

**COMP9321**

# **Data Services Engineering**

**Term 1, 2019**

**Week 8 Lecture 2**

# **Content-based recommendation**

Introduction to Recommender Systems

# Content-based recommendation

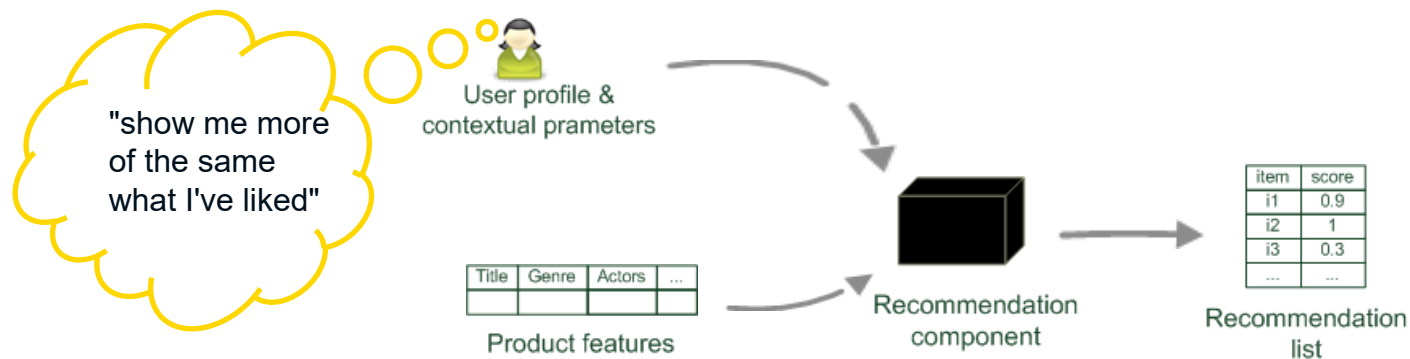
While CF – methods do not require any information about the items,  
– it might be reasonable to exploit such information; and  
– recommend fantasy novels to people who liked fantasy novels in the past

What do we need:

- some information about the available items such as the genre ("content")
- some sort of *user profile* describing what the user likes (the preferences)

The task:

- learn user preferences
- locate/recommend items that are "similar" to the user preferences



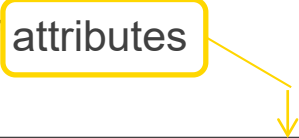
# What is the "content"?

Most CB-recommendation techniques were applied to recommending text documents.

- Like web pages or newsgroup messages for example.

Content of items can also be represented as text documents.

- With textual descriptions of their basic characteristics.
- Structured: Each item is described by the same set of attributes



Title	Genre	Author	Type	Price	Keywords
The Night of the Gun	Memoir	David Carr	Paperback	29.90	Press and journalism, drug addiction, personal memoirs, New York
The Lace Reader	Fiction, Mystery	Brunonia Barry	Hardcover	49.90	American contemporary fiction, detective, historical
Into the Fire	Romance, Suspense	Suzanne Brockmann	Hardcover	45.90	American fiction, murder, neo-Nazism

- Unstructured: free-text description.

# Content representation and item similarities

## Item representation

Title	Genre	Author	Type	Price	Keywords
The Night of the Gun	Memoir	David Carr	Paperback	29.90	Press and journalism, drug addiction, personal memoirs, New York
The Lace Reader	Fiction, Mystery	Brunonia Barry	Hardcover	49.90	American contemporary fiction, detective, historical
Into the Fire	Romance, Suspense	Suzanne Brockmann	Hardcover	45.90	American fiction, murder, neo-Nazism

## User profile

Title	Genre	Author	Type	Price	Keywords
...	Fiction	Brunonia, Barry, Ken Follett	Paperback	25.65	Detective, murder, New York

$keywords(b_j)$   
describes Book  $b_j$   
with a set of  
keywords



## Simple approach

- Compute the similarity of an unseen item with the user profile based on the keyword overlap (e.g. using the Dice coefficient)
- Or use and combine multiple metrics



$$\frac{2 \times |keywords(b_i) \cap keywords(b_j)|}{|keywords(b_i)| + |keywords(b_j)|}$$

# Term-Frequency - Inverse Document Frequency ( $TF - IDF$ )

Simple keyword representation has its problems

- in particular when automatically extracted as
  - not every word has similar importance
  - longer documents have a higher chance to have an overlap with the user profile

Standard measure: TF-IDF

- Encodes text documents in multi-dimensional Euclidian space
  - weighted term vector
- TF: Measures, how often a term appears (density in a document)
  - assuming that important terms appear more often
  - normalization has to be done in order to take document length into account
- IDF: Aims to reduce the weight of terms that appear in all documents

# TF-IDF II

Given a keyword  $i$  and a document  $j$

$TF(i, j)$

–term frequency of keyword  $i$  in document  $j$

$IDF(i)$

–inverse document frequency calculated as  $IDF(i) = \log \frac{N}{n(i)}$

»  $N$  : number of all recommendable documents

»  $n(i)$  : number of documents from  $N$  in which keyword  $i$  appears

$TF - IDF$

• is calculated as:  $TF-IDF(i, j) = TF(i, j) * IDF(i)$

# Example TF-IDF representation

Term frequency:

- Each document is a **count vector** in  $\mathbb{N}^{|v|}$

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	157	73	0	0	0	0
Brutus	4	157	0	1	0	0
Caesar	232	227	0	2	1	1
Calpurnia	0	10	0	0	0	0
Cleopatra	57	0	0	0	0	0
mercy	1.51	0	3	5	5	1
worser	1.37	0	1	1	1	0

Vector  $v$  with dimension  $|v| = 7$



# Example TF-IDF representation

## Combined TF-IDF weights

- Each document is now represented by a real-valued vector of *TF-IDF* weights  $\in \mathbb{R}^{|v|}$

		Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth	
Antony	157	73	0	0	0	0		
Brutus	4		Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Caesar	232							
Calpurnia	0							
Cleopatra	57							
mercy	1.5	Antony	5.25	3.18	0	0	0	0.35
worser	1.3	Brutus	1.21	6.1	0	1	0	0
		Caesar	8.59	2.54	0	1.51	0.25	0
		Calpurnia	0	1.54	0	0	0	0
		Cleopatra	2.85	0	0	0	0	0
		mercy	1.51	0	1.9	0.12	5.25	0.88
		worser	1.37	0	0.11	4.15	0.25	1.95

# Improving the vector space model

Vectors are usually long and sparse

remove stop words

- They will appear in nearly all documents.
- e.g. "a", "the", "on", ...

use stemming

- Aims to replace variants of words by their common stem
- e.g. "went" → "go", "stemming" → "stem", ...

size cut-offs

- only use top n most representative words to remove "noise" from data
- e.g. use top 100 words

# Improving the vector space model II

Use lexical knowledge, use more elaborate methods for feature selection

- Remove words that are not relevant in the domain

Detection of phrases as terms

- More descriptive for a text than single words
- e.g. "United Nations"

Limitations

- semantic meaning remains unknown
- example: usage of a word in a negative context
  - "there is nothing on the menu that a vegetarian would like.."
  - The word "vegetarian" will receive a higher weight then desired
    - ➡ an unintended match with a user interested in vegetarian restaurants

# Cosine similarity

Usual similarity metric to compare vectors: Cosine similarity (angle)

- Cosine similarity is calculated based on the angle between the vectors

$$\text{sim}(\vec{a}, \vec{b}) = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}| * |\vec{b}|}$$

Adjusted cosine similarity

- take average user ratings into account ( $\bar{r}_u$ ), transform the original ratings
- U: set of users who have rated both items a and b

$$\text{sim}(\vec{a}, \vec{b}) = \frac{\sum_{u \in U} (r_{u,a} - \bar{r}_u)(r_{u,b} - \bar{r}_u)}{\sqrt{\sum_{u \in U} (r_{u,a} - \bar{r}_u)^2} \sqrt{\sum_{u \in U} (r_{u,b} - \bar{r}_u)^2}}$$

# Recommending items

## Simple method: nearest neighbors

- Given a set of documents  $D$  already rated by the user (like/dislike)
  - Either explicitly via user interface
  - Or implicitly by monitoring user's behavior
- Find the  $n$  nearest neighbors of an not-yet-seen item  $i$  in  $D$ 
  - Use similarity measures (like cosine similarity) to capture similarity of two documents
- Take these neighbors to predict a rating for  $i$ 
  - e.g.  $k = 5$  most similar items to  $i$ .  
4 of  $k$  items were liked by current user → item  $i$  will also be liked by this user
- Variations:
  - Varying neighborhood size  $k$
  - lower/upper similarity thresholds to prevent system from recommending items the user already has seen
- Good to model short-term interests / follow-up stories
- Used in combination with method to model long-term preferences

# Limitations of content-based recommendation methods

Keywords alone may not be sufficient to judge quality/relevance of a document or web page

- up-to-date-ness, usability, aesthetics, writing style
- content may also be limited / too short
- content may not be automatically extractable (multimedia)

Ramp-up phase required

- Some training data is still required
- Web 2.0: Use other sources to learn the user preferences

