

推荐系统 概述

recommendation system

张娇昱
2018.09.07

目录

Contents

- 一、推荐系统概述**
 - 1. 基本概念
 - 2. 算法分类
 - 3. 存在问题
- 二、协同过滤算法 (CF)**
 - 1. Memory-Based CF
 - 2. Model-based CF
- 三、深度学习在推荐算法中的应用**
- 四、问题与解决**
 - 1. 冷启动

推荐系统定义：

推荐系统，是利用收集的用户对项目的偏好有好有坏、用户特征、项目特征等内容，用特定的知识表示方式进行处理存储；再利用机器学习或者统计学习的相关算法分析已有数据，针对特定的用户需求偏好为其推荐相应项目，帮助用户做出决策的智能系统。

数据输入类型：

- **显示反馈** (explicit feedback) : 用户直接反映其对产品的喜好信息，如评分、喜欢/不喜欢等等。
- **隐式反馈** (implicit feedback) : 间接反映出用户对于产品的喜好，如：购买历史、浏览记录、搜索记录等。

对于隐式反馈，可以将其表示为下图：

	i_1	i_2	i_3	i_4
u_1	?	+	+	?
u_2	+	?	?	+
u_3	+	+	?	?
u_4	?	?	+	+
u_5	?	?	+	?



	i_1	i_2	i_3	i_4
u_1	0	1	1	0
u_2	1	0	0	1
u_3	1	1	0	0
u_4	0	0	1	1
u_5	0	0	1	0

算法分类：

- 根据输出结果，推荐系统可以分为三种类型：评分预测、排名预测（top-n推荐）和分类。
- 根据输入数据类型的不同，推荐系统可以分为：**协同过滤**，基于内容的推荐系统，基于不同输入数据类型的混合推荐系统。

算法分类：

➤ 基于内容的推荐系统（Content-based Recommendation）：

主要根据用户已经选择或者评分的项目，挖掘其他内容上相似的项目作为推荐。

首先通过显式反馈（例如评分、喜欢/不喜欢）或隐式反馈（例如观看、搜索、点击、购买等行为）的方式获取用户交互过的项目，然后从这些项目的特征中学习用户的偏好并用特征表示，并计算用户与待预测项目在内容（由项目特征刻画）上的匹配度（或相似度），最后根据匹配度所有待预测项目进行排序，从而为用户推荐潜在感兴趣的项目。

问题：特征提取困难

算法分类：

➤ 基于协同过滤的推荐系统(Collaborative Filtering-based Recommendation)：

源于现实生活中口碑相传 (word-of-mouth) 的过程，协同过滤利用相似用户之间具有相似兴趣偏好的方法，来发现用户对项目的潜在偏好。

协同过滤仅仅需要利用用户的历史评分数据，因此简单有效，是目前应用最为成功的推荐方法。

问题：1.数据稀疏 2.冷启动 3.可拓展性

算法分类：

➤ 混合推荐：

常见的组合策略主要包括三种，分别为：后融合、中融合和前融合。

后融合是指将两种或两种以上的推荐算法产生的推荐结果，以投票机制、线性组合或者可信度选择组合等方式来产生最终的推荐结果，本质上是决策层面上的混合。

中融合的基本框架是以一种推荐算法为基础，同时融合另一种推荐算法，例如：以协同过滤算法为框架，融入基于内容的推荐算法可以有效缓解数据稀疏问题，本质上是模型层面上的混合。

前融合则是直接将多种推荐算法融合到统一的模型中，然后从各类数据中提取特征作为模型的输入，由统一的模型产生推荐结果，例如，将所有用户属性、用户行为等数据作为输入，通过训练一个统一的分类器产生推荐结果，本质上是特征层面的融合。

协同过滤算法 CF

评价矩阵：

CF需要的输入为评价矩阵，用户（user）对项目（item）进行打分，如下图的例子：

	Movie1	...	Movie5	...	Movie32	...	
User1	0	...	0	...	4	...	
User2	5	...	3	...	3	...	
User3	4	...	3	...	0	...	
User4	0	...	0	...	5	...	
	
User667	0	...	0	...	4.5	...	
User668	3	...	2.5	...	3	...	

