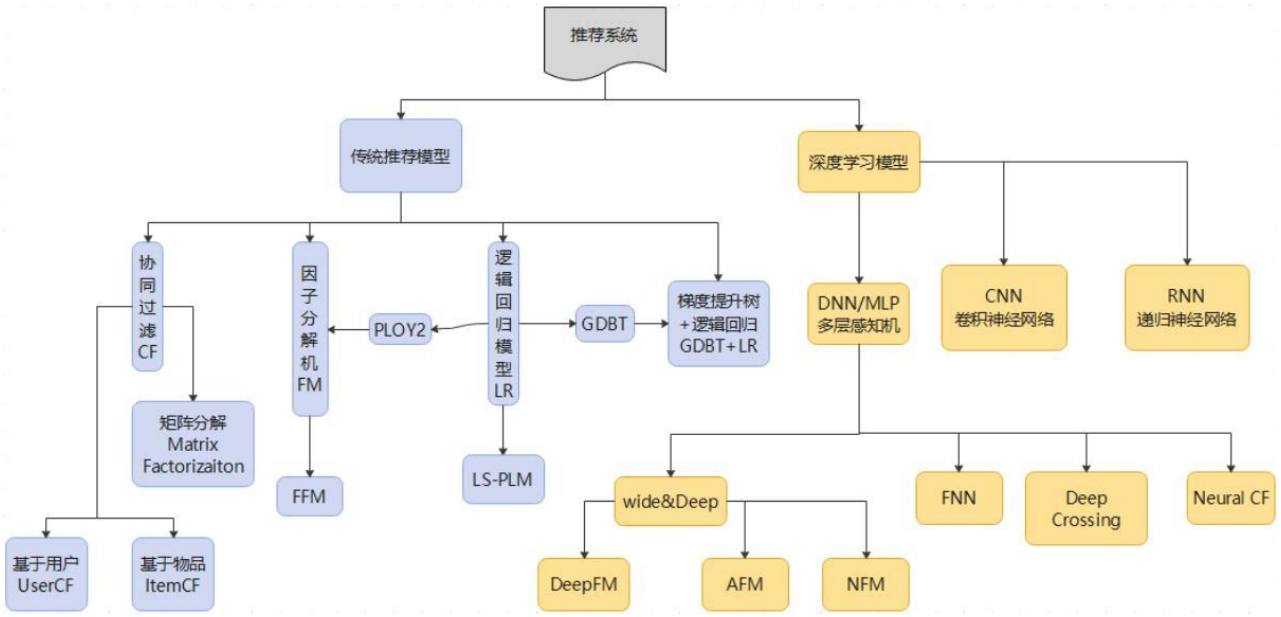
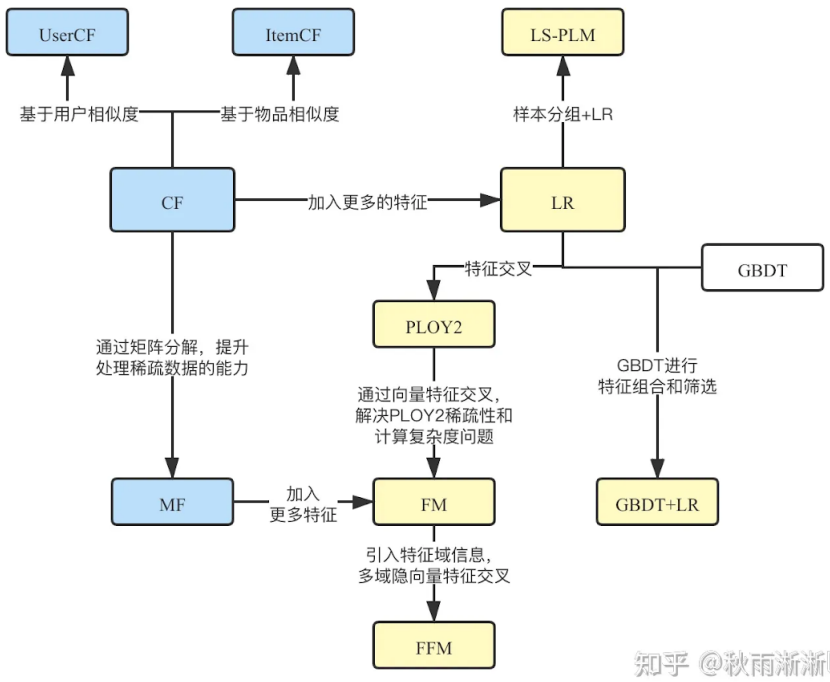
Overview

