Business objective & Quantified metrics

- increase user engagement: 多用某个app，和app里的信息互动
  - Maximize the number of user clicks：clickbait
  - Maximize the number of completed videos：shorter videos that are quicker to watch.
  - Maximize total watch time
  - --- relevance score:
- increase sales/listening time： 针对某种服务
  - 服务相关 metrics
- increase user satisfication/retention/adherence: 多用app提供的功能
  - Implicit reaction: dwell time, user clicks
    - Pro: large amount; Cons: not accurate, click not means like
  - Explicit reaction (interaction): like, comment, share, (Negative reaction) hide, block
    - Pro: accurate; Cons: very few, like but not click like
  - --- weighted score of reactions.

ML Category

There are several ways to approach a recommendation problem:

- Simple rules, like recommending popular items:
  - Pros: Easy to implement and interpret; requires minimal computational resources.
  - Cons: Lacks personalization since the same recommendations are shown to all users.
- Embedding-based models, where "embedding" refers to low-dimensional representations:
  These models learn embeddings for users and items, then recommend based on the similarity between those embeddings.
  - Pros: Simple, efficient, and scalable; great for real-time recommendations; handles cold-start problems well.
  - Cons: Relying solely on similarity may not fully optimize business objectives like click-through rates or engagement.
- Learning to Rank approach, which reformulates the recommendation system as a ranking problem:
  This involves answering, “Given a query and a list of items, what’s the optimal order of these items from most to least relevant to the query?”

    - Pros: Directly optimizes the quality of recommendation lists based on user feedback (e.g., clicks, purchases); highly flexible, incorporating various features like embeddings, user behavior, and contextual data.
    - Cons: Computationally expensive, especially in real-time applications.

Given that we have a large dataset and want to balance efficiency and accuracy, a two-stage approach works best:

- Stage 1: Candidate Generation
  The goal here is to narrow down the pool of potential videos from billions to thousands. At this stage, efficiency matters more than accuracy, and false positives are acceptable. We can use simple rules or embedding-based models for multi-channel candidate retrieval.
- Stage 2: Scoring
  In this stage, the focus shifts to accurately scoring and ranking the candidate videos. Since accuracy takes priority over efficiency, we use a Learning to Rank approach to generate the final recommendations.

Recommendation System

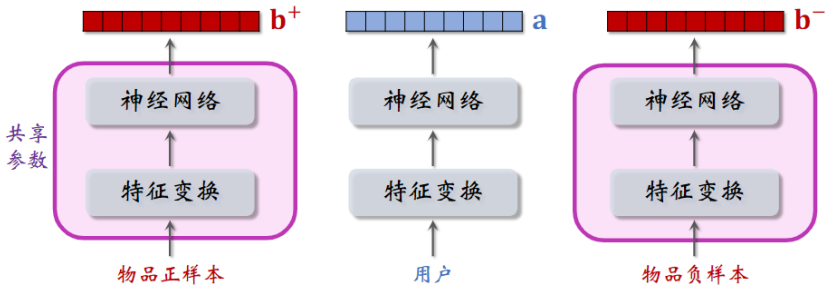
模型 - 训练 - 测试 - 评价

(样本) 训练样本的匹配

1. Pointwise 匹配

- 把召回看做二元分类任务
- 对于正样本，鼓励$\cos(\mathbf{a}, \mathbf{b})$ 接近 +1；对于负样本，鼓励$\cos(\mathbf{a}, \mathbf{b})$接近 −1
- 控制正负样本数量为 1: 2 或者 1: 3