协同过滤算法 CF

基于内存的方法 (Memory-based CF) : 最近邻模型 (Nearest Neighbor Model)

基于用户的推荐 (User-based CF)

基于项目的推荐 (Item-based CF)

基于模型的方法 (Model-based CF) : 隐语义模型 (Latent factor model)

Memory-based CF

➤ 基于用户的协同过滤推荐(**User-based Collaborative Filtering Recommendation**) :

使用统计技术寻找与目标用户有相同喜好的邻居，然后根据目标用户的邻居的喜好产生向目标用户的推荐。基本原理就是利用用户访问行为的相似性来互相推荐用户可能感兴趣的资源，它的基本假设是，喜欢类似物品的用户可能有相同或者相似的口味和偏好。

➤ 基于项目的协同过滤推荐(**Item-based Collaborative Filtering Recommendation**) :

根据所有用户对物品或者信息的评价，发现物品和物品之间的相似度，然后根据用户的历史偏好信息将类似的物品推荐给该用户。基于物品本身的属性特征信息进行推荐。

Memory-based CF

➤ 相似度计算：

➤ 欧几里德距离 (Euclidean Distance)

$$d(x, y) = \sqrt{(\sum (x_i - y_i)^2)}$$

$$sim(x, y) = \frac{1}{1 + d(x, y)}$$

➤ 皮尔逊相关系数 (Pearson Correlation Coefficient)

$$p(x, y) = \frac{\sum x_i y_i - n \bar{x} \bar{y}}{(n-1)s_x s_y} = \frac{n \sum x_i y_i - \sum x_i \sum y_i}{\sqrt{n \sum x_i^2 - (\sum x_i)^2} \sqrt{n \sum y_i^2 - (\sum y_i)^2}}$$

➤ Cosine 相似度 (Cosine Similarity)

$$T(x, y) = \frac{x \bullet y}{\|x\|^2 + \|y\|^2 - x \bullet y} = \frac{\sum x_i y_i}{\sqrt{\sum x_i^2} + \sqrt{\sum y_i^2} - \sum x_i y_i}$$

UBCF – 实现

➤ 计算用户相似度矩阵

将每个用户表示
为N维向量

A	a	b	d
B	a	c	
C	b	e	
D	c	d	e

创建物品-用户
倒排表

a	A	B
b	A	C
c	B	D
d	A	D

建立用户相似度
矩阵

	A	B	C	D
A	0	1	1	1
B	1	0	0	1
C	1	0	0	1
D	1	1	1	0

将每个用户表示
为N维向量

A	0	$\frac{1}{\sqrt{3 \times 2}}$	$\frac{1}{\sqrt{3 \times 2}}$	$\frac{1}{\sqrt{3 \times 3}}$
B	$\frac{1}{\sqrt{3 \times 2}}$	0	0	$\frac{1}{\sqrt{3 \times 2}}$
C	$\frac{1}{\sqrt{3 \times 2}}$	0	0	$\frac{1}{\sqrt{3 \times 2}}$
D	$\frac{1}{\sqrt{3 \times 3}}$	$\frac{1}{\sqrt{3 \times 2}}$	$\frac{1}{\sqrt{3 \times 2}}$	0

UBCF – 实现

- 产生推荐

$$p(u, i) = \sum_{S(u, K) \cap N(i)} W_{uv} * R_{vi}$$

- $S(u, K)$ 和用户u最相似的K个用户

Memory-based CF

算法	原理	区别	使用场合
User-Based CF	推荐与目标用户相似的用户喜欢的项目	<ol style="list-style-type: none">采用用户相似度矩阵，适用用户较少的场合只推荐热门项目，长尾能力不足冷启动问题严重稀疏矩阵问题严重	新闻类网站 ，个性化要求不强，热门程度和实效性是推荐的重点。
Item-based CF	推荐与目标用户之前喜欢的物品相似的物品	<ol style="list-style-type: none">采用项目相似度矩阵，适用项目较少的场合，维护成本相对较高长尾丰富，但多样性欠缺对于新用户，无冷启动问题稀疏矩阵问题严重	电商、音乐、图书类网站 ，用户的兴趣喜好相对稳定，个性化要求较高，同时对实效性要求不高

假设目标用户的喜好分为3个领域A、B、C，其中A是他主要喜欢的领域，则UserCF会将A、B、C三个领域中比较热门的项目推荐给目标用户；ItemCF只推荐A领域的项目给目标用户，但不一定都是项目

Memory-based CF

➤ 实验结果

数据集：

MovieLens

ML-latest-small

100,000 ratings and 1,300

tag applications applied to

9,000 movies by 700 users

```
F:\python\Python35\python.exe F:/py_cv_group/reco_Mo/classical_CF/userCF.py
the number of similar users = 20
the number of recommended movies = 10
Load success : F:/py_cv_group/dataset/ml-latest-small/ratings.csv
Split sets success!
TrainSet : 74812 TestSet : 25192
Build movie-user table success!
Build similar user matrix success!
Calculate user matrix success!
eval result:
precision=0.2793    recall=0.0744    coverage=0.0473
time for userCF : 4.03597354888916
```

```
F:\python\Python35\python.exe F:/py_cv_group/reco_Mo/classical_CF/itemCF.py
the number of similar movies = 20
the number of recommended movies = 10
Load success : F:/py_cv_group/dataset/ml-latest-small/ratings.csv
Split sets success!
TrainSet : 75063 TestSet : 24941
Build movie-popular list success!
Build similar movie matrix success!
Calculate movie matrix success!
eval result:
precision=0.2607    recall=0.0701    coverage=0.0815
time for itemCF : 157.45418214797974
```

