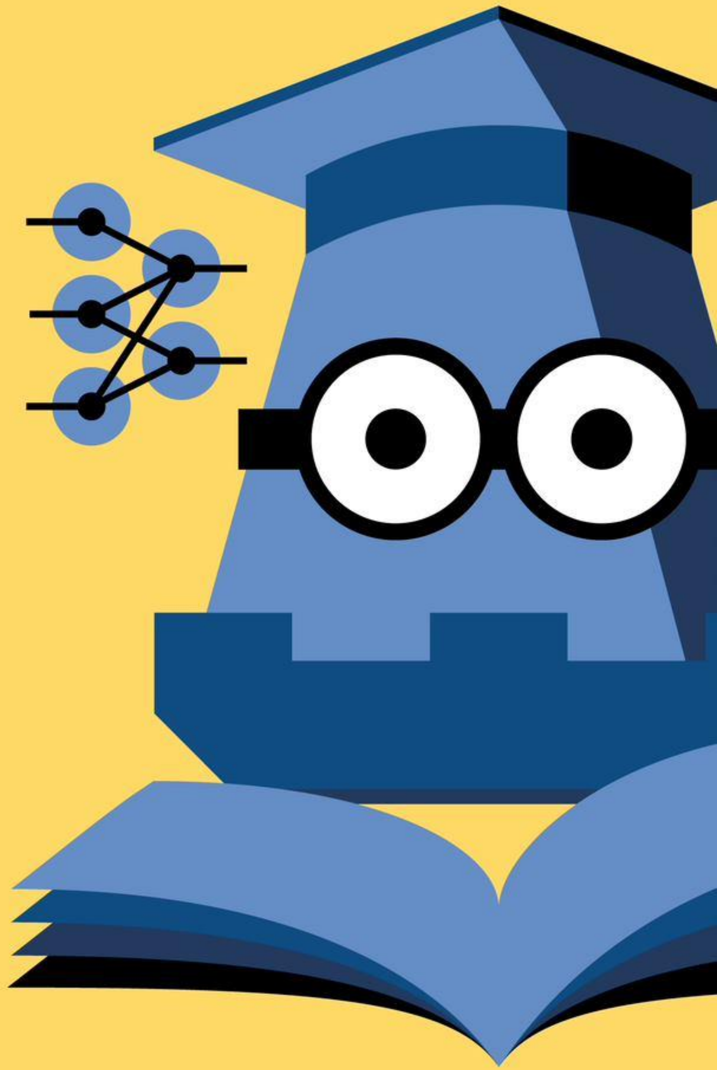
31.7.-10.8.

PETNICA SUMMER INSTITUTE

MACHINE LEARNING 6

PETNICA SUMMER INSTITUTE
MACHINE LEARNING 6



Recommender systems

– an introduction –

Miloš Jovanović

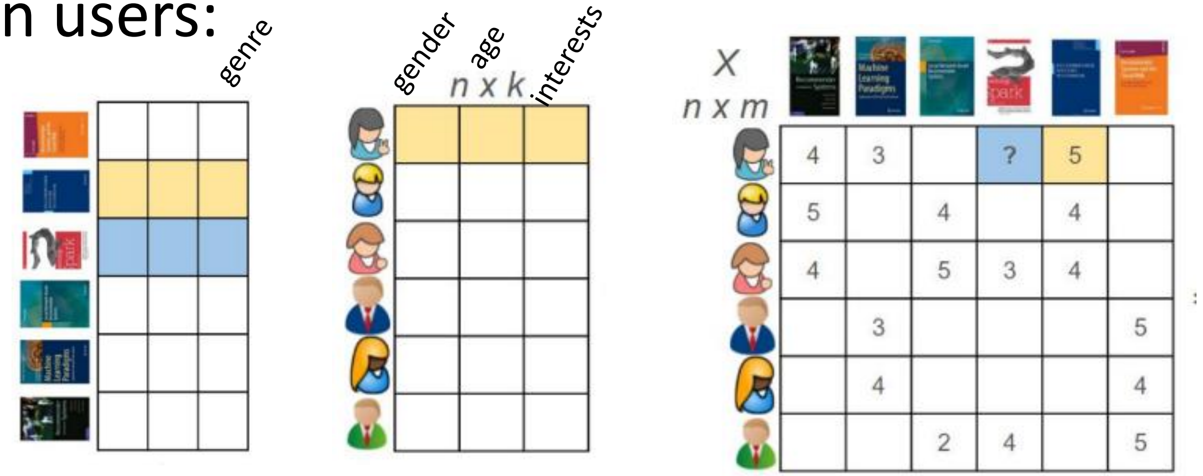
milos.jovanovic@fon.bg.ac.rs

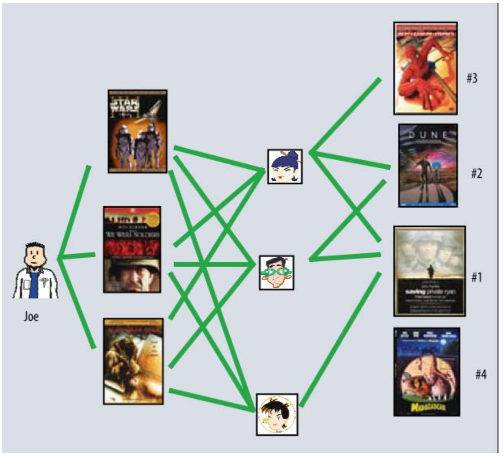
ФОН

УНИВЕРЗИТЕТ У БЕОГРАДУ
ФАКУЛТЕТ ОРГАНИЗАЦИОНИХ НАУКА

Recommendation problem

- Recommend items to users that **we know** have **interest** in them.
- How do we know what people might like, when we recommend:
  - who they are
  - what they previously liked or showed some interest for (bipartite graph)
- Data on users:





- $P(x_4, x_5, x_6 | x_1, x_2, x_3)$ Discriminative or Generative?
- Content-based recommenders VS Collaborative filtering VS Hybrid
- Multi-label classification, Learning to Rank

https://www.altoros.com/blog/building-recommenders-with-multilayer-perceptron-using-tensorflow/

Baseline methods

- Baseline rule
  - $p_{u,i} = \mu + b_u + b_i$
  - Predict high scores on popular movies for people who like movies
- Association rules

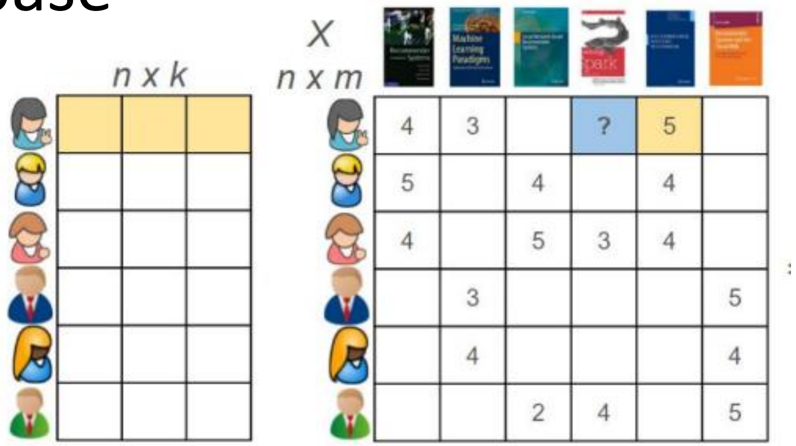
Rule No.	Frequent itemset
1	Apple ⇒ Cereal
2	Beer ⇒ Eggs
3	Eggs ⇒ Beer
4	Beer, Cereal ⇒ Eggs
5	Cereal, Eggs ⇒ Beer

- SlopeOne

$$\hat{r}_{u,i} = \frac{1}{card(R_u)} \sum_{j \in R_u} (dev_{i,j} + r_{u,j}) \qquad dev_{i,j} = \frac{1}{card(s(i,j))} \sum_{v \in s(i,j)} (r_{v,i} - r_{v,j})$$

Neighbourhood methods

- "Similar people like similar things"
- Content-based or Collaborative similarity
  - tags, movie genres, director name, visited URLs, skills on a resume, description
- Database



