深度协同过滤

1. 协同过滤的思想

协同过滤的基本假设是，**有着相同兴趣的用户会喜欢相似的商品**；**有着相同受众的商品的属性是类似的。**

> Users with similar preferences tend to like the same items, and items with similar audiences tend to have the same features.

基于用户的协同过滤算法

当一个用户A需要个性化推荐时，可以先找到和他有相似兴趣的其他用户，然后把那些用户喜欢的、但是用户A没有听说过的商品召回出来，并预测user对这些item的评分，最终召回top k个商品。这里计算用户的"相似性"是根据用户购买商品的重合度来计算的。

其中，还有一个优化点就是，对于热门的商品可能大家都买过，这并不能说明两个用户是相似的；然而两个用户同时购买了冷门商品，这可能说明两者的兴趣是相似的。（例如，大家都买过《新华字典》，所以这不能说明两个人品味相似，但如果两个人都买过《数据挖掘》，这似乎说明两者品味相似。）所以，对热门的商品进行惩罚，让它不要过多影响用户相似度的得分计算。

$$w_{uv} = \frac{\sum_{i \in N(u) \cap N(v)} \frac{1}{\log 1 + |N(i)|}}{\sqrt{|N(u)||N(v)|}}$$

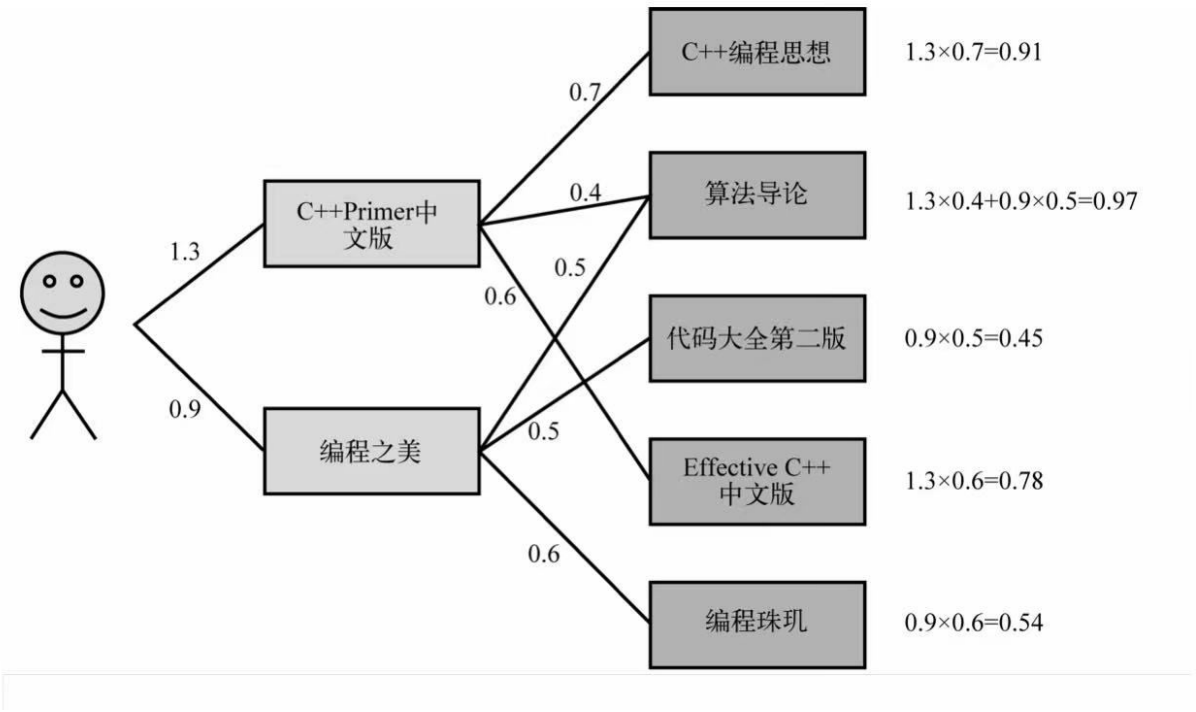
→ 减少热门物品对相似度的影响

基于物品的协同过滤算法

给用户推荐那些和它们之前喜欢的物品相似的物品，这里的相似度是基于物品的**受众**来说的。同理，在这里也需要惩罚活跃用户，即活跃用户对两个**物品相似度**的贡献应该小于不活跃用户：

$$w_{ij} = \frac{\sum_{u \in N(i) \cap N(j)} \frac{1}{\log 1 + |N(u)|}}{\sqrt{|N(i)||N(j)|}}$$

