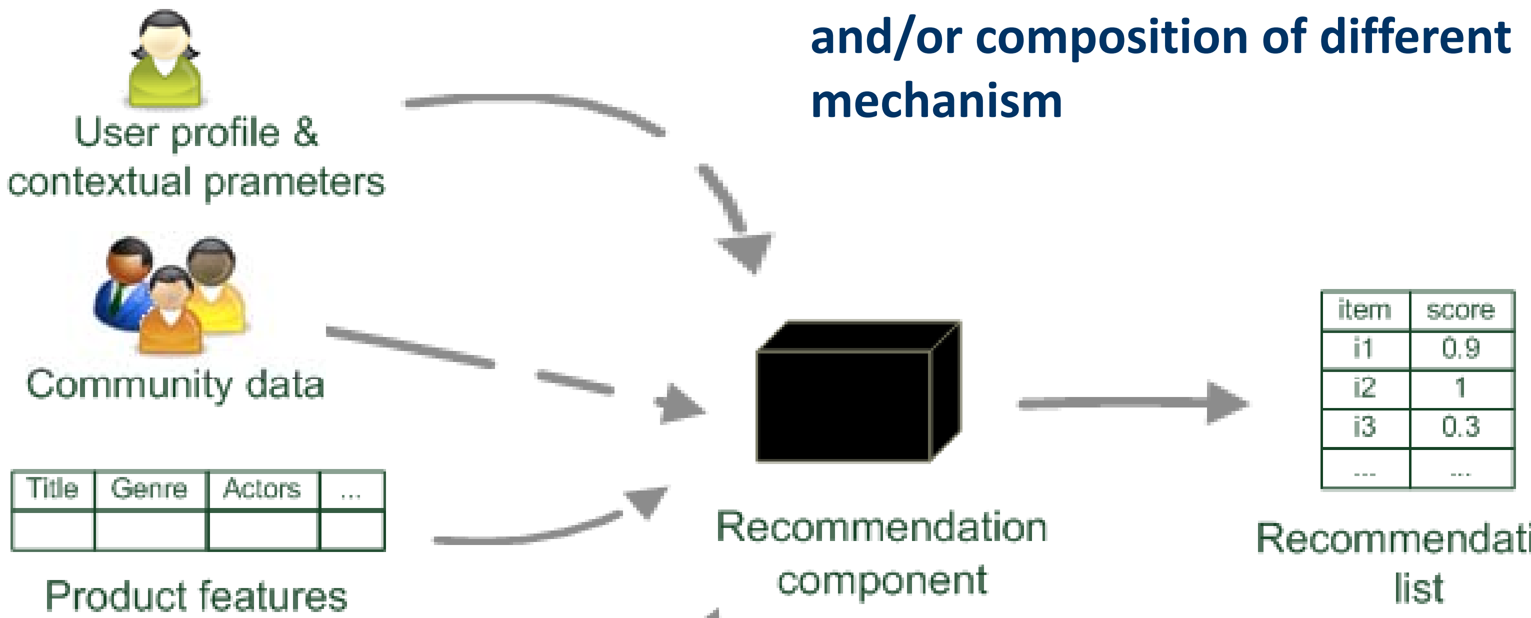
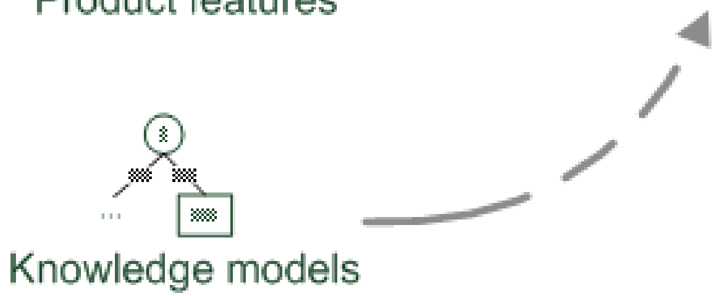
Hybrid recommendation approaches

Hybrid recommender systems

Hybrid: combinations of various inputs and/or composition of different mechanism





Collaborative: "Tell me what's popular among my peers"

Content-based: "Show me more of the same what I've liked"

Knowledge-based: "Tell me what fits based on my needs"

