**Thinking1 常用的路径规划算法有哪些?**

* 最短路径规划：
* Dijkstra算法

原理：

指定起点s,引进两个集合S和U。

S:是记录已求出最短路径的顶点(以及相应的最短路径长度)

U:记录还未求出最短路径的顶点(以及该顶点到起点s的距离)。

Step1，S中只有起点s，从U中找出路径最短的顶点，将其加入到S中

Step2，更新U中的顶点和顶点对应的路径

重复Step1和Step2 直到遍历完所有顶点

重复Step1，即更新S：从U中找出路径最短的顶点，加入到S中；

重复Step2，即更新U中的顶点和顶点对应的路径

* Floyd算法

原理：

使用了动态规划的思想，将图中顶点编号为1-n

以两点之间最短距离经过的顶点中最大的顶点编号作为阶段，两点间目前算出的最短路径作为状态的值

假设为顶点编号i和j两点经过 最大顶点编号不超过k的最短路径长度，那么：



* 基于高德地图的路径规划：
  + Step1，数据采集，获取不同城市的地点名称
  + Step2，获取地点的坐标（经度、维度）。使用高德地图API，获取指定地点的所在经度、维度
  + Step3，计算两点之间的距离。使用高德地图API，获取指定两点之间的距离
  + Step4，使用Dijkstra计算从start到end的最优路径
  + Step5，使用数据进行测试验证
  + 后续可以完善的地方：

1）不同城市的地图路径规划

2）计算更多的地点（不仅是地铁站点，可以是其他道路名称）

3）如果是乘坐地铁，可以找到附近的地铁站（start附近，end附近）

4）给出不同交通工具的路径规划（乘车、公交、地铁）

5）将路径规划，在地图中进行显示（flask部署）

6）将算法进行API接口封装

**Thinking2 推荐系统的架构是怎样的？**

一般来说，推荐系统包括四种推荐方式：

热门推荐：就是热门排行榜的概念。这种推荐方式不仅仅在IT系统，在平常的生活中也是处处存在的。这应该是效果最好的一种推荐方式，毕竟热门推荐的物品都是位于曝光量比较高的位置的。

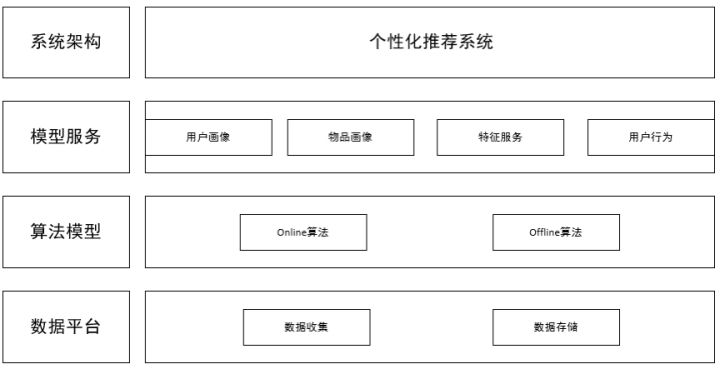
人工推荐：人工干预的推荐内容。相比于依赖热门和算法来进行推荐。一些热点时事如世界杯、nba总决赛等就需要人工加入推荐列表。另一方面，热点新闻带来的推荐效果也是很高的。

相关推荐：相关推荐有点类似于关联规则的个性化推荐，就是在你阅读一个内容的时候，会提示你阅读与此相关的内容。

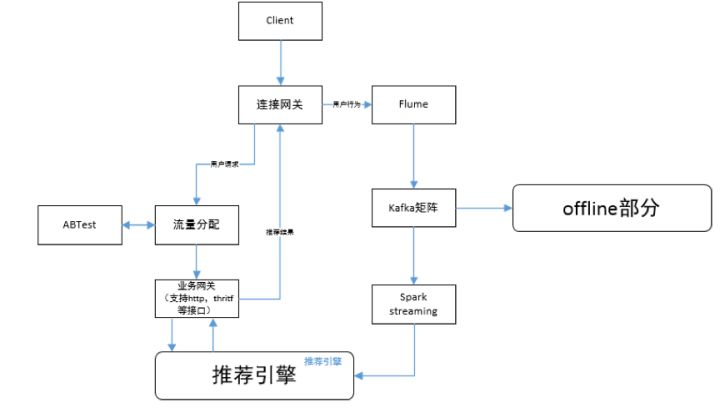
个性化推荐：基于用户的历史行为做出的内容推荐。也是本文主要讲述的内容。

其中，前三者是和机器学习没有任何关系的，但却是推荐效果最好的三种方式。一般说来，这部分内容应该占到总的推荐内容的80%左右，另外20%则是对长尾内容的个性化推荐。

