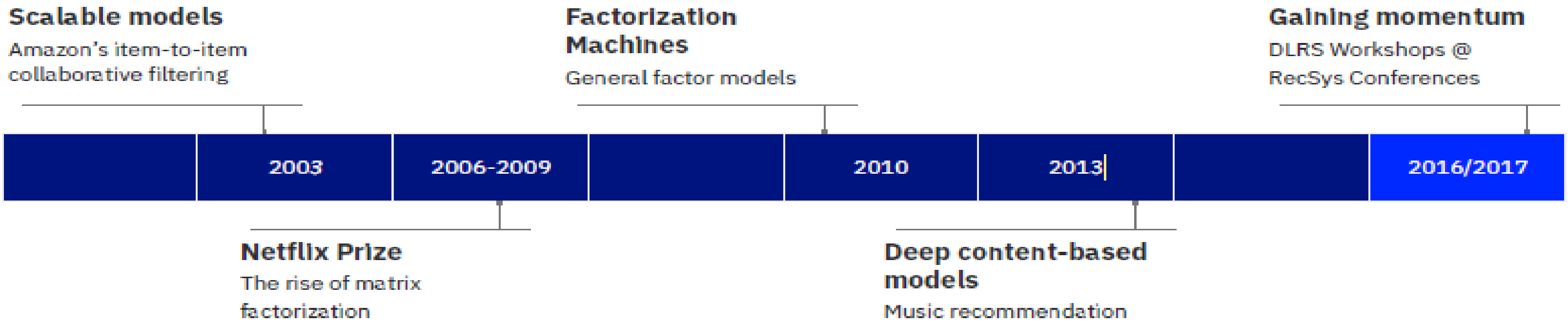
RECOMMENDER SYSTEMS

- PINAKI BHAGAT
- SYDNEY CORREA

Collaborative Filtering

Recommender Systems

- More than 80 per cent of the TV shows people watch on Netflix are discovered through the platform's recommendation system
- Nearly two decades ago, Amazon.com launched recommendations to millions of customers over millions of items, helping people discover what they might not have found on their own



Types of Recommender Systems

- Content based filtering
- Collaborative filtering
- Association rules learning
- Knowledge based systems
- Hybrid approaches

Content based Recommender Systems

Recommendations based on content of items rather than on other users opinion/interests

Content based Recommender Systems

“The Hobbit” has very similar ratings against these same attributes

“Lord of the Rings”

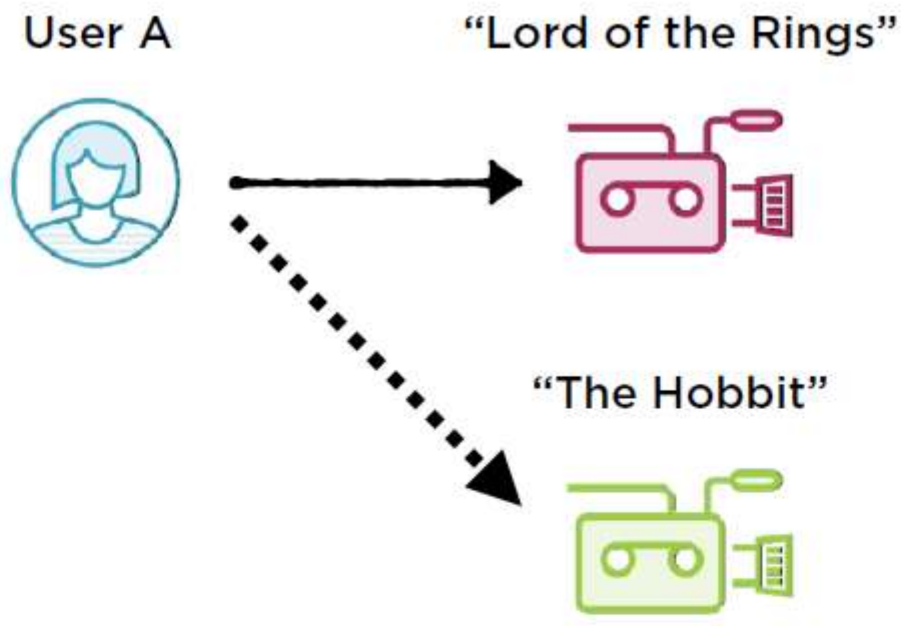
	Rating
Direction	10
Cast	8
Cinematography	10
Story	9