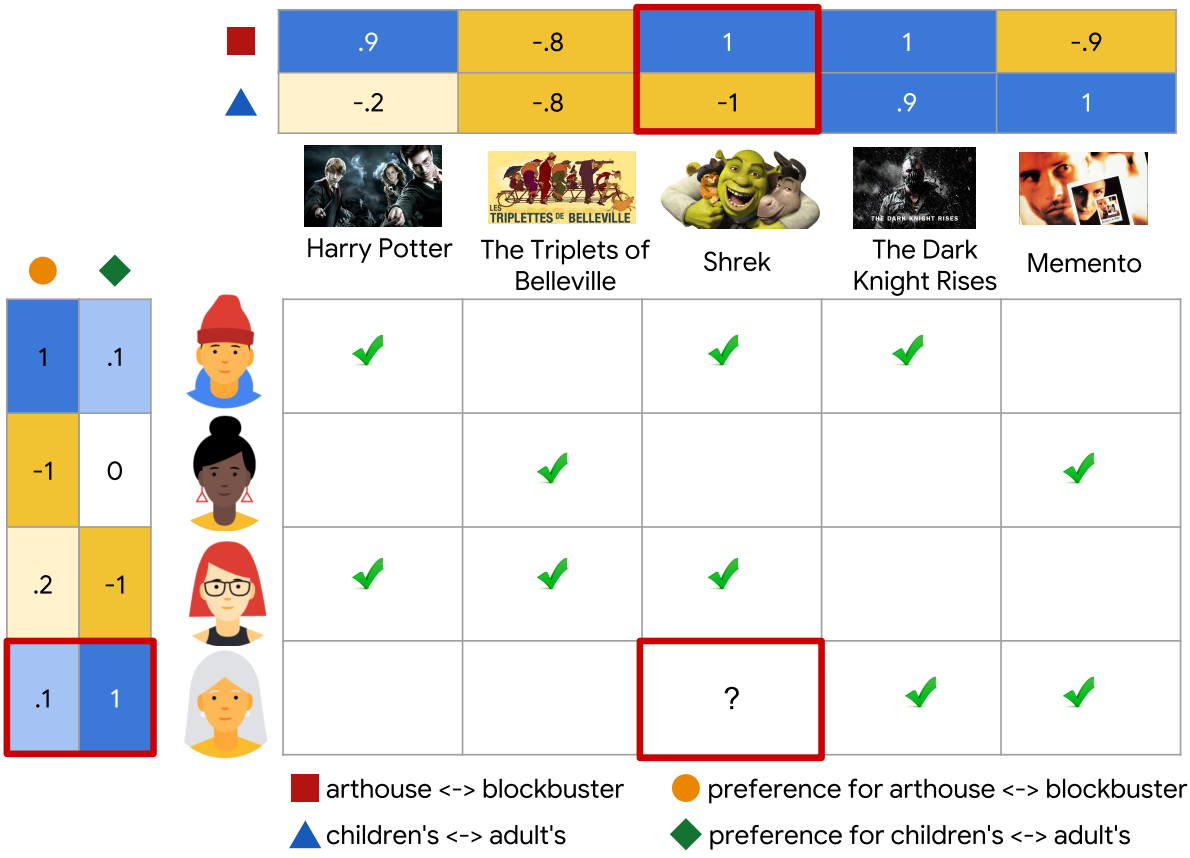
我们去召回这些相似的物品，然后预测user对他们的评分，最终召回top k个商品。基于物品的协同过滤的一个好处就是，它可以为推荐提供良好的可解释性。例如下图中，用户购买了《C++ primer》和《编程之美》，然后我们去计算和这两本书最接近的其他书，这样我们就可以告诉用户是基于他的什么历史行为做出的推荐。



