搜索引擎技术

王树森　著

目录

第一部分

搜索引擎基础

第 1 章 搜索引擎技术概要

本章主要概述搜索引擎的关键技术，后面的章节将围绕这些关键技术展开。在讲解技术之前，首先在 1.1 节介绍搜索本书用到的一些基本概念。1.2 节讨论相关性、内容质量、时效性、个性化、地域性，它们是决定用户对搜索结果满意程度的主要因子。1.3 节讲解搜索引擎的链路，即用户搜索一条查询词之后，搜索引擎具体做了哪些计算。

1.1 基本概念

在开始讲解技术内容之前，首先介绍一些搜索引擎的基本概念。如图 1.1(a) 所示，用户在搜索框中输入查询词（query），此时搜索框下方有多条查询建议（suggestion，缩写 SUG），用户可以按键盘上的“搜索”键触发搜索，也可以点击 SUG 中的查询词触发搜索。查询词会被发送给服务器，服务器开始做计算，并将计算的结果传输给用户的 APP 或网页。

(a) 搜索中　(b) 搜索结果页（双列）　(c) 搜索结果页（单列）

图 1.1: 搜索的界面

服务器返回的结果会显示在搜索结果页上，以双列或单列的产品形态呈现，如图 1.1 所示。本书按照信息检索领域的惯例，将搜索结果叫作文档（document），不论它是何种类型。谷歌搜出的文档主要是网页，B 站搜出的文档主要是长视频，小红书搜出的文档主要包括图文笔记和短视频，京东搜出的文档主要是商品。在搜索框与文档之间通常有“最新”、“最热”、“用户”这样的标签，用途是对搜索结果做过滤或更改排序的方式。如果点击“最新”，那么搜索结果主要按照文档发布时间排序；如果用户点击“最热”，那么主要按

照文档热度排序；如果用户点击“视频”标签，那么搜索结果只保留视频。

曝光（impression）的意思是文档被用户看到。用户做搜索之后，首先会看到首屏的几篇文档，这些文档一定会曝光。如果用户对结果不满意，会下滑、翻页、甚至离开。如果用户对搜到的文档感兴趣，则会点击打开文档。点击率（click-through-rate，CTR）是指在文档 d 曝光的前提下，用户点击 d 的概率。通常来说，点击率高说明用户对搜到的文档满意。

用户点击进入文档之后，还可以发生交互行为（engagement）。在小红书、抖音、B 站等平台上，用户可以点赞、收藏、转发，可以关注作者，还可以留下评论。点赞、收藏等交互行为说明搜到的文档非常符合用户的需求。交互率是指在点击的前提下，用户发生交互的概率。交互率包括点赞率、收藏率、转发率、关注率、评论率、等等。交互率高说明用户对搜到的文档满意。对于京东、淘宝、亚马逊等电商搜索，在用户点击商品之后，可能会加购物车、下单、付款。对于电商搜索来说，最重要的指标是成交率与成交金额。

业内普遍将搜索引擎分为垂直搜索（垂搜）与通用搜索（通搜）两大类，本书主要基于通搜讲解搜索引擎技术，在这里简单对比垂搜和通搜。

- 垂搜是针对某一个行业的专业搜索引擎，典型代表为电商搜索、学术论文搜索、本地生活搜索、酒店机票搜索、租售房搜索、法律文书搜索、招聘网站搜索、股票基金搜索。垂搜的文档普遍是结构化的，可以根据文档属性做筛选。比如电商搜索中，文档是商品，有名称、品牌、卖家、价格、颜色；在学术论文搜索中，文档是论文，有标题、关键词、学科、作者、刊物名、发表时间。用户使用垂搜的意图通常很明确，使用京东是为了购物，使用谷歌学术是为了查论文，使用美团外卖是为了点餐，使用携程是为了搜酒店或航班。
- 通搜的典型代表是大家耳熟能详的谷歌、百度、必应、雅虎、头条，它们的覆盖面很广，不局限于一个垂类领域，且搜到的文档普遍是非结构化的。用户使用通搜的意图并不单一，查询词非常多样，给搜索带了很大的挑战。

小红书、抖音、B 站属于用户生成内容（user generated content，UGC）的平台，平台上的文档绝大多数为普通用户创作，文档覆盖面很广、且没有结构化。这几个 UGC 平台都提供搜索功能，它们不是传统意义上的通搜，但它们的场景、问题与通搜非常接近，技术方案可以照搬通搜。本书主要讨论通搜的技术，尤其适用于 UGC 平台的搜索引擎。

1.2 什么决定用户满意度？

搜索算法工程师的日常工作是对策略做迭代优化，目标是让搜索引擎变得“更好”。在讲解搜索技术之前，首先需要明确什么样的搜索结果好，什么样的搜索结果不好。好的搜索结果，就是让用户满意的结果；这听起来像是一句废话，但搜索引擎迭代优化的目标的的确确是提升用户体验。用户对搜索结果更满意，就会更喜欢这款搜索引擎，用户数量、活跃程度会增长，带动营收的增长。通用搜索引擎的营收主要是靠关键词广告费，即广告主向某些查询词投放广告，按照广告的曝光次数或点击次数收费。

我们已经明确了这个结论：通用搜索引擎迭代优化的方向是提升用户满意度。经过

业界长期的探索与实践，总结出影响用户满意度的三大因子——相关性、内容质量、时效性——其中任何一条没做好，都很容易被用户感知到，影响用户体验。此外，近些年来，随着智能手机的普及，移动端 APP 可以通过用户的历史记录刻画用户的兴趣点，让搜索引擎做到个性化，且我们发现个性化可以显著提升用户体验。我们还发现地域性也是影响用户体验的一个因子，尽管它的效用低于相关性、内容质量、时效性、个性化。接下来我们逐一讨论上述 5 个影响用户满意度的因子。

相关性是影响用户满意度的因子中最重要的一个，没有之一。相关性是指查询词 q 和文档 d 在语义上相关的程度，而不仅是文本匹配程度。相关性是一个客观标准，只取决于 q 和 d，而与具体的用户 u 无关。如果有背景知识的人普遍认为 q 与 d 语义相关，且 d 能满足 q 的需求，那么两者就具有高相关性。高相关性容易理解，但是低相关性有时不容易理解，有些 (q, d) 二元组貌似相关，但实则相关性很低。表 1.1 用几个负面的例子说明什么是低相关性。

表 1.1: 低相关性的例子与分析

查询词	文档	低相关性原因
阿迪跑步鞋	《阿迪王跑步鞋测评》	文本匹配，但语义不匹配， 用户想搜的是阿迪达斯
亚马逊 雨林生物	……我在亚马逊上买了一本 东南亚热带雨林生物书……	语义不匹配， 此亚马逊非彼亚马逊。
属相相冲的人 能否在一起	《不同属相的性格分析》	对用户有参考价值， 但没有回答用户的问题。
正宗跷脚牛肉	《我的乐山游记》 ……去四川乐山旅游……看乐山大佛…… 大渡河……吃了正宗跷脚牛肉……	只是顺便一提，不是用户 需要的教程或探店内容

内容质量是给文档 d 打的分数，这个分数相对静态，不容易发生变化。内容质量取决于文档的来源和图文质量。

- 文档是否可信，很大程度上取决于文档的来源（包括作者和网站）。对于谷歌这样的网页搜索引擎，网站的声誉是内容质量的维度之一，比如用户搜热门新闻，BBC、美联社远比小报、自媒体可信，因此排名更高。对于电商搜索，卖家的信誉影响商品排名，品牌官方旗舰店、常年信誉良好的卖家比普通卖家可信。对于小红书这样的 UGC 平台，作者的专业性是内容质量的维度之一，比如经过平台认证的医生、律师创作的专业内容比普通用户创造的内容更可信。
- 文档图文质量会影响用户的体验。文字质量考察事实是否准确、传递的信息是否对他人有价值、写作是否专业和认真。两篇旅游景点的攻略，一篇详细阐述自己被坑的经历，另一篇只是说自己被坑、发泄对景区的不满情绪；前者写作认真、对读者有帮助，因此文字质量优于后者。很多文档包含图片、视频，它们的分辨率、画质、美感是影响用户体验的部分因素。两篇讲解搜索技术的博客，一篇用精美的原创图片，另一篇用各种来源的截图、甚至还有多个水印，很显然前者的图片质量更高。