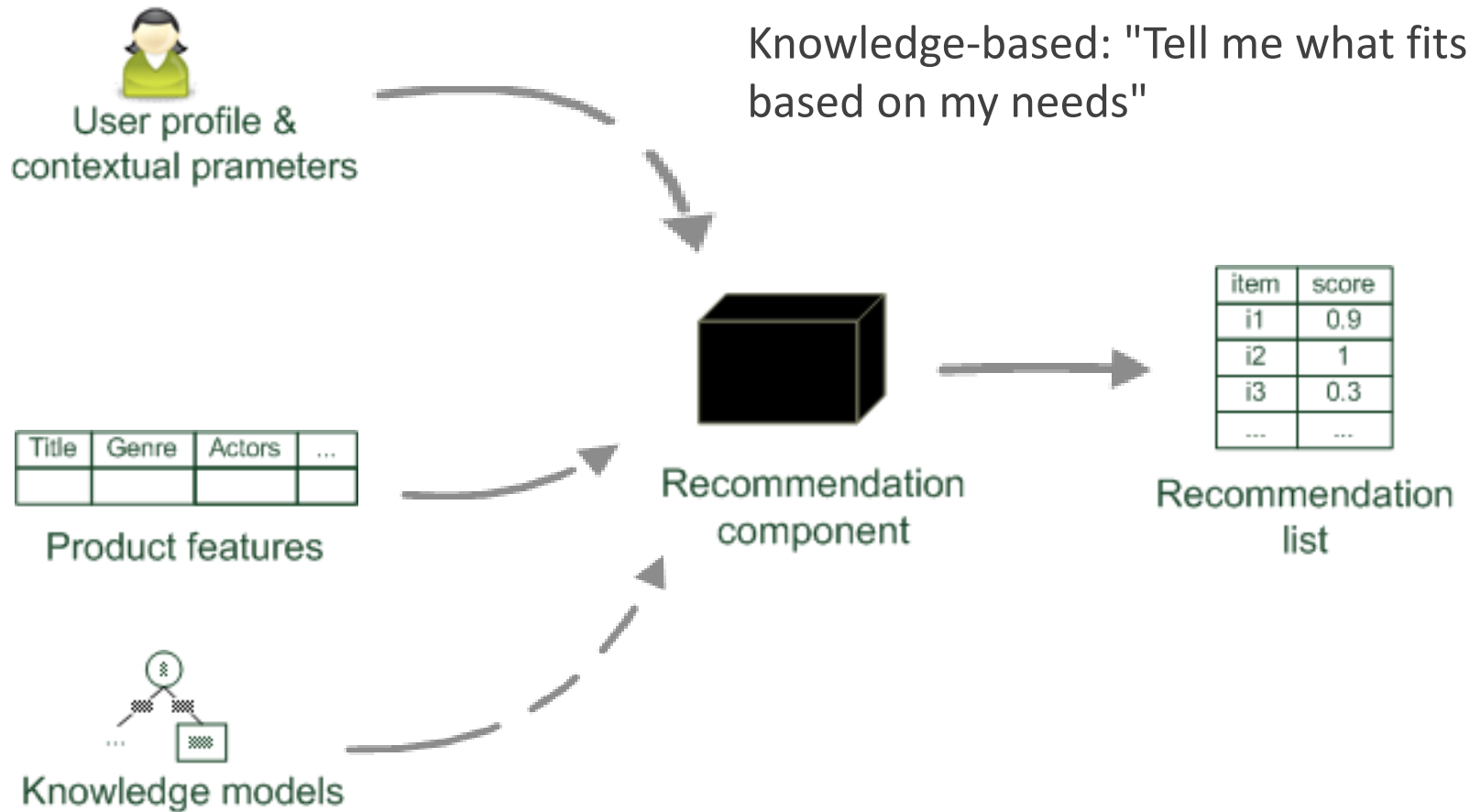
Overspecialization

- Algorithms tend to propose "more of the same"
- Or: too similar news items

# Knowledge-based recommendation

Introduction to Recommender Systems

# Basic I/O Relationship





# Why do we need knowledge-based recommendation?

Products with low number of available ratings



Time span plays an important role

- five-year-old ratings for computers
- user lifestyle or family situation changes

Customers want to define their requirements explicitly

- "the color of the car should be black"

# Knowledge-based recommender systems

## Constraint-based

- based on explicitly defined set of recommendation rules
- fulfill recommendation rules

## Case-based

- based on different types of similarity measures
- retrieve items that are similar to specified requirements

Both approaches are similar in their **conversational** recommendation process

- users specify the requirements
- systems try to identify solutions
- if no solution can be found, users change requirements

# Constraint-based recommender systems

## Knowledge base

- usually mediates between user model and item properties
- variables
  - user model features (requirements), Item features (catalogue)
- set of constraints
  - logical implications (IF user requires A THEN proposed item should possess feature B)
  - hard and soft/weighted constraints
  - solution preferences

## Derive a set of recommendable items

- fulfilling set of applicable constraints
- applicability of constraints depends on current user model
- explanations – transparent line of reasoning

# Constraint-based recommendation tasks

Find a set of user requirements such that a subset of items fulfills all constraints

- ask user which requirements should be relaxed/modified such that some items exist that do not violate any constraint

Find a subset of items that satisfy the maximum set of weighted constraints

- similar to find a maximally succeeding subquery (XSS)
- all proposed items have to fulfill the same set of constraints
- compute relaxations based on predetermined weights

Rank items according to weights of satisfied soft constraints

- rank items based on the ratio of fulfilled constraints
- does not require additional ranking scheme

# Constraint-based recommendation problem

Select items from this catalog that match the user's requirements

id	price(€)	mpix	opt-zoom	LCD-size	movies	sound	waterproof
P <sub>1</sub>	148	8.0	4×	2.5	no	no	yes
P <sub>2</sub>	182	8.0	5×	2.7	yes	yes	no
P <sub>3</sub>	189	8.0	10×	2.5	yes	yes	no
P <sub>4</sub>	196	10.0	12×	2.7	yes	no	yes
P <sub>5</sub>	151	7.1	3×	3.0	yes	yes	no
P <sub>6</sub>	199	9.0	3×	3.0	yes	yes	no
P <sub>7</sub>	259	10.0	3×	3.0	yes	yes	no
P <sub>8</sub>	278	9.1	10×	3.0	yes	yes	yes

User's requirements can, for example, be

- "the price should be lower than 300 €"
- "the camera should be suited for sports photography"

# Case-based recommender systems

Items are retrieved using similarity measures

Distance similarity

$$\text{similarity}(p, REQ) = \frac{\sum_{r \in REQ} w_r * \text{sim}(p, r)}{\sum_{r \in REQ} w_r}$$

Def.

- $\text{sim}(p, r)$  expresses for each item attribute value  $\phi_r(p)$  its distance to the customer requirement  $r \in REQ$ .
- $w_r$  is the importance weight for requirement  $r$

In real world, customer would like to

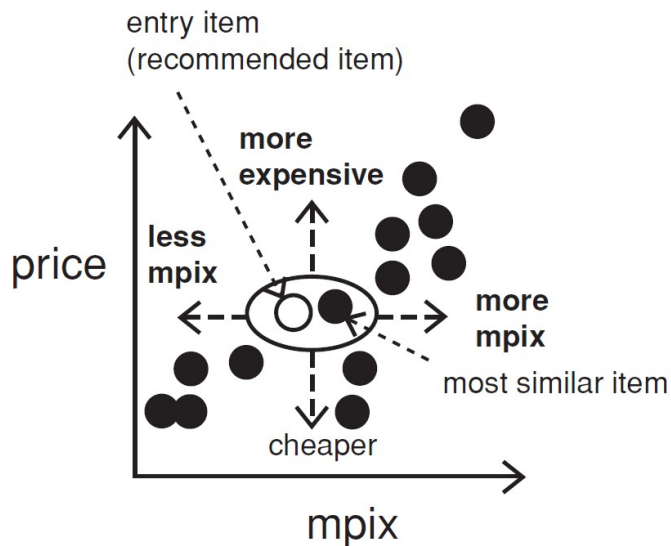
- maximize certain properties. i.e. resolution of a camera, "more is better"(MIB)
- minimize certain properties. i.e. price of a camera, "less is better"(LIB)

# Interacting with case-based recommenders

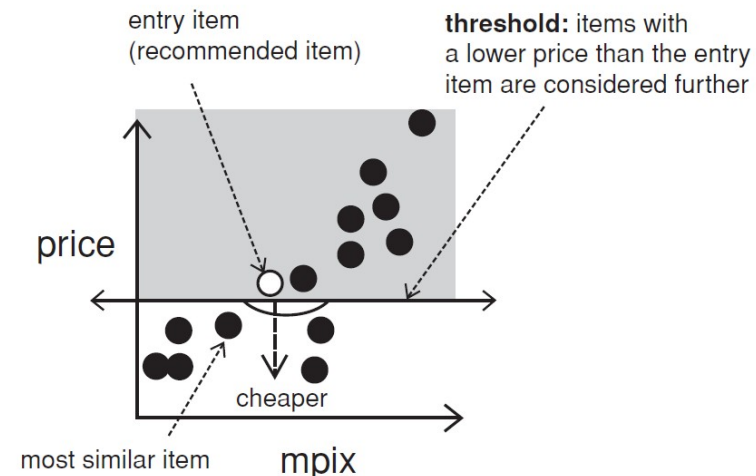
Customers maybe not know what they are seeking

Critiquing is an effective way to support such navigations

Customers specify their change requests (*price or mpix*) that are not satisfied by the current item (*entry item*)