“The Hobbit”

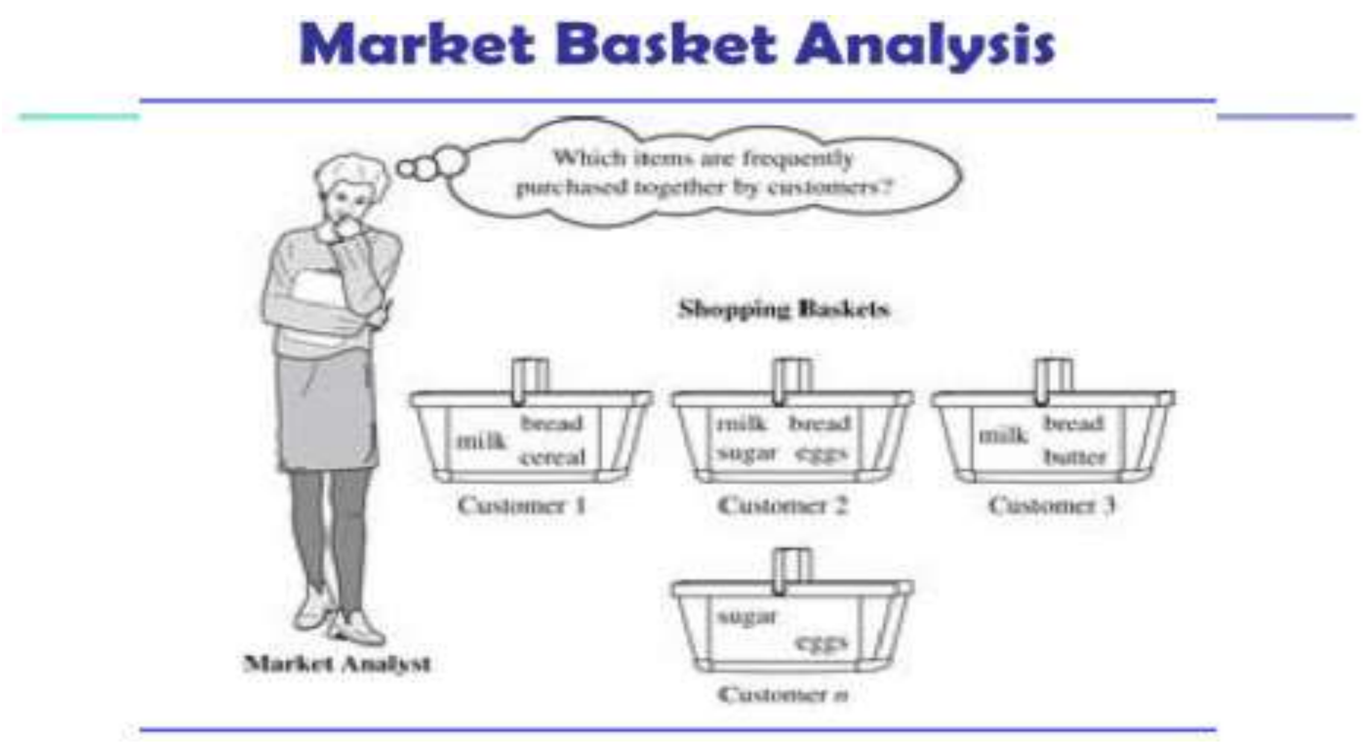
Rating
9.5
8
9
10

Content based Recommender Systems