内容质量对排序起多大作用，要看用户搜哪方面的话题。有一类文档叫做“金钱或生命”（your life or your money，YMYL），主要指医疗、理财、法律这些类目，高内容质量的文档应该是由专业人士撰写，且事实准确、行文严谨。如果用户搜索 YMYL 方面的话题，那么内容质量在排序中的权重应当很大。如果用户搜索搞笑、八卦等不严肃的话题，那么内容质量对用户满意度没有多大影响。

时效性主要指查询词 q 对新内容的需求有多强，需求越强，则文档年龄对排序起到的作用越大。查询词的时效性主要分为无时效性、突发时效性、一般时效性、周期时效性。

- 很多查询词并没有对新内容的需求或偏好。比如“柴犬饲养方法”、“狭义相对论”、“干呕是不是咽炎的症状”，都是想找客观存在的事实和答案，新旧内容的价值相同。比如“可爱的猫的视频”、“情人节文案”、“肖申克的救赎”都是长期相对稳定的资源。再比如“83 版射雕视频”、“08 奥运歌曲”带有具体时间，是已经发生过的事物，发生之后不太可能再变化。
- 突发时效性是指突发的新闻、热点、舆论，过一段时间热度会消退，变成无时效性。举个例子，在 2022 年世界杯决赛前后，q =“阿根廷”具有突发时效性，用户搜“阿根廷”普遍是想看比赛、队员最新的情况，而非想了解这个国家。如果在那个时间点无法判断“阿根廷”具有突发时效性，则搜不出用户想看的文档。如何判别突发时效性呢？假如一个人不看新闻，他单从 q =“阿根廷”的文本上看不出突发时效性。既然人无法从字面判断突发时效性，那么自然语言模型同样也做不到。判别突发时效性的方法主要是数据挖掘，比如监控到 q 最近搜索量激增，那么 q 很可能有突发时效性。
- 一般时效性是指查询词表达出对新文档的偏好，体现在字面上，比如“新款爱马仕包”、“拼多多薅羊毛”、“现在办护照容易吗”。一个人只需有基本常识、不需要看新闻，他就能判断出一般时效性，因此自然语言模型同样可以判断一般时效性。一般时效性通常分几档，比如强、中、弱，对于查询词 q，年龄大的问题提供的价值越低，则 q 的时效性需求越强。
- 周期时效性是指查询词在每年的特定时间段表现为突发时效性，而在其他时间段无时效性。比如查询词带有“双 11”、“春运”、“春晚”、“高考”，则在特定时间段会体现出突发时效性。比如在高考结束几天内，搜“高考语文”，应当出最新的文档；再过一段时间，“高考语文”就没有时效性，用户可能是想找历年的题。

地域性的意思是当查询词 q 表达出对距离近的需求时，搜索引擎需要考虑到用户定位地点与文档中的地点（point of interest，POI）的距离。如果不对地域性的查询词做特殊处理，上海的用户搜索 q =“附近的美食”，搜索引擎可能给用户曝光文档“北京前门附近的美食”。当然，只有当用户允许 APP 获取设备定位时，搜索引擎才能解决好地域性的查询词。搜索引擎需要从下面几个维度对查询词做分析。

- 显式与隐式地域性。显式的意思是查询词明确表达了对地域性的需求，例如查询词是“附近的酒店”、“本地景点”、“同城租房”。隐式的意思是查询词没有明说，但是隐含了对地域性的需求，例如“情人节餐厅”、“去哪遛娃”、“周边游”。

- 地域性需求的强度。“附近的美食”、“本地景点”这样的查询词的地域性需求明确，因此强度为 100%，全部的搜索结果都应该是符合地域性需求的。“火锅店”、“星巴克”的地域性需求不明确，搜索引擎不确定用户的需求究竟是什么，因此部分搜索结果符合地域性需求，其余搜索结果不考虑地域性。但“火锅店”与“喜茶”的地域性需求强度有所不同，前者强于后者，用户搜“火锅店”很可能是想找附近的火锅店，而搜“喜茶”很可能是想看最近的新品，有较低的可能性是想看附近的店。
- 对距离的敏感程度。“附近的美食”、“周边游”对距离的敏感程度是不同的，前者通常要求步行可以到达的餐厅，而后者是驾车可以到达的景点。因此，搜索引擎做距离筛选时要考虑到查询词对距离的敏感程度。

一个稍加训练的人不难从查询词的字面判断出查询词是否带有地域性需求、地域性需求有多强、对距离的敏感度有多高，因此可以用自然语言模型识别查询词的地域性。

个性化的意思是搜索引擎要考虑到用户的兴趣点，不同的用户搜索相同的查询词，搜索结果有所差异，这就是所谓的“千人千面”。为什么搜索要需要个性化？什么样的搜索引擎可以做到个性化？搜索引擎如何利用用户特征做到千人千面？

- 当查询词比较宽泛时，个性化可以帮助用户找到真正想找的文档。举个例子，用户搜索“头像”这个宽泛的查询词，如果系统知道用户 u 喜欢《指环王》和《火影忍者》，那么搜索结果中应当多包含《指环王》和《火影忍者》中的人物头像。否则，泛泛地给用户展示各式各样的头像，大概率不是用户想要找的。
- 有些搜索引擎能做好个性化，而有些搜索引擎较难做好。小红书、抖音、YouTube 的用户在登录之后使用推荐和搜索，留下丰富的历史行为记录，算法可以推断出用户的兴趣点，让搜索变得更精准。但是网页版的百度就很难做到个性化，原因是用户往往不登录，且用户行为不够丰富，算法不够理解用户，较难做到千人千面。
- 搜索引擎将查询词 q、文档 d、用户 u 三者的特征输入机器学习模型，模型预估点击率和交互率，与相关性等分数一同作为排序的依据。预估的意思是在 d 尚未曝光给 u 之前，模型就预测 u 有多大概率会点击、点赞、收藏、转发、关注、评论。如果模型预测 u 点击和交互 d 的概率很高，那么搜索引擎会将 d 排在靠前的位置；反之，如果模型预测的概率很小，那么搜索引擎恐怕不会让 d 出现在搜索结果页上。

值得注意的是，即便是非个性化的搜索引擎，也会使用 q 和 d 的特征预估点击率和交互率，有助于把更优质的文档排在前面。当然，如果有丰富的用户行为特征，那么点击率和交互率的预估会更准确，有助于提升用户体验。

1.3 搜索引擎链路

用户在搜索框中输入查询词，搜索引擎返回若干文档，并按照顺序呈现给用户。从用户点击搜索按钮，到看到最终搜索结果，中间耗时约数百毫秒。在这期间，搜索引擎具体做了什么呢？这就要说到搜索引擎的链路。如图 1.2 所示，搜索引擎的链路主要分为查询词处理（query processing，QP）、召回（retrieval）、排序（ranking）。接下来我们逐一讨论 QP、召回、排序。