2. Pairwise 匹配



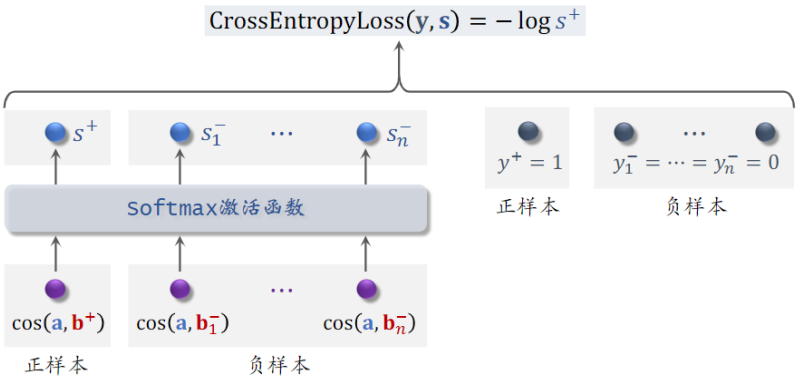
- 一条数据包含：一个用户，特征向量记作$\mathbf{a}$；

  一个该用户的正样本，特征向量记作 $\mathbf{b}^+$，一个该用户的负样本，特征向量记作 $\mathbf{b}^-$
- 鼓励 $\cos(\mathbf{a}, \mathbf{b}^+)$ 大于$\cos(\mathbf{a}, \mathbf{b}^-)$，如果 $\cos(\mathbf{a}, \mathbf{b}^+)$ 大于$\cos(\mathbf{a}, \mathbf{b}^-) + m$，则没有损失
- Triplet hinge loss: $L(\mathbf{a}, \mathbf{b}^+, \mathbf{b}^-) = \max\{0, \cos(\mathbf{a}, \mathbf{b}^-) + m - \cos(\mathbf{a}, \mathbf{b}^+)\}$

3. Listwise 匹配

- 一条数据包含：一个用户，特征向量记作$\mathbf{a}$；一个该用户的正样本，特征向量记作 $\mathbf{b}^+$，

  多个该用户的负样本，特征向量记作 $\mathbf{b}_1^-, \ldots, \mathbf{b}_n^-$
- 鼓励 $\cos(\mathbf{a}, \mathbf{b}^+)$尽量大， $\cos(\mathbf{a}, \mathbf{b}_1^-), \ldots, \cos(\mathbf{a}, \mathbf{b}_n^-)$尽量小
- 正样本 $y^+ = 1$，即鼓励 s^+趋于 1；负样本 $y_1^- = \ldots = y_n^- = 0$，即鼓励 $s_1^- \ldots s_n^-$趋于 0
- 用 y 和 s 的交叉熵作为损失函数，意思是鼓励 $\mathrm{Softmax}$ 的输出 s 接近标签 y



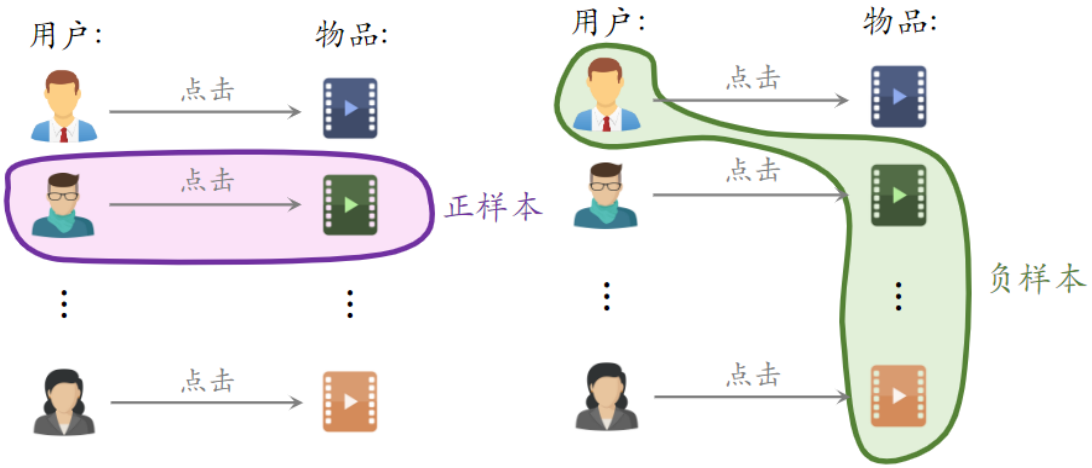
(样本) 训练样本选择

1. 正样本

- 曝光而且有点击的 用户—物品 二元组（用户对物品感兴趣）
- 问题：少部分物品占据大部分点击，导致正样本大多是热门物品
- 解决方案：过采样冷门物品，或降采样热门物品
  - 过采样（over-sampling）：正样本出现多次
  - 降采样（under-sampling）：以一定概率抛弃热门物品，抛弃的概率与样本的点击次数正相关

