

(第一版)

淘宝搜索基础算法团队出品 编著



搜索事业部

搜索基础算法团队工作室·杭州

各类主要的互联网服务，包括搜索、广告、推荐、等等，它们的一个典型共同特征，就是利用不断增强的计算处理能力和日益丰富的资源种类，对百万乃至上百亿量级以上的超大规模数据进行分析和挖掘，数据维度包罗万象，比如网页内容、用户行为、时间序列、等等，去充分理解消费者需求，定位供给端的品类和质量，建立一个良性的买家和卖家的公平交易平台；

淘宝搜索作为平台的一个重要联系买家和卖家的产品形态，由于其以下的特有属性，使其成为大数据智能化应用的最佳场景；

海量消费者与平台的互动行为

买家和卖家的公平交易平台

海量商家在平台进行的商业活动行为

本书将围绕淘宝搜索智能化体系的演进历程进行系统化阐述，如何依托于工程架构体系的逐步完善，逐步实现从简单人工运营加简单算法规则的时代，发展成为阿里电商平台辅助消费者与商品（卖家）的互动更加趣味化和效率化的智能中枢，不仅仅可以从海量用户行为数据中寻找行为规律，结构化行为序列，并从规律中预测结果，更重要的是给出有效的流量中心化和去中心化的投放决策，从而实现消费者，卖家，平台三者社会福利的最大化。淘宝的搜所和推荐发展到今天，正在从智能的依靠机器学习能力解决业务问题，向更高效的从不确定性中探索目标的学习 + 决策的能力进化。



$x$  & 变量

$\boldsymbol{x}$  & 向量

$\mathbf{A}$  & 矩阵

$\mathbf{I}$  & 单位阵

$\mathcal{X}, \mathcal{Y}$  & 输入与输出空间

$\mathcal{D}$  & 概率分布

$D$  & 数据样本（数据集）

$\mathcal{N}$  & 正态分布

$\mathcal{U}$  & 均匀分布

$\mathcal{H}$  & 假设空间

$H$  & 假设集

$h(\cdot)$  & 假设（学习器）

$\mathfrak{L}$  & 学习算法

$p(\cdot)$  & 概率密度函数

$p(\cdot | \cdot)$  & 条件概率密度函数

$P(\cdot)$  & 概率质量函数

$P(\cdot | \cdot)$  & 条件概率质量函数

$\mathbb{E}_{\sim \mathcal{D}}[f(\cdot)]$  & 函数  $f(\cdot)$  对  $\cdot$  在分布  $\mathcal{D}$  下的数学期望；意义明确时将省略  $\mathcal{D}$  和/或  $\cdot$

$\text{var.}_{\sim \mathcal{D}}[f(\cdot)]$  & 函数  $f(\cdot)$  对  $\cdot$  在分布  $\mathcal{D}$  下的方差

$\mathbb{I}(\cdot)$  & 指示函数, 若  $\cdot$  为真则取值 1, 否则取值 0

$\text{sign}(\cdot)$  & 符号函数, 在  $\cdot < 0$ ,  $\cdot = 0$ ,  $\cdot > 0$  时分别取值  
为 -1, 0, 1

$\text{err}(\cdot)$  & 误差函数

$(\dots)$  & 行向量

$(\dots)^\top$  & 列向量

$\{\dots\}$  & 集合

$|\{\dots\}|$  & 集合  $\{\dots\}$  中元素个数

$\|\cdot\|_z$  &  $L_z$  范数

# 目 录



# 第一章 序言

## 学习目标与要求

### 1.1 算法演进之路

从 pc 互联网到移动互联网，阿里巴巴电商平台一路高歌猛进，数据规模，计算能力都发生了天翻地覆的变化；如图 1 所示，

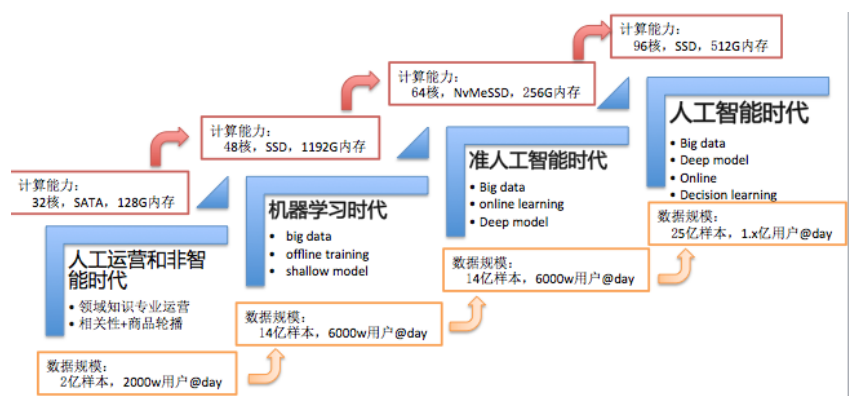


图 1.1: 搜索智能化体系演进图



### 1.1.1 人工 + 弱算法时代

这个时代的关键词：**规则 + 轮播**

算法及模型在搜索和推荐系统领域占据统治地位之前，具有领域知识的专业运营和产品往往充当信息展示规则的制定者，根据主观的判断和对市场的敏锐度来制定查询词背后的商品展示逻辑。“人工规则”的好处是容易理解和操控，坏处则不言而喻，随着平台规模的增大，简单规则无法精细的表达人货匹配的效率，并且容易被一些不良商家利用规则来扰乱市场秩序；实际上，早期的搜索和推荐系统也会运用一些基本的算法逻辑来保证信息匹配的正确性和人货匹配的公平性，基于传统搜索引擎技术的相关性模型，保证用户查询词语商品标题的有效匹配；基于商品成交与否的销售人气指数模型，保证有助于被消费者接受的商品得到更多的展示机会；另外还有一个就是系统为了保证让更多商家有机会得到展现，设置的按照虚拟下架周期为参考的轮播因子，即将下架的商品会得到相对较高的展示机会。

$$score(item) = 1 - \frac{ItemOffshelfTime - QueryTime}{secondsOfTwoweek} \times \left( \frac{docFound}{delta} \right)$$

这个时代遗留下来几个关键问题需要解决：

### 1.1.2 大规模机器学习时代

这个时代的关键词：**big data, offline + shallow model**

随着平台规模的扩大，大规模商家入驻，积极的在平台上打理店铺，发布商品，相对结构化的商品组织体系，类目结构，属性信息，基于商品为主键的销量的累积，评论的累积，这些为更好的理解商品积累了重要的原始数据资料；消费者通过搜索产品的各级页面与平台的互动越来越频繁；数据的组织形成了以人为主键的结构体系，反馈信号也得以在闭环系统中有效的流转；所有的这些都为理解用户积累了重要的数据资料。有效数据的积累为大规模运用机器学习技术解决问题提供了必要的土壤。

这方面各大互联网公司和科研机构，学校公开发表出来的有参考价值的工作有不少，典型的有价值工作，logistic regression, gbdt；

## 1.2 业务问题的思考 @ 淘宝搜索

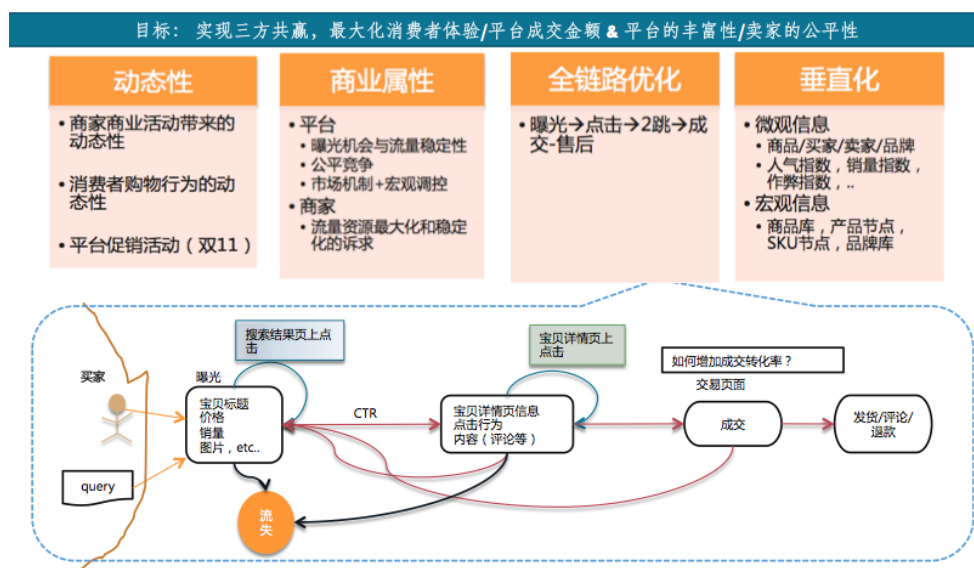


图 1.2: 电商搜索特性

淘宝的搜索平台是致力于提供一个买家和卖家的公平交易平台；作为一个公平的市场调节员，调整供需平衡，为卖家引导潜在的兴趣用户，以提升其 ROI (return on investment)，为用户提供满足其需求 (user intent) 的商品；商业流量下的搜索自然带有其特有的技术特点：

### 1.2.1 动态性

网页搜索的对象的是分布于各类网站发布的网页，从索引单元上对比，数量上是绝对要远远大于商品搜索的对象集合，如果把每个商品展示页当成是淘宝网站的普通网页的化，表象上讲，商品页面的信息集合应该是网页搜索对象

集合的一个子集；网页搜索中的基本对象也存在网页更新，然而淘宝搜索的商品库具有更强的动态性，宝贝的循环搁置，新卖家加入，卖家新商品的推出，价格的调整，标题的更新，旧商品的下架，换季商品的促销，上下架，降价，宝贝图片的更新，销量的变化，卖家等级的提升，商品竞争程度的提升等，都需要淘宝的商品搜索引擎在第一时间捕捉到变化，并及时反映到索引结构中的相应信息单元，而最终的排序环节，这些变化也会动态的融入排序因子，带来排序的动态调整；因此对于商品搜索引擎，要求建立高效的索引更新体系，适应商品类目体系，倒排索引结构，匹配机制的召回逻辑，以及应对商品排序信息及时生效的 cache 分层机制；

## 1.2.2 全链路优化

众所周知，相比类似百度这样的网页搜索平台，一个明显的差异是，淘宝搜索平台拥有网购消费者从查询到完成目标商品订单，这样一条完整的行为数据闭合式链路；因此对于用户的一次查询的满意度衡量绝不能止于搜索结果页上看到一个标题相关的商品而发生了点击来判别，post-click 之后的商品详情页上的行为，甚至于进入 post-pay 之后的评论信息都应该成为度量某商品对于某次查询（query）的满意度影响因子；因此，全链路的行为建模会是淘宝搜索体系相比于网页搜索的重要差异之处；既然谈到这点了，再多啰嗦两句，京东也是一家做电子商务的公司，也有着不小的规模，那么如何来看淘宝搜索与京东搜索在全链路优化上的差异呢？从京东模式来看，post-pay 环节，由于销售，物流仓储的自营性，可以认为是无差异竞争的；而对于淘宝来说，售后的服务，发货速度，以及纠纷退款等环节是取决于商家与消费者之间的互动来决定的，差异性不言而喻，因此淘宝搜索有必要建立 post-pay 环节的排序度量因子；

## 1.2.3 商业属性

电子商务平台的搜索自然具备商业流量的根本属性，商家希望所经营商品通过得到足够的曝光而带来成交；因此，流量资源（曝光）也就成了商家必争之地。搜索排序体系的白盒化和可解释性自然是至关重要。淘宝搜索的 ranking,

更接近于一个带约束的优化问题，而不是一个简单的排序，优化的目标是最大化平台的成交金额；而约束则是卖家流量分配的诉求；这个环节的涉及到的课题也是电商平台最复杂之处，我会在下面集中阐述下我的一些观点；

### 1.2.4 垂直化

电子商务搜索属于 vertical search 范畴，相比于网页搜索，对于平台上内容的结构化梳理，以及商业平台上积累的买家，卖家和商品关系数据的挖掘都有更高的要求；因此需要建立 micro analysis 和 macro analysis 双位一体的搜索内容加工体系，宏观分析层面指的是：除了目前已经积累并广泛运用的 5 级类目之外，完善的商品库建设，spu 节点，sku 节点，品牌库等，都是必不可少的；微观分析层面则从商品的人气指数，销量指数，作弊指数等角度给出商品自身质量的度量信息；使得搜索结果能够为消费者提供，不仅仅停留在标题相关层面的服务，可以通过合理的宏观分析带来的数据结构化，实现高效的结果查询，通过细致的微观分析，保证优质的商品优先展示给消费者；

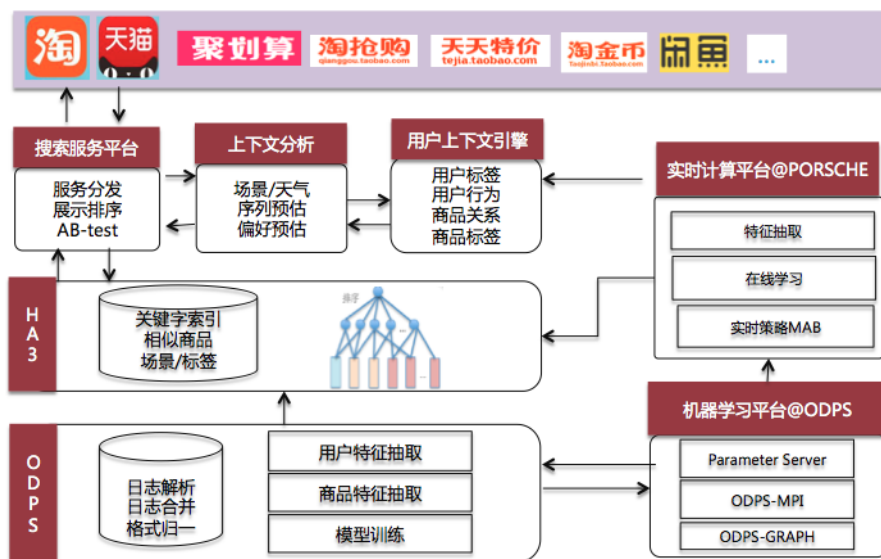


图 1.3: 智能化助力商业产品

## 1.3 技术挑战 @ 业务问题

### 1.3.1 算法模型

像众多互联网企业一样，大数据环境下，基于用户行为建模所面临的技术挑战很多，大家耳熟能详的点列举如下：

投放逻辑带来的数据 bias 对行为建模的影响

用户行为数据的稀疏性

因果关系的模糊性

用户行为的时效

行为个性化和非个性化 unified ranking

Cold start modeling

多样性与精确性的 tradeoff（过度个性化）

长短期个性化融合

### 1.3.2 工程技术

随着数据规模的指数级增长，完成复杂数据建模对于工程技术体系的挑战也是不言而喻：

千亿行为和关系数据存储、实时更新和查询

翻页陷阱

Cache 机制

分级实时体系 (数天/小时/秒/ms)

### 1.3.3 效果评估

效果评估是保证体系迭代朝正向发展的关键保障。

模型正确性评估

AB 体系下分群评估

社会化评测

## 参考文献

- [1] Bilinear+LinUcb 的个性化主题推荐, <http://www.atatech.org/articles/67847>
- [2] 依托搜索技术的个性化平台之路, <http://www.atatech.org/articles/13748>
- [3] 用户意图预估之实时意图篇, <http://www.atatech.org/article/detail/12636/152>
- [4] 知人知面需知心——论人工智能技术在推荐系统中的应用, <http://geek.csdn.net/news/detail/112318>
- [5] Google, Ad Click PredictionL a View from the trenches. pCTR 使用 LR, 通过 FTRL Proximal 算法实现在线模型更新, 频率学派, 写的很细致, 也有工程细节
- [6] Bing, Web-Scale Bayesian Click-through Rate Prediction for sponsored Search Advertising in Microsoft 's Bing Search Engine. Online Bayesian Probit Regression, 贝叶斯学派, 涉及采样算法的模型
- [7] Facebook, Practical Lessones from Predicting Clicks on Ads Clicks on Ads at facebook. DT+LR. 和 GBDT 非常类似, 不同之处在于用 LR 重新训练了每棵树投票的权重, 人气很旺的 xgboost, 在这一块也是做了优化, 利用二阶导数信息得到更快收敛的步长。缺点是处理不了高纬度特征, 处理连续值特征有优势。

- [8] 我所经历的大数据平台发展史（三）：互联网时代 • 上篇, <http://www.infoq.com/cn/articles/the-development-history-of-big-data-platform-paet02>,
- [9] Fast and Reliable Online Learning to Rank for Information Retrieval, <https://khofm.files.wordpress.com/2013/04/thesis-katja-hofmann-online-learning.pdf>
- [10] Dawei Yin, etc., Ranking Relevance in Yahoo Search, KDD'16
- [11] C.J.C. Burges, FromRankNettoLambdaRanktoLambdaMART: An overview, Technical report, Microsoft Research 2010
- [12] Z. Cao, T. Qin, etc., Learningtorank: from pairwise approach to listwise approach, ICML'07
- [13] A. Dong, Y. Chang, etc., Towards recency ranking in web search. In WSDM'10.
- [14] AI 在双 11 中的个性化搜索和决策实践
- [15] 电子交易欺诈层出不穷，如何用深度学习系统步下天罗地网
- [16] 场景驱动结合软硬创新
- [17] 一文综述所有用于推荐系统的深度学习方法
- [18] 卷积网络视角下的大信息过滤理论
- [19] A/B 测试那些事
- [20] 文本分类的选择：词袋模型 vs. 深度序列模型
- [21] 深度报告：“数据革命”终极方向是人工，金融/汽车最快落地
- [22] 当机器学习遇上复杂网络



[23] 机器学习在工业应用中的新思考

[24] 深度学习在搜索和推荐领域的应用

[25] 知人知面需知心-人工智能技术在推荐系统中的应用

## 第二章 系统架构

### 学习目标与要求

### 2.1 泛搜索框架架构图

对于一个完整的泛搜索框架，是一个复杂系统，1. 狭义层面的搜索引擎，涉及到离线的数据 schema 化，索引构建，召回和排序模块；

1. 1. hippo: 调度系统，用于分配机器
2. 2. suetz\_ops: 管控平台，用于启动引擎与分发任务, 详细介绍见第十章
3. 3. suetz: 用于执行 suetz\_ops 4. deployexpress : 5. build service: 用于处理文档和构建索引
5. 6. swift: 用于实时消息的传递
6. 7. cm2: 用于服务的发布，外界可以通过 cm2 定位 qrs 与 searcher

ha3 引擎通常是由一组 Qrs 和 searcher 组件，ranker 组件，summary 组件组成的：

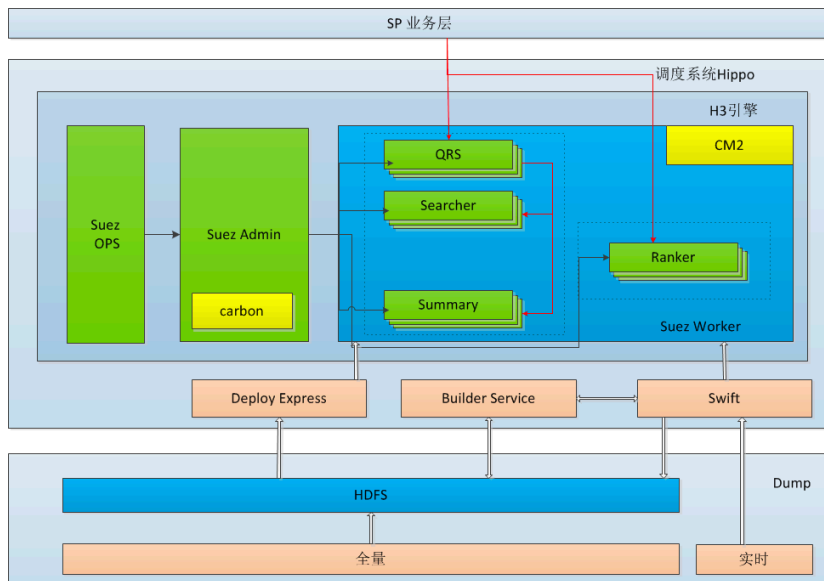


图 2.1: 20210708121500

1. . Qrs 的作用是:对输入的查询做解析和校验,通过后把查询转发给 searcher, 收集合并 searcher 返回的结果, 最后对结果进行一些加工并返回给用户;
2. . searcher 组件: 基本对应的是文档的召回任务。
3. . ranker 组件: 打分和排序服务或者。
4. . summary 组件: 文档的摘要服务。

三个数据流: 1. 查询流:

1. a. 当用户提交一个查询时, 通常会经过 sp 业务层。sp 层主要处理业务相关的逻辑, 例如访问 QP 对查询进行扩展纠错, 访问 igraph 获取相关的个性化信息等。经 sp 层处理后查询已经被处理成引擎所认识的查询串,
2. . SP 从 cm2 中拿一台 qrs 机器, 把查询转发到 QRS
3. . Qrs 根据内部的 cm2 信息, 把查询转发到相应的 searcher

4. . searcher 根据查询，返回结果给 qrs
5. . qrs 合并多个 searcher 的结果返回给 sp，再由 sp 返回给用户

2. 控制流这里的控制流指包含：a. 服务的启停 b. 索引和配置的分发与切换

a. 用户在 `suez_ops` `HA` `cm2` `suezadmin`

3. 数据流 searcher 需要的数据通常索引目录或实时文档，这个索引通常由 `build service` 定期生成,实时文档则通过 `swift` 获取。`suez_ops` `buildservice`

`build service` 除了构建全量索引后，还会定期的构建增量索引。searcher 的实时数据则直接读取 `processor` 写到 `swift` 上数据：

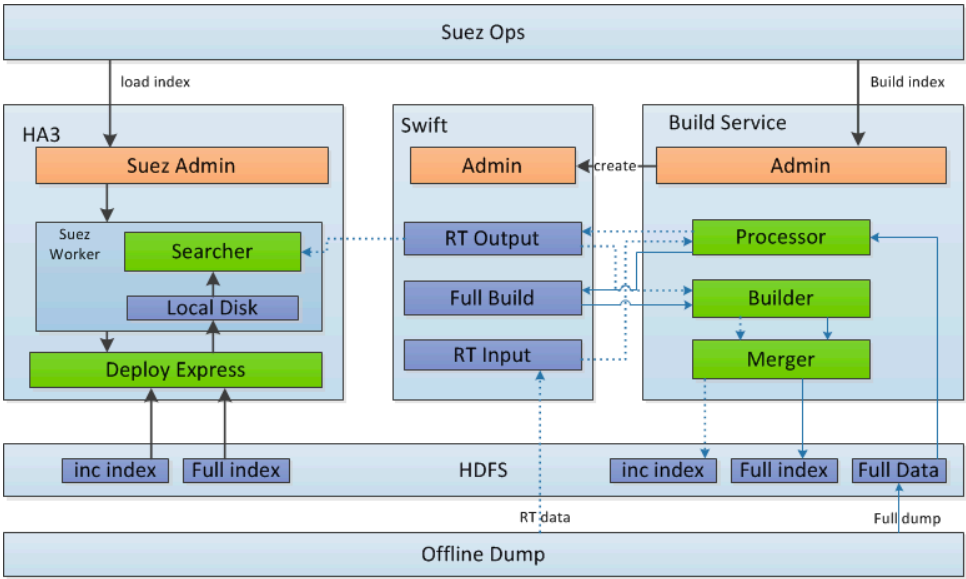


图 2.2: 20210708122651

Suez Turing 框架介绍

suez turing 中把 suez 的 worker 抽象成了一个 local 的执行引擎，把检索流程抽象成图的执行，其中图中节点是各种算子，边是数据，远程 worker 的调用抽象成远程算子图，目前 suez turing 的 worker 只支持 tensorflow 的图执行，未来可能会加入更多的图执行引擎；各个应用整体流程用一张全图来描述，在 suez

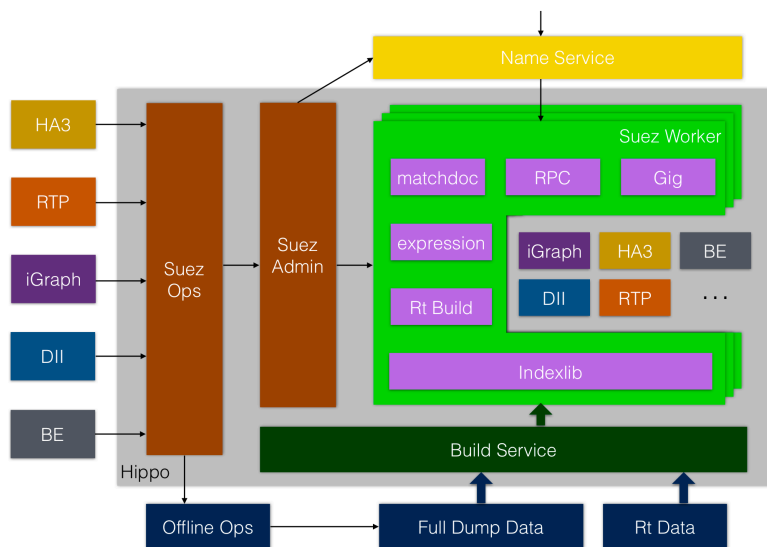


图 2.3: 20210708154912

ops 那边图进行切分，然后分配子图到各个 worker 进行加载，其带来的好处是，采用算子级别的复用，减少相似算子的重复开发，通过图画的架构，达到流程的灵活定制；算子执行的并行化，异步化可以降低 latency，统一的插件机制为算法开发提供统一的接口；codegen 等技术的应用可让更多的业务受益；小业务的全图 local 执行可减少服务的调用；

suze turing 的服务接口

采用 arpc 的 rpc 框架，其 pb 的定义接口如下，主要传输的数据是 tensor

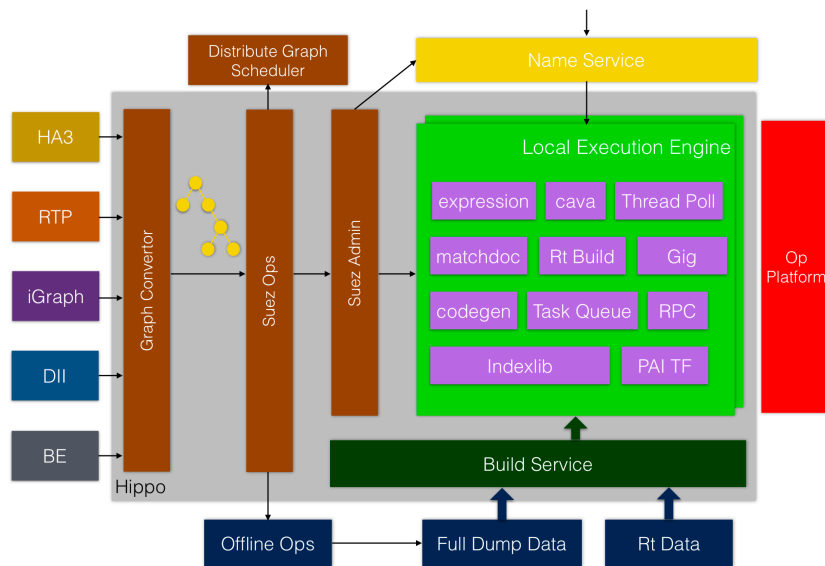


图 2.4: 20210708155805

## 参考文献

- [1] C. Burges, T. Shaked, etc., Learning to rank using gradient descent. In Proceedings of the 22nd international conference on machine learning, ACM

## 第三章 搜索词相关性的技术

### 学习目标与要求

淘宝的平台上有数十亿的商品，消费者在平台上想要快速找到自己想买的商品，只能在淘宝搜索输入查询词，也就是我们通常说的 query，来表达购物的需求。如果能够理解用户 Query 背后的购物意图，就能够帮助搜索引擎自动将符合用户意图的商品返回给用户，提升结果的准确率，从而提高用户在平台上的购物满意度和体验。

### 3.1 类目预测

#### 3.1.1 Query 类目预测与分析

用户搜索意图的理解在搜索排序体系下有着重要的作用。要理解用户的意图，一部分可以通过用户的关键词文本来理解语义上的意图；另一方面可以通过用户的行为积累来获得用户的潜在需求意图，即基于用户行为的意图预测 □。

文本意图中的类目意图预测是淘宝搜索相关性中的重要组成部分。商品在关键词索引召回之后，在第一轮海选粗排阶段通过类目相关性，可以优先选择更相关类目的商品进入第二轮精排中。一方面保证排序的效率，使得排序在类目相关的商品集合上进行；另一方面从最上层保证类目的相关性，保证用户的体验效果。



Query 类目预测主要目标是，分析用户的搜索 Query 和哪些类目的意图更相关。Query 类目预测不同于商品类目预测和一般的文本分类问题，主要是因为 Query 带有的描述信息比较少，而且往往意图比较分散，也就是对于一个用户搜索的 Query 会有多个可能相关的类目，对预测意图的难度会比较大。例如上图用户搜索“电脑显示器”，其中转接线就属于与用户意图不相关的类目商品。

图 3.1

### 3.1.2 线上模型实现

目前线上的版本模型主要包括点击模型和先验模型两部分。

#### 3.1.2.1 点击模型

用户对于搜索的 query 的点击商品行为，很大程度上反应出类目的相关，即点击越多的类目，越有可能和 query 的意图在类目维度更相关。

所以点击模型主要依赖于搜索词的历史行为，也就是 Query 在各个召回商品的类目下的历史一段时间的点击行为做了统计，并且根据各个类目的点击分配比例，通过分数阈值的规则划分类目相关与否的档位。

#### 3.1.2.2 先验模型（新）

先验模型主要为了解决点击模型带来的马太效应问题，相似类目的点击行为为差异和新发类目的商品冷启动问题。

目前依赖的方法主要是是通过 Query 在类目下的召回结果数以及在类目下商品的占比等因素，相当于商品体量的一个估计。两个模型之后进行融合，通过规则的方式确定相关和不相关的档位阈值。

#### 3.1.2.3 在线实现

每天通过搜索日志中选择一段时间内的中高频 Query，通过上述统计规则方法得到 Query 的类目预测结果，并存储为离线词典。（在线访问时，主要依赖于这份基础数据词典。对于不在词典中的偏长尾的 Query，则通过线上丢词的长尾类目预测逻辑进行识别，将在另一篇文章中介绍）

### 3.1.3 存在的问题与分析

基于这种融合模型的方法，点击模型带来更准确的类目相关度量，而先验模型对于行为稀疏的类目可以得到召回上的提升，使得线上的相关性效果得到一定的保障。但是在应用中还是存在着以下几个问题：

首先是点击模型，统计历史的类目点击数据对于曝光较多的类目会占优势，对于类目商品体量不均的情况，会使得点击分数更偏向体量大的类目；而且如果存在类目新发商品或新增类目的情况，即使点击数据在近日内有增量，但是也很难在统计值上追上已有类目，所以需要有一个根据增量预估整体的方法。其次，在先验模型中的一个重要假设，就是召回的商品越多，越可能是相关的类目，其实并没有考虑语义上是否能保证相关，例如：“茶几”这个 Query 可以召回“纸巾盒”类目的很多商品，但其实文本语义上这两个意图并不相关。所以先验模型中对于召回的类目，要做语义上的判断和区分。最后，通过人工经验的规则方式将两者融合，得到最终的类目判别结果。而随着商品分布的和用户行为的变化，固定的阈值方法难以满足线上的准确率要求，需要有更合理的综合特征的方式。

基于以上的问题，我们提出了基于 DNN+GBDT 的类目预测融合模型，对其中的问题进行优化，并得到了准确率的效果提升。在第二部分中会详细介绍每个部分。

### 3.1.4 基于 GBDT 融合模型的类目预测

#### 3.1.5 基于反馈的模型

##### 3.1.5.1 点击模型（新）

类似于排序中 ctr 预估的方法，通过 Query 和类目下的商品的历史点击行为，预测 Query 和类目的未来的点击行为。主要使用 Query 历史 7 天、15 天的曝光、点击等统计特征。

考虑到前面提到的第二个问题曝光带来的不公平点击差异，所以在回归的目标上加上了该类目下展现 pv 的正则项，使得本身展现高的类目会在未来的展现比例得到一定的弱化。

##### 3.1.5.2 价格 & 成交模型

类目之间的商品的价格差异，在区分类目意图的差异上有着重要作用。

例如，用户搜索“手机”，自然召回的商品有主件“手机”类目和配件“手机保护套”类目，我们可以通过 query 下的成交价以及各个类目的商品价格加入到特征中。

所以我们建立了一个基于成交和价格的回归模型，同样利用历史的统计信息作为特征主要包括 □

- query 在类目下的成交价格
- 类目下商品的价格
- 类目的成交金额
- query 的成交均价等

其中的类目下商品价格，为了排除掉过高过低的商品价格，我们在模型中使用在一段时间窗口内，该类目下成交商品价格平均值。

### 3.1.5.3 基于 DNN 的先验模型 (Deep Prior Model)

首先，对于 Query 能召回商品的类目的集合可以作为相关类目的一个较小候选集合；然后，对于这部分类目，需要计算词构成的语义的匹配程度的分数。在计算语义的匹配模型中，我们采用了多层神经网络的方法，通过词的 embedding 方式表示 Query 和类目，然后通过 Query 和类目的采样做目标，来训练这个网络。网络结构图如下：

### 3.1.5.4 特征说明

- Query Embedding: 用每个词 id 的 embedding 做组合，其中加入了每个词的统计权重和意图权重。词的统计权重主要为词在 Query 日志中的搜索 pv；意图权重主要为词的标签信息设置的人工权重，例如品牌词、品类词、型号词可能在意图中有更高的权重。最终词的权重表示为  $\log(pv) * tag\_weight$ 。

图 3.2

- Category Words Embedding: 每个类目下的词, 计算  $tf \cdot idf$ , 这里的  $idf$  计算将每个类目作为一个  $doc$ , 即在越多类目中出现的词, 约不重要。选取每个类目下的最高权重的词来表示类目, 并进行带权的  $embedding$  计算 (全连接)。
- Cate Id Embedding: 类目  $id$  直接做  $embedding$

- Cate State Feature: 主要为类目下商品的平均成交价等连续值统计特征。

### 3.1.5.5 样本 & 采样

样本为 Query 和类目的 pair 对，主要来源于以下几个部分：

- 当前类目与该类目路径名称（其中的词作为 Query）为正样本；当前类目和与该类目具有相同父类目（只向上一层）的叶子类目的名称为负样本。
- 对于有行为的类目，每个类目下随机抽取有行为商品，对每个商品，获取其 ctr 高的 Query 作为正样本；同时，这部分 Query 可以召回的非该行业或一级类目的商品类目为负样本。
- 对于没有行为的类目和商品，对商品标题中的词做随机采样，作为正样本。此时选择组成 Query 的词数目有限制，选择过短的随机 Query 会带来很大的歧义。同时和前面的方法相同，生成的 Query 召回的非该行业或一级类目的商品类目为负样本。
- 基于少量人工规则的采样，基于类目之间的互斥关系，Query 的正样本类目对应的互斥类目为负样本。

### 3.1.5.6 总结 & 思考

- 为什么使用点击的数据？

点击的样本在准确性上有较高的保证，而且从数据的观察发现，每个类目下的高点击的 Query 并不一定是和类目相关，但是 Ctr 高的 query 往往与类目的关联性很大。而且由于我们丰富了类目的表示，即类目用词来表示的同时也增加了一定的泛化性，也就是说对于词表达比较相似的类目，即使自身行为并不丰富，也可以通过与其表达相似的类目来学习得到和 Query 的关联性。

选择类目下各个商品的 Query 与选择类目整体的 Query，前者可以带来更丰富的特征，后者可以在统计的丰富上保证准确性。

- 为什么不使用多分类目标的模型？

最开始设计方案时其实调研过多分类的方案，最后还是选择二分类的框架主要原因有两个：首先 Query 的类目预测不同于商品，往往会存在多个合适的类目预测结果，所以如果采用 Softmax loss 的多分类，则会由于 loss 本身的限制，难以使得多个类目同时为正，就算是整体的平均 loss 最低，对于多 label 的样本中的 loss 也是比较高；另外，最重要的一点是先验模型的计算是可以预先知道候选类目的，也就是通过召回限制，可以得到 Query 的候选类目在一个小的集合范围中，相比多分类的全 label 空间，要缩小了很多倍 (15000->50)，而且如果以类目作为特征而不是 label，可以得到很多类目的统计以及描述类特征。

### 3.1.6 基于 GBDT 的 Ensemble 模型

前面所述分别从行为反馈和商品分布得到不同维度的相关性描述，在线上版本的模型中对于各个特征采用分数的规则来决策得到最终的相关类目。我们改进了原有通过人工经验的方式设置相关性分档的规则，通过人工标注的样本，将多个特征通过 GBDT 模型做最后的决策分档模型。整体图如下：目前在 GBDT 的特征包括：

- 先验模型分数 (Deep Prior Score)
- 点击分数、价格 & 成交分数，在 Query 下所有类目的归一化、比例等。
- Query 召回类目下的商品数、占 Query 的商品数比例、占类目下的总商品比例。
- Query 分词长度、Query 召回的类目总数（描述宽泛性的 Query）
- 类目所属的一级类目、行业等



图 3.3

### 3.1.7 模型效果与分析

#### 3.1.7.1 数据效果 & 页面效果

新老版本效果对比，Query 搜索“电脑显示器”，类目预测的结果对比（上图为老版本，下图为新版本）

在页面 BTS 的商品类目对比效果：



图 3.4

### 3.1.8 指标效果

先验模型的训练中用人工标注的样本作为 validation 集合，通过网络参数以及样本的分布调整，模型在验证集合的 auc 可以达到 0.8。

人工评测整体模型时会根据算法产生的类目档位相关 (2 档)、不相关 (1 档) 进行准确率和召回率的判别，主要关注的是 2 档类目的准确率和召回率。样本抽取时为了更好的暴露各个行业中可能存在的问题，采用分别对每个行业下按照

图 3.5

Query pv 分布进行采样的方法。在评测样本中，新方法相比老的方法在 2 档准确率上有 10% 左右的提升（68.51%→78.75%），召回率略有下降 1.61%（60.98%→59.37%）。（因为评测采样的数据样本是每个行业的覆盖量在同样的量级，所以对于较难的长尾行业的数据也会放大，相比线上真实的流量比例，是偏难的一种评测方法）。目前整体的模型数据已经在主搜 PC 开始 BTS。

从效果看对于覆盖率的提升空间还比较大，主要源于一些宽泛 Query 例如

图 3.6

搜索品牌、宽泛描述的品类等问题，对于子品类类目的覆盖率需要在先验模型的采样优化中多加考虑；而且在一些长尾行业的类目上的预测准确率还是需要提升，因为行为比较稀疏，需要更多的商品分布特征。

后续的优化空间主要在先验模型的部分和整体模型的调优，现在只有基于 DNN 的网络，后续可以考虑更有局部特征能力的 CNN 和全局性表示的 LSTM 作为 Query 和类目的表示。目前由于受限于人工标注样本的数量限制，没有办

法做到 end to end 的整体框架模型，这个也是后续值得优化的一个方向。

## 3.2 基于类目树结构的 query 长尾类目预测

### 3.2.1 背景介绍

#### 3.2.1.1 淘宝的类目树体系介绍

淘宝上有上亿的商品，通过类目来管理这些商品。类目就是商品分类，是商品信息的一种结构化描述，目的是为了管理、导购。淘宝的类目是树状结构的，如下图。一个类目可以细分为更多的下一级类目。类目 id=0 的类目叫根类目；没有下一级类目的类目叫叶子类目；没有上一级类目的叫一级类目（实际上，一级类目的父类目是根类目，但是根类目是大家不可见的虚拟类目，因为可以认为没有）；有下一级类目的类目是下一级类目的父类目；有上一级类目的类目是上一级类目的子类目。如图为例：“电烤箱”、“电饭煲”、“电压力锅”、“电蒸锅”、“微波炉”等就是叶子类目；“厨房电器”就是一级类目；“厨房电器”是“电烤箱”、“电锅煲类”、“微波炉”的父类目；“电烤箱”是厨房电器的子类目；大家也常常叫一级类目的子类目为二级类目，二级类目的子类目为三级类目，依次类推。淘宝类目树体系的数据一般可以通过 `tbcdm.dim_tb_cate` 表获取。

#### 3.2.1.2 基于类目树的类目预测

query 的类目预测是淘宝搜索相关性的重要组成部分，通过类目预测，可以把与用户搜索 query 意图相关的类目对应的商品靠前呈现，从而保证用户的体验效果。

对于高频的 query，淘宝已经积累了用户的一些行为数据和统计信息，通过这些数据 [建立模型](<https://www.atatech.org/articles/85571>) 往往可以取得较好的预测效果。但对于低频的 query，由于缺乏足够的反馈数据，只能通过 query 本身的语义来推测 query 意图相关的类目，本文将低频 query 的类目预测定义为长尾类目预测。

图 3.7

长尾类目预测问题,是一种 [multi-label 的分类问题](<http://ieeexplore.ieee.org/document/6>)  
可以考虑转为 multiclass 的方式来处理。之所以将 multilabel 问题转为 multiclass 问题是基于如下考虑:

- multilabel 的标注数据比较难获取, 对于一个 query, 要罗列出它所有相关的类目很困难, 现有 top query 的 2 档类目预测准确率也才 80%;
- multilabel 大多数模型是把二档类目都当做同样重要的 label 来处理, 但由

于行业下商品、用户行为差异，一个 query 对应的 2 档类目分布应该有主次之分，例如” 连衣裙” 在” 女装→ 连衣裙” 类目应该占大多数，在” 大码女装” 类目下应该占少数。

传统处理多分类的方式利用 softmax 得到每个节点的概率，对应淘宝的类目预测即只预测 \*\* 叶子类目 \*\* 出现的概率，这种方法存在着三个严重的缺陷：

- 淘宝叶子类目存在着上万个，直接预测上万个叶子类目对应的概率，一方面每次需要对上万个节点计算对应的值，复杂度较高，另一方面 softmax 函数在 top 1 的概率区分度较好，但 top3-5 的类目往往概率值都很接近
- 没有利用淘宝类目树的体系，相当于只取了类目树叶子节点的信息进行分类
- softmax 划分两档类目难以确定阈值，top K 的处理方式对处理宽泛 query 或明确意图 query 都不准确

为了合理利用淘宝类目树的信息，我们提出了基于类目树的层次分类 (以下简称层次分类) 方法，其效果如下图，对于 query” 实木欧式床”，我们先预测其到一级类目” 住宅家具”、” 家装主材”、“尿片/洗护/喂哺/推车床” 的概率，然后逐层传导，得到对应每个叶子节点的概率。” 实木床” 的概率为 0.8260 是按如下公式计算的，即将概率值逐层传导：

$$P(\square\square\square|\square\square\square\square) = P(\square\square\square\square|\square\square\square\square) \times P(\square\square|\square\square\square, \square\square\square\square) \times P(\square\square|\square\square, \square\square\square\square)$$

基于类目树的层次分类方法相比 softmax 有着明显的优点：

- 层次分类可以直接得到各个父节点的概率
- 层次分类做预测时计算量会小很多，可以考虑树分层遍历贪婪地选出出现可能性大的类目，例如在预测一级类目时直接把概率值较低的类目的分支就不进行预测了。如果贪婪的方式选取每个父节点 top 的子节点进行预测，层次分类只需要对至多  $3^6 = 729$  个叶子节点排序取 topK, 而 softmax 则需要对 1 万多个节点排序取 topK

图 3.8

- 层次分类的准确率会优于 softmax 分类，层次分类每次对几十个类目预测相应概率，而层次分类是一次对一万多个类目预测概率。实验的效果也证明层次分类预测到叶子类目的准确率比 softmax 高 5%。

本文接下来几章将重点介绍层次分类，具体如下：

- 第二章主要介绍下业界常用的层次分类模型
- 第三章主要介绍基于类目树的层次分类的模型及公式推导
- 第四章主要介绍淘宝长尾类目预测模型的整体框架及相关实验总结

### 3.2.2 相关研究

大规模层次分类领域，业界有很多研究，主要的研究方向在文本分类、蛋白质结构识别等领域。在 [A Survey of Hierarchical Classification Across Different Application Domains](<https://link.springer.com/article/10.1007/s10618-010-0175-9>) 中介绍了常用的层次分类方法，

主要有三大类：

1. 平面分类
2. 局部分类器
3. 全局分类器

### 3.2.3 平面分类 (Flat classification)

如下图示例，不考虑类别层次，将类别树种所有叶子节点看做相互独立的平级类别，作为一个多类别分类问题 (multi-class) 处理，一般使用 softmax 函数预测各个叶子的概率，选 top K 作为预测结果。常见的文本分类、手写数据识别都是使用 softmax 做分类。

图 3.9

### 3.2.4 局部分类器 (Local-model hierarchical classification approaches)

局部的分类器的方法又细分三类

1. 预测每个节点
2. 每个母节点建立分类器
3. 逐层分类

#### 3.2.4.1 预测每个节点 (Local classifier per node approach)

如下图，对每个节点 (叶子和非叶子) 建立二分类模型预测，得到所有预测为 1 的节点，然后遍历树结构找到从根节点到叶子节点全为 1 的路径，这些路径对应的叶子节点就作为输出的 label。如果层次结构中有很多节点，这种方法训练的分类器将非常多，正负样本的采集工作量也是巨大的。



图 3.10

#### 3.2.4.2 每个母节点建立分类器 (Local classifier per parent node approach)

如下图，基本想法就是给每个非叶子节点建立分类器，想法比较直观，本文的层次分类模型主要借鉴该方法。该方法的缺点是无法用在有向无环图的结构里。

图 3.11

#### 3.2.4.3 逐层分类 (Local classification per level approach)

如下图，逐层建立分类器进行。这种方法在近几年的论文中结合神经网络使用较多，如 [Hierarchical Classification of Gene Ontology-based Protein Functions with Neural Networks](<http://pdfs.semanticscholar.org/0a0c/8c6eab6152910da4165688b2352cc82>)

就是使用这种方法改进的网络, 第一层的预测结果作为特征加入第二层中训练, 每层的激活函数都是 logistic, 通过卡一个阈值来输出 multi-label。

图 3.12

图 3.13: Hierarchical Classification of Gene Ontology-based Protein Functions with Neural Networks

#### 3.2.4.4 全局分类 (Global or big-bang classifier)

根据整个类别层次学习一个分类模型。如 [Decision Trees for Hierarchical Multi-label Classification](<https://link.springer.com/article/10.1007/s10994-008-5077-3>) 就通过决策树来学习一个全局模型。Cai 等人基于 SVM 构造层次分类方法, 利用类别层次信息构造判别函数, 由 SVM 模型计算文档在某个类别以及该类别

所有祖先类别上得分，然后将这些得分加权作为文档最终得分，以此判别文档类别。

图 3.14

### 3.2.5 本文层次分类模型介绍

本文层次分类层的实现方法类似于 word2vec 中 hierarchical softmax 的方法，计算从根节点到叶子节点这条路径的概率，路径的概率是按每个非叶子节点走到其子节点概率之积，即损失函数是取 \*\* 负对数似然函数 \*\*。

层次分类模型中主要用到了树结构的概率逐层传导，我们通过对每一层建立判别模型可以简化层次分类的计算。假设 query 输入数据传入的为 K 维的向量  $X$ ，对某一父节点  $C_{\square}$ ，其子节点为  $C_{\square i}, i \in \{1, 2, \dots, M\}$ ，则应该满足概率  $\sum_i P(C_{\square i} | C_{\square}, X) = 1$ 。从  $C_{\square}$  到  $C_{\square i}$  计算概率是一个明显的多分类问题，我们可以用 softmax 函数做分类来处理，只是这里的 M 比较小，一般是 2-20。

#### 3.2.5.1 基本定义

为了方便说明，我们定义一些符号，

1. query 输入为 K 维的向量  $X$
2. 设某条选中的类目全路径 (从根节点到叶子节点) 长度为  $l_c$ .