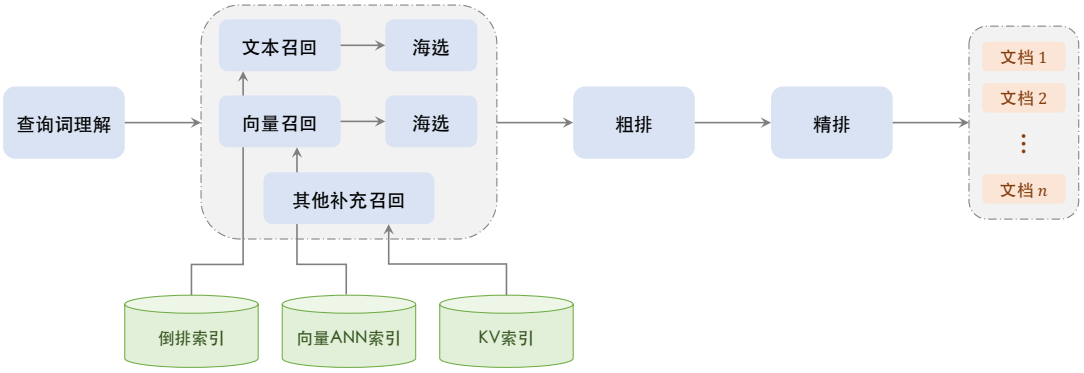

图 1.2: 搜索引擎的链路

QP 是链路上的第一环，用自然语言处理（natural language processing，NLP）技术从查询词中提取很多信息，共链路下游的召回和排序使用。这里列出 QP 最重要的一些任务。

- 分词的意思是把整条查询词切分成多个词，比如把“冬季卫衣推荐”切分成 3 个词“冬季　卫衣　推荐”。分词的主要用途是文本召回，分别检索包含“冬季”、“卫衣”、“推荐”的文档，对得到的 3 个文档集合取交集，得到的文档均同时包含 3 个词。
- 词权重的意思是计算分词得到的每个词的重要性。在上面的例子中，词的权重为“卫衣”>“冬季”>“推荐”。丢掉“卫衣”，搜“冬季推荐”，完全无法满足用户需求；丢掉“冬季”，搜索结果中有春秋卫衣和冬季卫衣，部分搜索结果满足用户需求；丢掉“推荐”，几乎不影响搜索结果。词权重主要用途是召回的丢词。对于长度较大的查询词，完全匹配查询词的文档数量往往不足，此时需要丢弃权重低的词，增加召回的文档数量。
- 改写的意思是把用户输入的查询词 q 改写成其他表达方式 $q'_1, \cdots, q'_n$，分别用 n 个改写查询词做召回，补充到 q 召回的结果中。举个例子，用户输入 q =“吃布洛芬会有什么不良反应”，改写成“布洛芬的不良反应”、“布洛芬的副作用”，这样可以召回文本匹配检索找不到的文档。
- 意图识别是一类任务的总称，包括时效性、地域性等做种意图。时效性意图的意思是用户对新内容的需求有多强，分为无时效性、突发时效性、一般时效性（细分为强、中、弱三档）、周期时效性。地域性意图的意思是用户对文档地理位置的需求，QP 需要判断显式或隐式地域性、地域性需求的强度、对距离的敏感程度。

QP 对搜索引擎起到两方面作用。第一，QP 输出的分词、词权重、改写是召回所需的输入，上面已经做出解释。第二，QP 意图识别的结果可以决定下游链路的调用。举个例子，“附近的美食”这样的查询词具有地域性意图，那么下游链路完全不同于普通的召回和排序，召回需要根据距离设阈值，排序需要把距离作为一个重要因子。

召回是链路上的第二环，从数亿的文档库中快速取回数万篇文档，并用简单的模型和规则给文档打分，把分数最高的数千篇文档作为召回结果。搜索引擎通常有数十路召回通道，它们可以分为 3 大类。

- 文本召回使用倒排索引做检索，给定 q，寻找与之匹配的文档 d。倒排索引是从词（token）到文档的索引，记作 $t \to \text{List}\langle d \rangle$。QP 环节将 q 分割成词 $[t_1, \cdots, t_k]$，利用倒排索引，检索每个 t 对应的文档，得到 k 个文档集合。对集合取交集（或其他布尔运算），得到最终文本召回的结果。除了用原查询词 q，还可以用丢词、改写后的 $\{q'\}$ 做文本召回，因此文本召回有多条通道。此外，为了应对时效性查询词，还需要为新文档设独立的召回通道。
- 向量召回的意思是分别把 q 和 d 表征为向量 $\boldsymbol{x}_q$ 与 $\boldsymbol{z}_d$，给定 $\boldsymbol{x}_q$，查找相似度最高的 $\boldsymbol{z}_d$，作为召回结果。文档数量巨大，不可能逐一计算 $\boldsymbol{x}_q$ 与所有文档向量的相似度，因此向量数据库提供近似最近邻索引（approximate nearest neighbor，ANN），快速从向量数据库中找出一批相似度高的文档向量。业界普遍用双塔模型学习 q 和 d 的向量表征，模型拟合的标签可以是相关性或用户点击。使用不同的标签做训练，得到的模型和向量表征有所不同，因此向量召回通道不止一条。向量召回有两方面的意义：一方面是语义召回，即召回字面不匹配、但是语义匹配的文档；另一方面是个性化，可以结合用户特征与查询词特征计算向量 $\boldsymbol{x}_q$，召回既相关、又符合用户兴趣的文档。
- 其他补充召回通过离线计算建立 key-value（KV）索引，在线上直接读取索引，作为召回结果。可以离线挖掘高相关性的 (q, d) 二元组，以 $q \to \text{List}\langle d \rangle$ 的形式存储，线上用 q 作为 key，直接从索引上获取文档列表。也可以离线计算查询词改写，以 $q \to \text{List}\langle q' \rangle$ 的形式存储，线上先用 q 检索改写词 q'，再用 q' 检索文档。

文本召回、向量召回检索到的文档数量过大，在送入排序之前，需要做一个快速的初步筛选，我们称之为海选，具体方法与排序类似。

排序包括召回海选、粗排、精排，形成一个三级漏斗。打分量逐级减小，模型规模逐级增大，取得效果与成本间的平衡。

- 海选用轻量级的模型和规则给数万篇文档打分，从中选出数千篇，作为粗排的候选集。粗排用稍大的模型打分，从数千篇文档中选出数百篇，作为精排的候选集。精排用深度神经网络打分，给数百篇文档打分，按照融合分数排序，展示在搜索结果页上。
- 三级漏斗都需要在离线或线上调用各种模型，计算相关性、点击率（或交互率）、内容质量、时效性、地域性等多个分数。表 1.2 总结了用到的模型和规则，具体技术在后面的章节中解释。
- 在算出各项分数之后，需要用公式或模型融合各项分数，按照融合分数做排序。迭代优化搜索引擎的初期最好使用人工设定的公式融合分数，比如按照相关性分数分档，档内用各项分数的加权和做排序，这样方便迭代优化，发现问题之后可以通过调权重解决。迭代优化进行到一定阶段时，建议改用模型融分，通过拟合人工标注的满意度来训练模型，这样可以让业务指标大幅提升，但是会让迭代优化变得困难。

表 1.2: 搜索链路上的打分模型

	相关性 （线上计算）	**点击率** （线上计算）	**内容质量** （离线计算）	**时效性、地域性** （离线 + 线上）
召回海选	双塔 +GBDT	双塔	人工标注 + 离线模型	人工标注 + 离线挖掘 + 线上模型
粗排	DNN 或 BERT+GBDT	多目标双塔 或 多目标 DNN		
精排	BERT+GBDT	多目标 DNN		