2. 简单负样本

- 定义：由于未被召回的物品大概率是用户不感兴趣的，所以简单负样本就是从未被召回的物品抽样。
  - 由于未被召回的物品 ≈ 全体物品， 简单负样本 ≈ 全体物品
- 抽样方法：均匀抽样 or 非均匀抽样 or 其他?
  - 均匀抽样的问题：由于大部分物品都是冷门物品，如果均匀采样，负样本大多是冷门物品，对冷门物品不公平。
  - 非均抽采样：为了打压热门物品，负样本抽样概率与热门程度（点击次数）正相关
  - Batch 内负样本：一个 batch 内有 n 个正样本，每个用户和 $n-1$ 个物品组成负样本，这个 batch 内一共有 $n(n-1)$ 个负样本，都是简单负样本。（因为第一个用户不喜欢第二个物品）
    - 问题： 一个物品出现在 batch 内的概率正比于点击次数，热门物品成为负样本的概率过大，
    - 解决：做训练的时候，将兴趣分数调整为$\cos(\mathbf{a}, \mathbf{b}_i) - \log p_i$



3. 困难负样本

- 定义：被粗排淘汰的物品（对于模型，这些物品比较困难成功识别为负样本）

  精排分数靠后的物品（非常困难）

训练数据：混合几种负样本，50% 负样本是简单负样本，50% 负样本是困难负样本（没通过排序的物品）

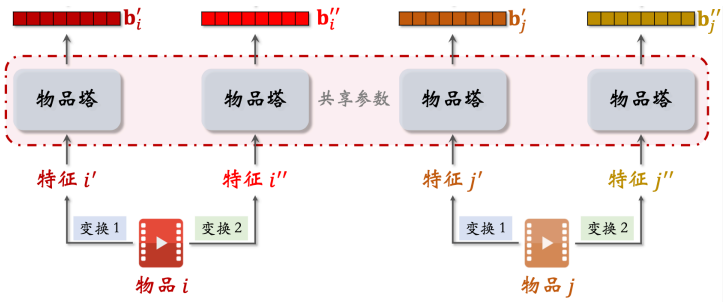
注意：召回的目标是快速找到用户可能感兴趣的物品，即区分用户 **不感兴趣** 和 **可能感兴趣** 的物品，而不是区分 **比较感兴趣** 和 **非常感兴趣** 的物品。曝光没点击的物品已经非常符合用户兴趣了，可以作为排序的负样本，不能作为召回的负样本

(数据) 自监督模型（长尾效应）

1. Motivation

- 推荐系统的头部效应严重: 少部分物品占据大部分点击，大部分物品的点击次数不高。
- 这导致高点击物品的表征学得好，长尾物品的表征学得不好。
- 所以用自监督学习做 data augmentation，更好地学习长尾物品的向量表征。

2. 模型结构



- 物品 i 的两个向量表征 $\mathbf{b}'_i$ 和 $\mathbf{b}''_i$ 有较高的相似度。
- 物品 i 和 j 的向量表征 $\mathbf{b}'_i$ 和 $\mathbf{b}''_j$ 有较低的相似度。
- 鼓励 $\cos(\mathbf{b}'_i, \mathbf{b}''_i)$ 尽量大， $\cos(\mathbf{b}'_i, \mathbf{b}''_j)$ 尽量小。