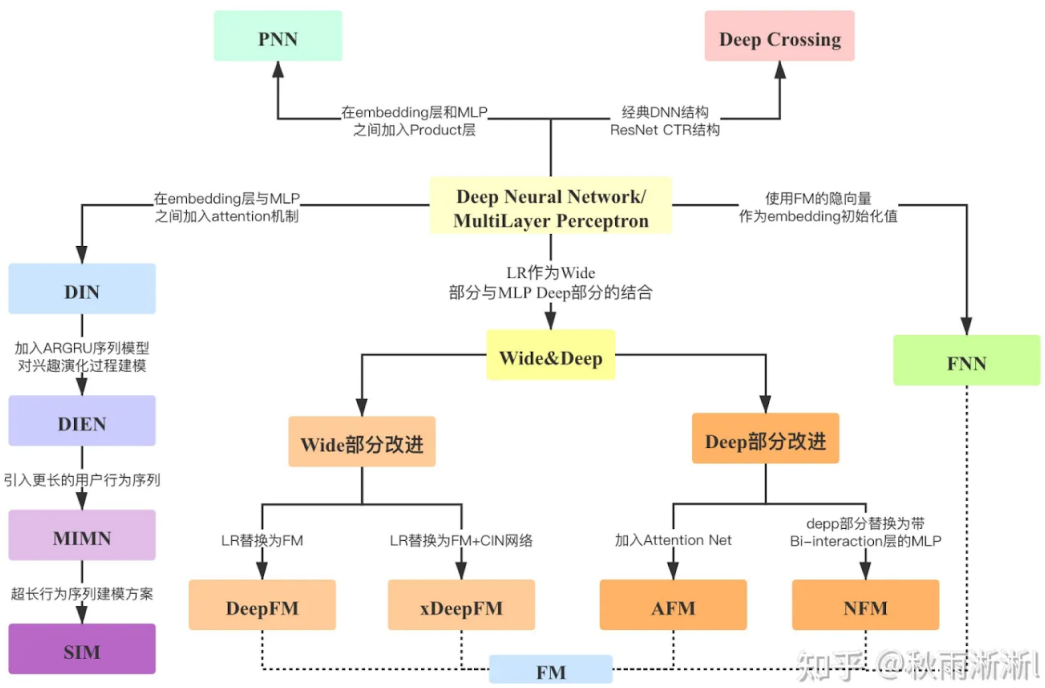
- https://zhuanlan.zhihu.com/p/407545134
- https://zhuanlan.zhihu.com/p/414309832
- https://zhuanlan.zhihu.com/p/32614346
- https://zhuanlan.zhihu.com/p/148207613