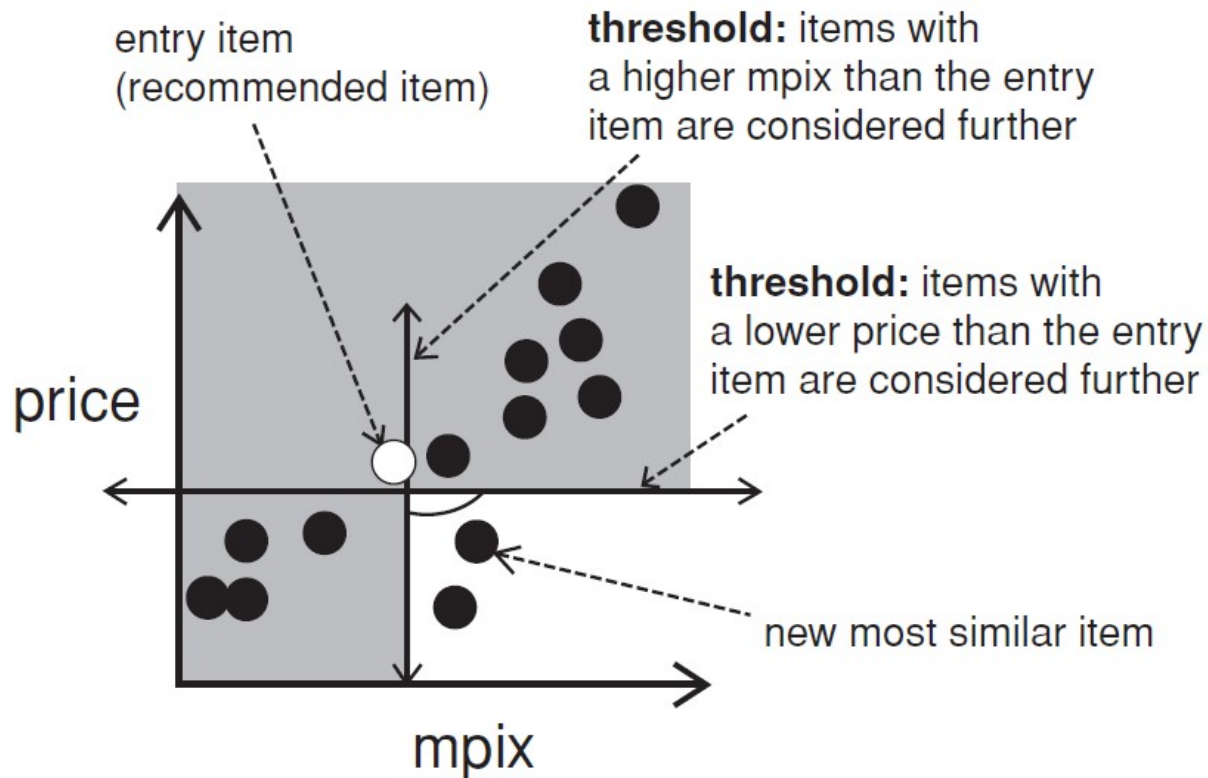


*Critique on price*



# Compound critiques

Operate over multiple properties can improve the efficiency of recommendation dialogs





# Dynamic critiques

## Association rule mining

## Basic steps for dynamic critiques

- $q$ : initial set of requirements
- $CI$ : all the available items
- $K$ : maximum number of compound critiques
- $\sigma_{min}$ : minimum support value for calculated association rules.

Algorithm 4.4 DynamicCritiquing( $q, CI$ )  
Input: Initial user query  $q$ ; Candidate items  $CI$ ;  
number of compound critiques per cycle  $k$ ;  
minimum support for identified association rules

$\sigma_{min}$

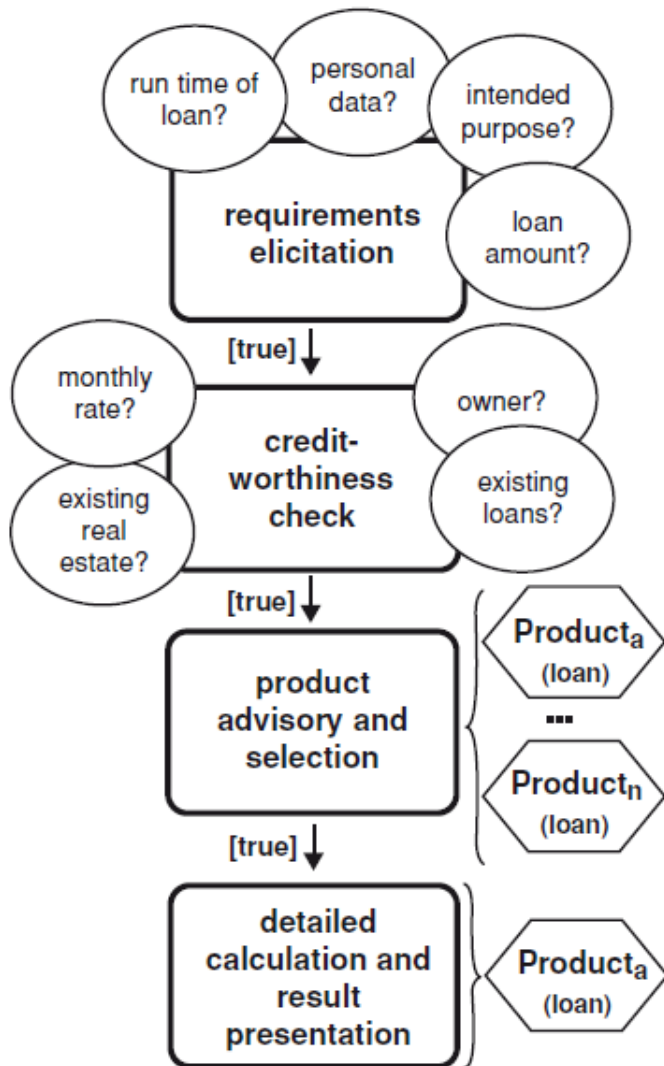
```
procedure DynamicCritiquing( $q, CI, k, \sigma_{min}$ )  
  repeat  
     $r \leftarrow \text{ItemRecommend}(q, CI)$ ;  
     $CC \leftarrow \text{CompoundCritiques}(r, CI, k, \sigma_{min})$ ;  
     $q \leftarrow \text{UserReview}(r, CI, CC)$ ;  
  until empty( $q$ )  
end procedure
```

```
procedure ItemRecommend( $q, CI$ )  
   $CI \leftarrow \{ci \in CI: \text{satisfies}(ci, q)\}$ ;  
   $r \leftarrow \text{mostsimilar}(CI, q)$ ;  
  return  $r$ ;  
end procedure
```

```
procedure UserReview( $r, CI, CC$ )  
   $q \leftarrow \text{critique}(r, CC)$ ;  
   $CI \leftarrow CI - r$ ;  
  return  $q$ ;  
end procedure
```

```
procedure CompoundCritiques( $r, CI, k, \sigma_{min}$ )  
   $CP \leftarrow \text{CritiquePatterns}(r, CI)$ ;  
   $CC \leftarrow \text{Apriori}(CP, \sigma_{min})$ ;  
   $SC \leftarrow \text{SelectCritiques}(CC, k)$ ;  
  return  $SC$ ;  
end procedure
```

# Example: sales dialogue financial services



## In the financial services domain

- sales representatives do not know which services should be recommended
- improve the overall productivity of sales representatives

## Resembles call-center scripting

- best-practice sales dialogues
- states, transitions with predicates

## Research results

- support for KA and validation
  - node properties (reachable, extensible, deterministic)

# Example software: VITA sales support

**VITA - Virtuelle Beratung Kombiprodukte**

Eingelogg: Administrator

**current user of VITA**

**recommendation process: requirements identification, creditworthiness check, ... , result presentation**

Bedarfsermittlung → Bonitätsprüfung → Bausparen → Ergebnis

**one product recommendation**

**requirements articulated by the customer**

**explanation as to why the product is recommended**

**further details regarding product**

**Funktionen**  
Beratungen verwalten:  
-- Bitte auswählen --  
■ Beratung speichern  
■ Beratungsprotokoll

**Kundenanforderungen:**

- Kreditsumme: 1,00 (in Mio HUF)
- Kundenalter: 37 Jahre
- Auszahlungszeitpunkt: 2007.03.01
- Summe Monatsraten: 20.000 HUF
- Laufzeit: 180 (in Monaten)
- Verwendungszweck: Um eine Neuwohnung zu kaufen
- Kreditwährung: in HUF
- Bausparsumme: 2.460.000 HUF

**Daher wurden folgende Produkte ermittelt:**

■ Duo - gefördert - Staat. Zins  
Értjük egymást.

<b>Kombiprodukt:</b>	1.259.008 HUF	<b>Anbieter:</b>	Erste Bank
<b>Bankdarlehen:</b>	1.577.379 HUF	<b>Zinssatz+Gebühr:</b>	4,49%+2,28%
<b>monat. Belastung 1. Jahr:</b>	25.642 HUF	<b>Bauspargebühr:</b>	37.250 HUF

■ Produkt-Details      ■ Warum dieses Produkt?

■ Duo - gefördert - Pfandbrief  
Értjük egymást.

<b>Kombiprodukt:</b>	1.325.258 HUF	<b>Anbieter:</b>	Erste Bank
<b>Bankdarlehen:</b>	1.725.489 HUF	<b>Zinssatz+Gebühr:</b>	5,99%+2,28%
<b>monat. Belastung 1. Jahr:</b>	26.892 HUF	<b>Bauspargebühr:</b>	37.250 HUF

■ Produkt-Details      ■ Warum dieses Produkt?