3. 设对应的类目树全路径节点为  $c_0 \square c_1 \square \dots \square c_{l_c}$ , 其中  $c_0$  为根节点,  $c_1$  为行业,  $c_{l_c}$  为叶子节点, 一般  $l_c$  不大于 7.
4. 每个  $c_i$  对应的子节点数定义为  $N_{c_i}$ , 其中  $i \in \{0, 1, 2, \dots, l_c - 1\}$
5. 每个  $c_k$  对应父节点的位置为  $I_{c_k}$ , 其中  $k \in \{1, 2, \dots, l_c\}$ , 且有  $0 \leq I_{c_k} \leq N_{c_{k-1}} - 1$
6. 每个非叶子节点对应的向量为  $\theta_{c_i} \in R^{N_{c_i} \times K}$ , 其中  $i \in \{0, 1, 2, \dots, l_c - 1\}$ ,  $\theta_{c_i}^{(t)}$  表示  $\theta_{c_i}$  第  $t$  行的向量

针对  $i=1 \square 2 \square \dots \square c_{l_c}$ , 我们用 softmax 得到每个节点  $c_i$  的条件概率, 公式如下 (假设  $b$  相同):

$$P(c_i | X, c_{i-1}) = \frac{\exp(\theta_{c_{i-1}}^{(I_{c_i})} \times X)}{\sum_{s=0}^{N_{c_{i-1}}-1} \exp(\theta_{c_{i-1}}^{(s)} \times X)} \quad (3.1)$$

### 3.2.6 损失函数定义

层次分类的损失函数类似 word2vec, 用 \*\* 负对数似然函数 \*\*. 我们把 multi-label 输入的样本转为 multiclass 的样本作为训练, 即每次传入的数据是 (query, cate) 这样的单样本, 对每个样本而言, 其损失函数的计算公式如下:

$$L = \sum_{i=1}^{l_c} L(c_i) = - \sum_{i=1}^{l_c} \ln P(c_i | X, c_{i-1}) = - \sum_{i=1}^{l_c} \ln \frac{\exp(\theta_{c_{i-1}}^{(I_{c_i})} \times X)}{\sum_{s=0}^{N_{c_{i-1}}-1} \exp(\theta_{c_{i-1}}^{(s)} \times X)}$$

$$L(c_i) = \ln \left( \sum_{s=0}^{N_{c_{i-1}}-1} \exp(\theta_{c_{i-1}}^{(s)} \times X) \right) - \theta_{c_{i-1}}^{(I_{c_i})} \times X$$

#### 3.2.6.1 前向传播计算

前向传播主要是计算 top 的 Loss 值, 可以根据 Loss 的公式由全路径节点为  $c_0 \square c_1 \square \dots \square c_{l_c}$  计算得到

### 3.2.6.2 反向传播计算

反向传播类似于[word2vec](http://blog.csdn.net/itplus/article/details/37969979)中推导的方式，求出对  $\theta \square X$  的偏导。

以下偏导的计算公式是对单个训练样本而言的，如果是 batch 训练，则通过循环对每个样本更新反向传播值。

### 3.2.6.3 非路径上的 $\theta$ 偏导

非类目全路径上的偏导为 0

### 3.2.6.4 关于路径上 $\theta$ 偏导

对于每个  $\theta_{c_i}^{(t)}$  而言 (其中  $i \in \{0, \dots, l_c - 1\}$ )，关于 L 的偏导只和  $L(c_{i+1})$  有关，故偏导 K 维向量如下：

- 当  $t \neq I_{c_{i+1}}$  时偏导为

$$\frac{\partial L(c_{i+1})}{\partial \theta_{c_i}^{(t)}} = \frac{\exp(\theta_{c_i}^{(t)} \times X)}{\sum_{s=0}^{N_{c_i}-1} \exp(\theta_{c_i}^{(s)} \times X)}$$

- 当  $t = I_{c_{i+1}}$  时偏导为

$$\frac{\partial L(c_{i+1})}{\partial \theta_{c_i}^{(t)}} = \frac{\exp(\theta_{c_i}^{(t)} \times X)}{\sum_{s=0}^{N_{c_i}-1} \exp(\theta_{c_i}^{(s)} \times X)} - X$$

### 3.2.6.5 关于路径上 X 偏导

可以先求关于每个  $L(c_i)$  的偏导

$$\frac{\partial L(c_i)}{\partial X} = \sum_{s=0}^{N_{c_{i-1}}-1} \frac{\exp(\theta_{c_{i-1}}^{(s)} \times X)}{\sum_{s=0}^{N_{c_{i-1}}-1} \exp(\theta_{c_{i-1}}^{(s)} \times X)} \times \theta_{c_{i-1}}^{(s)} - \theta_{c_i}^{(I_{c_i})}$$

$$\frac{\partial L}{\partial X} = \sum_{i=1}^{l_c} \frac{\partial L(c_i)}{\partial X}$$

### 3.2.7 层次分类基本框架

前面介绍了层次分类的损失函数、前向、后向传播的定义，下面我们具体看下层次分类层的实现框架，如下图：

- Hierarchical loss layer(图的左边部分), 主要用于层次分类的模型训练
  - \* 输入 K 维向量 X, 及其 label 为叶子类目 cate\_id, 以及淘宝类目树结构
  - \* 前向传播计算损失函数 L, 是根据传入的 cate\_id 得到对应类目树的路径, 然后计算每个节点对应的损失函数  $L(c_i)$ , 然后对 loss 求和得到  $L = \sum_{i=1}^{l_c} L(c_i)$
  - \* 反向传播则根据类目全路径, 将偏导传给底层, 得到  $\frac{\partial L}{\partial X}$
- Hierarchical predict layer(图的右边部分), 主要用于层次分类的模型预测。模型利用已训练好的层次分类参数及结构进行预测, 使用贪婪的算法处理(例如每个父节点选 top3 的子节点进行计算), 通过树的层次遍历, 可以得到概率值较高的叶子类目, 以此作为输出。

图 3.15

### 3.2.8 前、后向传播优化

我们注意到损失函数  $L$  可以把沿路径计算 loss 问题 (左图) 转化为右图展开形式, 并行计算, 互相不受影响。

图 3.16

### 3.2.9 基于层次分类的 query 类目预测

我们利用层次分类模型来完成淘宝 query 的长尾类目预测, 模型训练的主要框架如下:

\* 输入数据有三部分:

- Query, 用户输入的 query
- Label, 训练时使用的 label 数据
- Tree structure, 淘宝的类目树结构, 用于层次分类模型构建和先验概率计算

\* 特征提取:

- 从输入 Query 提取语义相关的特征 (semantic feature),
- 从淘宝用户历史行为、商品历史分布中提取统计特征 (statistical feature)

- \* 通过神经网络组合特征进行学习，然后用层次分类模型做训练和预测

图 3.17

下面分别从数据和模型详细介绍下我们的工作，类目预测划档的工作不在上面框架图中，我们会在本章第三节介绍下相关工作。

### 3.2.10 训练数据

深度模型通常需要大量训练数据，但对于长尾 query 而言，multilabel 问题中高质量的标注 label 难以获取。我们考虑一些规则的方法，获取训练数据

- \* top Query 的数据。top Query 由于存在用户的行为数据，往往类目预测准确率较高，我们可以根据一些规则，提取 top Query 的训练数据，以此来迁移学习长尾类目预测：
  - 每个商品关联的 top K 的 query
  - 历史 top query 对应的 2 档类目
- \* 商品标题及其对应类目。商品放置的类目是最为准确的 label，虽然标题数据与 query 分布上存在着差异，但可以使用标题的数据预训练，得到词和字符的向量表达
- \* 标题词采样。标题词采样生成 query 由于生成 query 的质量难以保证，准确率较低。



- \* 人工标注积累的一批长尾 query 及其 label。

### 3.2.11 特征提取

query 长尾类目预测可以提取的特征主要有两部分组成:

1. 语义特征。语义特征主要是 query 本身表达的语义, 这可以从词、字符维度通过 DNN/CNN/LSTM 等方式学习 query 的语义表达, 如上面框架图中分别对 query 以词、字符维度做了 embedding, 然后再分别得到 query 的语义表达向量 (128 256 维)。另外语义特征还可以考虑词用 word2vec 预训练, 加入词性、tagging 等特征 (框架图中未画出) 等。
2. 统计特征。统计特征主要是从 query 或词的历史行为数据来获取特征, 主要包括:(a) 词在各类目分布, 互信息;(b) 词在搜索中的 pv,uv,gmv,ctr, 互信息;(c)query 历史的 pv,uv,gmv,ctr;(4) 相似 query 历史的 pv,uv,gmv,ctr;

上面框架图中的特征提取工作如下:

- \* 对 query 归一化后提取语义特征:
  - 中粒度分词进行 embedding 操作 (50w 词表), 每个词 128 维向量表达, 之后通过 DNN/CNN 等方式得到 query 的语义向量 128 维
  - 提取 query 中的字符进行 embedding 操作)(7000 个字符), 每个字符 64 维向量表达, 之后通过 DNN/LSTM 等方式得到 query 的语义向量 128 维
  - 把词维度表达的语义向量和字维度表达的语义向量 concat, 得到 256 维语义向量
- \* query 归一化后提取统计特征, 这里主要考虑通过词类目分布来预估 query 的类目分布
  - 根据一定规则得到每个词的先验分布

- 根据词加权平均得到 query 的类目先验分布，见下面”苹果手机”例子
- 得到 query 在叶子类目的概率分布后，以类目树结构解析得到每个节点的概率
- 将叶子节点和非叶子节点铺平，按概率值降序排，截断 top 300 的节点及其值作为 query 的先验类目特征
- 将 top300 的先验类目特征由稀疏值转为 1.5 万维向量

\* 将语义特征和统计特征做全连，得到组合特征。

> 若词”苹果”先验类目概率分布如: 新鲜水果»苹果:0.5, 手机:0.5 词”手机”先验类目概率分布如: 手机:0.4, 手机壳:0.4, 手机配件:0.2 则 query”苹果手机”对应的类目先验分布为: 手机: $0.5 \times w_1 + 0.4 \times w_2$ , 手机壳: $0.4 \times w_2$ , 苹果: $0.5 \times w_1$ , 手机配件: $0.2 \times w_2$

### 3.2.12 规则划档

query 的特征经过层次分类模型后，我们可以得到 query 在类目树中的概率分布，我们可以贪婪的选出叶子节点概率值最大的 top 20 个类目作为候选类目，在这些候选类目中进行划档。

现阶段划档使用的是规则划档，主要有两个规则：

1. 叶子节点概率大于一定阈值 (如 0.05 概率)
2. 类目全路径下子节点在父节点下概率大于一定阈值 (如 0.15)

如 query “衣柜简约现代经济型宜家”，规则可以方便的将”衣柜”、”简易衣柜”划为 2 档

在实验中划档也考虑了简单的模型版划档，效果和规则版相当，后期有时间可以优化模型版划档

图 3.18

分组	模型	准确率	召回率	F1	备注						
baseline	规则版	0.8194	0.6194	0.7055	规则是子节点在父节点下出现的概率大于 0.15, 叶子节点概率大于 0.05						
组一	xgboost	0.8151	0.5897	0.6844	xgboost 用 1w 个 query 数据训练 (每个 query 10 条记录), 特征是 query 层次分类各层概率						
组一	xgboost	0.8061	0.6253	0.7043	xgboost 用 1w 个 query 数据训练, 特征是 query 层次分类各层概率 + 各层相对前一层概率 + 先验各层概率 + 先验各层相对前一层概率						

### 3.2.13 实验效果及评测

#### 3.2.13.1 模型实验对比

分组	分类模型	特征选用	准确率	召回率	F1	备注					
—	—	—	—	—	—	—	—	—	—	—	—
	基准	baseline	0.5736	0.8634	** 0.6893**	线上现有长尾方法, 作为 baseline					
	分组一	softmax word+char 的语义特征	0.6632	0.5725	0.6145	只考虑使用 word,char 得到的语义特征, 用了 DNN 提取特征, 最后分类模型是直接					
	分组一	softmax word+char 的语义特征 + 统计特征	0.6804	0.5984	0.6368	在上面基础上加了 query 的类目先验分布统计特征, 其他相同					
	分组二	hierarchical word+char 的语义特征	0.7342	0.5765	** 0.6459**	只考虑使用 word,char 得到的语义特征, 用了 DNN 提取特征, 最后分类模型是使用层次分类模型后规则划档					
	分组二	hierarchical word+char 的语									

图 3.19

义特征 + 统计特征 | 0.7911|0.6582| **\*\*0.718\*\***| 在上面基础上加了 query 的类目先验分布统计特征, 其他相同 |

从上表可以看出, 层次分类模型优于直接对叶子类目分类的 softmax 模型

### 3.2.13.2 人工评测结果

人工评测整体模型时会根据算法产生的类目档位相关（2 档）、不相关（1 档）进行准确率和召回率的判别，主要关注的是 2 档类目的准确率和召回率。下面是评测结果，可以看出层次模型准确率较高，但召回率偏低，主要的原因是宽泛 Query 往往只给 2-3 个类目划为 2 档，如 query “耐克魔术贴女鞋 nike”，“女童外套春秋 2017 新款韩版”。

算法	评测数据	准确率	召回率	2 档丰富度
层次模型	按行业比例采样 300 个长尾 query	77.04%	51.03%	1.84
线上长尾算法	按行业比例采样 300 个长尾 query	63.63%	65.59%	2.87
层次模型	按 pv 采样 300 个长尾 query	88.12%	52.98%	1.74
线上长尾算法	按行 pv 采样 300 个长尾 query	63.63%	60.60%	2.66

### 3.2.14 后续计划

1. 完善统计特征的生成逻辑。现有的统计特征计算方式最大的问题会带来大量噪声，如“男拖鞋”query, 词“男”会带引入大量与拖鞋不相关的类目的统计信息
2. 确保模型的稳定性。现有类目预测模型有一个很大的问题是 query 加上一些不相关词后类目预测结果变化较大，或者无法预测。后续考虑在模型中通过 attention 机制确保中心词相关的类目能预测正确。
3. 融入知识型数据到类目预测模型中。例如将上下位宽泛词、同质类目、互斥类目等规则信息融入到模型中。
4. 考虑和 querytagging、相关性等模型一起进行 multitask 的学习

### 3.2.15 导购思考

导购产品包含了底纹、搜索发现、首页分类、下拉等产品，主要功能是在搜索前与搜索中为用户提供便利的搜索引导。因此，为用户提供更加便捷的导购

路线，以及让更多的用户使用导购产品，对搜索用户的增长有一定的促进作用；当用户使用搜索时，他是有一定的明确需求的，而随着推荐类产品的不断发展，如直播、资讯等，老用户的使用习惯也在发生着变化，从之前的用户主动的需求发起，到现在的用户等待信息的供给，随着用户心智的变化，我们在产品上也需要因势利导的做出更满足用户需求的产品。另一方面，随着用户市场的下沉与扩展，在用户增长的前提下，势必会有大量的淘宝新用户进来，这批用户可能对如何使用搜索并不太熟悉，因此在产品方面，我们需要有更加方便易用的产品形态服务用户；

导购算法经过了多年的发展与优化，目前已经在召回、排序以及调控策略上有了较深的积累，这也促使我们不断思考如何在新形势下做出更满足用户需求的算法，从当前所面临的问题出发，我们主要从以下几个方面进行了思考：

1. 多领域联合: 导购产品包含了多个产品形态，而目前这几个产品形态之间是相互割裂的，在样本、特征以及用户反馈上并没有做到信息上的共享与利用，因此如何将这多个领域进行联合起来，作为一个统一的导购整体，并达到  $1+1>2$  的效果，是我们面临的一个主要问题
2. 用户全生命周期表示: 导购产品分别是在用户无输入情况下为他推荐 Query，以及在用户有部分输入情况下进行 Query 补全，因此对用户的全方位的理解也是所面临的一个问题，比如是否新用户、用户的活跃程度，或者是用户即将流失的状态；另一方面，如何积累用户的长期行为，比如一年前的行为，而目前的 *sequence* 学习方法可能会更加强调当前的行为，而遗忘了较长时间前的信息，所以如何基于 *lifelong learning* 的学习机制，对用户的长期 *knowledge* 进行积累，并进行不断的累加学习，也是我们需要研究的一个课题
3. 个性化与多样性的权衡: 目前的导购产品中，比较重视用户的个性化，之前也尝试过在个性化与多样性之间进行一些平衡，但对指标在短期内有一定的负向影响。但如果考虑较长的时间范围，多样性对于用户的活跃度是有正相关的，所以如何在个性化与多样性之间进行平衡，同时通过长期收

益来评价效果，也是在导购下需要解决的一个问题

4. 用户意图识别的负反馈: 在 Query 推荐场景下，对于用户的负反馈也面临较多的挑战，尤其是在伪曝光的前提下。因为在实际中，我们虽然在搜索框的底纹中，给用户推荐了某个 Query，但是他到底有没有真正看到，或者说看到了也确实满足他的意图，但因为各种原因，跳到了其它 app 上，所以这部分样本不能直接作为负样本。如何获取到用户真正不满意的推荐 Query，以及通过 bandit 的方式将其在后续的推荐中进行一定程度的降权，也是我们在持续优化的一个点
5. 推荐中的组合优化问题: 今年在底纹中做了较大的产品升级，从之前的单个底纹到现在的多个底纹轮流滚动的形式，在算法上就从之前的 top1 问题转换到了现在的 topN 问题，如何决定 N 的值，以及根据用户的浏览宽度等决定不同类目的组合也是一个新问题，比如有的用户的兴趣点比较狭窄，可能这 N 个 Query 就偏向于这一个意图，有的用户兴趣比较发散，我们就可能会在类目宽度上放得更广

搜索导购包含了多个产品形态，如底纹、首页分类、搜索发现以及下拉等产品形态，在产品定位上我们有下面的思考：

在用户路径上，底纹、首页分类以及搜索发现是在用户发起搜索前的无 Query 情况下的推荐，更加关注的是对于用户意图的理解与推荐。而在下拉场景下，当用户已经输入了某个前缀词之后，我们会从效率和个性化两个方面，为用户提供更好的下拉服务

作为一个导购的平台，对于用户的全生命周期学习与表示能够更好地识别用户意图，同时在不同场景下的样本、特征共享，可以将一个 domain 的信息平滑的迁移到另一个 domain 中，这两个方面我们在底纹中进行了初步的尝试。召回与排序一直是算法的核心，今年我们在下拉中尝试了文本生成作为召回的扩展，同时在排序上也加入了 position bias 的机制来解决位置所带来的偏差；

底纹: 底纹是导购中非常重要的一个产品，当用户打开手淘首页时，底纹推荐词就随之出现。因此底纹词推荐的精准性对于吸引用户进入搜索是至关重要




图 3.20

的。底纹词在这里承接了唤起用户搜索记忆以及提供更精准 Query 的功能。今年我们在底纹方面尝试了多种产品形态的优化以及算法优化。在产品形态上，今年上半年尝试了多底纹同时展示的产品形态，最终采用了当前的多个底纹词滚动展示的形态。在算法方面，我们分别从召回和排序两个方面进行了升级。主要的优化点有：产品层面上的优化，从传统的单个推荐词到当前的多个词轮播的形式；召回上，进行了扩覆盖、分层召回以及权益类 Query 的召回等的扩充；



图 3.21

排序上，我们正在尝试实时联合反馈的 LTR 排序模型，将多个导购领域的样本、特征进行共享，形成跨领域的学习后续，底纹是导购方面非常重要的算法着力点，后续会继续在 lifelong learning、multi-domain learning、组合优化等方面进行长期的深入探索；

底纹优化的演进历史：

1. 基础版本，截止今年 7 月份之前底纹的产品形态还是跟往年一致，用户每

次到达首页会触发一次底纹推荐服务请求，算法返回预测出的最有可能被用户点击的 top1 个 query 展示给用户；

2. 多底纹版本，主要考虑到部分用户会在首页停留的时间比较长，如果推给用户的 query 固定是一个的话没有充分利用到用户的停留曝光机会，我们尝试了在搜索框同时展示 1 3 个 query 的样式，但同时会受限于搜索框长度，展示的 query 不能过长，这一版样式 AB 效果没有太大变化；滚动底纹，顾名思义是指用户在首页停留期间可以看到多个 query 固定时间间隔轮播展示，是在以上版本总结和思考的基础上新的一种尝试。既能较好利用用户在首页停留期间的曝光机会，相比基础版本 query 长度又不会受限，此外加入动效后也能一定程度上吸引用户眼球。经过一段时间 bts 测试，线上 AB 效果也符合预期，底纹 uctr 提升 5

3. 大促样式，主要在双 11 当天的大促氛围下对底纹做了颜色加黑加粗的效果强化

底纹的召回和排序优化：底纹的在线召回以个性化为主，主要的召回类型有偏实时的 i2q, q2q, 以及偏长期的 longtime prefer q, 个性化覆盖占比 90 %, 众所周知个性化推荐效率要远高于非个性化，那么对于有历史行为的用户如何继续提升个性化覆盖占比，以及对于手淘拉新和用户下沉带来的新用户如何拉新到搜索是这我们在召回优化方面的主要出发点；

1. 整体扩召回，旨在扩大有行为的那部分用户的个性化 query 召回候选集，主要从 i2q, q2q 的离线扩覆盖，以及在线拉长用户点击/搜索行为周期入手。通过 item 的短标题生成，title2query 候选集合并等手段，我们的 i2q 覆盖从原来的 3000w 宝贝扩充到 2.1 亿，q2q 从覆盖 1600w source query 扩充到 1800w。这块带来的在线效果也比较明显，AB 测试使用 uv+4%
2. 分层扩召回，主要是将无少行为的那部分用户按手淘的新人分层 (N0 N8) 划分，离线计算每个分层用户偏好的 query，在线扩充基于群体的个性化召回候选池，这样做的一个好处是对于纯新登用户来说也能有一定的弱个性化推荐结果

3. 权益类 query，结合之前针对三四线下沉新用户的用研报告，新用户更关注切实的优惠信息，我们尝试了联合运营圈品和获取用户实时红包在底纹透出一些带权益 tag 类的 query，比如“家用拖把 9.9 元包邮”等

底纹的排序优化：在针对新老用户的整体/分层扩召回基础上，我们自然而然会想到排序方面是不是也可以做一些针对性的优化，以下从特征和模型两个方面做简要归纳；特征，1) user 泛化特征强化，新增用户手机品牌、30 天浏览类目宽度、浏览卖家数、最近 1 月手淘访问天数、未来 7 天气温等特征，离线 auc 提升 1.3%，在线使用 uv 也有 1 个点提升；2) 将 query 维度的先验 ctr，引导 pv/uv 等特征按用户分层细分；模型，1) 新老用户模型分层；2) 随着产品形态从单底纹到多底纹滚动的演进，离线模型用最新采样方式采到的样本来更新也能获得不错效果；

基于 Bandit 重排：重排阶段是介于精排和最后的展示之间对 query 排序结果的最后一次在线调整，有时候也称为策略层，一般会考虑排序 list 的上下文和满足一些体验保障的约束条件（比如类目的重复度等）。在这一阶段我们的优化目标既要兼顾推荐 query 的点击效率和丰富度，又要结合用户实时反馈尽量减少用户不感兴趣 query 在下一次的曝光机会。下图展示了底纹重排从最初的精排后 topN 随机->topN 轮播->Bandit 重排的优化历程以及背后的思考：

基于实时联合反馈的 LTR 排序模型：在开头问题思考部分我们提到对用户而言从首页跳转到搜索的使用过程中搜前->搜中->搜后是一个连贯而不是割裂的过程，在这个过程中用户实际是跟多个 query 导购产品有着丰富的交互行为，比如搜索前的底纹，搜索中的下拉、主动输入，以及搜索进入 srp 页后的锦囊/交互式搜索等。如何将这几个导购产品联动起来，更好地在各阶段满足用户的搜索意图是本部分我们研究的主要出发点。下图是用户从首页跳转搜索过程中与底纹、下拉、主动输入的一个交互示例，在这中间用户提交 query 的方式有多种，比如底纹、下拉推荐曝光的 query 有的会被点击，有的只曝光没有有效点击，有点击的我们认为是正反馈，只曝光未点击的是负反馈信息，它们都能一定程度上反应用户在当前时刻的偏好信息；

传统的底纹、下拉排序任务中都是独立分开优化的，在样本、特征以及用

图 3.22

户反馈上并没有做到信息的共享和传递。从上面的分析角度出发，我们思考是不是可以用一种基于实时联合反馈的多任务学习模型，在特征信息层实时拿到用户在各 query 导购产品的正负反馈时序信息，样本层在各 task 独有的特征基础上共享联合反馈特征，任务层做 multi-task learning，具体如下图所示

模型结构设计：基于以上设想，我们提出了一种基于实时联合反馈的 LTR 排序模型，与上不同的是在实验阶段我们对任务层做了一定简化，通过实时多

图 3.23

导购交互的正负反馈特征来学习用户向量，task 层先只优化底纹的排序模型。该模型的优化目标是预测底纹推荐给用户的 query 在未来是否会被点击。相比以往的底纹排序模型特点是在特征层通过利用用户在多导购场景的正负反馈信息强化用户表示。具体来讲模型主要有以下几层：

1. user 向量表达层，利用系统之前展示给用户但未点击的底纹 query，下拉推荐给用户但未点击的 query 作为负反馈信息，通过底纹/下拉/主动输入




图 3.24

提交的 query 作为正反馈信息。设计一种 Filter Attention 结构，利用这些负反馈信息来衡量用户以往意图中哪些意图在下一步将会减弱，反之负反馈中的无效曝光也可以一定程度上被鉴别

2. user 向量表达拼接传统的统计特征和候选 query 特征，因为原有特征也包含了 trigger, user, query 的丰富信息

### 3. 经过多层全连后，学习点击/不点击二分类问题的交叉熵损失函数

下拉：手淘的下拉推荐是流量最大的导购产品和搜索发起途径，日均使用 uv 1.05 亿，占搜索 uv 的 75% 以上；通过下拉推荐的引导 pv 占总的搜索 pv 46% 左右，比用户主动发起搜索的比例还要高。同时下拉又是一个古老的产品，伴随着搜索框的存在而存在，因此如何在一个具有悠久历史的产品上，针对所遇到的问题进行创新，是今年面临的挑战；

1. 通过纠错数据、序列数据以及相似 Query 和文本生成方面的技术，对下拉的召回池进行了扩展，对下拉的引导 PV 有约 4% 的提升
2. 排序中去除 position bias，并建立离线训练与在线效果之间的关联，和深度模型的继续优化，在排序方面对下拉的引导 PV 有约 1% 的提升
3. 工程方面，今年将下拉迁移到图引擎框架上，在召回排序扩展性和性能上，都有了明显提升。特别是多路召回的并行化和召回结果的混合排序，为下拉带来了 50% 的 latency 下降，同时使得整体线上流程拓展更加灵活，大大加快迭代效率

全网热榜：全网热榜的初衷是在用户增长这个大前提下，如何更好的增加用户对搜索的粘性。一方面，当用户带着需求来搜索完商品之后，他可能并不知道当前全网的热门或者趋势的数据是什么，目前的搜索发现部分比较强调个性化的因素；另一方面，当用户并没有明确的需求时，同样可以在全网热榜下面找到一些目前比较普遍热门的商品，让用户在搜索中逛起来。从而提高用户对搜索的使用粘性。是一个算法 + 运营互相协作的产品。全网热榜的系统流程包含了热点挖掘、榜单管理平台，以及前台投放和效果反馈几个方面；

热点挖掘：热点挖掘主要是从淘系内，以及淘系外的数据源中进行挖掘。在淘系内的数据中，我们挖掘的重点分为 2 个方面，一个是最热点的内容，另一个是趋势 Query。最热点的内容是指一直都很热门的，比如像“连衣裙”、“手机”这样的，而趋势 Query 是指那些之前不太热门，但是突然间火起来的，比如像圣诞节之前的圣诞节礼物，或者某个科技公司新发布的一款产品。对于趋势




图 3.25

Query 的挖掘，目前主要是通过一些比较重要的特征进行的拟合，比如像 Query 最近 30 天的 pv uv ctr 等指标的变化情况，比如当天 uv 和前 1、2、3、7、15、30 天的差值，以及当天 uv 的增速与前几天的 uv 的增速的差值等。又比如用户搜索 Query 之后，点击的商品是否新发布的情况等。通过约 70 个特征对 Query 的趋势度进行计算；



图 3.26

### 3.2.16 参考文献

题研究及其进展 (<http://cjc.ict.ac.cn/quanwenjiansuo/2012-10/hl.pdf>)

-layer perceptron (<https://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-016-1232-1>)

Learning Algorithms (<http://ieeexplore.ieee.org/document/6471714/>)

图 3.27

题 - 综述和实践 (<https://zhuanlan.zhihu.com/p/25928551>)

Text Classification (<http://www.aclweb.org/anthology/I13-1006>)

neural networks (<http://www.sciencedirect.com/science/article/pii/S0022000013000718>)

shopping queries ([https://www.cs.utexas.edu/~cywu/www2017\\_shopping\\_query.pdf](https://www.cs.utexas.edu/~cywu/www2017_shopping_query.pdf))

1. [https://github.com/brightmart/text\\_classification](https://github.com/brightmart/text_classification)

lication domains (<https://link.springer.com/article/10.1007/s10618-010-0175-9>)

## 3.3 query 语义改写

### 3.3.1 问题背景

在去年的 search 2.0 项目中 query 改写 [1] 取得了不错的效果，今年的迪拜塔项目我们准备再进一步。首先我们调研了一下突破点，主要的问题还是在于用户 query 和宝贝描述之间存在 GAP，特别是中长尾 query。把问题分成以下几种类型：

- 多种描述：划痕笔/补漆笔/修补笔/点漆笔
- 信息冗余：冰箱温控器温度控制 == 冰箱温控器
- 属性检索：118 冰箱、60 寸液晶电视机 4k 高清智能 60 曲面
- 宽泛意图：超美吊灯、大容量冰箱

### 3.3.2 所做工作

query 改写的目标空间可以分为文本空间和意图 ID 空间两种类型：文本空间包含词、短语、query，意图 ID 空间主要包括 pidvid、性别年龄尺码等自定义 tag、一些语义聚合的标签如：“奢侈”，“可爱”等。所以我们的工作还是主要基于以下两种形式：

- 向量化改写：query 映射到 query，主要针对“多种描述”和“信息冗余”类型
- 意图 ID 改写：query 映射到意图 ID，主要针对“属性检索”和“宽泛意图”类型

### 3.3.3 向量化改写

向量化改写的基本流程仍然是：query 向量化 □ 向量相似查找 □ 相关性判断

#### 3.3.3.1 query 向量化

主要优化点在于通过尝试一些有效的向量化方法，对任意 query 向量化。下面介绍一下我们的做法：

- seq2seq
- 完全无监督
- i2i 扩展

**3.3.3.1.1 seq2seq** 这里我们借鉴的是 Skip-Thought Vectors[2], 这个算法试图通过 seq2seq 重建句子周围的句子，如下图所示：模型的目标函数也是两个部分，

图 3.28

一个来自于预测下一句，一个来自于预测上一句。如下式：我们从 session 中获取训练预料，假设某个 session 序列是  $(s_1, s_2, \dots, s_n)$ , 那么一条训练数据就是  $(s_{i-1}, s_i, s_{i+1})$ , encoder 是  $s_i$  词序列的 lstm, decoder 是分别  $s_{i-1}$  和  $s_{i+1}$  的 lstm. 这样训练下来 decoder 的上下文向量就学到了这个句子在 session 中的上下文表示。

图 3.29

其实除了 query session，用 query 来重建其点过的宝贝标题序列同样适用，只不过 decoder 阶段换成 query 点过的标题。

**3.3.3.1.2 完全无监督** 这里我们主要借鉴的 WR 方法 [3]，作者在论文中证明：在维基百科上的非标签语料库中使用流行的方法训练 word embedding，将句子用词向量加权平均，然后使用 PCA/SVD 修改一下。这种权重在文本相似性任

务中将效果提高了约 10% 至 30%，并且击败了复杂的监督方法，包括 RNN 和 LSTM，它甚至可以改善 Wieting 等人 [4] 的 embeddings.