特征变换

- Random Mask： 随机选一些离散特征(比如类目)，把它们遮住。

  某物品的特征是$\mathcal{U}$= {数码,摄影}, Mask 后的类目特征是 $\mathcal{U}'$ = {default}。

- Dropout(仅对多值离散特征生效)：随机丢弃特征中 50% 的值。

  某物品的特征是$\mathcal{U}$= {数码,摄影}, Mask 后的类目特征是 $\mathcal{U}'$ = {摄影}。

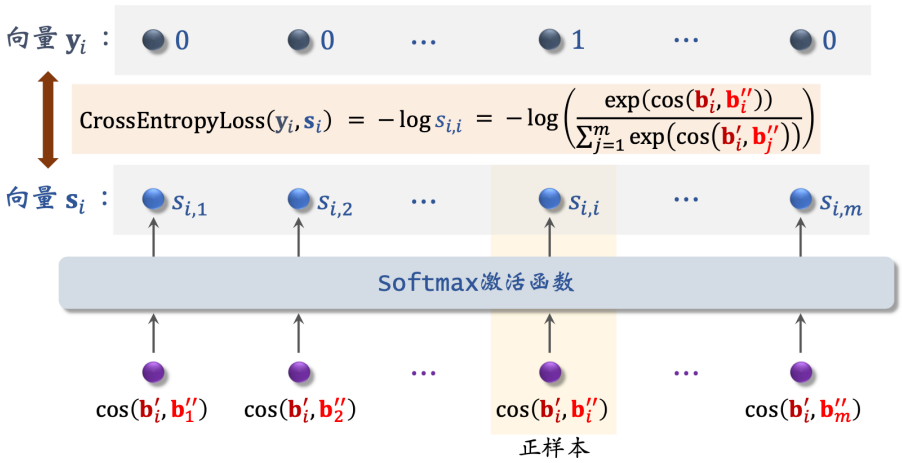
- 互补特征(complementary)：假设物品一共有 4 种特征， 随机分成两组。

  {ID，类目，关键词，城市}: { ID，default，关键词，default }, { default，类目，default，城市 }

- Mask 一组关联的特征。

  原始数据：u=女,v=美妆；Mask 后：u=女,v=[MASK]

3. 模型训练



- 从全体物品中均匀抽样，得到 m 个物品，作为一个 batch。
- 做两类特征变换，物品塔输出两组向量：

  $$\mathbf{b}'_1, \mathbf{b}'_2, \cdots, \mathbf{b}'_m \quad \text{和} \quad \mathbf{b}''_1, \mathbf{b}''_2, \cdots, \mathbf{b}''_m$$

- 第 i 个物品的损失函数：

  $$L_{\text{self}}[i] = -\log\left(\frac{\exp(\cos(\mathbf{b}'_i, \mathbf{b}''_i))}{\sum_{j=1}^{m}\exp(\cos(\mathbf{b}'_i, \mathbf{b}''_j))}\right).$$

- 对点击做随机抽样，得到 n 对用户—物品二元组，作为一个 batch。
- 从全体物品中均匀抽样，得到 m 个物品，作为一个 batch。
- 做梯度下降，使得损失减小：

  $$\frac{1}{n}\sum_{i=1}^{n} L_{\text{main}}[i] + \alpha \cdot \frac{1}{m}\sum_{j=1}^{m} L_{\text{self}}[j].$$

  双塔模型的损失　　自监督学习的损失

(数据) 冷启动

召回的难点

- 召回的依据
- ✔ 自带图片、文字、地点
- ✔ 算法或人工标注的标签
- ❌ 没有用户点击、点赞等信息
- 这些信息可以反映笔记的质量，以及哪类用户喜欢该笔记
- ❌ 没有笔记 ID embedding
- ID embedding 是从用户和笔记的交互中学习出来的
- 冷启召回的困难
- 缺少用户交互，还没学好笔记 ID embedding，导致双塔模型效果不好
- 双塔模型是推荐系统中最重要的召回通道，没有之一
- 缺少用户交互，导致 ItemCF 不适用

(样本) 探索

##

(训练) 模型更新-实时性

全量更新

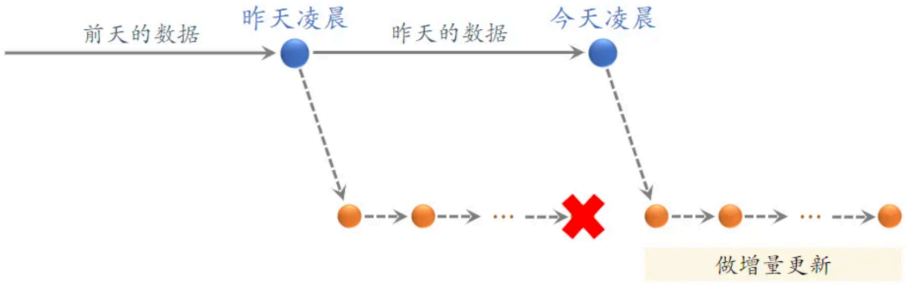
- 今天凌晨，用昨天全量的数据训练模型，在昨天模型的基础上更新所有。
- 用所有数据训练 1 epoch，即每天数据只用一遍。
- 发布新的 用户塔神经网络 和 物品向量，供线上召回使用。

增量更新

- 由于用户兴趣会随时发生变化，实时收集线上数据，对模型做 online learning，增量更新 User ID Embedding 参数 (不更新神经网络其他部分的参数。)
- 即嵌住在线接受的参数，只更新 Embedding 层的参数，这是出于工程实现的考虑。
- 发布 用户 ID Embedding，供用户塔在线上计算用户向量。
- 每天晚上，基于增量更新的模型会被删掉（图中的红x）

问题：能否只做增量更新，不做全量更新?