1.4 知识点小结

- 搜索引擎迭代优化的目标是提升用户满意度。满意度可以拆分为相关性、内容质量、时效性、地域性、个性化等维度。搜索引擎用模型、数据挖掘、规则、人工标注等方法给计算各项分数。
- 搜索链路分为查询词理解（QP）、召回、排序。召回的数万篇文档经过多级排序，最终有数十或数百篇进入搜索结果页，给用户曝光。
- QP 为下游链路提供所需的信息，甚至可以决定调用哪条下游链路。QP 的主要任务包括分词、词权重、改写、意图识别。
- 召回的目标是快速检索到可能满足用户需求的文档，作为排序的候选集。召回通道主要分为文本召回、向量召回、其他补充召回这 3 类。
- 排序通常采用多级漏斗，比如分为召回海选、粗排、精排，打分量逐级递减，模型规模逐级增大。每一级排序都计算相关性、点击率等分数，并用公式或模型融合各项分数，用得到的总分对文档做排序。

第 2 章 搜索引擎的评价指标

前文讨论过，搜索引擎迭代优化的目标是提升用户对搜索结果的满意度。如果有可靠的指标量化用户满意度，那么我们就可以用这样的指标作为牵引，向着提升指标的方向做策略的优化。根据工业界的实践经验，可以用三大类指标量化用户满意度。2.1 节介绍用户规模与留存指标，它们是业务的核心指标，也叫北极星指标，它们直接与公司的业务发展挂钩。2.2 节介绍一些中间过程指标，包括点击、交互、换词，它们通常与用户体验正相关，但它们不是搜索引擎迭代优化的主要目标。2.3 节介绍人工评估的方法与指标，这是最直接反映用户满意度的评价方法，但也存在不足之处。

2.1 用户规模与留存指标

搜索引擎迭代优化的目标是提升用户满意度，如果用户对搜索结果更满意，则会有更高的粘性，带来用户规模和用户留存的增长。用户规模和用户留存是搜索的核心指标，如果算法和产品的策略能带动规模和留存的增长，那么几乎可以肯定策略是有益的，值得全量上线。规模和留存指标与公司的商业利益也是高度一致的。通搜的营收主要来自关键词广告，即广告主对圈定的查询词投放广告，在搜索结果页显示，按照点击次数收费。显然，用户规模越大、粘性越高，则广告获得的曝光、点击越多，公司营收越高。

2.1.1 用户规模

业界常用日活用户数（daily active user，DAU）与月活用户数（monthly active user，MAU）衡量用户规模。如果一位用户今天使用 1 次 APP、或者使用 10 次 APP，他对 DAU 的贡献都是 1。如果一个用户在未来 28 天内使用 1 次 APP、或者使用 10 次 APP，他对今天的 MAU 的贡献都是 1。周活用户数（weekly active user，WAU）用类似的方式定义。

百度、小红书、抖音这样的手机 APP 同时有搜索和推荐两个功能，需要拆分搜索日活用户数（search daily active user，SDAU）和推荐日活用户数（feed daily active user，FDAU）。SDAU 比 DAU 更直接反映搜索做得好不好。某些用户登录 APP 只做搜索，找到想要的信息就离开，不使用推荐；而某些用户只看推荐信息流，不使用搜索。由于存在很多既用搜索也用推荐的用户，SDAU 和 FDAU 都小于 DAU，而 SDAU+FDAU 则大于 DAU。

SDAU 与 DAU 的比值叫作“搜索渗透率”，即有多大比例的 APP 用户使用搜索功能。搜索渗透率越高，说明搜索做得越好，用户越习惯使用 APP 的搜索功能。类似的，FDAU 与 DAU 的比值叫做推荐渗透率，反映推荐做得好不好。以百度为例，百度 APP 的搜索功能占据主导地位，因此不看搜索渗透率，而是看推荐渗透率。而小红书则相反，先有推荐、后有搜索，因此会将搜索渗透率作为核心指标之一。

2.1.2 用户留存

用户留存率（retention rate）衡量 APP 留住用户的能力。有两种留存指标，一个种做“次 n 日内留存率（次 n 留）”，另一种叫做“第 n 日留存率（第 n 留）”。次 n 留的意思是今天（即第 $t+0$ 天）使用 APP 的用户中，有多大比例的用户在未来 $t+n$ 天内使用至少一次 APP。第 n 留的意思是今天使用 APP 的用户中，有多大比例的用户在第 $t+n$ 天使用 APP。工业界使用最多的指标为次 1 留和次 7 留。次 n 留指标会滞后 n 天，也就是说实验上线 n 天之后才能计算次 n 留。

例 2.1. 次 1 留与第 1 留

2 月 1 日有 1 亿用户使用某 APP。这 1 亿用户中，有 6 千万人在 2 月 2 日使用了该 APP。2 月 1 日的次 1 留、第 1 留都等于 60%。请注意，当 $n=1$ 时，两种留存指标相等。

♣

例 2.2. 次 7 留与第 7 留

2 月 1 日有 1 亿用户使用某 APP。这 1 亿用户中，有 8 千万人在 2 月 2 日到 8 日之间使用该 APP 至少一次，那么 2 月 1 日的次 7 留等于 80%。这 1 亿用户中，有 3 千万人在 2 月 8 日当天使用该 APP 至少一次，那么 2 月 1 日的第 7 留等于 30%。请注意，次 n 留总是大于等于第 n 留，且次 n 留随 n 单调递增。通常来说，第 n 留随 n 增加而减小。

♣

有些手机 APP 同时有搜索和推荐的功能，可以把留存限定在搜索或推荐。搜索次 n 留的意思是今天（即第 $t+0$ 天）使用搜索的用户中，有多大比例的用户在未来 $t+n$ 天内使用至少一次搜索。搜索第 n 留的意思是今天使用搜索的用户中，有多大比例的用户在第 $t+n$ 天使用搜索。

留存指标都有个缺陷：真正地让用户变得更活跃，或者让不活跃的用户不再登录，那么留存指标都会增长。做一个假想实验：把留存率低的用户的账户全部关闭，能登录的用户都比较活跃，留存指标会增长，出现虚假的繁荣。反之，如果有策略能让流失的用户变成低活用户，原本是好事，反倒会拉低留存指标。如果发现留存指标增长，而 DAU 下跌，那么就应当分析是不是策略赶走了不活跃的用户；拆分人群分析的话，发现高活用户指标持平、而低活用户的 DAU 和留存下跌，则说明低活用户被赶走了。

留存指标不容易显著，而且具有一定的滞后性。对于用户点击指标，实验组和对照组只需要很小的流量就能测出具有统计显著性的差异（diff）。留存指标较难达到统计显著，需要较大的流量才能显著。对于用户点击指标，策略上线两三天、或是两三周，实验组和对照组的 diff 不会有很大区别，也就是说指标没有滞后性。而留存指标则不同，如果策略是有效的，上线之后留存指标可能会一直缓慢增长，需要观测较长的周期。

2.2 中间过程指标

我们希望设计出的策略能提升用户规模和留存指标，且实验测到的 diff 具有显著性。但这在实践中是比较困难的，很多有益的策略无法提升用户规模和留存，即便能提升，也需要很长的观测周期，或者提升不显著，这些会影响策略迭代的效率。工业界常使用一些中间过程指标，比如点击率、有点比、首点位置、交互率、换词率，它们很容易测量出显著的 diff，而且指标没有滞后性，对实验观测非常有利。这些中间过程指标的提升通常意味着用户体验的改善；根据我们长期观察的经验，如果中间过程指标显著提升，那么留存指标也会提升。当然这不是绝对的，有些“奇技淫巧”可以提升中间过程指标，但用户实际体验会变差。

2.2.1 点击率和有点比

搜索引擎返回的文档在搜索结果页上曝光给用户。在搜索结果页上，用户能看到首图、标题、摘要，根据这些信息决定是点击进入文档，或是选择下滑（或翻页）。APP 上的埋点会记录文档是否被曝光、点击、交互，这些数据被用来计算业务指标，也被用来训练召回和排序模型。