基于模型的协同过滤

协同过滤的核心在于构建user-item交互矩阵，其中包含了隐式反馈信息和显式反馈信息：

- implicit data: 用户的点击/购买行为
- explicit data: user对item的评分矩阵



那么，怎么去获得这个user-item矩阵中中每个user和item的表示呢？其实，所有的方法可以分成两类：**基于降维的方法**(dimensionality reduction-based approaches)和**基于深度学习的方法**（NN-based approaches）。在本节中，我们先介绍基于降维的方法。

其中，基于降维的方法就是**矩阵分解**(matrix factorization)。基于矩阵分解的协同过滤把userID和itemID打成embedding，这样我们就得到了用户的喜好信息和商品的特征信息，然后让其试图恢复原先的user-item矩阵，即在未见的user-item交互中预测该处user-item矩阵的值（用点积来计算），其本质就是矩阵补全问题。如下图中，把user-item矩阵都分解为embedding size = 2的小矩阵，即为低秩分解，能够节约存储空间，同时对未知的交互能够进行预测。

> Dimensionality reduction-based approaches optimize a function that reduces the dimension of a graph matrix (user-item矩阵) and then produce low-dimensional embeddings.

我们要理解的一点是，user-item矩阵中为0的项并不代表用户对此商品不感兴趣 -- 有可能这个商品根本就没有曝光，我们不知道用户对它的兴趣（exposure bias）；同时，有交互的项也并不一定表示用户喜欢该商品。所以user-item矩阵是**有噪声的**。

2. Neural MF

我们可以用神经网络来**模拟矩阵分解**的过程，这样可以避免巨大的user-item矩阵直接分解。一个比较古老的方法是ALS（交替最小二乘法）。ALS方法的损失函数为平方损失函数，如下图所示。rui为user u对item i的评分，x_u是user u的embedding，y_i是item i的embedding，我们需要用user和item的点积来还原user-item评分矩阵。为了矩阵的稳定性，还加入了正则项。

$$\min_{x_*,y_*} L(X,Y) = \min_{x_*,y_*} \sum_{u,i \text{ is known}} (r_{ui} - x_u^T y_i)^2 + \lambda(|x_u|^2 + |y_i|^2)$$

ALS方法随机初始化Embedding，然后在以下条件之间交替进行：

- 固定 x_u 求解 y_i。
- 固定 y_i 求解 x_u 。

迭代至收敛之后，我们就得到了比较好的user和item embedding。

但是，ALS方法仅使用点积来衡量user和item的相似度，似乎表达能力不够。于是**Neural Collaborative Filtering** (WWW2017) 用神经网络来训练user/item的隐向量，然后通过MLP得到最终的得分，表达能力强于点积。

损失函数可以是point-wise loss（MSE或者log loss）；也可以是pair-wise loss（需要给正样本比负样本高的分数），还可以是带温度系数的sampled softmax loss.