Model-based CF

基于模型的协同过滤算法最初是为了解决Memory-based CF的稀疏矩阵和可拓展性差的问题。它的基本思想是，采用机器学习技术，建立相关的学习模型，利用评价矩阵中的信息进行训练得到模型参数，最后利用该模型对推荐结果进行预测。

User /item	1	2	3	4	5
1	5	4	4.5	?	3.9
2	?	4.5	?	4.5	?
3	4.5	?	4.4	4	4
4	?	4.8	?	?	4.5
5	4	?	4.5	5	?
.....

矩阵分解模型（Matrix factorization，MF）是其中广泛被使用的模型。其中，又详细有SVD（奇异值分解）、PMF（概率矩阵分解）等。

SVD 奇异值分解

线性代数 基础

1. 特征向量与特征值：

$$Av = \lambda v$$

2. 特征值分解：

$$A = Q\Sigma Q^{-1}$$

① 特征分解：设 A 为方阵， λ 为特征值， \vec{x} 为特征向量
有 $A\vec{x}_1 = \lambda_1 \vec{x}_1$
 $A\vec{x}_2 = \lambda_2 \vec{x}_2$
 \vdots
 $A\vec{x}_n = \lambda_n \vec{x}_n$

$$\Rightarrow A[\vec{x}_1 \vec{x}_2 \cdots \vec{x}_n] = [A\vec{x}_1 A\vec{x}_2 \cdots A\vec{x}_n] = [\lambda_1 \vec{x}_1 \lambda_2 \vec{x}_2 \cdots \lambda_n \vec{x}_n]$$

$$\Leftrightarrow A = [Q \Sigma Q^{-1}] = [Q \Lambda Q^{-1}]$$

$$A = Q \Lambda Q^{-1}$$

又 Q 为正交矩阵，有逆矩阵

$$Q^{-1} = Q^T$$

$$\Rightarrow A = Q \Lambda Q^T$$

$$\Rightarrow A = Q \Lambda Q^T$$

SVD 奇异值分解

特征值分解基于方阵，但现实世界的矩阵通常M*N

因此，描绘普通矩阵时采用奇异值分解：

$$A = U\Sigma V^T$$

奇异值分解与特征值分解的对应：

$$(A^T A)v_i = \lambda_i v_i$$

由于在矩阵Σ中也是按从大到小的方式排列， σ 的值减小的特别的快，近似的有：

$$\sigma_i = \sqrt{\lambda_i}$$

$$u_i = \frac{1}{\sigma_i} A v_i$$

$$A_{m \times n} \approx U_{m \times r} \Sigma_{r \times r} V^T_{r \times n}$$

SVD

➤ Basic SVD原理

矩阵分解模型中，将评价矩阵U分解为两个矩阵相乘。

$$U = \begin{bmatrix} u_{11} & \cdots & u_{1k} \\ \vdots & \ddots & \vdots \\ u_{m1} & \cdots & u_{mk} \end{bmatrix} \begin{bmatrix} i_{11} & \cdots & i_{1n} \\ \vdots & \ddots & \vdots \\ i_{k1} & \cdots & i_{kn} \end{bmatrix}$$

即有，

$$\hat{r}_{ui} = q_i^T p_u$$

在这样的分解模型中,P_u代表用户隐因子矩阵（表示用户u对因子k的喜爱程度),Q_i表示电影隐因子矩阵（表示电影i在因子k上的程度）

SVD

➤ Basic SVD原理

则有预测值与真实值的误差：

$$e_{ui} = r_{ui} - \hat{r}_{ui}$$

继而可以获得误差平方和，再通过梯度下降拟合出最优模型。

$$\text{SSE} = \sum_{u,i} e_{ui}^2 = \sum_{u,i} \left(r_{ui} - \sum_{k=1}^K p_{uk} q_{ki} \right)^2$$

Baseline Predictor

评分值大部分情况下会与项目或用户本身的某些因素有关，造成一些干扰。例如：一些用户会倾向于给出较高的分数，一些项目平均分会高于其他项目。我们可以将其称为偏差（ bias ）：

$$b_{ui} = \mu + b_i + b_u$$

通过优化一个目标函数（如最小二乘法）来估计 b_u 和 b_i 的值：

$$\min_{b_*} \sum_{(u,i) \in \mathcal{K}} (r_{ui} - \mu - b_u - b_i)^2 + \lambda_1 (\sum_u b_u^2 + \sum_i b_i^2)$$