- https://zhuanlan.zhihu.com/p/414309832

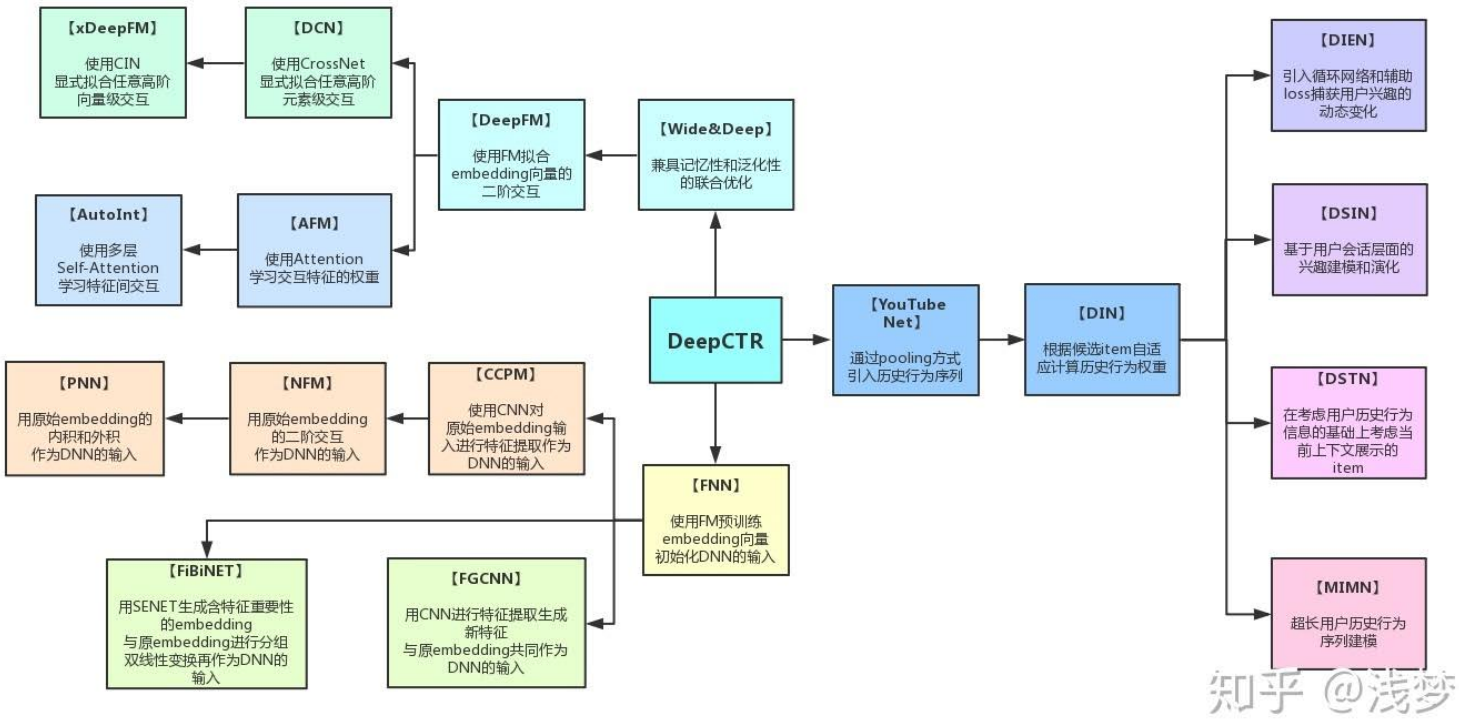


知乎 @秋雨淅淅l



知乎 @秋雨淅淅l

- https://zhuanlan.zhihu.com/p/53231955



Concept

From https://www.coursera.org/learn/unsupervised-learning-recommenders-reinforcement-learning/home/week/2

Notation:

$r(i,j) = 1$ if user j has rated movie i (0 otherwise)
$y^{(i,j)}$ = rating given by user j on movie i (if defined)
$w^{(j)}, b^{(j)}$ = parameters for user j
$x^{(i)}$ = feature vector for movie i

For user j and movie i, predict rating: $w^{(j)} \cdot x^{(i)} + b^{(j)}$
$m^{(j)}$ = no. of movies rated by user j

In below,

the two blue underlined parts are exactly the same

`n_u = number of users`

`n_m = number of movies`

`(i,j):r(i,j)=1 reads "all values of (i,j) that r(i,j) = 1"`

Collaborative filtering

Cost function to learn $w^{(1)}, b^{(1)}, \cdots w^{(n_u)}, b^{(n_u)}$:

$$\min_{w^{(1)},b^{(1)},\cdots,w^{(n_u)},b^{(n_u)}} \frac{1}{2}\sum_{j=1}^{n_u}\sum_{i:r(i,j)=1}\left(w^{(j)}\cdot x^{(i)}+b^{(j)}-y^{(i,j)}\right)^2+\frac{\lambda}{2}\sum_{j=1}^{n_u}\sum_{k=1}^{n}\left(w_k^{(j)}\right)^2$$