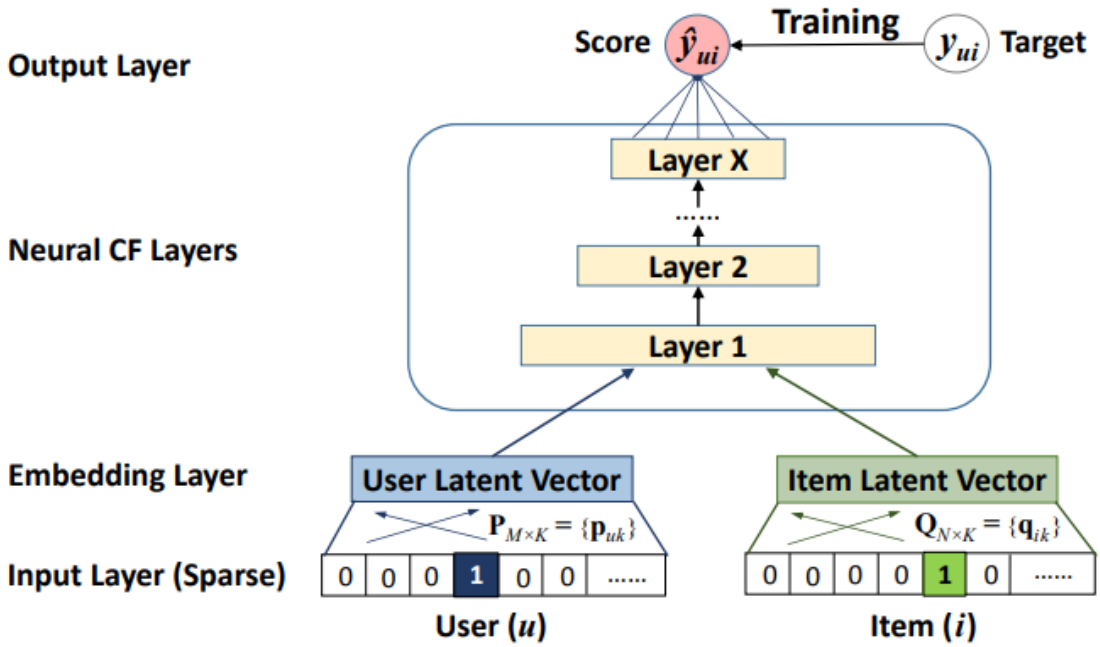
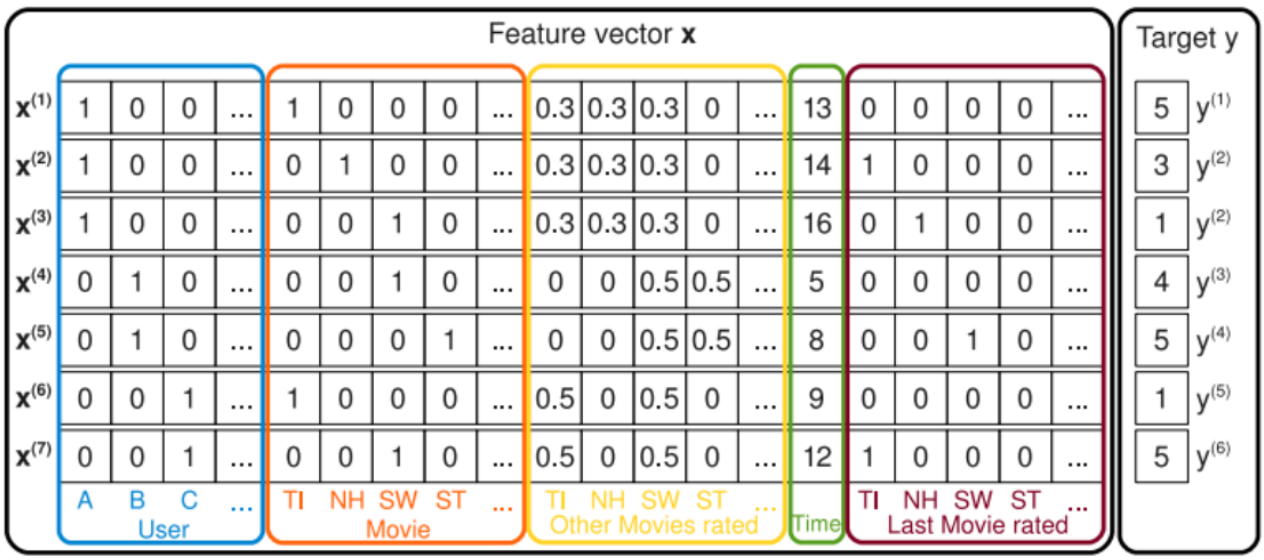


Figure 2: Neural collaborative filtering framework

得到一个好的user/item embedding之后，我们可以把物品的向量逐一离线算好，存入Faiss；每个user的向量离线算好，存入在线数据库中比如Redis。当用户登陆或者刷新页面时，可以根据用户ID取出其对应的embedding，然后和Faiss中存储的item embedding做高效检索，按照得分由高到低返回得分Top K的物料作为召回结果。