« ZURÜCK

LOGOUT   NEUSTART   HILFE   FEEDBACK   IMPRESSUM

**Fundamenta Lakáskassza** Alap, amelyre építhet

# Example: Critiquing

*Find your  
Favourite restaurant*



In Vienna you chose:

+43 1 123 123 123

**Biergasthof**

Mariahilferstrasse 123,  
1010 Wien

30€-50€

Local cuisine

local food, central in the city, weekend brunch, room with a view,  
famous for beer, seasonal dishes, group bookings, open all day

For Graz we recommend:

+43 316 45 45 45

**Brauhof**

Brauhofstrasse 45,  
8023 Graz

30€-50€

Local cuisine

local food, own beer, weekend lunch, open all day, private function room,  
famous for beer, seasonal dishes, group bookings, good transport connection

Less \$\$

Nicer

Cuisine

More Quiet

Traditional

Creative

Livelier

Similarity-based navigation in item space

Compound critiques

- more efficient navigation than with unit critiques
- mining of frequent patterns

Dynamic critiques

- only applicable compound critiques proposed

Incremental critiques

- considers history

Adaptive suggestions

- suggest items that allow to best refine user's preference model

# Summary

## Knowledge-based recommender systems

- constraint-based
- case-based

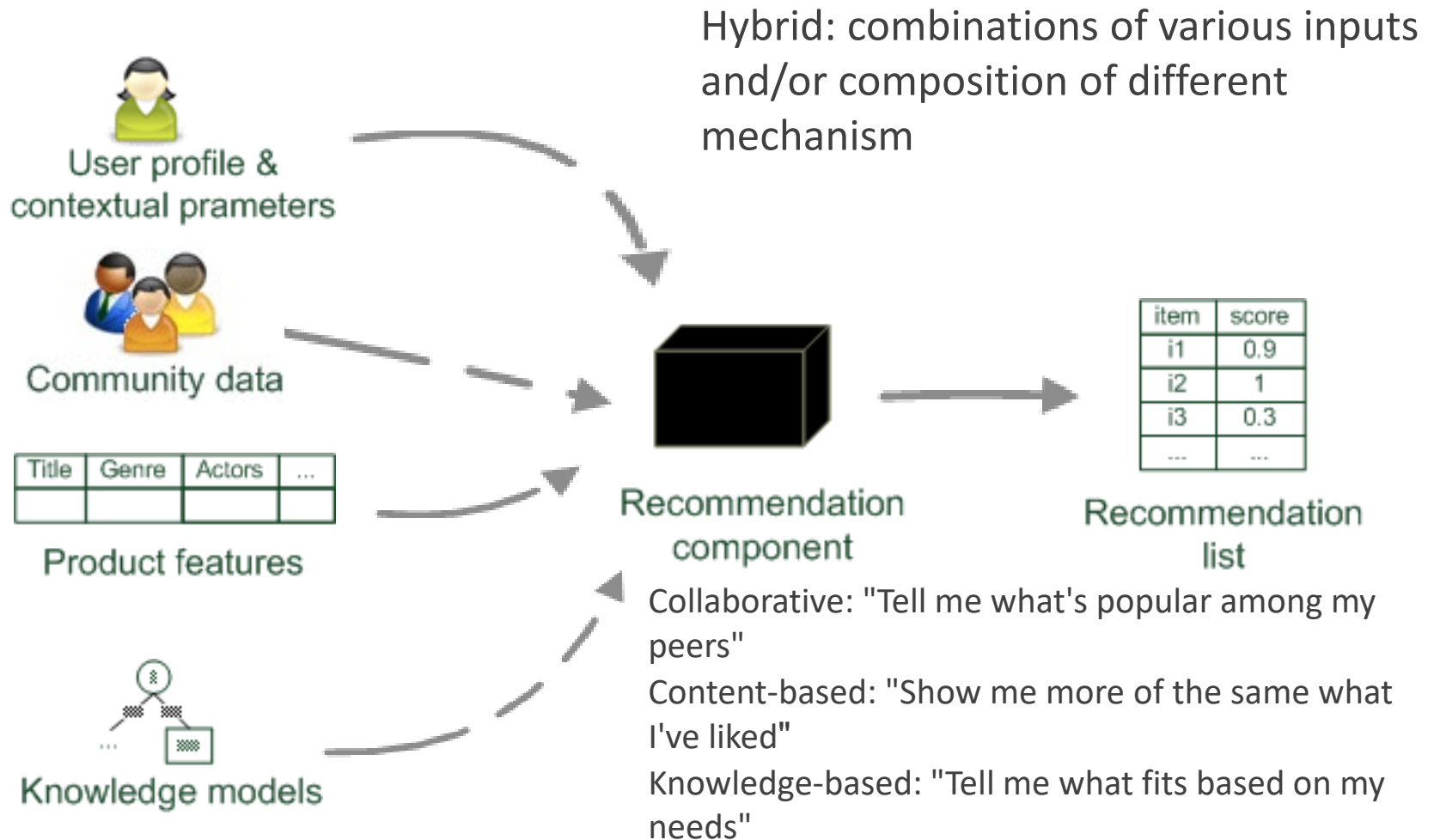
## Limitations

- cost of knowledge acquisition
  - from domain experts
  - from users
  - from web resources
- accuracy of preference models
  - very fine granular preference models require many interaction cycles
  - collaborative filtering models preference implicitly
- independence assumption can be challenged
  - preferences are not always independent from each other

# Hybrid recommendation approaches

Introduction to Recommender Systems

# Hybrid recommender systems



# Hybrid recommender systems

All three base techniques are naturally incorporated by a good sales assistant (at different stages of the sales act) **but** have their shortcomings

- For instance, cold start problems

Idea of crossing two (or more) species/implementations

- *hybrida* [lat.]: denotes an object made by combining two different elements
- Avoid some of the shortcomings
- Reach desirable properties not (or only inconsistently) present in parent individuals

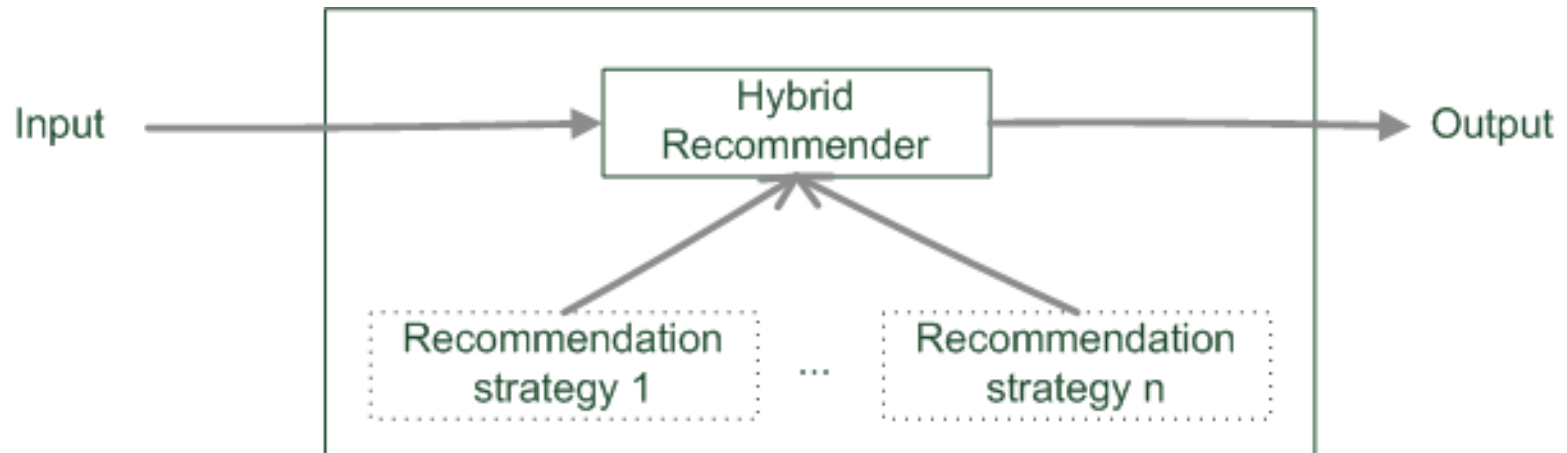
Different hybridization designs

- Parallel use of several systems
- Monolithic exploiting different features
- Pipelined invocation of different systems



# Monolithic hybridization design

Only a single recommendation component



Hybridization is "virtual" in the sense that

- Features/knowledge sources of different paradigms are combined

# Monolithic hybridization designs: Feature combination

Combination of several knowledge sources

- E.g.: Ratings and user demographics or explicit requirements and needs used for similarity computation

"Hybrid" content features:

- Social features: Movies liked by user
  - Content features: Comedies liked by user, dramas liked by user
  - Hybrid features: user likes many movies that are comedies, ...
- 
- *“the common knowledge engineering effort that involves inventing good features to enable successful learning”* [Chumki Basuet al. 1998]

# Monolithic hybridization designs: Feature augmentation

Content-boosted collaborative filtering [Prem Melville et al. 2002]

- Based on content features additional ratings are created
- E.g. Alice likes Items 1 and 3 (unary ratings)
  - Item7 is similar to 1 and 3 by a degree of 0.75
  - Thus Alice likes Item7 by 0.75
- Item matrices become less sparse
- Significance weighting and adjustment factors
  - Peers with more co-rated items are more important
  - Higher confidence in content-based prediction, if higher number of own ratings

Recommendation of research papers [Roberto Torres et al. 2004]