- Query user

Neighbourhood methods

- Similarity measures:

$$S_{u,v} = \frac{\sum\limits_{i \in U(u,v)} r_{u,i} \cdot r_{v,i}}{\sqrt{\sum\limits_{i \in U(u,v)} r_{u,i}^2} \cdot \sqrt{\sum\limits_{i \in U(u,v)} r_{v,i}^2}}$$

$$S_{u,v} = \frac{\sum\limits_{i \in U(u,v)} (r_{u,i} - \overline{r}_u) \cdot (r_{v,i} - \overline{r}_v)}{\sqrt{\sum\limits_{i \in U(u,v)} (r_{u,i} - \overline{r}_u)^2} \cdot \sqrt{\sum\limits_{i \in U(u,v)} (r_{v,i} - \overline{r}_v)^2}}$$

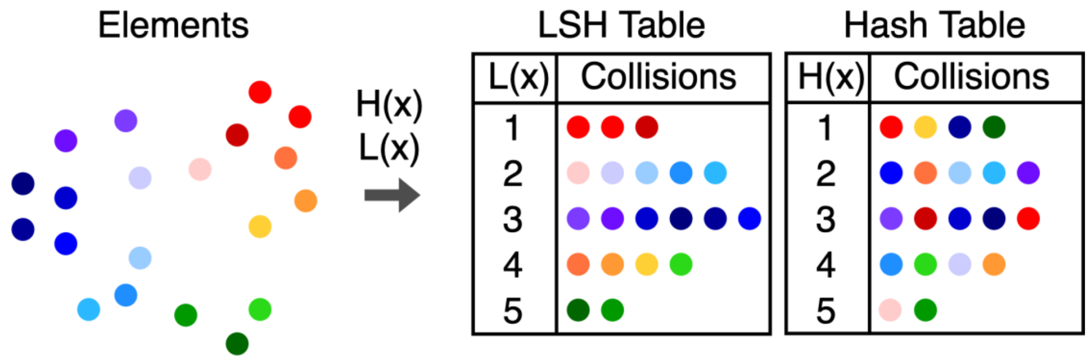
- Predict rating:

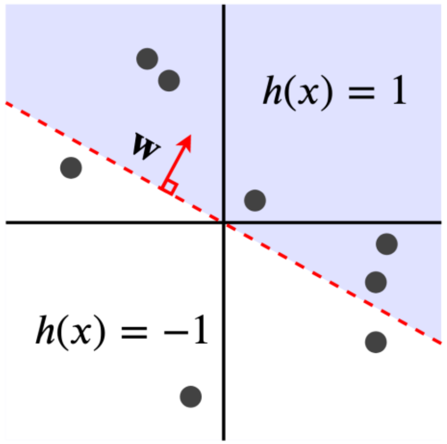
$$\hat{r}_{u,i} = \overline{r}_u + \frac{\sum\limits_{v \in N(u,i)} S_{u,v}(r_{v,i} - \overline{r}_v)}{\sum\limits_{v \in N(u,i)} S_{u,v}}$$

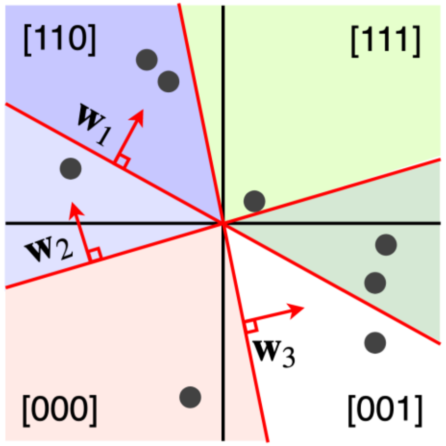
- Lazy-learning, Memory-based
- Evaluation $RMSE = \sqrt{\frac{1}{|T|} \sum\limits_{(u,i) \in T} (r_{u,i} - \hat{r}_{u,i})^2}$ $MAE = \frac{1}{|T|} \sum\limits_{(u,i) \in T} |r_{u,i} - \hat{r}_{u,i}|$

Approximate kNN - LSH

- kNN - Poor time complexity at inference
- Search => Indices and Hashes
    - but not for retrieval by exact value
- Locality-Sensitive Hashing
    - Collision probability – based on how similar objects are: $p(\mathbf{x},\mathbf{y}) = 1 - \frac{\theta(\mathbf{x},\mathbf{y})}{\pi}$



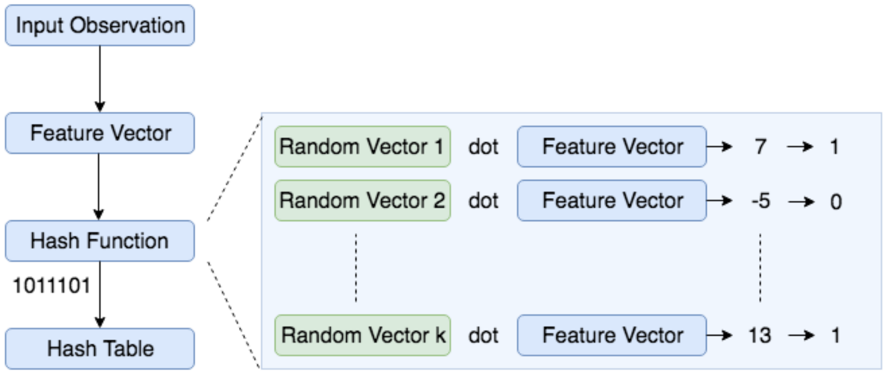




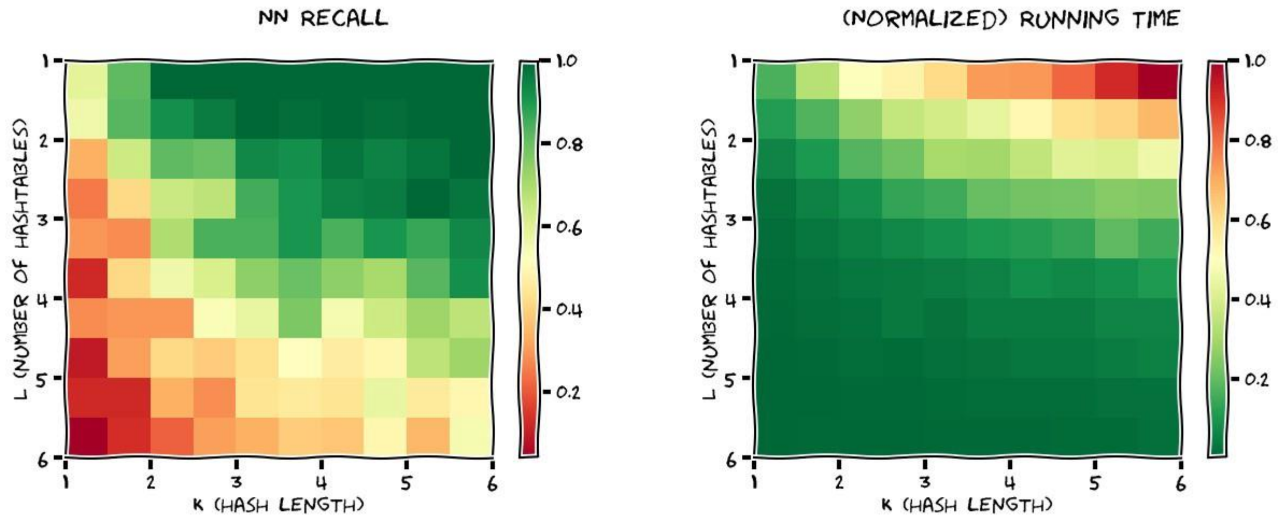
For cosine distance

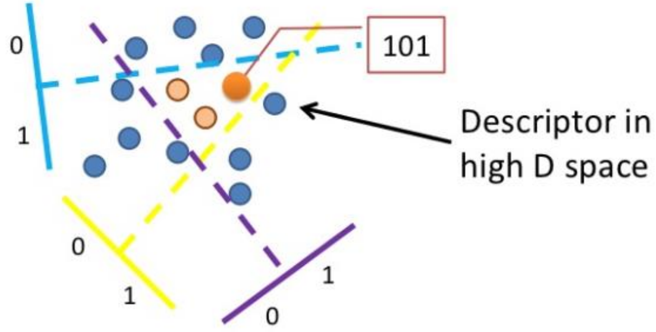
https://brc7.github.io/2019/09/19/Visual-LSH.html