算法如其名，主要分为 W 和 R 两个步骤：

- W：就是词向量加权平均的时候的权重。
- R：使用 PCA/SVD remove 向量中的 common 部分。

根据 query 的描述文本的类型，我们产出了两种向量：

- 基于 query 自身的词。
- 基于 query 和 query 点击的宝贝标题的词。

在点击数据集上，我们在 query 相似任务上测试 WR，比 pai-doc2vec[5] 在单特征 auc 上有 1.4 个点的提升。针对上述第二种类型，在原有算法的基础上，我们做了一些改进：

- 互信息权重

query 点了一些标题，并不是对标题中的所有词感兴趣，对于相关性任务来说，理论上我们只关注和 query 最相关的词，所以在 W 阶段我们加入词和 query 互信息的权重  $MI(q,w)$ , 即  $a/(a+p(w)) \square MI(q,w) * (a/(a+p(w)))$ , 对应的  $|s|$  也根据  $MI(q,w)$  进行了 scale. 上述权重的改变使得单特征 auc+3.5 个点。

- 适当加大 query 自身词的权重

基于 review case 发现因为线上已经上了相似 query 改写/意图改写 (标题中可能缺失了一些词)，导致原始 query 自身的某些词被 attention 的权重偏低。值得注意的是，当我们进一步适当提升 query 自身词的权重之后，auc 又会 +2.2 个点。更好的做法是把点击的对象的更多属性参与向量化，比如点击的 pidvid，意图 id 等。

**3.3.3.1.3 i2i 扩展** 对一些依赖行为的算法, 对于一些行为偏少的长尾 query 通过宝贝的 i2i 来扩展, 可以扩大计算的覆盖率和准确度。这部分我们做了以下工作:

直接召回 query 历史点击宝贝的 i2i 宝贝这个功能主要针对少结果 query 和低质量 query, 在去年已经做了, 这次我们做了更严格的改写限制和相关性限制, 收严上线。

- 协同过滤算法的扩展

q 对应的 item 越多, 基于 item 的 CF 算法覆盖率越高。针对中长尾 query 我们用 i2i 扩展了点击行为。

- 基于点击数据的 WR 方法

对于点击少于 20 的长尾 query 利用 i2i 扩展点击数据优化 embedding, 覆盖 query 在相似 query 任务上单特征 auc 有 4 个点的提升

### 3.3.3.2 向量相似查找

目前候选 query 集合设置的 500W 左右, 而需要计算相似的 query 集合数据量已经到 9 亿+, 相似搜索是一个非常大的计算量。计算步骤如下:

1. 对候选 query 集合进行 kmeans 聚类成 C 个簇
2. 遍历 C 个簇中心查找余弦距离最近的 topM 个类
3. 然后遍历 topM 个簇中的点获取 topK 个相似

我们也尝试做了以下的改进实验:

1. 随机投影: 同样数据量和 kmeans 对比, 4096 簇, top20% 簇召回率 99%, top10% 簇召回率 97% (kmeans top10% 簇召回率 99%)
2. 积量化 [6]: 由于我们的维数不高 (100 维), 虽然性能提升 1/3, 但是流程也复杂了不少。

大规模数据的向量近似搜索是目前很多业务面对的问题，后续还需要再继续调研业界和其他兄弟团队的高效做法，优化上述流程。

### 3.3.3.3 相似判断

目前线上我们还是继续沿用的 GBDT 模型，也正在尝试使用深度语义模型，下面介绍下样本选取和模型训练部分。

**3.3.3.3.1 样本选取** 深度模型依赖大量的样本喂养，而接近于真实分布的样本一直是一个难点。我们主要尝试了以下方式：

- 行为反馈获取样本
  - query 下有不同的改写串 rewq，计算出各个 rewq 所带来的召回和收益情况 (点击 + 成交)，通过收益的 gap 筛选正样本和负样本。
- 从相关性任务中迁移样本
  - 改写和相关性是联动的，query 和宝贝相关性的样本可以迁移过来使用。比如：一个宝贝和一个 query 不相关，那么这个 query 和点击到这个宝贝的所有 query 理论上都不相关。
  - 弱标签：好处是数据丰富，获取相对容易，比如：
    - q1 和 q2 的相似度我们用 (q1,q2 点击标题) 的 bm25 或者 (q2,q1 点击标题) 的 bm25 来构造弱标签
    - 用 q1 和 q2 下挂宝贝的相似度来衡量 q1 和 q2 的相似度：用 q1 去 attention q2 点击 doc 中的词，然后向量化和 q1 attention 自身 doc 的词向量化距离进行比较，选取一些高置信度的作为 label
    - 用简单模型的输出结果来获取样本，用我们已有的 GBDT 模型来预测，取得分数高于一定阈值或者低于一定阈值的分别作为正样本和负样本。



- 规则构造样本比如：使用同义词和冲突词的片段替换分别构造正样本和负样本

**3.3.3.3.2 模型训练** 这部分双 11 之前我们基于上述样本尝试了一部分，后面由于时间关系没有上线。下面就简单说讲一下我们初步调研的一个模型：高频词和短语通过语言模型或者点击文档能获取很多扩展的语义，容易获取高质量的 embedding。而长尾 query 由于行为稀少，单存依赖自身信息获取 embedding 的质量通常不理想。调研了一下基于 Skip Convolution 或 BiLSTM+Self-attention，通过合理设计联合训练网络，把词和短语级别丰富的 embedding 信息迁移到长尾 query 上，实现提升长尾字符串 embedding 质量的目标。

图 3.30

- 网络的左半部分：主要为了学习 query 在“session”、“点击”、“query 自身”三个维度的融合表示，在相似 q 任务上我们继续更新和 query 相关的向量三个维度主要目的在于学习词，短语粒度的语义信息，泛化到所有 query 每个维度的 query model 可以使用跳跃卷积 (skip-cnn)，主要是考虑商品搜索的 query 对 term 位置的信息较为不敏感，也可以选择双向 lstm+attention。其实这个思路和我们的评测人员的思路很像，评测两个 query 是否相关，除了一个整体判断，更多的是把 query 拆解成一些‘短语’去判断。
- 网络右半部分 第二部分主要是为了解决相关性判定的问题，用了一个二分类的模型这里把 q-q pair 的特征也作为网络的一部分输入

### 3.3.4 意图 ID 改写

#### 3.3.4.1 精确 cpv 改写

为了解决用户直接搜索属性而标题不写属性的问题，我们把属性改写做了全类目的覆盖。

另外把所有意图改写的逻辑做到了在线改写，这样更利于长尾 query 的覆盖。

#### 3.3.4.2 数字类宽泛 query 改写

对于意图比较宽泛的 query，比如畅销手机，大容量冰箱，虽然在 query 向量改写中会有所体现，但是对于行为较少的 query 改写的还是不太理想。我们采取的做法是把他们映射到我们的意图 id 空间，比如价格，销量，pidvid，年龄，性别，尺码，季节，新品等。

这次我们重点关注了数字类型的改写。主要分成两个步骤：

- 宽泛短语挖掘：首先阿里分词的粒度肯定是不够用的，我们首先进行了 phrase mining[7]，然后依据以下 3 点进行挖掘：根据种子词：这个就是尽可能寻找更多的数字类型的形容词，比如大，小，长，高，胖，瘦... 根据极限前缀：最，超.. 这些极限词为前缀的短语往往也是我们想要的根据 session 信息，在一些包含明确数字意图的 query 中，也可以挖掘过一些宽泛短语。比如用户搜索”节能冰箱”之后就有可能下一步搜索”一级能效冰箱”这种具体的量化词了。
- 短语到 pidvid 的映射：有了短语之后就可以进行短语到 pidvid 的映射了，也主要尝试了下面 3 种方法：基于一些映射规则：价格类的词到价格，畅销类词到销量/人气等。基于点击行为：当前 term 在某个属性值的分布 vs 属性值在所有 term 的整体分布，如果分布的差异大于一定阈值，则可以认为 term 和属性值之间有关联，可以挂靠基于向量相似度：根据短语点击的宝贝标题和宝贝 pidvid 对应的宝贝标题，用上面的 WR 方法分别把短语和属性都映射到标题空间的向量。给定一个短语，根据向量相似度就可以召

回其最相关的属性了，但是这个方法的准确率不太高，做推荐还可以，做改写的话还需要进一步的优化。

最后 show 几个挖掘的例子, 字段含义: 短语 | 叶子类目 ID | 映射的 pidvid | 映射的 pidvid 的文本

- | 大容量 | 350301 | pidvid::148776183:7563249 OR pidvid::148776183:92238 OR pidvid::148776183:92239 | 洗涤公斤量:8kg; 洗涤公斤量:10kg; 洗涤公斤量:15kg
- | 迷你 | 121366015 | pidvid::122216906:34872414 OR pidvid::122216906:75369125 OR pidvid::122216906:75475137 | 化妆品净含量:20g/ml; 化妆品净含量:20g; 化妆品净含量:8g/ml
- | 0 - 6 个月 | 211104 | pidvid::6932095:4097935 OR pidvid::20017:39834535 OR pidvid::20017:3306535 | 适用阶段: 一段; 适用年龄:6 个月以下; 适用年龄: 新生

### 3.3.4.3 知识图谱

这部分主要是和第三方数据的合作，进行了初步尝试，还在进展之中

- 商品知识图谱

商品知识图谱那边沟通了一些数据：品牌，新品，产品库，影视、明星同款标签, 同义词/上下位等，这些都可以在改写上应用。目前主要是品牌和一些标签数据在 bts。

- query 知识图谱

主要是 cpv 扩展的数据和一些知识的定义，如果”奢侈”在某些类目下可以用品牌来定义，这些对 query 扩召回也非常有用。

- 评论数据

把从评论里面抽取的标签参与召回，比如‘料子好’，‘尺码合身’，‘版型好看’；‘质量好’，‘服务好’等最后 show 一个搜索‘版型好看的连衣裙’的结果对比，见下图：

### 3.3.5 进一步计划

上述的 query 改写可以带来 GMV 和 UV 价值各 +2 个点的效果。可以想象 query 改写的空间依然是巨大的，但是我们还有很多没有做好，也还有很多要做。

对于深度模型我们会继续做，考虑 query 生成的思路特别是强化学习选词，Query 向量的表征空间扩展到属性空间，图像空间甚至是用户空间。关于改写的目标，相关性是基础，但绝不是最终目标，后续会考虑目标改写 query 的价值甚至是个性化改写。

”相似”对应衡量 query 之间的关系还是太粗了，后续我们要把 query 改写的类型明确化，比如同义和蕴含(上下位)，对于 query 之间的关系我们正在尝试一些方法 [8] 来挖掘。后续我们计划挖掘更多的知识，并考虑如何把知识融入到模型之中。

参考文献：

- <https://www.atatech.org/articles/66690>
- Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, Sanja Fidler. Skip-Thought Vectors
- Sanjeev Arora, Yingyu Liang, Tengyu Ma. A Simple but Tough-to-Beat Baseline for Sentence Embeddings
- John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. Towards universal paraphrastic sentence embeddings
- <http://help.aliyun-inc.com/internaldoc/detail/34590.html?spm=a2c1f.8259796.2.159.boklrB>
- <https://www.atatech.org/articles/15961>

- <https://www.atatech.org/articles/50958>
- Ruiji Fu , Jiang Guo , Bing Qin. Learning Semantic Hierarchies via Word Embeddings

## 3.4 Query Tagging

### 3.4.1 背景及历史

在整个搜索体系中，用户 query 的理解对搜索结果起着非常重要的作用，是搜索链路中较为关键的一环，会直接间接的影响到商品召回，相关性，排序等各个方面。而 QueryTagging 作为其中一个部分，承担着将原始 query 进行结构化打标任务，Demo 如下：

图 3.31: query: 2016 秋水伊人大码连衣裙

精准的 tagging 结果有利于帮助搜索引擎了解用户需求（例如用户所关注的品类，品牌，风格等），从而给用户提供更优质的搜索结果。同时，我们也能通过 QueryTagging 结果分析出用户的搜索习惯，各种品类品牌的搜索趋势等等，方便产品运营同学了解市场，了解用户。

要做打标，第一个面临的问题就是需要明确标签候选集。大家都知道，淘宝的每个商品上都包含着丰富的结构化信息（类目，属性等），而通过分析日志不难发现大部分的用户 query 内容都体现在这些信息里面。因此，我们基于商品的

cpv 数据, 结合用户最常见的搜索意图, 构建了一套适用于主搜业务的标签体系 (目前共计 42 种 Tag 类型), link: <http://searchwiki.alibaba.net/index.php/QT-tag>

在任务初期, 缺乏标注语料的年代, 引入任何高大上的模型都是空谈。因此为了构建打标任务, 我们采用了简单有效的词典匹配方法。针对不同类目, 在词典的质量和覆盖上都做了大量工作。效果也十分显著, 渐渐的将打标任务的准确跟召回都提升到了百分之八九十。对类目预测, 相关性等后续业务都带来了不小的帮助。

然而随着业务的逐渐精细和不断发展, 我们对 QueryTagging 也提出了更加严格的要求。词典版本的许多弊端慢慢开始展现, 例如无类目预测 query 下的召回问题, 一词多义问题等等。而深度学习的慢慢普及, 以及前期词典版本带来的一些语料沉淀, 也给我们解决这些弊端带来了可能, 因此我们尝试在 Query-Tagging 任务中引入 Deep Learning 模型进行优化, 并最终在目标数据集上取得了较好的效果提升。

### 3.4.2 Sequence Labeling 模型

QueryTagging 可以看成是一个分类问题: 对每个 term, 我们需要根据上下文信息来判断, 它属于标签体系中的哪个类别。我们尝试过将其作为一个标准的分类问题来处理, 抽取 term 本身的特征以及上下文特征作为输入, 标签作为输出, 但是效果不太理想, 一方面特征工程比较麻烦, 需要人为的去处理很多看似有用的特征; 二是这种做法对上下文信息的利用不是非常充分。

因此在做了一些尝试后我们将问题重新转换回序列标注问题, 使用目前较为流行的 bilstm+crf 框架, 避免了许多复杂的特征工程, 并且最终在无类目预测 query 上取得了较好的效果提升。

模型结构图如下:

整个网络包含三个层次, Embedding layer, Feature layer, Inference layer, 逐层介绍:

#### 1. Embedding layer:

Embedding layer 主要由三部分 concat 组成,

图 3.32

- \* term 维度 embedding
  - \* character 维度 embedding, 通常使用 CNN 或 RNN 将字符维度抽取成 term 维度的向量表达。
  - \* term 维度的一些手工特征, 例如是否是数字, 是否全英文等。
2. Feature layer Feature layer 根据输入的 embedding 向量进行特征抽取, 理论上可以是任意的 DNN,CNN 或者 RNN 网络。考虑到 RNN 在处理序列问题上的优点和特性, 我们真实实验中使用的是双向 LSTM, 同时我们对比过多层堆叠的 LSTM 结构, 在效果上相差不大。
  3. Inference layer Inference layer 根据 feature layer 抽取的特征, 需要对每一个 term 做分类任务, 可以是一个简单的 softmax layer, 也可以是序列标注中常用的 crf layer。考虑到 CRF 能考虑到不同 tag 之间的转移概率信息, 我们在实验中采用了 CRF。

### 3.4.3 与知识图谱的结合

QueryTagging 任务与搜索目前正在构建的知识图谱体系息息相关, 双方互为输入输出。一方面, QueryTagging 的打标结果能帮助知识图谱进行实体识别, 实体链接等工作。另一方面, 知识图谱内丰富的知识数据有利于帮助 Tagging

任务处理未识别词，提升已识别词的准确率。相信随着知识图谱的不断完善，tagging 也会有更好的表现。

### 3.4.4 实验

QueryTagging 任务一直以来存在的一个问题是无类目预测的 query 下，打标效果较差。主要原因在于词典版本是基于类目维度的，在无类目信息下，词典的覆盖跟准确都有较大问题。而在带类目预测 query 下，打标的质量较高。因此我们使用带类目预测的 query 作为训练样本对模型进行训练，对无类目预测的 query 进行预测，从而提升无类目预测 query 下的打标准确率与召回率。

#### 3.4.4.1 评测结果

我们在无类目预测 query 下，针对词典版本以及模型版本的打标结果分别进行了评测。评测流程是随机抽取 2000 条左右无类目预测 query 提供给外包同学对打标结果进行审核。同时对外包审核结果会进行抽查，确保审核准确率在 95% 以上。

从评测结果上看，模型版优化后的数据指标明显优于词典版本评测指标。具体表现为 `accuracy +12%, precision +4.6%, recall +8%`。

### 3.4.5 思考及展望

从实验效果看，带类目 query 的学习对无类目 query 的打标帮助很大，同时也能解一些词典版本无法解决的问题，这是我们第一次将 DeepLearning 引入到主搜 QueryTagging 任务中，使用了业界比较流行成熟的算法架构，带来了不错的效果提升。但实际上，主搜 QueryTagging 的场景下仍有许多问题待解，例如分词错误导致的打标错误，型号词识别问题等等，这些都不是一个模型能通吃的，它们也给我们带来了更多的场景供算法发挥，例如带类目与无类目数据之间的迁移学习，tag+ 分词的 multitask 任务，打标与知识图谱的结合等等，这些也都是后续优化的方向。



### 3.4.6 部分参考文献与链接

- Adversarial Multi-Criteria Learning for Chinese Word Segmentation
- Named Entity Recognition with Bidirectional LSTM-CNNs
- End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF
- Part-of-Speech Tagging for Twitter with Adversarial Neural Networks
- QueryMatrix 平台 (<http://sqi.alibaba-inc.com/MatrixQ/index.htm>)
- Query 全景洞察 (<https://sqi.alibaba-inc.com/babel/queryintention/queryUser/industryDesc>)
- 知识图谱阡陌平台 (<http://sqi.alibaba-inc.com/qianmo/toSetKgPinlei.htm>)

## 3.5 Query 纠错

### 3.5.1 背景

在实际的搜索输入框中用户输入的查询词 query 不一定是完全正确的, 此时如果直接使用用户输入的 query 去查倒排索引, 一般情况下都无法召回正确的宝贝, 甚至召回不了宝贝. 为了提高用户的使用体验, 一般的搜索系统都会提供 query 纠错的功能, 在检测到用户输入的 query 有误的情况下, 会提示用户对应的正确 query 或者直接对用户的 query 进行纠正, 方便用户直接得到想要的结果.

### 3.5.2 任务说明

在英文拼写纠错系统中, query 错误的形式主要分为两种: 一种是 Non-word Error, query 中部分英文单词不是实际词典中存在的单词, 比如: `theree is nothing` -> . 另外一种 Real-word error, 即部分英文单词是词典中存在的词, 只是在当前语境下, 这个单词是错误的. 比如: `three is nothing` -> `there is nothing`

考虑到中文的特殊情况, 以及中文存在拼音输入的问题, 因此相比单纯的英文纠错之外, 中文系统的拼写纠错会稍微复杂一些. 在淘宝搜索系统中, 拼写纠错包括三个子任务:

### 1. 英文纠错

淘宝中常见的英文输入错误主要是英文品牌输入错误. 例如: ipone 手机 -> iphone 手机, barberry 围巾 -> burberry 围巾等, 和英文系统中的错误类型基本一致.

图 3.33

### 2. 中文纠错

中文汉字的错误形式一般是”别字”错误, 比如: 拼音相似以及不同地区发音不一致导致的错误, 比如: 雅麻连衣裙 -> 亚麻连衣裙. 字形相似导致的错误: 榨汗机 -> 榨汁机.

### 3. 拼音转换

由于中文目前大多是拼音输入, 因此用户输入的 query 可能是原始拼音串或者是错误的拼音串. 此时需要将原始 query 转换成正确的中文.

图 3.34

图 3.35

### 3.5.3 实现方案

Noisy Channel Model(即噪声信道模型), 是一种比较常用的用于语言识别, 拼写纠错, 机器翻译等领域的一种普适性模型. 模型形式很简单:

噪声信道模型主要是根据带有噪声信号的输入来还原输入的信号. 形式化定义为:

即: 在给定输出  $O$  的情况下, 找出概率最大的输入  $I$ . 在实际的纠错系统中, 用户实际输入的 query 为  $Q$ , 我们需要找出对应的最大可能的正确输入 query  $C^*$ , 这个 query 才是用户真实想要搜索的 query. 我们总共尝试了三个版本的纠错方案, 但本质上都是在观测到用户输入 query  $Q$  的情况下, 求解对应的概率最大的正确 query. 下面介绍下我们在淘宝搜索中的拼写纠错的工作.

图 3.36

图 3.37

### 3.5.4 基于 HMM 的拼写纠错

首先我们复现了基于 HMM 的拼写纠错流程. 整个流程如下图所示, 整个过程主要分为两个部分: 候选 query 生成和候选 query 排序.

为了生成候选正确 query(C), 首先针对输入 query (Q) 中的每一个 word 都生成对应的候选正确 word 集合. 然后从每一个输入 word 的候选 word 集合中选择 topK 来组合生成对应的正确 query.

#### 3.5.4.1 候选 query 生成

针对原始 query 中的每一个 word, 根据对应的候选生成逻辑生成候选 word 集合.

图 3.38

1. 当前 word 为中文主要根据拼音相似, 字形相似两种先验知识生成候选的正确中文 word, 同时我们根据 session log 中统计的中文常见输入错误形式补充对应的候选正确中文 word. eg: 瓶 -> 平, 品, 屏, 评, 频, 拼, 凭
2. 当前 word 为英文字符串针对英文字符串, 主要是根据编辑距离来生成候选正确英文 word. 线上主要是根据编辑距离  $\leq 2$  来生成候选正确英文 word.
3. 当前 word 为 pinyin 串由于拼音输入法的问题, 因此英文字符串有可能是中文对应的拼音. 为了处理这种情况, 离线我们根据 query log 获取了常见的中文 ngram, 并以此构建了常见的拼音串以及对应的切分位置. 比如: longjing -> long | jing  $\hat{A}$  253. 在线生成候选时会先对拼音串进行切分, 然后根据切分之后的每个拼音生成对应的候选中文 word.

#### 3.5.4.2 候选 query 排序

在候选 query 排序阶段, 针对原始 query 对应的 word 序列, 从每个 word 对应的候选正确 word 集合中选择一组概率最大的 word 序列, 即是对应的候选正确 query. 形式化定义为:

$$C^* = \arg \min_C P(C|Q) = \arg \min_C \frac{P(C, Q)}{P(Q)} = \arg \min_C P(C, Q)$$

$$= \arg \min_C (\pi_{C_1} \prod_{i=1}^{l-1} a_{C_i C_{i+1}} \prod_{i=1}^l b_{C_i}(Q_i))$$

C\*: 最优正确 query ; C: 任一候选 query; C<sub>i</sub>: 候选 query 中的第 i 个字; Q: 原始 query ; Q<sub>i</sub> : 原始 query 中对应的第 i 个词; L :query 长度

候选 query 排序中每个 query 对应的概率由两部分组成:Language Model 和 Error Model.

$$\prod_{i=1}^{l-1} a_{C_i C_{i+1}}$$

Language model: 表示候选 query C 出现的可能性大小.

$$\prod_{i=1}^l b_{C_i}(Q_i))$$

Error model: 表示候选 C 被错写成 Q 的可能性大小.

为了计算候选 query C 出现的可能性大小. 我们尝试了两种概率计算方式: 一种是传统的 ngram 语言模型, 另外一种是 LSTM 语言模型.

### 3.5.4.3 N-gram 语言模型

首先我们使用了经典的 N-gram 语言模型来计算候选 query C 的概率. N-gram 语言模型也称为 n-1 阶马尔科夫模型. 它有一个基本的假设: 当前词的出现概率仅仅与前面 n-1 个词有关. 因此候选 query C 出现的可能性大小为:

$$P(S) = P(W_1, W_2, \dots, W_k) = \prod_{i=1}^k P(W_i | W_{i-n+1}, \dots, W_{i-1})$$

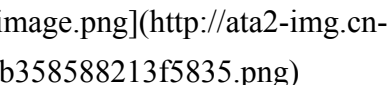
离线基于 7 天的淘宝搜索 Query log, 使用 srilm 工具生成了 trigram 语言模型, 用于在线计算候选 query 的语言模型概率.

### 3.5.4.4 LSTM 语言模型

在计算候选 query C 出现的可能性大小时, 传统的 N-gram 语言模型由于仅仅考虑了前面的 n(n≤4) 个字, 在计算整个 query 的概率时会存在一定的偏差. 为

图 3.39

此我们尝试使用 LSTM 语言模型来计算整个 query 的概率. LSTM 语言模型计算整个 query 的概率示意图如下图所示.

在每一个时间步  $t$ , 都会根据前面所有时间步的语义信息来计算当前 word 出现的概率. 最终通过各个时间步的概率  $\log$  相加即是当前句子出现的概率. 通过上面的计算可以发现, 因此相比于  $n$ -gram 语言模型, LSTM 语言模型能够根据前面所有字的信息来计算下一个字的概率.  (http://ata2-img.cn-hangzhou.img-pub.aliyun-inc.com/e3383a51bcf910e47b358588213f5835.png)

$$P(W_1, W_2, \dots, W_T) = \prod_{i=1}^T P(W_i | W_1, \dots, W_{i-1})$$

#### 3.5.4.5 基于 Seq2seq 的拼写纠错

前面两个版本的候选 query 空间是通过先验知识和 search log 统计确定的, 因此只能覆盖一些常见的错误形式. 但是实际系统中错误的 query 可能是各种形式, 导致统计模型覆盖不到. 比如: 运动鞋  $\rightarrow$  运动鞋. 为了解决这个问题, 我们实验了端到端的 Seq2Seq 拼写纠错模型

#### 3.5.4.6 模型结构

sequence-to-sequence(序列模型) 的输入和输出都是一个序列. 这种模型的一个重要特点就是输入和输出序列的长度是变长的. 而且由于是端到端的结构, 该

模型能够减少很多人工处理以及人为制定规则的工作. Seq2Seq 模型结构通常由 Encoder 和 Decoder 两部分组成, 而且一般通过 RNN 来实现对应的功能.

在用户输入错误的 query 的情况下, 求解出对应的正确 query 也是一种 encoder-decoder 过程. 模型 encoder 的输入是用户的原始 query. 模型 decoder 的输出是对应正确的 query. 整体结构图如下图所示.

图 3.40

#### 3.5.4.7 encoder 网络

该部分主要是用来将输入的句子按照顺序进行编码, 直到最后输出表示整个句子的语义向量  $C$ . 形式化定义为:

$$h_t = f(x_t, h_{t-1})$$

$$c = \phi(h_1, \dots, h_T)$$

其中  $x_t$  为当前时刻的输入,  $h_{t-1}$  为 lstm 上一个时间步的输出.  $C$  一般是 RNN 最后一个时间步的输出, 或者是各个时间步输出的加权和.

由于 RNN 的特点是把前面每一步的输入信息都考虑进来, 因此理论上这个语义向量  $C$  包含了整个句子的语义向量.



### 3.5.4.8 Decoder 网络

该部分主要是在根据 encoder 网络输出的代表句子语义的向量  $C$ , 通过另外一个单独的 RNN 网络, 来解码获取对应的输出序列. 形式化定义为:

$$h_t = f(h_{t-1}, y_{t-1}, c)$$
$$P(y_t | y_{t-1}, \dots, y_1, c) = g(h_t, y_{t-1}, c)$$

其中,  $h_{t-1}$  为 Decoder 阶段上一个时间步的隐向量输出. 而  $y_{t-1}$  为上一个时间步预测的输出符号. 而  $y_t$  为根据上一个 RNN 时间步的隐状态  $h_{t-1}$  以及上一个时间步预测的输出字符  $y_{t-1}$ , 并结合整个句子的语义向量来预测当前时间步的输出符号  $y_t$ .

### 3.5.4.9 实验细节

**3.5.4.9.1 Char-based embedding** 为了方便处理错误的英文单词以及减少模型复杂度, 在进行 encoder 和 decoder 过程中, 我们选取的是 char-based word embedding.

首先, 即使是一个在词典中从未出现过的错误英文单词, 根据字符维度的 embedding 并通过 lstm encoder 也能够正确获取到输入 query 对应的向量表示.

其次, 在 decoder 阶段, 通过使用 char-based embedding, 在每个时间步预测下一个 word 的过程中, softmax 的量级也只是 K 级别 (常见的中文字符 +26 个英文字符), 相比于字维度以及词维度几十万维的参数空间, char-based embedding 能够显著的减少模型复杂度.

**3.5.4.9.2 训练样本** 我们主要从以下两个部分构造基于 seq2seq 拼写纠错模型的训练样本.

#### 1. 基于 HMM 版本线上覆盖的数据

通过离线埋点, 获取线上纠错覆盖的 wrong query? correct query 的 query 纠错 pair 作为模型的训练数据.

## 2. 人工构造的训练样本

仍然是从 query log 中, 抽取没有发生错误的 query, 并随机更改其中的一个 word 来构造训练样本. 同时, 为了让模型学习正确 query 本身的信息, 我们抽取了 Query log 中正确 query 以及其自身作为目标也作为训练样本的一部分.

### 3.5.5 模型效果及上线优化

#### 3.5.5.1 模型效果

通过人工标注构建了 2500+ 的错误 query 以及对应的正确 query 作为验证集. 在验证集的基础上, 我们对比了各个模型的效果. 在验证集上的召回率指标如下表所示.

图 3.41

从实验效果上来看, 相比基本的 HMM+N-gramm, 通过增加 LSTM 语言模型分数以及使用 seq2seq 模型都能显著提高 spellcheck 的召回率.

同时通过实际的 case 发现, 基于 Seq2Seq 的模型能够解决部分基于 HMM 版本没有覆盖的问题. Eg:

图 3.42

### 3.5.5.2 LSTM 上线

HMM + LSTM 语言模型版本, 目前正在线上做 `bts`. 前期由于在线 `lstm forward` 比较耗时, 经过优化, 最终达到了上线的性能要求. LSTM `forwrad` 的性能优化指标为:

图 3.43

### 3.5.5.3 服务调用

通常针对 `query` 的处理都可以通过离线 `cache` 的方式来搞, 不过针对纠错 `query` 来说, 由于错误 `query` 的形式多种多样, 而且 `pv` 不会大都不会很高. 因此完全使用 `cache` 的方式不合适, 还是需要配合在线纠错来提高用户体验.

1. 基本的 HMM + Ngram 语言模型版本目前已经在线全量, 实际纠错的效果可以在 taobao 直接搜索查看; 或者通过: [Matrix Q Demo](<http://sqi.alibaba-inc.com/MatrixQ/index.htm?id=3&query=xinkuanlianyiqun>) 查看更多详细的信息:
2. 为了满足业务方的直接调用, 我们提供了: [basic\_service 的调用接口]([http://gitlab.alibaba-inc.com/newqp/basic\\_service/wikis/Rewrite](http://gitlab.alibaba-inc.com/newqp/basic_service/wikis/Rewrite) )
3. 如果需要根据业务自己的 log 数据进行纠错数据建模, 请参考: [OpenQP wiki]([http://gitlab.alibaba-inc.com/newqp/openqp/wikis/spellcheck\\_rewrite](http://gitlab.alibaba-inc.com/newqp/openqp/wikis/spellcheck_rewrite))

附录: 欢迎体验 Matirx Demo: [<http://sqi.alibaba-inc.com/MatrixQ/index.htm?id=3>](<http://sqi.alibaba-inc.com/MatrixQ/index.htm?id=3>)

参考:

- Xie Z, Avati A, Arivazhagan N, et al., 2016 Neural Language Correction with Character-Based Attention.
- Bahdanau et al., 2014. Neural Machine Translation by Jointly Learning to Align and Translate
- Cho et al., 2014 . Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation
- qSpell : Spelling Correction of Web Search Queries using Ranking Models and Iterative Correction
- Peter Norvig. 2007. How to Write a Spelling Corrector. On <http://norvig.com>. Very nice summary with straightforward Python demo code.
- I Sutskever, O Vinyals, QV Le. Sequence to sequence learning with neural networks

- Sundermeyer M, Schlüter R, Ney H. LSTM neural networks for language modeling[C]//Thirteenth Annual Conference of the International Speech Communication Association. 2012.

## 3.6 语义相关性

传统的搜索文本相关性模型，如 BM25 通常计算 Query 与 Doc 文本 term 匹配程度。由于 Query 与 Doc 之间的语义 gap, 可能存在很多语义相关，但文本并不匹配的情况。为了解决语义匹配问题，出现很多 LSA, LDA 等语义模型。随着深度学习在 NLP 的应用，在 IR 和 QA(问答系统) 中出现了很多深度模型将 query 和 doc 通过神经网络 embedding，映射到一个稠密空间的向量表示，然后再计算其是否相关，并取得很好的效果。本文调研了微软，IBM Watson 实验室、Google 等在这方面的一些工作，并介绍我们在淘宝搜索上做的些工作。□□□

### 3.6.1 DSSM、CDSSM, LSTM-DSSM 及相关系列工作

微软的 DSSM 及相关系列模型是深度语义模型中比较有影响力的。集团内 PS 上有 DSSM 分布式实现, 而且也有多业务应用: <https://www.atatech.org/articles/67415> DSSM 首先将 query 和 doc 表示成一个高维且稀疏的 BOW 向量，向量的维度即词典的大小，每一维表示该 term 在 query 或 doc 中出现的频次；如果向量每一位直接用单词，会出现维度非常高，而且对一些未登录词无法处理。作者做了一个非常有用的 trick word-hash: 将每个单词表示成一个 letter-tri-gram 的序列，例如：boy 切分成 #-b-o, b-o-y, o-y-#，然后再表示成 letter-tri-gram 向量。把每个单词向量累加起来即表示整段文本的向量。

然后，通过几层全连的网络连接将这个高维稀疏向量压缩成一个稠密低维向量，在这个向量空间内，通过计算 query 与 doc 向量的 cosin 相似度来衡量相关程度。训练的目标是对同一 query 下取 1 个点击 doc 作为正样本，随机 4 个未点击 doc 作为负样本，让正负样本的区分尽可能大：

由于 DSSM 对文本 embedding 时没有考虑 term 的顺序信息，又陆续提出了




图 3.44

采用 Convolution 和 LSTM 对文本 embedding, 可以保留词序信息。其中, Convolution 是实现方式通过对 query 或 doc 用固定大小滑动窗口取片段, 对每个片段内文本用 word-hash+dnn 压缩, 然后取 max-pooling 表示整个 query 或 doc 向量。

此外, 无论是 Convolution 还是 LSTM 对文本 embedding, 都涉及到通过词或局部片段的向量生成整个句子的向量, 比较简单粗暴的方法是直接取 sum、avg

图 3.45

或者 max 等。微软的学者们进一步做了改进，提出利用 Attention 机制来学习各个词组合成句子向量的权重。以 LSTM-DSSM 为例，LSTM 在每个时间步 (term) 上输出的隐向量  $h$ ，输入给一个 attention 网络  $s(h)$ ，输出权重后 softmax 归一，然后对每个词的隐向量加权平均生成句子向量。 $s(h)$  的参数和相关性目标一起来训练。这种 Attention 机制也比较弱，因为不同的 query 对同一个 doc 的“关注”点可能是不一样的，这种方式只能对 doc 生成唯一的向量。

图 3.46

最近，微软的学者们又提出了一个观点：query 与 doc 的相关程度是由 query 里的 term 与 doc 文本精准的匹配，以及 query 语义与 doc 语义匹配程度共同决定。而且，term 匹配与 term 在 doc 中的位置和紧密度有较大关系。因此，他们用一个 local model 来表达 term 匹配程度，distribute model 表达语义匹配程度，把

这两个子模型放在同一个模型来训练。distribute model 类似与 DSSM 来学习语义匹配关系。Local model 的输入是一个  $n_q * n_d$  的矩阵  $m$ ,  $n_q$  是 query 中 term 个数,  $n_d$  是 doc 中 term 个数, 位置  $m(i, j) = 0 \text{ or } 1$  表示 query 里的第  $i$  个词是否与 doc 里的第  $j$  个词匹配, 对这个输入矩阵通过 convolution 抽取特征并向量化。据其实验结果, 这种结合 term 匹配信息的模型效果要优于 DSSM 等语义模型。

□□□

图 3.47

### 3.6.2 Google 相关工作

Google 的学者在用 convolution 对文本向量化是相比 CDSSM 做了些改进。Convolution 的方法参考了 Nal Kalchbrenner 等对文本用卷积来做分类的方法。□

首先, 对句子中的每个词做 embedding, 然后将词的 embedding concat 起来组合成一个矩阵, 有点类似图像的表达。然后, 在这个矩阵上通过不同 feature map 抽取特征, 然后 pooling 生成一个维度的向量来表达句子。对 Query 和 Doc 的语义向量, 再通过一个 bilinear 的模型计算其语义相似度:  $\text{sim}(x_q, x_d) = x_q * M * x_d$ 。



最终，语义相似度与其它相关排序特征，以及 query 和 doc 向量一起作为决定排序的因素，通过 pointwise 的 DNN 模型来训练。

图 3.48

### 3.6.3 IBM Waston 实验室相关工作

问答系统有很多种类型，其中给定一个 Question 和候选 Answer，从候选 Answer 中挑选最合适的答案，这个过程与信息检索中的相关性模型非常相似。Watson 实验室在 InsuranceQA 数据集实验了上述类似的模型，并综合 CNN 和 LSTM 的优势，提出了几种有意思的混合模型：

(1) Convolutional-pooling LSTM □ 用一个 Bi-LSTM 作为 word embedding 的方法，然后 word embedding concat 成矩阵表达句子，用卷积来抽取组合特征作为 question 和 answer 的向量表达，再计算 cosin loss.

(2) Convolution-based LSTM 先对原始文本用卷积捕捉局部的 N-gram 信息，然后在这个基础上用 Bi-LSTM 来学习更大范围的上下文依赖关系。

□□ (3) Attentive-LSTM □ 相比 LSTM-DSSM, 在 Attention 机制上做了些改进，与 NMT 的 Attention 机制接近，即：通过 Answer 中的词向量加权平均生成整个 Answer 的向量时，每个词的权重是由 Question 向量和词向量来决定的。Question 的表达仍由其所有词向量的 avg 或 sum, max 来表示。□

图 3.49

图 3.50

### 3.6.4 其它相关工作

上述工作主要集中在如何更好生成 Query 和 Doc 向量表达，如何设计两个向量 comparison function 以计算相似度也有很多种方法。Shuohang Wang 总结了 6 种方法：NN, NTN, EUCCOS, SUB, MULT , SUBMULT+NN。分别对 query 和 doc 向量计算乘、减、欧式距离、cosin、bilinear、concat，以及这几种计算的组合。□

图 3.51

图 3.52

另外在机器阅读理解也有很多类似工作，本文就不展开描述了。下面介绍下我们的相关工作。□□

### 3.6.5 我们的工作

我们对淘宝搜索做了大量的语义改写后 (参加 Search2.0 之 QueryRewrite, <https://www.atatech.org/articles/66690>), matching 不仅局限于 term 的匹配了, 下

面分别从数据和模型介绍下我们的工作。

### 3.6.5.1 训练数据

深度模型通常大量的训练数据，而对商品搜索相关性这个问题，获取大量高质量训练数据并不容易。网页搜索通常直接采用点击数据作为是否相关的 label，在商品搜索上不是很有效：用户点击行为与价格、图片、个性化偏好等很多因素相关，仅依赖点击数据对相关性样本有太多噪声；而采用人工标注数据，准确率相对较高，但受时效性、成本等因素限制较大。因此我们结合了这两种方式来获取训练数据：

1. 用多维度规则对行为数据采样，获取大量 (亿级别) 准确率相对较低的训练数据，先用这些数据 training 一个较好的模型；
2. 再采用数量相对少 (100w)、准确率高的人工标注数据 fine-tuning 之前 training 好的模型。

### 3.6.5.2 模型

模型设计主要考虑的几个因素：

1. 淘宝上 Query 和商品标题存在大量长尾词，尤其大量数字和英文组合的货号、型号、容量等，分词无法穷尽。仅通过词来对 query 和标题 embedding 会损失很多信息，需要考虑字符维度。
2. 商品除了标题外了，还有图片、类目、属性等信息可以利用。
3. 工程实现线上计算要轻量，两个向量的 compare function 要控制计算复杂度。

我们现在采用的模型如下：□□□

1. 对 Query 和标题向量我们采用 DNN + Char-LSTM 组合的方式：DNN 能高效地学到 TOP 词的 embedding, Char-LSTM 能捕获到较长尾的字符组合。引入 Char-LSTM 后模型比较难训练，我们使用 query 和标题文本语料 pretraining LSTM-AutoEncoder, 获得比较好的初始参数；同时 TOP 词的 embedding 采用 word2vec 初始化，模型能更快收敛。
2. 在商品标题的 embedding 上增加了一个类目预测的 multi-task, 使得不同类目的商品在向量空间内有更好的区分度，对模型效果和收敛速度都有比较好的提升。
3. online ranking 对 latency 要求比较高，除了工程优化外，模型上也有优化空间。在我们数据上实验发现 compare function 中全连层的深度和宽度对模型影响比较大。全连层宽一些效果会比较好，但计算量增加会很大；借鉴 ResNet 全连层设置窄一些，并加深模型，可以保证效果同时较大减少计算量。

我们抽样部分 query 抓取线上排序结果, 与该模型排序后 TOP30 人工评测 GOOD 比例提升 1.31%。该模型现在正在线上 BTS 中。□

体验 demo:<http://sqi.alibaba-inc.com/MatrixQ/index.htm?id=28>□□5.3 后续计划商品除了标题和类目，图片也是很重要的信息来源，后续加入图片信息，同时也在尝试用 query 和商品向量做召回，实现 multi-modal 检索。

另外，Attention 机制也是一个被证明重要的提升点。受限于线上 ranking latency 的要求，不可能对每个商品标题根据 query 来计算其“关注”的部分，但可以引入一些 self-attention 的方法来生成更好的标题向量。□□□ 参考文献：

1. Shen, Y., He, X., Gao, J., Deng, L., & Mesnil, G. (2014). A Latent Semantic Model with Convolutional-Pooling Structure for Information Retrieval (pp. 101–110). Presented at the the 23rd ACM International Conference, New York, New York, USA: ACM Press. <http://doi.org/10.1145/2661829.2661935>
2. Services, E. U. C. (2014). Learning Deep Structured Semantic Models for Web Search using Clickthrough Data, 1–8.

3. Deep Sentence Embedding Using Long Short-Term Memory Networks: Analysis and Application to Information Retrieval. (2016). Deep Sentence Embedding Using Long Short-Term Memory Networks: Analysis and Application to Information Retrieval, 1–25.
4. Zhai, S., Chang, K.-H., Zhang, R., & Zhang, Z. M. (2016). DeepIntent: Learning Attentions for Online Advertising with Recurrent Neural Networks (pp. 1295–1304). Presented at the the 22nd ACM SIGKDD International Conference, New York, New York, USA: ACM Press. <http://doi.org/10.1145/2939672.2939759>
5. Mitra, B., Diaz, F., & Craswell, N. (2016). Learning to Match Using Local and Distributed Representations of Text for Web Search, 1–9.
6. Improved Representation Learning for Question Answer Matching. (2016). Improved Representation Learning for Question Answer Matching, 1–10.
7. Feng, M., Xiang, B., Glass, M. R., Wang, L., & Zhou, B. (2015). APPLYING DEEP LEARNING TO ANSWER SELECTION: A STUDY AND AN OPEN TASK , 1–8.
8. Severyn, A., & Moschitti, A. (2015). Learning to Rank Short Text Pairs with Convolutional Deep Neural Networks (pp. 373–382). Presented at the the 38th International ACM SIGIR Conference, New York, New York, USA: ACM Press.
9. Kalchbrenner, N., Grefenstette, E., & Blunsom, P. (2014). A Convolutional Neural Network for Modelling Sentences
10. Wang, S., & Jiang, J. (2017). A COMPARE-AGGREGATE MODEL FOR MATCHING TEXT SEQUENCES, 1–11.
11. Lin, Z., Feng, M., Santos, dos, C. N., Yu, M., Xiang, B., Zhou, B., & Bengio, Y. (2017). A STRUCTURED SELF-ATTENTIVE SENTENCE EMBEDDING, 1–15.