因为在不同时段，用户行为不一样，所以分批数据间偏差很大。全量更新随机打乱 shuffle 一天的数据，随机打乱消除了不同时段的差别。而增量更新按照数据从早到晚的顺序，会使得训练效果不好。



推荐模型

- 召回: (1) 分别获得 item 和 user 的 embedding; (2) 计算 interest score
  - 如何获得 embedding: (a) interaction vector, (b) 分解的 interaction vector, (c) NN
  - 如何计算 interest score: (a) similarity index (e.g., cosin similarity), (b) NN
- 排序: (1) 计算 interest score
  - wide NN: (a) simple wide; (b) FM; (c) DCN
  - deep NN

(召回) 协同过滤

1. 模型的核心思想

协同过滤的核心思想是利用用户与商品之间的交互信息，基于**相似性原则**进行推荐：

- 如果两个用户的行为相似，他们可能喜欢相同的商品（用户协同过滤）。
- 如果两个商品被类似的用户喜欢，这两个商品可能彼此相似（商品协同过滤）。

The core idea is to make recommendations based on similarities between users or items.

2. 模型训练/准备（离线阶段）

1. 构建”用户-物品“交互列表：
    - 收集用户行为数据（如点击、购买、评分等），记录每个用户与物品的交互信息。
    - 构建 ”用户-物品交互“列表，使得给定任意用户ID，可以找到他近期感兴趣的物品列表。
2. 构建”物品-物品“相似列表：
    - 使用共现次数或余弦相似度计算两两物品之间的相似度：
    - 为每个物品找到最相似的Top-K物品，并构建物品-物品“相似列表，使得给定任意物品ID，可以快速找到它最相似的 k 个物品。

3. 推理/Inference（在线阶段）

1. 获取用户历史行为：根据用户ID，通过”用户-物品“交互列表获取用户最近感兴趣的物品列表。
2. 检索相似物品：对用户感兴趣的每个物品，通过”物品-物品“相似列表找到其Top-K相似物品。
3. 兴趣分数计算：对所有检索到的物品，计算用户对每个物品的兴趣分数
4. 排序并返回结果：根据兴趣分数对候选物品排序，选出Top-M个物品作为推荐结果。

4. Pros and Cons

5. Discussion

- 两两物品之间的相似度的其他方法：
    - **基于内存的协同过滤（Memory-Based CF）：** 直接利用交互数据计算相似度。

      基于模型的协同过滤（Model-Based CF）： 构建预测模型来获得物品向量，来预测用户对未交互商品的兴趣。
- Model-Based CF： **矩阵分解**方法（如SVD或ALS）

  [矩阵分解用不同方式获得embedding，同样方法计算相似度]
    1. 将用户-商品的交互数据表示为一个稀疏矩阵（例如用户评分矩阵）。
    2. 利用矩阵分解技术将该矩阵分解为两个低维矩阵：
        - 用户矩阵（用户的潜在特征）。
        - 商品矩阵（商品的潜在特征）。
    3. 用户和商品的潜在特征向量通过点积计算评分，用于预测用户对未交互商品的兴趣。
- Model-Based CF： **深度学习**方法（如神经协同过滤 NCF）

  [NCF用不同方式获得embedding，不同方法计算相似度]
    1. 用户和商品的ID通过嵌入层映射到低维向量。
    2. 用户和商品的嵌入向量作为模型输入，经过一系列神经网络层学习非线性交互特征。
    3. 最终输出用户对商品的预测评分。

(召回) 双塔模型

1. 模型的核心思想

双塔模型的核心思想是通过两个独立的神经网络（两个塔）分别对用户和商品的信息进行编码，将其转化为低维向量（embedding）。然后，利用相似度度量（如点积或余弦相似度）计算用户与商品的匹配程度。匹配程度越高的商品，越有可能被推荐给用户。

The core idea is to use two independent neural networks, or towers, to encode information about users and items into two low-dimensional vectors called embeddings. Then, we calculate the similarity between these embeddings using methods like cosine similarity to measure how well an item matches a user. The higher the similarity score, the more likely the item will be recommended to the user.