Locality-Sensitive Hashing (LSH)



- Low dimension and binary code (embedding?)
- Hyper parameters:
  - Hash length and Number of hash tables



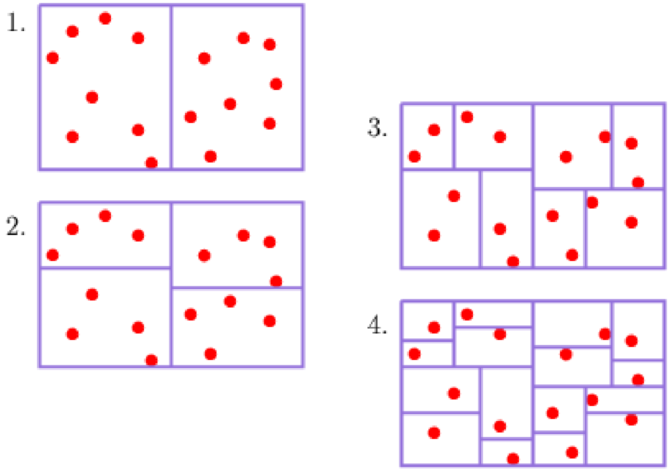


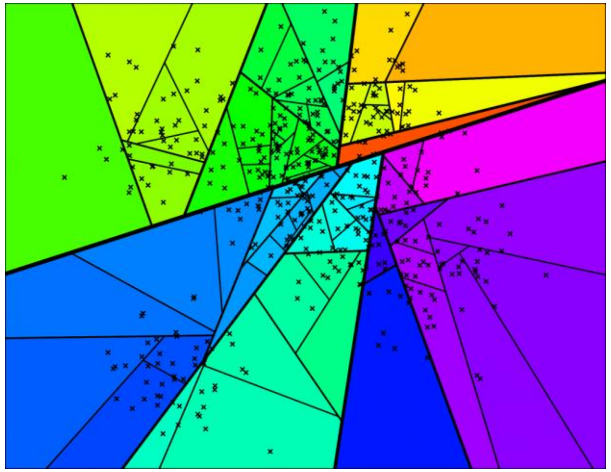
- Great for high dimensional problems and NLP (and DeepNN?)

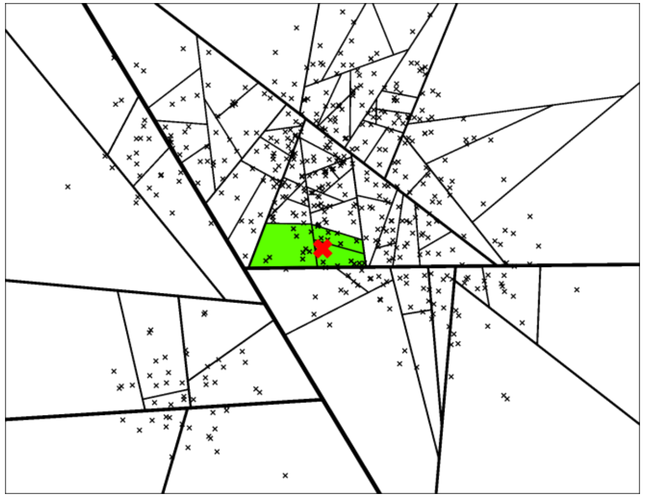
https://towardsdatascience.com/fast-near-duplicate-image-search-using-locality-sensitive-hashing-d4c16058efcb

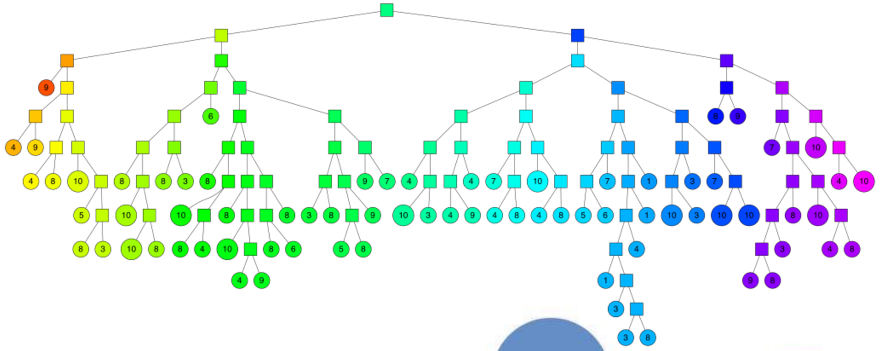
Tree-index for similarity search

- KD-tree
  - splits on median
  - random dimension
  - re-sort the data (as in quick-sort)
- Annoy (Spotify)
  - Forest of decision trees
  - Each split creates a hyperplane to separate two random points
  - better recall









https://www.analyticsvidhya.com/blog/2017/11/information-retrieval-using-kdtree/
https://erikbern.com/2015/10/01/nearest-neighbors-and-vector-models-part-2-how-to-search-in-high-dimensional-spaces.html

Item2item approach

- Amazon 2003:
  - 29 million customers
  - several million items
  - 30% of page views from recommends
  - patented in 2001.
  - reported use from Youtube in 2010
  - Test of time in 2017
- Cosine similarity between items
- Scallable (offline computation)
- Good with limited user data
  - “cold start”
- Explainable

Figure 2. Amazon.com shopping cart recommendations. The recommendations are based on the items in the customer’s cart: The Pragmatic Programmer *and* Physics for Game Developers.

Netflix prize competition

- Year 2006., data: 100M ratings (1-5 stars) on 17K movies from 500K customers
- $1M for the decrease by 10% from Netflix recommendation engine (RMSE of 0.9514)
- 48K teams, 182 countries
- No team won, yearly progress prize $50K
- The rise of Matrix Factorization for RecSys

Matrix factorization (low rank)

- We recommend based on features, not exact items
  - we need to **match** item features to user preferences toward **those** features
  - similarity of user to item (both need to be in same space)

Item

User	W	X	Y	Z
A		4.5	2.0	
B	4.0		3.5	
C		5.0		2.0
D		3.5	4.0	1.0

Rating Matrix

=

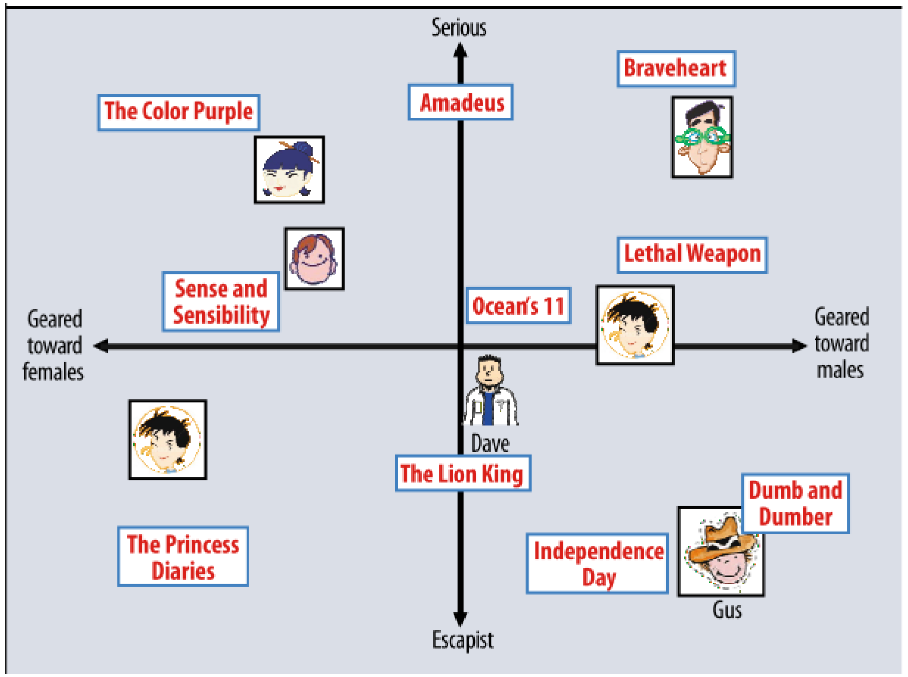
A	*1.2*	*0.8*
B	*1.4*	*0.9*
C	*1.5*	*1.0*
D	*1.2*	*0.8*

User Matrix

X

W	X	Y	Z
1.5	*1.2*	*1.0*	*0.8*
1.7	*0.6*	*1.1*	*0.4*

Item Matrix



- similarity (match) = dot product

Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. Computer 42, 8 (2009), 30–37.