需要注意的是，基线预测因子（ Baseline Predictor ）本身是不能作为推荐系统模型，因为它只是描述了数据集的一部分特征，没有考虑到项目与用户之间的交互。

SVD

➤ BiasSVD原理

BiasSVD就是一种加入Baseline Predictors优化的matrix factorization model。

SVD公式如下：

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

加入防止过拟合的 λ 参数，可以得到下面的优化函数：

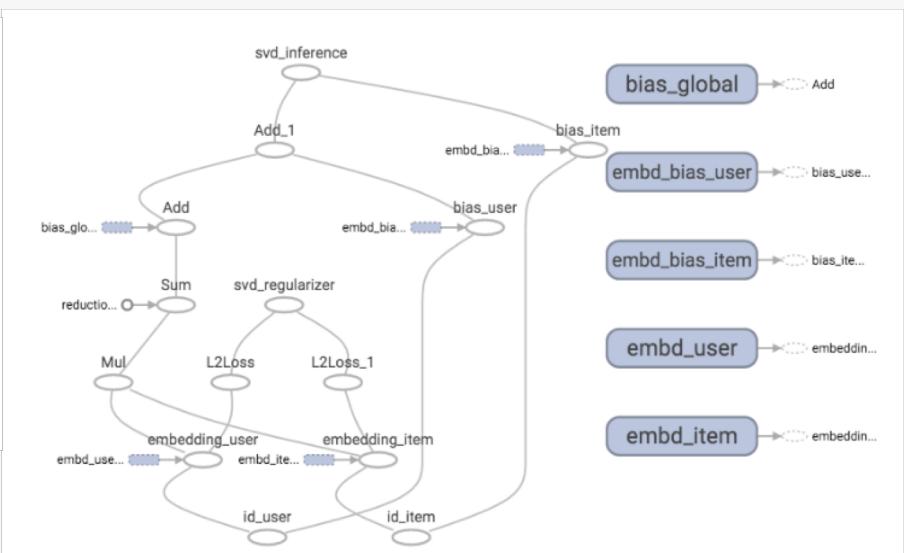
$$\min \sum (r_{ui} - \mu + b_i + b_u + q_i^T p_u)^2 + \lambda(b_i^2 + b_u^2 + \|q_i\|^2 + \|p_u\|^2)$$

SVD

➤ Tensorflow实现的BiasSVD

```
bias_global = tf.get_variable("bias_global", shape=[])
w_bias_user = tf.get_variable("embd_bias_user", shape=[user_num])
w_bias_item = tf.get_variable("embd_bias_item", shape=[item_num])
bias_user = tf.nn.embedding_lookup(w_bias_user, user_batch, name="bias_user")
bias_item = tf.nn.embedding_lookup(w_bias_item, item_batch, name="bias_item")
w_user = tf.get_variable("embd_user", shape=[user_num, dim],
                         initializer=tf.truncated_normal_initializer(stddev=0.02))
w_item = tf.get_variable("embd_item", shape=[item_num, dim],
                         initializer=tf.truncated_normal_initializer(stddev=0.02))
embd_user = tf.nn.embedding_lookup(w_user, user_batch, name="embedding_user")
embd_item = tf.nn.embedding_lookup(w_item, item_batch, name="embedding_item")
```

```
infer = tf.reduce_sum(tf.multiply(embd_user, embd_item), 1)
infer = tf.add(infer, bias_global)
infer = tf.add(infer, bias_user)
infer = tf.add(infer, bias_item, name="svd_inference")
regularizer = tf.add(tf.nn.l2_loss(embd_user), tf.nn.l2_loss(embd_item), name="svd_regularizer")
```



SVD

➤ SVD++原理

引入隐式反馈，使用用户的历史浏览数据、用户历史评分数据、电影的历史浏览数据、电影的历史评分数据等作为新的参数。

$$\hat{r}_{ui} = \mu + b_i + b_u + q_i^T(p_u + \frac{1}{\sqrt{\|R_u\|}} \sum_{j \in R_u} y_j)$$

隐式反馈的原因比较复杂，每个item对应一个向量 y_i ，通过user隐式反馈过的item的集合来刻画用户的偏好。 $N(u)$ 是用户u的行为记录（包括浏览的和评过分的商品集合），是收缩因子取集合大小的根号是一个经验公式，并没有理论依据。