Cost function to learn $x^{(1)},\cdots,x^{(n_m)}$:

$$\min_{x^{(1)},\cdots,x^{(n_m)}} \frac{1}{2}\sum_{i=1}^{n_m}\sum_{j:r(i,j)=1}\left(w^{(j)}\cdot x^{(i)}+b^{(j)}-y^{(i,j)}\right)^2+\frac{\lambda}{2}\sum_{i=1}^{n_m}\sum_{k=1}^{n}\left(x_k^{(i)}\right)^2$$

Put them together:

$$\frac{1}{2}\sum_{(i,j):r(i,j)=1}\left(w^{(j)}\cdot x^{(i)}+b^{(j)}-y^{(i,j)}\right)^2+\frac{\lambda}{2}\sum_{j=1}^{n_u}\sum_{k=1}^{n}\left(w_k^{(j)}\right)^2+\frac{\lambda}{2}\sum_{i=1}^{n_m}\sum_{k=1}^{n}\left(x_k^{(i)}\right)^2$$

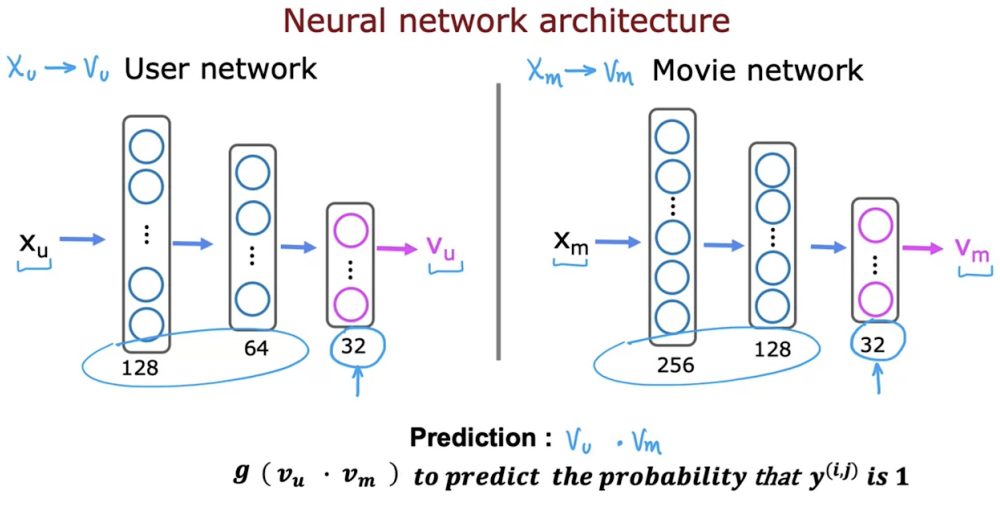
$$w_i^{(j)} = w_i^{(j)} - \alpha\frac{\partial}{\partial w_i^{(j)}} J(w,b,x)$$
$$b^{(j)} = b^{(j)} - \alpha\frac{\partial}{\partial b^{(j)}} J(w,b,x)$$
$$x_k^{(i)} = x_k^{(i)} - \alpha\frac{\partial}{\partial x_k^{(i)}} J(w,b,x)$$

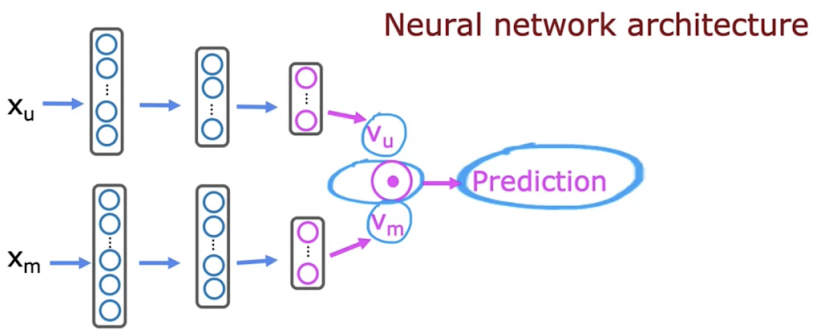
Collaborative filtering vs Content-based filtering