- Citations interpreted as collaborative recommendations

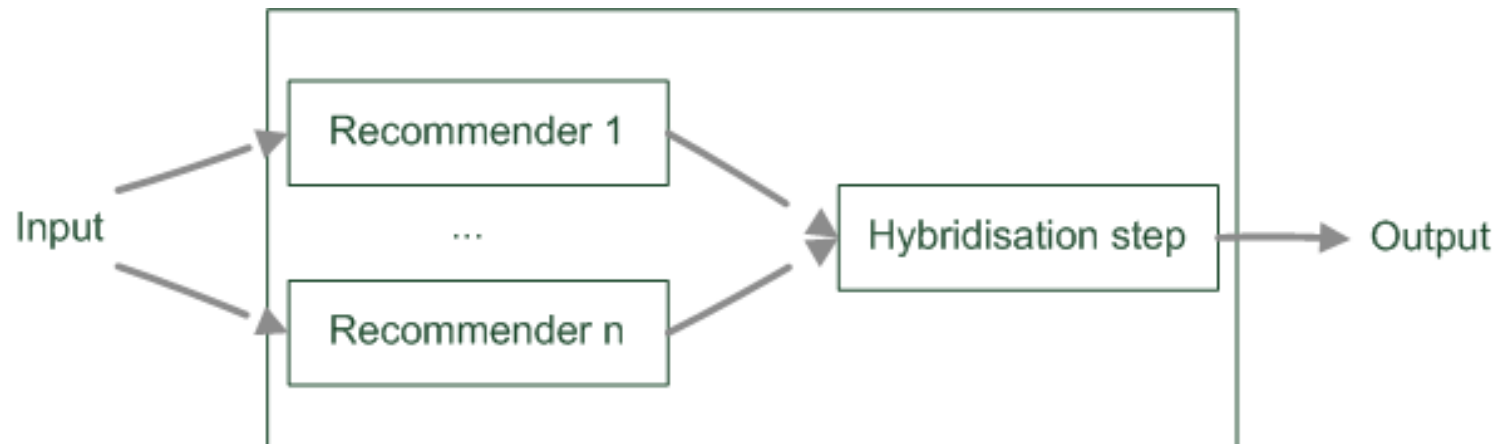
# Parallelized hybridization design

Output of several existing implementations combined

Least invasive design

Some weighting or voting scheme

- Weights can be learned dynamically
- Extreme case of dynamic weighting is switching



# Parallelized hybridization design: Weighted

- Compute weighted sum:  $rec_{weighted}(u,i) = \sum_{k=1}^n \beta_k \times rec_k(u,i)$

Recommender 1		
Item1	0.5	1
Item2	0	
Item3	0.3	2
Item4	0.1	3
Item5	0	

Recommender 2		
Item1	0.8	2
Item2	0.9	1
Item3	0.4	3
Item4	0	
Item5	0	

Recommender weighted(0.5:0.5)		
Item1	0.65	1
Item2	0.45	2
Item3	0.35	3
Item4	0.05	4
Item5	0.00	

# Parallelized hybridization design: Weighted

BUT, how to derive weights?

- Estimate, e.g. by empirical bootstrapping
- Dynamic adjustment of weights

## Empirical bootstrapping

- Historic data is needed
- Compute different weightings
- Decide which one does best

## Dynamic adjustment of weights

- Start with for instance uniform weight distribution
- For each user adapt weights to minimize error of prediction

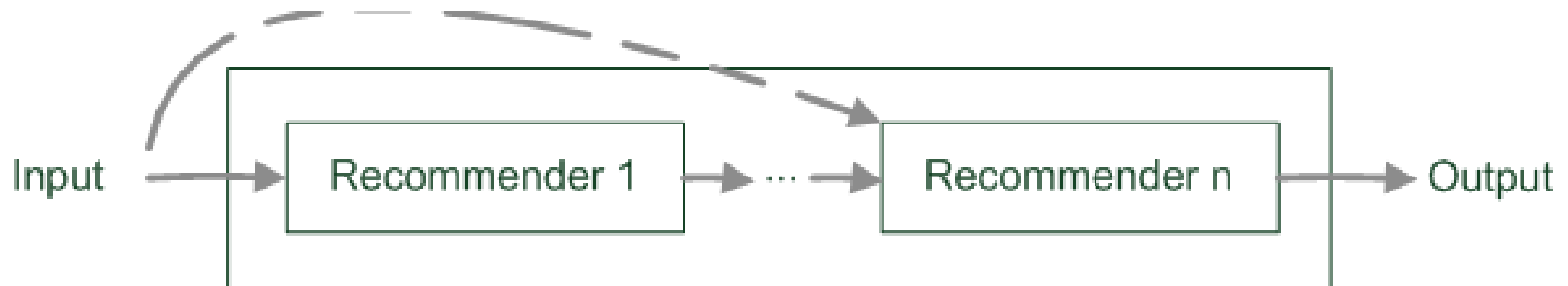
# Pipelined hybridization designs

One recommender system pre-processes some input for the subsequent one

- Cascade
- Meta-level

Refinement of recommendation lists (cascade)

Learning of model (e.g. collaborative knowledge-based meta-level)



# Pipelined hybridization designs: Cascade

Successor's recommendations are restricted by predecessor

$$rec_{cascade}(u, i) = rec_n(u, i)$$

Where for all  $k > 1$

$$rec_k(u, i) = \begin{cases} rec_k(u, i) & : rec_{k-1}(u, i) \neq 0 \\ 0 & : otherwise \end{cases}$$

Subsequent recommender may not introduce additional items

Thus produces very precise results



# Pipelined hybridization designs: Cascade

Recommendation list is continually reduced

First recommender excludes items

- Remove absolute no-go items (e.g. knowledge-based)

Second recommender assigns score

- Ordering and refinement (e.g. collaborative)

# Pipelined hybridization designs: Cascade

Recommender 1		
Item1	0.5	1
Item2	0	
Item3	0.3	2
Item4	0.1	3
Item5	0	

Removing no-go items

Recommender 2		
Item1	0.8	2
Item2	0.9	1
Item3	0.4	3
Item4	0	
Item5	0	

Ordering and refinement

Recommender 3		
Item1	0.80	1
Item2	0.00	
Item3	0.40	2
Item4	0.00	
Item5	0.00	

# Pipelined hybridization designs: Meta-level

Successor exploits a model delta built by predecessor

$$rec_{meta-level}(u, i) = rec_n(u, i, \Delta_{rec_{n-1}})$$

## Examples:

- Fab:
  - Online news domain
  - CB recommender builds user models based on weighted term vectors
  - CF identifies similar peers based on these user models but makes recommendations based on ratings
- Collaborative constraint-based meta-level RS
  - Collaborative filtering learns a constraint base
  - Knowledge-based RS computes recommendations

# Limitations of hybridization strategies

Only few works that compare strategies from the meta-perspective

- Like for instance, [Robin Burke 2002]
- Most datasets do not allow to compare different recommendation paradigms
  - i.e. ratings, requirements, item features, domain knowledge, critiques rarely available in a single dataset
- Thus few conclusions that are supported by empirical findings
  - Monolithic: some preprocessing effort traded-in for more knowledge included
  - Parallel: requires careful matching of scores from different predictors
  - Pipelined: works well for two antithetic approaches

Netflix competition – "stacking" recommender systems

- Weighted design based on >100 predictors – recommendation functions
- Adaptive switching of weights based on user model, context and meta-features

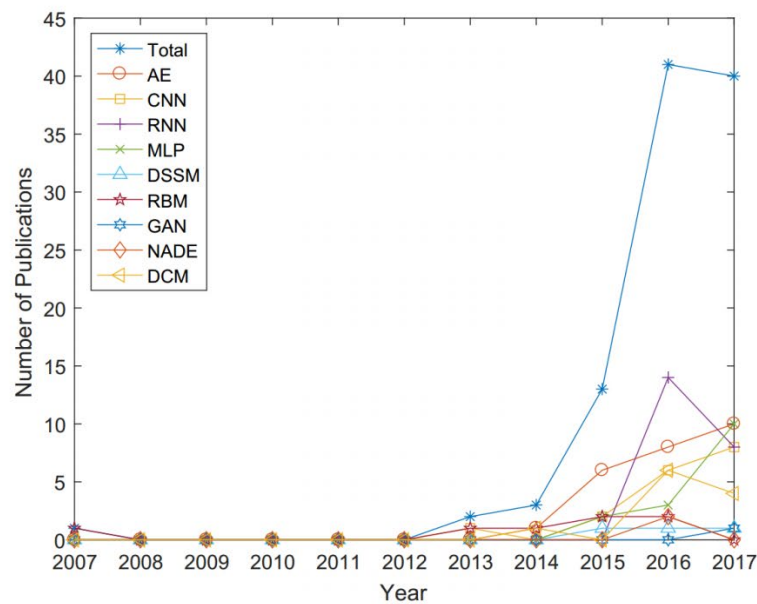
# A example of CF-based recommender system

Introduction to Recommender Systems

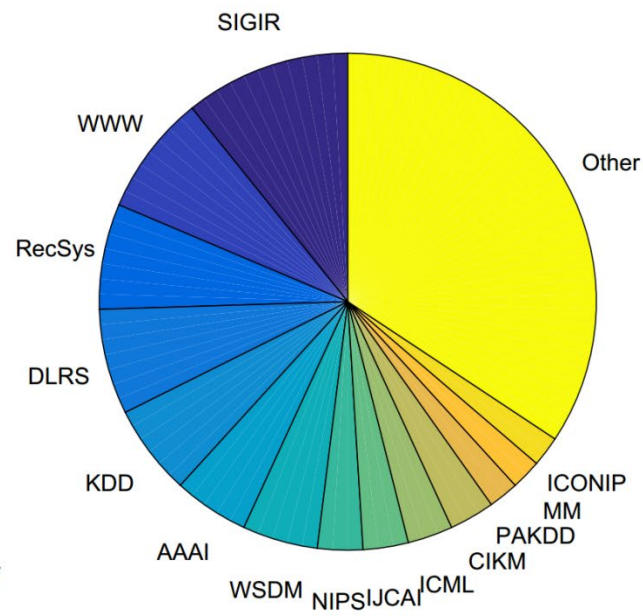
[https://www.cse.unsw.edu.au/~cs9321/19T1/mf\\_rec.slides.html](https://www.cse.unsw.edu.au/~cs9321/19T1/mf_rec.slides.html)