Bayesian Personalized Ranking (BPR)用于优化隐式反馈。

深度学习推荐模型

- 输入层的数据主要包括：用户的显式反馈（评分、喜欢\不喜欢）或隐式反馈数据（浏览、点击等行为数据）、用户画像（性别、年龄、喜好等）和项目内容（文本、图像等描述或内容）数据、用户生成内容（社会化关系、标注、评论等辅助数据）。
- 模型层，使用的深度学习模型比较广泛，包括自编码器、受限玻尔兹曼机、卷积神经网络、循环神经网络等。
- 输出层，通过利用学习到的用户和项目隐表示，通过内积、Softmax、相似度计算等方法产生项目的推荐列表。

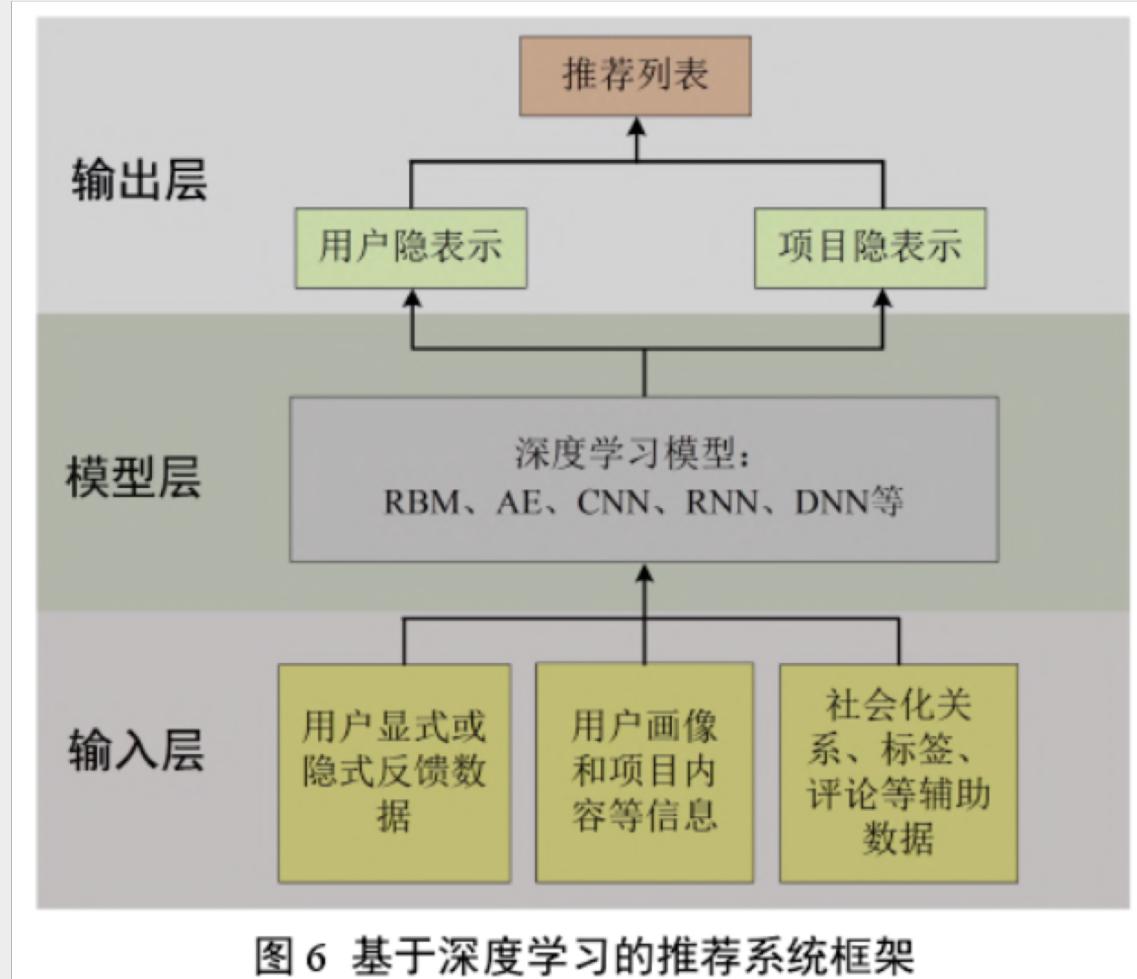


图 6 基于深度学习的推荐系统框架

深度学习推荐算法

虽然深度学习应用于推荐系统的形式多种多样，但一般来说，深度学习的作用主要为以下三种：

- **学习潜在特征向量**，例如用自动编码器（AE）来学习潜在因子特征向量，提高模型性能
- **从多媒体内容中提取特征**，传统的协同过滤推荐系统只是基于项目与用户之间的交互信息来建模，不借助外来的信息，而我们可以用深度网络提取多媒体内容（例如图片、视频等）的特征，然后送入模型中一起训练；
- **深度协同过滤**，利用深度网络来学习用户与项目之间复杂的非线性交互，提高模型的表达能力和性能。