推荐系统架构



online部分架构



核心模块

· 业务网关，推荐服务的入口，负责推荐请求的合法性检查，组装请求响应的结果。

· 推荐引擎，推荐系统核心，包括online逻辑，召回、过滤、特征计算、排序、 多样化等处理过程。

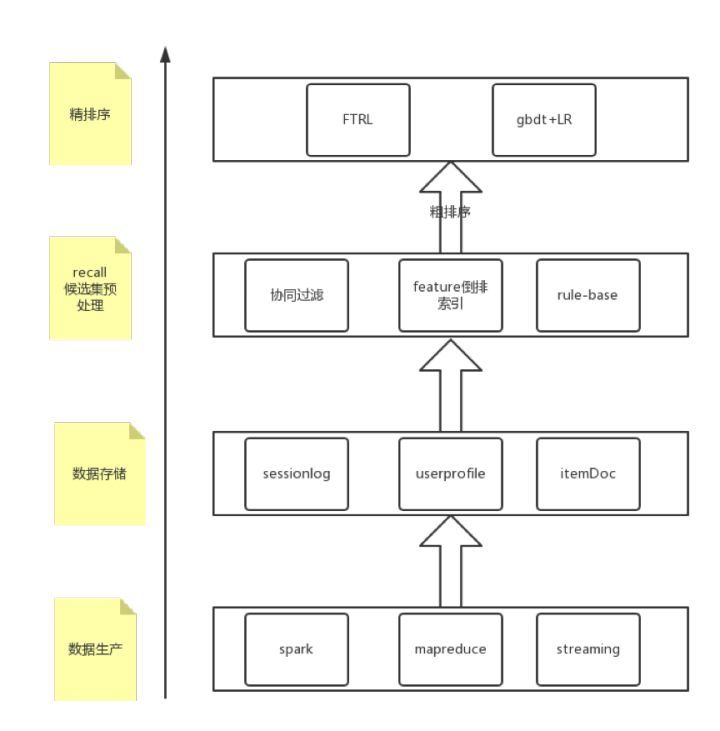
数据路径

1. 请求的刷新从gateway，经过流量分配模块，传到业务gateway，业务gateway支持http，tcp（使用thirtf协议或者protobuf 协议）等多种类型接口；

2. 用户行为数据，从gateway到Flume agent，然后到kafka，为后面online，realtime userprofile部分的提供实时数据，也为offline部分的数据存储系统提供数据。

offline部分架构

本文从大框架上介绍推荐系统架构，在许多公司面试中会给你一个推荐或者数据挖掘的问题，比如让你简单设计一个feed流推荐系统，所以需要对推荐系统的整体框架要了解。下面是一个推荐系统的主要部分：



从框架的角度看，推荐系统基本可以分为数据层、召回层、排序层。

数据层包括数据生成和数据存储，主要是利用各种数据处理工具对原始日志进行清洗，处理成格式化的数据，落地到不同类型的存储系统中，供下游的算法和模型使用。

Session log：对原始数据进行清洗合并，session log一般就是清洗合并后的数据，后续的算法和统计都是根据session log进行再加工。

User profile：对用户属性和行为等信息进行采集和统计，为后续算法提供特征支持。

Item Doc：对视频、商品等属性、曝光、点击等字段进行统计， 为后续算法提供特征支持。

召回层主要是从用户的历史行为、实时行为等角度利用各种触发策略产生推荐的候选集，对不同的策略和算法产生的候选集进行融合并按照产品规则进行过滤，一般融合和过滤后的候选集还是比较多的，一次线上请求过来之后线上系统无法对那么多的候选集进行排序，所以在召回层一般还会有粗排序，对融合的候选集进行一次粗排序，过滤掉粗排分数较低的候选集。

排序层主要是利用机器学习的模型对召回层筛选出来的候选集进行精排序。

数据特征

数据决定了特征，特征决定了效果的上限，模型决定了接近效果上限的程度。



1. 用户主动行为数据记录了用户在平台的的各种行为，这些行为一方面用于候选集触发算法（在下一部分介绍）中的离线计算（主要是浏览、下单），另外一方面，这些行为代表的意图的强弱不同，因此在训练重排序模型时可以针对不同的行为设定不同的回归目标值，以更细地刻画用户的行为强弱程度。此外，用户对deal的这些行为还可以作为重排序模型的交叉特征，用于模型的离线训练和在线预测。