文档点击率等于系统总点击次数除以总曝光次数。举个例子，今天搜索引擎一共给用户曝光 50 亿篇文档（同一篇文档可以重复计数），其中有 5 亿篇文档被点击，那么文档点击率等于 $5/50 = 10\%$。

有点比等于有点击的搜索次数除以总搜索次数。用户做一次搜索之后，如果点击了结果页的某篇文档，则本次搜索算是有点击。有点比等于发生点击的搜索次数除以总的搜索次数。举个例子，今天搜索引擎一共做了 3 亿次搜索，其中 1.8 亿次搜索有点击，那么有点比等于 $1.8/3 = 60\%$。

首屏有点比等于有首屏点击的搜索次数除以总点击次数，它小于有点比。用户做一次搜索之后，如果点击了首屏的某篇笔记，则本次搜索算是有首屏点击。百度和谷歌的首屏有 10 个网页，小红书的首屏有 4 篇笔记，这些是用户最容易看到的文档，对用户体验影响最大，因此理应是最符合用户需求的文档。如果搜索引擎做得好，那么首屏有点比应该高。举个例子，今天搜索引擎一共做了 3 亿次搜索，其中 1.5 亿次搜索有首屏点击，那么首屏有点比等于 $1.5/3 = 50\%$。

上述指标满足关系“文档点击率 $\ll$ 首屏有点比 $<$ 有点比”。这几种指标具有很强的关联，有效的策略往往会同时提升这三种指标。此外，用户经常会手滑误点，为了排除误点对统计指标的干扰，我们使用“有效点击”代替“点击”。如果用户点击文档，且发生两种行为之一——在文档中停留时间超过某个阈值，或有点赞、收藏、转发、关注任意一种交互——则点击被视作有效点击。

点击率与有点比的缺点。如果一个策略提升了上述所有点击指标，但是用户规模和留存是持平的，那么这种策略是好的策略吗？其实是存疑的，原因可以用以下反例解释。

- 实践经验表明，把相关性控制得越紧，则点击指标越差。反之，如果放松相关性的控制，比如原本是分为 4 个相关性档位，现在换成 2 个相关性档位，那么点击指标

会大幅提升。我们发现，优化相关性模型，让模型对相关性的判断更准，这显然是有利于用户体验的，但几乎不会改善点击指标。我们还发现，对时效性的优化反而会损害点击指标。

- 提升点击指标可以用一些“奇技淫巧”，比如往搜索结果中插入推荐结果，即完全忽视查询词，只根据用户兴趣选择召回文档，而不控制查询词与文档的相关性。这种方法确实能提升点击指标，但未必利于用户体验。我们在实践中还发现，如果单纯以点击指标作为优化目标，那么模型倾向于给男性用户看封面图是性感美女的文档。我们曾经尝试用策略打压这样的搜索结果，后果是点击指标出现下跌。

2.2.2 首点位置和浏览深度

搜索引擎的排序模块应该让符合用户需求的文档的排名尽量靠前，让用户感到更方便、体验更好。在一次搜索中，用户点击排第 1 的文档、或是排第 4 的文档，对有点比、首屏有点比的贡献是同等的，但是前者说明排序的结果更优。用户在搜索结果页面上点击的第一篇文档的位置会被记录到日志中，关于当天所有的搜索求平均，得到的平均首点位置可以作为搜索的评价指标。首点位置与首屏点击率起到相似的作用，都鼓励搜索引擎把符合用户需求的文档排在前面。

首点位置的统计口径需要商榷，否则首点位置降低未必能反映用户体验的改善。如果在一次搜索中，用户没有点击搜索结果页上任何文档，是应当忽略这条样本，还是用默认的最大值作为本次搜索的首点位置？如果选择忽略没有点击的搜索，那么有一篇相关文档排在靠后位置（首点位置大），还不如搜索结果页上都是毫不相关的文档（没有点击）。是否应该设置一个阈值，当首点位置超出阈值时，取阈值作为首点位置，而非使用真实首点位置？如果不使用阈值，那么一次首点位置是几百的搜索，则会抵消掉几十次令用户满意的搜索。

用户在浏览搜索结果页时，会做下滑或翻页动作查看更多文档。浏览深度是指搜索结果页曝光给用户的最后一篇文档的位置，比如用户在百度网页端浏览完前 3 页，每页有 10 篇文档，那么他的浏览深度为 30。浏览深度变大其实是坏事，如果排在前面的文档已经能很好满足用户需求，那么用户没有必要多次下滑和翻页。谷歌、百度这样业界顶尖的搜索引擎经过长年优化，用户大多能在首页找到所需的文档，无需翻页。浏览深度与首点位置的原理类似，两个指标都是越小越好。

2.2.3 交互指标

用户在点击进入文档之后，APP 的埋点可以记录用户的交互行为，包括点赞、收藏、转发、关注、评论。此外，还能记录用户在文档上的停留时长；如果文档是视频，还可以记录播放时长、播放完成率，例如 100 秒的视频播放 40 秒，则播放完成率为 40%。交互和时长均可以反映用户对搜索结果的满意程度，这些信号比点击更强；美女首图、标题党可以骗到点击，但是骗不到交互和时长。如果搜索结果对用户很有帮助，用户会完成阅读或播放，会发生点赞等交互行为。

搜索结果页上的转化链路为“曝光 → 点击 → 阅读、交互”。每次搜索的交互次数，或每次点击的交互次数，都可以反映出用户对搜索结果的满意程度。举个例子，今天搜索引擎一共做了 3 亿次搜索，产生了 10 亿次点击，一共发生了 0.6 亿次点赞，那么平均每次搜索的点赞次数为 0.2，平均每次点击的点赞次数为 0.06，这两个数越高，说明用户对搜索结果越满意。

2.2.4 换词指标

主动换词率的作用与点击指标类似，不是搜索的核心指标，但可以在一定程度上反映出用户对搜索结果的满意程度。用户在一个 session 中，如果已经搜到满意的结果，通常不会换个相似的查询词再搜一次。换查询词，通常意味着没搜到想要的结果。举个例子，一个年轻男生搜“春季穿搭建议”，搜到的结果大多是不符合性别、不符合年龄的，那么他会换个词“男大学生春季穿搭”。之所以要强调“主动换词”，是因为搜索结果页面会有推荐的查询词，如果用户点击，就相当于换词，但这样的被动换词不能被视作搜索结果不好。如果一个策略降低主动换词率，通常意味着策略让用户体验变好。

搜索引擎的实验平台往往会记录用户平均查询词数量。举个例子，今天的搜索日活用户数为 SDAU = 5000 万，今天的总查询词数量（不做去重）为 3 亿，那么用户平均查询词数量为 6 条。好的策略会提升这个指标，还是降低这个指标？如果策略好，那么换词率会下降，会降低用户平均查询词数量。但策略好会让用户更倾向于用该搜索引擎，比如用户发现更容易在小红书上找到答案，那么用小红书会更多，用百度会更少，会让小红书用户平均查询词数量增加。总之，单独看用户平均查询词数量这一个指标，它的增加或降低，并不能说明策略是好还是差。

2.3 人工体验评估

假如一个策略没有提升用户规模和留存，但是提升了点击指标，这样的策略是否有价值呢？在实践中，有一些优化时效性、权威性、内容生态的策略，线上实验不但不提升任何指标，还损害了点击指标，这样的策略应当被允许全量上线吗？自家的搜索引擎跟竞争对手比，在哪些方面有优势和劣势？自家的搜索引擎跟自己上个月比，哪些方面的体验有提升或下降？要回答这些问题，就需要做人工体验评估。