YouTube 深度学习推荐算法

Deep Neural Networks for YouTube Recommendations (2016)

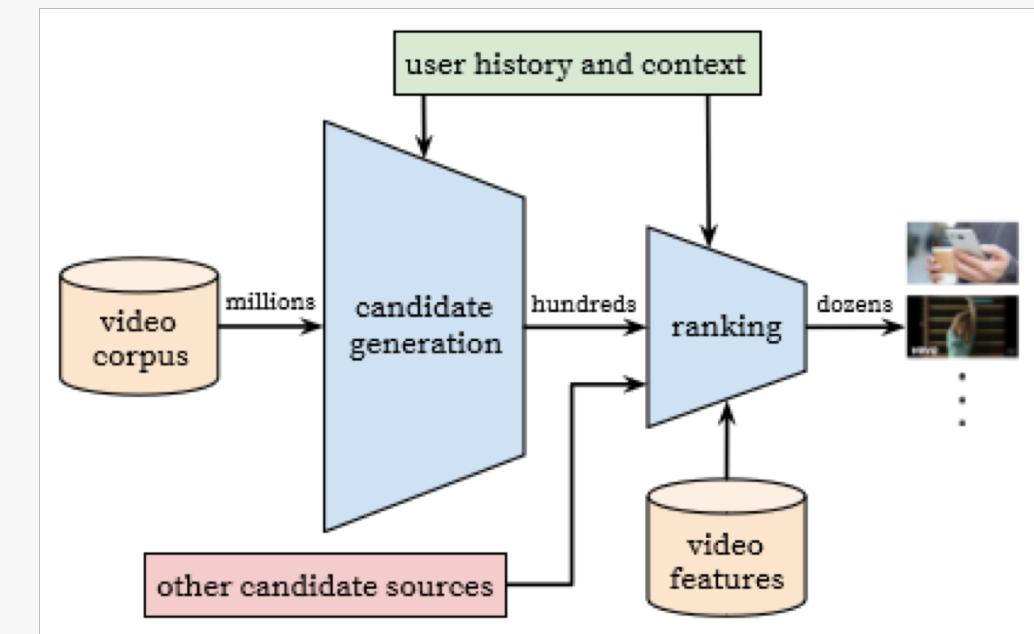
采用两个神经网络

1. Candidate generation:

采用CF , classification

2. Ranking :

logistic回归



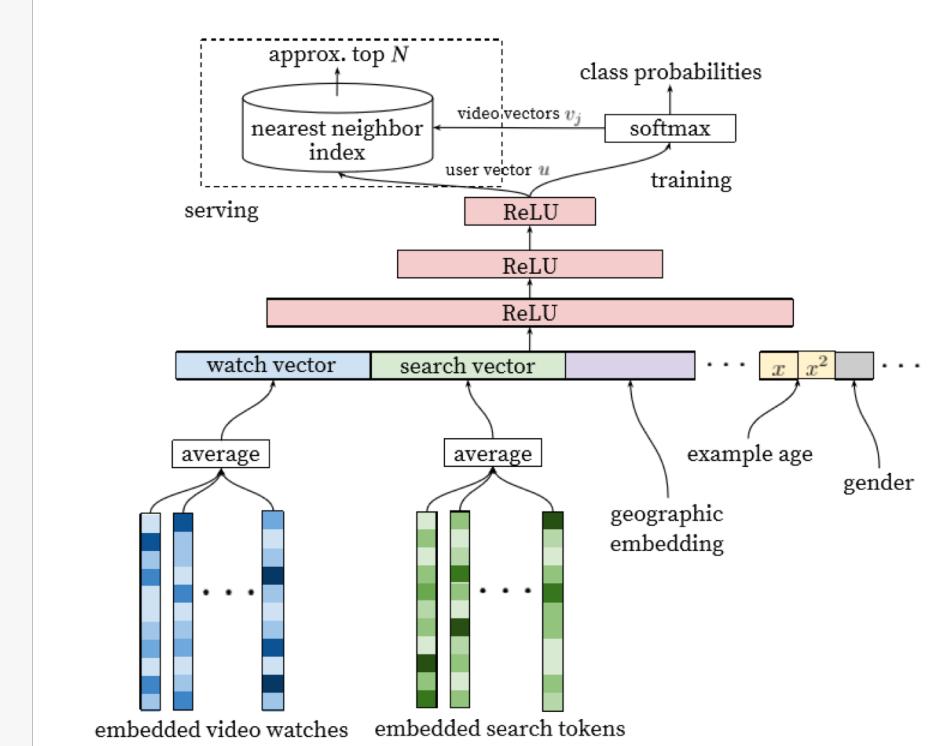
YouTube 深度学习推荐算法

Candidate generation

$$P(w_t = i|U, C) = \frac{e^{v_i u}}{\sum_{j \in V} e^{v_j u}}$$

u = 表示用户的高维embedding

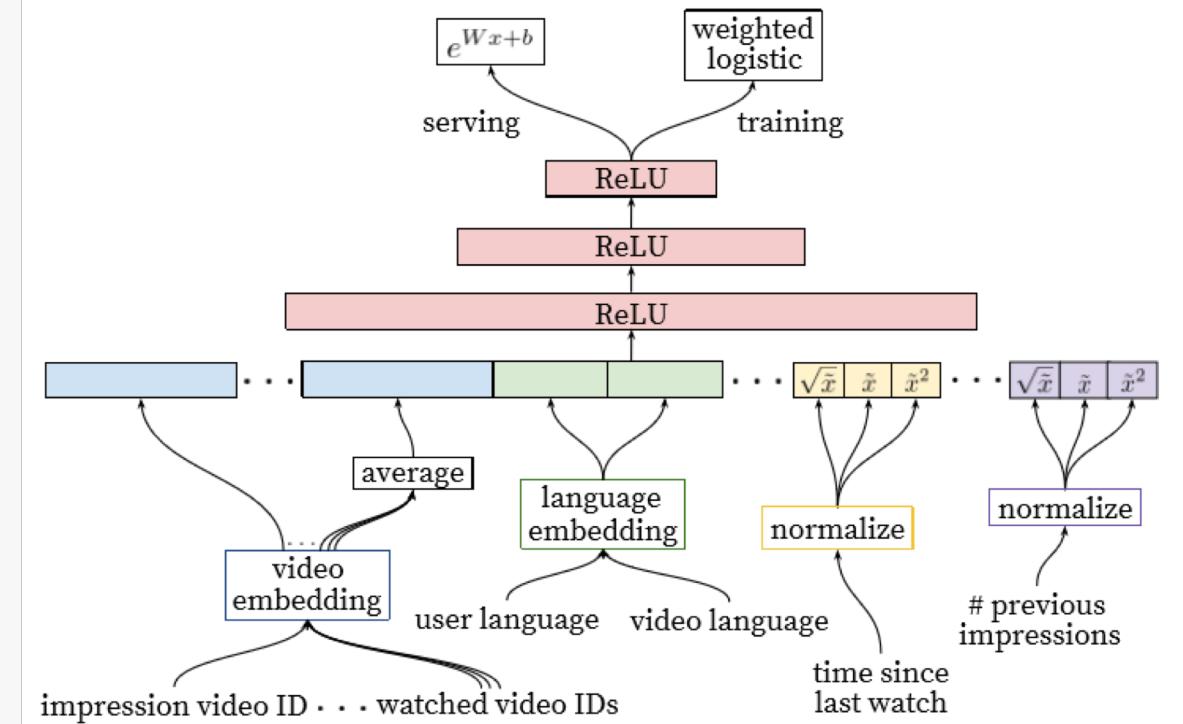
v_i = 表示视频的高维embedding



YouTube 深度学习推荐算法

Ranking

logistic 回归



其他案例

1. Facebook的推荐算法， CF算法， 分布式优化

<https://code.fb.com/core-data/recommending-items-to-more-than-a-billion-people/>