## 3.7 Multi-Modal Matching Model

### 3.7.1 背景

在淘宝商品搜索系统中，除了词维度的 matching 外，在语义维度 matching 我们分别在 query 改写和深度语义相关性模型上做了一些工作，参考 [1][2]。这些工作主要还是局限在文本域上。然而，商品搜索相比较网页搜索一个很大的差别在于，商品除了拥有标题文本以及结构化属性外，每个商品还必须包含商品主图：一方面，这些图片是对文本信息很好的补充；另一方面，我们认为对于风格、款式等一些抽象的概念，图片比文字可能更具有表达力。如下图，这些 Query 下对应的商品通过文本域上的改写和语义相关性，已经无法召回和计算相关程度：Query 中标红的词无法匹配，标题中也不存在语义相关的描述。□□

图 3.53

因此，如何融合商品上文本、图像多个模态的信息来表达的商品，并解决 Query 与商品 Matching 问题，是一个很有价值和意思的课题。我们和 idst 图像团队的同学一起做了一些探索。

图 3.54

### 3.7.2 模型

受益于 Deep Learning 强大的表达能力，不仅在文本、图像这两个领域取得很大进展，一些跨模态的应用也被越来越多的研究，包括：image caption, visual qa, sentiment analysis, content-based multi-modal retrieval 等。基本思路是将自然语言和图像两种模态的数据经各自的编码模型转换为同一特征空间的向量，然后针对特定任务设计特征向量之间 loss 函数。

我们所采用的网络设计如下图所示，此系统主要包括：文本编码模型、图像编码模型和图像文本匹配关系的学习，下面逐个进行介绍。□

- 文本编码：文本编码的作用就是将用户搜索 query 转为定长实数向量  $V_{text} \in R^D$ ，该向量能够反映用户语义信息，便于和图像编码向量进行比较。我们在实验中尝试过 LSTM、Char-CNN 等结构，最后综合考虑性能和效率采用了如上图所示的分词 DNN 编码结构。其首先对 query 进行分词，每个分词的 word embedding 初始参数随机；分词长度固定，不足补零，过长直接截断；后续接入两个全连接层，经过 Batch Normalization 和 L2Norm 作为文本特征编码输出。整个文本所有参数 random 初始化，并利用梯度下降进行训练。



图 3.55

- 图像编码：图像编码模型主要是将图像文件转化为能反应图像内容的特征向量  $V_{image} \in R^D$ 。图像 CNN 基础模型采用 ResNet152, 由于图像 CNN 参数过多，如果采用 end2end 方式进行模型训练，batch size 受限于显存大小不能设置太大、同时训练耗时会很长。因此，我们利用淘宝商品主图、以叶子类目为分类目标，对 ImageNet pre-trained ResNet 152 进行 finetune，最终训练得到 CNN 基础特征模型。图文模型中的图像编码特征是在分类 CNN 特征基础上进行 transfer，transfer 和文本编码网络类似，包括全连接层、Batch Normalization 和 L2Norm。
- 图像文本匹配关系学习：我们利用内积来衡量图像和文本之间的相似性，即：□

$$S(V_{image}, V_{text}) = V_{text} * V_{image}$$

整个网络的目标函数采用 ranking loss：□

$$L = \sum \{ \max(0, \mu - S(V_{image}^i, V_{text}^i)) + S(V_{image}^j, V_{text}^i) + \max(0, \mu - S(V_{image}^i, V_{text}^j)) + S(V_{image}^i, V_{text}^j) \}$$

其中  $i, i$  图像和文本是正对样本， $i, j$  是负对样本， $\mu$  为正对和负对之间相似性得分差异性的一个常数阈值，可根据不同应用场景进行设定。通过最小化上述目

标函数，可使得相关的图像和文本之间具有较大的相似性得分，而不相关的图像和文本之间的相似性得分则较小。淘宝商品搜索具体实现：

- 选择服饰行业的 9 个一级类目；
- 图像基础模型即使用淘宝商品主图、以 9 个一级类目对应的叶子类目为分类目标训练得到；
- 图文模型训练数据选自用户搜索点击 log, 第一版 BTS 模型共利用 6000w 左右图文 pair；
- 最终输出特征向量维度为 256 维。

BTS 上线前我们抽取了一批服饰行业 query 进行了相关性评测，评测从图像相关性、商品相关性两个角度进行，结果如下：□ Top1-5 的图像相关性达到 88%，

图 3.56

商品相关性 69%。

### 3.7.3 模型改进 A：挖掘图像信息表达

我们在实验中发现，图像编码特征对 Cross Modal search 的性能影响很大。对于淘宝商品而言，除了主图、多图数据也是对同一商品的更多展示信息。而

且图像基础特征提取可以放在离线进行，因此离线部分模型可以做的相对复杂一些。相比现在 BTS 采用的模型，改进如下图所示：□

图 3.57

商品 item 的多张图像基础特征向量表示为： $\{U_{item}^k\}_{k=1}^K$ ，利用如下 self-attention 机制学习权重，最终得到统一商品向量表示：□□

$$h^k = \sigma(W_u U_{item}^k) \odot \sigma(W_m m_{item})$$

□

$$\alpha_k = softmax(W_h h^k)$$

□

$$V_{item} = F_v \left( \sum_{k=1}^K \alpha_k U_{item}^k \right)$$

□ 其中 □

$$m_{item} = \sum_{k=1}^K U_{item}^k$$

下面是一些利用 visual self-attention 学到的多图权重分布，可以看到一些文字图、细节、吊牌图权重基本会比较小，符合我们的期望。□□

在我们自己的一个评测集上，此改进模型相比线上 BTS 模型效果提升明显，mean rank 指标从 49 降低到 27(越小越好)。




图 3.58




图 3.59

### 3.7.4 模型改进 B: 图文联合解决图像不可识别意图

前面我们主要解决的是图像上可识别的搜索 query，对于一些含有图像上比较难以处理的 tag (e.g. 品牌、型号)，我们的文本编码词典暂时未覆盖。下一步我们要考虑的是同时解决图像不可识别 tag，而这些信息而从商品标题、属性进行挖掘。基本的网络结构如下图所示：□

图 3.60

其中 query 利用 QueryTagging 对分词进行打标，将图像不可识别的 tag 独立进行编码 (对应图中的 Slice)；商品编码信息来源除了图像，同时加入标题、属性等信息。对于 query 中不包括相应 tag，则对应 Slice 编码为全 0，这样采用内积相似性度量情况下，不会影响该 query 其他部分的正常搜索召回。该版本模型还在进行调试训练，实际效果敬请期待。

### 3.7.5 线上实验 □

该模型在线应用面临一个较大的挑战：给定一个 Query 向量，如何快速与海量商品集合计算相似度，确保精度地召回 TOP K 个相似商品。靠谱的主搜工程团队同学已实现了基于图近邻查找算法 NSG, 参考 [3]，从 500w 商品向量中检索 top1200 个商品仅需 4ms, 召回精度 99.8%。而且，与之前主搜的文本倒排

图 3.61

召回完美无缝结合，对算法同学使用起来很方便。BTS 效果：第一阶段覆盖了服务行业下 9 个一级类目的中长尾 Query，占整体 PV 大约 8%。这部分覆盖 Query 因为文本召回结果有限，通过召回了更多相关商品，用户可以有更多优质商品挑选，对于引导成交效果非常明显，成交笔数能提升 10 个点以上：□□

图 3.62

### 3.7.6 总结

一期覆盖了流量较大的服饰行业，后续将继续扩展行业。当然，有些行业可能也不适合做图文检索，例如：3C、家电等行业。

现在商品的文本和图像特征融合还做的比较简单、粗糙，后续会继续深耕模型，期望能做到商品文本、属性和图像真正的互补和融合；同时，商品的表达不仅限于召回阶段的应用，后续和语义相关性模型融合实现端到端的 Multi-modal Matching。

参考：

1. <https://www.atatech.org/articles/66690> Search2.0 之 Query Rewrite
2. <https://www.atatech.org/articles/85492> DL 语义相关性以及在淘宝搜索中的应用
3. <https://www.atatech.org/articles/95738> □□[4] Nam, H., Ha, J.-W., & Kim, J. (2016). Dual Attention Networks for Multimodal Reasoning and Matching
4. He, K., Zhang, X., Ren, S., & Sun, J. (2017). Deep Residual Learning for Image Recognition
5. Morency, L.-P. (2017). Tutorial on Multimodal Machine Learning
6. Zadeh, A., Chen, M., Poria, S., Cambria, E., & Morency, L.-P. (2017). Tensor Fusion Network for Multimodal Sentiment Analysis

## 参考文献

- [1] C. Burges, T. Shaked, etc., Learning to rank using gradient descent. In Proceedings of the 22nd international conference on machine learning, ACM



## 第四章 排序算法

### 学习目标与要求

用户在淘宝输入关键词“手机”后，搜索引擎召回了几百万以上的标题中带有“手机”的商品，这些商品中有大家常见理解的手机，也有“手机壳”，“手机配件”，“手机充电器”等商品，找到真正的手机，是在第 2 章的搜索相关性中解决的问题。再接下来，真正的手机也是成千上万的，这些商品如何排序，取决于多种因素，分为个性化相关的（下一章讲述），与非个性化相关的。本章主要讲述商品非个性化的排序因素，除比较传统的 Query-Item 的 ctr 以外。淘宝商品搜索排序与传统网页搜索对比，存在了很多不同的地方，这也决定了算法有很大的不同。首选用户所有的路径都在淘宝内部，可以拿到所有的数据，对其建模，再应用于各个环节。

### 4.1 商品详情页满意度模型

#### 4.1.1 为什么需要详情页满意度

用户在搜索上的整个购物路径大概包括搜索 query，浏览 SRP 点击宝贝，浏览宝贝详情页，成交四个环节。我们要做的事情就是优化用户购物路径上的每个环节，让用户最终能够顺利成交。

我们以前的工作主要围绕在上述购物路径上的两个环节：

1. 用户搜索 query 到宝贝点击，这一步我们通常用 ctr (click through rate) 来衡量，我们通过各种算法把点击率高的宝贝展现给用户来提升 ctr。
2. 宝贝点击到成交，这一步我们通常用 ipvr (ipv conversion rate) 来衡量，我们通过各种算法把 ipv 转换率高的宝贝展现给用户来提升 ipvr。

粗看，上述两个工作好像已经覆盖了从最开始的搜索 query 到最后成交的整个路径，但细看就会发现，在 ipvr 这一步我们跳过了用户在宝贝详情页上的信息。那么我们是否应跳过这个环节，只看用户最终是否成交了呢？答案是否定的，因为成交还是非常稀疏的，这个稀疏性可以通过下面两组数据来体现：

1. ipvr 只有 2%，即 100 个 ipv 只能换来 2 个成交，绝大部分 ipv 最终无法成交。这些没有成交的 ipv 没有区分度，都被认为是用户不满意的。
2. 一天有成交的宝贝有 500 万，30 天有成交的宝贝有 6000 万，大部分宝贝 30 天都没有一笔成交；有成交的宝贝中，95%++ 的宝贝成交也在 10 笔以下。

通过上面两组数据可以看到如果仅用成交信息来选取优质宝贝显然是不够的。既然成交是不够的，那么用户在详情页上发生的信息是否够充分呢，是否能够反映用户对宝贝的满意程度呢？

### 4.1.2 详情页信息

详情页上的信息主要两类：

1. 详情页上控件的点击信息，比如下图中的红框中各种控件的点击信息都能捕捉到。

相对于成交，详情页点击信息还是非常丰富的。一天搜索引导的详情页点击为 2 亿，而成交只有 500 万，详情页点击是成交的 40 倍，平均 1 个 ipv 有 1.4 个详情页点击。

图 4.1

详情页行为除了丰富性，是否能反映用户的满意程度？我们假设用户如果成交，那么肯定是满意度的。我们来分析一下详情页上各种点击行为和成交的关系，是不是发现了详情页行为的 ipv 更有可能成交？

每一种点击行为发生与否，后续的 ipv 转换率差异（红色是发了该行为的转换率，蓝色是没有发现该行为的转换率，总体的平均转换率为 2%）：

可以看到，对于每一种详情页点击，ipv 上发生了该行为的转换率都要高于



图 4.2

没有发生该行为的概率,意思就是说用户在详情页上点了肯定比没点好的,用户的点击也是有成本,不会没事瞎点。对于几个比较特殊的详情页点击,如“加入购物车”,“立刻购买”,后续的 ipv 转换率是非常高的。

- 详情页停留时间

这个也非常直观,用户在详情页上花的时间越多,表明该详情页越能吸引用户。在互联网上,页面停留时间,网站停留时间也是各个网站非常关注的指

图 4.3

标，用户在网站上花更多的时间意味着网站的黏性越高，意味着网站能够为用户提供更有价值的内容和服务，相应的用户在网站上转化的机会也越多。

但不幸的是我们并没有现成的详情页停留时间，进入详情页的时间是有日志记录的，但离开详情页的时间并没有记录，那么怎么得到用户在详情页的停留时间？首先我们看一下用户在搜索上的行为示意图：用户来到搜索，会有一个或者多个搜索意图，每一个搜索意图我们称之为一个搜索 session，用户在一

图 4.4

个 session 中会使用一个或者多个 query 来表达他的需求，同一个 session 内的不同 query 表达的需求主题上应该是类似的。用户搜索了 query 后，浏览系统返回给他的搜索结果，在结果页上点击他感兴趣的宝贝，进入宝贝详情页，然后在详情页上发生各种点击动作。浏览完一个宝贝详情页后，继续点击，浏览下一个宝贝或者换一个 query 继续搜索。

如果用户的行为完全是按照我们上面的设计，是一个串行的模式，那么按

照 session 为 key 把用户的所有行为串起来，可以得到如下的行为时间序列： 在

图 4.5

这个时间序列中，我们可以计算出每一个详情页的停留时间：

宝贝 1 详情页停留时间 =  $t_4 - t_3$

宝贝 2 详情页停留时间 =  $t_6 - t_5$

宝贝 3 详情页停留时间 = ? -  $t_6$ ，无法计算，session 结尾点击。

当然实际上有些用户的行为并不一定是串行的，比如用户可能会搜索了一

个 query 后，连续打开多个详情页，再一个一个慢慢看。

对于每一个 ipv 我们都能计算出的页面停留时间，信息非常丰富。

那么详情页停留是否能反映用户的满意度呢？同样我们来看一下详情页停留时间和转换率的关系：

上面的数据可以看出：

1. 停留时间越长，ipv 转换率越高
2. 最前面的一个停留时间为-1 的表示的是 session 结尾点击，可以看到，session 结尾点击转换率非常高，符合预期。
3. 前面几段（0-5 秒，5-10 秒，10-15 秒）的 ipv 转换率并没有完全单调递减，这个有可能就是前面说的当用户行为不是串行模型下产生的数据。

### 4.1.3 如何建模

有了详情页点击信息，也有了详情页停留时间，下面说下如何使用这些信息来计算用户对于一个 ipv 的满意程度。

怎么样算是用户满意呢？我们模型的目标是什么？这是我们建模遇到的第一个问题。答案是我也不知道用户是否满意，我们并没有办法让用户显示的告诉我们他是否对满意，所以我们还是只能认为如果用户最终成交了，那么用户肯定是满意的，即我们的模型目标是这个 ipv 最后是否成交，一个二分类问题。

对于这个二分类问题，我们选取的是常用的逻辑回归模型。

特征全为 0/1 特征 (binary feature)，对于详情页点击类特征，1 表示 ipv 上发生了该类点击行为，0 表示没有发生该类点击行为；对于停留时间特征，我们把连续的停留时间离散化成 50 档，停留时间落在某一档，则该档特征为 1，其余档特征全为 0。



## 4.2 用户浏览模型 & 用户点击满意度模型

用户在搜索上的整个购物路径大概包括搜索 query，浏览 SRP 点击宝贝，浏览宝贝详情页，成交四个环节，搜索记录的日志数据在用户在整个搜索流程中的各个环节都存在一定的精度的损失，从输入 Query 到搜索结果的展示，再到后续的点击和成交，只有有效的 PV 才有可能获得点击，同理，也只有有效的 IPV 才能获得成交，而日志中记录了大量的无效 PV 及 IPV，因此，我们的目标就是消除这些无效信息对模型的影响。此外，手机端的搜索，由于搜索结果页信息过少，因此分析用户在详情页的行为显得更加必要。通过对用户在详情页的行为进行建模，区分出哪些点击是无效点击，哪些点击是有效点击是详情页满意度模型所需要达到的目标，而最终区分出来各种点击之后，我们需要将通过详情页满意度模型学得的有用信息作用于我们最终的 cvr 预估模型，如下图所示，我们通过 UBM (User Browsing Model) 和 UCM (User Click Model) 来分别量化有效的 pv 和有效的点击。

图 4.6

### 4.2.1 UBM(用户浏览模型)

搜索的日志提供了大量非常宝贵的相关性信息，然而日志中记录的 pv 信息与实际的 pv 与一定的差距，日志中记录的展现宝贝在实际上用户并不一定真的看到，排序的位置对用户是否点击也存在一定的影响。User Browsing Model(UBM)旨在消除这些因素对 ctr 的影响，使得根据历史信息得到的 ctr 能更好的拟合宝贝原始的 ctr。下图为各个位置的 CTR 的分布

图 4.7

UBM 通过对用户行为数据的建模，构建出 Query 到宝贝历史 CTR 【History CTR】与（Query 到宝贝展示概率【Examination】和 Query 到宝贝真实 CTR 【attractiveness】）这二者之间的关系。 $\text{History CTR} = \text{Examination} * \text{attractiveness}$ 。在后续的使用中，UBM 得到的模型可以有两种使用方式：1) 宝贝真实 CTR 【attractiveness】可以用于宝贝的 CTR 预估模型；2) 宝贝的展示概率【Examination】可以用于修正宝贝预估模型中的目标。

考虑这样的场景，当一个用户输入一个 query 时，会由引擎返回给他一个结果列表，该用户从第一个结果开始按照排序结果的顺序进行浏览，对于每一个位置的结果，用户需要决定是否仔细查看这个结果，如果用户对该结果进行了

一次点击，则说明当前结果对于用户具有一定的吸引力。从这个场景中我们可以看出，整个点击的行为串需要两次的判定，首先是“是否看到”(Examination)，接着是“是否点击”(attractiveness)。UBM 中假设用户是否看到的概率与排序的位置  $r$  以及距离上一次点击的距离  $d$  有关，之所以考虑距离上一次点击的距离是基于这样的假设：当用户看到一大串的不相关的结果时，用户更倾向于放弃此次搜索。此外，是否点击的概率则是由当前的  $query(q)$  与当前展示的结果 ( $u$ ) 决定的，因此 Examination 和 Attractiveness 可以定义如下：

$$P(a|u, q) = \alpha_{uq}^a (1 - \alpha_{uq}^a) \quad (4.1)$$

$$P(e|r, d) = \gamma_{rd}^e (1 - \gamma_{rd}^e) \quad (4.2)$$

那么，一次点击行为序列则可以使用联合概率  $P(c, a, e|u, q, d, r)$  来表示：

$$\begin{aligned} P(c, a, e|u, q, d, r) &= P(c|a, e)P(e|r, d)P(a|u, q) \\ &= P(c|a, e)\gamma_{rd}^e(1 - \gamma_{rd})^{1-e}\alpha_{uq}^a(1 - \alpha_{uq})^{1-a} \end{aligned} \quad (4.3)$$

为了计算一个观测组  $(c, u, q, d)$ ，则需要分别考虑点击与否的情况，当  $c=1$  时，即当前发生一次点击，我们可以以此推断当前用户既看到了这条结果 ( $e=1$ )，同时这条结果对用户有一定的吸引力 ( $a=1$ )，与此相反，若当前没有发生点击，则说明当前结果用户没看到或者没有吸引力，形式化的表示以上两种情况如下：

$$P(c = 1|u, q, r, d) = \alpha_{uq}\gamma_{rd} \quad (4.4)$$

$$P(c = 0|u, q, r, d) = 1 - \alpha_{uq}\gamma_{rd} \quad (4.5)$$

## 4.2.2 UCM(用户点击模型)

用户在详情页的行为可以很好的反应出这次  $ipv$  的质量，例如：用户在进入详情页之后立刻点击了返回，那么用户这次点击可能成交的概率就相对较小，而如果用户进入详情页之后查看了大图，和卖家进行了旺旺交流，那么用户这次点击可能成交的概率就更大，我们则称后面这一次的点击的质量更高。那么我们要怎么对这些行为进行建模呢？由于我们往往通过一次搜索是否引导成交

衡量这次搜索是否满意，因此，我们对详情页行为进行建模时，也是以单次 ipv 是否最终有引导成交作为目标对详情页的各种行为进行建模。同时，除了用户在详情页的虚拟点击行为外。由于在无线的应用场景中，还有一个很重要的因素，即详情页的停留时间，pc 上由于浏览器可以开多个 tab，因此停留时间不好计算，而无线端由于其自身特点，我们能够更好的把停留时间这一维度的特征使用进来。下图为各种虚拟点击行为训练出来的权重分布，其中 5,11,17 这 3 个凸起的点分别对应加购，立即购买，和旺信三种虚拟点击的行为。

图 4.8

使用 UCM 对有效点击进行折算，最终得到的有效点击数如下面公式所示：  
其中，参数  $W$  基于下面的目标函数使用 LR 进行求解：

图 4.9




图 4.10

### 4.2.3 模型融合

使用 UBM 可以去除日志中记录的无效 pv, 而使用 UCM 可以预测出每一次 ipv 的质量, 从而对每次点击做一次折算, 消除无效点击, 但是这些都不是我们最终的目标, 我们要做的还是 cvr 预估, 那我们如何将我们通过 UCM 以及 UBM

图 4.11

学得的有用信息最终作用于我们的预估模型中呢？对于这个问题，我们做了以下两个方面的尝试：

### 4.2.3.1 UBM 模型与 UCM 融合统计版

从前面的描述中，我们可以很直接的看到，实际上我们使用 UBM 得到的结果即为用户真实看到的 pv，而使用 UCM 得到的即为真正有效的点击，因此很容易想到的一种融合方式即为使用这两个模型的预测结果对日志中记录的 pv 和点击都做对应的折算，然后使用折算后的历史 ctr 作为宝贝的 ctr 作用于线上，使用这种简单的方式作为最初步的尝试，在 ctr, 转化率等上均有一定收益，也验证了模型融合的可行性。

### 4.2.3.2 UBM 模型与 UCM 融合模型版

过使用统计版验证有效性后，最终我们还是需要做模型的融合，需要解决的主要问题还是如何将我们使用 UBM 和 UCM 得到的有效信息应用到我们最终的预估模型中。传统的 CVR 预估模型中，在训练目标中使用 pv 作为负样本，点击与成交作为正样本，如前文描述中可以看出，由于日志中的 pv 和点击都是包含噪音数据的，这样自然会使得模型训练的目标本身就存在一定的偏差，而模型训练的目的实际上是无限的接近训练时的目标，但是如果目标本身存在偏差，那即使模型训练的再接近目标仍然无法达到最好的效果。在融合模型的过程中，我们实际上是使用前面两个模型的训练结果来作为最终预估模型的目标，这样做的好处在于通过消除目标中的有偏信息，修正模型的目标。提高模型训练优化能够达到的上界，使得预估模型的优化空间得到很好的提升。具体的融合方式如图所示：

如图所示，我们保留了传统预估模型的基本框架，但同时我们需要把我们通过其他模型训练得到的有用的信息应用过来，我们的作用方式是，通过使用 UBM 和 UCM 的预测结果作为预估模型最终的训练目标来达到模型融合的效果。



图 4.12

#### 4.2.4 实验结果

最终融合模型在线上的 BTS 效果在成交转化率这些相应指标上都得到了一定的收益, 如下图所示:

分别使用三种方法与基准桶进行对比实验, 方法一为统计版 CVR, 即直接使用历史 30 天的点击和成交信息作为宝贝的 CVR 预估的得分作用于线上, 后两种方法为前文提到的 UBM 和详情页满意度模型融合的统计版和模型版, 从对比数据中我们可以看出, 我们最终的融合模型版相比纯统计版的 CVR 模型在转化率方面有较大的提升 (与基准桶的 gap 由 0.3% 提升至 1.67%), 同时, 由于我们结合的是详情页满意度的模型, 因此我们最终的融合模型在 CTR 方面不如原始的统计版 CVR 模型, 但是融合模型主要是通过提升 ipv 转化率达到最终转化率提升的效果, 从对比效果中可以看出, 融合模型在 IPV 转化率上相比原始的 CVR 模型也有很大的提升。

模型融合需要解决的主要问题还是如何将我们使用 UBM 和 UCM 得到的有效信息应用到我们最终的预估模型中。传统的 CVR 预估模型中, 在训练目标中使用 pv 作为负样本, 点击与成交作为正样本, 如前文描述中可以看出, 由于日

图 4.13

志中的  $p_v$  和点击都是包含噪音数据的，这样自然会使得模型训练的目标本身就存在一定的偏差，而模型训练的目的实际上是无限的接近训练时的目标，但是如果目标本身存在偏差，那即使模型训练的再接近目标仍然无法达到最好的效果。在融合模型的过程中，我们实际上是使用前面两个模型的训练结果来作为最终预估模型的目标，这样做的好处在于通过消除目标中的有偏信息，修正模型的目标。提高模型训练优化能够达到的上界，使得预估模型的优化空间得到很好的提升。具体的融合方式如图所示：

### 4.2.5 相关文章

- Modeling dwell time to predict click-level satisfaction Youngho Kim, Ahmed Hassan, Ryen W. White, Imed Zitouni WSDM 2014
- A user browsing model to predict search engine click data from past observations Georges E. Dupret, Benjamin Piwowarski SIGIR 2008
- BBM: Bayesian Browsing Model from Petabyte-scale Data Chao Liu, Fan Guo, Christos

Faloutsos KDD 2009

- A Dynamic Bayesian Network Click Model for Web Search Ranking Olivier Chapelle, Ya Zhang

## 4.3 商品网络效应分

### 4.3.1 背景

什么是商品的网络效应？

简单讲，就是利用商品间的网络关系来对商品的质量做出评价。

长期以来搜索更多关注用户到 query 到商品到成交的整个转化效率，衍生出 CTR、CVR、详情页满意度等方向的产品，取得了重大的突破。随着用户行为越来越丰富，特别在无线上，用户可以很方便的通过点击行为在多个产品间跳转，完成他的购物。我们所采用的数据应该不仅仅来自于搜索，也需要使用来自于全网的用户行为链路。目前庞大的用户群体正在编织起这张巨型的商品关系大网，如何应用好商品关系网络，用于商品质量的评判，是我们需要进行探索的课题。

### 4.3.2 产品策略

基于全网的用户行为数据，用户在一个 session 内的一次购物行为，通常都是先浏览一些商品，最终选定一个商品进行购买。从浏览商品到最终购买商品，其实说明了用户对最终购买的商品的满意度高于最初浏览的商品。这个过程形成了一次有方向的投票行为。众多的用户的购物便形成了众多的投票，彼此交织在一起，成为一个商品间的关系网络。

另外，i2i 这个名词，我们并不陌生。根据用户的共点击、共成交行为构建起了商品的 i2i 的网络。i2i 网络的建立缘于用户在一个 session 内，他所共同点击的商品，共同成交的商品具有一定的共性。这个共性可以是价格，可以是品牌，可以是品质，可以是一个调性，总之用户认为他们具有一定的相似性。我们

也利用了 i2i 的商品的相似性特点来对商品的关联网络进行了进一步的扩展，使之能对更多的商品的质量加以描述。

最终，网络建立起来，采用合适的图模型算法来对各个商品的质量做收敛的计算，得出商品的网络效应分。

商品网络效应分的建立，有效的在全网范围内对商品的质量做出了更好的描述。也使搜索排序和外部产品的连接变得更为紧密和敏感。对于搜索的长期外部性价值都有重要的远期影响。

### 4.3.3 技术实现

#### 4.3.3.1 商品网络关系构建

##### 4.3.3.1.1 基础置信边

1. 投票：某用户在一个 session 内分别点击了 A、B，且只购买了 B，此次行为构成 pairwise 组合，这个行为我们可以理解为 A 商品以自己的价值给 B 商品做了一次背书，我们将其抽象为 A 商品给 B 商品的一次投票。
2. 基础边：基础边是能反映商品间投票关系的有向边。
3. 置信边：某基础边  $A \rightarrow B$  在一个月内出现两次或两次以上我们称其为置信边，如下图所示

置信边真实反映了用户行为，众多置信边构成的网络我们称为基础置信网络。

**4.3.3.1.2 构建过程** 按天统计基础边的 uv 作为计数（已过滤炒信行为。另外，一条边在一天内无论多少人经历过，都只算一个 uv），按月统计基础边的计数作为能否称为置信边的凭证（一条边若想上升成为置信边，需要在一个月至少要有 2 个这个的 uv）。在基础置信网络中，我们将所有置信边的权重统一设为 1。

图 4.14

#### 4.3.3.2 i2i 扩展边

1. 淘系商品的价值体系是分等级的，且单个商品价值在体系中等级是稳定的。
2. i2i 关系判定的相似商品在价值体系中处于相同等级。
3. i2i 扩展边：存在置信边  $A \rightarrow B$ ，若  $B$  与  $M$  是相似商品，则存在 i2i 扩展边

$A \rightarrow M$ ; 若  $A$  与  $N$  是相似商品, 则存在  $i2i$  扩展边  $N \rightarrow B$ 。

基于上述假设, 我们做一个大胆猜想: 存在一种  $i2i$  扩展边, 它同样能够反映商品间的投票关系。而这样一种投票关系需要我们通过  $i2i$  的关系去发现。

### 4.3.4 构建过程

扩展边的权重(置信度)是我们需要着重考虑的。一个节点(商品)可能会有多条  $i2i$  扩展边, 那么分配权重的依据就是相似节点与置信边节点的相似度, 如果两者相似度高则对应扩展边权重也高。第  $i$  个相似节点的扩展边权重计算公式为:  $W_i = R_i / \text{SUM}(R)$  以一个例子来说明问题: 存在置信边  $A \rightarrow B$  (红色边), 若  $B$  与  $M$ 、 $N$  是相似商品, 相似度分别为  $r_1$ ,  $r_2$ , 则存在  $i2i$  扩展边  $A \rightarrow M$  和  $A \rightarrow N$  (绿色边), 那么  $A \rightarrow M$  边的权重  $w_1 = r_1 / (r_1 + r_2)$ ,  $A \rightarrow N$  边的权重  $w_2 = r_2 / (r_1 + r_2)$ ,

### 4.3.5 虚拟边

1. 虚拟节点: 假设存在的具备一定价值的节点。
2. 虚拟边: 由虚拟节点指向真实节点的有向边, 虚拟节点唯一指向一个真实节点。

在网络中如果我们希望把节点(商品)本身的特征分数引入, 则可以借助虚拟边。一个真实节点(商品)本身特征分数越高则射向它的虚拟边越多。

#### 4.3.5.1 构建过程

根据我们待引入的特征分来分布虚拟边, 例如, 当我们要引入人气分,  $A$  节点人气分为 1,  $B$  节点人气分为 3, 则有 1 条虚拟边射向  $A$ , 3 条虚拟边射向  $B$ , 如下图所示:

图 4.15

#### 4.3.5.2 构建过程网络迭代

使用 PAI 平台的 pagerank 算法进行 100 次迭代至网络基本收敛，产出节点（商品）的 PR 分数。

## 4.4 基于用户实时点击序列的表征学习




图 4.16

### 4.4.1 项目背景

在电商搜索中，个性化一直是业务致力追求的目标，以期通过为每位消费者呈现更精准、多元、个性化的商品来满足不同客户的浏览和购买需求，努力追求提升用户体验、促进消费升级、改善商业生态的多方共赢局面。基于这样的目标我们已经挖掘出许多诸如 User2Item、User2Shop、User2Brand 等一系列的个性化数据标签，本质上都是在学习如何从 Item、Shop、Brand 等不同维度对



用户进行准确的刻画或表征，以便在搜索或者推荐场景下能够为用户提供个性化的结果呈现。随着深度学习的不断发展，模型自身的表征能力得到了巨大提升；同时随着系统的不断改进，Wide & Deep、Online Learning 等机器学习方案不断落地；在现有的计算能力和模型基础上如何挖掘出新的信息进一步的完善系统，既是机遇也是挑战。

我们从现有的系统框架出发，主搜系统召回、海选、精排的三级火箭模式，从一个角度看是通过分层的过滤机制将最核心的计算力尽可能预留给最具价值的商品集合，避免计算资源的浪费，毕竟绝大多数的商品在一次请求中都不会得到有效曝光；而从另一个角度看，三级火箭的模式也是性能和效果的权衡折中，总存在一些优秀的商品候选因上游计算力不足而错失最终展现的机会。现有系统经过长期的优化和技术升级，在精排阶段形成了深厚的技术沉淀并取得了瞩目的业务效果，而本文的工作则将更多精力放到了召回和海选阶段，尝试在系统的不同阶段、从另一个角度提升系统对用户的表征能力，在有限计算力的前提下为用户提供更完备的个性化服务。

## 4.4.2 技术路线

基于业界已有的研究成果和主搜自身的系统框架，我们延续了将用户和商品 Embedding 到潜层空间的表征学习思路，逐步建立了以用户和商品向量为基础，通过大规模图结构搜索技术实现的个性化召回、海选方案。该方案面临的挑战主要有如下两点：

### 1. 如何通过有限的向量操作准确刻画用户与不同商品间的关联关系？

受限的向量操作是为了满足对海量候选商品计算的性能需求，因此向量自身需要能够通过 Cosine、Dot Product 等简单计算准确描绘出用户的行为偏好、即时兴趣与不同商品自然属性、历史销量等信息间的关联关系和关联程度。此外更值得考虑的是系统整体信息的有效增益，避免不同模型之间的相互覆盖，许多尝试都验证了对线上已覆盖信息做重复的刻画最终都难以实现业务指标的有效提升。

## 2. 如何在召回、海选阶段对大规模的商品候选集合进行高效地排序和筛选？

受限于候选规模，召回和海选阶段使用大规模深层网络带来的密集计算会给引擎造成较大的系统延迟，所以传统方案往往采用少量静态指标的简单排序，导致在召回、海选阶段缺失了对用户即时行为和个性化信息的充分考量，暴露出不同用户的召回结果类似等问题。因为即便只是进行简单的向量内积，系统也不能对每个用户遍历全部的商品候选，如何通过近似计算和启发式算法实现大规模地个性化召回在许多业务场景下一直都是亟待突破的系统瓶颈。

本文更多的围绕第一个问题展开讨论，主要解决如何在受限的向量操作下通过表征学习实现对用户和商品关联关系的准确刻画，进而在个性化召回与海选阶段提供准确的度量标准，而基于大规模图结构的启发式搜索算法，参见 [《基于辐射伸展图 (Navigating Spreading-out Graph) 近似最近邻检索算法》](<https://www.atatech.org/articles/95738>)。针对用户和商品的向量表示，我们参照信息互补的原则调研发现现有的模型更多的还是基于用户在主搜场景下的历史行为进行学习，如果可以更多的引入用户在全网其他场景的行为信息、同时强调对用户即时兴趣的刻画，应该会是对现有系统的一个有效补充。基于此，我们以用户全网的实时点击序列为基础利用深度学习对用户和商品的向量表示进行建模，寄期望于从用户行为的丰富性和即时性上对系统进行有效补充。

### 4.4.3 用户行为

上图是从训练样本中随机抽取的一条数据记录，可以看出用户在连续的点击商品间有着清晰的行为模式，在即时兴趣生效的范围内，用户所表现的价格、款式属性存在着相对稳定的个人倾向，而商品序列的颜色变化又体现了用户兴趣的短暂停留与演化迁移。用户的实时行为往往透漏着丰富的个人特征，却又难以用固定的指标进行准确的量化。

图 4.17

#### 4.4.4 模型结构

模型上我们针对点击序列部分对比了 Full Connection、LSTM、CNN、CNN with Attention 等多种网络结构, 其中全连接方案的效果最弱, LSTM 稍弱于 CNN 方案, 但起训练速度却远远落后于 CNN, Attention 部分我们基于 Item 向量进行 Self-Attention 处理, 再接入 CNN 网络, 不过实验发现提升并不明显, 最终线上采用了效果和性能比较理想的 CNN 方案, 具体实现主要参考了 Facebook 的




图 4.18

《Convolutional Sequence to Sequence Learning》。CNN 方案除了利用卷积核可以在不同商品向量间实现跨区块的交叉学习以外，通过 Linear Gated Unit 的开关控制，还能有效降低用户点击序列中的噪声信息，同时彼此独立的卷积操作天然支持并行化的数据处理。当然我们也对比了不同的激活函数、优化方法及正则化手段，目前深度学习模型依然需要针对特定的场景做大量的参数调优工作，这里不做赘述，不过整体上网络结构和与特征工程对模型最终的效果起着决定

性的作用；

受限于召回、海选阶段的计算性能，模型结构上用户和商品的向量只进行了一层内积操作，同时为了控制特征分的值域范围，用户和商品网络结构最后一层的激活函数从 ReLU 替换为 Tanh。在特征层面，除了用户的点击序列外，还包括了浏览、加购、收藏等历史统计信息，点击序列在经过卷积层和 Linear Gated Unit 处理后和统计信息拼接到一起再接入后续的全连接层。

## 4.4.5 目标函数

主搜系统的优化目标除了大盘的 GMV 外，还希望提高用户品质，尤其对部分高价优质的商品希望可以得到更多的曝光展现，因此在目标函数的选择上，我们以交叉熵损失为基础，为不同商品引入了价格因子：

$$Loss = -\frac{1}{N} \sum_{i=1}^N price_i^\alpha [y_i \log p_i + (1 - y_i) \log (1 - p_i)]$$

同时在模型评估指标上，我们希望高价优质的商品排序能够更靠前，所以在标准 AUC 的基础上也融入了商品的价格因素，其中  $m_i$  指排名低于正样本 ‘i’ 的负样本个数， $K$  是负样本的总个数，对于分数相等的情况也和标准 AUC 类似需要对价格权重做折半处理，本质就是希望打破原来所有正样本权重相同的情况。

$$Metric_{price} = \frac{\sum_{i \in positive} m_i * price_i}{K * \sum_{j \in positive} price_j}$$

整体而言，对于主搜现有业务的优化，如何构建目标函数以及使用怎样的度量指标进行离线评估并不是件容易的事情。一来我们要解决的不再是基于点击率、转化率、笔单价等任何单一指标的排序问题，甚至很难落脚到由哪个单一指标主导、或几个指标分批次优化，而是需要同时结合成交额、货单价、人群分层等多个指标联合求解，这就导致模型很难按照某个单一的目标函数进行优化求解；二来现有的线上系统包含了一系列独立的模型，任何一个新增的特征或一批候选宝贝的替换，最终效果都是线上全体模型的复合输出，因此诸如 AUC 等单一离线评估指标很可能出现线上线下的效果不一致，即便我们对原始

排序做了针对性的调权，依然碰到离线评估良好的模型上线不凑效的情况。因此，在问题切入、数据选择、目标构建阶段我们要做的就是方向上保证模型求解的目标与我们的愿景尽可能一致，而后便是冗长的尝试与调参工作。

#### 4.4.6 线上流程

图 4.19

上线时依托 Porsche 的实时数据流，模型对用户在全网范围内的每次点击进

行实时更新，通过 IGraph 实时反映到用户下一次的搜索结果中，而对于商品向量则是每天进行预测，按天 Build 到索引中。召回和海选时，基于实时的用户向量在商品构成的相似图谱中进行启发式搜索，召回用户感兴趣的商品集合。值得一提的是，在召回阶段，我们在商品和用户向量内积的基础上又补充了价格指数  $price_i^\beta$ ，超参  $\beta$  负责在转化率与货单价之间斡旋，间接寻求一个较为理想的成交金额和货单价。

不过用户的点击序列也会包含一些噪声或不适当的行为模式，比如观察用户的实时点击会发现，部分场景下用户会对同一个商品进行连续性的点击，如用户从搜索页点击进入详情页，再点击进行加购、收藏等行为，这会在有限长度的点击序列中连续多次出现相同的商品。少量的重复会强化用户某一部分的即时兴趣，但过多的相同商品也会挤压点击序列的有效存储空间，使得用户特征变得单一，不利于对用户后续行为的预测。针对该问题，一方面可以适当的延长点击序列，另一方面也可以对点击商品做适当的去重处理。

## 4.5 商品簇模型

整个淘宝平台有数十亿的商品，商品与商品之间的推荐关系和用户与用户之间的关系都是非常稀疏的，而且这些关系不管是在离线模型训练还是在线预测偏好，考虑到性能和存储空间的问题，都是要做阶段的，所以对于商品的聚类，压缩到一个比较小的空间上，还是比较有必要的。对于如此大规模商品进行聚类有很高的性能要求，本节将介绍一下如何通过启发式的方法寻找商品簇中心，从而快速的对商品进行聚类，有点类似于 [7]。我们用这个启发式的算法最后聚出来的商品簇在一定程度上满足两个特性，商品簇内的商品要足够相似，并且商品簇与商品簇之间的商品要有足够的区分度。

### 4.5.1 商品与商品之间的相似性计算

所谓的商品聚类，就是将相似的商品聚合在同一个商品簇中，而且簇与簇之间还要有保证一定的距离。所以商品与商品之间的相似关系的计算对于商品

聚类是非常重要的。淘宝上商品之间的相似度分数计算一般是根据用户点击购买收藏等行为计算出来的，两个商品之间相似分数越高代表着两个商品越相似，具体如何计算会在下面的章节进行介绍，本节主要介绍如何用给定商品与商品之间的相似数据给商品聚类。

## 4.5.2 相似商品数据预处理

在同一个商品簇内的所有商品之间需要都需要保证有一定的相似性，所以我们首先需要构造出一个商品与商品之间的 **Graph** 结构数据，如果商品与商品之间有边相连接，那么认为这两个商品是相似的，否则就是不相似的。而且这些商品与商品之间的边需要满足下面两个特性。

1. 商品与商品之间的边是无向边的：如果商品 **a** 能够推荐出 **b**，那么 **b** 也能够推荐出 **a**。
2. 商品与商品之间的边是可以传递的：如果 **a** 与 **b** 是相似的，并且 **b** 和 **C** 也是相似的，那么 **a** 和 **C** 也是相似的。

根据上面两个特性，我们对商品的相似数据进行一些预处理。处理后保留下来的商品与商品的关系，能够在一定程度上满足上面两个特性就可以了。

1. 对于每一个商品 **a**，保留 **top K** 个最相似的商品。
2. 对于每一个商品 **a**，假设商品 **b** 在 **a** 的 **top k** 个相似商品之中，如果 **a** 也在 **b** 的 **top k** 个相似商品中，那么保留 **a** 和 **b** 之间的边，否则将 **a** 和 **b** 之间的边去掉。
3. 在第二步之后，对于每条边 **ab**，统计商品 **a** 和商品 **b** 的共同邻居的个数，如果商品 **a** 和商品 **b** 的共同邻居个数大于等于一定的个数 **M**，那么保留这条边，否则去掉这条边。



### 4.5.3 商品聚类过程

在对相似商品数据进行与处理之后，得到一个比较高置信度的商品与商品之间的图关系数据，那么就可以根据这个图来对商品进行聚类了。我们聚类的基本思想是，首先从这个图中启发式的找出最靠谱的簇中心，然后其他商品根据与哪个簇中心相连接，并且权重最高，决定属于哪个簇。

商品簇的中心需要满足这两个条件，1. 商品簇中心必须与商品簇内所有其他商品都相似。2. 簇中心与簇中心必须是不相似的。根据这两个条件，我们制定了两个快速决定簇中心的两个策略：1. 簇中心的度越大越好；2. 簇中心之间不能有边相连接。具体寻找簇中心有下面几个步骤

1. 根据前面构造的图数据，统计每一个商品的度，也就是这个商品对应边度条数。
2. 根据度从高到低的顺序遍历所有的商品
  - (a) 如果该商品的所有相似商品都不是簇中心，那么将该商品设为簇中心，否则就不将这个商品设为簇中心。

在根据启发式方法找出所有的簇中心之后，就要对所有的其他非簇中心的商品进行了，决定每一个商品归属于哪个商品簇。

1. 根据商品与所有簇中心商品是否有连接的边，来决定归属哪个簇中心。
2. 由于数据做了很多截断，有些商品不一定和簇中心有直接。所以在上一步的基础上，用 knn 算法，看该商品的相似商品中属于哪个簇的相似商品最多。
3. 如果还归属不到任何一个簇的话，拿就不对这个商品进行聚类了，属于比较冷门的商品，没有与其他的商品比较相似，不能归属于任何一个簇。

整个聚类过程基本上都是对商品与商品之间的关系进行统计，排序，截断等操作，特别适合于利用 `mapreduce` 进行大规模数据处理，能够快速的对大规模商品进行聚类。

根据前面的数据处理后，平均每一个商品簇能够有差不多 100 个商品左右，而且每个簇的商品个数比较平均，不会有特别大。将商品簇数据利用于搜索，原本推荐出商品，现在可以推荐出商品簇，极大的扩大了推荐的覆盖率而且也没有耗费很大的存储空间和计算性能。也可以根据商品簇来计算商品簇与商品之间的关系，扩大推荐的范围，大大的提高推荐召回率。

## 4.6 实时计算

相比传统的网页搜索引擎，淘宝搜索由于以下几个原因对实时性的要求非常高：

首先，众所周知，淘宝具有很强的动态性，宝贝的循环搁置，新卖家加入，卖家新商品的推出，价格的调整，标题的更新，旧商品的下架，换季商品的促销，宝贝图片的更新，销量的变化，卖家等级的提升，等等，都需要搜索引擎在第一时间捕捉到这些变化，并在最终的排序环节，把这些变化及时地融入匹配和排序，带来结果的动态调整。