Hybrid recommender systems

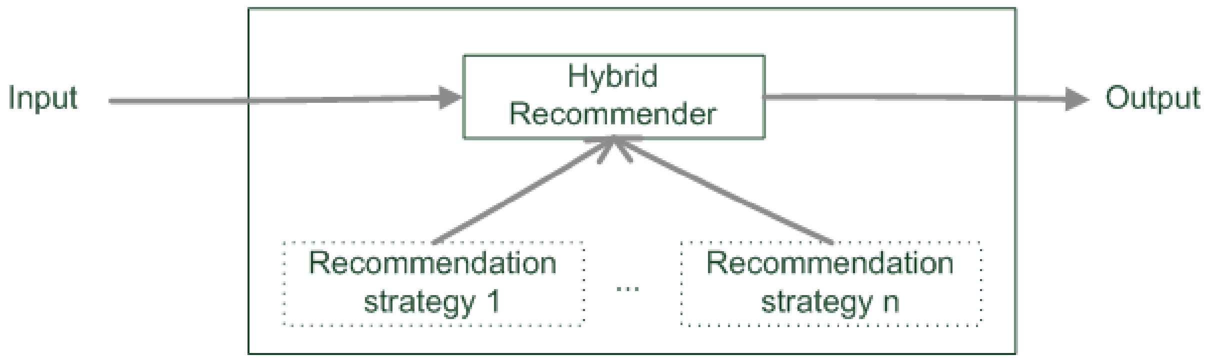
- **All three base techniques are naturally incorporated by a good sales assistant (at different stages of the sales act) but have their shortcomings**
  - For instance, cold start problems

- **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

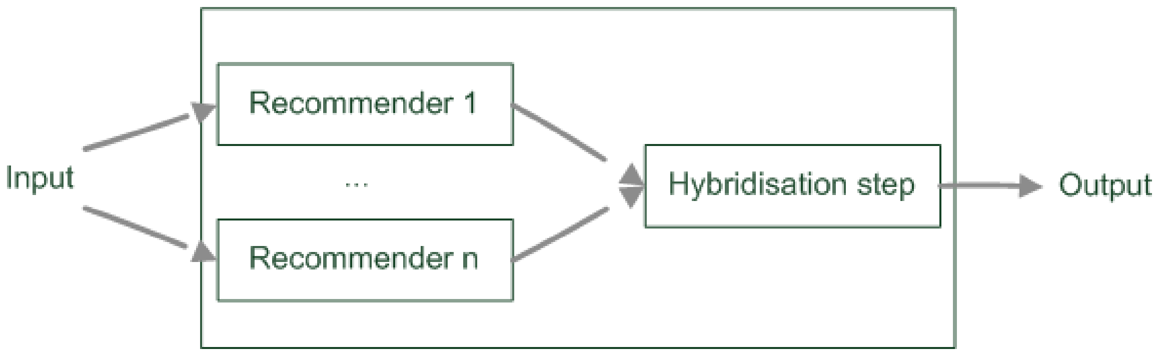
- **"Hybrid" content features:**
  - Social features: Movies liked by user
  - Content features: Comedies liked by user, dramas liked by user
  - Hybrid features: user likes many movies that are comedies, …
  - *"the common knowledge engineering effort that involves inventing good features to enable 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

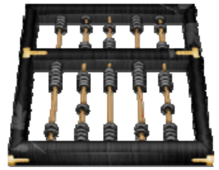
- **BUT, how to derive weights?**
  - Estimate, e.g. by empirical bootstrapping
  - Dynamic adjustment of weights

- **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

Parallelized hybridization design: Weighted

- **Let's assume Alice actually bought/clicked on items 1 and 4**
  - Identify weighting that minimizes Mean Absolute Error (MAE)



Absolute errors and MAE						
Beta1	Beta2		rec1	rec2	error	MAE
0.1	0.9	Item1	0.5	0.8	0.23	0.61
		Item4	0.1	0.0	0.99	
0.3	0.7	Item1	0.5	0.8	0.29	0.63
		Item4	0.1	0.0	0.97	
0.5	0.5	Item1	0.5	0.8	0.35	0.65
		Item4	0.1	0.0	0.95	
0.7	0.3	Item1	0.5	0.8	0.41	0.67
		Item4	0.1	0.0	0.93	
0.9	0.1	Item1	0.5	0.8	0.47	0.69
		Item4	0.1	0.0	0.91	

