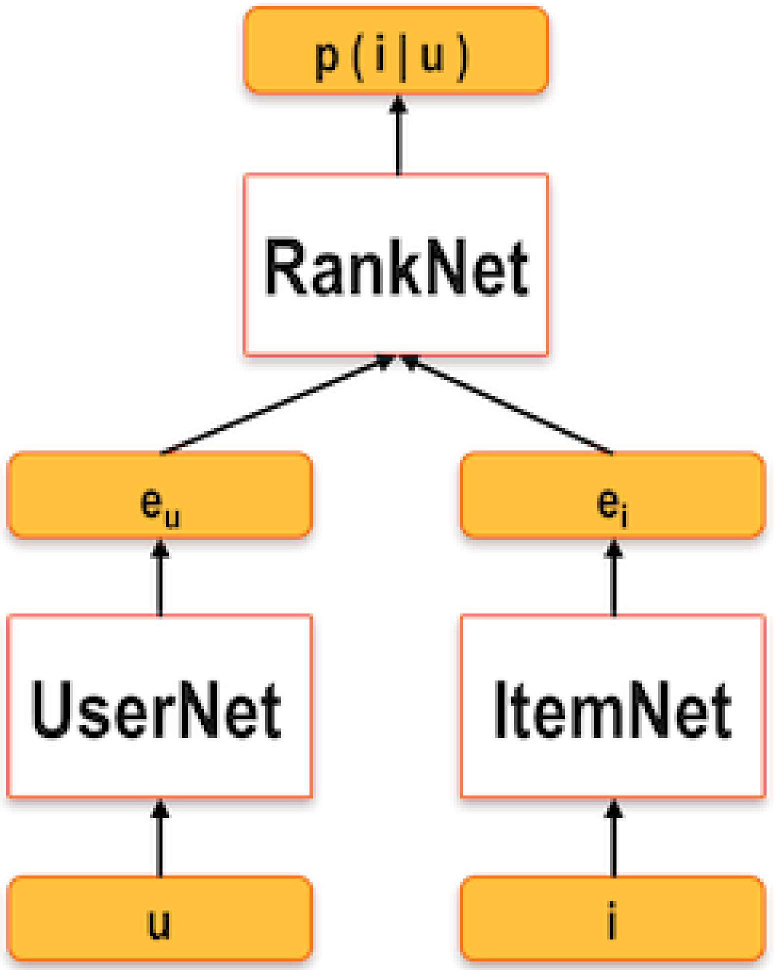


[M. Kurovski]

Preference Prediction Model

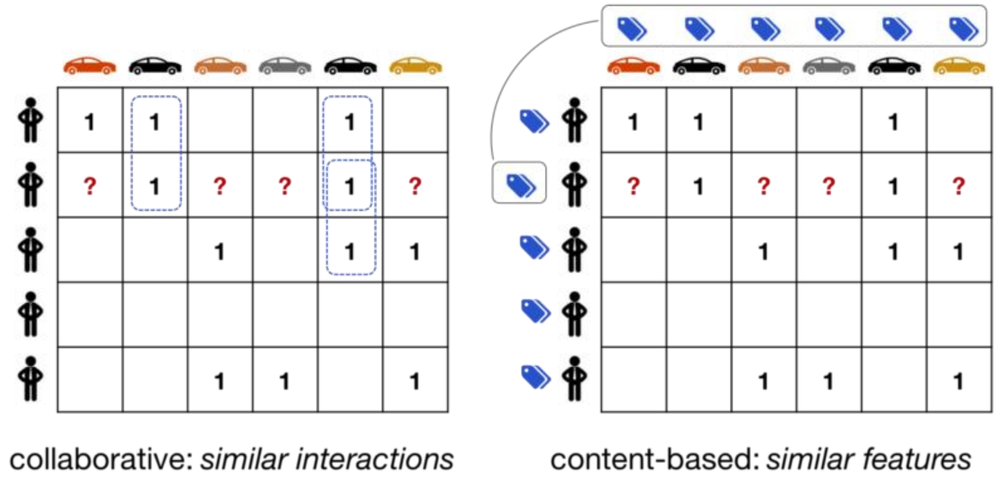
The overall network consists of three subnetworks: **UserNet**, **ItemNet** and **RankNet**.

These networks are combined and trained jointly. Afterwards, we split them to present an overall architecture capable of serving the recommendations in production.



Candidate Generation

- To quickly find candidates that are likely to be relevant for a user, we use approximate nearest neighbor search. Starting with a user embedding as query, we can efficiently fetch the *T* closest items for a specific distance metric, e.g. cosine or Euclidean distance.
- There are many implementations, including Locally Optimized Product Quantizations (LOPQ) from Yahoo or Approximate Nearest Neighbor Oh Yeah (ANNOY) provided by Erik Bernhardsson from Spotify.