Recommend "The Hobbit" to User A



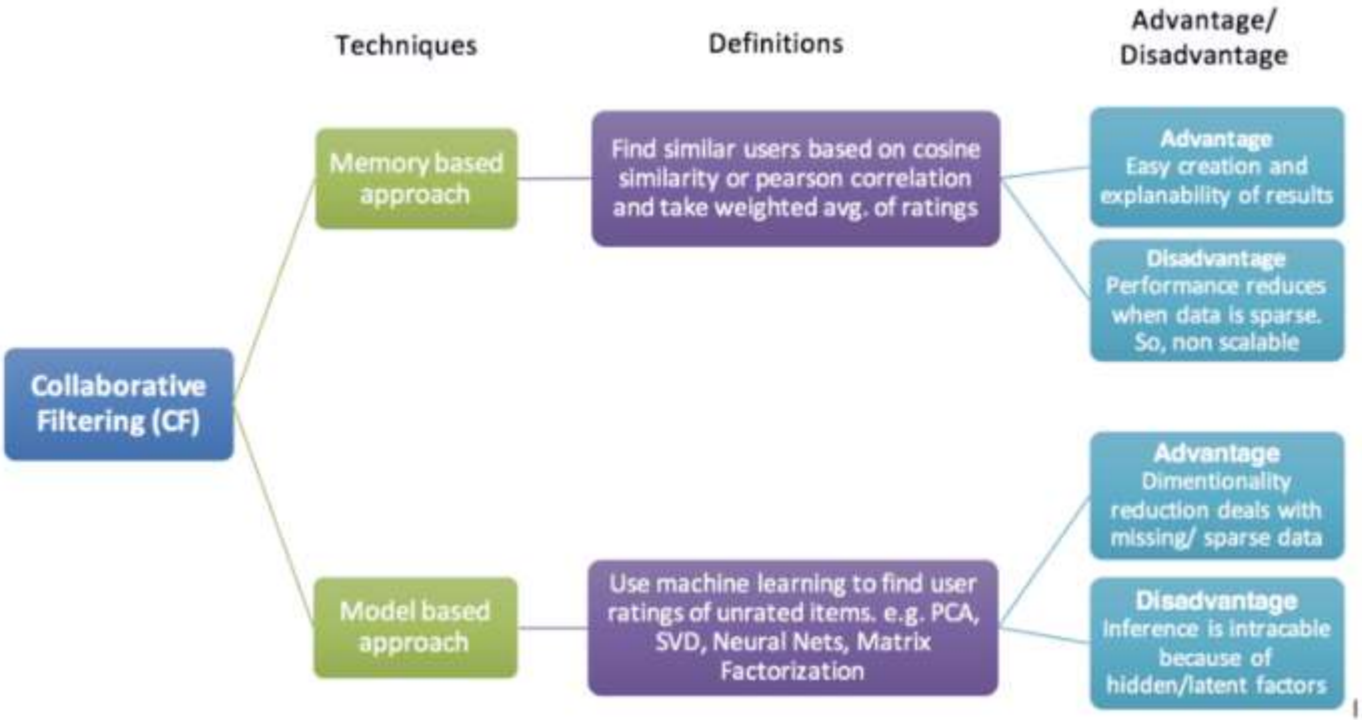
Association Rules based Recommender Systems