Collaborative filtering:
Recommend items to you based on ratings of users who gave similar ratings as you

Content-based filtering:
Recommend items to you based on features of user and item to find good match

Deep learning for content-based filtering





Cost function $J = \sum_{(i,j):r(i,j)=1} \left(v_u{}^{(j)} \cdot v_m{}^{(i)} - y^{(i,j)}\right)^2$ + NN regularization term

Tensorflow Recommender

- https://www.kaggle.com/code/kanruwang/tensorflow-recommender-two-tower-multitask (two-tower retrieval and multi-task learning)

Factorization Machines

- https://zhuanlan.zhihu.com/p/50426292
- https://zhuanlan.zhihu.com/p/153500425
- https://medium.com/@jchen001/10-recommendation-techniques-introduction-comparison-7ba4a3a2c940

$$\tilde{y}(x) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} < v_i, v_j > x_i x_j$$

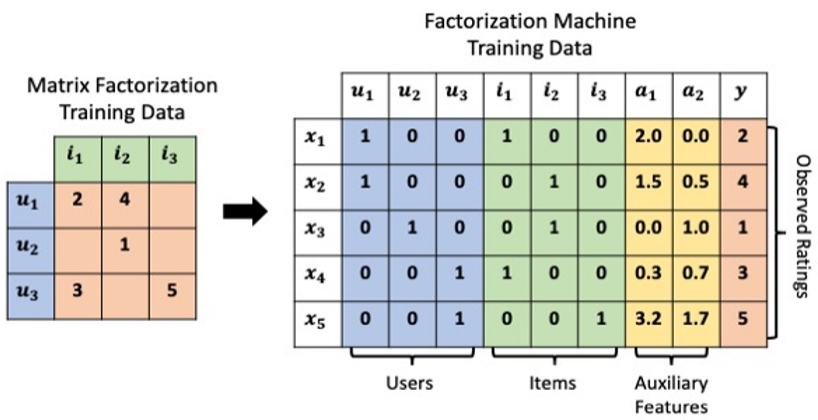
$< \cdot, \cdot >$ 是两个 k 维向量的内积：

$$< v_i, v_j >= \sum_{f=1}^{k} v_{i,f} v_{j,f}$$

A 2-way FM (degree $d = 2$) captures all single and pairwise interactions between variables:

- w_0 is the global bias.
- w_i models the strength of the i-th variable.
- $\hat{w}_{i,j} := \langle \mathbf{v}_i, \mathbf{v}_j \rangle$ models the interaction between the i-th and j-th variable. Instead of using an own model parameter $w_{i,j} \in \mathbb{R}$ for each interaction, the FM models the interaction by factorizing it. We will see later on, that this is the key point which allows high quality parameter estimates of higher-order interactions ($d \geq 2$) under sparsity.

- https://towardsdatascience.com/factorization-machines-for-item-recommendation-with-implicit-feedback-data-5655a7c749db training data format:



- https://stats.stackexchange.com/questions/108901/difference-between-factorization-machines-and-matrix-factorization

1. **Matrix Factorization cannot use side-features. For a movie recommendation system, we cannot use the movie genres, its language etc in Matrix Factorization. The factorization itself has to learn these from the existing interactions. But we can pass this info in Factorization Machines.**
2. **Factorization Machines can also be used for other prediction tasks such as Regression and Binary Classification. This is usually not the case with Matrix Factorization**

- **FM implementation packages:**
  - https://xlearn-doc.readthedocs.io/en/latest/demo/index.html
  - **For FM, the input data format can be CSV or libsvm. For FFM, the input data should be the libffm format:** https://xlearn-doc.readthedocs.io/en/latest/python_api/