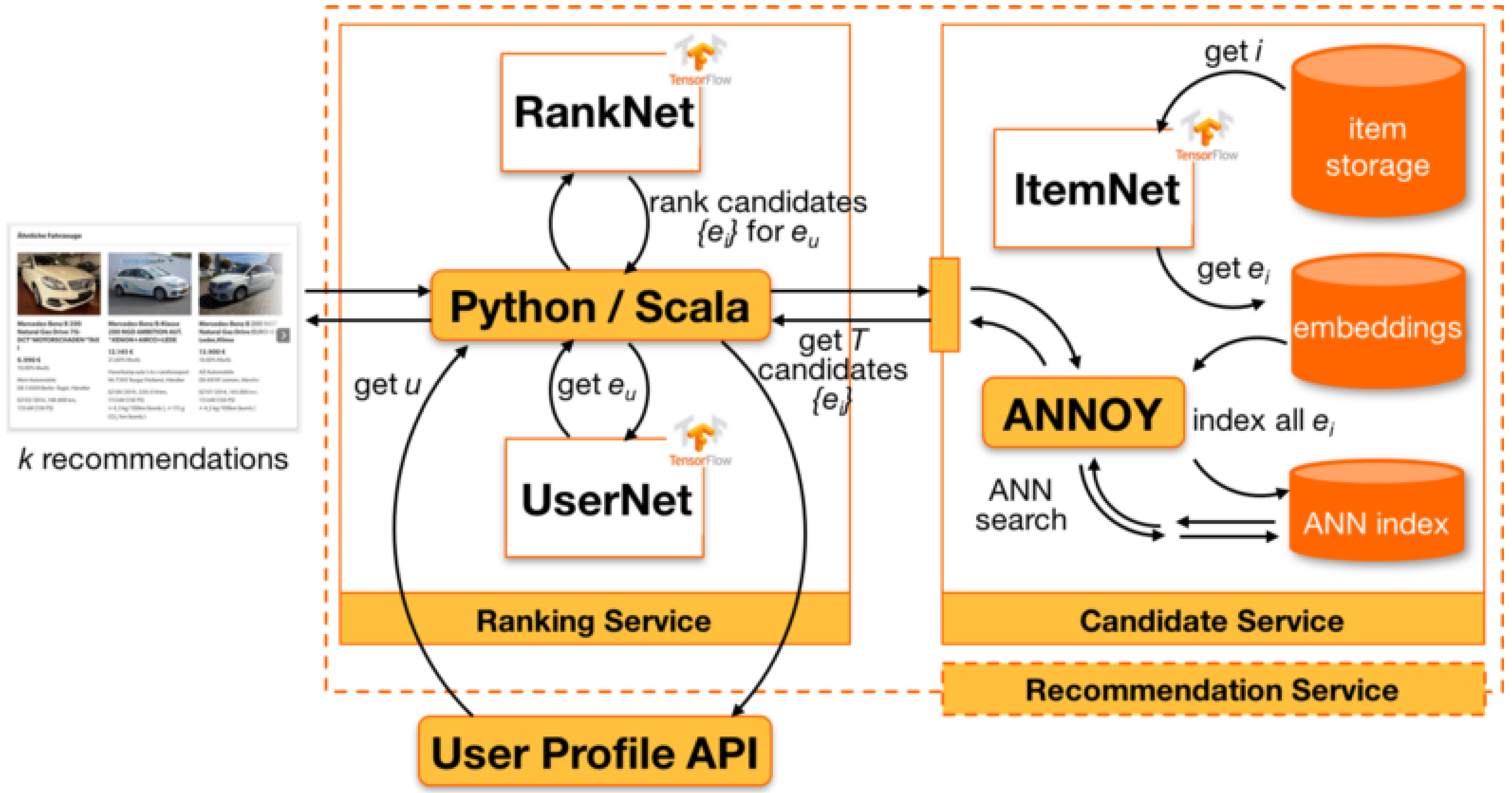


collaborative: *similar interactions*
content-based: *similar features*

[M. Kurovski]

Ranking

- For *T* item candidates for our user, we can use the **RankNet** to score each candidate.
- Finally, we sort the candidates by decreasing score and take the top *k* most promising ones.
- These items are then provided as recommendations