Matrix factorization - Learning

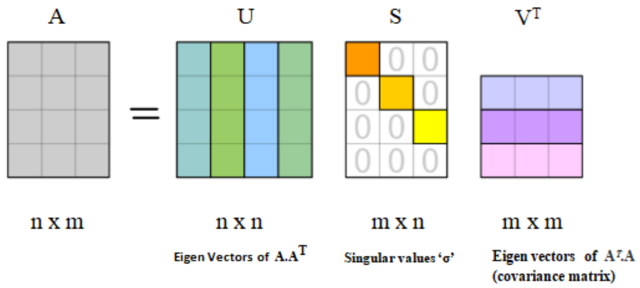
- Loss:

$$\min_{q^*,p^*} \sum_{(u,i)\in\kappa} (r_{ui} - q_i^T p_u)^2 + \lambda(\| q_i \|^2 + \| p_u \|^2)$$

- SGD:

$$q_i \leftarrow q_i + \gamma \cdot (e_{ui} \cdot p_u - \lambda \cdot q_i)$$
$$p_u \leftarrow p_u + \gamma \cdot (e_{ui} \cdot q_i - \lambda \cdot p_u)$$

- Alternating Least Squares
- Alternatives: SVD, SVD++



Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. Computer 42, 8 (2009), 30–37.

Matrix factorization - Learning

- Add bias $\hat{r}_{ui} = \mu + b_i + b_u + q_i^T p_u$

$$\min_{p*,q*,b*} \sum_{(u,i)\in\kappa} (r_{ui} - \mu - b_u - b_i - p_u^T q_i)^2 + \lambda$$
$$(\| p_u \|^2 + \| q_i \|^2 + b_u^2 + b_i^2)$$

- Non-negative matrix factorization (NMF)
  - oprimize, subject to: $p_u > 0, q_i > 0, \forall i, u$
  - interpretable factors

Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. Computer 42, 8 (2009), 30–37.

Matrix factorization

- Item features and User attributes

$$\hat{r}_{ui} = \mu + b_i + b_u + q_i^T [p_u + |N(u)|^{-0.5} \sum_{i \in N(u)} x_i + \sum_{a \in A(u)} y_a]$$

- Temporal dynamics (Y. Koren, “Collaborative Filtering with Temporal Dynamics,” 2009)

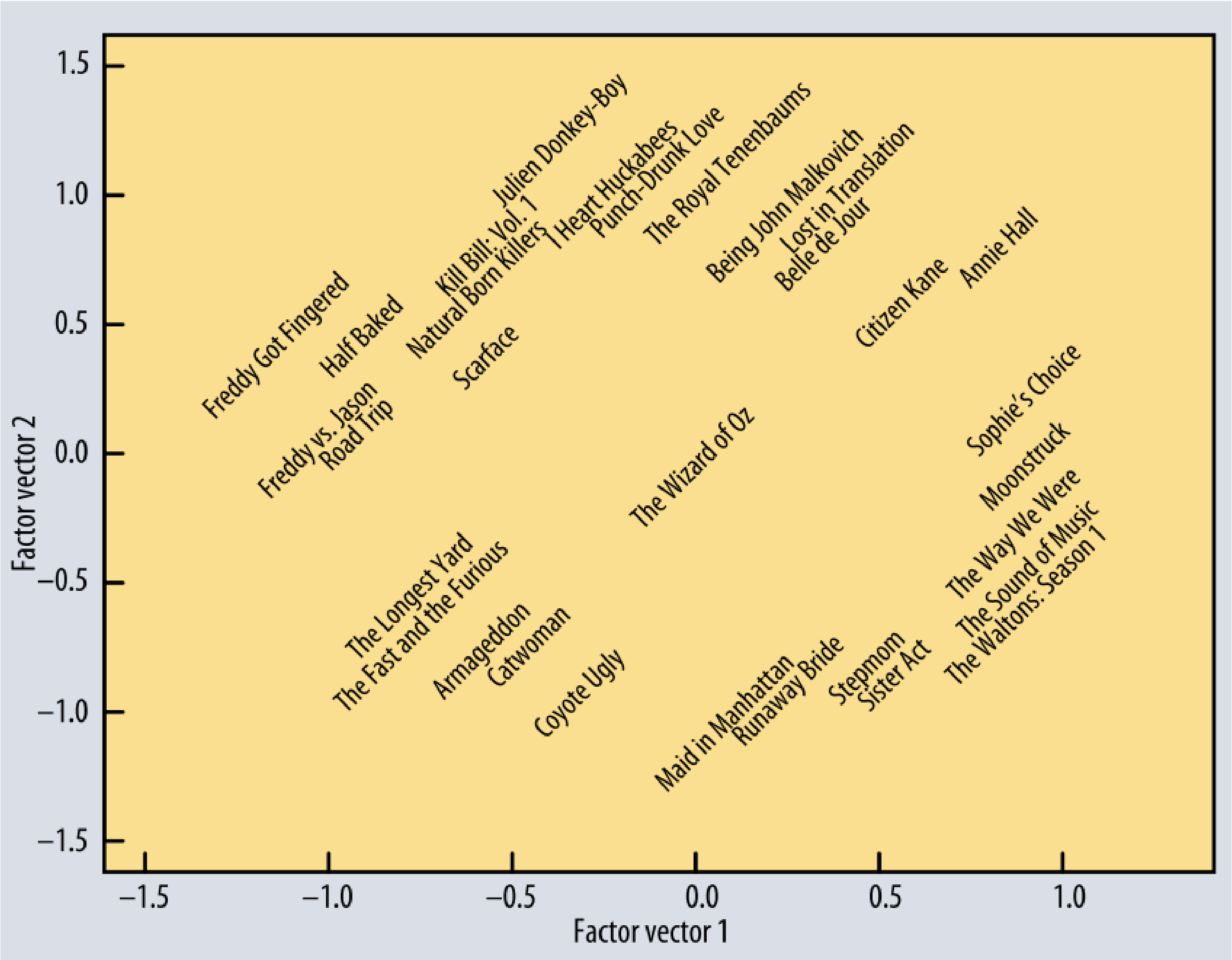
$$\hat{r}_{ui}(t) = \mu + b_i(t) + b_u(t) + q_i^T p_u(t)$$

- Varying confidence in data points

$$\min_{p*,q*,b*} \sum_{(u,i) \in \kappa} c_{ui}(r_{ui} - \mu - b_u - b_i - p_u^T q_i)^2 + \lambda (|| p_u ||^2 + || q_i ||^2 + b_u^2 + b_i^2)$$

Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. Computer 42, 8 (2009), 30–37.

Matrix Factors on Netflix



Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. Computer 42, 8 (2009), 30–37.