Wide & Deep

- https://zhuanlan.zhihu.com/p/53361519
- https://zhuanlan.zhihu.com/p/54464005

Field-aware Factorization Machines

- https://zhuanlan.zhihu.com/p/170607706

DeepFM

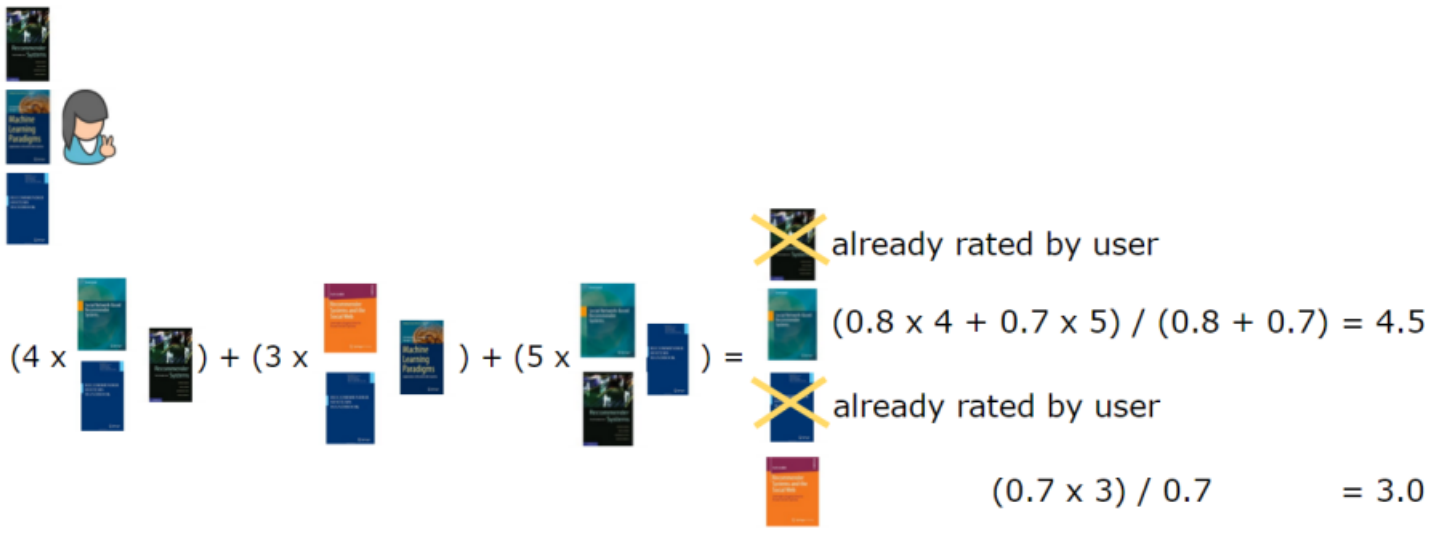
- **Implementation best package**
  - https://deepctr-doc.readthedocs.io/en/latest/index.html
    1. https://tianchi.aliyun.com/mas-notebook/preview/51950/54206/-1?lang=
    2. https://www.jianshu.com/p/3e40157c3fae
- **Secondary packages**
  - https://elliot.readthedocs.io/en/latest/elliot/elliot.recommender.neural.DeepFM.html
  - https://github.com/RUCAIBox/RecBole
  - https://github.com/ChenglongChen/tensorflow-DeepFM **(Slides show how to integrate text info, but didn't use embedding from Bert (Bert was only invented later)** https://github.com/ChenglongChen/tensorflow-XNN/blob/master/doc/Mercari_Price_Suggesion_Competition_ChenglongChen_4th_Place.pdf**)**
  - https://pytorch.org/torchrec/torchrec.models.html

Collaborative filtering

From https://buildingrecommenders.wordpress.com/2015/11/18/overview-of-recommender-algorithms-part-2/

Neighbourhood methods (aka memory-based CF)

- User-based collaborative filtering
  - See link
- Item-based collaborative filtering
  - For each of the 3 books liked by the lady, pick the top two most similar books
  - The number 4, 3, 5 are how this lady rated these 3 books. It makes sense that she would also give related books the same ratings.
  - The number 0.8, 0.7 come from the Similarity Matrix for books.