2. Collaborative Topic Regression<CTR>

冷启动

➤ 问题描述：

冷启动是协同过滤推荐算法的一个重点难题。协同过滤推荐系统根据用户的基本信息与历史行为信息，分析出用户的行为偏好和项目的特征属性，因此缺少相关用户与项目的交互信息，就无法给出准确的推荐。冷启动问题分为：

1. **新用户冷启动**：新用户个人信息与行为信息不完善，系统无法分析出新用户的偏好，无法有效推荐。
2. **新项目冷启动**：新项目没有被用户评价的信息，导致评分数据的稀少，系统不能建立更详细的特征信息表，也就不能将其与用户的喜好进行很好的匹配，因此新物品被推荐的可能性极低。
3. **新系统冷启动**：即所有的用户、项目对于系统来说都是新的，一个全新的系统无法产生相应的推荐

冷启动的解决策略

- 数值类推荐：根据数值方法如随机数、均值、众数、信息熵等产生推荐，再利用用户反馈更新信息。推荐结果有一定可靠性，但是个性化低，项目覆盖率低。
- 混合类推荐：混合使用多种算法，基于内容推荐与协同过滤推荐相结合，能够产生比较可靠的结果，很大程度上缓解了冷启动和稀疏矩阵问题。
- 融合多数据源推荐：引入用户的社交关系信息、评分、评论、位置信息等，用于解决新用户冷启动。