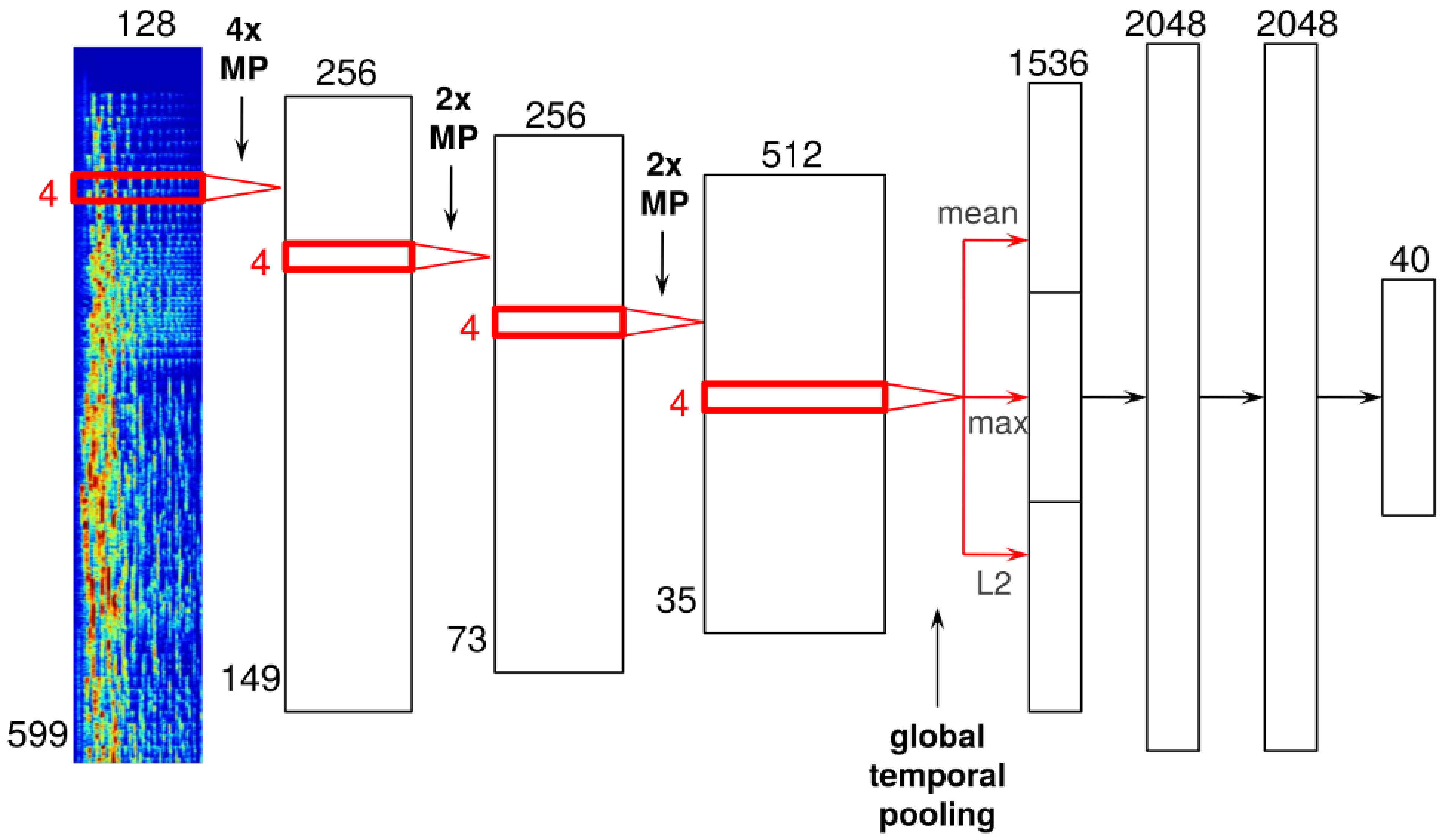


[M. Kurovski]

Deep content-based music recommendation

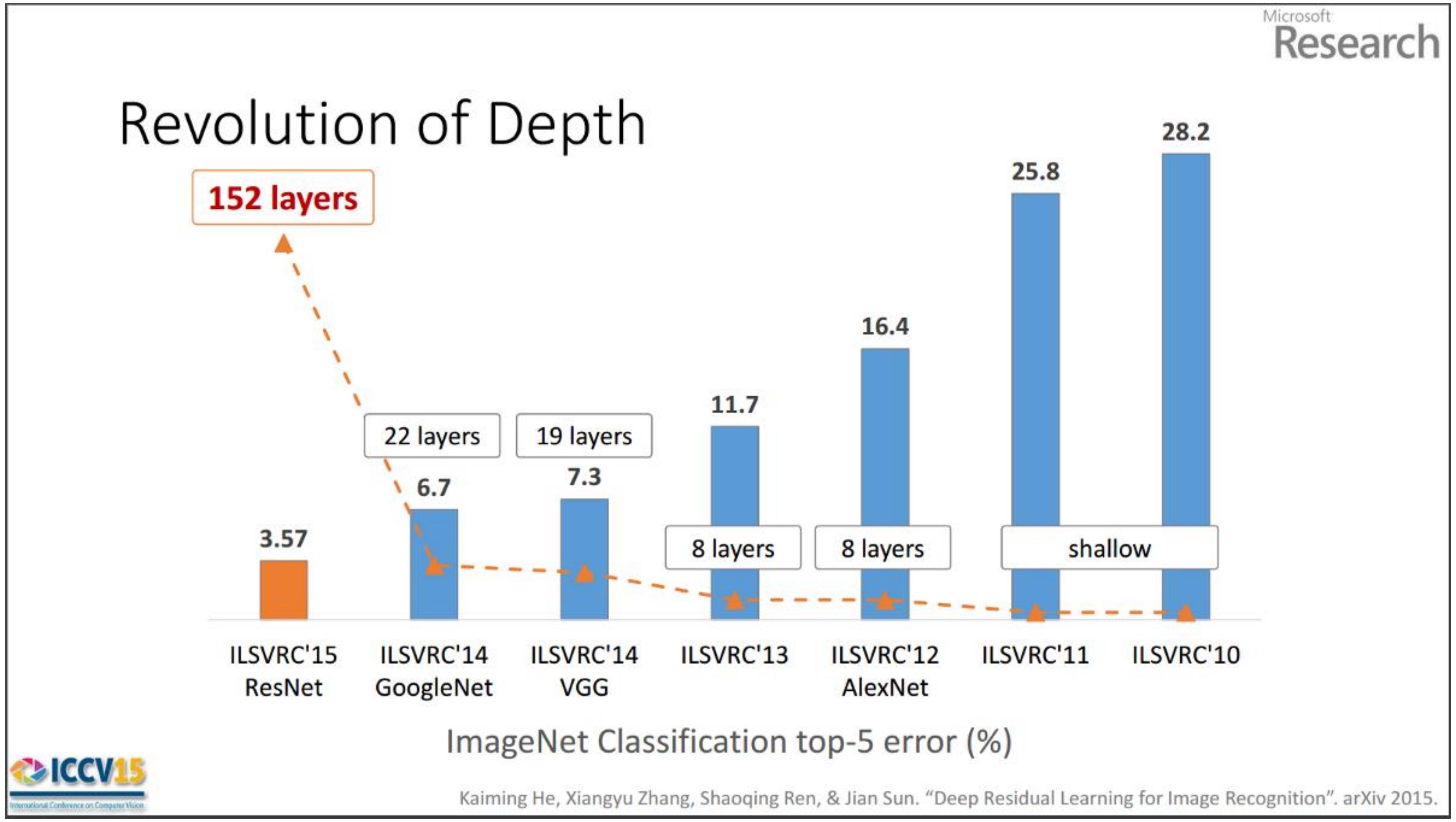
Pioneer work from [Spotify](#) als uses CNNs to extract audio features from music tracks.

The content features could then used to cluster similar tracks and to produce personalized playlists.



https://papers.nips.cc/paper/5004-deep-content-based-music-recommendation.pdf

Is deeper better?