Model-based methods

- Matrix Factorization

Content based filtering

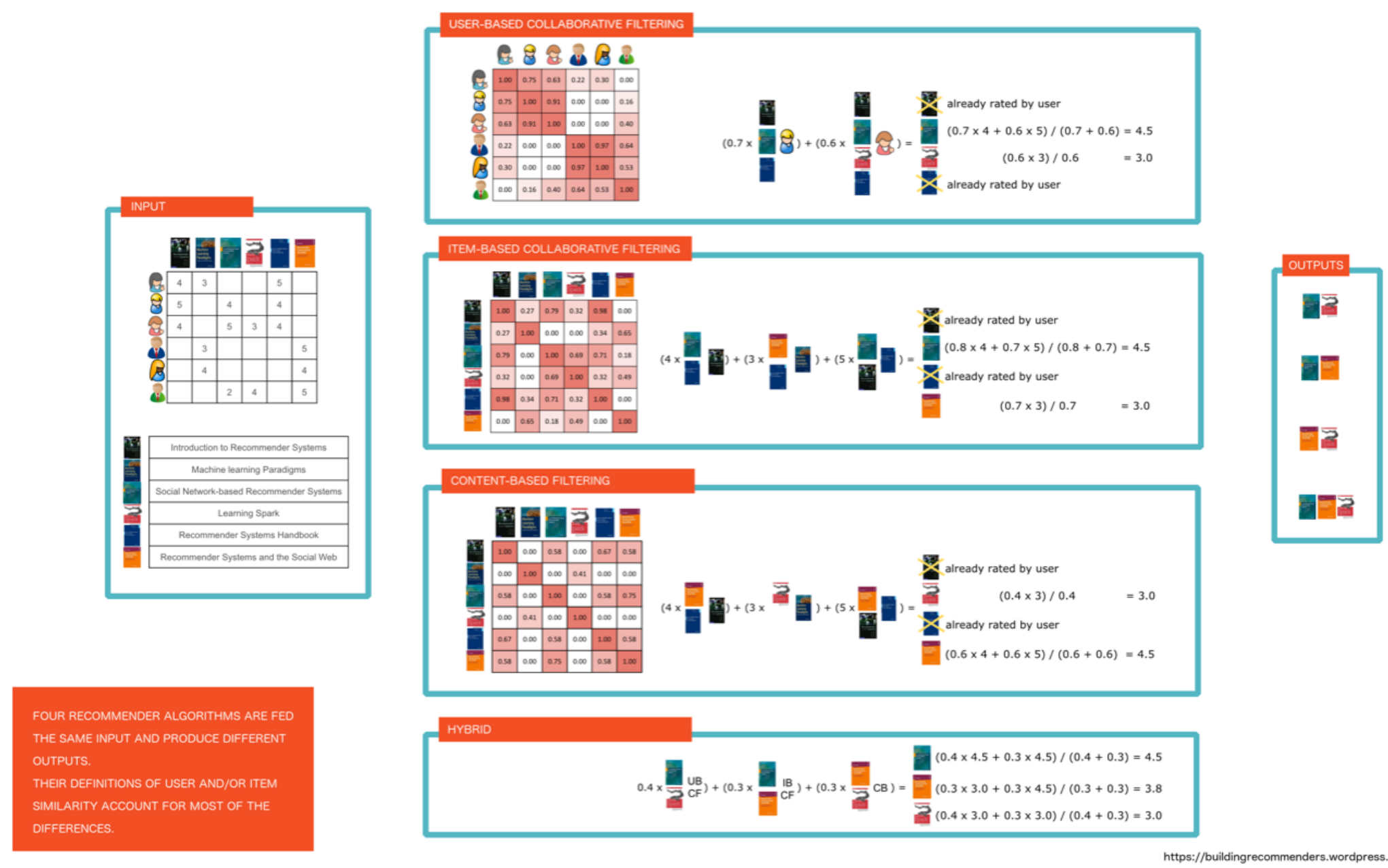
- E.g. TF-IDF / Information Retrieval from book title
- https://buildingrecommenders.wordpress.com/2015/11/19/overview-of-recommender-algorithms-part-3/

Hybrid (Collaborative filtering + Content based filtering)

- https://buildingrecommenders.wordpress.com/2015/11/20/overview-of-recommender-algorithms-part-4/

Traditional Method Comparison

- https://buildingrecommenders.wordpress.com/2015/11/23/overview-of-recommender-algorithms-part-5/