其次，从 2013 年起，淘宝搜索就进入千人千面的个性化时代，搜索框背后的查询逻辑，已经从基于原始 Query 演变为【Query+ 用户上下文 + 地域 + 时间】，搜索不仅仅是一个简单根据输入而返回内容的不聪明的“机器”，而是一个能够自动理解、甚至提前猜测用户意图（比如用户浏览了一些女士牛仔裤商品，然后进入搜索输入查询词“衬衫”，系统分析用户当前的意图是找女性相关的商品，所以会展现更多的女士衬衫，而不是男生衬衫），并能将这种意图准确地体现在返回结果中的聪明系统，这个系统在面对不同的用户输入相同的查询词时，能够根据用户的差异，展现用户最希望看到的结果。变化是时刻发生的，商品在变化，用户个体在变化，群体、环境在变化。在搜索的个性化体系中合理地捕捉变化，正是实时个性化要去解决的课题。

最后，电商平台也完成了从 PC 时代到移动时代的转变，随着移动时代的到来，人机交互的便捷、碎片化使用的普遍性、业务切换的串行化，要求我们的系统能够对变化莫测的用户行为以及瞬息万变的外部环境进行完整的建模。

## 4.7 在线学习系统架构

早先的搜索学习能力，主要是基于批处理的离线机器学习。每天离线训练模型，然后将训练得到的模型推送到线上系统做打分预测，线上预测阶段模型不会变化。这种离线模型 + 在线预测也是工业界常用的模式，但如上节说过，淘宝搜索因为其自身的特征，这种模式有很大弊端。在离线批量学习中，一般会假设样本独立服从一个未知的分布，但如果分布变化，模型效果会明显降低。而在淘宝搜索业务中，很多情况下，一个模型生效后，样本的分布会发生大幅变化，因此离线学到的模型已经不能很好地匹配线上数据了。所以需要在线学习，不断地实时调整模型，拟合最新的线上数据，特别是在大促这种数据分布变化很大的场景下。

为了满足在线学习的需求，搜索技术团队自主开发了基于 parameter server 的在线学习框架，如下图所示，目前已经实现了 FTRL，矩阵分解，深度学习等多种在线学习模型，模型更新后会实时推送到线上打分引擎。

## 4.8 在线 FTRL

本小节以 FTRL 为例，说明在线学习的简单应用。其他在线学习的模型，可以再个性化等章节找到。

逻辑斯蒂回归 (LogisticRegression) 应该是工业界最经常使用的一个模型，能处理超大规模的样本量和特征量。为了防止学习算法过拟合 (overfitting)，通常会在算法中加入正则项 (regularization)。在逻辑斯蒂回归模型中，优化目标一般如下式：

上式中的第一项是模型 loss，第二项为正则项。最常见的正则是使用 L2-norm，这是由于 L2-norm 是 smooth and strong convex，导数比较好计算。另外一种常见的是 L1-norm，相对 L2-norm，L1-norm 产生的解容易落在坐标轴上，落在坐标轴也就意味着对应的  $w$  为 0，从而得到稀疏解。L1-norm 的好处在于可以使模型得到更多的稀疏解。稀疏解不仅能提升模型的泛化能力，而且在线部署阶段也能节省模型存储空间。

但传统的在线优化算法，比如随机梯度下降（SGD），并不能有效得到稀疏解。如何 onlinelearning 得到稀疏解也是一个研究的课题。

### 4.8.1 Truncated Gradient

为了得到稀疏解，最简单粗暴的方式就是设定一个阈值，当权重小于这个阈值时将其设置为 0，如下面的伪代码。这种方法实现起来很简单，也容易理解。但实际应用中，简单进行截断会造成这部分特征的丢失。上述的截断法太过简单粗暴，因此 TG 在此基础上进行了改进，如下：其中  $\alpha(\beta) \leq \beta$  且  $\alpha(\beta) \geq 0$ 。TG 同样是以  $\beta$  为窗口，每  $\beta$  步进行一次截断。当  $\beta/\alpha$  不为整数时  $\alpha(\beta)=0$ ，当  $\beta/\alpha$  为整数时  $\alpha(\beta)=\beta/\alpha$ 。从公式 (3-1-3) 可以看出， $\alpha$  和  $\beta$  决定了  $\alpha$  的稀疏程度，这两个值越大，则稀疏性越强。尤其令  $\alpha=\beta$  时，只需要通过调节一个参数就能控制稀疏性。

### 4.8.2 FOBOS

先前向后切分（FOBOS, Forward-Backward Splitting）是由 John Duchi 和 Yoram Singer 提出，在 FOBOS 中，将权重的更新分为两个步骤：前一个步骤就是一个标准的梯度下降过程，后一步是对前一步下降的结果进行微调。观察第二个步骤发现对  $W$  的微调也分两部分，第一部分保证微调不要偏离第一步找到的梯度下降方向太远，第二部分则用于处理正则化，产生稀疏解。

### 4.8.3 RDA

前面提到的 TG、FOBOS 都还是建立在 SGD 的基础上，属于梯度下降类型的方法，这类型方法的优点就是精度比较高，并且 TG 和 FOBOS 也都能产生稀疏解。正则对偶平均（RDA, Regularized Dual Average）是从另一个方面来求解 Online Optimization。RDA 算法中优化的是前  $T$  轮所有 loss 的平均，可以达到平均效果较好。

#### 4.8.4 FTRL: Follow The Regularized Leader

FTRL 是 google McMahan 提出的算法，它综合考虑了 FOBOS 和 RDA 的优点，兼具 FOBOS 的精确性和 RDA 优良的稀疏性，其特征权重更新公式为：

该优化公式分为 3 部分，第一部分表示累计梯度和，就是让损失函数下降的方向；第二部分是表示新的迭代结果不要偏离已经产生的迭代结果太远；第三部分是正则项，有用于产生低遗憾的强凸二范数和产生稀疏解的一范数。另外值得一提的是 FTRL 采用针对每个特征不同的自适应的学习速率，直观的解释是如果特征出现的次数多，对这个这个特征的权重比较置信，学习速率就比较小，对于很少出现的特征学习速率就比较大。

## 参考文献

- [1] C. Burges, T. Shaked, etc., Learning to rank using gradient descent. In Proceedings of the 22nd international conference on machine learning, ACM
- [2] 流量个性化 v.s 商业化 - 双 11 珠峰项目中控算法, <http://www.atatech.org/articles/67132>
- [3] 确定性保证下流量分配在线全局优化策略, <http://www.atatech.org/articles/55983>
- [4] 搜索流量确定性项目总结, <http://www.atatech.org/articles/59651>
- [5] 网络效应分介绍, <https://www.atatech.org/articles/52962>
- [6] Unbiased Learning-to-Rank with Biased Feedback, <http://weibo.com/ttarticle/p/show?id=2309404077533346815648>
- [7] A. Rodriguez and A. Laio, “Clustering by fast search and find of density peaks”, Science, Vol.344, No.6191, pp.1492–1496, 2014.

## 第五章 个性化搜索背后的核心技术

### 学习目标与要求

随着网络的流行和互联网信息的爆炸性增长，如何从海量的信息中准确找到自己需要的信息成为了互联网发展面临的一大难题。传统的搜索引擎由于其通用性，对于相同的查询，所有用户得到的都是同样的结果，显然不能满足不同背景、不同目的和不同时期用户的个性化需求。在这个讲究个性和以人为本的时代，人性化的搜索引擎已经成为了时代的需求。

在淘宝的环境下，用户主要的目的就是购物。不同的用户个体在购物上更是天然就存在着很大的差异，这种差异可以反映在很多方面，有价格偏好、类目偏好、地域偏好、品质偏好等等。而我们要做的就是挖掘，理解用户的差异和不同需求，做出个性化的搜索体验，最终能够帮组用户在淘宝上购物。

个性化搜索最根本的目标是要根据用户的意图和喜好展示搜索结果页，抛弃目前这种千篇一律的展现方式。个性化体现了搜索系统的智能性，从提高用户体验方面加深了用户对淘宝的粘性。

进入正题之前首先谈一谈个性化方向后续需要关注的几个点，搜索个性化时至今日，已经成为互联网网站的技术标配，虽然业界取得了一些成绩，但挑战仍然存在；

首先来看看为什么要作个性化，搜索中引入个性化的目的是什么每天有近30000个查询“连衣裙”的消费者，query + “user context”的查询逻辑能够实



现不同的消费群体看到不同商品投放结果，实现平台上人-货匹配在搜索流量上的个性化细分，比如说，“肥胖”的女性查询结果里面更多的展现宽松风格的商品，而消费能力高的消费者更多的展现高品位的大牌商品，从而达到流量投放效率的最大化；总而言之，目的是两个：a). 提升流量匹配效率：具体表现在购物路径上的效果指标；b). 改善宽泛 query 下得流量集中性，提升宽泛 query 下不同人群看到的展示商品不同，而带来成交商品和点击商品的丰富性；总而言之：对于广大消费者，由于个性化能够细分搜索意图，拟合个体偏好，有助于更快捷找到需求；

弄清楚了搜索个性化的目的，下面想来澄清几个问题：

### 1. 个性化并不是定制化

Customization：实现的境界是 You are what you say you are? 或者说，平台按照系统理解的用户 profile，并按照某种特定个性化规则去投放，即是，实现 you are what the system thinks you are. 对于针对用户的查询结果的信息展示是局限在 explicit 的“feature”层面，比方说，按照购买力匹配规则，按照品牌偏好规则等等；定制化的好处的确能够在某种程度上带来强的个性化体验，但是带来的伤害也是不言而喻；而 Personalization 是：根据用户行为所挖掘的偏好信息来进行展示商品的投放，即是，实现 you are what you click on and what you buy；对于针对用户的查询结果的商品展示是基于“内容和数据”层面；不去刻意的假设用户的行为是由于某个特定维度（人口统计类特征，偏好类特征，人群特征）造成的，消费者点击或成交行为的发生，是所有个性化信息的综合表现；为什么我在这里先提出这个问题，因为经常听到的很多关于，目前线上个性化效果不尽人意的反馈，在这里，也不去刻意回避我们自己的问题，个性化数据，模型的覆盖率，准确性和时效性等都需要进一步的优化和改进；然而，对于那些为了增强所谓的个性化体验而实行的规则式匹配逻辑，都是极其不科学的做法，对于消费者而言，他们需要的是找到一个符合他 / 她需求的商品，而个性化体验强弱与否并非是最最终的目的，我相信的是，消费者不会因为我们预测到他的性别，购买力，偏好的品牌就做出点击或购买的决策，个性化是我们系统实现高效的【人 - 货】匹配效率的手段，并非是消费者的购物诉求；在这里也请从事个性化方面的运营，产品，甚至算法同学能时刻理解这点；



## 2. 不要陷入活跃/资深用户的悖论

正常的用户无论其活跃与否，都不会愿意浪费时间去填写所谓的友好的交互式表单来帮助系统去理解他们，从而得到更好的个性化体验；他们关注的是展示商品整体是否满足他们的需求，而不会去刻意的由于商品的某个维度匹配了他/她得某个偏好而做出最终的选择；这里列举一个曾经的产品设计，在搜索结果页，给出用户可以定制的个性化偏好交互界面，希望消费者能告诉我们他们的个性化 profile，出发点是好的，结局大家懂的；

## 3. 个性化 explore 的重要性

随着个性化元素在搜索全链路的渗透，从 query 的个性化标注，海选的个性化召回，精排中的个性化排序因子，以及个性化 rerank，个性化展示，使得最终呈现给用户的内容取决于系统底层根据用户历史行为所挖掘的个性化特征，人口统计学维度，兴趣点偏好维度，session 级别实时特征，过度的” user specific historical behaviour driven “的个性化投放，会使得用户逐渐丧失对展示结果的新鲜感，并且视野变得越来越狭窄，进而使得底层的用户数据模型丧失自我修复和自我扩展能力；因此一个完整的个性化体系，必须考虑 explore 机制的设计环节；

## 4. 个性化评估的方法论

要想推动个性化效果的正向迭代，首先需要建立起合理的效果评估体系；然而这仍然是一个很大的问题，学术界流行的准确率，召回率，F1 值，AUC，RMSE，等都有很大的局限性，这一层面的评估，只能保证数据模型的正确性；而在实际工作中，这些指标上的不一定能保证线上效果的收益；因此我们需要第二层次的评估手段，来看个性化算法效果。实现个性化的投放效果，是系统层面的主动而为，而且并没有去引导消费者端在一次搜索看到展示结果后，做出选择。因此在消费者不知情情况下，消费者的行为反馈可以用来作为个性化效果评估的一个手段。对于已经上线的个性化特征，需要部署相应的统计分析模块，在 ABtest 机制下，监控各个特征的覆盖率，以及覆盖流量下的点击率，转化率等；虽然无法直接统计到这些特征对于点击和转化带来的精确影响，但是通过追踪高权重 user 的体验 - 点击率，2 跳率，转化率等，能够了解这些特征的影响趋势，及早发现问题；这里特别强调下，采用高权重 user 的行为数据来分析的原因是，高权重用户意味着是活跃用户，意味着行为丰富，而这类用户的个性化特征的

表现会更有代表性；最后，我也来谈谈对于针对个性化效果的社会化评测的意见和想法，便于理解，就以用户购买力为例来讲讲，为了更好来把握该维度数据的有效性，经常利用的手段是社会化评测来给定一些查询下，看看展示结果里面展示商品的价格是否符合评测者的价格偏好，从表象上看，似乎没有问题，然而，这里面确有一个本质上的问题，我们限定了这些参与评测人得判断角度，只关注商品价格，并给出满意与否的结论，而在实际购物场景下，用户对于商品满意与否接受与否，并不是只限定在价格本身，因此这样的评测还是有一定的局限性；我个人的观点，还是去真实的模拟线上的判断环境，不去刻意要求消费者去关注某个固定的维度，只是给出 site by site 的结果，让用户判断哪边展示的商品更符合他的口味，当然，这不同 site 的展示结果的差异，背后只是某个维度的个性化带来的影响，这样去评测，才更加客观；总结一句话，就是众包评测的关键是，希望参与者能做出客观的反馈，不应该做任何主观性的引导；

## 5. 个性化体系对于系统和框架的影响

在搜索场景下实现个性化的效果，就需要去建模分析【query-user-商品】三元组构成下得海量数据分析，数据是极端稀疏的，算法时间和空间的复杂性，都对于系统能很好的支持分布和并行的数据分析和建模能力提出了很高的要求；另外，用户偏好的时效性，也需要我们能够实现增量，实时计算能力，个性化的实施，使得传统引擎依赖的性能优化利器，cache 机制无法施展手脚，因此对于引擎的创新性改造也提出了更高的要求；另外，个性化数据的挖掘都是存在不确定性的，如何来设计一套能够保证误差不会累积的算法体系，也就是说需要建立一套数据自我修复的实时反馈体系，来保证由消费者端实时获取的客观反馈数据参与到个性化投放环节来保证模型的自我修复能力快于数据误差的传播速度；这样才能保证数据产生的价值形成良性的循环，构成大数据生态体系；

个性化是一种解决“长尾需求”的方式，“长尾理论”说的是用户需求集中度越来越低，用户和用户之间不一样，我们如何来区分这种不一样？个性化搜索就是融合推荐元素，以实现：用户个体需求主导的“pull”式搜索加平台以数据驱动的方式对用户进行“push”式相关信息推送；

综述性的东西，@ 三桐，@ 公达

## 5.1 用户肖像建模

用户肖像是对用户全方面的描述，一般分为人群和偏好两种。我们通常说的用户人群，是指用户长期稳定属于的某一个群体，如男性、女性。而用户偏好，在淘宝中是指用户对某种商品的偏好，是会经常或实时变化的，如偏好欧美风格的女装。有的时候，用户人群和偏好又容易混在一起，例如用户是男性，是指他的真实性别是男性，而他可能会偶尔给老婆或者母亲购买女性服装，这时的实时偏好则会是女性商品。因此，性别这个维度，既包含了人群，也包含了偏好。当我们讨论长期性别时，指的是人群；当我们讨论实时性别时，指的是偏好。用户的年龄也是类似的。

物以类聚，人以群分。不同的人群，在总体上有着不同的行为特点和购物需求。我们对用户的了解，是从他/她所属的人群开始的。人群可以按不同的维度划分，如性别、年龄、购买力、地域等。例如，在服饰行业中，男性用户更喜欢买男装，女性用户更喜欢买女装。这样，在用户搜索“T 恤”时，我们可以根据他的性别展示更符合他偏好的结果。不同年龄段的用户的购物需求也会有明显的差异，例如穿衣风格或者手机款式。为了识别用户所属的人群，需要使用尽量多的数据。最基础的是用户注册的信息，不过这种信息有时并不准确。比如，用户注册时填的不准确，或者用户把账号长期给家人使用。所以还需要使用用户在网站上的行为数据来校正这些数据。这时会使用机器学习的方法，把用户肖像建模看成一个分类问题，使用各种来源的数据来预测用户所属的人群。

在个性化搜索中，首先需要满足用户的基本体验，使用户看到的商品基本符合用户的需求，即至少不要看到不喜欢的商品。同时，用户也希望看到多样的不同款式的商品，在其中进行筛选。这种基本的商品集合，一般是通过商品的属性标签来表示的，如性别、款式、价格档位、年龄段等。因此，为了实现好的个性化搜索体验，必须准确的预估用户的偏好。预估用户偏好的方法分为三种：长期偏好，实时偏好，和人群偏好。淘宝上的活跃用户，在某些商品属性上有长期稳定的偏好，这时可以计算他们的长期偏好。当用户在购物的过程中，有些偏好还会随时发生变化，或者这次搜索和上次搜索的状态不同，比如刚刚发了年终奖，这时需要计算用户的实时偏好。对淘宝上的不活跃用户，长期没

有行为或行为少的用户，用他的行为很难准确预估他的偏好，这时只能用他所属的人群来预估他的偏好。实际上，我们会把这 3 种偏好综合成一个，即某个维度上的综合偏好，这个综合偏好体现了最终个性化体验的效果。

### 5.1.1 性别

性别作为用户最重要的基本属性之一，必然是个性化考虑因素。对电子商务网站来讲，性别也是搜索和推荐系统决策因素之一。淘宝主要消费群体是女性，用户数据容易被女性行为主导，人气排序下表现尤为明显。性别个性化则是根据用户的性别影响排序，在用户 query 没有明确表明性别的情况下提前与用户性别相同的商品，旨在减少翻页次数 or 换 query 次数从而提高 ctr。另外，如果用户能看到更多与其购买意图相关的商品，可能会提高成交转化率。

**5.1.1.0.1 商品的性别** 为了利用性别影响排序，首先需要解决如何标记商品性别。商品的性别可通过类目或者属性来表现，而类目的性别表现又分为窄义性别和广义性别。窄义性别表现类目有服饰、鞋和包等，此类型只要提前相应类目或者类目下具有某些特征的商品即可；广义性别表现类目包括窄义类目和诸如手机、电脑、游戏币等隐含类别，该类型下商品的性别与类目无关，而是由商品本身的特征决定的，如颜色、风格等（与性别无明显关系），这类性别标签需要挖掘才能发现。

**5.1.1.0.2 用户性别模型** 性别个性化另外一个重要的方面是如何预测用户性别。用户注册时的性别信息和支付宝实名认证都可以作为判断性别的依据，但考虑到用户可能填错以及实名认证用户少、甚至有账号被同时多个用户使用的情况，我们不能直接应用这些信息。个性化用到的性别必须有物理性别与淘宝性别之分，所以必须建立一套合理的性别预测方案。

**5.1.1.0.3 建模** 这里，我们直接使用 LogisticRegression 模型，预测用户为男性还是女性。训练目标是购买商品类目的性别和身份证性别一致的，即购买过类

目性别和身份证性别都为男性，才认为是男性。

图 5.1: 性别模型训练流程

#### 5.1.1.0.4 特征

- 分别 15 个女性、男性倾向类目的点击总数
- 分别 15 个女性、男性倾向类目的购买总数
- 强女性类目点击天数
- 强男性类目点击天数
- 总点击天数
- 强女性类目购买天数
- 强男性类目购买天数
- 总购买天数
- 女性倾向类目点击次数占比

表 5.1: 性别准确率

新版	样本数	预测男	预测女	召回率	准确率
真实男	1380636	1192135	73080	86.3%	93.4%
真实女	1353568	84840	1156177	85.4%	94.1%

- 男性倾向类目点击次数占比
- 点击占比熵
- 女性倾向类目购买次数占比
- 男性倾向类目购买次数占比
- 购买占比熵
- 强女性、男性类目类目点击天数差占有点击天数的比例
- 强女性、男性类目类目购买天数差占有购买天数的比例

**5.1.1.0.5 效果** 最终效果，总体召回率：86%，总体准确率：94%。

5.1.2 年龄

**5.1.2.0.1 用户年龄** 用户年龄的识别可以简单的使用身份证上的生日计算年龄。但由于家庭账号，或者代买的情况存在，直接使用身份证上的年龄也可能不完全准确，这时可以使用在某些能反映年龄的商品上的行为，来修正身份证的年龄。例如，经常买“中老年女装”的人，可能年龄较大。具体的做法和性别类似，这里就不再详述了。

从业务上来说，在电商场景下似乎更加关注用户所处的年龄段，而不一定需要细化到年龄值的信息，使用年龄段的数据可以确定用户的其他属性，比如学历，职业，人生阶段等特征，进而在的用户画像、推荐、个性化搜索、反作弊识别等方面提供有力的数据支持。我们一般把用户的年龄分为如下 7 个阶段：



- 0-18 岁：未成年
- 18-22 岁：大学生
- 22-25 岁：研究生/刚参加工作
- 25-30 岁：轻熟期
- 30-35 岁：成熟期
- 35-50 岁：中年
- 50 岁-：中老年

当然，这种划分方式并不是唯一的，也不是固定的，只是尽量反映当代用户心理和购物需求的不同。

**5.1.2.0.2 商品年龄段** 用户年龄识别准确，还需要商品年龄才能完成用户-商品的加权，这就涉及到商品年龄打标的问题，首先我们分析了商品在不同行业下的年龄区分度，发现年龄只在服饰、鞋包类目下区分度较高，其他行业如 3C 数码等区分度很小。淘宝服饰行业商品中有些是注明了年龄段的，但存在以下几个问题：

1. 年龄缺失
2. 年龄填写错误
3. 年龄填写多个年龄段

为了解决这个问题，我们需要来提升商品年龄打标的覆盖率，我们可用的商品特征有商品的图片、标题、属性等，这里我们以商品 pv 下标题分词、属性分词后的数量大于一定阈值的结果作为特征，最终选出大约 15w 特征，通过 softmax 模型来预测商品年龄，基于 label 的预测准确率约 60%，label 上下一档都设定为正确的准确率的大于 90%，而我们的生效逻辑是为用户年龄上下一档的商品加权，所以基于 softmax 的年龄模型效果已经可以达到我们的要求。如果还需要继续提升准确率还可以考虑：加入商品图像信息，用 active learning 的方式

标注错分样本，解决代买（中年人给孩子和长辈代买的最多）、用户年龄不准导致商品年龄 label 不准等问题。

### 5.1.3 购买力

前面提到了，每个用户在很多方面都是有自己独特的偏好，而购买力就是最明显的一个方面。高帅富和屌丝对商品的价格和品质有完全不同的需求。同样是搜索“T 恤”，高帅富需要的是面料材质好的品牌货；而屌丝需要的是 100 元 3 件还包邮的大路货。这种需要的差异就是我们需要做个性化购买力的原因。

**5.1.3.0.1 商品的价格档** 为了便于在业务中分析各种数据，可以将用户的购买力分成几个档次（如 17 档），档次越高表示用户的购买力越大。用户的购物行为中可以很方便的体现购买力。这时需要先确定商品的价格档。在不同类目下，不同的价格代表了不同的购买力，比如在 200 元的 T 恤是比较高端了，但 200 元的手机是非常低端的手机，所以我们首先需要计算每个类目下各个购买力档对应价格区间。将类目下所有宝贝的成交价格从低到高排序，然后按照 20%，20%，20%，15%，10%，10%，5% 的比例划分成 7 档。对于每个宝贝，查询上面的类目成交价格分布表，看一下宝贝价格是在哪一档价格区间中，将这个宝贝打上购买力档的标签。

**5.1.3.0.2 用户购买力模型** 我们使用了 GBDT 模型，训练用户的购买力。以未来搜索引导的成交在类目下的价格分档为目标，建立了一个多分类的模型。

#### 5.1.3.0.3 特征

- 服饰类成交额、笔单价（衣）
- 食品类成交额、笔单价（食）
- 日用品开销（家居百货）
- 是否有房 + 住房档次（住）



- 装修档次（住）
- 是否有车 + 车档次（行）
- 酒店门票类开销（行）
- 购买品牌、奢侈品
- 职业
- 教育程度
- 年龄段
- 手机型号
- 资产等级
- 好友关系

**5.1.3.0.4 跨类目购买力** 在淘宝中，用户在很多类目的行为很少，如果只用这个类目的行为，很难准确预测购买力。所以可以使用协同过滤的思想，参考相关的有行为的类目的购买力，来预测少行为类目的购买力。

通过大量的数据分析，可以计算任意两个类目之间购买力的相关性。越相关的类目，对预测类目的购买力贡献也越大。

经过类目扩展之后，购买力的覆盖率可以显著提高。

## 5.1.4 地域

南方人认为豆腐脑是甜的，北方人认为豆腐脑是咸的。不同地域的用户，购物需求也会有很大的不同，比如很多大件商品、或者生鲜商品、本地生活服务等都只能在同省或者同城才能提供，而且即使是普通商品，距离越远运费也越高。不同地域的换季时间也有一定的差异，这也会提前反映到为换季准备的衣服上。通过天气的预测，可以预测用户的消费行为，这是一件神奇的事情。

图 5.2: 跨类目购买力

从大的范围来划分，可以分为南方、北方、东部、西部、中部。从小的范围来看，可以看用户所在的省、市、区。更细的粒度，是用户的生活轨迹，如家庭位置、工作位置、休闲娱乐位置等。

目前能定位用户位置的信息源主要包括 IP、LBS、WIFI 以及物流收货地址等，这些数据源与用户结合便构成了一张地理信息网，如下图所示：

由于这些位置数据源表现形式及采集方式不一样，首先需要对这些位置数据源本身进行分析和挖掘，比如物流地址是用户的自然语言文本描述，随意不规范，需要对其进行结构化解析，去获取地址中的重要信息，包括省、市、区县以及路和门牌号等信息。另外，物流地址是用户自己描述，现实中是否真的存在也是很多业务应用中关注的焦点，因此还需要对物流地址进行有效性判断。目前地址结构化解析及有效性判断主要采用了文本挖掘中的相关技术去分析和挖掘，除此之外还有对地址进行类型划分及归一化等研究。而对于 WIFI 和 IP 主要是研究它们是否正常以及所属的类型（如：家庭 WIFI/IP、公司 WIFI/IP 及公共 WIFI/IP 等）。同时，WIFI、IP、LBS 以及文本地址虽然在空间位置描述方式不一样，但是他们之前是相互联系的，而且也有非常大的应用价值，比如地址中经纬度识别，这样就能从地图上更精准的定位到用户所在地。

图 5.3: 用户地域数据来源

### 5.1.5 职业

在用户肖像中，职业是不可或缺的一环。不同职业的用户，偏好的商品也是不同的。比如买衣服，学生、白领、公务员，他们的穿着风格也会不同。因此作为搜索引擎，也应该给他们更多合适的商品。

我们怎么知道用户的职业呢？通过用户的注册信息肯定不行，没人会认真填写职业，而且职业也不是一定不变的。所以，最可能的方式就是用户的收货地址了。比如经常在“公司”地址收货的妹子，那就很可能是“公司职员”办公白领了，而经常在学校收货的妹子，那就很可能是可爱的“学生”MM了。

我们把用户的职业分成下面几类：

- 学生：初高中、大中专及以上的学生群体
- 教职工：幼儿园、初高中、大中专及以上的教工群体，包含教师和其他职工
- 工人：“化工厂”、“化肥厂”，“焦化厂”等工厂工作者
- 媒体从业者：“报社”、“电台”、“杂志社”、“制片厂”等媒体机构从业者

- 金融从业者：“银行”、“证券”、“担保”等金融行业从业者
- 医务人员：“医院”、“诊所”、“防疫站”等医疗场所工作人员
- 公司职员：收货地址中包含“公司”、“写字楼”、“商务楼”、“科技大厦”等关键词的用户，但不包含上述“媒体从业者”、“工人”中提到的公司
- 个体经营/服务人员：“便利店”、“奶茶店”等个体店面经营者，或者“营业厅”、“洗浴中心”、“美发沙龙”、“酒店宾馆”等娱乐行业从业者
- 科研人员：“研究所”、“研究院”、“科研所”等场所研究人员
- 公务员：“人民法院”、“地税局”、“国税局”等事业单位工作人员

通过收货地址上的关键词匹配，我们可以识别一部分的用户职业。如果要识别更多用户，就需要用模型了。

图 5.4: 用户职业模型

首先，我们对于各个职业的用户均匀采样作为模型训练集；然后将用户在各个叶子类目下，浏览、收藏、加购、下单的行为数据，以及年龄和性别这两个

基本属性作为特征输入，结合 LR 生成模型；最后，从规则标注用户中，再次采样与训练集不重合的用户作为测试集，可以计算出预测模型的平均准确率达到了 80% 以上。

### 5.1.6 人群

有了上面提到的用户性别、年龄、购买力、地域、职业，就可以对用户人群有一个基本的定位，比如“都市网购达人轻熟女白领”。这样就可以把所有的用户划分成几十到几百个小的群体，每个人群会有一些特有的属性和购物偏好。更进一步的，还可以识别用户是否有房有车，是否结婚生子，以及是否是“数码达人”、“文艺青年”之类的，逐渐形成一个完整的用户画像。

图 5.5: 用户画像

## 5.2 匹配学习

## 5.2.1 用户体验模型

**5.2.1.0.1 意义** 上一节讲了用户肖像的建模，是预测的用户单一维度的人群或者偏好。在实际的搜索排序中，是需要多个维度综合判断的，不同维度的重要程度也不一样。用户体验模型的目的就是把不同维度的人群数据和偏好结合起来，确定各个商品属性的影响因子，对每一个用户形成最终的用户体验分数。

用户在搜索时，首先需要输入 query，不同的 query，需要的个性化体验是不同的。例如，搜“衣服”的时候，需要性别个性化，而搜“连衣裙”的时候则需要年龄、风格个性化。因此，首先需要识别 query 的类型，如行业、是否跨类目、是否包含品牌词等。然后，获取用户的人群和偏好信息，和商品的属性信息，对 Query+User+Item 的三元组数据建模，预估这时的用户满意度。

## 5.2.2 用户-商品 CTR 预估

**5.2.2.0.1 背景** “个性化”在淘宝搜索中起着至关重要的作用，即让不同的用户看到最符合自己需求的商品。为了实现这个目标，最直接的方式就是预估商品到不同人群的 ctr。当用户搜索时，使用这个分数排序，就可以把符合用户所属人群的商品优先展示。

### 5.2.2.0.2 建模

## 5.2.3 序列匹配模型

1，前面三个章节我们递进的描述了用户与单个商品之间的匹配方式和模型。然而，上述方法均假设用户的购物行为之间是独立的——并不存在依赖、相关或序列关系。

举例来说，一个用户  $U_1$  2 天内依次购买了以下商品：烤箱、面粉、奶油；另一个用户  $U_2$  半年内依次购买了孕妇衣、尿布和奶瓶。我们先考虑用户  $U_1$ ，我们可以从他购买了烤箱和面粉 2 种商品推断他很可能想要做蛋糕（而这从每个单一买的商品都是很难推断的），因此也许需要奶油；再考虑用户  $U_2$ ，我们可以从

她依次购买了孕妇衣和尿布推断她很可能怀孕过并且已经生了小宝宝（从某一件来推荐会比较勉强），因此马上会需要奶瓶等婴儿用品。

从上面例子我们可以看出，用户的购物行为之间往往是存在高阶依赖关系的，即仅用户购买了一个商品集合  $\{A, B, C\}$  后，才会购买商品  $D$ ；同时，用户的购物行为也会存在序列关系，即用户购买  $C$ ，仅会在他依次购买了商品  $A$  和  $B$  之后。在这 2 种关系下，我们前 3 节使用的模型会很难捕捉这类规律。因此我们需要一种模型，能整体的考虑用户的行为历史（而不是将其行为拆分成一个一个的单独分析），进而推断他接下来的需求。

下面我们会首先介绍几种经典的序列模型以及带有记忆功能的模型，然后会详细介绍在淘宝搜索中，我们怎样使用这类模型做到用户与商品之间的序列匹配。

2，在机器学习的任务环境中，我们有大量的场景都是需要做一个序列预测和带有记忆的推断的。例如在 query 自动补全的场景下，我们需要根据用户输入的文字或者词序列来预测用户下一个最可能会输入的词语；又例如有这样一个问题，需要让机器在阅读了一整篇文章后，回答若干关于这个文章的问题。这类问题和场景下都需要模型具有一定的记忆能力，能在获取新信息的同时，记住部分老的信息。

A) 最常用而有效的方式是使用一个递归神经网络模型 (RNN [4, 5])。正如其名字描述的，递归神经网络在隐层结构上存在一个循环，即当前隐层的输入是上一个隐层的输出以及当前的输入 2 项一起。由于每个隐层的信息都能递归的输入到下一个隐层中，因此会具有一定的记忆能力。如图 5.6，我们将 RNN 按时间序列“打开”，可以看到前一时刻的隐层  $S_{t-1}$  和当前输入  $X_t$  会共同影响当前的隐层  $S_t$ 。

然而 RNN 存在的最大问题是“梯度消失和爆炸”问题 [6]。这是因为在神经网络进行反向传播 (backpropagation) 的时候，传播的梯度会是  $w_{l,h}(t)$  (递归网络的权重) 的倍数；因此在递归层数较深的时候，梯度会消失掉 (当  $|w_{l,h} * y'_l| < 1$ ) 或者爆炸 (当  $|w_{l,h} * y'_l| > 1$ )。由于 RNN 存在“梯度消失和爆炸”问题，RNN 的“记忆”只能是很短期的，并不具备长期的记忆。

B) 为了解决“梯度消失和爆炸”问题，一种更为巧妙的递归网络结构 LSTM (Long

图 5.6: 递归神经网络 (RNN) 示意图

Short-Term Memory)cite5 被设计了出来。在 LSTM 中，RNN 中递归的隐层单元被一个存储单元 (LSTMUnit) 所替代，每个存储单元由一个输入门 (InputGate)，一个输出门 (OutputGate) 和一个长期的内部的通过遗忘门 (ForgetGate) 更新的内部状态 (Cell) 相关联, 如图5.7。

图 5.7: LSTM(Long Short-Term Memory) 示意图

内部状态 Cell 可以理解为模型存储的长期记忆：每进行一次递归迭代的时候，Cell 会通过遗忘门遗忘掉部分记忆，同时通过输入门决定当前输入有多少有效信息是需要被记住的，从而得到新的记忆。最终的输出通过当前新的记忆得到，由输出们决定新的记忆中哪些是当前需要的。在 LSTM 的反向传播过程中，不同于 RNN 中梯度是一个连乘的形式 (由于链式法则)，可以转化成一个



连加的形式，因此有效的避免了梯度的消失和爆炸，从而具备一定的长期记忆的能力。在基础的 LSTM 基础上，学者们提出了多种 LSTM 的变种，比如 [10]、GRU[11] 和 Clockwork RNN[12]，他们在计算性能上会有较大的差别，然而效果基本没太大差距 [13, 14]。一个基本的 LSTM 更新公式如下：

$$i_t = \sigma(W_{hi} * h_{t-1} + W_{xi} * x_t + b_i) \quad (5.1)$$

$$f_t = \sigma(W_{hf} * h_{t-1} + W_{xf} * x_t + b_f) \quad (5.2)$$

$$o_t = \sigma(W_{ho} * h_{t-1} + W_{xo} * x_t + b_o) \quad (5.3)$$

$$g_t = \tanh(W_{hg} * h_{t-1} + W_{xg} * x_t + b_g) \quad (5.4)$$

$$c_t = (f_t * c_{t-1}) + (i_t * g_t) \quad (5.5)$$

$$h_t = o_t * \tanh(c_t) \quad (5.6)$$

C) 递归神经网络外主要能有效的处理“序列”相关的问题，因此被大量的用在 NLP 的问题中。除了 RNN 外，也有一些其他的模型有类似功能，例如神经图灵机 (Neural Turing Machine, NTM[7]) 和记忆神经网络 (Memory Networks, MemNN[8, 9])，他们在不同场景下会比 RNN “记住” 更久远的信息，从而得到更好的效果。神经图灵机的主要思想是使用一个  $M * N$  的矩阵取存储一份长期记忆（这与 LSTM 是类似的，只是 LSTM 维护的是一个向量），该矩阵和一个神经网络共同进行学习和预测。记忆矩阵会通过选择性的读和写来进行迭代更新，同时由于每部分都是可微的，因此可以通过梯度下降法进行训练。NTM 的基本工作原理如下图：

记忆神经网络 [8] 主要用在长期记忆的推断，网络会从一个长文本中自动的将重要的信息编码后记录下来。最后产出的模型能回答关于长文本的任何问题——根据问题从记忆中寻找相关内容，然后产生答案。一个经典的 MemNN 的预测过程由简单的 4 步组成：

- 将输入  $x$  编码成一个隐向量  $I(x)$ 。
- 更新记忆  $m_i$ ,  $m_i = G(m_i, I(x), m)$ 。即通过当前的隐向量，当前记忆，整体记忆，去更新记忆中的一块内容。

图 5.8: Neural Turing Machine (NTM) 示意图

- 通过当天的记忆内容和输入决定输出向量。 $o = O(I(x), m)$
- 最后将输出向量解析成最终的回答。 $r = R(o)$

然而 MemNN 的一个问题在于并不能 End-to-End 的去学习,同时 NTM 和 MemNN 并没有关注输入的顺序信息。

3, 在个性化搜索中, 最为重要的是怎么去理解和认识一个淘宝的用户。除了用户的一些基本画像信息, 我们拥有最为关键的、与其他平台不同的数据是用户在淘宝上的行为。由于用户在淘宝上的行为天然是一个长期的行为序列, 因此很自然考虑使用 RNN 等序列模型取进行处理。一个最基础的模型结构如图5.9。

图 5.9: 淘宝序列匹配模型示意图

从图中我们可以看到网络主要由 3 大部分组成, 分别是: 1, 首先得到用户的行为序列; 2, 将用户的行为序列经过一个带记忆的网络编码成隐向量  $H$ ; 3,

将  $H$  通过多目标网络训练不同的目标。那么下面我们将从这三方面详细说明淘宝搜索中的序列匹配模型。

a) 首先是用户序列部分。我们使用用户有过行为的商品序列作为用户的表示, 每一个商品被 embedding 到一个 128 维的向量中, 这个向量可以从 word2vec 的方法进行无监督学习得到, 也可以从一个长期 fine tune 的深度神经网络得到。在获得商品表示的算法中, 一个商品 embedding 前的编码主要包括商品 ID、店铺、类目、价格信息。

图 5.10: 商品 embedding

商品的 embedding 部分是在训练训练网络前提前训练好的, 我们并没有将其放到训练序列的网络中, 主要是因为 ID 特征十分稀疏, 在一个 LSTM 的网络中, 数据可能并不支持训练这么大规模的特征维度, 从而影响模型的整体效果。然而这样的问题是, 预训练得到的商品只包含了基本的商品特征, 可能并不最适合当前的序列匹配网络。因此受 transfer learning 的启发, 向量经过 embedding 的商品向量并不是直接作为特征输入到 LSTM 或者 MemNN 中, 而是根据商品的行为类型 (点击、成交、收藏、加购) 和来源经过不同的卷积核生成一个新的 128 维向量, 然后输入到序列网络。这样既对用户的行为进行了区分, 可以学习到不同行为的重要性; 同时对预训练得到的商品向量往新的目标上调整。

b) 在得到用户的商品行为序列后, 我们需要使用一个序列或者记忆模型, 将序列编码成一个通用的用户状态  $H$ 。这里我们对比了 LSTM 和 End-to-end memory network[9]。LSTM 在上文中已经有过一些介绍, 而一个 End-to-end memory network 与经典的 memNN 的区别在于它可以通过一个整体的网络去学习, 基本的网络结构如图5.11。它首先将输入序列中的每一项同时映射成 2 个向量  $m_i$  和  $c_i$ , 分别表示“输入记忆”和“输出记忆”。“输入记忆”决定序列中每一项的重要性  $p_i$ ,  $p_i$  和  $c_i$  相乘求和得到输出向量  $o$ 。输出向量  $o$  和用户向量决定最终的答案  $a$ 。LSTM 使用一个记忆单元 *Cell* 去记忆历史信息; 而 End-to-end

memory network 正着重于将原始序列压缩, 并自动挖掘序列中元素的重要性。我们使用两种方法在大量数据上进行了实验对比, 从 AUC 来看 LSTM 会略优于 End-to-end memory network, 但是 End-to-end memory network 在计算速度上会远优于 LSTM。

$$c_i = \sigma(W_c * x_i) \quad (5.7)$$

$$m_i = \sigma(W_m * x_i) \quad (5.8)$$

$$p_i = \text{Softmax}(u^T m_i) \quad (5.9)$$

$$o = \sum_i p_i c_i \quad (5.10)$$

$$a = \text{Softmax}(W(o + u)) \quad (5.11)$$

图 5.11: End-to-end memory network

c) 在得到的用户的隐向量后, 进行匹配是容易的, 只需要一个不用太深的 DNN 网络对用户向量  $H$  和商品向量放到一起进行预测即可。但是为了学到更加鲁棒的网络结构, 我们使用 multi-task 的相关技术 [16] 建立了多个辅助目标共同学习。因为 multi-task learning 不是本章重点, 因此不再详细介绍, 下一小节会有相关对比结果。

4, 本小节详细介绍序列匹配模型的离线实验以及在线效果。离线部分抽取

淘宝搜索一周的用户行为日志，按用户随机分成 5 份，做交叉验证；在线部分使用的标准 A/B Test, 大于 3-5% 的流量作为测试样本。基本结论如下：

a) 对比序列模型和仅使用用户画像作为用户状态。使用序列模型的情况下，主目标匹配的 AUC 从 0.62 提升到 0.68；辅助目标 LTR 的 AUC 从 0.65 提升到 0.72；辅助目标价格档预测的准确率从 29% 提升到 41%。

b) 对比使用 multi-task 和不使用的情况下，匹配预测的准确率。在商品向量良好的情况下 (使用 DNN 做 pre-train)，AUC 能从 0.67 提升到 0.68, 提升非常有限；但是在商品向量并不理想的情况下 (使用 word-to-vec 初始化)，不使用 multi-task 方法的 AUC 只有 0.56，使用后 AUC 提升到 0.66，提升非常明显。

c) 对比不用的商品输入向量表示，分别使用了 DNN、word-to-vec 以及 DSSM 去预学习一个商品向量，各自或组合作为序列的输入。word-to-vec 向量表现的较差，而 DNN 以及 DSSM 的向量表现类似；多种向量 concat 到一起，结果与其中最优的比较一致。

d) 在在线实验中，我们通过序列模型对用户的购买力进行了预估，同时对商品的 CTR 进行了预估。其中购买力的部分的覆盖率较使用前提升了 50%，准确率提升了 30% 左右。CTR 预估部分，在 A/B Test 中，使得覆盖人群的点击率提升了 3% 左右。