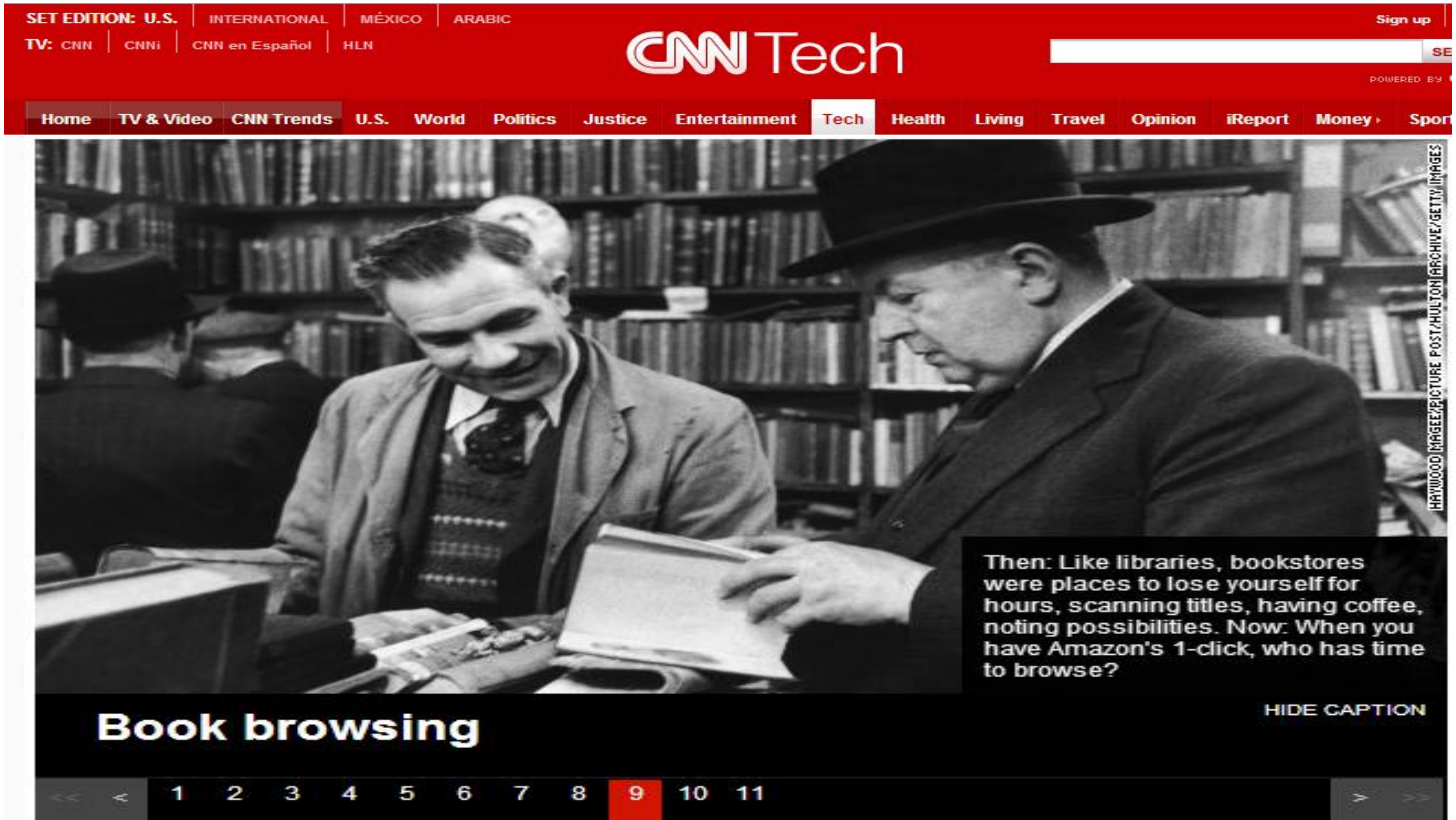


For image classification deeper models with hundreds of layers and novel architecture shave shown impressive improvements reducing the classification error more that 24 percentage points in the last few years.

What about DL for RecSys? are such improvement in recommendation performance possible?

https://medium.com/@libreai/a-glimpse-into-deep-learning-for-recommender-systems-d66ae0681775