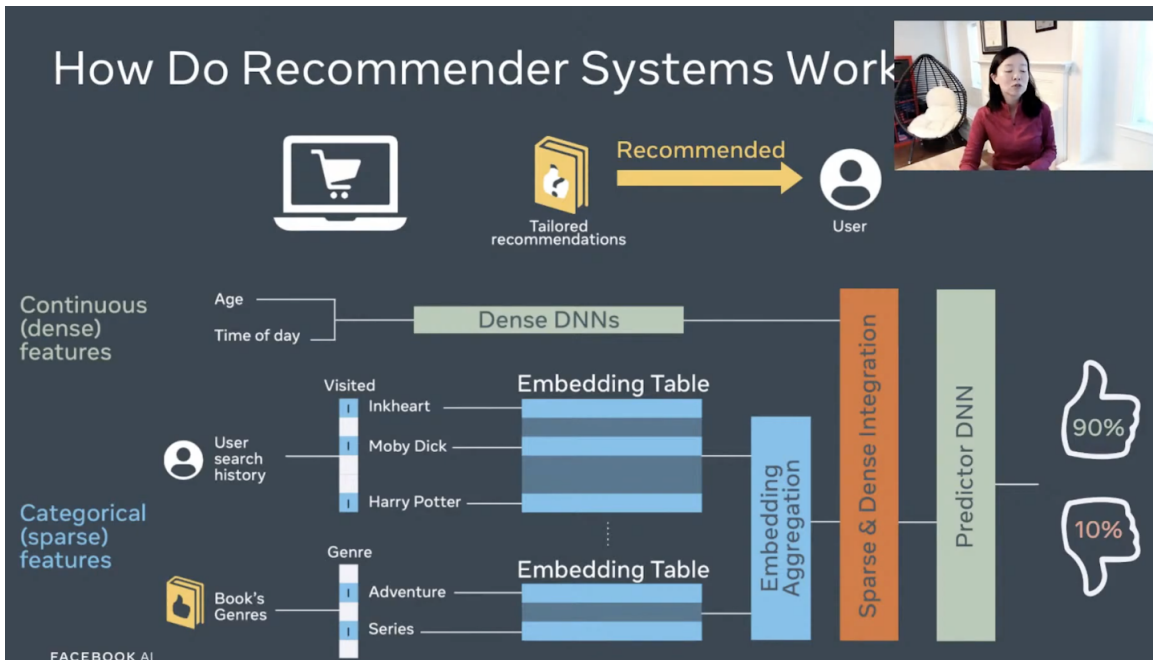


Others

LightFM - Matrix factorization that incorporates user metadata and item metadata

- https://github.com/lyst/lightfm
- https://stackoverflow.com/questions/33757974
- https://www.youtube.com/watch?v=EgE0DUrYmo8
- https://towardsdatascience.com/how-i-would-explain-building-lightfm-hybrid-recommenders-to-a-5-year-old-b6ee18571309
- https://github.com/tdeboissiere/Kaggle/blob/master/Ponpare/ponpare_lightfm.ipynb
- https://www.kaggle.com/code/niyamatalmass/lightfm-hybrid-recommendation-system
- Summary
  - The fit function takes:
    1. A required user-item interaction matrix (filled with 0s and 1s)

2. An optional user-item interaction weight (rating strength) matrix (filled with e.g. 1, 2, 3, 4, 5)
3. An optional user feature matrix from one-hot encoding user features (need to bin each continuous variable)
4. An optional item feature matrix from one-hot encoding item features (need to bin each continuous variable)

- I don’t know how matrix factorization is done with these tables, but after calculating the representations/embeddings, the inference for a user-item pair would be like:
  1. The representation for the book title “Intro to Machine Learning” is the element-wise sum of representations for “intro”, “machine” and “learning”.
  2. The representation for a female user with ID 100 is the element-wise sum of representations for “female” and “ID 100”.
  3. Calculate the dot product of the two vectors above, and then add the bias scalar of the user, and add the bias scalar of the item.
  4. (May use a sigmoid function at the end if making binary prediction)

https://www.youtube.com/watch?v=5xcd0V9m6Xs