## 5.3 实时个性化模型

淘宝搜索对海量用户、卖家之间提供服务。希望用户能有更好的搜索体验的同时，期望卖家也能有更好销售。传统搜索引擎对全部用户展示相同的搜索结果，结果会按相关性、质量进行排序；而个性化搜索更专注于不同的个体，根据每个用户属性与偏好为其定制排序。这样带来的好处是多方面的：首先用户会拥有更好的搜索体验，能看到更多自己偏好的内容从而促进更多的交易；然后会为卖家带来更优质的用户，店铺会吸引更多目标人群；从流量上来看，在搜索流量一定的情况下，个性化能减少无效流量的产生，提高流量的使用效率。

长期个性化在淘宝搜索上已经起着尤为重要的作用。然而存在的问题是，此类模型并不能及时的获取用户、店铺、商品的实时信息，因此在处理冷启动问

题(新用户、新商品)、实时兴趣/属性变化等问题上,表现的并不理想。实际上,该类问题是大量存在的,因此我们需要一个实时的个性化排序模型,结合用户的长期与实时兴趣,结合商品的静态与动态属性,为用户提供更好的排序服务。

### 5.3.1 整体框架

在传统的推荐系统中,我们可以根据用户的历史购买记录为用户推荐商品以尽可能地促成下一次成交。在众多方法之中,矩阵分解(Matrix Factorization)凭借其对隐空间的建模、对稀疏数据的适应能力以及强大的拟合能力得到了广泛的关注,并在很多实际场景中得到验证,例如Netflix 百万大奖赛。然而,搜索场景的特殊性必然使得我们直接套用传统的基于最小化平方损失(square loss)的MF模型的结果是悲观的,我们更关注的是序(rank),而不是某个绝对的偏好程度(preference level)。同时,我们注意到,用户群体的偏好也是实时变化的,如果突然某个时刻开始和你偏好相似的一批用户开始购买某个商品,我们希望能将这个变化及时捕捉到并实时地体现在对你的搜索排序结果中,即将这个商品尽可能排在结果页前面。因此,我们设计了一套针对排序优化的实时协同过滤框架,并在这个框架上设计和实现了2个算法。

在传统的隐空间分解模型中,我们为每个用户  $i$  学习一个向量  $U_i \in R^k$ , 为每一个商品  $j$  学习一个向量  $V_j \in R^k$ , 此时模型对用户  $i$  在商品  $j$  上的预测为

$$\hat{y}_{ij} = U_i^T V_j \quad (5.12)$$

针对不同的场景,我们可以设计不同的损失函数  $l(y_{ij}, \hat{y}_{ij})$  去学习  $U$  和  $V$ 。在实际搜索场景中,我们一般可以拿到更多的信息,例如用户相关的特征  $x_i^u \in R^{d_u}$ , 和商品相关的特征  $x_j^v \in R^{d_v}$ , 对这些信息的处理,通常的做法是分别利用用户特征和商品特征计算对应的相似度矩阵,并对其做特征值分解,用较大特征值对应的特征向量做 basis 去学习对应的  $U$  和  $V$ 。然而,我们都知道特征值分解的复杂度是  $(n^3)$ , 对于我们的规模(千万级用户,亿级别的商品)来说是不可接受的。因此,我们使用了一种更直接的方式,令

$$\begin{aligned} U_i &= W_i^1 x_i^u, W_i^1 \in R^{k \times d_u} \\ V_j &= W_j^2 x_j^v, W_j^2 \in R^{k \times d_v} \end{aligned} \quad (5.13)$$

此时，模型的预测值为

$$\begin{aligned} \hat{y}_{ij} &= (W_i^1 x_i^u) W_j^2 x_j^v \\ &= (x_i^u)^T (W_i^1)^T W_j^2 x_j^v \\ &= (x_i^u)^T W_{ij} x_j^v \end{aligned} \quad (5.14)$$

其中  $W_{ij} = (W_i^1) W_j^2$ ，等同于一个双线性模型 (bilinear model)。我们对以上两种预测模型分别设计了 2 个算法，并都实现了离线的基于 ODPS-GRAPH 的 BSP 并行训练和基于淘宝实时计算平台 PORA 的在线异步并行训练。下文将分别对此进行介绍。

### 5.3.2 实时矩阵分解 (Realtime Matrix Factorization for Ranking with Side Information)

假设我们有  $m$  个用户和  $n$  个商品，除了上文提到的用户向量  $U_i \in R^k$  和向量  $V_j \in R^k$  外，我们还为每个商品  $j$  学习一个 bias,  $b_j \in R$ 。此时的模型预测更新为

$$\hat{y}_{ij} = b_j + U_i^T V_j \quad (5.15)$$

在搜索排序中，我们更关注的是商品之间的顺序，而不是模型对偏好程度的预测值和真实值之间的平方误差或绝对值误差。从搜索日志中我们抽取训练三元组  $D = \{(i, j, k)\}$ ，其中每一个三元组  $(i, j, k)$  表示用户  $i$  更偏好商品  $j$ ，相比于商品  $k$ 。对于这样的一个三元组，我们定义损失函数如下

$$l(y_{ij}, y_{ik}, \hat{y}_{ij}, \hat{y}_{ik}) = \max(y_{ij} - y_{ik} - (\hat{y}_{ij} - \hat{y}_{ik}), 0) \quad (5.16)$$



其中,  $y_{ij}, y_{ik}$  是用户在商品  $j$  和  $k$  上的真实偏好程度, 例如对于一次搜索展现引导的行为, 可以设置偏好程度为: 成交 > 加购 > 收藏 > 点击 > PV。注意这里我们引入了 **Hinge loss**, 目的是当我们的模型排序正确时不去做冗余的抑制。此外, 我们还使用了已有的一份数据  $W \in \{0, 1\}^{n \times n}$ , 描述的是商品之间关于被购买行为的相似度矩阵。如果 2 个商品在被购买的行为上相似, 那么它们在隐空间的向量应该是尽可能靠近的。因此, 类似于经典的拉普拉斯特征映射 (Laplacian Eigenmap), 我们使用了一个正则项体现这项约束

$$\Omega(V) = \sum_{m \neq n} \|V_m - V_n\|^2 W_{mn} \quad (5.17)$$

最后, 我们需要优化的目标函数可以形式化为:

$$\begin{aligned} f(U, V, b) &= l(y_{ij}, y_{ik}, \hat{y}_{ij}, \hat{y}_{ik}) + \frac{\lambda_1}{2} \|b\|^2 + \frac{\lambda_2}{2} (\|U\|^2 + \|V\|^2) + \frac{\lambda_3}{2} \Omega(V) \\ &= \sum_{i \in S} \sum_{(j, k) \in P_i} \max(y_{ij} - y_{ik} - (b_j + U_i^T V_j - b_k - U_i^T V_k), 0) \\ &\quad + \frac{\lambda_1}{2} \|B\|^2 + \frac{\lambda_2}{2} (\|U\|^2 + \|V\|^2) \\ &\quad + \frac{\lambda_3}{2} \sum_{m \neq n} \|V_m - V_n\|^2 W_{mn} \end{aligned} \quad (5.18)$$

由于上述目标函数不像优化平方误差的矩阵分解, 因此也无法使用 ALS 进行加速求解。我们使用随机梯度下降对以上目标函数进行求解。

通常来说, 一个好的初始值将很大程度上加速学习的过程, 因此, 相比于从随机向量开始学习, 我们更希望为线上的在线学习提供一份通过离线训练的解作为初值。然而, 即使是离线训练, 对千万级用户, 亿级商品规模进行特定损失函数的矩阵分解, 也是集团现有的算法无法满足的。因此, 我们在 ODPS-GRAPH 上实现了我们的算法, 其详细过程为

1, 读入搜索日志, 建立 GRAPG 节点。具体地, 为每个用户和每个商品分别建立对应的节点, 存储对应的隐向量, 同时将用户相关的所有搜索展现点击购买记录存储在用户节点, 每个商品节点不存储任何日志信息。此外, 对每个商品, 我们要额外为其添加指向购买过该商品的所有的用户的边



2, 每个偶数超步内, 所有的商品节点从用户节点接受发送过来的梯度, 并用来更新自身的向量, 同时将更新后最新的向量发送给所有相关用户节点。

3, 每个奇数超步内, 所有的用户节点从商品节点接受发送过来的最新商品向量, 结合自己的用户向量和搜索日志, 计算对相关用户向量和商品向量的梯度, 最后更新自身向量, 并将商品向量梯度发送给相关商品节点

整个过程如下图所示:

我们将每轮迭代评估的每次搜索上的 NDCG 绘制出来如下图所示:

对于线上的在线学习部分, 由于实时个性化的需求, 我们需要当用户发生了一次行为之后, 可以在几分钟之内更新到模型, 并迅速影响到后续的搜索排序。因此, 我们只在搜索实时计算平台 PORA 上实现了如下的基于 parameter server 的异步并行架构, 每次用户行为日志被 Log parser 搜集后被送到某个 gradient worker, 计算在特定用户和商品上的梯度, 并发送到存储在 Hbase 上的 parameter server 进行参数更新, 并同时最新的用户/商品向量同步到 UPS 和引擎正排上, 实时影响线上排序效果。

### 5.3.3 实时双线性模型 (Realtime Matrix Factorization for Ranking with Side Information)

本节提出的实时双线性模型，是一个实时、pairwise、双线性模型。下面简要说明我们设计这样模型的原因。

Why real-time? 从商品角度上：□ 商品上架、下架是频繁的，新商品在长期个性化中没有特征，因此需要实时个性化处理；□ 同一线上商品的某些属性也是动态变化的，如图，即使热门商品的 CTR 也会随时间不断变化，因此长期属性不如实时准确（想象一下，春晚后某件主持人穿的礼服，以往排序下，搜索是排不出来的；但使用实时模型后，卖家上架后，大量用户搜索，该礼服能立刻排在前面）。从用户角度上：□ 用户的兴趣会变化的，长期兴趣并不永远是不变的；□ 新用户缺少信息；□ 同号用户的大量存在，不同用户共同使用同一账号会对长期个性化产生极大干扰

Why pairwise? 不得不说，Pairwise 的方法在搜索中会存在很大的问题，这主要是由于样本分布极不均匀造成的。例如，可以分析得到男性用户对女装的

CTR 反而大于对男装的 CTR；这并不表示对男性个性化的时候需要出女装，而是由于男性用户搜女装的时候，目标是更明确的。Pairwise 的方式能很好的解决该问题，它考虑的是 pair 间商品属性的差异，因此在商品属性相同的时候，参数不会更新。

Why bilinear? 用户在淘宝上的行为时丰富的，商品的属性也是丰富的，即使对冷启动的情况，我们也能获取用户的手机型号，ip 等信息，商品的类目、价格等信息。在有这些信息的情况下，我们利用它们能使模型的结果更加准确。

定义  $U$  为全体用户， $I$  为全部商品，用户反馈  $S$  表示 implicit feedback,  $S \subseteq U \times I$

□ 每个商品,  $\text{Item}(I \in R^C)$ , 由 2 部分组成, 分别是静态特征 (Static descriptors) 和动态特征 (Temporal characteristics)。前者包含价格、风格、性别等 (d), 后者包含实时 ctr、实时销量等。

□ 每个用户,  $\text{User}(U \in R^D)$ , 同样由 2 部分组成, 分别是特征 (U) 与用户 ID(B)。

□ 行为,  $r_{u,i}$  (Interactive Feedback), 其中  $r_{u,i}$  是 0/1 (实际上也可以是实数, 这里只考虑 0/1 的 implicit feedback), 表示用户  $u$  对商品  $i$  的偏好。

我们排序的目标是对用户设置全部商品上的排序方案,  $>_u \subset I^2$ , 其中  $>_u$  必须满足:

$$\begin{cases} \forall i, j \in I : i \neq j & \Rightarrow i >_u j \vee j >_u i \\ \forall i, j \in I : i >_u j \wedge j >_u i & i = j \\ \forall i, j \in I : i >_u j \wedge j >_u k & i >_u k \end{cases} \quad (5.19)$$

另外, 我们定义  $I_u^+ = \{i \in I : (u, i) \in S\}$  为用户  $u$  点击的全部商品,  $U_i^+ = \{u \in U : (u, i) \in S\}$  为点击过商品  $i$  的全部用户。在 bilinear model 中, 每个用户对商品的偏好表示为:

$$s_{i,j} = \sum_{a=1}^C \sum_{b=1}^D U_{i,b} I_{j,a} W_{a,b} + \sum_{a=1}^C B_{i,a} I_{j,a} \quad (5.20)$$

$D$  和  $C$  分别表示用户和商品的特征维度,  $B$  表示每个用户 ID 在商品特征上的偏移。将用户的  $B$  看做自身 feature 的一个稀疏维度, 那么考虑商品有静态

和动态 2 类属性，如下图所示，该模型可以看做是将用户 feature 的每个维度向 item 的每个维度做投影，然后在商品空间上做点乘， $W$  表示投影矩阵。

有了上面表示后，我们需要用日志数据去 fit 我们的模型，根据贝叶斯公式，我们求一个最大后验估计 (MAP)。假设所有参数表示为  $\theta$ ，则后验分布为：

$$p(\theta | y_u) \propto p(y_u | \theta)p(\theta) \quad (5.21)$$

我们算法的目标是最大化该后验。在计算梯度公式后，简单的使用随机梯度下降法就能求解该问题，但是速度和精度并不理想。为此，我们尝试了使用 bootstrap sampling 方法和 adaptive oversampling 方法，最终采用的是后者。整个模型的训练也包括 2 部分，离线学习和在线学习。离线部分主要是学习 bilinear 部分中的矩阵  $W$ ，这部分认为是稳定的，作为在线部分的初始值。在线部分主要是学习每个用户 ID 对应的属性偏好，捕捉用户当前的信息。学习方法类似上一节的矩阵分解模型，这里不再重复。

### 5.3.4 实验与结果

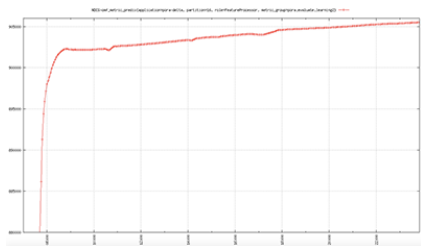
我们实时个性化算法的目标是在淘宝搜索中提升用户的搜索体验，提高流量效率。从指标上表现为搜索的 NDCG/MRR 等指标，以及线上的 CTR 等指标。实时模型于 7 月底上线手淘无线，rank 仍在调整权重中，但是线上真实的 NDCG 和 MRR 值已经可以获取。

下图 1 是矩阵分解模型的线上 NDCG 变化曲线，图 2 是双线性模型的线上 MRR 变化曲线，日期相同。从结果上，我们可以看到，随着每天实时模型的运行，NDCG 和 MRR 均能产生大幅的提升。这表示一天中，用户点击、成交的商品，在结果页中越来越靠前，即获得的排序能更贴近用户的兴趣。

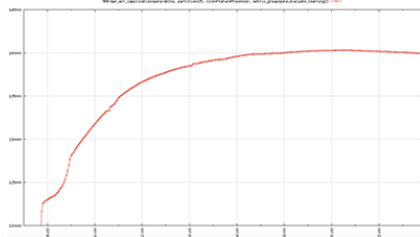
下面 2 张图是一天的模型曲线，左图是矩阵分解的 NDCG 变化情况，右图是双线性模型的 MRR 变化情况。除了可以看出曲线的上升外，我们可以看到，矩阵分解模型直到每天结束仍未收敛到最优状态，曲线上升趋势依然明显；双线性模型在每天最后，曲线基本不会上升，甚至有略微下降。因此我们仍然需要对模型参数进行调整，例如前者应该加大学习率，后者则应有所降低。

## 5.4 排序学习

@ 元涵，@ 凌运，@ 龙楚



(a) Recall rate of Price Level



(b) Precision of Price Level

## 5.5 展示学习

个性化展示 @ 席奈个性化短标题: @ 苏哲, @ 仁重

## 5.6 深度学习时代的排序模型优化

### 5.6.1 用户行为建模

用户建模是搜索与推荐模型的核心技术。淘宝搜索排序算分的对象是  $\langle \text{user}, \text{query}, \text{item} \rangle$  三元组, 我们从样本特征表达的角度上来看,  $\text{item}$  是比较稠密而且稳定的部分, 在大样本的环境下, 大部分信息都能够被  $\text{id embedding}$  所表达, 相反  $\text{user}$  是三者中比较稀疏的部分, 所以对于  $\text{user}$  的描述, 需要大量的泛化特征。从模型分类的角度上来看, 用户与商品的静态特征作用在于增强模型的泛化性, 而用户实时行为的引入与建模, 可以大大增强样本之间的区分性, 显著地提升模型的分类精度。我们把用户建模的过程看作是对用户的信息抽象和信息组织的过程。信息抽象方面我们不断地优化与丰富建模方式: 1.  $\text{user profile}$  用来表征用户的静态属性信息; 2. 偏好标签的挖掘, 从行为上预测用户的一般性偏好; 3. 实时行为建模, 更细粒度的对当前请求下的兴趣刻画与描述。信息处理方面, 我们从行为周期和行为内容方面对用户行为数据进行合理的组织: 1. 从行为周期上, 我们将行为序列划分成中短期和长期, 分别使用不同的时间跨度, 描述不同粒度的兴趣; 2. 从行为内容维度上, 直接行为反馈商品和曝光商品分别被用来显式和隐式的表达用户意图, 与此同时, 我们也将用户行为数据从传统的

电商商品，延伸到一些泛内容信息；

### 5.6.2 大规模深度排序模型的实时更新与热启动

超大模型如何实现快速更新，我们探索了 Multi-Layer Multi-Frequency 的更新方法：

增量训练模式下，线上 base 模型已经训练了很多天的数据，新模型（加特征，改结构）如何快速恢复 base 模型的效果，一步就站在 base 模型巨人的肩膀上继续往上走，而不是慢慢训练苦苦追赶？我们实现一个成熟的解决方案，并且已经全面应用起来

### 5.6.3 大规模深度学习模型的量化 & automL

## 5.7 轻量排序融合框架

工业级排序系统发展至今，单靠深度精排大模型已无法满足业务需求，大模型之后通常配备以效率为目标的离线/实时 LTR (Learning-To-Rank) 与重排序 (Re-Rank)，而纯业务驱动的沙盘调控、价格调节、新品与包邮置顶、打散与择优等策略再对排序结果施加影响。在 AliExpress (AE) 搜索排序系统中，上述若干子模型通常对排序分做加法和乘法来修改排序结果，从而实现其目标。典型地，最终排序分可以表示成的形式；其中，子模型利用加法/线性组合改变最终排序分，而子模型利用乘法改变最终排序分；加法的影响尚且可控，而乘法的影响则牵一发而动全身。实际上，排序链路中常常面临多个子模型竞争最终排序结果的困境。倘若不加以限制，子模型彼此独立加分或乘分，排序分将很快膨胀，而各路子模型在最终排序结果中所占比重，也将混沌一团、无从分析。

从宏观上看，各路排序模型的目标可分为两类：效率驱动与业务驱动。深度精排、LTR、重排序等模型属于前者，而针对货品、卖家等等的流量调控与排序规则属于后者。一个自然的想法是，将效率驱动与业务驱动的子模型加分逻辑分开，凯撒的归凯撒，上帝的归上帝。业务驱动的子模型不是本文关注的重点，暂且按下不表。针对效率驱动的子模型，既然它们有着共同的目标，我们有

理由相信，这些子模型对商品的排序有内在一致性；人为对排序分做加法或乘法，由于加数和乘数的超参数难以调节（例如，上文公式中的），超参数过大导致对应的子模型具有排他性，而过小则对应子模型的影响无法体现；商品排序的内在一致性被忽略或者破坏，从而让某几路子模型主导排序结果，其他子模型的贡献被抹去。

受社会学投票理论（Voting Theory）中的排序聚合（Rank Aggregation）方法启发，我们不再关注排序分，而是转向排序的顺序本身，根据商品在各个排序结果中的相对位置，获得各路子模型聚合后的排序结果。将子模型的排序结果看作是商品集合的一个置换（Permutation），商品的打分被忽略，只剩下彼此的顺序关系。给定一组置换 [1]，通过定义置换之间的距离，排序聚合方法找到这组置换的平均值，把它当作聚合之后的排序。对应于投票理论，排序系统中的子模型相当于投票者，而商品相当于候选人；投票理论和排序系统的目标，都是要找到一组最优的最终排序，最大程度满足所有投票者/子模型的好恶。

然而，阿罗悖论（Arrow Paradox）证实，不存在一种投票制度能够满足所有标准 [2]。因此，我们需要确定，各路子模型排序结果的合并中，哪些标准是必须遵守的，哪些标准是可以放弃的？例如，阿罗悖论中提及的非独裁标准，即，“不存在一个投票者，使得投票结果总是等同于他的排序”，在排序系统中显然可以放弃。经过文献调研与场景分析，本文主要考虑投票制度中的孔多塞标准（Condorcet Criterion），即，“如果一个候选人在成对比较里击败了所有候选人，则他一定排在第一位”。前人研究 [3] 表明，定义置换间的距离为逆序对（Discordant Pair）的个数，即，Kendall Tau 距离 [4]，可以得到满足孔多塞标准的投票结果，本文也将依照这一原则。

当下工业界也存在聚合多路子模型打分的排序模块，比如，协调（Ensemble）多路排序结果的 LTR 模型，以及着眼于全页面优化的重排序。与 LTR 相比，经典的排序聚合不需要监督信息，并且以整页 PV（Page View）为单位优化目标。与重排序相比，经典的排序聚合不需要用户和商品特征；另外，排序聚合是针对上下游链路联合优化的尝试，目标是全链路指标最大化，而重排序的重点是商品候选集内排序的最优化。事实上，排序聚合与 LTR、重排序完全可以共存，LTR 可以作为排序聚合的一路子模型输入，而排序聚合可以作为重排序的上游，



为重排序提供更加多样化的商品集合。以 AE 搜索排序系统为例，我们观察到，LTR 模型对 Top20 商品的排序有优势，而精排大模型对 Top20 之后的商品排序效率更高，即，一个较优的策略是：Top20 商品按照 LTR 模型打分排序，而 Top20 之后的商品按照精排大模型打分排序。现有的模型无法做到这一点。

## 参考文献

- [1] 淘宝搜索全链路有效行为量化模型 (UBM&UCM),  
<http://www.atatech.org/articles/38550>
- [2] User Browsing Model 的实现与应用, <http://www.atatech.org/articles/23111>
- [3] 搜索个性化介绍, <http://www.atatech.org/articles/48548>
- [4] Mikolov, T., Karafi'at, M., Burget, L., Cernock'y, J., Khudanpur, S.: Recurrent neural network based language model. J. Interspeech. 1045–1048 (2010)
- [5] Hochreiter, S., Schmidhuber J.: Long short-term memory. J. Neural Computation. 9(8), 1735–1780 (1997)
- [6] Learning Long-Term Dependencies with Gradient Descent is Difficult
- [7] Graves, A., Wayne, G., Danihelka, I.: Neural Turing Machine. arXiv preprint:1410.5401v2 (2014)
- [8] Weston, J., Chopra, S., Bordes, A.: Memory Networks. C. International Conference on Learning Representations. arXiv:1410.3916 (2015)
- [9] Sukhbaatar, S., Szlam, A., Weston, J., Fergus, R.: End-To-End Memory Networks. J. Advances in Neural Information Processing Systems. 28, 2440–2448 (2015)

- [10] Gers, Felix A and Schmidhuber, J: Recurrent Nets that Time and Count. J. in IJCNN 2000
- [11] Cho, Kyunghyun and Van Merriënboer, Bart and Gulcehre, Caglar and Bahdanau, Dzmitry and Bougares, Fethi and Schwenk, Holger and Bengio, Yoshua: Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. arXiv preprint arXiv:1406.1078 (2014)
- [12] Koutnik, Jan and Greff, Klaus and Gomez, Faustino and Schmidhuber, Juergen: A Clockwork RNN. J. arXiv preprint arXiv:1402.3511 (2014)
- [13] Zaremba, Wojciech: An Empirical Exploration of Recurrent Network Architectures. in LMLR 2015
- [14] Greff, Klaus and Srivastava, Rupesh K and Koutnik, Jan and Steunebrink, Bas R and Schmidhuber, J: A search space odyssey. IEEE transactions on neural networks and learning systems (2016)
- [15] Misra, Ishan and Shrivastava, Abhinav and Gupta, Abhinav and Hebert, Martial: Cross-stitch networks for multi-task learning. CVPR (2016)

# 第六章 智能决策体系的建立

## 学习目标与要求

### 6.1 基于 MAB 的排序策略优化

### 6.2 MAB

在许多预测问题中，预测器在采取某个行为之后，能够衡量其损失（或收益），但是如果他选择了另一种可能的行为，那么他就没有机会知道了。这种预测问题被称为 **multi-armed bandit problems**。这个名字来源于一个在玩着老虎机池的赌徒，每次将赌注放在一个可能不同的老虎机上，每个老虎机以一定的概率吐出硬币，但是这个概率赌徒并不知道，赌徒的目标是赢得几乎像预先知道哪个老虎机将返回最大奖励的总奖励。因此，**multi-armed bandit problem** 的目标是使得在每个时刻选择动作的累积损失无限趋近于理想情况下的最小的损失：

$$\lim_{n \rightarrow \infty} \frac{1}{n} \left( \sum_{t=1}^n l(I_t, Y_t) - \min_{i=1, \dots, N} \sum_{t=1}^n l(i, Y_t) \right) = 0$$

若每个动作对应的奖赏是一个确定值，那么尝试一遍所有的动作便能找出奖赏最大的动作。然而，更一般的情况是一个动作的奖赏值来自于一个概率分布，仅通过一次尝试并不能确切地获得平均奖赏值。若仅为获知每个动作的期望奖赏，

则可采用“仅探索”(exploration-only)法, 将所有的尝试机会平均分配给动作, 最后以每个动作的平均奖赏作为其奖赏期望的近似估计; 若仅为执行奖赏最大的动作, 则可采用“仅利用”(exploitation-only)法, 选择目前最优的动作。显然, “仅探索”法能很好的估计每个动作的奖赏, 却会失去很多选择最优动作的机会; “仅利用”法则相反, 它没有很好的估计动作的奖赏, 很可能经常选不到最优的动作。因此, 两种方法都难以使最终的累积资深最大化。事实上, “探索”和“利用”这两者是矛盾的, 因为尝试次数有限, 加强了一方则会削弱另一方, 即“探索-利用窘境”, 显然, 欲使得累积奖赏最大, 则必须在探索与利用之间达成较好的折中。 $\epsilon$ -贪心法基于一个概率来对探索和利用进行折中, 每次尝试时, 以 $\epsilon$ 的概率进行探索, 即以均匀概率随机选取一个动作; 以 $1 - \epsilon$ 的概率进行利用, 即选择当前平均奖赏最高的动作。若动作奖赏的不确定性较大, 例如概率分布较宽时, 则需要更多的探索, 此时需要较大的 $\epsilon$ 值; 若动作的不确定性较小, 例如概率分布较集中时, 则少量的尝试就能很好的近似真实奖赏, 此时需要的 $\epsilon$ 较小。然而, 若尝试次数非常大, 那么在一段时间后, 动作的奖赏都能很好的近似出来, 不需要再探索, 这种情况下, 可以令 $\epsilon$ 随着尝试次数的增加而逐渐减小, 例如令 $\epsilon = 1/\sqrt{t}$ 。Softmax 算法基于当前已知的动作平均奖赏来对探索和利用进行折中。若各个动作的平均奖赏相当, 则选取各策略的概率也相当; 若某些策略的平均收益明显高于其他策略, 则他们被选取的概率也明显更高。Softmax 算法中的策略概率的分布是基于 Boltzmann 分内布:

$$P(k) = \frac{e^{\frac{Q(k)}{\gamma}}}{\sum_{i=1}^K e^{\frac{Q(i)}{\gamma}}}$$

其中,  $Q(i)$  记录当前策略的平均收益;  $\gamma > 0$  称为“温度”,  $\gamma$  越小则平均收益高的策略被选中的概率越高。 $\gamma$  趋于 0 时 Softmax 将趋于“仅利用”,  $\gamma$  趋于无穷大时 Softmax 则将趋于“仅探索”。 $\epsilon$ -贪心法与 Softmax 算法孰优孰劣, 主要取决于具体应用。可以证明, 当尝试次数足够大时, 都可以收敛到最优解。

**Algorithm 1** Multi-armed bandit problem**Parameters:** 策略数  $N$ ,  $0 \leq \beta, \eta, \gamma \leq 1$ .**Initialization:**  $w_{i,0} = 1$  and  $p_{i,1} = \frac{1}{N}$ ,  $i = 1, \dots, N$ 对每一轮  $t = 1, 2, \dots$ 

- (1) 根据概率  $p_t$  选择一个策略  $I_t \in 1, \dots, N$ ;
- (2) 计算估计值

$$g'(i, Y_t) = \tilde{g}(i, Y_t) + \frac{\beta}{p_{i,t}} = \begin{cases} \frac{(g(i, Y_t) + \beta)}{p_{i,t}} & \text{if } I_t = i \\ \frac{\beta}{p_{i,t}} & \text{otherwise} \end{cases}$$

- (3) 更新权重  $w_{i,t} = w_{i,t-1} e^{\eta g'(i, Y_t)}$
- (4) 更新概率分布

$$p_{i,t+1} = (1 - \gamma) \frac{w_{i,t}}{W_t} + \frac{\gamma}{N}, \quad i = 1, \dots, N.$$

**6.2.1 一种改进的 bandit 策略**

在实使用中, 我们使用的是一种改进的 bandit 策略, 能显著提高收敛的速度。定义策略  $i$  的收益  $g(i, Y_t) = 1 - l(i, Y_t)$ , 其估计值为:

$$\tilde{g}(i, Y_t) = \begin{cases} \frac{g(i, Y_t)}{p_{i,t}} & \text{if } I_t = i \\ & i = 1, \dots, N. \\ 0 & \text{otherwise} \end{cases}$$

其中,  $\mathbf{E}[\tilde{g}(i, Y_t) | I_1, \dots, I_{t-1}] = g(i, Y_t)$ , 所以  $\tilde{g}(i, Y_t)$  是  $g(i, Y_t)$  的无偏估计。更新算法如 Algorithm 1。

在实际中, 我们选取  $\beta = \sqrt{\frac{1}{nN} \ln \frac{N}{\delta}}$ ,  $\gamma = \frac{4N\beta}{3+\beta}$ ,  $\eta = \frac{\gamma}{2N}$  (有理论证明, 参数这样选取时, 总的损失与理想情况下的最小的损失之差正在上界)。

## 6.3 Zero-order optimization

零阶优化之所以称之为零阶，就要是因为它只用到了因变量，而不用到它的偏导数，它是在一定次数的抽样基础上，拟合目标函数的响应函数，从而寻求最优解，优化处理器通过随机搜索建立变量和目标函数的逼近。由于是随机搜索，收敛的速度可能很慢，因此，一般用户可以通过给出多个合理的起始设计来加速收敛 (例如 MAB)。零阶优化只简单的运行一系列的随机搜索并删除所有不合理的设计，也可以运行多次单独的循环，并在每次运行前指定新的设计变量序列来生成起始设计序列。

零阶方法与一阶方法的对比：

1、零阶方法 (直接法)：是一种直接的最常用优化的方法，只用到因变量而不利用一阶导数信息；

2、一阶方法 (间接法)：使用偏导数，即使用因变量的一阶导数。

一阶方法计算量大，结果精确。但是，精度高并不能保证所得的就是最优解。一阶方法可能在不合理的设计序列上收敛，如果起点很接近局部最小值的话，这时可能找到了一个局部最小值，或是不存在合理的设计空间。而零阶方法可以更好的研究整个设计空间，也可以先运行随机搜索，确定合理的设计空间。

零阶优化一般用于复杂的黑盒场景下，即输入变量  $x$ ，我们知道  $f(x)$  的输出，但  $f$  可能是一个复杂变化的黑盒场景 (非突、不可导、不连续、多模态等)，我们并不知道其具体的函数形式，这时可以利用零阶优化方法求得  $x^* = \operatorname{argmin}_{x \in R^d} f(x)$ 。零阶优化有许多方法，比如随机搜索、Shubert Algorithm[?]、Nelder–Mead Simplex[?]、Random Gradient Method[?]、Random Pursuit[?] 等。

### 6.3.1 Extra Gradient Algorithm

在实际使用中，我们使用的是一种叫做 Extra Gradient 的算法。促销活动变化、时间变化、用户变化、流量变化等都影响着函数  $f$ 。因此， $f$  的具体形式是不知道的，令  $f_t(\cdot) \square t = 1, \dots, T$  为优化的序列函数，假定所有的函数  $f_t(\cdot)_{t=1}^T$  满足  $|\nabla f_t(x') - \nabla f_t(x)| \leq \beta |x - x'|$ ，其中  $x \in \Omega$ ， $x' \in \Omega$ ，即目标函数是平滑的。定义

**Algorithm 2** The Extra Gradient Method for Bandit Learning**Input:** Step size  $\eta > 0$ **Output:** Solution  $x_*$ 

1. Initialize  $x_1$  and  $z_1$
2. Sample  $u_1$  from  $\mathcal{N}(0, I_d)$
3. **for**  $t=1, \dots, T$  **do**
4. Sample  $u_{t+1}$  from  $\mathcal{N}(0, I_d)$
5. Call zero order oracle  $\mathcal{A}$  with  $(\mathbf{x}_t, \mathbf{u}_t)$  and receive feedback  $g_t = \mathbf{u}_t^T \nabla f(\mathbf{x}_t)$
6. Call zero order oracle  $\mathcal{A}$  with  $(\mathbf{x}_t, \mathbf{u}_{t+1})$  and receive feedback  $h_t = \mathbf{u}_{t+1}^T \nabla f(\mathbf{x}_t)$
7. Update  $\mathbf{z}_{t+1}$  by

$$\mathbf{z}_{t+1} = \pi_{\Omega}(\mathbf{z}_t - \eta d g_t \mathbf{u}_t) = \operatorname{argmin}_{\mathbf{z} \in \Omega} \frac{1}{2} |\mathbf{z} - \mathbf{z}_t|^2 + \eta d g_t \mathbf{u}_t^T (\mathbf{z} - \mathbf{z}_t)$$

8. Update  $\mathbf{x}_{t+1}$  by

$$\mathbf{x}_{t+1} = \pi_{\Omega}(\mathbf{z}_{t+1} - \eta d h_t \mathbf{u}_t) = \operatorname{argmin}_{\mathbf{z} \in \Omega} \frac{1}{2} |\mathbf{x} - \mathbf{z}_{t+1}|^2 + \eta d g_t \mathbf{u}_{t+1}^T (\mathbf{x} - \mathbf{z}_{t+1})$$

9. **end for**

10. Set  $\mathbf{x}_* = \mathbf{x}_T + \mathbf{1}$

oracle  $\mathcal{A}$ : 输入  $x \in \Omega$  和方向向量  $u$ , 则 oracle  $\mathcal{A}$  会输出  $u^T \nabla f_t(x)$ 。oracle  $\mathcal{A}$  可以通过 A/B 测试得到  $f(x+\delta u)$  和  $f(x-\delta u)$ , 则  $u^T \nabla f_t(x) = (f(x+\delta u) - f(x-\delta u)) / [2\delta]$ 。我们的目标就是利用 oracle  $\mathcal{A}$  优化  $\sum_{t=1}^T f_t(x)$ 。具体算法如下:

其中,  $g_t \mathbf{u}_t$  和  $h_t \mathbf{u}_t$  是  $\nabla f_t(x)$  的无偏估计, 即  $E[d g_t \mathbf{u}_t] = E[d \mathbf{u}_t \mathbf{u}_t^T] \nabla f_t(x) = \nabla f_t(x)$ ,  $E[d h_t \mathbf{u}_{t+1}] = E[d \mathbf{u}_{t+1} \mathbf{u}_{t+1}^T] \nabla f_t(x) = \nabla f_t(x)$ 。我们的实验表明, 该方法收敛速度快, 在我们的应用场景中取得了不错的效果。



## 6.4 应用

而在淘宝的搜索场景中，1、传统的 ltr(learning to rank) 都是通过假定优化目标函数，利用用户的历史行为数据，优化求解，而真实的情况是我们线上排序系统非常复杂，更像一个只知道输入和输出的黑盒系统，因此，任何目标函数的假设，都与线上真实的目标有一定的 gap；

2、所有模型的优化目标都是 NDCG，而 NDCG 高，并不代表实际成交额就高；

3、在双 11 等大促期间，搜索流量变化非常快，与平时差异很大，因此，平时最优的排序策略不一定适合大促流量的变化。

因此，MAB+zero-order optimization 是解决上述问题的一个比较好的方案。

### 6.4.1 算法流程

淘宝实际线上排序系统特征多，参数空间大，同时，我们希望整体收益最大化，即对不好的策略投入做尝试的流量尽量少，因此，就要求尽快的收敛到比较好的策略，因此，我们设计的算法流程图如下：

在 mab 收敛之前，为了保证不会把比接收桶还要差的策略发送过去，我们将概率  $p_{i,t}$  最大的策略与接收桶比较，如果确实比接收桶好，才发送到接收桶；在 mab 收敛之后，因为实际最优的策略有可能并不在已知的有限策略集里，因此，我们在以收敛的策略为起点，继续采用 extra gradient 的优化方法，继续在连续空间寻优。同理，在发送到接收桶之前，我们仍会采用与接收桶 PK 的方式，才决定是否发送到接收桶。

判断 mab 是否收敛的原则是当前出现某一个策略的概率明显且稳定几轮都比其他策略更大时，我们认为在当前以及未来的一段时间内，都会稳定的收敛到该策略。这时，以该策略为 extra gradient 迭代的初始点，在连续空间内探索更优的策略。收敛规则比较严格是为了保证在一段时间内 mab 的最优策略的稳定，在收敛之前的最优策略可能会在某几个相对好的策略之间波动，这些策略可能也是比基准桶要好，因此，为了整体收益最大，我们也会通过与基准桶 pk 的方式，判断是否发送到基准桶。

图 6.1: 算法流程

我们将线上特征的权重进行高、中、低划分，并剔除了一些不靠谱的策略，最终，我们选择了  $N=18$  个候选策略组成了策略集， $\mu_1, \dots, \mu_N$  是  $N$  个策略收益分布的均值，每轮  $t$  选择某一个策略  $I_t = i$ ，根据线上的用户的反馈情况，都可以得到其收益  $g(i, Y_t)$ ，因此，在  $T$  轮之后，定义损失函数  $\rho = T\mu^* - \sum_{t=1}^T g(i, Y_t)$ ， $\mu^*$  是每轮最大收益的均值，即  $\mu^* = \max_n \mu_n$ 。如果每一轮都选择最优的策略，那么损失  $\rho$  就会等于 0。为了使得最终收益的最大化，选取小部分流量（比如 5%）做 bandit，剩余的大部分流量作为接收桶，通过 Algorithm 1，找到不同时间段的最优策略。

## 6.4.2 计算收益用到的累积时间

在线上实时计算策略收益的时候，一个非常重要的问题是选择的实时数据累积的时间的长短，如果时间选取得长，则收益计算的波动小，更稳定，但是相应的算法迭代的轮数就少，收到的反馈少，不容易及时发现当前时刻的最优策略。在实时应用中，我们是以成交为目标，而成交一般是具有延后性，即成交与点击之间有一定时间的间隔。因此，在尽量拿到更多的成交的前提下，选择更短的时间，以增加迭代次数。我们对成交发生的时间进行分析，如下图：横

图 6.2: 成交笔数随时间分布

轴表示发生点击之后的时间区间，纵轴表示在该时间区间内成交的占比。因此，

我们可以大致估算出，在当前时刻前 30 分钟内成交的占比大概约 53%，而当前时刻前 60 分钟内，成交占比约 85%，显然，如果选择 60 分钟迭代一轮，显然太慢，不能满足大促的时候，快速发现最优的策略；因此，我们采用延后 30 分钟的收益计算方式，即计算当前时间前 [60,30) 的时间区间内的收益，来更新模型，即保证累积更多的成交，使得收益更加稳定，又尽可能增加模型替代次数。

### 6.4.3 收益的计算

在实际应用中，可以采用转化率，或成交笔数，或成交额等做为收益。但由于受各种噪音等影响，计算得到的收益可能波动比较大，或者各个策略的 reward 的差异非常小，也会影响模型收敛。因此，需要对收益进行平滑、归一化等处理。理论上，如果迭代次数足够，采样点实际收益的波动对最终策略的收益的预估是没有影响的，因此  $E[\tilde{g}] = g_i$ ， $\sum \tilde{g}_i \approx \sum g_i$ 。但是，如果在活动期间，特别是双 11 当天，需要在少量的几轮迭代就能快速找出最优的策略，同时，还能根据流量的变化，探索到最优的策略，同时，mab 在计算当前概率  $p_i$  的时候，会考虑历史累积的收益，因此，策略的波动就会产生不小的影响。

容易想到，将策略的收益减去基准的收益，以消除 bias。事实确实如此，但这还不足以完全消除波动以及扩大收益之间的差异。另一方面，我们发现，在选取的 [60,30) 时间区间内，我们仅仅用了最后一个累积收益来表示该时间的最终收益，利用的信息太少，如果波动发生在最后的 5 分钟，则完全因为这 5 分钟的波动，使得收益的计算不准确。基于该想法，我们在 30 分钟内采样了多个点，计算两个策略分布的差异，因此，收益计算公式为：
$$\frac{\text{ave}(\text{reward}) - \text{ave}(\text{base reward})}{\sqrt{\text{var}(\text{reward}) * \text{var}(\text{base reward})}}$$
。最后，采用反高斯归一化方法 (如下图) 对计算得到的收益进行归一。