2. 负反馈数据反映了当前的结果可能在某些方面不能满足用户的需求，因此在后续的候选集触发过程中需要考虑对特定的因素进行过滤或者降权，降低负面因素再次出现的几率，提高用户体验；同时在重排序的模型训练中，负反馈数据可以作为不可多得的负例参与模型训练，这些负例要比那些展示后未点击、未下单的样本显著的多。

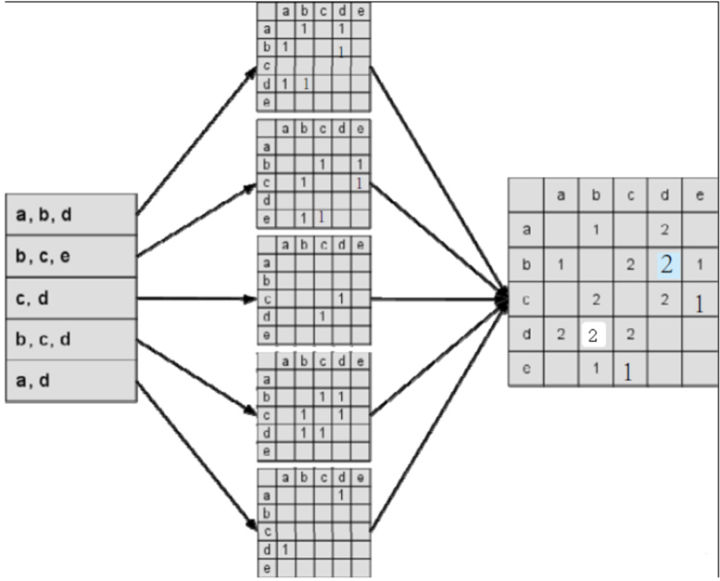
3. 用户画像是刻画用户属性的基础数据，其中有些是直接获取的原始数据，有些是经过挖掘的二次加工数据，比如用户的聚类和向量化，这些属性一方面可以用于候选集触发过程中对deal进行加权或降权，另外一方面可以作为重排序模型中的用户维度特征。

召回层（Recall）

协同过滤

协同过滤（Collaborative Filtering）可说是推荐系统里资历最老最经典的一种算法了，如 userCF、itemCF。原理是基于用户对内容的行为协同，为某一用户没有看过的某条内容作出点击预测。实现方法有很多种，如传统的 Memory-based 方法、基于矩阵分解的方法（LFM/SVD/SDV++）、基于 DNN 的方法。

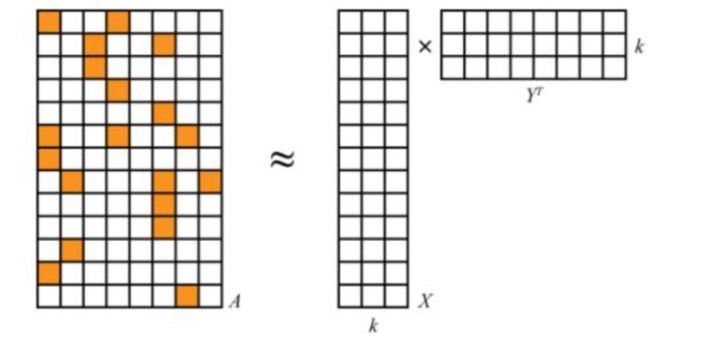
Memory-based 方法很简单，是基于统计的一种算法。以 item-based CF 举例：



根据用户点击行为，我们可以统计出 item-item 的共现矩阵（矩阵单元内为 item i 与 item j 共同被用户点击的次数），再依此通过Jaccard相似度/余弦相似度/欧氏距离得出 item 相似度矩阵，最后根据用户的点击记录检索出 topK 相似的内容推荐给用户。在计算过程中需要考虑一些因素，比如热门物品对相似度计算的影响、不同倾向的用户的影响等等。

然而 Memory-based 方法不能解决的问题是，当我们的矩阵很稀疏时，大多数 item 和 item 之间是没有关联的（相似度为0），这也就造成最后我们召回的内容覆盖率很低，也许大多集中在头部内容。于是基于矩阵分解的方法诞生了。