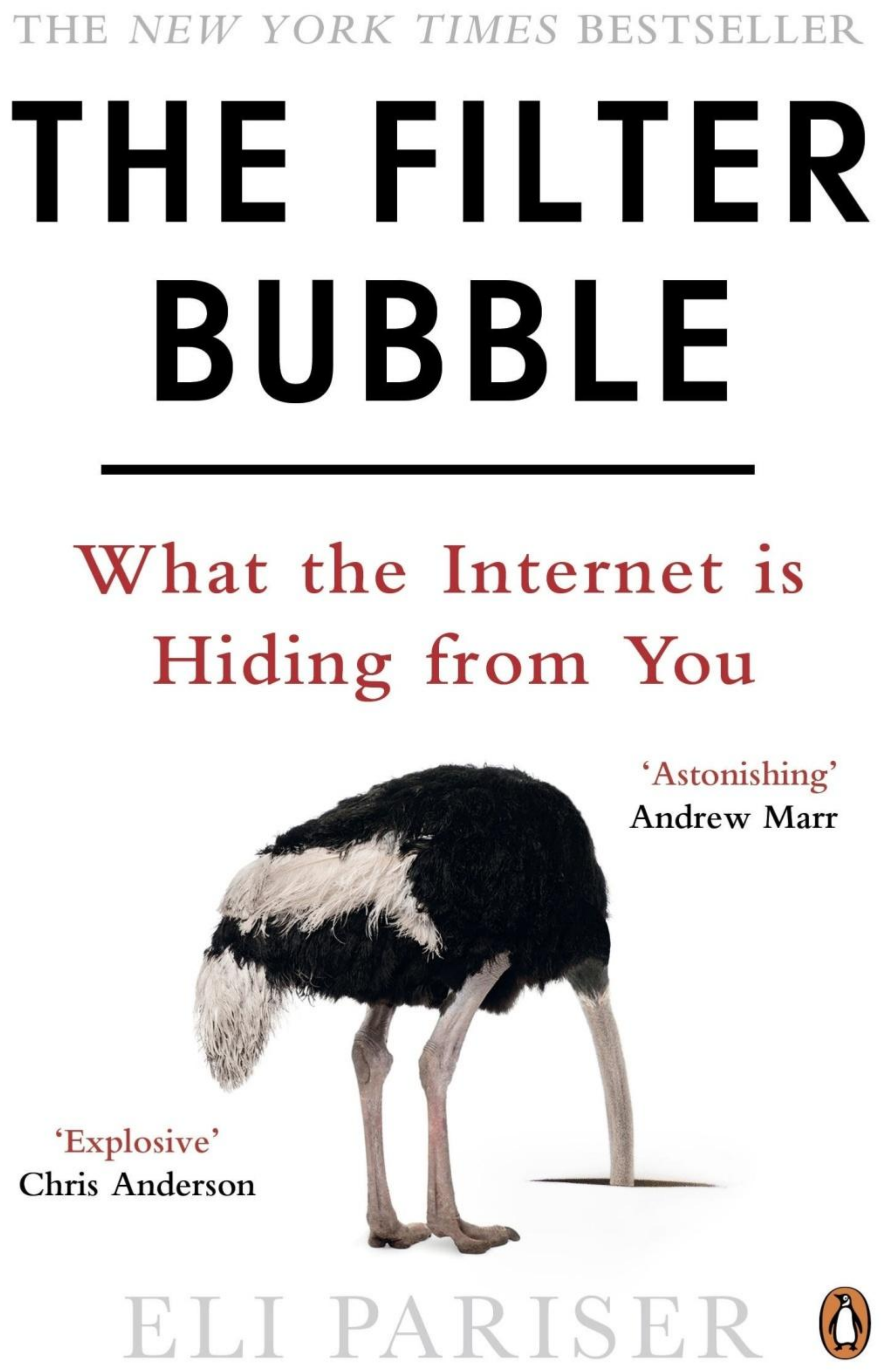
Unexpected & Serendipitous RSes


SET EDITION: U.S. | INTERNATIONAL | MÉXICO | ARABIC
TV: CNN | CNNi | CNN en Español | HLN
CNN Tech
Sign up
SE
POWERED BY
Home TV & Video CNN Trends U.S. World Politics Justice Entertainment Tech Health Living Travel Opinion iReport Money › Spor
HAYWOOD MAGEE/PICTURE POST/HULTON ARCHIVE/GETTY IMAGES
HIDE CAPTION
< 1 2 3 4 5 6 7 8 9 10 11 >

Then: Like libraries, bookstores were places to lose yourself for hours, scanning titles, having coffee, noting possibilities. Now: When you have Amazon's 1-click, who has time to browse?

Book browsing

- "A world constructed from the familiar is a world in which there's nothing to learn ... (since there is) invisible autopropaganda indoctrinating us with our own ideas." *Eli Pariser,* *Economist*, 2011
- "Simplistic" recommender systems can contribute to this filter bubble by recommending obvious and trivial items
- Collaborative filtering systems are characterized by *over-specialization* and *concentration* biases