当然了，这样计算出来的user/item embedding只包含了user-item交互信息，没有显式的融合content信息。也就是说，矩阵分解只能得到user/item的ID embedding，而不能得到其他特征的embedding。为了能够得到更丰富的embedding信息，我们可以再融合经过**FM预训练**得到的特征embedding，然后把user/item embedding存索引。下图是FM的一个训练矩阵，可以看到不仅包含了user 和 movie的ID信息，还包含了很多的side information（其他特征），这样就相当于在知道一些side information的前提下去训练好的ID embedding和特征embedding。



或者，可以直接把特征信息(side information)加入NCF中进行**端到端**的训练，就是我们熟知的MLP模型。即训练每个特征的embedding table。这样就相当于做融合了side information的矩阵分解，而我们知道，加入side information可以解决【数据稀疏】和【冷启动问题】。

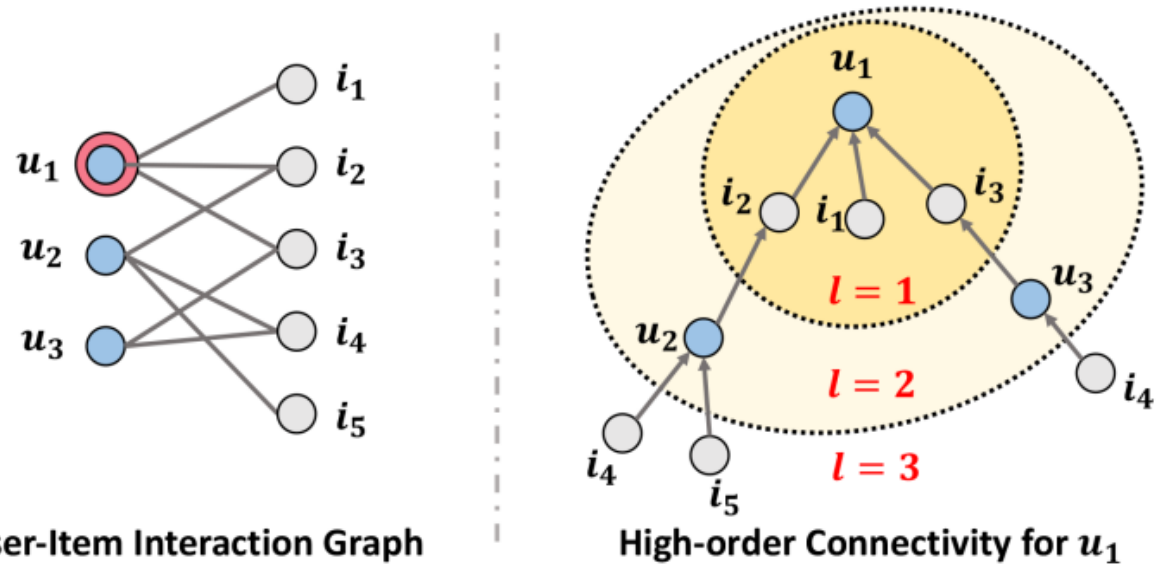
3. 图网络做协同过滤

使用图网络做协同过滤的好处有三个：

- 图网络中聚合邻居k次，即可得到k阶的关系，因此可以显式把握**高阶关系**信息
- 解决【冷启动问题】。一个新的user/item是没有交互信息的，但是我们可以根据content信息把它放在图中的合适位置，然后通过聚合其邻居得到好的表征。
- 缓解数据稀疏问题。原始的user-item评分矩阵中，长尾的user/item占了大多数，这些user/item并没有很多的交互数据（非热门商品、非活跃用户），所以user-item矩阵的大部分项都是0.而在图中，只有一个user对item有交互，两者之间才有边。这和用邻接矩阵表示稀疏图的优势是类似的。