### 6.4.4 Zero-order optimization

mab 的策略集可以看作是对搜索的参数空间的一种划分，通过 mab，可以排除掉不靠谱的参数，大大缩小搜索的参数空间。但是，确定的策略集一定是有限且离散的，全局最优的策略很可能并不在策略集里，因此，mab 收敛 (某一个或几个策略的概率要明显高于其他策略) 到的策略一般也不是最优的。在这个

图 6.3: 高斯归一化函数

问题中，我们只知道执行某个策略，并得到相应的收益，不知道优化的目标函数，所以，就没办法对目标函数求导，得到梯度方向，这正好是一个 zero-order 优化问题。在实际中，我们以 mab 收敛的策略为起点，采用 Extra Gradient 方法来继续在连续的参数空间里寻优。同理，在发送到接收桶之前，我们仍会采用与接收桶 PK 的方式，才决定是否发送到接收桶。

## 6.5 强化学习简介

### 6.5.1 背景

随着搜索技术的持续发展，我们已经逐渐意识到监督学习算法在搜索场景的局限性：

- 搜索场景中，只有被当前投放策略排到前面的商品，才会获得曝光机会，从而形成监督学习的正负样本，而曝光出来的商品，只占总的召回商品中的很小一部分，训练样本是高度受当前模型的 bias 影响的。
- 监督学习的损失函数，和业务关注的指标之间，存在着不一致性
- 用户的搜索、点击、购买行为，是一个连续的序列决策过程，监督模型无法对这个过程进行建模，无法优化长期累积奖赏。

与此同时，强化学习的深度学习化，以及以 Atari 游戏和围棋游戏为代表的应用在近几年得到了空前的发展，使得我们开始着眼于这项古老而又时尚的技术，并以此为一条重要的技术发展路线，陆陆续续地在多个业务和场景，进行了强化学习建模，取得了一些初步成果。我们也深知，目前强化学习的算法理论上和工业界中大规模噪声数据之间，还存在着很大的 gap，需要有更多的智慧去填补。

## 6.6 基于强化学习的实时搜索排序调控

### 6.6.1 背景

淘宝的搜索引擎涉及对上亿商品的毫秒级处理响应，而淘宝的用户不仅数量巨大，其行为特点以及对商品的偏好也具有丰富性和多样性。因此，要让搜索引擎对不同特点的用户作出针对性的排序，并以此带动搜索引导的成交提升，是一个极具挑战性的问题。传统的 Learning to Rank (LTR) 方法主要是在商品维度进行学习，根据商品的点击、成交数据构造学习样本，回归出排序权重。尽管 Contextual LTR 方法可以根据用户的上下文信息对不同的用户给出不同的排序结果，但它没有考虑到用户搜索商品是一个连续的过程。这一连续过程的不

同阶段之间不是孤立的，而是有着紧密的联系。换句话说，用户最终选择购买或不够买商品，不是由某一次排序所决定，而是一连串搜索排序的结果。

实际上，如果把搜索引擎看作智能体 (Agent)、把用户看做环境 (Environment)，则商品的搜索问题可以被视为典型的顺序决策问题 (Sequential Decision-making Problem)：(1) 在用户每一次请求 PV 时，Agent 做出相应的排序决策，将商品展示给用户；(2) 用户根据 Agent 的排序结果，给出点击、翻页等反馈信号；(3) Agent 接收反馈信号，在新的 PV 请求时做出新的排序决策；(4) 这样的过程将一直持续下去，直到用户购买商品或者退出搜索。以前向视角 (Forward View) 来看，用户在每个 PV 中的上下文状态与之前所有 PV 中的上下文状态和 Agent 的行为有着必然因果关系，同一个 PV 中 Agent 采取的不同排序策略将使得搜索过程朝不同的方向演进；反过来，以后向视角 (Backward View) 来看，在遇到相同的上下文状态时，Agent 就可以根据历史演进的结果对排序策略进行调整，将用户引导到更有利于成交的 PV 中去。Agent 每一次策略的选择可以看成一次试错 (Trial-and-Error)，在这种反复不断地试错过程中，Agent 将逐步学习到最优的排序策略。而这种在与环境交互的过程中进行试错的学习，正是强化学习 (Reinforcement Learning, RL) 的根本思想。

强化学习最早可以追溯到巴甫洛夫的条件反射实验，它从动物行为研究和优化控制两个领域独立发展，最终经 Bellman 之手将其抽象为马尔可夫决策过程 (Markov Decision Process, MDP) 问题而完成形式化。对于环境反馈的有利奖赏，Agent 将强化引发这种奖赏的动作，并在以后与环境交互的过程中更偏向于执行该动作。我们尝试将强化学习方法引入商品的搜索排序中，以优化用户在整个搜索过程中的收益为目标，根据用户实时行为反馈进行学习，实现商品排序的实时调控。图 1 比较直观地展示了的用强化学习来优化搜索排序的过程。如图所示，在三次 PV 请求之间，Agent 做出了两次排序决策 ( $a_1$  和  $a_2$ )，从而引导了两次 PV 展示。从效果上来看， $a_1$  对应 PV 中并没有发生商品点击，而  $a_2$  对应 PV 上发生了 3 次商品点击。如果将商品点击看成是对排序策略的反馈信号，那么 Agent 第二次执行的排序策略  $a_2$  将得到正向的强化激励，而其第一次排序策略  $a_1$  得到的激励为零。本文接下来的内容将对我们具体的方案进行详细介绍。

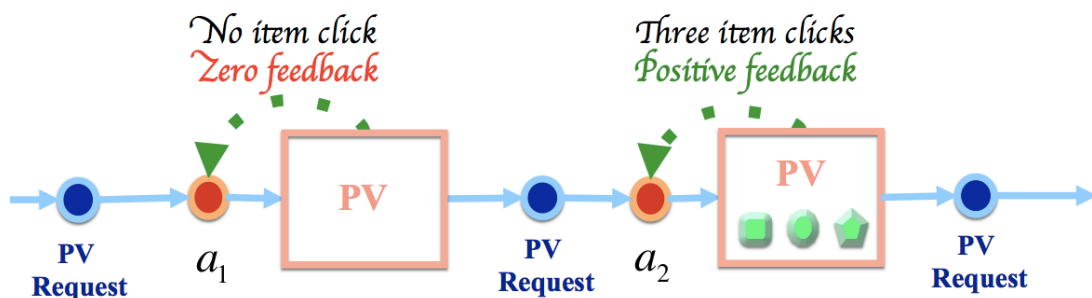


图 6.4: 搜索的序列决策模型

## 6.6.2 问题建模

### 6.6.2.1 强化学习简介

马尔可夫决策过程 (Markov Decision Process, MDP) 是强化学习的最基本理论模型。一般地, MDP 可以由一个四元组  $\langle S, A, R, T \rangle$  表示: (1)  $S$  为状态空间 (State Space), 包含了 Agent 可能感知到的所有环境状态; (2)  $A$  为动作空间 (Action Space), 包含了 Agent 在每个状态上可以采取的所有动作; (3)  $R: S \times A \times S \rightarrow \mathbb{R}$  为奖赏函数 (Reward Function),  $R(s, a, s')$  表示在状态  $s$  上执行动作  $a$ , 并转移到状态  $s'$  时, Agent 从环境获得的奖赏值; (4)  $T: S \times A \times S \rightarrow [0, 1]$  为环境的状态转移函数 (State Transition Function),  $T(s, a, s')$  表示在状态  $s$  上执行动作  $a$ , 并转移到状态  $s'$  的概率。

在 MDP 中, Agent 和环境之间的交互过程可如图 2 所示: Agent 感知当前环境状态  $s_t$ , 从动作空间  $A$  中选择动作  $a_t$  执行; 环境接收 Agent 所选择的动作之后, 给以 Agent 相应的奖赏信号反馈  $r_{t+1}$ , 并转移到新的环境状态  $s_{t+1}$ , 等待 Agent 做出新的决策。在与环境的交互过程中, Agent 的目标是找到一个最优策略  $\pi^*$ , 使得它在任意状态  $s$  和任意时间步骤  $t$  下, 都能够获得最大的长期累积奖赏, 即



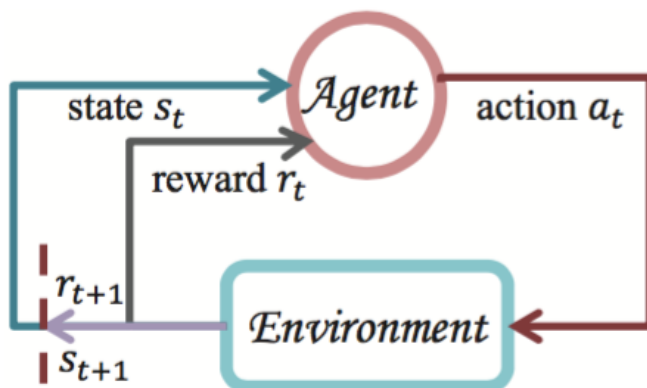


图 6.5: 强化学习 agent 和环境交互

$$\pi^* = \operatorname{argmax}_{\pi} \mathbb{E}_{\pi} \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k} \mid s_t = s \right\}, \forall s \in S, \forall t \geq 0. \quad (6.1)$$

在这里,  $\pi : S \times A \rightarrow [0, 1]$  表示 **Agent** 的某个策略 (即状态到动作的概率分布),  $\mathbb{E}_{\pi}$  表示策略  $\pi$  下的期望值,  $\gamma \in [0, 1)$  为折扣率 (Discount Rate),  $k$  为未来时间步骤,  $r_{t+k}$  表示 **Agent** 在时间步骤  $(t + k)$  上获得的即时奖赏。

强化学习主要通过寻找最优状态值函数 (Optimal State Value Function)

$$V^*(s) = \max_{\pi} \mathbb{E}_{\pi} \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k} \mid s_t = s \right\}, \forall s \in S, \forall t \geq 0 \quad (6.2)$$

或最优状态动作值函数 (Optimal State-Action Value Function)

$$Q^*(s, a) = \max_{\pi} \mathbb{E}_{\pi} \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k} \mid s_t = s, a_t = a \right\}, \forall s \in S, \forall a \in A, \forall t \geq 0 \quad (6.3)$$

来学习最优策略  $\pi^*$ 。经典的强化学习算法 (如: SARSA, Q-learning 等) 将值函数用状态空间到动作空间的一张表来进行表达, 并通过广义策略迭代 (Generalized Policy Iteration) 方法对最优值函数进行逼近, 从而得到最优策略  $\pi^*$ 。然而, 在大规模状态/动作空间问题 (包括连续状态/动作空间问题) 中, 值表形式的值函

数所需要的存储空间远远超过了现代计算机的硬件条件，使得这些经典的算法不再适用。这也即强化学习中的著名的“维度灾难”问题。

值函数估计 (Value Function Approximation) 是解决维度灾难问题的主要手段之一，其主要思想是将状态值函数或动作值函数进行参数化，将值函数空间转化为参数空间，达到泛化 (Generalization) 的目的。以状态值函数为例，在参数向量  $\theta$  下，任意状态  $s$  的值  $V(s)$  可以表达为

$$V_{\theta}(s) = f_{\theta}(\phi(s)). \quad (6.4)$$

其中， $\phi(s)$  为状态  $s$  的特征向量， $f$  为定义在状态特征空间上的某个函数，其具体形式取决于算法本身。因此，对值函数  $V(s)$  的学习也就转化为了对参数向量  $\theta$  的学习。基于对函数  $f$  采用的不同形式，强化学习领域也发展出了不同的子分支，这其中包括：线性函数估计方法，回归树方法，神经网络方法，基于核的强化学习方法等。值得一提的是，深度强化学习 (Deep Reinforcement Learning, DRL) 本质上属于采用神经网络作为值函数估计器的一类方法，其主要优势在于它能够利用深度神经网络对状态特征进行自动抽取，避免了人工定义状态特征带来的不准确性，使得 Agent 能够在更原始的状态上进行学习。

### 6.6.2.2 状态定义

在我们的方案中，用户被视为响应 Agent 动作的环境，Agent 需要感知环境状态进行决策。因此，如何定义环境状态使其能够准确反映出用户对商品的偏好是首要问题。假设用户在搜索的过程中倾向于点击他感兴趣的物品，并且较少点击他不感兴趣的物品。基于这个假设，我们将用户的历史点击行为作为抽取状态特征的数据来源。具体地，在每一个 PV 请求发生时，我们把用户在最近一段时间内点击的物品的特征（包括：价格、转化率、销量等）作为当前 Agent 感知到的状态，令  $s$  代表状态，则有

$$s = (price_1, cvr_1, sale_1, \dots, price_n, cvr_n, sale_n). \quad (6.5)$$

其中， $n$  表示历史点击商品的个数，为可变参数， $price_i$ 、 $cvr_i$ 、 $sale_i$  分别代表商品  $i$  ( $0 \leq i \leq n$ ) 的价格、转化率和销量。另外，为了区别不同群体的用户，

我们还将用户的长期特征加入到了状态的定义中，最终的状态定义为

$$s = (price_1, cvr_1, sale_1, \dots, price_n, cvr_n, sale_n, power, item, shop). \quad (6.6)$$

其中， $power$ 、 $item$  和  $shop$  分别代表用户的购买力、偏好宝贝以及偏好店铺特征。在具体算法实现时，由于状态特征不同维度的尺度不一样，我们会将所有维度的特征值归一化到  $[0, 1]$  区间内，再进行后续处理。

### 6.6.2.3 奖赏函数设定

当状态空间  $S$  和动作空间  $A$  确定好之后（动作空间即 Agent 能够选择排序策略的空间），状态转移函数  $T$  也随即确定，但奖赏函数  $R$  仍然是个未知数。奖赏函数  $R$  定义的是状态与动作之间的数值关系，而我们要解决的问题并非是一个天然存在的 MDP，这样的数值关系并不存在。因此，另一个重要的步骤是把我们要达到的目标（如：提高点击率、提高 GMV 等）转化为具体的奖赏函数  $R$ ，在学习过程中引导 Agent 完成我们的目标。

幸运的是，这样的转化在我们的场景中并不复杂。如前所述，Agent 给出商品排序，用户根据排序的结果进行的浏览、商品点击或购买等行为都可以看成对 Agent 的排序策略的直接反馈。我们采取的奖赏函数定义规则如下：（1）在一个 PV 中如果仅发生商品点击，则相应的奖赏值为用户点击的商品的数量；（2）在一个 PV 中如果发生商品购买，则相应奖赏值为被购买商品的价格；（3）其他情况下，奖赏值为 0。从直观上来理解，第一条规则表达的是提高 CTR 这一目标，而第二条规则表达的则是提高 GMV。在第四章中，我们将利用奖赏塑形（Reward Shaping）方法对奖赏函数的表达进行丰富，提高不同排序策略在反馈信号上的区分度。

## 6.6.3 算法设计

### 6.6.3.1 策略函数

在搜索场景中，排序策略实际上是一组权重向量，我们用  $\mu = (\mu_1, \mu_2, \dots, \mu_m)$  来表示。每个商品最终的排序次序是由其特征分数和排序权重向量  $\mu$  的内积所

决定的。一个排序权重向量是 Agent 的一个动作，那么排序权重向量的欧式空间就是 Agent 的动作空间。根据 2.1 节对状态的定义可知，我们的状态空间也是连续的数值空间。因此，我们面临的问题是在两个连续的数值空间中学习出最优的映射关系。

策略逼近 (Policy Approximation) 方法是解决连续状态/动作空间问题的有效方法之一。其主要思想和值函数估计方法类似，即用参数化的函数对策略进行表达，通过优化参数来完成策略的学习。通常，这种参数化的策略函数被称为 Actor。我们采用确定性策略梯度算法 (Deterministic Policy Gradient, DPG) 算法来进行排序的实时调控优化。在该算法中，Actor 的输出是一个确定性的策略 (即某个动作)，而非一个随机策略 (即动作的概率分布)。对于连续动作空间问题，确定性策略函数反而让策略改进 (Policy Improvement) 变得更加方便了，因为贪心求最优动作可以直接由函数输出。

我们采用的 Actor 以状态的特征为输入，以最终生效的排序权重分为输出。假设我们一共调控  $m$  ( $m \geq 0$ ) 个维度的排序权重，对于任意状态  $s \in S$ ，Actor 对应的输出为

$$\mu_{\theta}(s) = (\mu_{\theta}^1(s), \mu_{\theta}^2(s), \dots, \mu_{\theta}^m(s)). \quad (6.7)$$

其中， $\theta = (\theta_1, \theta_2, \dots, \theta_m)$  为 actor 的参数向量，对于任意  $i(1 \leq i \leq m)$ ， $\mu_{\theta}^i(s)$  为第  $i$  维的排序权重分，具体地有

$$\mu_{\theta}^i = \frac{C_i \exp(\theta_i^{\top} \phi(s))}{\sum_{j=1}^m \exp(\theta_j^{\top} \phi(s))}. \quad (6.8)$$

在这里， $\phi(s)$  为状态  $s$  的特征向量， $\theta_1, \theta_2, \dots, \theta_m$  均为长度与  $\phi(s)$  相等的向量， $C_i$  为第  $i$  维排序权重分的常数，用来对其量级进行控制 (不同维度的排序权重分会有不同的量级)。

### 6.6.3.2 策略梯度

回顾一下，强化学习的目标是最大化任意状态  $s$  上的长期累积奖赏 (参考 2.1 节中对  $V^*$  和  $Q^*$  的定义)。实际上，我们可以用一个更一般地形式来表达这

一目标，即

$$\begin{aligned}
 J(\mu_\theta) &= \int_S \int_S \sum_{t=1}^{\infty} \gamma^{t-1} p_0(s') T(s', \mu_\theta(s'), s) R(s, \mu_\theta(s)) \, ds' \, ds \\
 &= \int_S \rho^\mu(s) R(s, \mu_\theta) \, ds \\
 &= \mathbb{E}_{s \sim \rho^\mu} [R(s, \mu_\theta(s))].
 \end{aligned} \tag{6.9}$$

其中， $\rho^\mu(s) = \int_S \sum_{t=1}^{\infty} \gamma^{t-1} p_0(s') T(s', \mu_\theta(s'), s) \, ds'$  表示状态  $s$  在一直持续的学习过程中被访问的概率， $p_0$  为初始时刻的状态分布， $T$  为环境的状态转移函数。不难推测， $J(\mu_\theta)$  实际上表达的是在确定性策略  $\mu_\theta$  的作用下，Agent 在所有状态上所能够获得的长期累积奖赏期望之和。通俗地讲，也就是 Agent 在学习过程中得到的所有奖赏值。

显然，为了最大化  $J(\mu_\theta)$ ，我们需要求得  $J(\mu_\theta)$  关于参数  $\theta$  的梯度，让  $\theta$  往梯度方向进行更新。根据策略梯度定理（Policy Gradient Theorem）， $J(\mu_\theta)$  关于  $\theta$  的梯度为

$$\begin{aligned}
 \nabla_\theta J(\mu_\theta) &= \int_S \rho^\mu(s) \nabla_\theta \mu_\theta(s) \nabla_a Q^\mu(s, a) |_{a = \mu_\theta(s)} \, ds \\
 &= \mathbb{E}_{s \sim \rho^\mu} [\nabla_\theta \mu_\theta(s) \nabla_a Q^\mu(s, a) |_{a = \mu_\theta(s)}].
 \end{aligned} \tag{6.10}$$

其中， $Q^\mu(s, a)$  为策略  $\mu_\theta$  下状态动作对（State-Action Pair） $(s, a)$  对应的长期累积奖赏。因此，参数  $\theta$  的更新公式可以写为

$$\theta_{t+1} \leftarrow \theta_t + \alpha_\theta \nabla_\theta \mu_\theta(s) \nabla_a Q^\mu(s, a) |_{a = \mu_\theta(s)}. \tag{6.11}$$

在这个公式中， $\alpha_\theta$  为学习率， $\nabla_\theta \mu_\theta(s)$  为一个 Jacobian Matrix，能够很容易地算出来，但麻烦的是  $Q^\mu(s, a)$  及其梯度  $\nabla_a Q^\mu(s, a)$  的计算。因为  $s$  和  $a$  都是连续的数值，我们无法精确获取  $Q^\mu(s, a)$  的值，只能通过值函数估计方法进行近似计算。我们采用线性函数估计方法（Linear Function Approximation, LFA），将  $Q$  函数用参数向量  $w$  进行表达：

$$Q^\mu(s, a) \approx Q^w(s, a) = \phi(s, a)^\top w. \tag{6.12}$$

在这里,  $\phi(s, a)$  为状态动作对  $(s, a)$  的特征向量。采用线性值函数估计的好处不仅在于它的计算量小, 更重要的是在它能让我们找到合适的  $\phi(s, a)$  的表达, 使得  $\nabla_a Q^w(s, a)$  可以作为  $\nabla_a Q^\mu(s, a)$  的无偏估计。一个合适的选择是令  $\phi(s, a) = a^\top \nabla_\theta \mu_\theta(s)$ , 则可以得到

$$\nabla_a Q^\mu(s, a) \approx \nabla_a Q^w(s, a) = \nabla_a (a^\top \nabla_\theta \mu_\theta(s))^\top w = \nabla_\theta \mu_\theta(s)^\top w. \quad (6.13)$$

因此, 策略函数的参数向量  $\theta$  的更新公式可以写为

$$\theta_{t+1} \leftarrow \theta_t + \alpha_\theta \nabla_\theta \mu_\theta(s) (\nabla_\theta \mu_\theta(s)^\top w). \quad (6.14)$$

### 6.6.3.3 值函数的学习

在更新策略函数  $\mu_\theta$  的参数向量  $\theta$  的同时, 值函数  $Q^w$  的参数向量  $w$  也需要进行更新。最简单地,  $w$  的更新可以参照 Q-learning 算法 [4, 5] 的线性函数估计版本进行, 对于样本  $(s_t, a_t, r_t, s_{t+1})$ , 有:

$$\begin{aligned} \delta_{t+1} &= r_t + \gamma Q^w(s_{t+1}, \mu_\theta(s_{t+1})) - Q^w(s_t, a_t) \\ &= r_t + w_t^\top (\gamma \phi(s_{t+1}, \mu_\theta(s_{t+1})) - \phi(s_t, a_t)) \\ w_{t+1} &= w_t + \alpha_w \delta_{t+1} \phi(s_t, a_t) \\ &= w_t + \alpha_w \delta_{t+1} (a_t^\top \nabla_\theta \mu_\theta(s_t)). \end{aligned} \quad (6.15)$$

其中,  $s_t, a_t, r_t$  和  $s_{t+1}$  为 Agent 在  $t$  时刻感知的状态、所做的动作、从环境获得的奖赏反馈和在  $(t+1)$  时刻感知的状态,  $\delta_{t+1}$  被称作差分误差 (Temporal-Difference Error),  $\alpha_w$  为  $w$  的学习率。

需注意的是, Q-learning 的线性函数估计版本并不能保证一定收敛。并且, 在大规模动作空间问题中, 线性形式的  $Q$  函数较难在整个值函数空间范围中精确地估计每一个状态动作对的值。一个优化的办法是引入优势函数 (Advantage Function), 将  $Q$  函数用状态值函数  $V(s)$  和优势函数  $A(s, a)$  的和进行表达。我们用  $V(s)$  从全局角度估计状态  $s$  的值, 用  $A(s, a)$  从局部角度估计动作  $a$  在状态  $s$  中的相对于其他动作的优势。具体地, 我们有

$$Q(s, a) = A^w(s, a) + V^v(s) = (a - \mu_\theta(s))^\top \nabla_\theta \mu_\theta(s)^\top w + \phi(s)^\top v. \quad (6.16)$$

在这里， $w$  和  $v$  分别为  $A$  和  $V$  的参数向量。最后，我们将所有参数  $\theta$ 、 $w$  和  $v$  的更新方式总结如下：

$$\begin{aligned}
 \delta_{t+1} &= r_t + \gamma Q(s_{t+1}, \mu_\theta(s_{t+1})) - Q(s_t, a_t) \\
 &= r_t + \gamma \phi(s_{t+1})^\top v_t - ((a_t - \mu_\theta(s_t))^\top \nabla_\theta \mu_\theta(s_t)^\top w_t + \phi(s_t)^\top v_t) \\
 \theta_{t+1} &= \theta_t + \alpha_\theta \nabla_\theta \mu_\theta(s_t) (\nabla_\theta \mu_\theta(s_t)^\top w_t) \\
 w_{t+1} &= w_t + \alpha_w \delta_{t+1} \phi(s_t, a_t) = w_t + \alpha_w \delta_{t+1} (a_t^\top \nabla_\theta \mu_\theta(s_t)) \\
 v_{t+1} &= v_t + \alpha_v \delta_{t+1} \phi(s_t)
 \end{aligned} \tag{6.17}$$

## 6.6.4 奖赏塑形

第二章定义的奖赏函数是一个非常简单化的版本，我们在初步的实验中构造了一个连续状态空间/离散动作空间问题对其进行了验证。具体地，我们采用了 2.2 节定义的状态表示方法，同时人工选取 15 组固定的排序策略，通过线性值函数估计控制算法 GreedyGQ 来学习相应的状态动作值函数  $Q(s, a)$ 。从实验结果中，我们发现学习算法虽然最终能够稳定地区分开不同的动作，但它们之间的差别并不大（最优动作和最次动作的  $Q$  值相差的比例大约 10%）。一方面，这并不会对算法收敛到最优产生影响；但另一个方面，学习算法收敛的快慢却会大受影响，而这是在实际应用中我们必须要考虑的问题。

在淘宝主搜这种大规模应用的场景中，我们较难在短时间内观察到不同的排序策略在点击和成交这样的宏观指标上的差别。因此，我们有必要在奖赏函数中引入更多的信息，增大不同动作的区分度。以商品点击为例，考虑不同的用户  $A$  和  $B$  在类似的状态中发生的商品点击行为。若  $A$  点击商品的价格较高， $B$  点击商品的价格较低，那么就算  $A$  和  $B$  点击的商品数量相同，我们也可以认为的对  $A$  和  $B$  采用的排序策略带来的影响是不同的。同样的，对于商品的成交，价格相同而销量不同的商品成交也具有差别。因此，在原有的基础上，我们将商品的一些属性特征加入到奖赏函数的定义中，通过奖赏塑形（Reward Shaping）的方法丰富其包含的信息量。

奖赏塑形的思想是在原有的奖赏函数中引入一些先验的知识，加速强化学习算法的收敛。简单地，我们可以将“在状态  $s$  上选择动作  $a$ ，并转移到状态  $s'$ ”



的奖赏值定义为

$$R(s, a, s') = R_0(s, a, s') + \Phi(s). \quad (6.18)$$

其中,  $R_0(s, a, s')$  为原始定义的奖赏函数,  $\Phi(s)$  为包含先验知识的函数, 也被称为势函数 (Potential Function)。我们可以把势函数  $\Phi(s)$  理解学习过程中的子目标 (Local Objective)。例如, 在用强化学习求解迷宫问题中, 可以定义  $\Phi(s)$  为状态  $s$  所在位置与出口的曼哈顿距离 (或其他距离), 使得 Agent 更快地找到潜在的与出口更近的状态。根据上面的讨论, 我们把每个状态对应 PV 的商品信息纳入 Reward 的定义中, 将势函数  $\Phi(s)$  定义为

$$\Phi(s) = \sum_{i=1}^K \text{ML}(i|\mu_\theta(s)). \quad (6.19)$$

其中,  $K$  为状态  $s$  对应 PV 中商品的个数,  $i$  表示的第  $i$  个商品,  $\mu_\theta(s)$  为 Agent 在状态  $s$  执行的动作,  $\text{ML}(i|\mu_\theta(s))$  表示排序策略为  $\mu_\theta(s)$  时对商品的点击 (或成交) 的极大似然 (Maximum Likelihood) 估计。因此,  $\Phi(s)$  也就表示在状态  $s$  上执行动作  $\mu_\theta(s)$  时, PV 中所有商品能够被点击 (或购买) 的极大似然概率之和。

下面我们给出的  $\text{ML}(i|\mu_\theta(s))$  具体形式。令商品  $i$  的特征向量 (即价格、销量、人气分、实时分等特征) 为  $x_i = (x_i^1, x_i^2, \dots, x_i^m)$ , 则  $x_i^\top \mu_\theta(s)$  即为商品  $i$  在状态  $s$  下的最终排序分数。又令  $y_i \in \{0, 1\}$  为商品  $i$  实际被点击 (或成交) 的 label, 并假设意商品  $i$  的实际点击 (或成交) 的概率  $p_i$  与其排序分数  $x_i^\top \mu_\theta(s)$  满足  $\ln \frac{p_i}{1-p_i} = x_i^\top \mu_\theta(s)$ , 则商品  $i$  的似然概率为

$$\text{ML} = p_i^{y_i} (1 - p_i)^{1-y_i} = \left( \frac{1}{1 + \exp(-x_i^\top \mu_\theta(s))} \right)^{y_i} \left( \frac{1}{1 + \exp(x_i^\top \mu_\theta(s))} \right)^{1-y_i}. \quad (6.20)$$

为简化计算, 我们对 BL 取对数, 得到对数似然概率

$$\text{ML}_{\log}(i|\mu_\theta(s)) = y_i x_i^\top \mu_\theta(s) - \ln(1 + \exp(x_i^\top \mu_\theta(s))). \quad (6.21)$$

将 PV 中所有商品的对数似然概率综合起来, 则有

$$\Phi(s) = \sum_{i=1}^K y_i x_i^\top \mu_\theta(s) - \ln(1 + \exp(x_i^\top \mu_\theta(s))). \quad (6.22)$$



我们最终实现的奖赏塑形方法将点击和成交均纳入考虑中，对于只有点击的 PV 样本，其对应的奖赏势函数为

$$\Phi_{clk}(s) = \sum_{i=1}^K y_i^c x_i^\top \mu_\theta(s) - \ln(1 + \exp(x_i^\top \mu_\theta(s))). \quad (6.23)$$

其中， $y_i^c$  是商品  $i$  被点击与否的 label。而对于有成交发生的 PV 样本，我们将商品价格因素加入到奖赏势函数中，得到

$$\Phi_{pay}(s) = \sum_{i=1}^K y_i^p x_i^\top \mu_\theta(s) - \ln(1 + \exp(x_i^\top \mu_\theta(s))) + \ln \text{Price}_i. \quad (6.24)$$

其中， $y_i^p$  和  $\text{Price}_i$  分别是商品  $i$  被购买与否的 label 和它的价格。从直观上来理解， $\Phi_{clk}(s)$  和  $\Phi_{pay}(s)$  将分别引导 Agent 对点击率和 GMV 的对数似然进行优化。

实际上，我们所采用的奖赏塑形方法来自于 LTR 方法的启发。LTR 方法的有效性在于它能够利用商品维度的信息来进行学习，其最终学习到的排序权重和商品特征有直接相关性。我们通过把商品的特征灌注到奖赏函数中，能让 Agent 的动作在具体商品上产生的影响得到刻画，因此也就能更好地在数值信号上将不同的动作区分开来。另外，与以往的奖赏塑形方法不同的是，我们采用的势函数是随着策略的学习变化的，它让 Reward 和 Action 之间产生了相互作用：Action 的计算将朝着最大化 Reward 的方向进行，而 Action 的生效投放也反过来影响了 Reward 的产生。因此，学习算法实际上是在非独立同分布的数据上进行训练的，我们将在最后一章对该问题进行探讨。

## 6.6.5 实验效果

基于强化学习的实时排序调控方案的实现主要在 QP 端和 Porsche 平台上进行，两者通过 TT 日志和 iGraph 数据进行连接，形成一个闭环。其中，QP 部分主要负责从 iGraph 表中读取学习算法学习到的参数模型，计算排序策略，并将状态、动作等信息记录到 TT 日志中。Porsche 端负责对日志进行解析处理，提取学习算法需要的训练数据，实时更新模型参数并写回到 iGraph。

在双十一期间，我们在 21 和 22 号桶对强化学习方案进行了测试。下图展示了我们的算法在学习的过程中的误差变化情况，截取的时间范围为 11 月 10 日 18:00 到 11 月 11 日 8:00。衡量学习误差的指标为 Norm of the Expected TD Update (NEU)，是差分误差 (TD Error) 与状态动作特征向量乘积的期望值，图中的 RNEU 表示 NEU 的平方根。从理论上讲，RNEU 越小表示算法学习到的策略越接近最优。

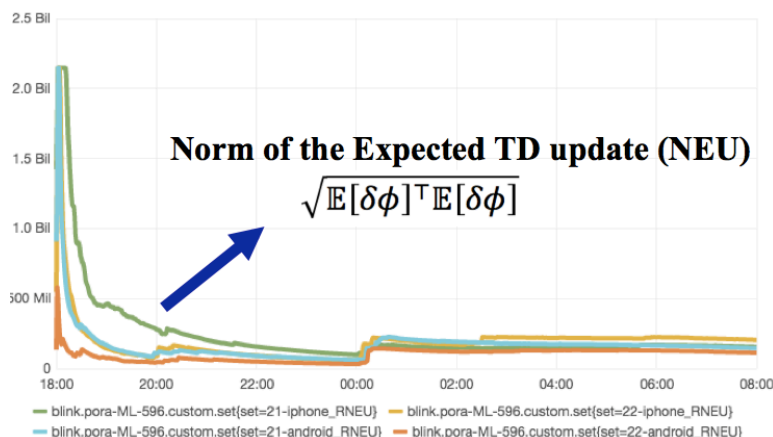


图 6.6: Norm of the Expected TD Update

可以看到，从 11 月 10 日 18:00 启动开始，每个桶上的 RNEU 开始逐渐下降。到当天 20:00 之后，下降趋势变得比较缓和，说明学习算法在逐步往最优策略进行逼近。但过了 11 月 11 日 0 点之后，每个桶对应的 RNEU 指标都出现了陡然上升的情况，这是因为 0 点前后用户的行为发生了急剧变化，导致线上数据分布在 0 点以后与 0 点之前产生较大差别。相应地，学习算法获取到新的 reward 信号之后，也会做出适应性地调整。

接下来，我们再对双十一当天排序权重分的变化情况进行考查。我们一共选取了 22 个精排权重分来进行实时调控，包括 TaobaoRenqiScoreWireLess、WirelessCtrPredictBts、i2IFeatureExtractor1、ItemEnsembleCVR、Double11 等。下面两幅图分别展示了 21 号桶的 iphone 和 android 中，每个维度的排序权重分在一天内的变化。

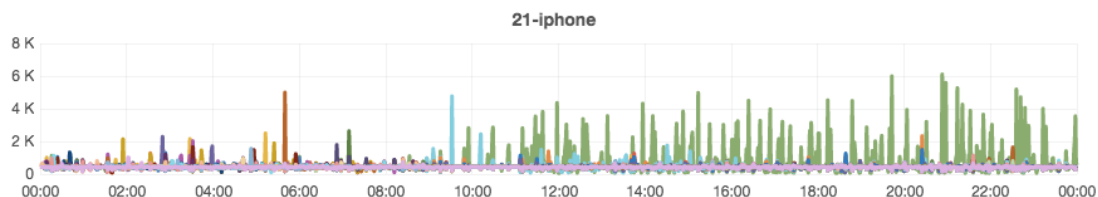


图 6.7: 每个维度的排序权重分在一天内的变化 (iphone)

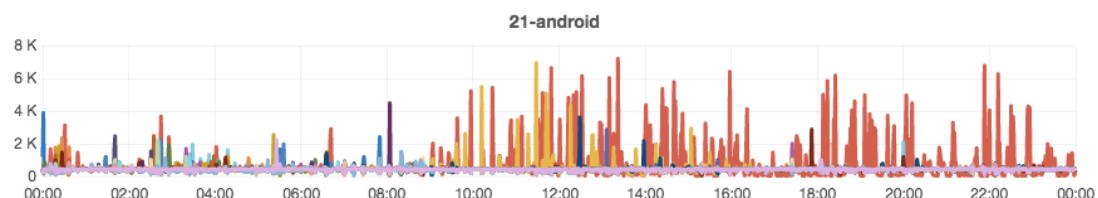


图 6.8: 每个维度的排序权重分在一天内的变化 (android)

需注意的是，图中每条曲线上的数值并非对应排序权重分的生效分数，而是该排序权重分在所有排序权重分中的占比（即 3.2 节中的  $\mu_{\theta}^i$  除以  $C_i$ ）。从 0 点到早上 10:00 这一时间段内，无论是在 android 端还是 iphone 端，都没有出现某个维度的排序权重分占绝对主导地位，22 个排序权重分似乎在均等地发挥效力。在 11 号凌晨和上午，全网大部分的成交其实不在主搜端。在这段时间内，用户产生的数据相对没有这么丰富，可能还不足以将重要的排序权重分凸显出来。而到了 10:00 以后，我们可以发现某一个维度的排序权重分逐渐开始占据主导，并且其主导过程一直持续到了当天结束。在 iphone 端占据主导的是 Double11（绿色曲线），而 android 端的则是 ItemEnsembleCVR（红色曲线）。这其实也从侧面说明了 iphone 端和 android 端的用户行为存在较大差别。

在最终的投放效果上，21 和 22 号桶在强化学习和其他模块的共同作用下，GMV 相对于基准桶分别提升 14.38% 和 14.4%（整体数据）。同时，21 和 22 号桶在 CTR 方面的提升高于其它绝大部分非强化学习桶（4 号和 7 号桶除外），证明我们所采用的奖赏塑形方法确实有效地将优化 CTR 的目标融入了奖赏函数中。

### 6.6.6 总结与展望

总的来说，我们将强化学习应用到淘宝的搜索场景中只是一次初步尝试，有很多方面都需要进一步探索，现将我们在未来需要改进的地方以及可能的探索方向归纳如下：

(1) 状态的表示：我们将用户最近点击的商品特征和用户长期行为特征作为状态，其实是基于这样的一个假设，即用户点击过的商品能够较为精确地反映用户的内心活动和对商品的偏好。但实际上，用户对商品的点击通常具有盲目性，无论什么商品可能都想要看一看。也就是说，我们凭借经验所设定的状态并非那么准确。深度强化学习对状态特征的自动抽取能力是它在 Atari Game 和围棋上取得成功的重要原因之一。因此，在短期内可以考虑利用深度强化学习对现有方案进行扩展。同时，借助深度神经网络对状态特征的自动抽取，我们也可以发现用户的哪些行为对于搜索引擎的决策是比较重要的。

(2) 奖赏函数的设定：和状态的定义一样，我们在第二章设定的奖赏函数也来自于人工经验。奖赏塑形 (Reward Shaping) 虽然是优化奖赏函数的方法，但其本质上也是启发式函数，其更多的作用在于对学习算法的加速。逆强化学习 (Inverse Reinforcement Learning, IRL) 是避免人工设定的奖赏函数的有效途径之一，也是强化学习研究领域的重要分支。IRL 的主要思想是根据已知的专家策略或行为轨迹，通过监督学习的方法逆推出问题模型的奖赏函数。Agent 在这样的奖赏函数上进行学习，就能还原出专家策略。对于我们的问题，IRL 的现有方法不能完全适用，因为我们的搜索任务并不存在一个可供模仿的专家策略。我们需要更深入思考如何在奖赏函数与我们的目标（提升 CTR，提升成交笔数）之间建立紧密的关系。

(3) 多智能体强化学习 (Multi-Agent Reinforcement Learning, MARL)：我们将搜索引擎看作 Agent，把用户看成响应 Agent 动作的环境，属于典型的单智能体强化学习 (Single-Agent RL) 模式。在单智能体强化学习的理论模型（即 MDP）中，环境动态 (Environmental Dynamics，也即奖赏函数和状态转移函数) 是不会发生变化的；而在我们的问题中，用户的响应行为却是非静态的，同时也带有随机性。因此，单智能体强化学习的模式未必是我们的最佳方案。要知道，

用户其实也是在一定程度理性控制下的，能够进行自主决策甚至具有学习能力的 Agent。从这样的视角来看，或许更好的方式是将用户建模为另外一个 Agent，对这个 Agent 的行为进行显式地刻画，并通过多智能体强化学习 [21] 方法来达到搜索引擎 Agent 和用户 Agent 之间的协同（Coordination）。

（4）第四章的末尾提到了奖赏函数与 Agent 的动作的相互作用带来的非独立同分布数据问题，我们在这里再进行一些讨论。在 MDP 模型中，奖赏函数  $R(s, a, s')$ （有时又写成  $R(s, a)$ ）是固定的，不会随着 Agent 策略的变化而变化。然而，在我们提出的奖赏塑形方法中，势函数  $\Phi_{clk}(s)$  和  $\Phi_{pay}(s)$  中包含了策略参数  $\theta$ ，使得 Agent 从环境获得的奖赏信号在不同的  $\theta$  下有所不同。这也意味着我们的 Agent 实际上是处于一个具有动态奖赏函数的环境中，这种动态变化不是来自于外部环境，而是源于 Agent 的策略改变。这有点类似于人的行为与世界的相互作用。这样一来，我们可以将 3.2 节的  $J(\mu_\theta)$  重写为

$$\begin{aligned}\bar{J}(\mu_\theta) &= \int_S \int_S \sum_{t=1}^{\infty} \gamma^{t-1} p_0(s') T(s', \mu_\theta(s'), s) R_\theta(s, \mu_\theta(s)) \, ds' \, ds \\ &= \int_S \rho^\mu(s) R_\theta(s, \mu_\theta) \, ds.\end{aligned}\tag{6.25}$$

其中， $R_\theta$  为 Agent 的策略参数  $\theta$  的函数。虽然  $J(\mu_\theta)$  与  $\bar{J}(\mu_\theta)$  之间只有一个符号之差，但这微小的变化也许会导致现有的强化学习算法无法适用，我们在未来的工作中将从理论上来深入研究这个问题的解决方法。

## 6.7 强化学习为何有用？——延迟奖赏在搜索排序场景中的作用分析

### 6.7.1 背景

我们用强化学习（Reinforcement Learning, RL）在搜索场景中进行了许多的尝试，例如：对商品排序策略进行动态调节、控制个性化展示比例、控制价格 T 变换等。在此之后，我们也做了若干的改进，例如和监督网络的多任务学习：