THE *NEW YORK TIMES* BESTSELLER

THE FILTER BUBBLE

What the Internet is Hiding from You

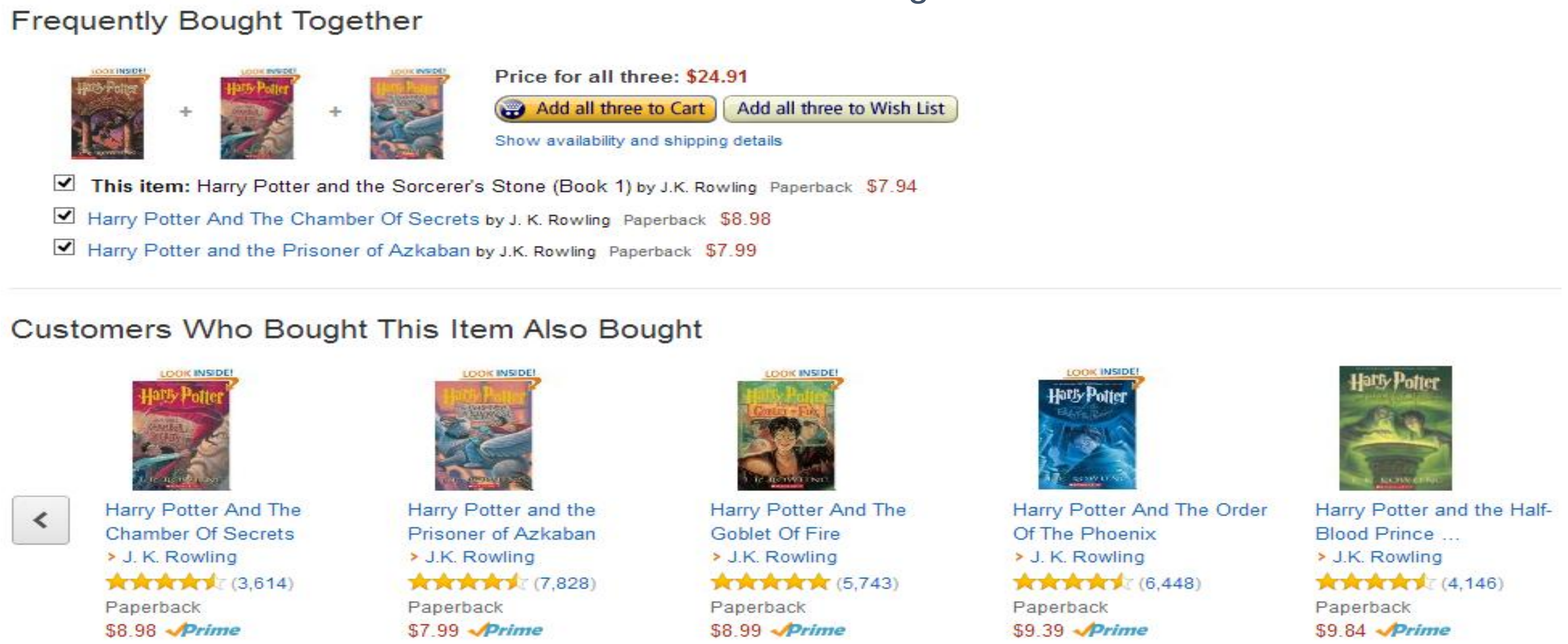
'Astonishing' Andrew Marr

'Explosive' Chris Anderson

ELI PARISER

The Filter Bubble Example

- Problem with accuracy: can lead to boring recommendations



Frequently Bought Together

Price for all three: $24.91

Add all three to Cart | Add all three to Wish List

Show availability and shipping details

- [✓] **This item:** Harry Potter and the Sorcerer's Stone (Book 1) by J.K. Rowling Paperback $7.94
- [✓] Harry Potter And The Chamber Of Secrets by J. K. Rowling Paperback $8.98
- [✓] Harry Potter and the Prisoner of Azkaban by J.K. Rowling Paperback $7.99

Customers Who Bought This Item Also Bought

Harry Potter And The Chamber Of Secrets	Harry Potter and the Prisoner of Azkaban	Harry Potter And The Goblet Of Fire	Harry Potter And The Order Of The Phoenix	Harry Potter and the Half-Blood Prince ...
› J. K. Rowling	› J.K. Rowling	› J.K. Rowling	› J. K. Rowling	› J.K. Rowling
(3,614)	(7,828)	(5,743)	(6,448)	(4,146)
Paperback	Paperback	Paperback	Paperback	Paperback
$8.98 Prime	$7.99 Prime	$8.99 Prime	$9.39 Prime	$9.84 Prime

Serendipity and Unexpectedness: Breaking out of the Filter Bubble

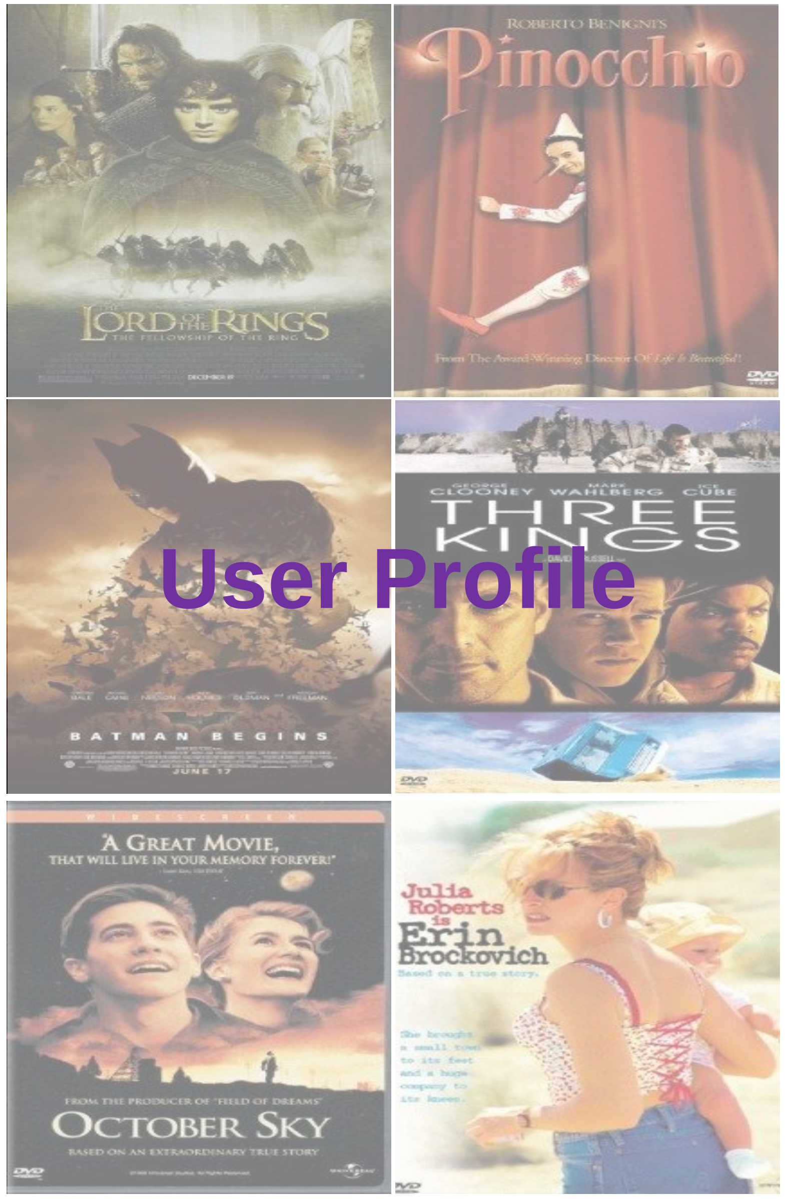
Serendipity: Recommendations of novel items liked by the user that he/she *would not discover autonomously* (accidental discovery)