MF（Matrix Factorization）的原理是将一个高维稀疏矩阵分解成两个低秩矩阵，其中 k 被称为隐向量维度。在原始的稀疏矩阵 R 中，大部分二阶特征的关系系数是缺失的。而通过训练模型最小化 R 和预测矩阵 R‘ 的损失（如最小二乘），可以求出任意 Ri,j 的值。



MF 可说是大部分推荐系统里协同过滤的标杆方法了，但仍然存在一些问题。比如过于稀疏的矩阵对于最后评分的预测依然有很大影响，并且当用户特征或者内容特征缺失（即冷启动）时，无法进行合理的预测。此时，基于深度学习的一些尝试开始了。如基于DNN实现，可以很轻易地将内容的一些语义特征，以及用户的固有属性与行为特征拼接在一起作为神经网络输入来训练，可以在之前行为协同的前提下加入对内容特征的学习，从而解决冷启动问题。感兴趣的同学可以阅读相关论文，在此不做展开。

基于内容的召回

主要是以之前 NLP 得到的内容画像为基础，以item 对应分类/主题/关键词的权重建立召回，依据用户画像的相应权重和内容画像的距离排序召回。

基于用户群

首先我们需要对用户分群，聚类的方案有很多，

1、对item进行向量化（w2v）然后对item进行聚类，用户对item的行为就可以把item的簇赋值到user身上。

2、直接对用户进行向量化，比如降维。

总之最终的目的就是将用户embedding成一个向量，然后在对用户向量进行聚类，一般k-means就可以胜任大部分的场景。

倒排链

tag-itemList，对每个用户的tag进行遍历，然后通过倒排链快速找到含有该tag的itemList然后topN抽取。

子策略融合

为了结合不同触发算法的优点，同时提高候选集的多样性和覆盖率，需要将不同的触发算法融合在一起。常见的融合的方法有以下几种[3]：

1. 加权型：最简单的融合方法就是根据经验值对不同算法赋给不同的权重，对各个算法产生的候选集按照给定的权重进行加权，然后再按照权重排序。

2. 分级型：优先采用效果好的算法，当产生的候选集大小不足以满足目标值时，再使用效果次好的算法，依此类推。

3. 调制型：不同的算法按照不同的比例产生一定量的候选集，然后叠加产生最终总的候选集。

4. 过滤型：当前的算法对前一级算法产生的候选集进行过滤，依此类推，候选集被逐级过滤，最终产生一个小而精的候选集合。

目前我们使用的方法集成了调制和分级两种融合方法，不同的算法根据历史效果表现给定不同的候选集构成比例，同时优先采用效果好的算法触发，如果候选集不够大，再采用效果次之的算法触发，依此类推。

模型排序（Ranking）

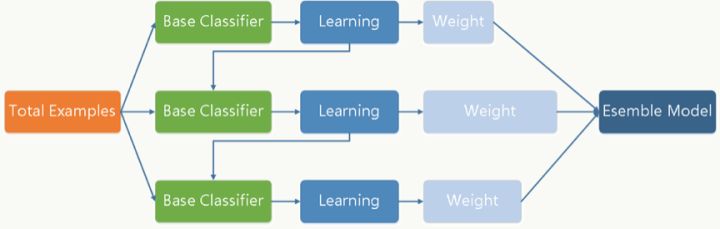
如上所述，对于不同算法触发出来的候选集，只是根据算法的历史效果决定算法产生的item的位置显得有些简单粗暴，同时，在每个算法的内部，不同item的顺序也只是简单的由一个或者几个因素决定，这些排序的方法只能用于第一步的初选过程，最终的排序结果需要借助机器学习的方法，使用相关的排序模型，综合多方面的因素来确定。

1、模型选择和比较

非线性模型能较好的捕捉特征中的非线性关系，但训练和预测的代价相对线性模型要高一些，这也导致了非线性模型的更新周期相对要长。反之，线性模型对特征的处理要求比较高，需要凭借领域知识和经验人工对特征做一些先期处理，但因为线性模型简单，在训练和预测时效率较高。因此在更新周期上也可以做的更短，还可以结合业务做一些在线学习的尝试。在我们的实践中，非线性模型和线性模型都有应用。

非线性模型

目前我们主要采用了非线性的树模型gbdt，相对于线性模型，非线性模型可以更好的处理特征中的非线性关系，不必像线性模型那样在特征处理和特征组合上花费比较大的精力。gbdt是一个加性模型，由很多个树组成，后面的树不断拟合前一颗树的残差，而且每一个树带入的都是全训练集，由此可以减小过拟合的影响。