3.1 Neural Graph Collaborative Filtering (SIGIR 2019)：显式把握高阶关系信息

基于矩阵分解的协同过滤得到user/item 的 ID embedding，不免会遇到**数据稀疏**问题。同时，矩阵分解的协同过滤**只能把握一阶的user-item关系，不能得到高阶的关系。**何谓"高阶的连接"？下图中i4->u2->i2->u1就是一个高阶的关系。user2喜欢item2，同时user1也喜欢item2，这就把user1和user2显式的连接起来了。



User-Item Interaction Graph　　High-order Connectivity for u_1

为了得到**显式的高阶的连接性(high-order connectivity)**, 我们把user-item矩阵用一个**二部图**来表示。让图**聚合**k次邻居节点的信息，就能够把握住k阶的连接性。如上图中，聚合两次就可以显式把握住u1->i2->u2的联系，聚合三次就可以显式把握住u1->i2->u2->i4的联系。

具体的做法是，首先将经过MF得到的user/item ID embedding作为每个节点的**初始化**embedding；然后，在图上对邻居（user的邻居是item，item的邻居是user）进行**聚合**以把握住高阶的连接关系，以此来对MF得到的初始embedding做更精细的调整(refine). 下面，我们来详细介绍图中是如何聚合的。

1）信息传递

对于相连的user-item对(u,i), i传到u的信息计算公式：

$$\mathbf{m}_{u\leftarrow i} = \frac{1}{\sqrt{|\mathcal{N}_u||\mathcal{N}_i|}}\left(\mathbf{W}_1\mathbf{e}_i + \mathbf{W}_2(\mathbf{e}_i \odot \mathbf{e}_u)\right), \quad (3)$$

和普通的GCN邻居聚合方法不同的是，这里还增加了item embedding和邻居user embedding的哈达玛积，引入了交互信息。那么，第一次聚合之后，user的表示为：

$$\mathbf{e}_u^{(1)} = \text{LeakyReLU}\left(\mathbf{m}_{u\leftarrow u} + \sum_{i\in\mathcal{N}_u} \mathbf{m}_{u\leftarrow i}\right),$$

即为user本身，传到user的信息+所有邻居item传到user的信息。经过L次如此的聚合，我们就可以得到融合了L阶关系的user/item embedding。这L个不同阶的embedding我们可以将他们进行拼接/加权/过LSTM等方式融合，得到最终的user/item embedding。这个embedding比起简单的用矩阵分解方式得到的embedding就融合了不同阶的user-item交互信息，因此包含的信息更为丰富了。

这样，我们就得到了user/item embedding，二者求点积即为预测的得分（相似度）。训练时损失函数选用pair-wise的Bayesian personalized ranking (BPR) loss，即：

$$Loss = \sum_{(u,i,j)\in O} -\ln\sigma(\hat{y}_{ui} - \hat{y}_{uj}) + \lambda\|\Theta\|_2^2,$$