Unexpectedness: tell me something surprising that goes *against my expectations*

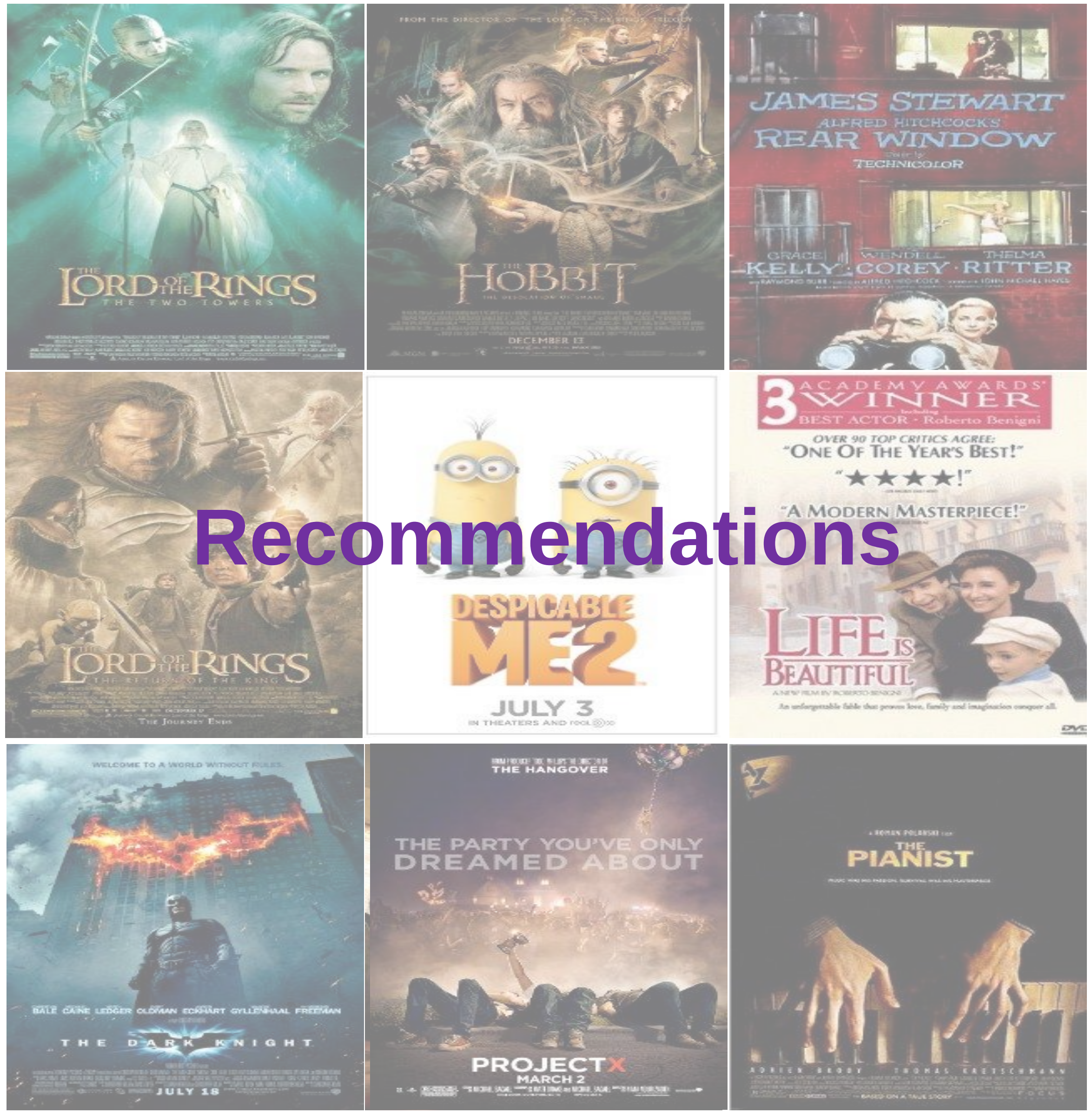
Definition of Unexpectedness

- “If you do not expect it, you will not find the unexpected, for it is hard to find and difficult.” - Heraclitus of Ephesus, 544-484 B.C.
- Idea:
- Define user expectations
- Identify those items that *depart from those expectations*
- Recommend *high quality and unexpected* items to the user

Examples of Unexpected Recommendations


LORD OF THE RINGS
THE FELLOWSHIP OF THE RING
Roberto Benigni's
Pinocchio
BATMAN BEGINS
THREE KINGS
A GREAT MOVIE, THAT WILL LIVE IN YOUR MEMORY FOREVER!
OCTOBER SKY
Julia Roberts is Erin Brockovich

User Profile



Recommendations

Expected Recommendations

- **Expectation set** of a user: a finite collection of items that the user considers as familiar/known/expected.

- Multiple ways to define this set.

Examples of sets of user expectations

Domain	Mechanism	Method
Movies	Past Transactions	Explicit Ratings
	Domain Knowledge	Set of Rules
Books	Past Transactions	Implicit Ratings
	Domain Knowledge	Related Items
	Data Mining	Association Rules

Operationalization of Unexpectedness

- First, we define the **distance** of item i from the set of expected items E_u for user u,
  - $\delta_{u,i} = d(i; E_u)$
- Then, **unexpectedness** of item i with respect to user expectations is defined as some unimodal function Δ of this distance,
  - $\Delta(\delta_{u,i}; \delta_u^*)$, where δ_u^*, the mode of distribution Δ, is the most preferred unexpected distance for user u
- Recommending items with the highest levels of unexpectedness could be unreasonable and problematic