线性模型

目前应用比较多的线性模型非Logistic Regression莫属了。为了能实时捕捉数据分布的变化，我们引入了online learning，接入实时数据流，使用google提出的FTRL方法对模型进行在线更新。  
  
主要的步骤如下：

· 在线写特征向量到HBase

· Storm解析实时点击和曝光日志流，改写HBase中对应特征向量的label

· 通过FTRL更新模型权重

· 将新的模型参数应用于线上

2. 数据

· 采样：对于点击率预估而言，正负样本严重不均衡，所以需要对负例做一些采样。

· 负例：正例一般是用户产生点击、下载、分享等转换行为的样本，但是用户没有转换行为的样本是否就一定是负例呢？其实不然，很多展现其实用户根本没有看到，所以把这样样本视为负例是不合理的，也会影响模型的效果。比较常用的方法是skip-above，即用户点击的item位置以上的展现才可能视作负例。当然，上面的负例都是隐式的负反馈数据，除此之外，我们还有用户主动删除的显示负反馈数据，这些数据是高质量的负例。

· 去噪：对于数据中混杂的刷单等类作弊行为的数据，要将其排除出训练数据，否则会直接影响模型的效果。

3. 特征

在我们目前的重排序模型中，大概分为以下几类特征：

· item维度的特征：主要是item本身的一些属性，包括category、pv、ctr、sub-category、tag等

· user维度的特征：包括用户等级、用户的人口属性、用户的客户端类型等

· user、deal的交叉特征：包括用户对item的category的点击、收藏等

对于非线性模型，上述特征可以直接使用；而对于线性模型，则需要对特征值做一些分桶、归一化等处理，使特征值成为0~1之间的连续值或01二值。

**Thinking3 你都了解推荐系统中的哪些常用算法？原理是怎样的？**

推荐算法大致可以分为三类：基于内容的推荐算法、协同过滤推荐算法和基于知识的推荐算法。

* 基于内容的推荐算法，原理是用户喜欢和自己关注过的Item在内容上类似的Item，比如你看了哈利波特I，基于内容的推荐算法发现哈利波特II-VI，与你以前观看的在内容上面（共有很多关键词）有很大关联性，就把后者推荐给你，这种方法可以避免Item的冷启动问题（冷启动：如果一个Item从没有被关注过，其他推荐算法则很少会去推荐，但是基于内容的推荐算法可以分析Item之间的关系，实现推荐），弊端在于推荐的Item可能会重复，典型的就是新闻推荐，如果你看了一则关于MH370的新闻，很可能推荐的新闻和你浏览过的，内容一致；另外一个弊端则是对于一些多媒体的推荐（比如音乐、电影、图片等)由于很难提内容特征，则很难进行推荐，一种解决方式则是人工给这些Item打标签。
* 协同过滤算法，原理是用户喜欢那些具有相似兴趣的用户喜欢过的商品，比如你的朋友喜欢电影哈利波特I，那么就会推荐给你，这是最简单的基于用户的协同过滤算法（user-based collaboratIve filtering），还有一种是基于Item的协同过滤算法（item-based collaborative filtering），这两种方法都是将用户的所有数据读入到内存中进行运算的，因此成为Memory-based Collaborative Filtering，另一种则是Model-based collaborative filtering，包括Aspect Model，pLSA，LDA，聚类，SVD，Matrix Factorization等，这种方法训练过程比较长，但是训练完成后，推荐过程比较快。
* 基于知识的推荐算法，也有人将这种方法归为基于内容的推荐，这种方法比较典型的是构建领域本体，或者是建立一定的规则，进行推荐。
* 混合推荐算法，则会融合以上方法，以加权或者串联、并联等方式尽心融合。
* 基于spark的推荐系统
* 推荐系统还包括很多方法，其实机器学习或者数据挖掘里面的方法，很多都可以应用在推荐系统中，比如说LR、GBDT、RF（这三种方法在一些电商推荐里面经常用到），社交网络里面的图结构等，都可以说是推荐方法。

**Thinking4 我们在课上讲解过常用的机器学习，深度学习模型，推荐系统算法，以及启发式算法，路径规划原理等，针对这些模块，请你针对其中一个进行思维导图梳理**

机器学习思维导图：