2.3.1 人工评估的利弊

百度曾经在很长的一段时间中，以线上实验的有点比、首点位置、换词率等指标牵引搜索引擎的优化。后来高层发出一个疑问：所有线上实验的指标都在上涨，可我的亲身体验怎么越来越差了呢？近几年里，人工体验评估成为了百度搜索引擎迭代优化的牵引目标。每个策略上线之前都需要做人工体验评估，评估结果决定策略是否全量上线，而各种线上实验指标只供辅助决策。

如果以点击指标作为牵引目标，点击指标一定会增长，但是相关性、时效性、生态这些用户体验有可能会变差。举个例子，如果排序团队悄悄弱化一点相关性，那么点击指

标会上涨，但是用户体验可能会变差。以人工体验评估作为牵引目标的好处很明显：只要人工评估的方法得当且准确，那么搜索体验一定会越来越好。但是人工评估也有一些弊端，在下面详细讨论。

- 人工评估是主观的，评估质量很大程度上取决于评估人员的专业素质。对于一个待评估的算法策略，通常取几百条查询词的结果送给人工评估，这个量本身就不大，如果评估人员的专业素质不过硬，那么结果很难具有说服力。万一人工评估跟线上实验指标矛盾怎么？如果评估团队专业性一般，且测出的人工体验指标上涨，但是线上实验的点击指标下降，算法团队是否敢放心大胆让策略全流量上线？如果测出的人工体验指标下降，而各种线上指标都提升，算法团队是否会以专业性不足为理由要求重新评估呢？如果人工评估结果的随机性较大，那么多评几次总能遇到人工评估指标正向的情况，人工评估也就形同虚设了。
- 靠人工评估决定策略是否能推全，背后需要有一支规模较大的评估团队，公司需要付出很大的人工成本。同样的金钱成本，用来给相关性、内容质量、排序学习模型标注训练数据，或是用于给策略做人工评估，哪一种带来的回报更大呢？抛开钱的问题不谈，建设一支专业素质过硬的评估团队是需要时间和积累的，人的经验越丰富，那么评估的可靠性越高。如果评估团队薪资低、人员频繁流动，那么团队的专业性很难稳步提升。
- 人工评估会在一定程度上降低算法迭代升级的效率。给策略做线上小流量 A/B 测试非常容易，一两天就能得到较为准确的点击等指标。而人工评估的速度较慢，通常需要一两周时间，如果待评估的策略较多，那么还需要排队等待，非常影响效率。线上 A/B 测试的数量可多可少，没有任何限制；但是评估团队的规模在短时间内无法快速增长，新人需要长时间的培训才能具备较高的专业素质。总而言之，如果以人工评估作为策略全流量上线的决策依据，那么算法团队的进度会变慢。
- 网页版百度几乎没有个性化，即不论哪位用户搜，搜索结果页几乎一致。对这样的搜索引擎做人工评估是比较容易的，评估人员只需要判断文档是否符合查询词的需求，而不需要研究用户的画像、推测用户的意图。小红书这样的搜索引擎个性化很强，搜索结果千人千面，会给人工评估带来很大困难。举个例子，一个王者荣耀游戏的玩家搜索“狄仁杰”，出的结果大多是影视和历史、而非游戏角色，那么这样的结果是好是坏呢？如果搜索引擎没有个性化，那么评估人员只需要看查询词和文档，就能判断文档符合查询词意图。如果搜索引擎个性化很强，那么评估人员需要根据用户画像做判断，引入更多决策因素，导致评估的随机性变大、准确率降低。

2.3.2 side by side 评估

想要知道一个新策略对搜索结果的影响，需要做人工 side by side（SBS）评估，判断实验组和对照组谁的搜索结果更让评估人员满意。用下面的方法抽取人工评估的样本。首先要设置人工评估的查询词数量，比如 $n = 300$ 条查询词。然后根据曝光日志随机抽样查询词，需要覆盖头、中、尾部的查询词，公平对待搜索频次高、中、低的查询词。接下来，对于每条查询词，固定用户画像和场景，分别运行实验组和对照组策略，取各自

结果页上排名最高的前 k 篇文档。如果两个文档列表无差异，则丢弃该查询词。最终凑够 $n = 300$ 条查询词的搜索结果，交付人工评估。

评估人员看到的界面如图 2.1 所示。页面上方展示查询词、用户画像、场景信息（搜索发生的时间、用户所在地点），评估人员可以点击进入查看更详细的用户画像。页面下方是对照组和实验组的搜索结果页，左边为对照组，右边为实验组。评估人员可以看到每篇文档的首图、标题、正文、标签等信息，还可以点击进入查看文档详情。每一行还会显示实验组与对照组的差异，比如 1 号位置无差异（两篇文档相同），2 号位置实验组的文档排序上升 2 位（实验组排序第 2 的文档在对照组排序第 4），3 号位置有差异（实验组排序第 3 的文档没有出现在对照组）。评估人员对比两个搜索结果页，评价搜索结果页作为一个整体的优劣。对于整个搜索结果页，如果实验组胜出的文档数大于对照组，则为 good；如果持平，则为 same；如果落败，则为 bad。最终报告 GSB，举个例子，对 $n = 300$ 条查询词做评估，得出 G : S : B = 50 : 220 : 30，说明策略组优于对照组。

Session 信息：查询词、用户画像、时间、用户所在地点　　**人工评估：** ● G　○ S　○ B

排名	对照组	实验组	差异
1	首图、标题、正文、标签	首图、标题、正文、标签	无差异
2	首图、标题、正文、标签	首图、标题、正文、标签	排序上升2位
3	首图、标题、正文、标签	首图、标题、正文、标签	有差异

图 2.1: side by side 人工体验评估

2.3.3 月度评估

搜索算法团队的目标包括用户规模和留存的提升、点击等指标的提升、以及人工体验指标的提升。前两者可以通过 A/B 测试得出，而人工体验指标则需要定期做评估，比如月度评估。人工体验以 discounted cumulative gain（DCG）作为评价指标，如果策略迭代持续提升用户体验，那么 DCG 会逐月提升。

每个月随机从曝光日志中抽取数千次搜索的结果，每次搜索包含一个用户的画像 u，一条查询词 q，以及排名最高的 k 篇文档 $d_1, \cdots, d_k$ 的详情。抽样不是均匀的，而是要覆盖头、中、尾部查询词。k 的量级取决于用户平均下滑深度，如果用户普遍只看搜索结果页前十，那么只需要 $k = 10$；如果用户的下滑深度是好几十，那么 k 就需要设置得比较大。评估人员为每一个 (u, q, d_i) 三元组标注相关性、时效性、内容质量、个性化档位，或是直接给 (u, q, d_i) 打一个综合满意度分数。

在做完人工评估之后，计算每次搜索的 DCG 分数。每次搜索包含 $(u, q, d_1, \cdots, d_k)$，其中 d 的下标表示文档在搜索结果页的排名。设 $\text{score}(u, q, d_i)$ 为人工评估的分数，可以是相关性等专项分数，也可以是综合满意度分数。用 DCG 评价一次搜索（一个用户、一条查询词、多篇文档）的满意度：

$$\text{DCG@}k = \sum_{i=1}^{k} \frac{\text{score}(u, q, d_i)}{\log_2(i+1)}.$$

DCG 是人工评估分数的加权和，文档排名越靠前，则权重越大。也就是说，我们更关心排名靠前的文档是否让用户满意。人工评估数千次搜索，每次搜索有一个 DCG 分数，对数千个 DCG 分数取平均，作为最终的人工体验指标。