# Deep learning based RS

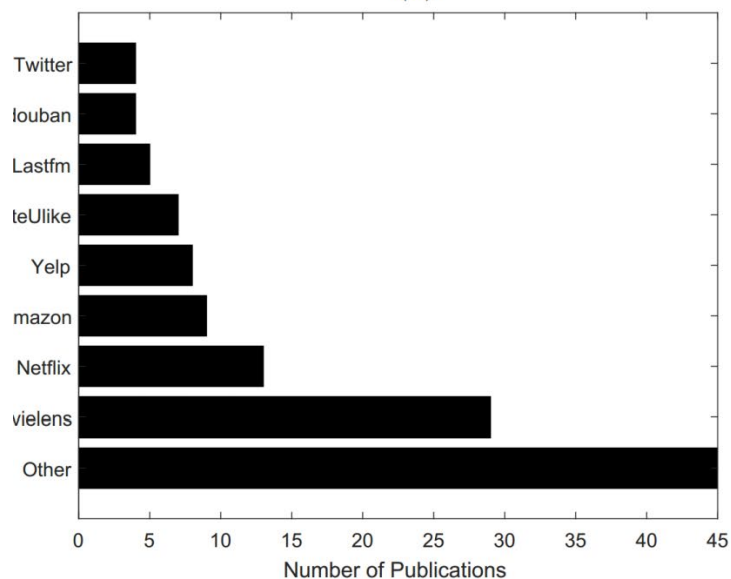
Introduction to Recommender Systems



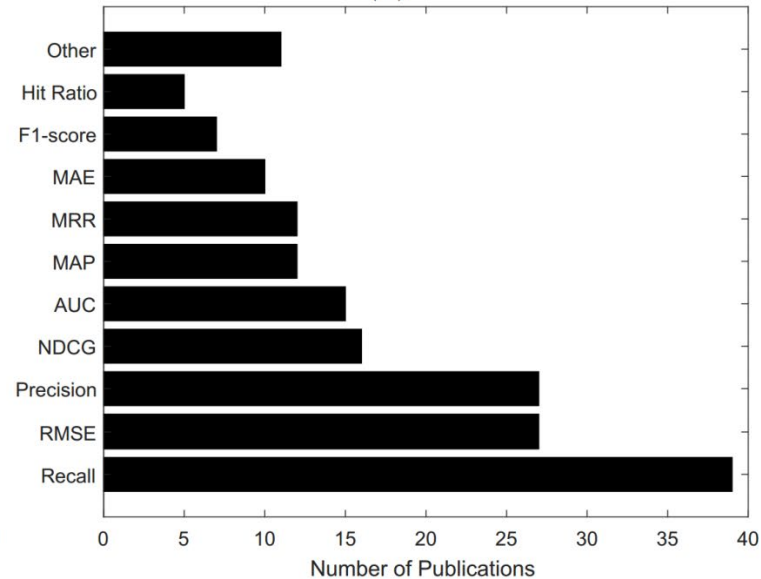
(a)



(b)



(c)



(d)

# Main DL Techniques

Multilayer Perceptron (MLP) is a feedforward neural network with multiple (one or more) hidden layers between input layer and output layer. Here, the perceptron can employ arbitrary activation function and does not necessarily represent strictly binary classifier.

Autoencoder (AE) is an unsupervised model attempting to reconstruct its input data in the output layer. In general, the bottleneck layer (the middle-most layer) is used as a salient feature representation of the input data. There are many variants of autoencoders such as denoising autoencoder, marginalized denoising autoencoder, sparse autoencoder, contractive autoencoder and variational autoencoder.

Convolutional Neural Network (CNN) is a special kind of feedforward neural network with convolution layers and pooling operations. It is capable of capturing the global and local features and significantly enhancing the efficiency and accuracy. It performs well in processing data with grid-like topology.



# Main DL Techniques

Recurrent Neural Network (RNN) is suitable for modelling sequential data. Unlike feedforward neural network, there are loops and memories in RNN to remember former computations. Variants such as Long Short Term Memory (LSTM) and Gated Recurrent Unit (GRU) network are often deployed in practice to overcome the vanishing gradient problem.

Deep Semantic Similarity Model (DSSM), or more specifically, Deep Structured Semantic Model, is a deep neural network for learning semantic representations of entities in a common continuous semantic space and measuring their semantic similarities.

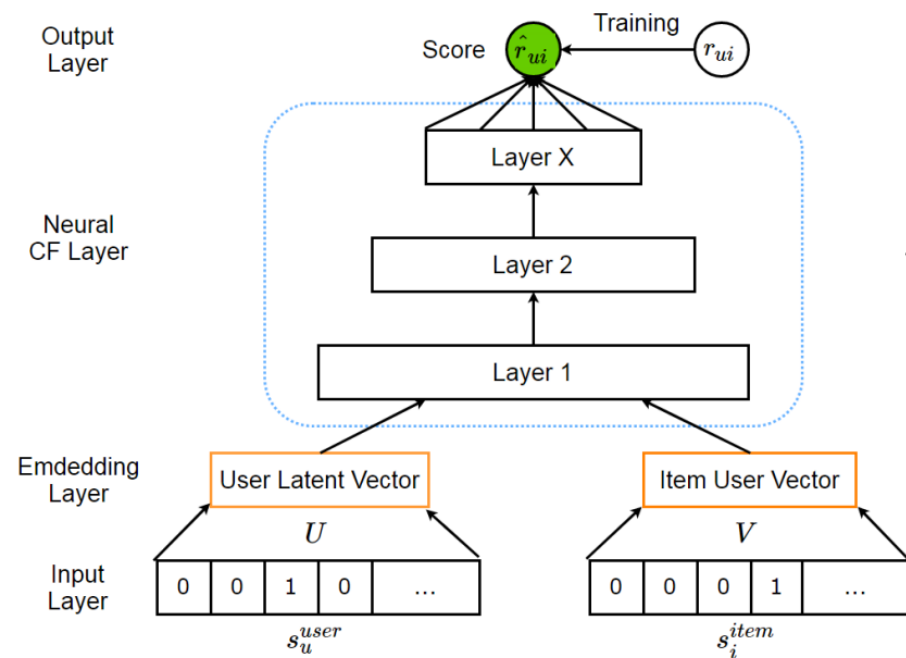
Restricted Boltzmann Machine (RBM) is a two layer neural network consisting of a visible layer and a hidden layer. It can be easily stacked to a deep net. Restricted here means that there are no intra-layer communications in visible layer or hidden layer.

# Main DL Techniques

Neural Autoregressive Distribution Estimation (NADE) is an unsupervised neural network built atop autoregressive model and feedforward neural network. It is a tractable and efficient estimator for modelling data distributions and densities.

Generative Adversarial Network (GAN) is a generative neural network which consists of a discriminator and a generator. The two neural networks are trained simultaneously by competing with each other in a minimax game framework.

# Neural Collaborative Filtering



$s_u^{user}$

denote the side information (e.g. user profiles and item features), or just one-hot identifier of user  $u$  and item  $i$

$s_i^{item}$

$$\hat{r}_{ui} = f(U^T \cdot s_u^{user}, V^T \cdot s_i^{item} | U, V, \theta)$$

where function  $f(\cdot)$  denotes the multilayer perceptron, and  $\theta$  is the parameters of this network

$$\mathcal{L} = \sum_{(u,i) \in \mathcal{O} \cup \mathcal{O}^-} w_{ui} (r_{ui} - \hat{r}_{ui})^2$$

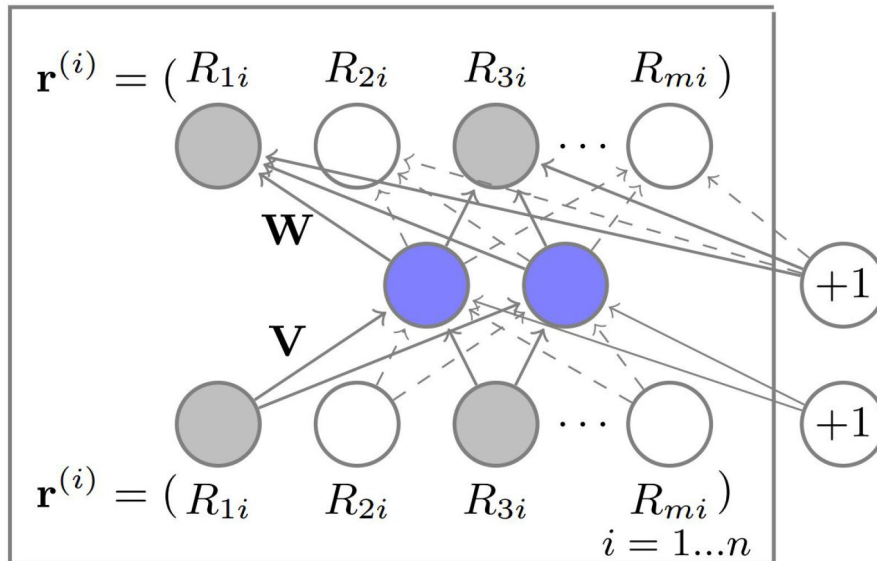
As there are a large number of unobserved instances, NCF utilizes negative sampling (3~6 negative samples) to reduce the training data size, which greatly improves the learning efficiency. Traditional matrix factorization can be viewed as a special case of NCF. Therefore, it is convenient to fuse matrix factorization with NCF to formulate a more general model which makes use of both linearity of MF and non-linearity of MLP to enhance recommendation quality.

# Autoencoder-based RS

Two general ways of applying autoencoder to recommender system:

- (1) Using autoencoder to learn lower-dimensional feature representations at the bottleneck layer;
- (2) Learning the blanks of rating matrix directly in the reconstruction layer.