Implicit feedback

- Preferences elicitation (revealed vs expressed)
- Implicit feedback:
  - user opened the product
  - user flagged the product as favorite
  - user bought the product
  - time the user took to read a book
  - time spent on watching a video
  - “long” clicks
  - **weighted sum**

1. Numerical value indicates confidence not preference/rating (Classificaton instead of regression)
2. No negative feedback
3. Inherently noisy
4. Different evaluation (for ranking, or top N)

Evaluation metrics

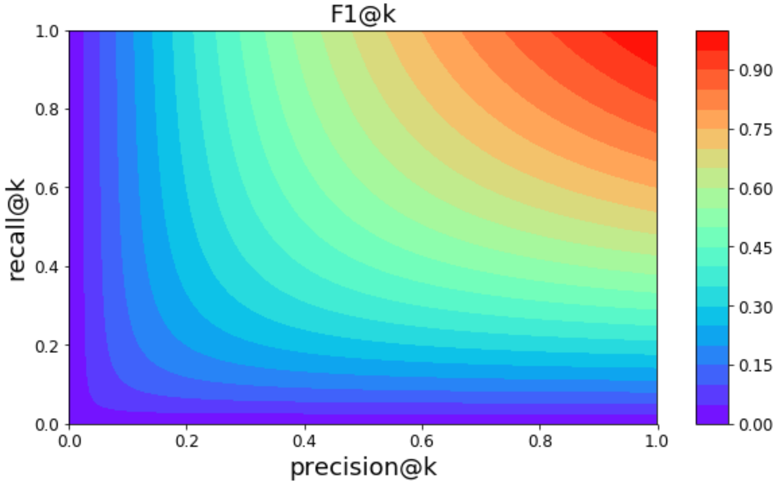
$$\text{Precision} = \frac{\#\text{tp}}{\#\text{tp} + \#\text{fp}}$$

$$\text{Recall (True Positive Rate)} = \frac{\#\text{tp}}{\#\text{tp} + \#\text{fn}}$$

- RMSE i MAE (explicit)
- Precision@k, Recall@k:
  - recommend top-k items

	Recommended	Not recommended
Preferred	True-Positive (tp)	False-Negative (fn)
Not preferred	False-Positive (fp)	True-Negative (tn)

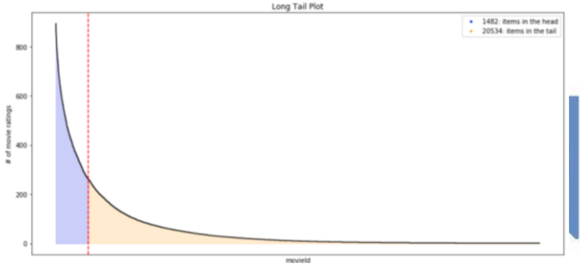
$$\text{F1@k} = \frac{2}{\frac{1}{\text{precision@k}} + \frac{1}{\text{recall@k}}}$$



- Normalized Discounted Cumulative Gain
  - ranking

$$\mathrm{DCG_p} = \sum_{i=1}^{p} \frac{rel_i}{\log_2(i+1)} \qquad \mathrm{nDCG_p} = \frac{DCG_p}{IDCG_p}$$

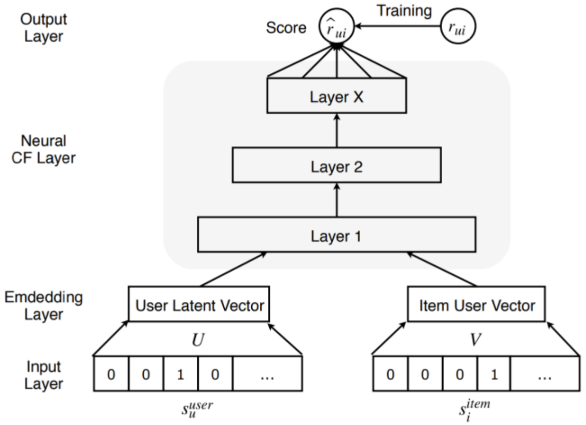
- Other: MeanAveragePrecision, Coverage, Novelty, ...
  - Recommendation bubble (exploration vs exploitation)
- Business: click-through, conversion rate, A/B test



Modern approaches to recommendation

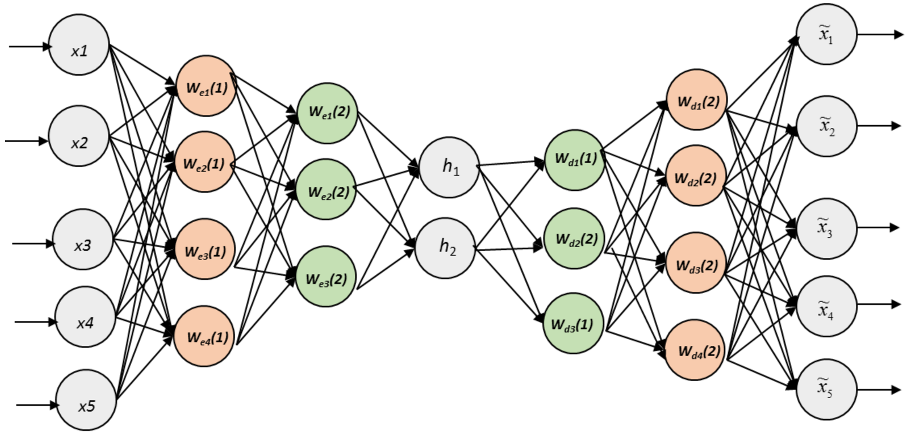
Deep Learning approaches

- Neural embedding vs Linear embedding



He et al (2017). Neural collaborative Filtering. In WWW.

- Autoencoders - Matrix completion methods



Embedding approaches

- word2vec approach
  - "E-commerce in Your Inbox: Product Recommendations at Scale" (Grbovic, Radosavljevic, Djuric et al, 2016)
  - product sequences from email receipts
  - improve ad targeting (+9% CTR)
  - prod2vec ⇔ word2vec, user2vec ⇔ paragraph2vec
- node2vec approach:
  - Stanford 2016 – "Recommending Related YouTube Videos"
  1. Use node2vec to embed videos from youtube graph
  2. Use LSH to obtain similar video candidates
  3. Use NN for link prediction

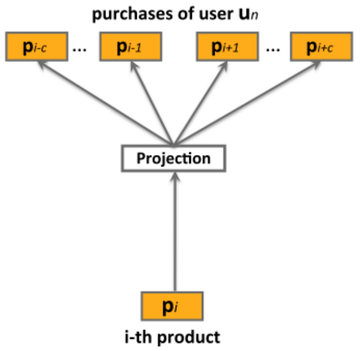


Figure 2: prod2vec skip-gram model

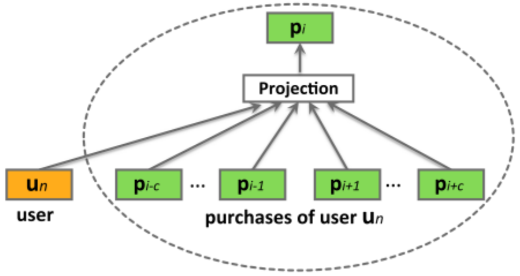
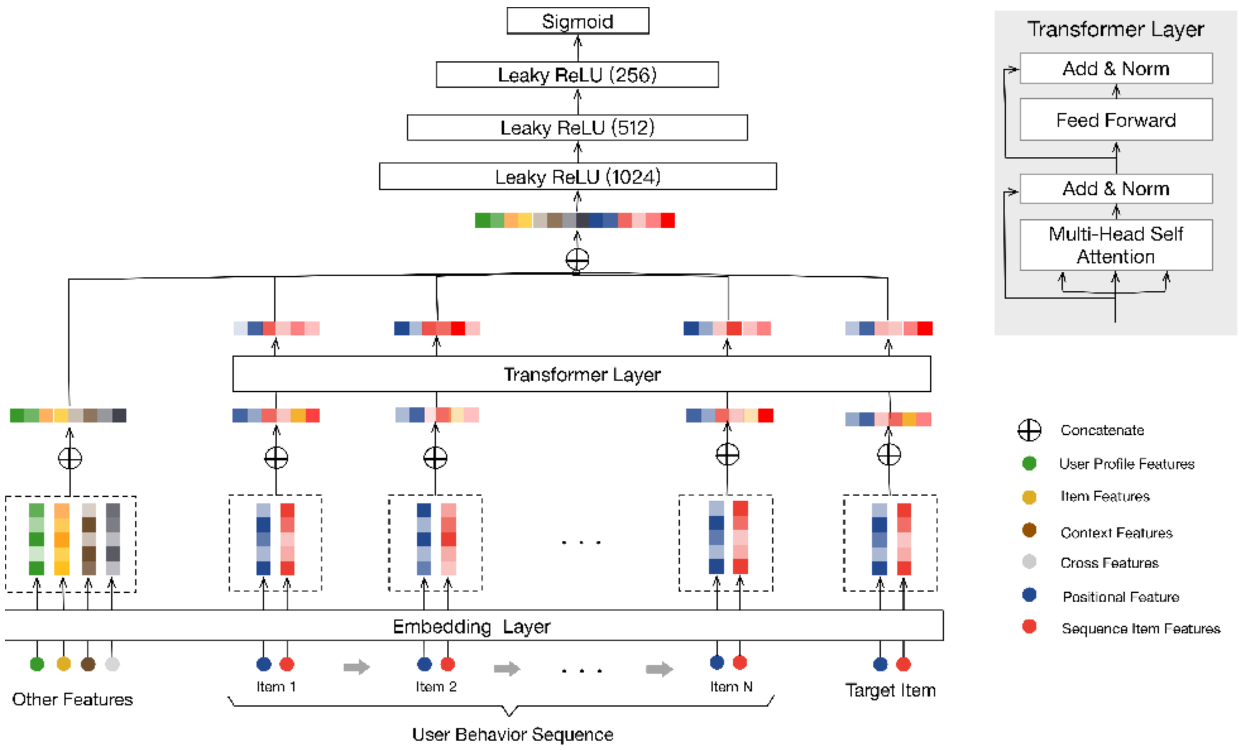