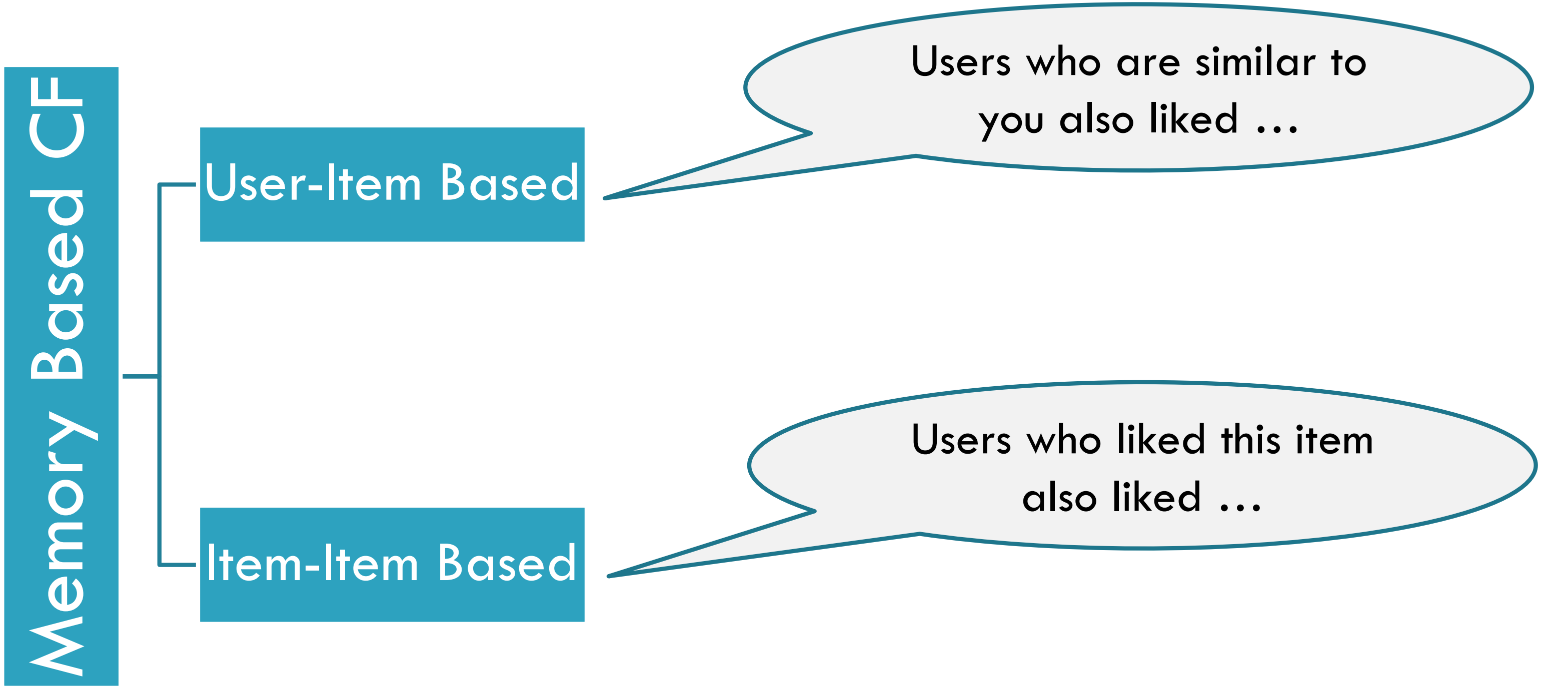
Collaborative Recommender Systems

Find users who share the same interests in the past to predict what the current user will be interested in.

Types of Collaborative Filtering



Memory based Collaborative Filtering



Memory based Collaborative Filtering

Approaches to memory based CF:

- Cosine similarity
- Pearson correlation coefficient
- KNN (non parametric ML clustering)

Rating Matrix

	P_1	P_2	P_3	P_4	P_5
U_1	3	4	-	-	-
U_2	3	2	-	-	5
U_3	-	2	-	5	4
U_4	-	-	4	-	-
U_5	1	-	-	-	-
U_6	3	4	-	-	5

Users

Products

Memory based Collaborative Filtering

Approaches to memory based CF:

- Cosine similarity

  Two users' similarity is measured as the cosine of the angle between the two users' vectors. For users u and u′, the cosine similarity is:

$$sim(u,u') = cos(\theta) = \frac{\mathbf{r}_u \cdot \mathbf{r}_{u'}}{\|\mathbf{r}_u\| \|\mathbf{r}_{u'}\|} = \sum_i \frac{r_{ui} r_{u'i}}{\sqrt{\sum_i r_{ui}^2} \sqrt{\sum_i r_{u'i}^2}}$$

Rating Matrix

Users

	P_1	P_2	P_3	P_4	P_5
U_1	3	4	-	-	-
U_2	3	2	-	-	5
U_3	-	2	-	5	4
U_4	-	-	4	-	-
U_5	1	-	-	-	-
U_6	3	4	-	-	5

Products

Memory based Collaborative Filtering

Approaches to memory based CF:

- Cosine similarity

  We can predict user-u's rating for product-i by taking weighted sum of product-i ratings from all other users (u′s) where weighting is similarity number between each user and user-u.

$$\hat{r}_{ui} = \frac{\sum_{u'} sim(u,u')r_{u'i}}{\sum_{u'} |sim(u,u')|}$$

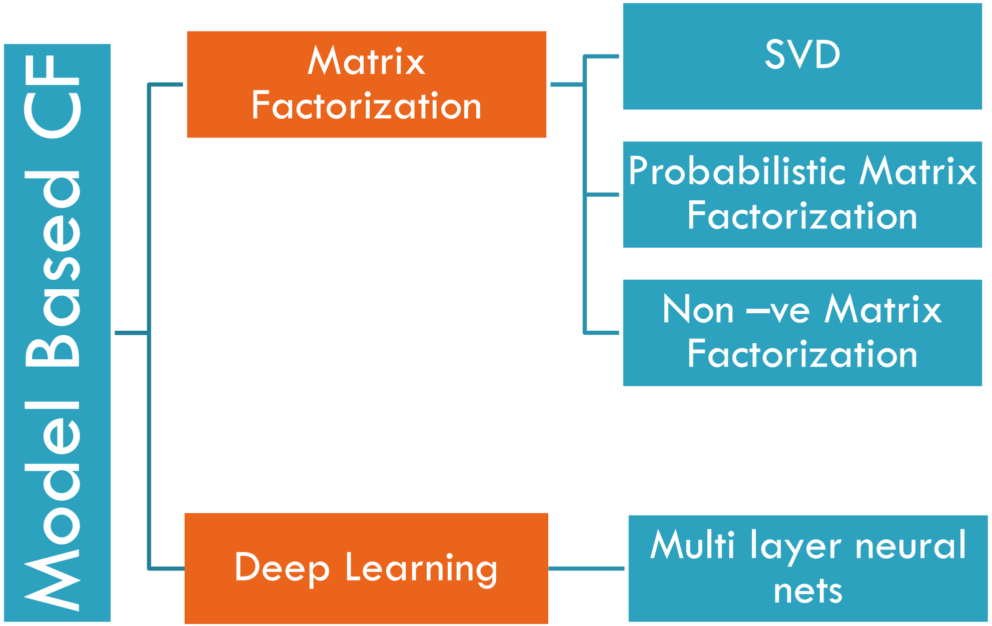
Rating Matrix

Users

	P_1	P_2	P_3	P_4	P_5
U_1	3	4	-	-	-
U_2	3	2	-	-	5
U_3	-	2	-	5	4
U_4	-	-	4	-	-
U_5	1	-	-	-	-
U_6	3	4	-	-	5

Products

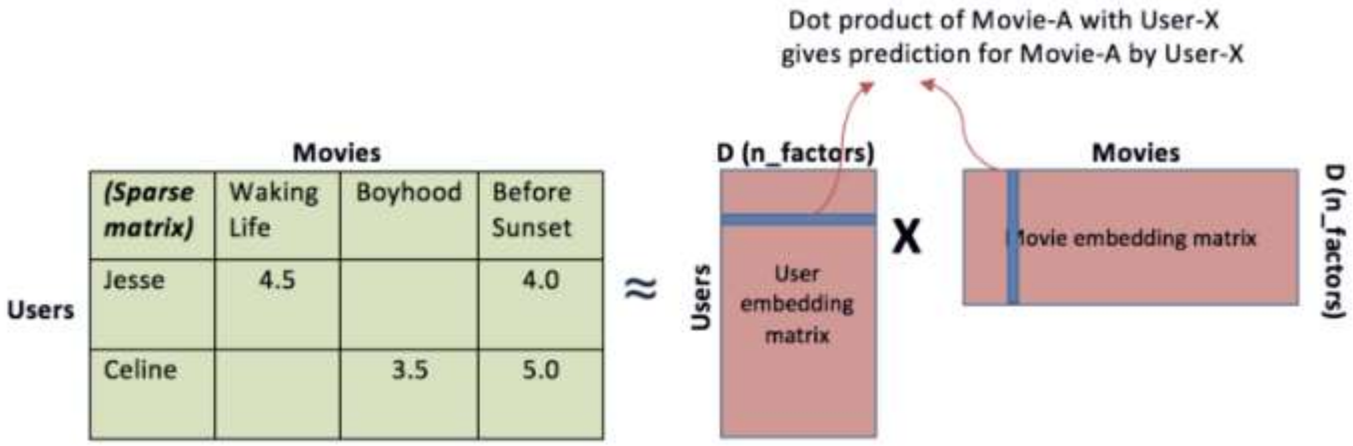
Model based Collaborative Filtering



Model based Collaborative Filtering

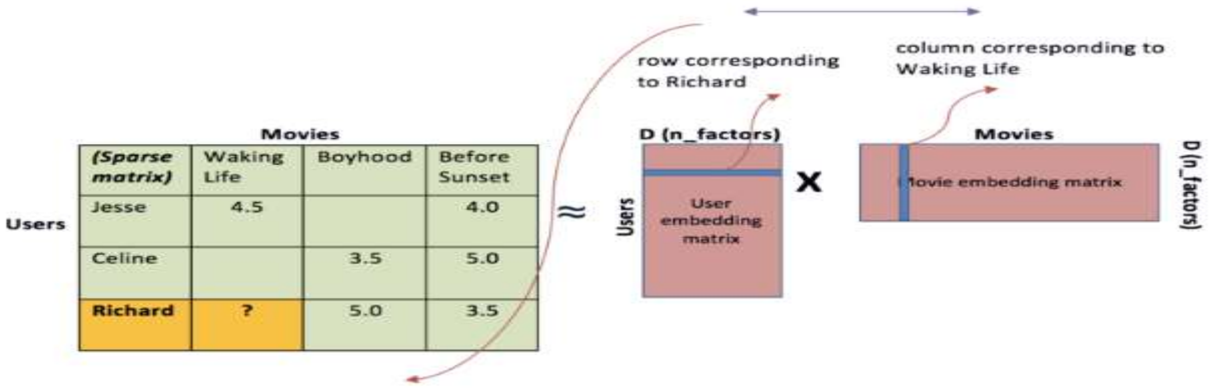
- Matrix Factorization

The idea behind such models is that attitudes or preferences of a user can be determined by a small number of hidden factors known as Embeddings



Model based Collaborative Filtering

- Neural Nets/Deep Learning



Linear and nonlinear layers

Input (X) → Target (Y)

Users	Movies
Jesse	Waking Life
Jesse	Boyhood
Jesse	Before Sunset
Celine	Waking Life
Celine	Boyhood
Celine	Before Sunset
Richard	**Waking Life**
Richard	Boyhood
Richard	Before Sunset

User latent features	Movie latent features
0.2, 0.4, 2.8, 4.8, 2.4	0.1, 0.5, 5.0, 3.7, 2.8
....	.....
....	.....
....	.....
....	.....
....	.....
2.2, 1.4, 2, 1.8, 4.4	0.1, 0.25, 4.5, 3.1, 2
....	.....
....	.....