The core idea is to separately learn embeddings for users and items using two independent networks, and then make recommendations based on user and item embedding similarity.

2. 模型结构

双塔模型由两个塔组成：

- 用户塔（User Tower）：输入用户特征，通过一个神经网络将这些特征编码为用户的embedding。
- 商品塔（Item Tower）：输入商品特征，通过另一个神经网络将这些特征编码为商品的embedding。

用户塔和商品塔分别生成用户和商品的embedding，通过点积、余弦相似度或其他距离函数计算匹配分数，衡量用户与商品之间的相关性。

The model has two towers: user tower and item tower.

- User Tower : The user tower processes user-related features through a neural network to produce the user embedding.
- Item Tower :The item tower processes item-related features through another neural network to generate the item embedding.

The two towers generate embeddings for users and items separately. The similarity between these embeddings, calculated using methods like cosine similarity, determines how relevant an item is to a user.

3. 模型训练

训练样本由两种样本组成：

- 正样本： 用户与商品的实际交互的对（如用户点击或购买了某个商品）。
- 负样本： 用户未与商品发生交互的对。负样本通常通过负采样生成，如随机从用户未交互的商品中挑选。

双塔模型主要使用对比学习的损失函数，目标是拉大正样本和负样本得分之间的差距。

训练完成后，可以用商品塔对所有商品生成embedding，并存储起来，供推断阶段使用。

Training samples consist of two types:

- Positive Samples: These are user-item pairs where the user interacted with the item, like clicking on or purchasing it.
- Negative Samples: These are user-item pairs where there was no interaction. Negative samples are created by sampling items that the user didn't interact with.

The two-tower model uses contrastive loss to train the model, aiming to make the score for positive samples significantly higher than for negative ones.

After training is complete, the item tower is used to generate embeddings for all items. These embeddings are precomputed and stored, making them readily available during the inference stage.

4. 推理过程（Inference）

输入：实时用户信息（如用户行为序列、上下文特征）。

1. 用户信息通过用户塔计算出用户的embedding。
2. 用用户的embedding 与预先计算的商品embedding 计算用户-商品相似度。
3. 根据匹配分数对商品排序，返回Top N的商品作为推荐结果。

Input: Real-time user information, such as their behavioral history or contextual features.

1. User information is processed by the user tower to generate the user embedding.
2. The user embedding is compared with precomputed item embeddings to calculate user-item similarity scores.
3. Items are ranked based on their similarity scores, and the top N items are selected as recommendations.

- 双塔模型的召回
  - 离线存储：把物品向量 $\mathbf{b}$ 存入向量数据库
    i. 完成训练之后，用物品塔计算每个物品的特征向量 $\mathbf{b}$
    ii. 把几亿个物品向量 $\mathbf{b}$ 存入向量数据库（比如 Milvus、Faiss、HnswLib ）
    iii. 向量数据库建索引，以便加速最近邻查找
  - 线上召回：查找用户最感兴趣的 k 个物品
    i. 给定用户 ID 和画像，线上用神经网络现算（**实时计算**）用户向量 $\mathbf{a}$
    ii. 最近邻查找：
      - 把向量 $\mathbf{a}$ 作为 query，调用向量数据库做最近邻查找
      - 返回余弦相似度最大的 k 个物品，作为召回结果
- 为什么事先存储物品向量 $\mathbf{b}$ ，线上现算用户向量 $\mathbf{a}$ ？
  - 每做一次召回，用到一个用户向量 $\mathbf{a}$ ，几亿物品向量 $\mathbf{b}$ （线上算物品向量的代价过大）
  - 用户兴趣动态变化，而物品特征相对稳定（可以离线存储用户向量，但不利于推荐效果）

5. Pros and Cons

(排序) 多任务模型

1. 模型的核心思想

在推荐系统中，我们通常需要同时预测多个关键指标，比如点击率、转化率和点赞率等。多任务模型的核心在于，先通过一个通用模型捕捉用户和物品之间的基础关系，提取一个共享的特征表示，然后这些共享特征被不同模块来分别用来预测点击率、转化率等目标。

In a recommendation system, we often need to predict multiple key metrics at the same time, such as click-through rate, conversion rate, and like rate. The core idea of a multi-task model is to use a shared model to capture the fundamental relationship between users and items, generating a shared feature representation. This shared representation is then used by seperate modules to predict metrics for each task .