Figure 4: User embeddings for user to product predictions

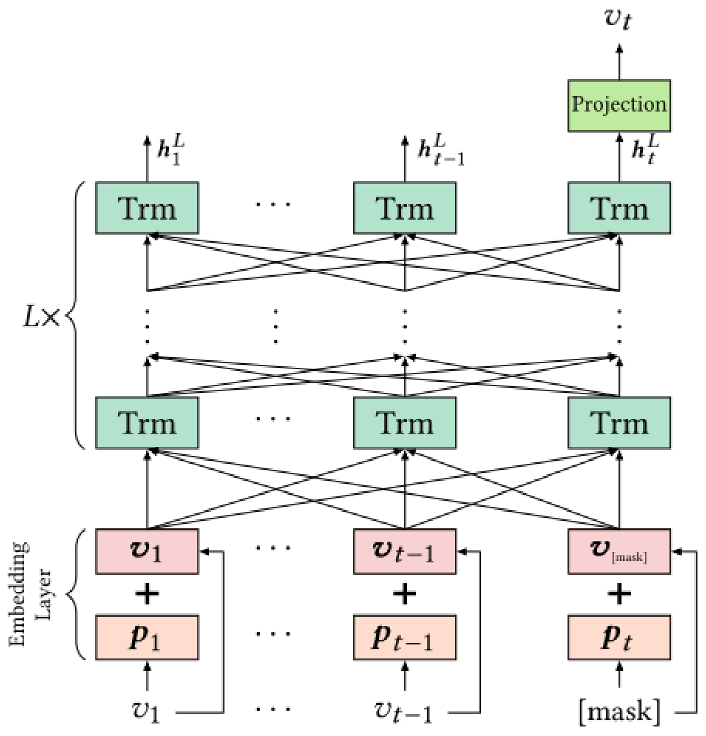
Transformer networks for RecSys

Behavior Sequence Transformer



Chen et al. (2019). Behavior Sequence Transformer for E-commerce Recommendation in Alibaba

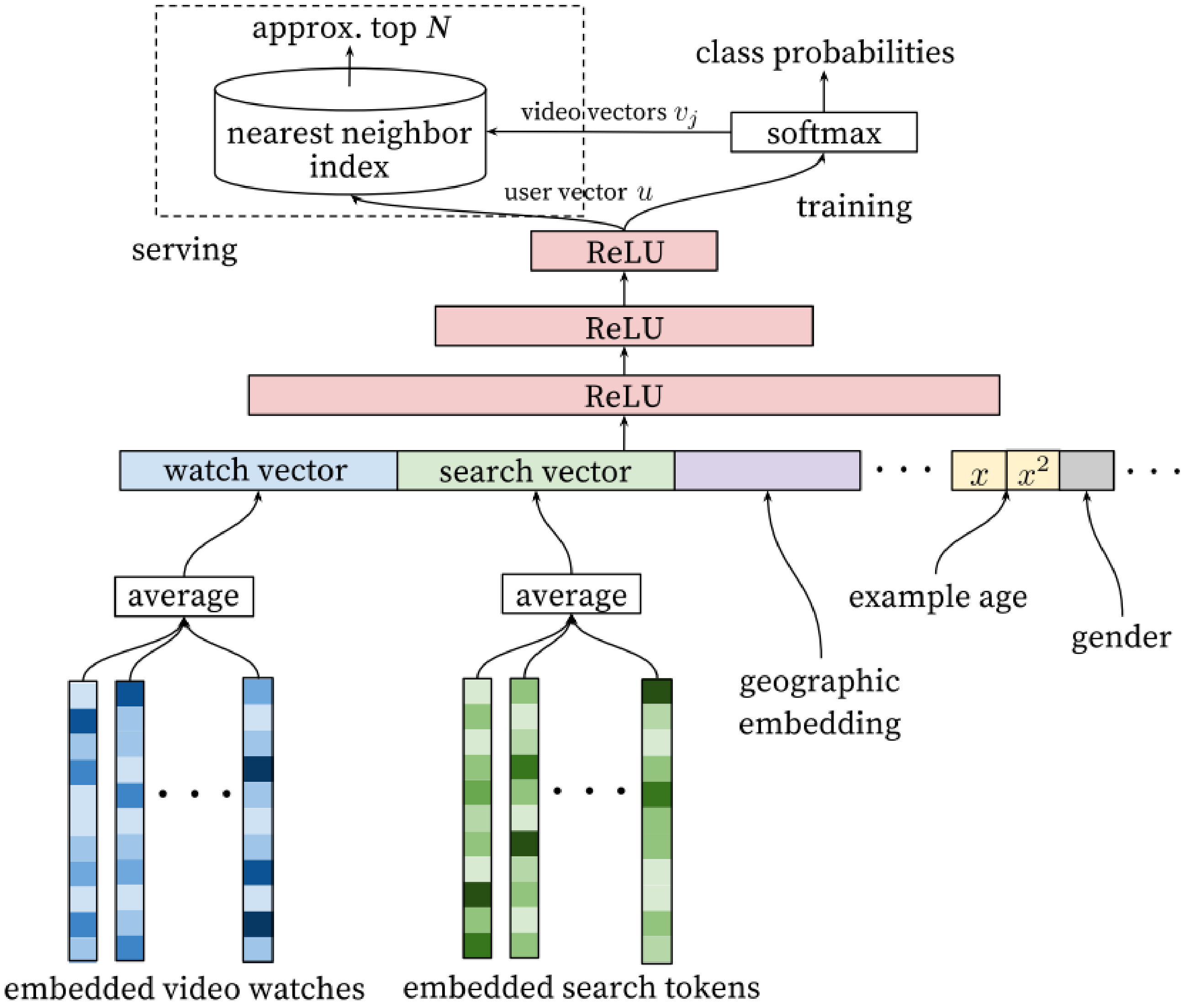
Recommendation as sequence prediction - BERT4Rec



(b) BERT4Rec model architecture.

Sun et al. (2019). BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer

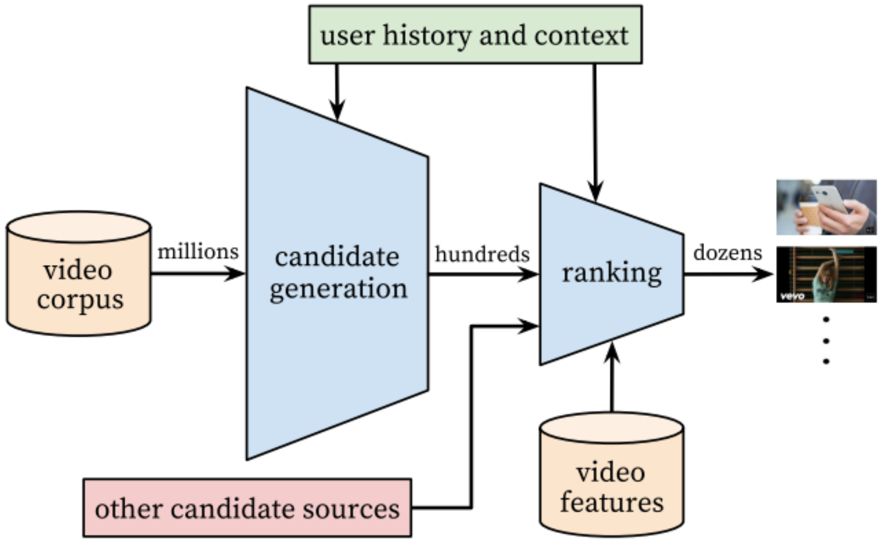
Deep NN for Youtube (2016)