冷启动的解决策略

➤ 针对新用户冷启动：

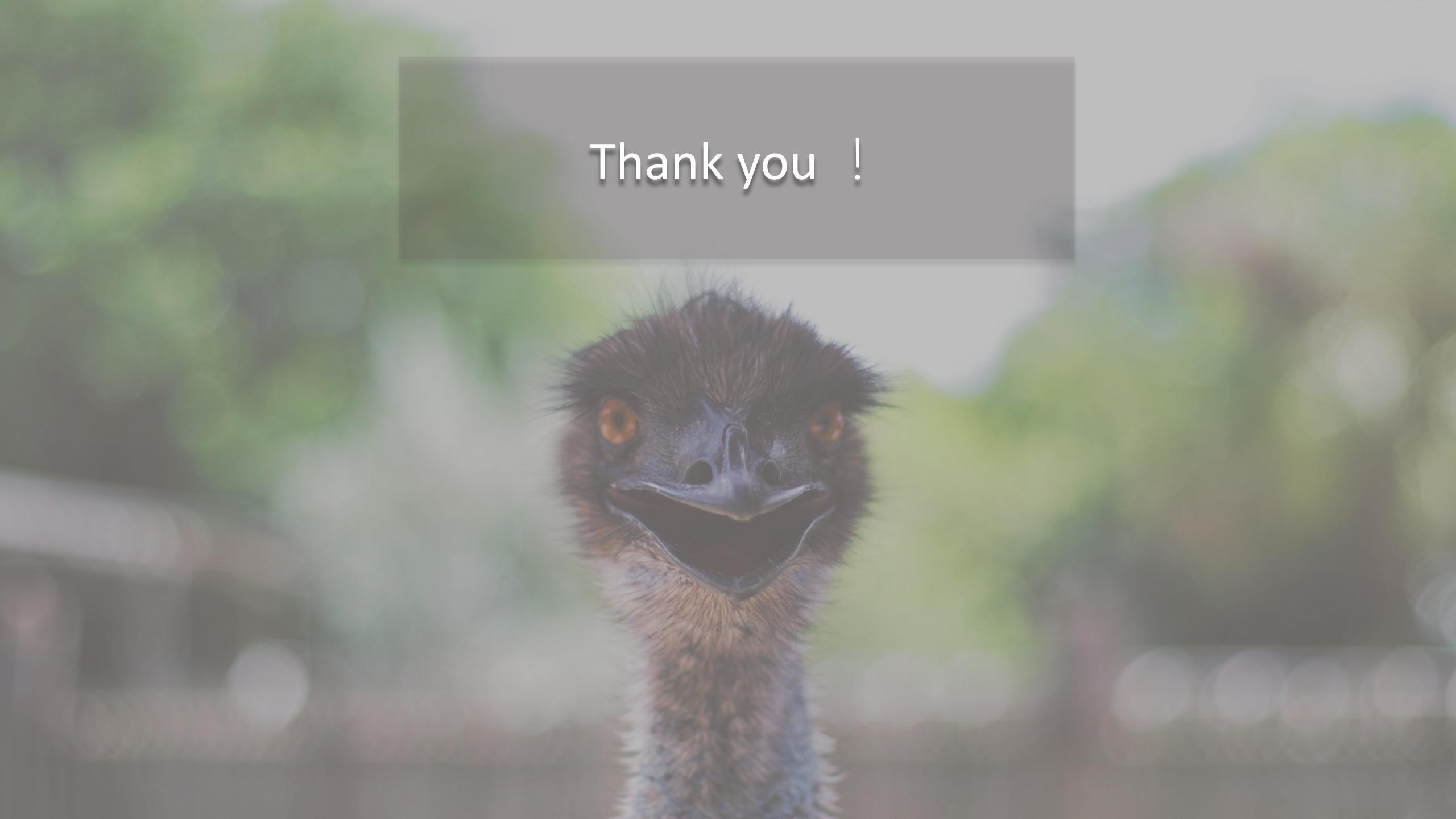
- 一) 提供个性化的推荐：比如热门排行榜。等用户行为数据收集到一定时候再切换为个性化推荐。
- 二) 利用注册信息，比如提供的年龄，性别等做粗粒化的推荐。
- 三) 要求用户在注册时对一些物品进行反馈。收集用户兴趣信息。

➤ 针对新项目冷启动：

对新加的物品，利用内容信息，将他们推荐给喜欢过和他们内容类似的用户

➤ 针对新系统冷启动：

引入专家知识，通过一定高效方式迅速建立物品的相关度表。



Thank you !