2. 模型结构

模型由两个部分组成:

- 共享层：共享层是一个NN，它的输入是所有的特征的concatenation，输出是一个 representation vector，表示用户和物品之间的交互信息。
- 任务特定层：每个任务对应一个NN，它的输入是 representation vector，输出是对每一个metric的预测。

The model has two main parts:

- Shared Layer: This is a neural network that takes all the features concatenated together as input and outputs a representation vector. This vector captures the interaction information between users and items.
- Task-Specific Layer: Each task has its own neural network that uses the representation vector as input and outputs predictions for its specific metric.

3. 模型训练

训练数据：使用包含多个目标标签的数据集，例如：点赞行为：1表示点赞，0表示未点赞。

损失函数：每个任务使用二分类交叉熵损失函数或MSE：$L_{\text{task}} = -y \cdot \ln(\hat{y}) - (1-y) \cdot \ln(1-\hat{y})$

总损失为各任务损失的加权和： $L = \alpha_1 \cdot L_{\text{CTR}} + \alpha_2 \cdot L_{\text{Like}} + \alpha_3 \cdot L_{\text{Collect}} + \alpha_4 \cdot L_{\text{Share}}$

5. Pros and Cons

(排序) 多专家模型

1. 模型的核心思想

在推荐系统中，我们通常需要预测多个关键指标，比如点击率、转化率和点赞率等。多专家模型的核心思想是通过多个独立的“专家模块”捕捉用户和物品之间不同方面的关系。每个专家模块生成一个向量，用于反映这些关系的特征。然后，这些向量被动态加权和组合，生成与每个指标最相关的多个特征表示。最后，这些特征表示被输入到对应的任务特定模块中，用于预测各项指标。

In a recommendation system, we often need to predict multiple key metrics like click-through rate, conversion rate, and like rate. The core idea of a mixture of experts model is to use several independent "expert modules“ to capture different aspects of the relationship between users and items, each expert module generats a vector reflecting the feature of the relationships. These vectors are then dynamically weighted and combined to create mutiple representations that are most relevant to each metric. Finally, these representations are fed into corresponding task-specific modules to predict metrics.

2. 模型结构

模型由三部分组成：

- 专家层（Expert Layer）： 由多个专家模块组成，每个模块是一个神经网络（NN）。
  - NN输入： 用户特征、物品特征，以及场景特征的concatenation
  - NN输出： 一个特征向量，表示其对学习用户和物品之间的关系的特定理解。
- 门控网络层（Gating Network Layer）： 由多个模块组成，每个模块对应一个任务，每个模块是一个NN
  - NN输入： 用户特征、物品特征，以及场景特征的concatenation
  - NN输出： 一个向量，向量中的值表示为每个专家分配的权重，用于专家层输出的特征向量的组合。
- 任务特定层（Task-Specific Layer）： 由多个预测模块组成，每个模块对应一个任务，每个模块是一个NN
  - NN输入： 加权后的任务特定特征向量。
  - NN输出： 每个任务的预测值（如点击率、转化率等）。

3. 模型训练

和多任务模型一样

5. Pros and Cons

(排序) 特征交叉 wide：Deep Feature Interaction

- Wide & Deep 模型

1. 模型的核心思想

Wide 部分依赖**手工设计的交叉特征**，捕获用户与商品之间的显式关系

2. 模型的结构

Wide 部分是一个线性模型，直接用稀疏向量表示交叉特征，公式为：$y_{\text{wide}} = \mathbf{w}_{\text{wide}}^{\top}\mathbf{x}_{\text{wide}}$

$\mathbf{x}_{\text{wide}} = [\mathbf{x}_{\text{basic}}, \mathbf{x}_{\text{cross}}] = [\text{年龄}, \text{价格}, \text{性别}, \text{商品类别}, \text{性别} \times \text{商品类别}] = [25, 100, 1,\ 0, 1, 0,\ 1, 0, 0, 0]$

3. 优缺点

优点：简单直观，适合已知显式交互关系的场景。

缺点：依赖领域知识，难以动态扩展到复杂特征组合。

- DeepFM 模型

1. 模型的核心思想

将类别特征嵌入到低维向量空间，并用内积计算二阶交叉特征，自动学习二阶交叉特征

2. 模型的结构

男性 [1,2]，美妆 [2,3]，x=[性别，类别，男_美妆，男_服饰，女_美妆，女_服饰]=[1,2,2,3,8,0,0,0]