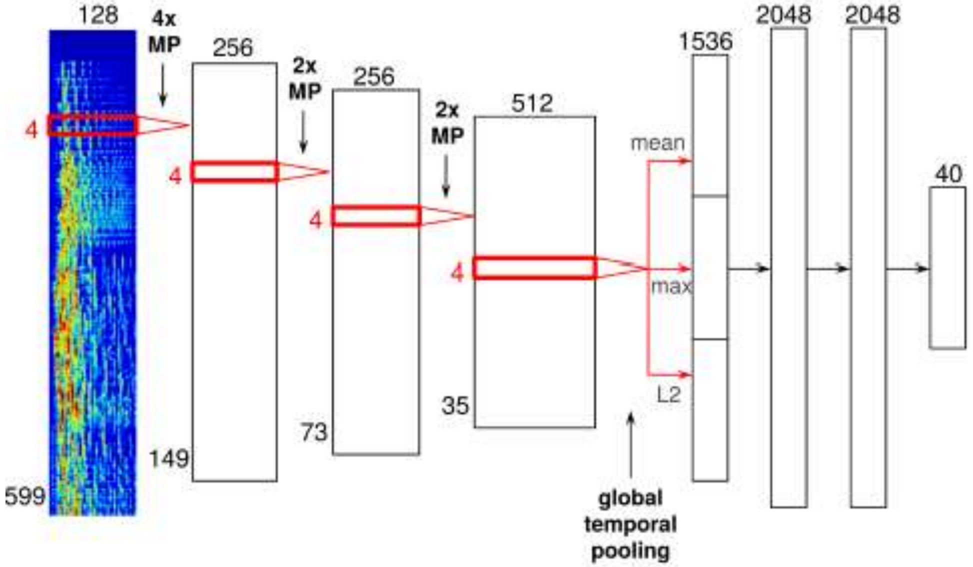
Ratings
4.5
4.0
3.5
5.0
?
5.0
3.5

```python
from keras.layers import Embedding, Dot, Flatten, Input
from keras.models import Model

user_input = Input((1, ))
item_input = Input((1, ))
user_embedding = Flatten()(
  Embedding(num_users, k, input_length=1)(user_input))
item_embedding = Flatten()(
  Embedding(num_items, k, input_length=1)(item_input))
loss = Dot(axes=1)([user_embedding, item_embedding])

model = Model([user_input, item_input], loss)
model.compile(loss="mse", optimizer="adam")
```

Model based Collaborative Filtering

- NIPS paper titled "Deep content-based music recommendation"

Example of using a neural network model to act as feature extractor for item content /metadata



Challenges

- Data Sparsity and Cold Start

One typical problem caused by the data sparsity is the cold start problem. As collaborative filtering methods recommend items based on users' past preferences, new users will need to rate sufficient number of items to enable the system to capture their preferences accurately and thus provides reliable recommendations.

- Scalability

As the numbers of users and items grow, traditional CF algorithms will suffer serious scalability problems. For example, with tens of millions of customers O(M) and millions of items O(N), a CF algorithm with the complexity of n is already too large

- Synonym

It refers to the tendency of a number of the same or very similar items to have different names or entries. Most recommender systems are unable to discover this latent association and thus treat these products differently.

Challenges

- Gray Sheep

Gray sheep refers to the users whose opinions do not consistently agree or disagree with any group of people and thus do not benefit from collaborative filtering.

- Shilling attacks

Everyone can give the ratings, people may give lots of positive ratings for their own items and negative ratings for their competitors. It is often necessary for the collaborative filtering systems to introduce precautions to discourage such kind of manipulations

- Diversity and the long tail

Because collaborative filters recommend products based on past sales or ratings, they cannot usually recommend products with limited historical data. This can create a rich-get-richer effect for popular products, akin to positive feedback. This bias toward popularity can prevent what are otherwise better consumer-product matches.

Citation & Useful links

- https://www.slideshare.net/databricks/deep-learning-for-recommender-systems-with-nick-pentreath
- https://github.com/maciejkula/spotlight/tree/master/examples/movielens_explicit
- https://recsys.acm.org/recsys18/tutorials/#content-tab-1-0-tab
- https://code.fb.com/core-data/recommending-items-to-more-than-a-billion-people/
- https://www.fast.ai/2017/07/28/deep-learning-part-two-launch/
- https://www.ethanrosenthal.com/2015/11/02/intro-to-collaborative-filtering/
- https://towardsdatascience.com/various-implementations-of-collaborative-filtering-100385c6dfe0
- https://www.youtube.com/watch?v=h9gpufJFF-0
- https://en.wikipedia.org/wiki/Collaborative_filtering