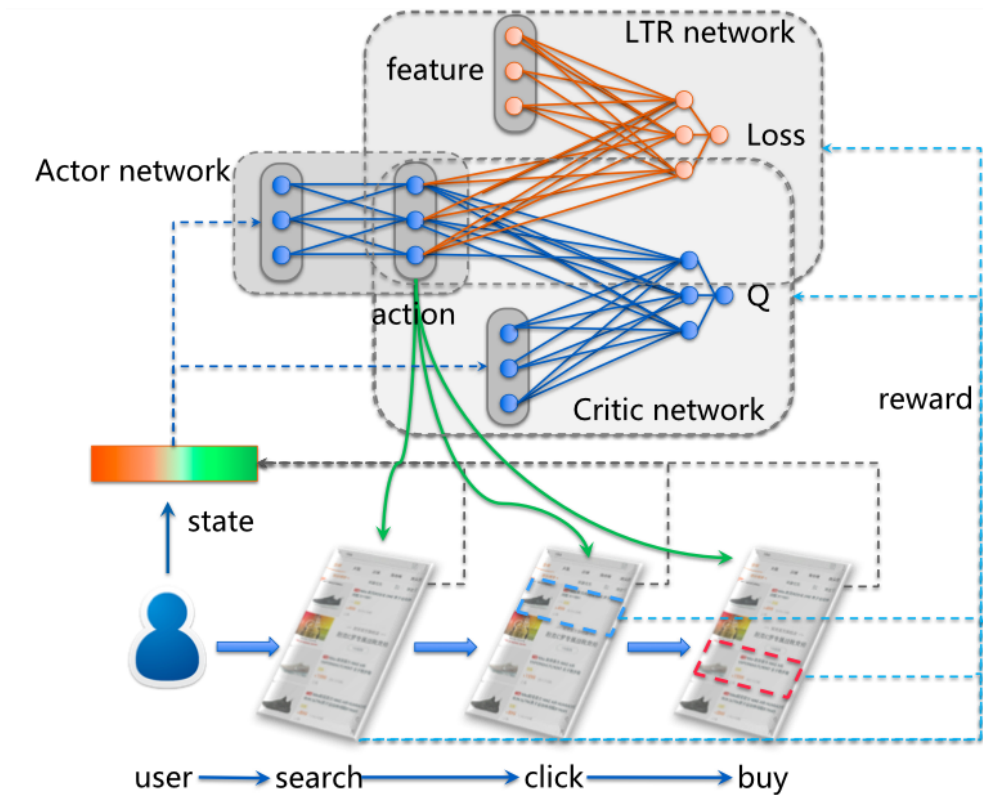


图 6.9: 监督学习和强化学习的多任务学习网络

虽然从顺序决策的角度来讲，强化学习在这些场景中的应用是合理的，但我们并没有回答一些根本性的问题，比如：在搜索场景中采用强化学习和采用多臂老虎机有什么本质区别？从整体上优化累积收益和分别独立优化每个决策步骤的即时收益有什么差别？每当有同行问到这些问题时，我们总是无法给出让人信服的回答。因为我们还没思考清楚一个重要的问题，即：在搜索场景的顺序决策过程中，任意决策点的决策与后续所能得到的结果之间的关联性有多大？从强化学习的角度讲，也就是后续结果要以多大的比例进行回传，以视为对先前决策的延迟激励。也就是说我们要搞清楚延迟反馈在搜索场景中的作用。本文将继续以搜索场景下调节商品排序策略为例，对这个问题展开探讨。本文余下部分的将组织如下：第二节对搜索排序问题的建模进行回顾，第三节将



介绍最近的线上数据分析结果，第四节将对搜索排序问题进行形式化定义，第五节和第六节分别进行理论分析和实验分析并得出结论。

## 6.7.2 搜索排序问题回顾

在淘宝中，对商品进行搜索排序以及重排序涉及搜索引擎与用户之间的不断交互。图 1 展示了这样的交互过程：(1) 用户进入搜索引擎，输入 query；(2) 搜索引擎根据用户输入的 query 和用户的特征，从若干个可能的排序动作中 ( $a_1, a_2, a_3, \dots$ ) 选择其中一个给对应 query 下的商品进行排序，选择 top  $K$  个商品 ( $K$  一般等于 10)；(3) 用户在看到展示页面的商品之后，会在页面中进行一些操作，比如：点击、加购感兴趣的商品；(4) 当用户进行翻页时，搜索引擎会再次选择一个排序动作，对未展示的商品重新进行排序，并进行商品展示；(5) 随着用户不断地翻页，这样的交互过程会一直进行下去，直到用户购买某个商品或者离开搜索引擎。

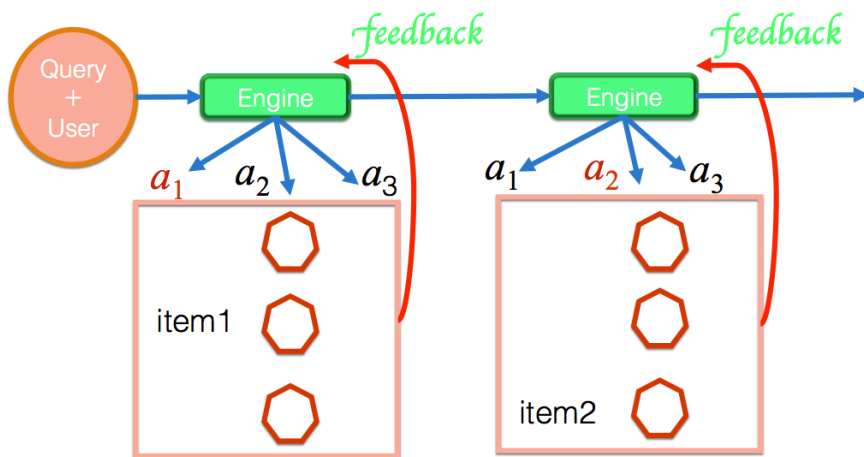


图 6.10: 搜索引擎与用户交互示意图

如果把搜索引擎看作智能体 (Agent)、把用户看做环境 (Environment)，那么图 1 展示的交互过程对于搜索引擎 Agent 来讲是一个典型的顺序决策问题。若从强化学习的视角来看，图 1 所展示的过程就是一次 Episode，可以重新用图 2

进行描述。在图 2 中，蓝色的节点表示一次 PV 请求，也对应 Agent 进行状态感知的时刻，红色的节点表示 Agent 的动作，绿色箭头表示对 Agent 动作的即时奖赏激励。

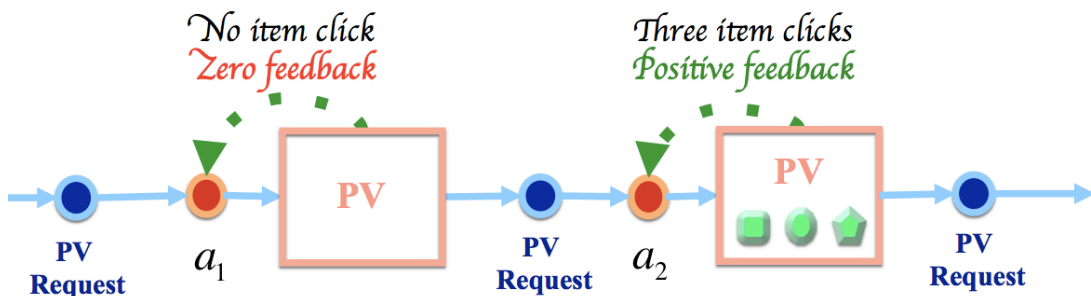


图 6.11: 搜索引擎 Agent 决策过程示意图

需要注意的是，由于搜索引擎每一次的决策都是在 PV 请求时发生的，所以决策过程中的状态与展示的商品页是一一对应的。更严格地来讲，每一个决策点的状态应该是“这个决策点之前所有商品页面包含的信息总和”，包括这些页面展示的商品信息，以及用户在这些页面上的实时行为。在目前的系统实现中，由于性能、信息获取条件的限制，现有的状态表示中并没有完全囊括这些信息。但抛开具体的状态表示方法不谈，我们可以认为一个商品页就是一个状态。在下一节中，我们将以 PV 为单位对线上数据进行统计分析，希望能够发现这个搜索排序问题的一些特性。

### 6.7.3 数据统计分析

在强化学习中，有很多算法都是分 Episode 进行训练的。所谓 Episode 是指一次从初始状态（Initial State）到终止状态（Terminal State）之间的经历。我们对线上产生的训练数据按照 Episode 进行了组织，也就是将每个 Episode 对应的所有商品展示页进行串联，形成“PV->PV->...->End”的一个序列，相当于是把 Episode 中的所有 State 进行串联。其中，“End”表示一个 Episode 的终止状态。在我们的场景中，终止状态表示“用户离开搜索引擎”或者“进行了购买”。由



于我们在日志中无法获取“用户离开搜索引擎”这样的事件，所以我们能够完整抽取 Episode 的数据其实都是有成功购买的 PV 序列。令  $n$  表示 Episode 的序列长度，我们分别在 6 号桶和 15 号桶统计了  $n = 1, 2, \dots, 30$  的 Episode 占总体成交 Episode 的比例，结果如图 3 所示。

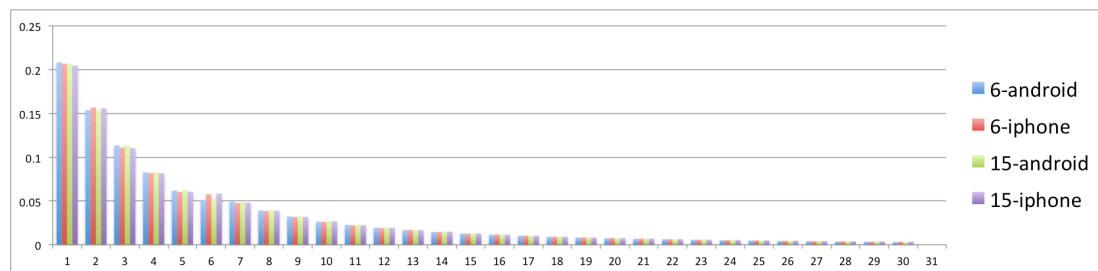


图 6.12: 全类目数据下所有成交 Episode 的长度分布情况

从图 3 的结果中，可以看到 Episode 的长度越大，其对应的占总体的比例越小。这与“越往后的 PV 转化率越低”的经验是相符的。从绝对数值上看，超过 60% 的成交都是在前 6 个 PV 中发生的，而  $n = 1, 2, 3$  的比例更是分别超过了 20%、15% 和 10%。当然，图 4 的结果来自于对全类目数据的统计。为了消除类目间差异给统计结果带来的影响，我们选取了“连衣裙”、“女鞋”和“婴幼儿服饰”这三个成交量较大的类目，分别进行了相同的统计分析，相应的结果展示在图 4-6 中。

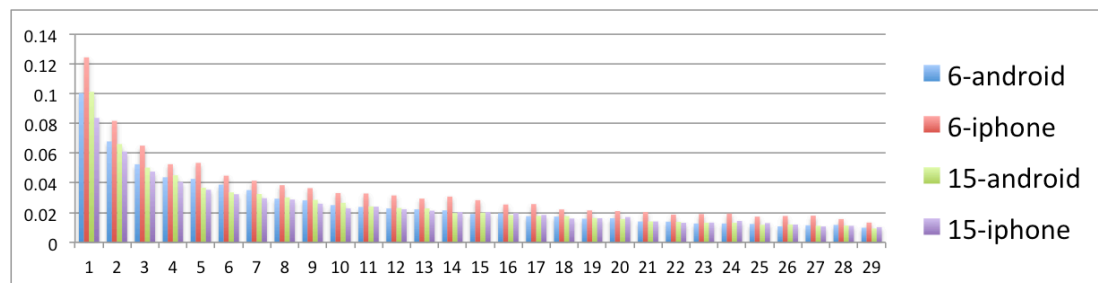


图 6.13: 连衣裙类目下所有成交 Episode 的长度分布情况

虽然分类目统计结果与全类目的结果在绝对数值上有一定差别，但还是呈现出了相同的趋势。如果不考虑具体的数值，我们至少可以得出一个结论：用户

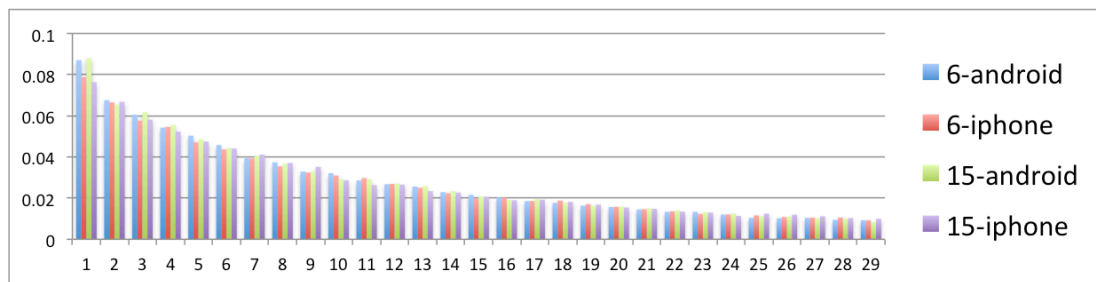


图 6.14: 女鞋类目下所有成交 Episode 的长度分布情况

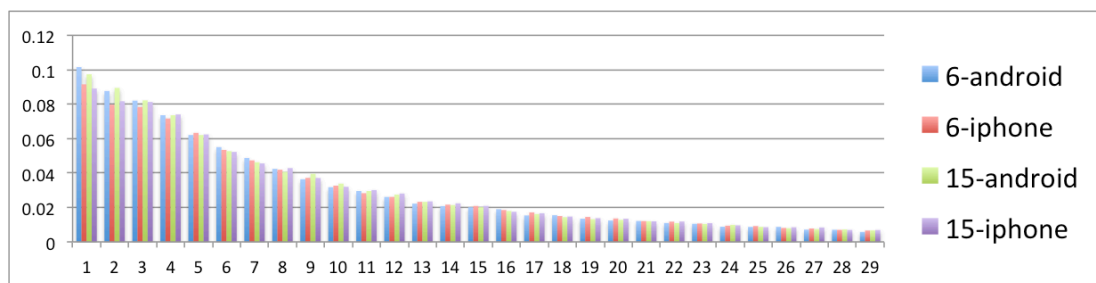


图 6.15: 婴幼儿服饰类目下所有成交 Episode 的长度分布情况

在看过任意数量的商品展示页之后，都有可能发生成交。根据这个结论，我们可以将一次搜索会话过程用图 7 的抽象示意图来描述。如图所示，垂直方向的箭头由上向下表示用户不停翻页的过程。每翻一页，用户选择商品的范围就增加一页，PV 的 History 也对应地发生变化。横向地来看，用户在任意的 PV History 下，都有可能选择购买某个被展示的商品，或者继续往下翻页。当然，如果考虑到用户也有可能离开搜索引擎，我们可以得到图 8 中的更一般的示意图。

在我们的场景中，“成交”和“离开搜索引擎”均被视为一个 Episode 的终止状态。如果把图 8 和马尔可夫决策过程 (MDP) 的状态、状态转移等要素对应起来，就可以发现搜索排序问题的明显特征：任意非终止状态都有一定的概率转移到终止状态。这同一些典型的强化学习应用场景相比有很大不同。比如，在网格世界和迷宫问题中，只有与邻近终点的位置才有非零的概率转移到终止状态。在接下来的内容中，我们将根据搜索排序问题的特点对其进行形式化定义，并在此基础上做相应的理论分析。

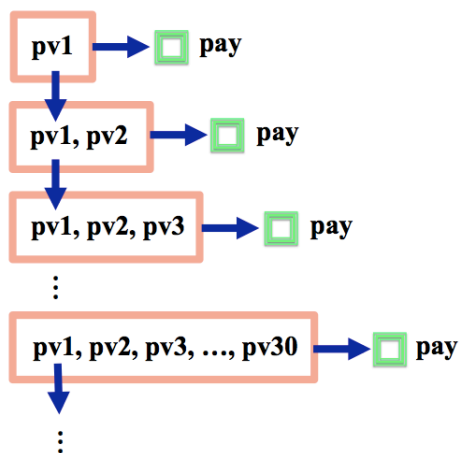


图 6.16: 仅考虑成交和翻页的搜索会话图示

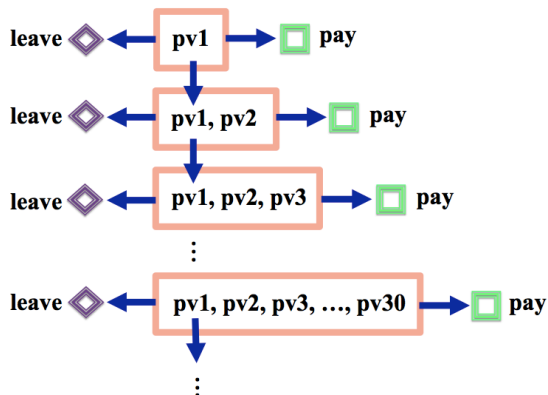


图 6.17: 考虑成交、翻页和用户离开的搜索会话图示

#### 6.7.4 搜索排序问题形式化

本节提出搜索会话马尔科夫决策过程模型 (Search Session Markov Decision Process, SSMDP), 作为对搜索排序问题的形式化定义。我们首先对搜索会话过程中的上下文信息和用户行为进行建模, 形式化定义商品页、商品页历史、成交转化率等概念, 它们是定义状态和状态转移关系的基础。

定义 1. [Top  $K$  List] 给定商品集合  $\mathcal{D}$ , 排序函数  $f$ , 以及一个正整数  $K$

( $1 \leq K \leq |\mathcal{D}|$ ), 关于  $\mathcal{D}$  和  $f$  的 top  $K$  list, 记为  $\mathcal{L}_K(\mathcal{D}, f)$ , 是用函数  $f$  对  $\mathcal{D}$  中商品进行打分以后的前  $K$  个商品的有序列表  $(\mathcal{I}_1, \mathcal{I}_2, \dots, \mathcal{I}_K)$ 。其中,  $\mathcal{I}_k$  是排在第  $k$  位的商品 ( $1 \leq k \leq K$ ), 并且对于任意  $k' \geq k$ , 都有  $f(\mathcal{I}_k) > f(\mathcal{I}_{k'})$ 。

定义 2. [Item Page] 令  $\mathcal{D}$  为关于某个 query 的商品全集,  $K$  ( $K > 0$ ) 为一个页面能够展示的商品数量。对于一个搜索会话的任意时间步  $t$  ( $t \geq 1$ ), 其对应的 item page  $p_t$  是关于的第  $(t-1)$  步的打分函数  $a_{t-1}$  和未展示商品集合  $\mathcal{D}_{t-1}$  的 top  $K$  list  $\mathcal{L}_K(\mathcal{D}_{t-1}, a_{t-1})$ 。对于初始时间步  $t = 0$ , 有  $\mathcal{D}_0 = \mathcal{D}$ 。对于其他任意时间步  $t \geq 1$ , 有  $\mathcal{D}_t = \mathcal{D}_{t-1} \setminus p_t$ 。

定义 3. [Item Page History] 令  $q$  为一个搜索会话的 query。对于初始时间步  $t = 0$ , 对应的初始 item page history 为  $h_0 = q$ 。对于任意其他时间步  $t \geq 1$ , 对应的 item page history 为  $h_t = (h_{t-1}, p_t)$ 。在这里,  $h_{t-1}$  为第  $(t-1)$  步的 item page history,  $p_t$  为第  $t$  步的 item page。

对于任意时间步骤  $t$ , item page history  $h_t$  包含了用户在  $t$  时刻能够观察到的所以上下文信息。由于商品全集  $\mathcal{D}$  是一个有限集合, 不难发现一个搜索会话最多包含  $\lceil \frac{|\mathcal{D}|}{K} \rceil$  个 item page。对于搜索引擎来讲, 它在一个搜索会话中最多决策  $\lceil \frac{|\mathcal{D}|}{K} \rceil$  次。根据我们之前的数据分析, 不同的用户会在不同的时间步上选择购买或者离开。如果我们把所有用户看作一个能够采样出不同用户行为的 environment, 就意味着这个 environment 可能会在任意时间步上以一定的成交转化概率 (conversion probability) 或者放弃概率 (abandon probability) 来终止一个搜索会话。我们形式化定义这两种概率如下。

定义 4. [Conversion Probability] 对于一个搜索会话中的任意 item page history  $h_t$  ( $t > 0$ ), 令  $B(h_t)$  表示用户在观察到  $h_t$  之后发生购买行为的随机事件, 则  $h_t$  的 conversion probability, 记为  $b(h_t)$ , 就是事件  $B(h_t)$  在  $h_t$  下发生的概率。

定义 5. [Abandon Probability] 对于一个搜索会话中的任意 item page history  $h_t$  ( $t > 0$ ), 令  $L(h_t)$  表示用户在观察到  $h_t$  之后离开搜索会话的随机事件, 则  $h_t$  的 abandon probability, 记为  $l(h_t)$ , 就是事件  $L(h_t)$  在  $h_t$  下发生的概率。

由于  $h_t$  是在  $(t-1)$  时刻的 item page history  $h_{t-1}$  上执行动作  $a_{t-1}$  的直接结果, 因此  $b(h_t)$  和  $l(h_t)$  也表征了 Agent 在  $h_{t-1}$  上执行动作  $a_{t-1}$  之后环境状态的转移: (1) 以  $b(h_t)$  的成交概率终止搜索会话; (2) 以  $l(h_t)$  的离开概率终止搜

索会话；(3) 以  $(1 - b(h_t) - l(h_t))$  的概率继续搜索会话。方便起见，我们对用户继续进行搜索会话的概率也进行形式化描述。

定义 6. [Continuing Probability] 对于一个搜索会话中的任意 item page history  $h_t$  ( $t > 0$ )，令  $C(h_t)$  表示用户在观察到  $h_t$  之后继续停留在会话中的随机事件，则  $h_t$  的 continuing probability，记为  $c(h_t)$ ，就是事件  $C(h_t)$  在  $h_t$  下发生的概率。

显然，对于任意 item page history  $h$ ，都有  $c(h) = 1 - b(h) - l(h)$  成立。特殊地，对于初始 item page history  $h_0$  来讲， $C(h_0)$  是一个必然事件（即  $c(h_0) = 1$ ）。这是因为在第一个 item page 展示给用户前，不可能存在成交转化事件和离开事件。

基于上面定义的几个概念，我们可以定义 search session MDP (SSMDP) 如下：

定义 7. [Search Session MDP] 令  $q$  为某个 query， $\mathcal{D}$  为和  $q$  相关的商品全集， $K$  为一个页面可展示的商品数量，关于  $q$ 、 $\mathcal{D}$  和  $K$  的 search session MDP 是一个元组，记为  $\mathcal{M} = \langle T, \mathcal{H}, \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P} \rangle$ 。该元组中的每个要素分别为：

- \*  $T = \lceil \frac{|\mathcal{D}|}{K} \rceil$  为搜索会话最大决策步数，
- \*  $\mathcal{H} = \bigcup_{t=0}^T \mathcal{H}_t$  为关于  $q$ 、 $\mathcal{D}$  和  $K$  的所有可能的 item page history 的集合，其中  $\mathcal{H}_t$  为  $t$  时刻所有可能 item page history 的集合 ( $0 \leq t \leq T$ )，
- \*  $\mathcal{S} = \mathcal{H}_C \cup \mathcal{H}_B \cup \mathcal{H}_L$  为状态空间， $\mathcal{H}_C = \{C(h_t) | \forall h_t \in \mathcal{H}_t, 0 \leq t < T\}$  是包含所有的继续会话事件的非终止状态集合 (nonterminal state set)， $\mathcal{H}_B = \{B(h_t) | \forall h_t \in \mathcal{H}_t, 0 < t \leq T\}$  和  $\mathcal{H}_L = \{L(h_t) | \forall h_t \in \mathcal{H}_t, 0 < t \leq T\}$  分别是包含所有成交转化事件离开事件的终止状态集合 (terminal state set)，
- \*  $\mathcal{A}$  为动作空间，包含搜索引擎所有可能的排序打分函数，
- \*  $\mathcal{R} : \mathcal{H}_C \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$  为奖赏函数，
- \*  $\mathcal{P} : \mathcal{H}_C \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$  为状态转移函数，对于任意时间步  $t$  ( $0 \leq t < T$ )、任意 item page history  $h_t \in \mathcal{H}_t$  和任意动作  $a \in \mathcal{A}$ ，令  $h_{t+1} = (h_t, \mathcal{L}_K(\mathcal{D}_t, a))$ ，则 agent 在状态  $C(h_t)$  上执行动作  $a$  后，环境转移到任意状态  $s' \in \mathcal{S}$  的概率为

$$\mathcal{P}(C(h_t), a, s') = \begin{cases} b(h_{t+1}) & \text{if } s' = B(h_{t+1}), \\ l(h_{t+1}) & \text{if } s' = L(h_{t+1}), \\ c(h_{t+1}) & \text{if } s' = C(h_{t+1}), \\ 0 & \text{otherwise.} \end{cases} \quad (6.26)$$

在一个 search session MDP 中，环境即是所有可能用户共同构成的总体，环境的状态表征了用户总体在对应 item page history 下的动向（继续会话、成交或离开）。环境状态的转移则直接基于我们之前定义的 conversion probability、abandon probability 以及 continuation probability。奖赏函数  $\mathcal{R}$  可以根据具体的业务目标进行定义。基于让天下没有难做的生意的使命，也就是尽可能多地促进用户与卖家之间的交易，我们给出如下的奖赏函数范例。对于任意时间步  $t$  ( $0 \leq t < T$ )、任意 item page history  $h_t \in \mathcal{H}_t$  和任意动作  $a \in \mathcal{A}$ ，令  $h_{t+1} = (h_t, \mathcal{L}_K(\mathcal{D}_t, a))$ ，则 agent 在状态  $C(h_t)$  上执行动作  $a$  并且环境转移到任意状态  $s' \in \mathcal{S}$  的奖赏为

$$\mathcal{R}(C(h_t), a, s') = \begin{cases} m(h_{t+1}) & \text{if } s' = B(h_{t+1}), \\ 0 & \text{otherwise,} \end{cases} \quad (6.27)$$

其中， $m(h_{t+1})$  表示 item page history  $h_{t+1}$  对应的成交均价。

## 6.7.5 理论分析

### 6.7.5.1 马尔可夫性质

上一节定义的 search session MDP (SSMDP) 可以看作是 MDP 模型的一个实例，但要保证 SSMDP 是良定义的，我们需要证明 SSMDP 中的状态都具有马尔可夫性质 (Markov Property)。马尔可夫性质指的是对于任意的状态动作序列  $s_0, a_0, s_1, a_1, s_2, \dots, s_{t-1}, a_{t-1}, s_t$ ，都有如下等式成立：

$$\Pr(s_t | s_0, a_0, s_1, a_1, \dots, s_{t-1}, a_{t-1}) = \Pr(s_t | s_{t-1}, a_{t-1}). \quad (6.28)$$

也即是说，当前状态  $s_t$  的发生概率仅仅取决于最近一个状态动作对  $(s_{t-1}, a_{t-1})$ ，而并非整个序列。我们可以证明对于一个 SSMDP，它的所有状态都具有马尔可夫性质。

命题 1 对于任意 search session MDP  $\mathcal{M} = \langle T, \mathcal{H}, \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P} \rangle$ ，其状态空间  $\mathcal{S}$  中的任意状态都具有马尔可夫性质。证明：我们只需证明对于任意时间步  $t$  ( $0 \leq t \leq T$ ) 和关于  $t$  的任意状态动作序列  $s_0, a_0, s_1, a_1, \dots, s_{t-1}, a_{t-1}, s_t$ ，都有等式  $\Pr(s_t | s_0, a_0, s_1, a_1, \dots, s_{t-1}, a_{t-1}) = \Pr(s_t | s_{t-1}, a_{t-1})$  成立即可。

除了状态  $s_t$  以外，序列  $s_0, a_0, s_1, a_1, \dots, s_{t-1}, a_{t-1}, s_t$  中的其他所有状态都是非终止状态 (non-terminal state)。根据状态的定义，对于任意时间步  $t'$  ( $0 < t' < t$ )，必然存在一个 item page history  $h_{t'}$  与状态  $s_{t'}$  相对应，且有  $s_{t'} = C(h(t'))$ 。因此，整个序列可以重写为  $C(h_0), a_0, C(h_1), a_1, \dots, C(h_{t-1}), a_{t-1}, s_t$ 。需注意的是，对于任意时间步  $t'$  ( $0 < t' < t$ )，都有

$$h_{t'} = (h_{t'-1}, \mathcal{L}_K(\mathcal{D}_{t'-1}, a_{t'-1})), \quad (6.29)$$

成立。其中， $\mathcal{L}_K(\mathcal{D}_{t'-1}, a_{t'-1})$  也就是关于  $(t' - 1)$  时刻的动作  $a_{t'-1}$  和未展示商品  $\mathcal{D}_{t'-1}$  的 top  $K$  list。给定  $h_{t'-1}$ ，集合  $\mathcal{D}_{t'-1}$  一定是确定的。所以， $h_{t'}$  也就是状态动作对  $(C(h_{t'-1}), a_{t'-1})$  的必然和唯一结果。那么事件  $(C(h_{t'-1}), a_{t'-1})$  也能够等价地表示为事件  $h_{t'}$ 。基于此，我们可以进行如下推导：

$$\begin{aligned} & \Pr(s_t | s_0, a_0, s_1, a_1, \dots, s_{t-1}, a_{t-1}) \\ &= \Pr(s_t | C(h_0), a_0, C(h_1), a_1, \dots, C(h_{t-1}), a_{t-1}) \\ &= \Pr(s_t | h_1, h_2, \dots, h_{t-1}, C(h_{t-1}), a_{t-1}) \\ &= \Pr(s_t | h_{t-1}, C(h_{t-1}), a_{t-1}) \\ &= \Pr(s_t | C(h_{t-1}), a_{t-1}) \\ &= \Pr(s_t | s_{t-1}, a_{t-1}). \end{aligned} \quad (6.30)$$

第三步推导成立是由于对于任意时间步  $t'$  ( $0 < t' < t$ )， $h_{t'-1}$  都包含在  $h_{t'}$  中。类似地，第四步推导成立是由于事件  $C(h_{t-1})$  已经包含了  $h_{t-1}$  的发生。



### 6.7.5.2 折扣率

在这一小节我们将讨论本文最重要的问题：延迟奖赏（delay reward）对于搜索排序的优化到底有没有作用？简单地来说，也就是 search session MDP 的折扣率（discount rate）到底应该设多大。在任意 MDP 中，折扣率  $\gamma$  的大小直接决定了 future rewards 在 agent 的优化目标中所占比重。我们将分析优化长期累积奖赏与优化搜索引擎的经济指标这两个目标之间的关系给出答案。

令  $\mathcal{M} = \langle T, \mathcal{H}, \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P} \rangle$  为一个关于 query  $q$ 、商品全集  $\mathcal{D}$  和正整数  $K$  ( $K > 0$ ) 的 SSMDP。给定一个确定性策略  $\pi : \mathcal{S} \rightarrow \mathcal{A}$ ，记每个时间步  $t$  ( $0 \leq t \leq T$ ) 对应的 item page history 为  $\pi$  by  $h_t^\pi$ ，我们把在策略  $\pi$  下能够访问的所有状态都展示在图 9 中。

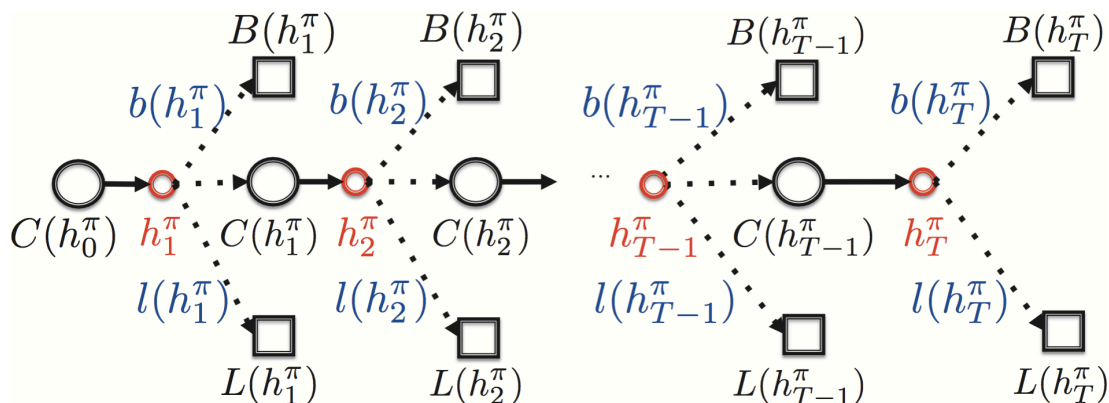


图 6.18: Agent 策略  $\pi$  下能够访问 SSMDP 中的所有状态

在这个图中，红色的节点表示 item page history，注意它们并不是 SSMDP 的状态。方便起见，在本文接下来的部分，我们将把  $C(h_t^\pi)$ 、 $c(h_t^\pi)$ 、 $b(h_t^\pi)$  和  $m(h_t^\pi)$  分别简化记为  $C_t^\pi$ 、 $c_t^\pi$ 、 $b_t^\pi$  和  $m_t^\pi$ 。

不失一般性，我们设 SSMDP  $\mathcal{M}$  的折扣率为  $\gamma$  ( $0 \leq \gamma \leq 1$ )。由于 SSMDP 是一个有限时间步 MDP（finite-horizon MDP），所以折扣率可以取到 1。对于任



意时间步  $t$  ( $0 \leq t < T$ ), 状态  $C_t^\pi$  的 state value 为

$$\begin{aligned} V_\gamma^\pi(C_t^\pi) &= \mathbb{E}^\pi \left\{ \sum_{k=1}^{T-t} \gamma^{k-1} r_{t+k} \middle| C_t^\pi \right\} \\ &= \mathbb{E}^\pi \{ r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{T-t-1} r_T \middle| C_t^\pi \}. \end{aligned} \quad (6.31)$$

其中, 对任意  $k$  ( $1 \leq k \leq T-t$ ),  $r_{t+k}$  为 agent 在未来时刻  $(t+k)$  的 item page history  $h_{t+k}^\pi$  中收到的即时奖赏。根据我们奖赏函数的定义,  $r_{t+k}$  在策略  $\pi$  下的期望值为  $\mathbb{E}^\pi \{ r_{t+k} \} = b_{t+k}^\pi m_{t+k}^\pi$ 。在这里,  $m_{t+k}^\pi = m(h_{t+k}^\pi)$  为 item page history  $h_{t+k}^\pi$  的成交价格期望。由于  $V_\gamma^\pi(C_t^\pi)$  表达的是在  $C_t^\pi$  发生的条件下的长期累积奖赏期望, 所以我们要把从  $C_t^\pi$  到达 item page history  $h_{t+k}^\pi$  的概率考虑进来。记从状态  $C_t^\pi$  到达  $h_{t+k}^\pi$  的概率为  $\Pr(C_t^\pi \rightarrow h_{t+k}^\pi)$ , 根据状态转移函数的定义可以得到

$$\Pr(C_t^\pi \rightarrow h_{t+k}^\pi) = \begin{cases} 1.0 & k = 1, \\ \prod_{j=1}^{k-1} c_{t+j}^\pi & 1 < k \leq T-t. \end{cases} \quad (6.32)$$

从状态  $C_t^\pi$  到 item page history  $h_{t+1}^\pi$  的概率为 1 是因为  $h_{t+1}^\pi$  是状态动作对  $(C_t^\pi, \pi(C_t^\pi))$  的直接结果。将上面的几个公式综合起来, 我们可以进一步计算  $V_\gamma^\pi(C_t^\pi)$  如下:

$$\begin{aligned} V_\gamma^\pi(C_t^\pi) &= \mathbb{E}^\pi \{ r_{t+1} \middle| C_t^\pi \} + \gamma \mathbb{E}^\pi \{ r_{t+2} \middle| C_t^\pi \} + \cdots + \gamma^{T-t-1} \mathbb{E}^\pi \{ r_T \middle| C_t^\pi \} \\ &= \sum_{k=1}^{T-t} \gamma^{k-1} \Pr(C_t^\pi \rightarrow h_{t+k}^\pi) b_{t+k}^\pi m_{t+k}^\pi \\ &= b_{t+1}^\pi m_{t+1}^\pi + \gamma c_{t+1}^\pi b_{t+2}^\pi m_{t+2}^\pi + \cdots + \gamma^{T-t-1} \left( \prod_{j=1}^{T-t-1} c_{t+j}^\pi \right) b_T^\pi m_T^\pi \\ &= b_{t+1}^\pi m_{t+1}^\pi + \sum_{k=2}^{T-t} \gamma^{k-1} \left( \left( \prod_{j=1}^{k-1} c_{t+j}^\pi \right) b_{t+k}^\pi m_{t+k}^\pi \right). \end{aligned} \quad (6.33)$$

根据图 9 中展示每个 item page history 的 conversion probability 以及成交价格期望, 我们也可以将搜索引擎在策略  $\pi$  的作用下在一个搜索会话中引导的

成交额期望表达出来，即

$$\begin{aligned}\mathbb{E}_{gmv}^{\pi} &= b_1^{\pi} m_1^{\pi} + c_1^{\pi} b_2^{\pi} m_2^{\pi} + \cdots + (\Pi_{k=1}^T c_k^{\pi}) b_T^{\pi} m_T^{\pi} \\ &= b_1^{\pi} m_1^{\pi} + \sum_{k=2}^T (\Pi_{j=1}^{k-1} c_j^{\pi}) b_k^{\pi} m_k^{\pi}.\end{aligned}\tag{6.34}$$

通过比较  $\mathbb{E}_{gmv}^{\pi}$  和  $V_{\gamma}^{\pi}$ ，不难发现当折扣率  $\gamma = 1$  时，有  $\mathbb{E}_{gmv}^{\pi} = V_{\gamma}^{\pi}(C_0^{\pi})$  成立。也就是说，当  $\gamma = 1$  时，最大化长期累积奖赏将直接带来搜索引擎成交额的最大化。当  $\gamma < 1$  时，由于  $\mathbb{E}_{gmv}^{\pi}$  是  $V_{\gamma}^{\pi}(C_0^{\pi})$  的上界，所以最大化  $V_{\gamma}^{\pi}$  并不一定能够最大化  $\mathbb{E}_{gmv}^{\pi}$ 。

命题 2 令  $\mathcal{M} = \langle T, \mathcal{H}, \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P} \rangle$  为任意 search session MDP。对于任意确定性策略  $\pi : \mathcal{S} \rightarrow \mathcal{A}$  和折扣率  $\gamma$  ( $0 \leq \gamma \leq 1$ )，都有式子  $V_{\gamma}^{\pi}(C(h_0)) \leq \mathbb{E}_{gmv}^{\pi}$  成立，其中  $V_{\gamma}^{\pi}$  为 agent 在策略  $\pi$  和折扣率  $\gamma$  下的状态值函数， $C(h_0)$  为搜索会话的初始状态， $\mathbb{E}_{gmv}^{\pi}$  为搜索引擎在策略  $\pi$  下的单次搜索会话成交额期望。仅当  $\gamma = 1$  时，有  $V_{\gamma}^{\pi}(C(h_0)) = \mathbb{E}_{gmv}^{\pi}$  成立。证明：我们只需证明当  $\gamma < 1$  时，有  $V_{\gamma}^{\pi}(C(h_0)) < \mathbb{E}_{gmv}^{\pi}$  成立。这是显然的，因为二者之差，即  $\sum_{k=2}^T (1 - \gamma^{k-1})(\Pi_{j=1}^{k-1} c_j^{\pi}) b_k^{\pi} m_k^{\pi}$  在  $\gamma < 1$  时一定为正。

至此，我们可以回答之前提出的问题：站在提高搜索引擎成交额的角度，搜索排序问题中考虑延迟奖赏是必要且必须的。从理论上，这是因为最大化无折扣累积奖赏能够直接优化搜索引擎的成交额。究其深层原因，是因为用户在搜索商品的每个步骤（即每个 item page history）的行为都是基于之前观察到的所有信息（或者大部分信息）作出的反应，这天然决定了搜索排序问题的 sequential decision-making 本质。

## 6.7.6 实验分析

我们根据图 9 设计了一个搜索排序的模拟实验来说明上一节理论分析结果。我们设定一个搜索会话的最大决策步数为 30，因为从第 2 节的数据分析来看，长度为 30 的 PV 序列总占比不超过 1%。我们通过线上真实数据生成商品候选集合。在每个状态上，agent 可以选择动作集  $A = \{a_1, a_2, a_3, a_4, a_5\}$  中的任意一

个动作对商品进行排序，进行页面展示。对于任意时间步  $t$  ( $0 < t \leq 30$ ) 以及任意 item page history  $h_t$ ， $h_t$  通过其最近 4 个商品页的商品特征来进行表示。同时， $h_t$  的 conversion probability、continuing probability 以及 abandon probability 也直接通过  $h_t$  的特征生成。

由于实验规模不大，我们可以直接用 Tabular 强化学习方法对问题进行求解。我们分别采用 Q-learning 算法、Actor-Critic 方法和 Dynamic Programming 方法，在折扣率  $\gamma$  为 0、0.1、0.5、0.9 和 1.0 的情况下进行测试。每一个实验设置运行算法 50 次，每一次包含 80 万次搜索会话，我们记录下学习过程中每个搜索会话的平均成交额以及初始状态  $h_0$  的 state value。这里仅给出部分结果。我们首先对比两个极端情况  $\gamma = 0$  和  $\gamma = 1.0$  时，每个算法取得的 GMV 指标。如图 10 所示，三个算法在  $\gamma = 1.0$  时的 GMV 都要高于  $\gamma = 0$  时的 GMV。单看结果最直观的 DP 方法，可以发现它在两个折扣因子下的 GMV 有 6% 的 gap。如前所述，折扣因子  $\gamma = 0$  即是分别独立优化 Agent 在每个状态上的即时奖赏，而  $\gamma = 1.0$  则是直接优化 GMV。

我们选取 Actor-Critic 方法，考察它在不同折扣率下的 GMV 曲线以及初始状态  $h_0$  的 state value  $V_\gamma(h_0)$  的变化情况，结果如图 11 所示。可以看到，Actor-Critic 方法在不同折扣率下的 GMV 指标与图 10 的结果一致，而图右边的值函数变化曲线更直接证明我之前的理论分析。只有折扣因子  $\gamma = 1.0$  时， $V_\gamma(h_0)$  的值才和对应的 GMV 几乎相等。需要注意的是，Actor-Critic 方法在  $\gamma$  等于 0.9 和 1.0 时的 GMV 曲线基本重合，并不能说明优化  $V_{0.9}$  能够优化 GMV，而只能说明优化  $V_{0.9}$  与优化  $V_{1.0}$  得到的最优策略恰好相同，二者本质上并不一样。

至此，我们已经通过理论分析和实验证明了搜索排序问题是一个有限时间步的无折扣 (Finite-Horizon Undiscounted) 的顺序决策问题。当然，我们所有的结论都是在以最大化成交额为目标的前提下得到的。如果追求点击率或者单纯转化率之类的目标，结论可能会有不同，但都可以采用与本文类似的方法进行分析。