3. 优缺点

优点：无需手动设计交叉特征，自动捕获二阶显式交互，适合稀疏高维场景。

缺点：仅限于二阶交叉关系，高阶交叉需要 Deep 部分建模。

- DCN（Deep Cross Network）模型

1. 模型的核心思想

Wide 用 CrossNet，将类别特征嵌入到低维向量空间，通过外积操作逐层生成高阶交叉特征

2. 模型的结构

$x_l + 1 = x_0 \circ (W_l^T x_l + b_l) + x_l$

$\mathbf{x}_0$：初始输入向量（拼接的原始特征或嵌入向量）$\mathbf{x}_l$：第 l层的输出向量。

$\mathbf{W}_l$、$\mathbf{b}_l$：权重矩阵和偏置。○： Hadamard Product。

3. 优缺点

优点：显式建模高阶交叉特征，计算效率高，且模型可解释性强。

缺点：外积操作对高维特征计算成本较高。

(模型) 多样性

Maximal Marginal Relevance (MMR)

1. 模型的核心思想

MMR 是一个逐步挑选内容的过程，每一步都在追求'相关性最高'的同时，避免选出的内容之间相似，从而平衡推荐的相关性和多样性。

2. 模型的结构

假设我有一份候选列表，比如 100 篇文章。它们已经按照与用户的相关性评分排列好，我们进行进一步挑选。

1. 初始化候选集合：首先，我把"已选清单"初始化为空，把所有文章放在"待选清单"里。然后，我会先挑评分最高的那一篇文章，把它放到"已选清单"中。
2. 开始迭代挑选：接下来，我会重复以下过程，直到选出需要的 k 篇文章：

- 计算分数：对于"待选清单"里的每一篇文章，我会重新计算它的得分。这次的得分不仅考虑它与用户需求的相关性，还会扣掉它与"已选清单"中文章的相似度。这样，那些内容重复的文章得分会被压低。
- 挑选得分最高的：从"待选清单"中选出得分最高的一篇，把它移到"已选清单"。

• 计算集合 $\mathcal{R}$ 中每个物品 i 的 Marginal Relevance 分数：

$$\mathrm{MR}_i = \underbrace{\theta \cdot \mathrm{reward}_i}_{\text{物品 } i \text{ 的精排分数}} - \underbrace{(1-\theta) \cdot \max_{j \in \mathcal{S}} \mathrm{sim}(i,j)}_{\text{物品 } i \text{ 的多样性分数}}.$$

3. 优缺点

优点：通用性强 (Sim 可以用多种方法计算)；

缺点：权重参数 θ 难以调优；难以扩展到大规模数据

Determinantal Point Process (DPP)

1. 模型的核心思想

DPP 的目标是从 n 个候选物品中，挑选 k 个物品，使得选出的物品的评分（相关性）尽可能高，选出的物品之间的相似度尽可能低。DPP 通过数学工具**行列式**（Determinant）来量化一个集合的多样性。

2. 模型的结构

假设我们有一份候选列表，比如 100 篇文章。这些文章不仅有与用户需求相关的评分，还用特征向量表示它们的内容信息，比如文章的主题、风格等。

1. 初始化候选集合：首先，我把“已选清单”初始化为空，把所有文章放在“待选清单”里。同时，我们用这些文章的特征向量构建一个“相似度矩阵”，用来衡量文章之间的相似程度。
2. 开始挑选文章：接下来，我会重复以下过程，直到选出需要的 kkk 篇文章：

- 计算子集质量：对于“待选清单”中的每一篇文章，尝试将它加入“已选清单”，并计算加入后的子集的整体质量和多样性。子集的多样性通过一个叫“行列式”来衡量，行列式值越大，表示文章的组合越多样性越高。
- 挑选最优文章：选择那篇能让“已选清单”的质量和多样性提升最多的文章，把它从“待选清单”中移到“已选清单”。

Tips: 标准的 DPP 从 n 中选 k 个，需要 C_k^n，这里是贪心算法版的DPP，每次只选一个。

3. 优缺点

优点：全局多样性更强

缺点：权重参数 θ 难以调优；行列式计算量较大

(模型) 时间序列

测试方案

指标