- Non-linear generalization of factorization techniques
- Training:
  - Negative sampling for softmax
- Inference
  - top N
  - calibrated likelihoods not needed for serving
  - approximate kNN search in the dot product space
- Embed videos from user histories (similar to word2vec)

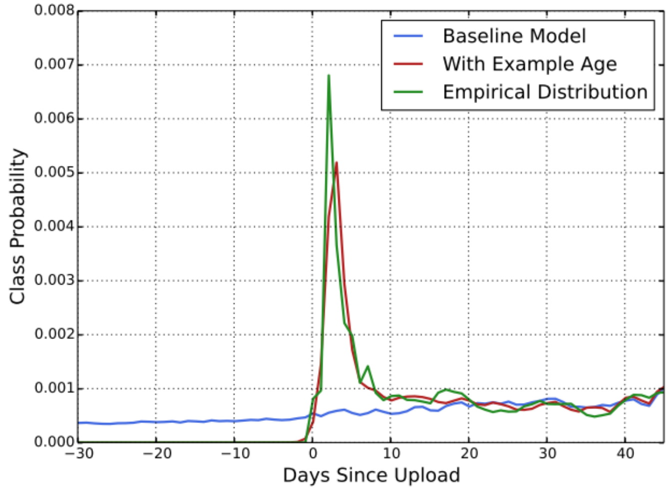
Deep NN for Youtube (2016)

- During development, offline metrics (precision, recall, ranking loss, etc.)
- Final determination, rely on A/B testing
  - measure subtle changes in click-through rate, watch time, ...
- Ranking based on “watch time per impression”

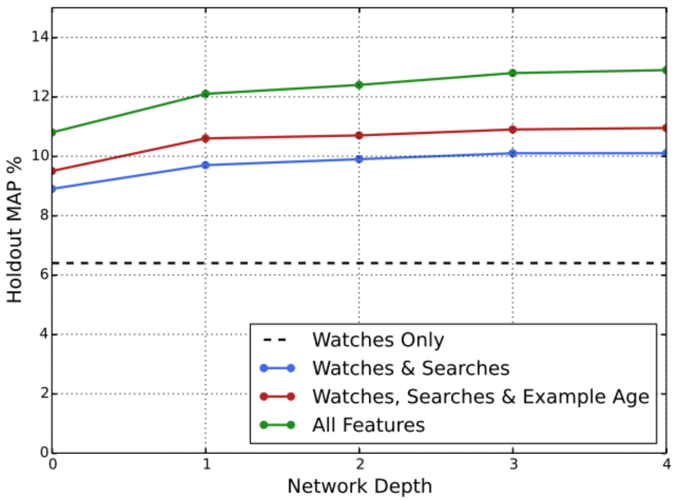


Deep NN for Youtube (2016)

Video Age:



Effect of sources and depth:

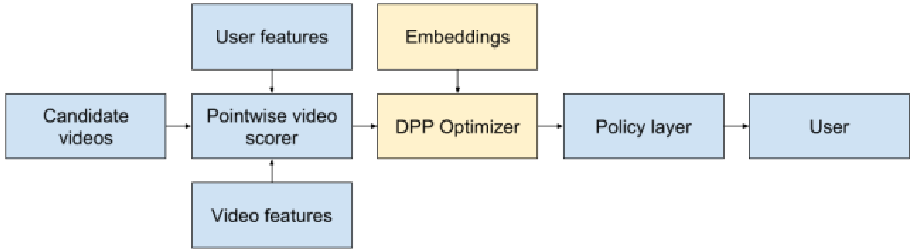


Hyper-parameters:

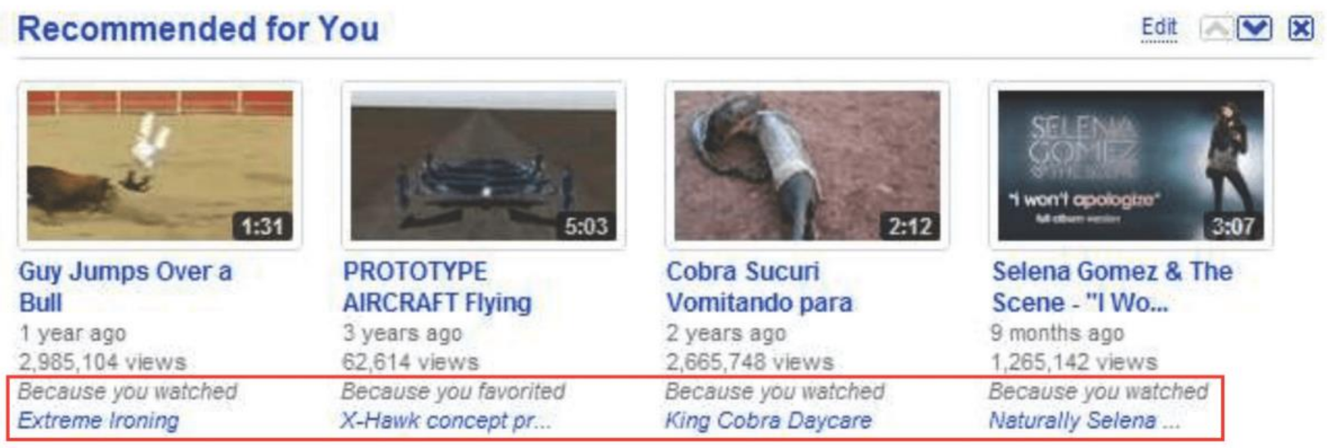
Hidden layers	weighted, per-user loss
None	41.6%
256 ReLU	36.9%
512 ReLU	36.7%
1024 ReLU	35.8%
512 ReLU → 256 ReLU	35.2%
1024 ReLU → 512 ReLU	34.7%
1024 ReLU → 512 ReLU → 256 ReLU	34.6%

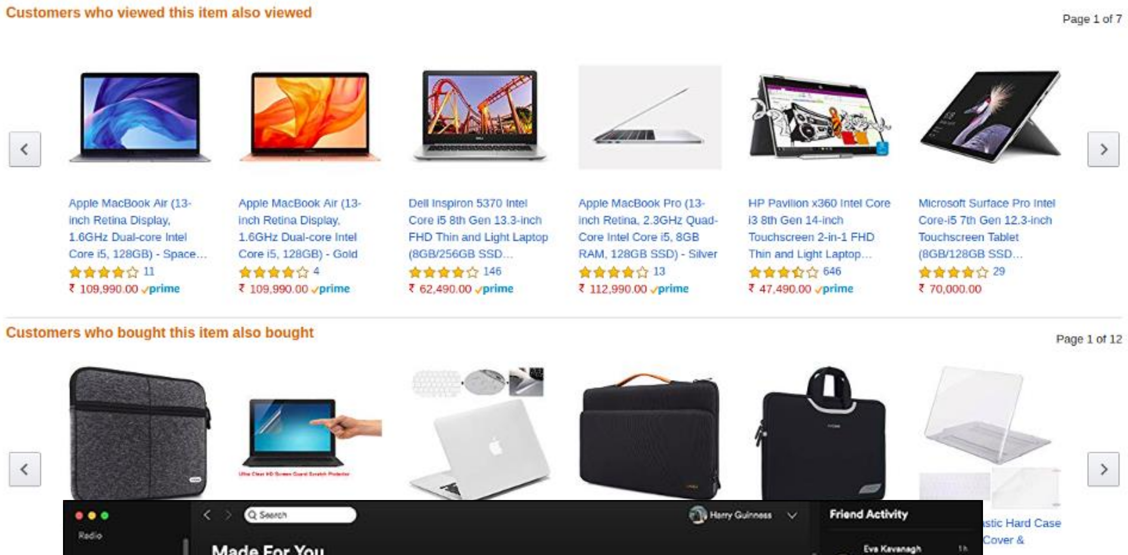
Diversity:

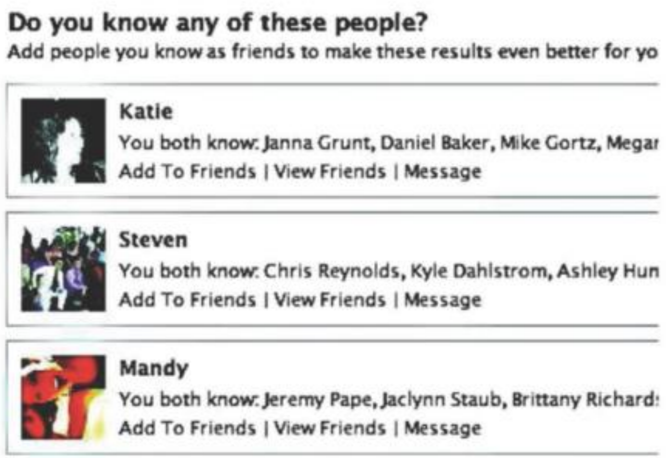
Youtube, 2018 – “Practical Diversified Recommendations on YouTube with Determinantal Point Processes”



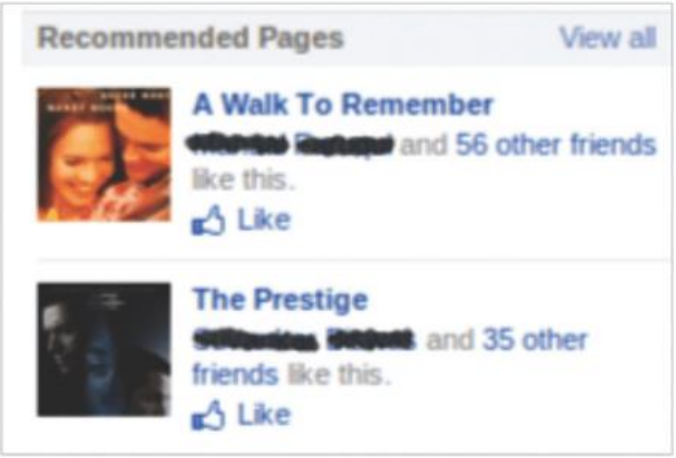
Explainability of recommendations



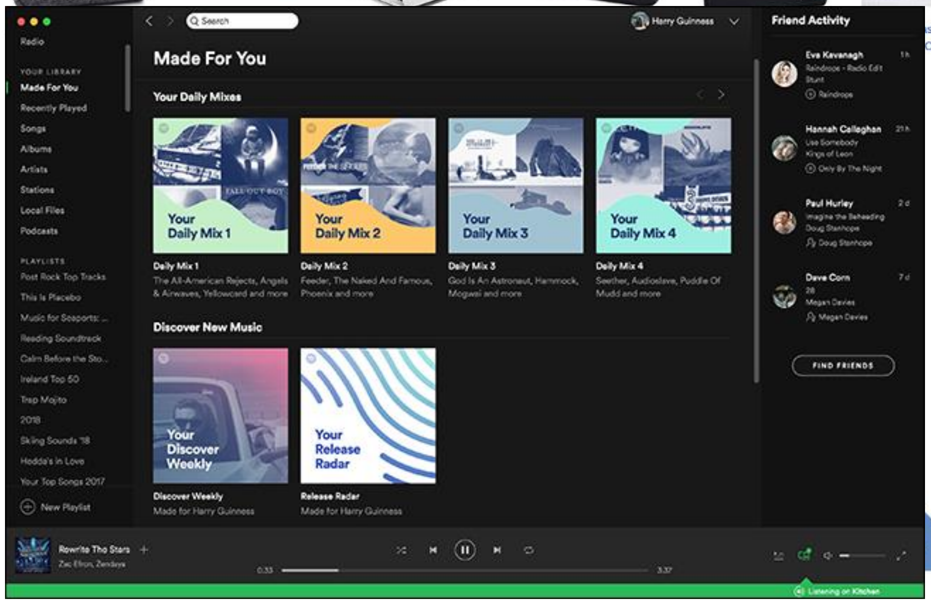




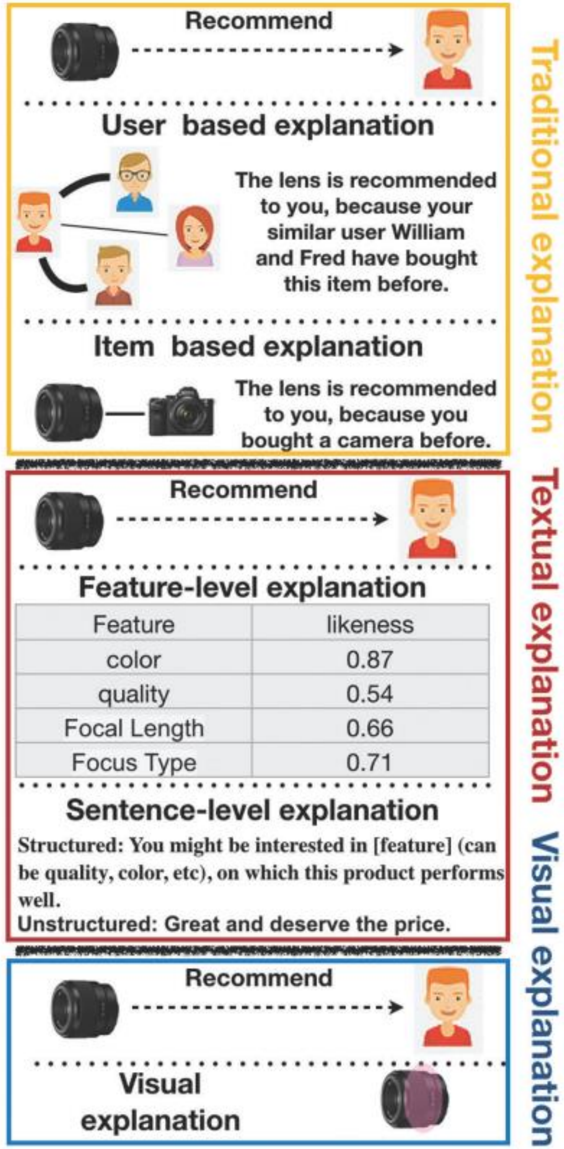
(a) "You both know"

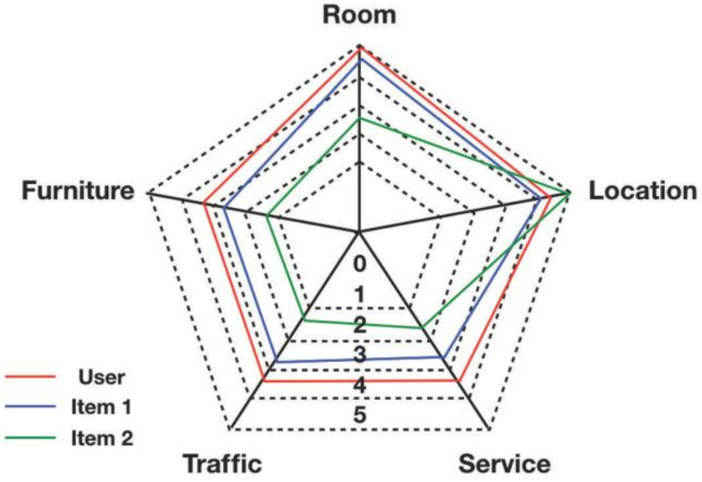


(b) "Others also liked"



Explainability of recommendations





From your Movielens profile it seems that you prefer movies tagged as space, this movie takes you in space and it feels claustrophobic to be there. It keeps you on the edge of your seat the whole time

From your Movielens profile it seems that you prefer movies tagged as visual, Gravity is unlike what we have seen on a cinema screen before and arguably it has one of the best users of 3D in a movie

From your Movielens profile it seems that you prefer movies tagged as intense, the movie s pretty intense ninety minutes, with Bullock's character constantly battling one catastrophe after another and all of it is amazing to use

ainable Recommendation: A Survey and New Perspectives

Questions?

THANK YOU