除了报告 DCG 分数，人工评估还能找到大量 bad case，即明显不符合预期的结果，比如相关性差、内容质量低、时效性差、不满足地域性、不符合用户性别。造成 bad case 的原因有很多，比如分词错误、相关性模型打分错误、QP 没有理解时效性和地域性诉求、点击率预估不准。经过专业搜索评估的负责人确认之后，将 bad case 给到相应的算法团队，用具体的 case 驱动算法的优化。bad case 解决率是搜索的一个业务指标，算法团队需要在一定期限内达到预设的目标，比如 3 个月内解决 40% 的 bad case。

上述内容属于公司对自己产品的自评，判断自己的搜索团队是否持续提升体验。提供搜索服务的公司都会私下评估竞争对手，判断自己在哪些方面有优势和劣势。大致方法是随机抽样数百、或数千查询词，在多个搜索引擎上做检索，并保存搜索结果页的前 k 篇文档。与 SBS 评估方法相同，人工分析相关性、时效性、内容质量这 3 个维度，给出相关性、时效性、内容质量、综合满意度 GSB 分数。通过与竞对的比较可以得知自己的优势与劣势是什么、差距有多大。

2.4 知识点小结

- 通用搜索引擎迭代优化的目标是提升用户体验，但是该用什么样的指标衡量用户体验的提升呢？为搜索引擎设定合理的评价指标至关重要，评价指标是什么，策略和产品的迭代优化方向就是什么。
- 用户规模指标（包括 DAU、WAU、MAU）与用户留存指标（包括次 n 留、第 n 留）通常作为通用搜索引擎的核心指标，核心指标的提升。规模和留存指标存在不足之处。第一，提升规模和留存指标很困难，只有少数策略能显著规模和留存指标。第二，规模和留存指标具有滞后性，无法在几天内就被准确观测。第三，如果规模和留存到达天花板，提升用户体验未必能提升规模和留存指标。
- 中间过程指标包括点击率、有点比、首点位置、交互率、换词率等等，它们可以弥补规模和留存指标存在的缺点。中间过程指标的提升通常意味着用户体验的改善，但这并非绝对的，因此点击率等指标只算中间过程指标、而不算核心指标。
- 在搜索引擎建设到相对成熟的阶段，在新策略全量上线之前需要做 SBS 人工体验评估，考察 GSB 指标，这样可以避免点击等指标提升，而用户真实体验下降。每个月抽取真实的搜索结果做人工体验评估，用 DCG 指标考察用户体验是否逐月改善。

此外，还可以取同一批查询词，用自己和竞对的搜索引擎做搜索，对结果做 GSB 评估，考察与竞对差距所在、以及差距大小。

第二部分

机器学习基础

第 3 章 机器学习任务

搜索引擎的 QP、召回、排序问题大多可以归纳到二分类、多分类、回归、排序这四类机器学习任务。本章 3.1 到 3.4 节分别介绍这四类任务，以及常用的模型和训练方法。

3.1 二分类任务

二分类任务是机器学习中最常见的一类任务，给定特征向量 $\boldsymbol{x}$，目标是预测真实标签 y。二分类的意思是标签的取值为 0（负样本）或 1（正样本）。以搜索引擎中的点击率预估为例，一条训练样本包括用户、查询词、文档，神经网络从样本中提取的特征向量记作 $\boldsymbol{x}$。如果用户实际点击文档，那么真实标签为 $y=1$，否则标签为 $y=0$。可以把点击率模型看做一个二分类器，它的目标是在文档尚未曝光给用户之前，预测用户有多大的概率会点击文档。

本节不讨论如何从训练样本中提取特征向量 $\boldsymbol{x}$，只讨论如何用一个简单的函数 $f(\boldsymbol{x};\boldsymbol{w},b)$ 基于 $\boldsymbol{x}$ 做二分类。sigmoid 分类器是实践中最常用的方法，定义为：

$$f\left(\boldsymbol{x};\boldsymbol{w},b\right) \triangleq \text{sigmoid}\left(\boldsymbol{x}^{\top}\boldsymbol{w}+b\right),$$

它将向量 $\boldsymbol{x}$ 映射到介于 0 和 1 之间的实数。此处的 sigmoid 是个激活函数（activation function），定义为：

$$\text{sigmoid}\left(z\right) \triangleq \frac{1}{1+\exp(-z)}.$$

如图 3.1 所示，sigmoid 可以把任何实数 z 映射到 0 和 1 之间。以点击率预估为例，设从样本中提取的特征向量为 $\boldsymbol{x}$，分类器的输出为 $\widehat{y}=f(\boldsymbol{x};\boldsymbol{w},b)$。可以将 $\widehat{y}$ 看做用户点击文档的概率；$\widehat{y}$ 越接近 1，说明模型认为用户点击文档的概率越大。

训练 sigmoid 分类器通常使用交叉熵（cross entropy，CE）作为损失函数。在具体讲解训练之前，先介绍交叉熵的定义。使用向量

$$\boldsymbol{p} = \left[p_1,\cdots,p_m\right]^T \quad \text{和} \quad \boldsymbol{q} = \left[q_1,\cdots,q_m\right]^T \tag{3.1}$$

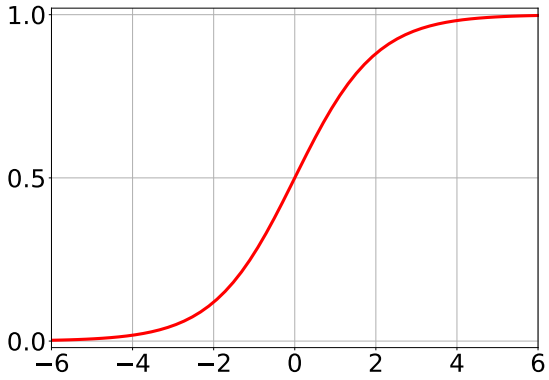

图 3.1: sigmoid 函数的图像

表示两个 m 维的离散概率分布。向量的元素都非负，且 $\sum_{j=1}^{m} p_j = 1$，$\sum_{j=1}^{m} q_j = 1$。它们之间的交叉熵定义为：

$$\mathrm{CE}\left(\boldsymbol{p}, \boldsymbol{q}\right) \ = \ -\sum_{j=1}^{m} p_j \cdot \ln q_j.$$

两个离散概率分布越相似，则它们的 CE 越小，这就是为什么常用交叉熵作为损失函数。

接下来讲解如何训练 sigmoid 分类器。收集 n 份样本，把特征向量记作 $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n$，把真实标签记作 $y_1, \cdots, y_n$。由于只有 0、1 两个类别，式 (3.1) 中的类别数量为 $m = 2$。设离散概率分布为：

$$\boldsymbol{p}_i \ = \ \begin{bmatrix} y_i \\ 1 - y_i \end{bmatrix} \qquad \text{和} \qquad \boldsymbol{q}_i \ = \ \begin{bmatrix} f(\boldsymbol{x}_i; \boldsymbol{w}, b) \\ 1 - f(\boldsymbol{x}_i; \boldsymbol{w}, b) \end{bmatrix}.$$

两个概率分布的交叉熵为：

$$\begin{aligned} \mathrm{CE}(\boldsymbol{p}_i, \boldsymbol{q}_i) \ &= \ \mathrm{CE}\left(\begin{bmatrix} y_i \\ 1 - y_i \end{bmatrix}, \begin{bmatrix} f(\boldsymbol{x}_i; \boldsymbol{w}, b) \\ 1 - f(\boldsymbol{x}_i; \boldsymbol{w}, b) \end{bmatrix} \right) \\ &= \ y_i \cdot \ln \left[f(\boldsymbol{x}_i; \boldsymbol{w}, b) \right] + (1 - y_i) \cdot \ln \left[1 - f(\boldsymbol{x}_i; \boldsymbol{w}, b) \right]. \end{aligned}$$