Utility of Recommendations

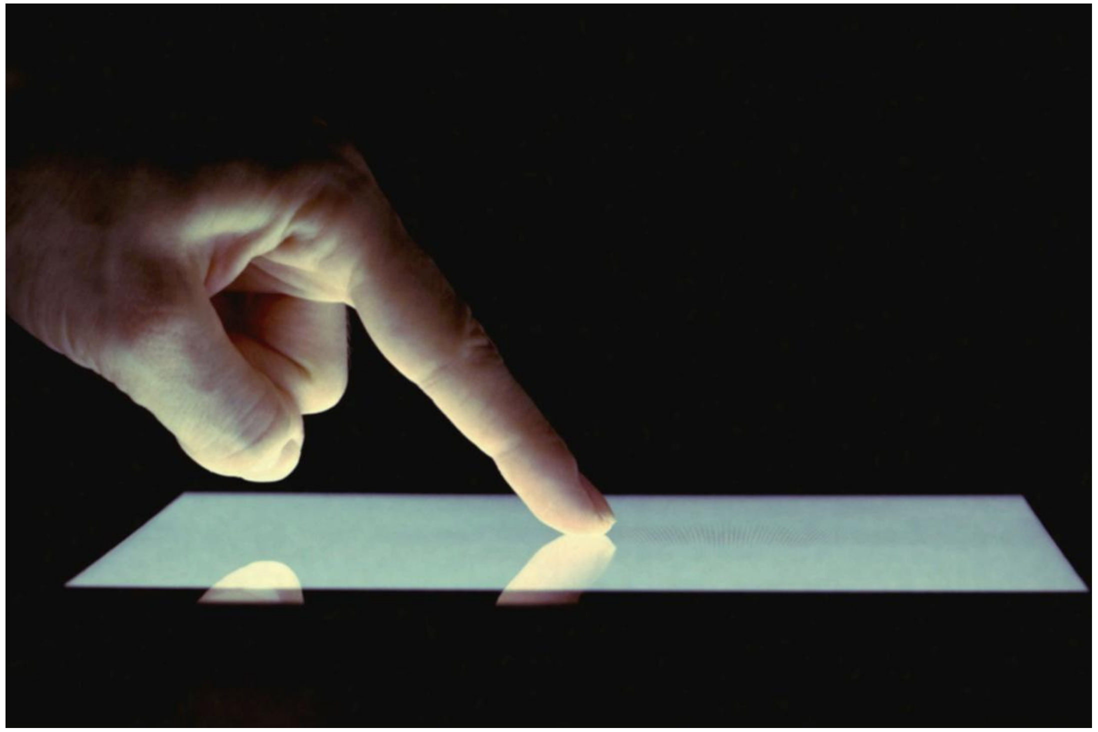
- Unexpectedness alone is not enough for providing useful recommendations. We introduce **utility** of a recommendation as a function of recommendation **quality** *and* its **unexpectedness**

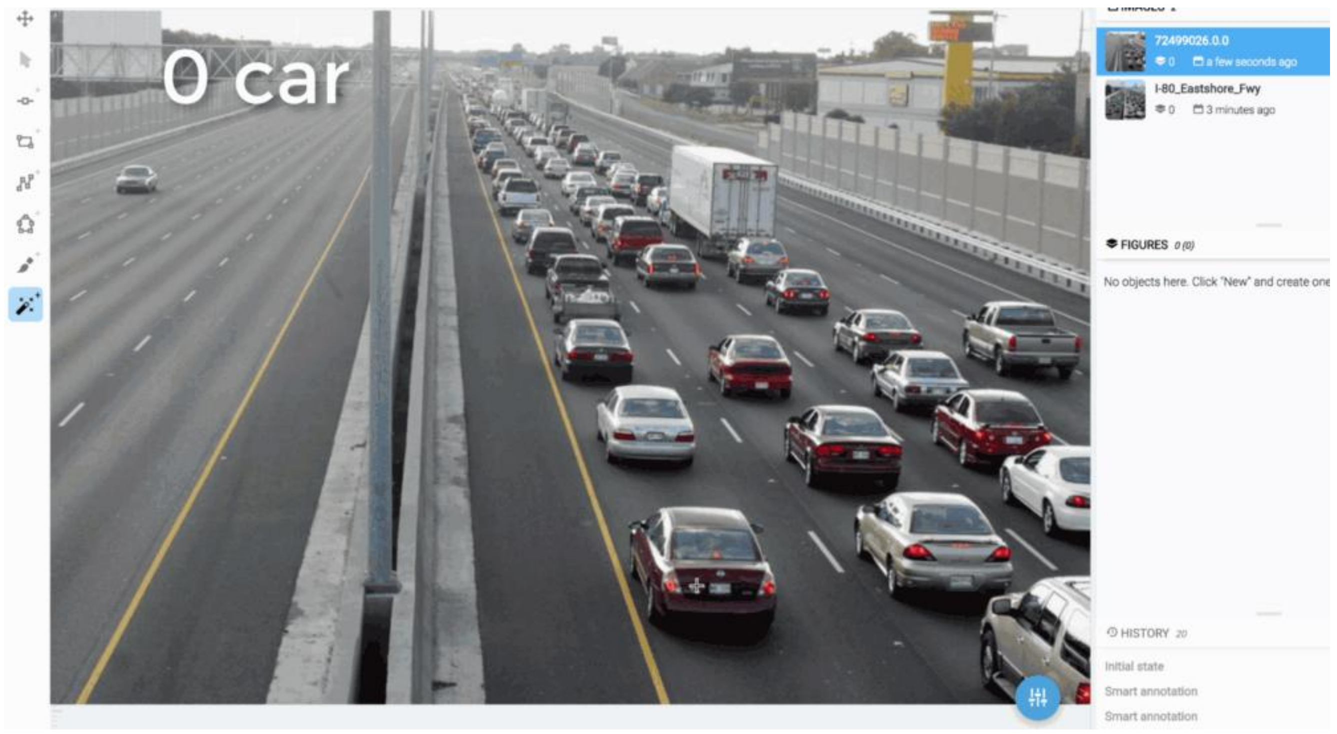
- We estimate the utility of a recommendation using two components, $U_{u,i} = U^{q}_{u,i} - U^{\delta}_{u,i}$:
  - $U^{q}_{u,i}$, the utility of quality, and
  - $U^{\delta}_{u,i}$, the loss in utility by the departure from the preferred level of unexpectedness δ^{*}_{u}

Unexpectedness and the Long Tail

- The “rich gets richer” problem of RSes (a.k.a. the “blockbuster” phenomenon)
  - Many RS algorithms tend to recommend popular items (from the “Head” of the Long Tail distribution), thus reinforcing the “filter bubble” phenomenon…
  - Whereas the *real* “action” is in the Long Tail
- Unexpected recommendations are *more from the Long Tail* because they
  - produce more diverse recommendations
  - do *not* recommend *expected* items from the Head

Tomorrow: Deep Learning for Human-Computer Interaction





Funded by the
Erasmus+ Programme
of the European Union

Thank you.

Inna Skarga-Bandurova
Computer Science and Engineering Depatrment
V. Dahl East Ukrainian National University
skarga_bandurova@snu.edu.ua