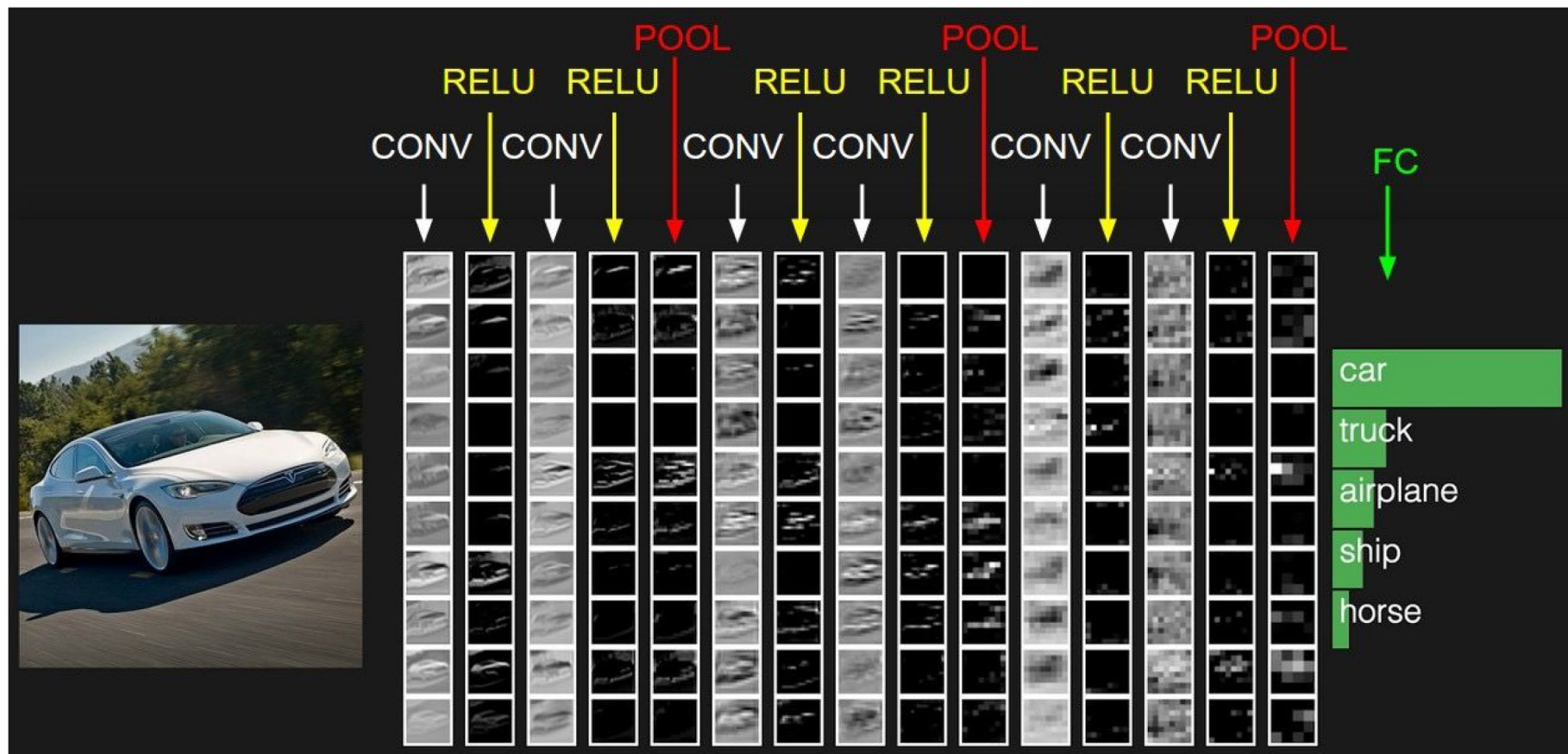
# Autoencoder-based - AutoRec



- I-AutoRec performs better than U-AutoRec, which may be due to the higher variance of user partial observed vectors.
- Different combination of activation functions will influence the performance considerably.
- Increasing the hidden unit size moderately will improve the result as expanding the hidden layer dimensionality gives AutoRec more capacity to model the characteristics of the input.
- Adding more layers to formulate a deep network further improves the performance.

$$\arg \min_{\theta} \sum_{i=1}^N \| \mathbf{r}^{(i)} - h(\mathbf{r}^{(i)}; \theta) \|_{\odot}^2 + \lambda \cdot \text{Regularization}$$

# CNN



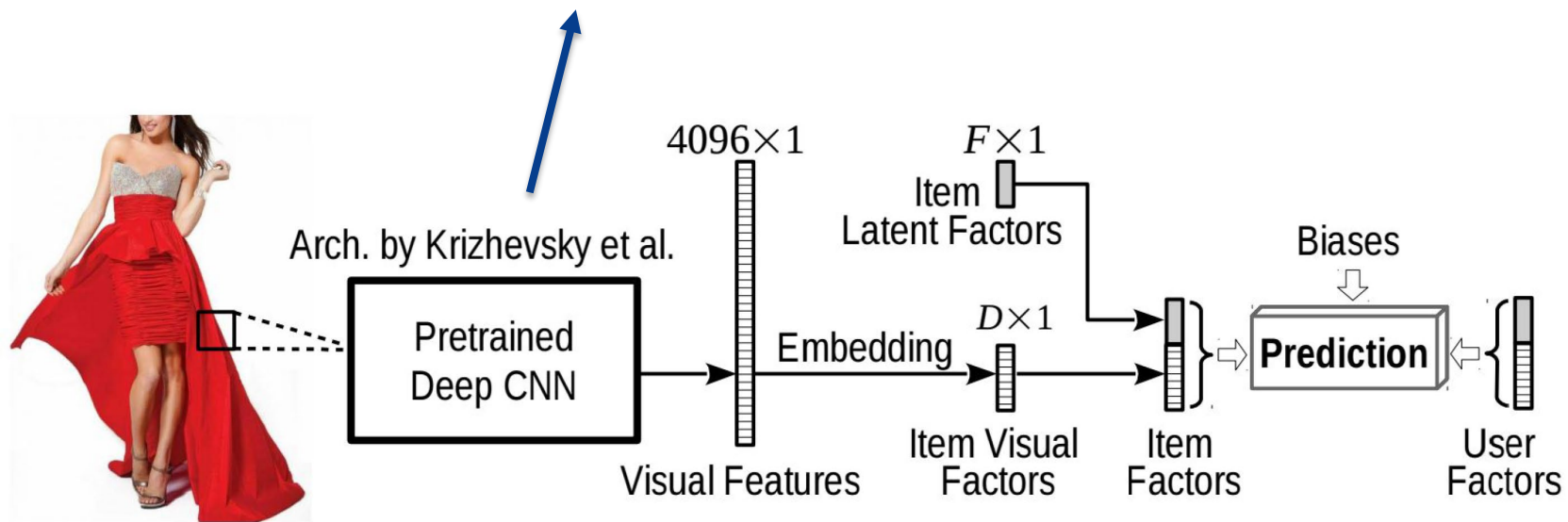
# CNN

There are two ways of CNN based RS

- CNN is used for learning features to enhance CF Image/Audio/Text...
- 2. Integrate CNN with Traditional Recommender System