- **MAE improves as *rec*2 is weighted more strongly**

$$MAE = \frac{\sum_{r_i \in R} \sum_{k=1}^{n} \beta_k \times |rec_k(u,i) - r_i|}{|R|}$$

Parallelized hybridization design: Weighted

- **BUT: didn't rec1 actually rank Items 1 and 4 higher?**

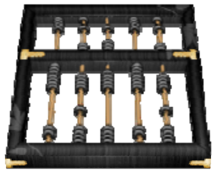
Recommender 1		
Item1	0.5	1
Item2	0	
Item3	0.3	2
Item4	0.1	3
Item5	0	

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

- **Be careful when weighting!**
  - Recommenders need to assign comparable scores over all users and items
    - Some score transformation could be necessary
  - Stable weights require several user ratings

Parallelized hybridization design: Switching

- **Requires an oracle that decides on recommender**



$$\exists_1 k : 1 \ldots n \; rec_{switching}(u,i) = rec_k(u,i)$$

- **Special case of dynamic weights (all except one Beta is 0)**

- **Example:**
  - Ordering on recommenders and switch based on some quality criteria
    - E.g. if too few ratings in the system use knowledge-based, else collaborative
  - More complex conditions based on contextual parameters, apply classification techniques

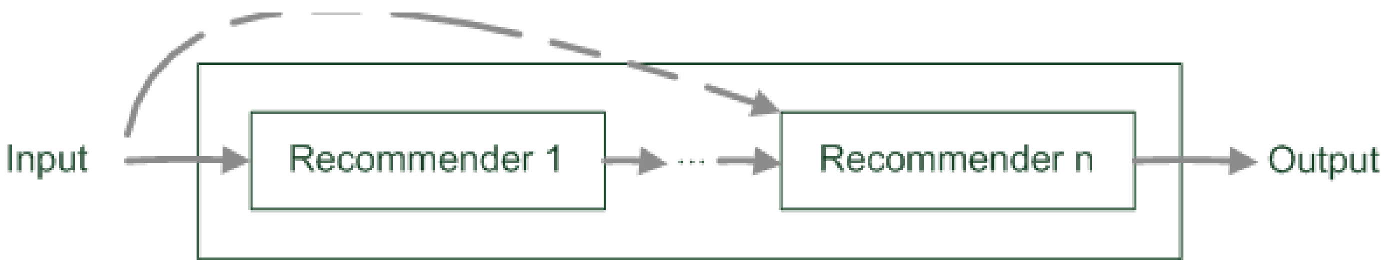
Parallelized hybridization design: Mixed

- **Combines the results of different recommender systems at the level of user interface**
- **Results of different techniques are presented together**
- **Recommendation result for user u and item i is the set of tuples $< score, k >$ for each of its n constituting recommenders rec_k**

$$rec_{mixed} = \bigcup_{k=1}^{n} \langle rec_k(u,i), k \rangle$$

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)**



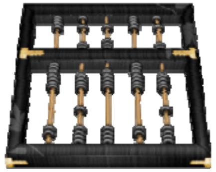
Pipelined hybridization designs: Cascade

- **Successor's recommendations are restricted by predecessor**

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

- **Where forall 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	

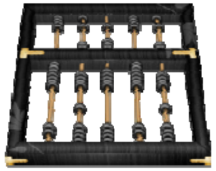
Ordering and refinement

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

Literature

- **[Robin Burke 2002]** Hybrid recommender systems: Survey and experiments, User Modeling and User-Adapted Interaction **12** (2002), no. 4, 331-370.
- **[Prem Melville et al. 2002]** *Content-Boosted Collaborative Filtering for Improved Recommendations*, Proceedings of the 18th National Conference on Artificial Intelligence (AAAI) (Edmonton,CAN), American Association for Artificial Intelligence, 2002, pp. 187-192.
- **[Roberto Torres et al. 2004]** *Enhancing digital libraries with techlens*, International Joint Conference on Digital Libraries (JCDL'04) (Tucson, AZ), 2004, pp. 228-236.
- **[Chumki Basuet al. 1998]** *Recommendation as classification: using social and content-based information in recommendation*, In Proceedings of the 15th National Conference on Artificial Intelligence (AAAI'98) (Madison, Wisconsin, USA States), American Association for Artificial Intelligence, 1998, pp. 714-720.