其中，i为正样本，j为负样本。

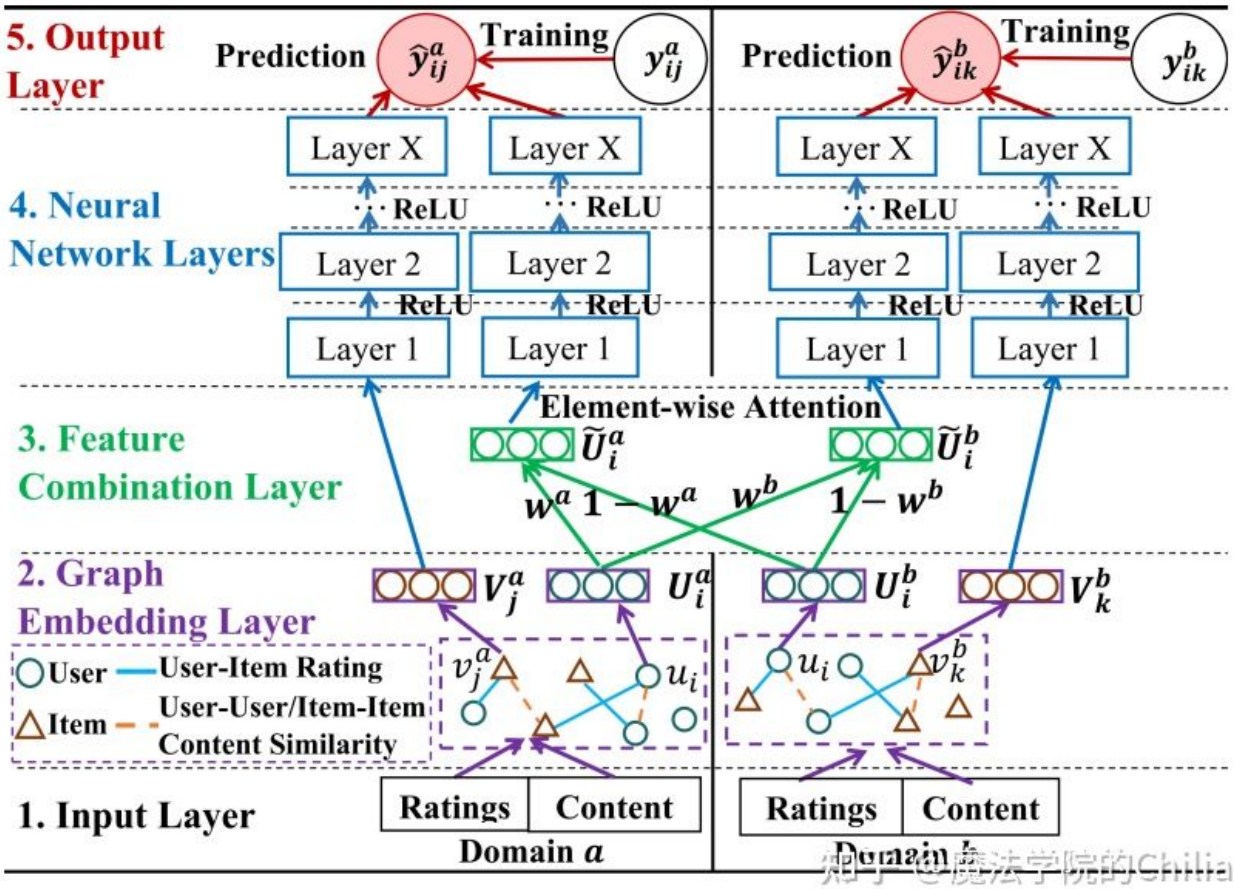
3.2 异构图网络做协同过滤：把握user-user和item-item的内容相似度信息

一个user/item的表征可以看成由两部分组成：内容本身(content-based)的表征，和在user-item图中的关系表征。上述讲的Graph MF方法只是利用了user-item矩阵，但是并没有考虑到user-user和item-item**基于内容的**相似度。

因此，我们除了根据评分矩阵建立user-item图（user-item边上的权重为归一化的评分），同时还可以在图上根据user-user和item-item的内容相似度构建user-user和item-item之间的边（边权重为归一化的相似度）。这里节点的初始化embedding可以是经过MF得到的embedding与user/item内容信息的某种组合。之后，我们使用一些graph embedding方法（DeepWalk，GraphSAGE等），得到user/item的embedding。

例如之前将跨域推荐的时候提到的： Graphical and Attentional Framework for Dual-Target Cross-Domain Recommendation [ijcai, 2020] 的第二层 graph embedding layer就是用这样的异构图来得到好的user/item embedding。

这样做的另一个好处是可以一定程度上**解决冷启动问题**。对于一个新来的item，我们虽然不知道它和user的交互信息是什么样子，但是我们可以知道它的content信息，这样就能够算出来它和其他item的相似度，就可以连接起它和图中的其他item。这样，通过GraphSAGE做几次聚合之后，我们就能够得到一个比较好的初始化embedding。


5. Output Layer
Prediction
Training
4. Neural Network Layers
Layer X
ReLU
Layer 2
Layer 1
3. Feature Combination Layer
Element-wise Attention
2. Graph Embedding Layer
User
Item
User-Item Rating
User-User/Item-Item Content Similarity
1. Input Layer
Ratings
Content
Domain a
Domain b