# Used as feature learning

ImageNet Classification with Deep Convolutional Neural Networks

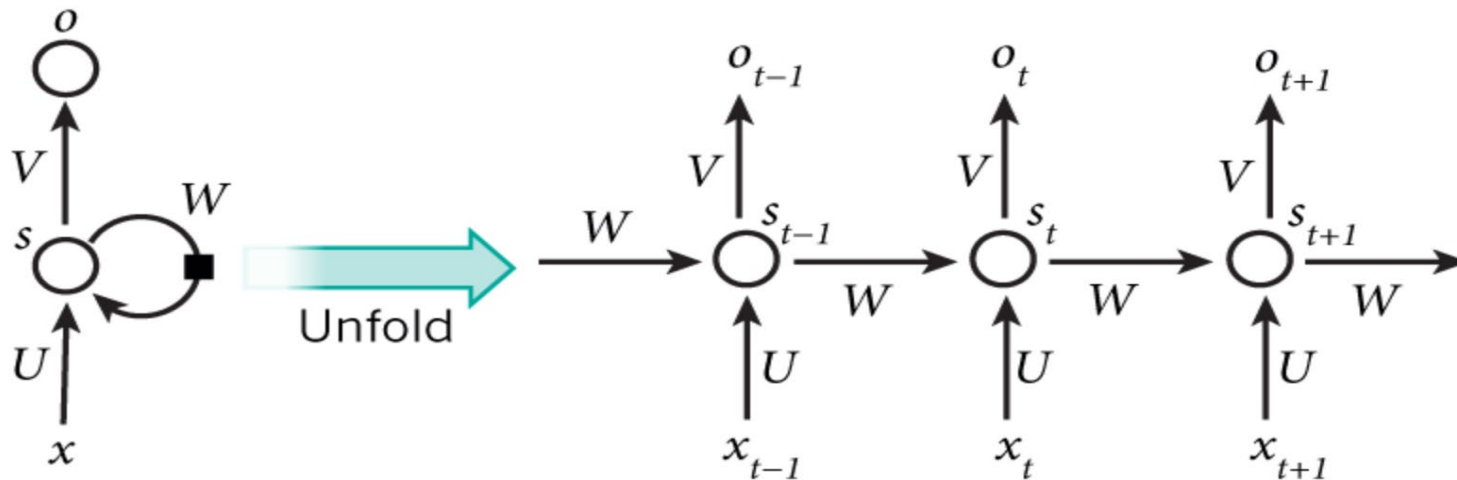


VBPR: Visual Bayesian Personalized Ranking from implicit feedback

Visual Appearance



# RNN

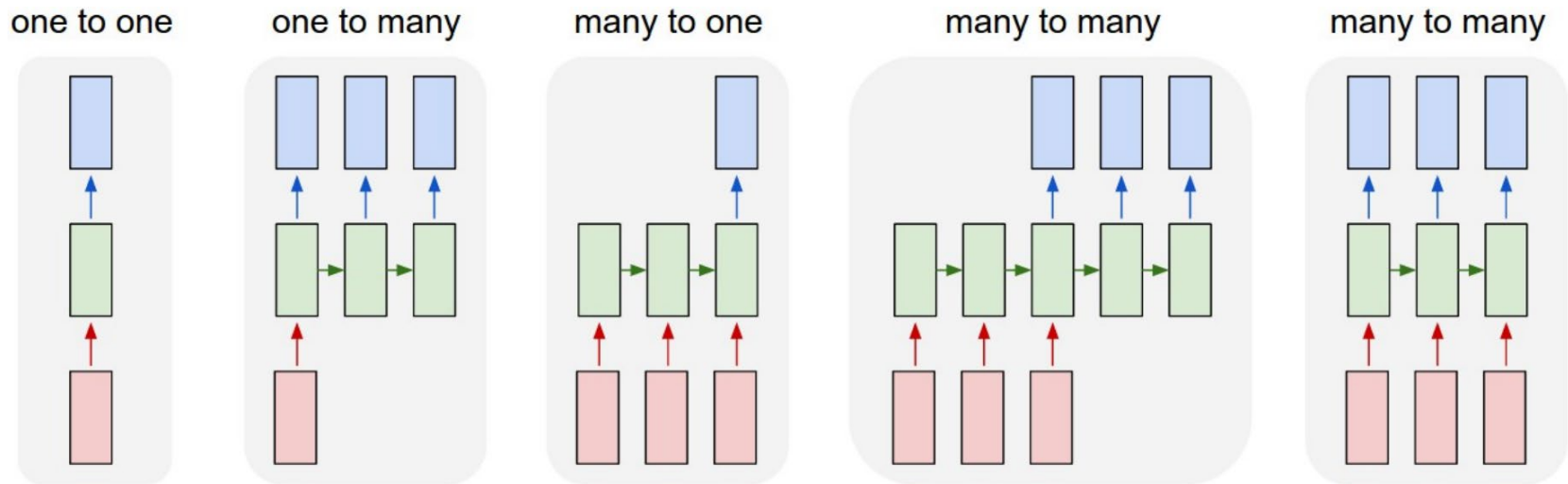


- Perform the same task for every element of a sequence, with the output being depended on the previous computations.
- Have a “memory” which captures information about what has been calculated so far.

Therefore,

- Recurrent neural network is suitable for coping with the temporal dynamics of ratings and sequential features in recommender system

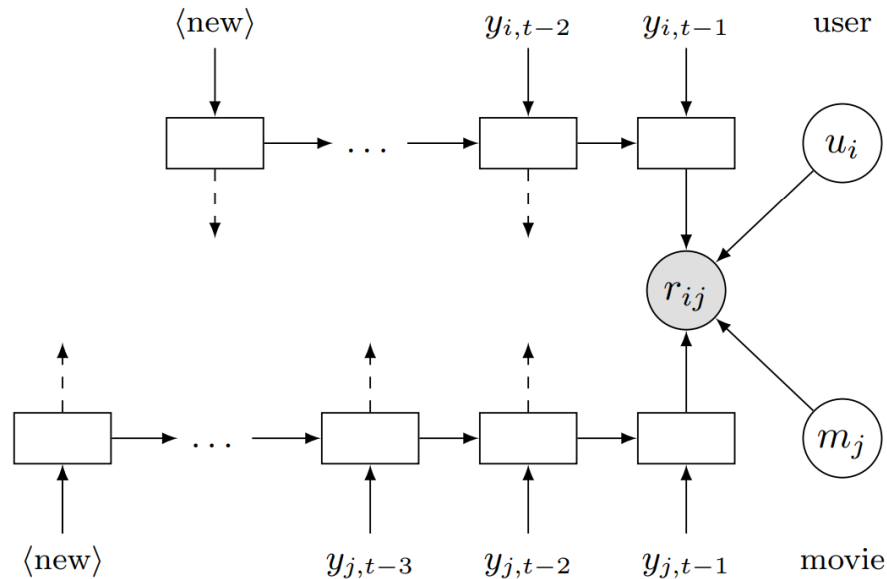
# RNN



Each rectangle is a vector and arrows represent functions (e.g. matrix multiply). Input vectors are in red, output vectors are in blue and green vectors hold the RNN's state

**“allow us to operate over sequences of vectors: Sequences in the input, the output, or in the most general case both”**

# Recurrent Recommender Network (RRN)



A non-parametric recommendation model built on RNN. It is capable of modelling the seasonal evolutions of items and changes of user preferences over time.

Two LSTM networks as the building block to model dynamic user and item states

Stationary properties such as user long-term interests and item features

# Recap: DL-based RS

DL (DNN, AE, CNN, RNN) boosts recommendation performance

CNNs ideal for content-based feature learning: ameliorate cold-start problem

RNNs very powerful for sequence based recommendation, multi-modal

learning, and order-aware distributed representations, trends forecast ...

# RS Measures

Introduction to Recommender Systems

# What are the measures in practice?

## Different perspectives/aspects

- – Depends on domain and purpose
- – No holistic evaluation scenario exists

## Retrieval perspective

- – Reduce search costs
- – Provide "correct" proposals
- – Assumption: Users know in advance what they want

## Recommendation perspective

- – Serendipity – identify items from the Long Tail
- – Users did not know about existence

# Purpose and success criteria

## Prediction perspective

- – Predict to what degree users like an item
- – Most popular evaluation scenario in research

## Interaction perspective

- – Give users a "good feeling"
- – Educate users about the product domain
- – Convince/persuade users - explain

## Finally, conversion perspective

- – Commercial situations
- – Increase "hit", "clickthrough", "lookers to bookers" rates
- – Optimize sales margins and profit

# How do we as researchers know?

## Test with real users

- – A/B tests
- – Example measures: sales increase, click through rates

## Laboratory studies

- – Controlled experiments
- – Example measures: satisfaction with the system (questionnaires)

## Offline experiments

- – Based on historical data
- – Example measures: prediction accuracy, coverage



# Accuracy Related

Precision: a measure of exactness, determines the fraction of relevant items retrieved out of all items retrieved

- – E.g. the proportion of recommended movies that are actually good

Recall: a measure of completeness, determines the fraction of relevant items retrieved out of all relevant items

- – E.g. the proportion of all good movies recommended

$$Precision = \frac{tp}{tp + fp} = \frac{|good\ movies\ recommended|}{|all\ recommendations|}$$

$$Recall = \frac{tp}{tp + fn} = \frac{|good\ movies\ recommended|}{|all\ good\ movies|}$$

# Accuracy Related

## Datasets with items rated by users

- – MovieLens datasets 100K-10M ratings
- – Netflix 100M ratings

## Historic user ratings constitute ground truth

## Metrics measure error rate

- – Mean Absolute Error (MAE) computes the deviation between predicted ratings and actual ratings
- – Root Mean Square Error (RMSE) is similar to MAE, but places more emphasis on larger deviation

$$MAE = \frac{1}{n} \sum_{i=1}^n |p_i - r_i|$$

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (p_i - r_i)^2}$$

# Not enough....

IR measures are frequently applied, however:

Ground truth for most items actually unknown

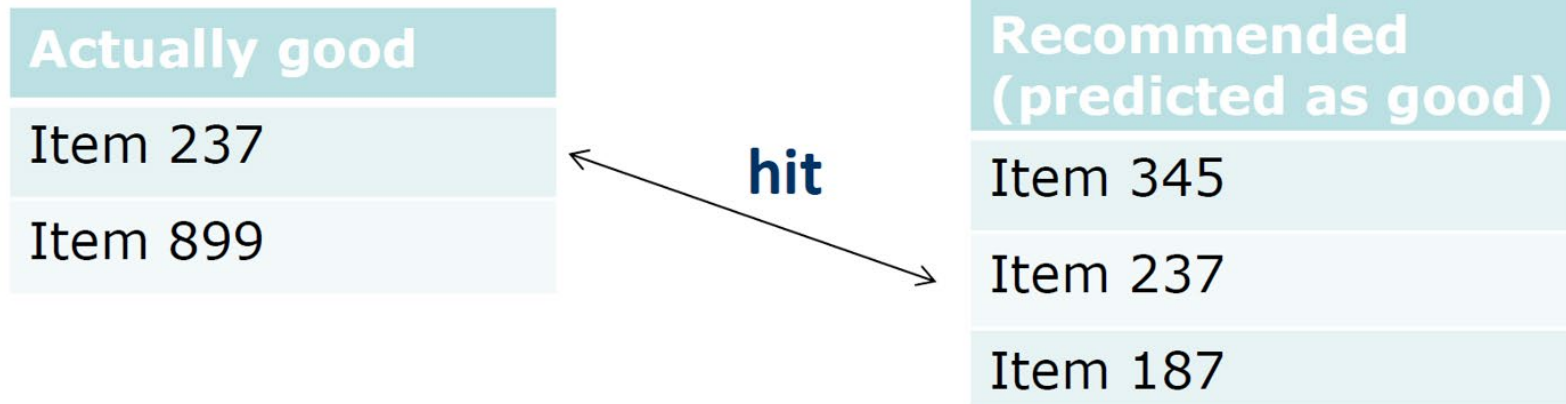
What is a relevant item?

Different ways of measuring precision possible

Results from offline experimentation may have limited predictive power for

Online user behaviour.

# Metrics: Rank Score – position matters



- Rank Score extends recall and precision to take the positions of correct items in a ranked list into account
- Particularly important in recommender systems as lower ranked items may be overlooked by users
- Learning-to-rank: optimize models for such measures (e.g., AUC)

# nDCG

Concept of graded relevance

Hits at the beginning count more (more "gain")

Documents of higher relevance are more important

Discounted gain at later positions

- Often an exponential decay (half life) is assumed e.g., based on the log function

Given a rank position  $p$ , and the graded relevance "rel" of an item  $I$

- nDCG: Normalized value at length  $n$  Compare with "ideal" ranking

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