对于 n 个样本，总的损失函数为 $\frac{1}{n} \sum_{i=1}^{n} \mathrm{CE}(\boldsymbol{p}_i, \boldsymbol{q}_i)$，用随机梯度下降等算法更新参数 $\boldsymbol{w}$、b。

3.2 多分类任务

多分类任务的设定是类似的，给定特征向量 $\boldsymbol{x}$，目标是预测真实标签 y。与二分类的区别在于 y 的取值范围不是 $\{0, 1\}$，而是 $\{0, 1, \cdots, k-1\}$，这里的 k 是类别数量。本节介绍 softmax 分类器，它是解决多分类问题最常用的方法。

本小节用 MNIST 手写数字识别为例讲解多分类问题。如图 3.2 所示，MNIST 数据集有 $n = 60,000$ 个样本，每个样本是 28×28 像素的图片。数据集有 $k = 10$ 个类别，每个样本有一个类别标签，它是介于 0 到 9 之间的整数，表示图片中的数字。为了训练 softmax 分类器，我们要对标签 y 做 one-hot 编码，把每个标签（0 到 9 之间的整数）映射到 $k = 10$ 维的向量：

图 3.2: MNIST 数据集中的图片。

$$\begin{aligned} 0 \quad &\Longrightarrow \quad [1, 0, 0, 0, 0, 0, 0, 0, 0, 0], \\ 1 \quad &\Longrightarrow \quad [0, 1, 0, 0, 0, 0, 0, 0, 0, 0], \\ &\vdots \\ 8 \quad &\Longrightarrow \quad [0, 0, 0, 0, 0, 0, 0, 0, 1, 0], \\ 9 \quad &\Longrightarrow \quad [0, 0, 0, 0, 0, 0, 0, 0, 0, 1]. \end{aligned}$$

对 y 做 one-hot 编码得到的向量记作粗体符号 $\boldsymbol{y}_1, \cdots, \boldsymbol{y}_n \in \{0,1\}^{10}$。对于每张 28×28 的图片，可以简单地拉伸成 768 维的向量，也可以用神经网络提取特征向量。把从 n 张图片中提取的特征向量记作 $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n$。

在介绍 softmax 分类器之前，先介绍 softmax 激活函数。它的输入和输出都是 k 维向量。设 $\boldsymbol{z} = [z_1, \cdots, z_k]^\top$ 是任意 k 维实向量，它的元素可正可负。softmax 函数定义为

$$\text{softmax}\left(\boldsymbol{z}\right) \triangleq \frac{1}{\sum_{l=1}^{k} \exp(z_l)} \left[\exp\left(z_1\right), \exp\left(z_2\right), \cdots, \exp\left(z_k\right) \right]^\top$$

函数的输出是一个 k 维向量，元素都是非负，且相加等于 1。

如图 3.3 所示，softmax 函数让最大的元素相对变得更大，让小的元素接近 0。图 3.4 是 max 函数，它把最大的元素映射到 1，其余所有元素映射到 0。对比一下图 3.3 和图 3.4，不难看出 softmax 没有让小的元素严格等于零，这就是为什么它的名字带有“soft”。

图 3.3: softmax 函数把左边红色的 10 个数值映射到右边紫色的 10 个数值

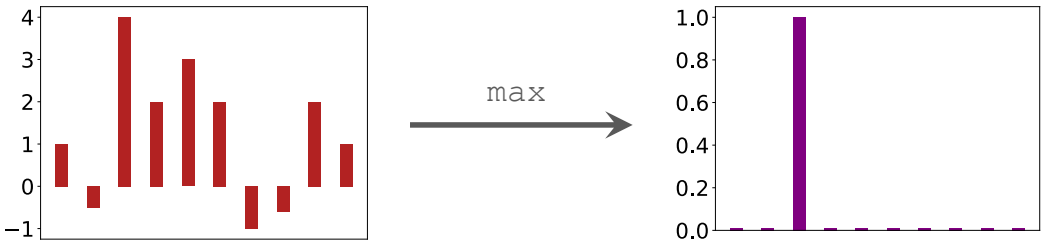

图 3.4: max 函数把左边红色的 10 个数值映射到右边紫色的 10 个数值

softmax 分类器是线性函数与 softmax 激活函数的组合，如图 3.5 所示。具体来说，线性 softmax 分类器定义为

$$\boldsymbol{\pi} = \text{softmax}(\boldsymbol{z}), \qquad \text{其中} \qquad \boldsymbol{z} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}.$$

设 $\boldsymbol{x}_i$ 与 $\boldsymbol{y}_i$ 分别是 d 维和 k 维向量。分类器的参数是矩阵 $\boldsymbol{W} \in \mathbb{R}^{k \times d}$ 和向量 $\boldsymbol{b} \in \mathbb{R}^k$。

softmax 分类器输出向量 $\boldsymbol{\pi}$ 第 j 个元素 π_j 表示输入向量 $\boldsymbol{x}$ 属于第 j 类的概率。在以上 MNIST 手写数字识别的例子中，假设分类器的输出是以下 10 维向量：

$$\boldsymbol{\pi} = \left[0.1,\, 0.6,\, 0.02,\, 0.01,\, 0.01,\, 0.2,\, 0.01,\, 0.03,\, 0.01,\, 0.01\right]^T.$$

可以这样理解该向量的元素：

- 第 0 号元素 0.1 表示分类器以 0.1 的概率判定图片 $\boldsymbol{x}$ 是数字“0”，
- 第 1 号元素 0.6 表示分类器以 0.6 的概率判定 $\boldsymbol{x}$ 是数字“1”，
- 第 2 号元素 0.02 表示分类器只有 0.02 的概率判定 $\boldsymbol{x}$ 是数字“2”，

以此类推。由于分类器的输出向量 $\boldsymbol{\pi}$ 的第 1 号元素 0.6 是最大的，分类器会判定图片 $\boldsymbol{x}$ 是数字“1”。

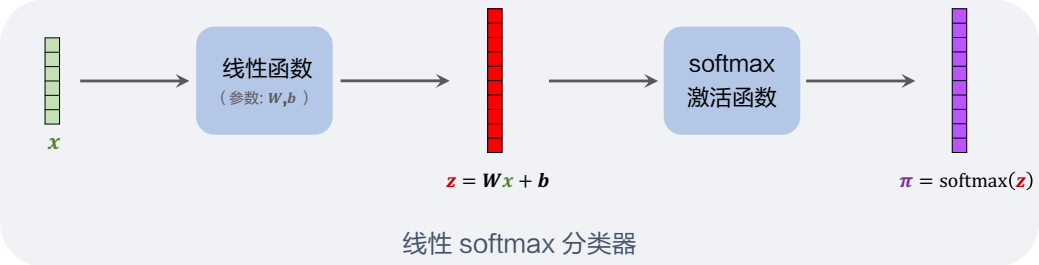

图 3.5: 线性 softmax 分类器的结构如图所示，输入是向量 $\boldsymbol{x} \in \mathbb{R}^d$，输出是 $\boldsymbol{\pi} \in \mathbb{R}^k$。

接下来讲解如何训练 sigmoid 分类器。收集 n 份样本，把特征向量记作 $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n$，把真实标签的 one-hot 向量记作 $\boldsymbol{y}_1, \cdots, \boldsymbol{y}_n$。one-hot 向量 $\boldsymbol{y}_i$ 和模型输出的 $\boldsymbol{\pi}_i$ 都是 k 个类别上的离散概率分布，两个概率分布的交叉熵为：

$$\mathrm{CE}(\boldsymbol{y}_i,\, \boldsymbol{\pi}_i) \;=\; \sum_{j=1}^{k} y_{i,j} \cdot \ln \pi_{i,j},$$

式中的 $y_{i,j}$ 是 $\boldsymbol{y}_i$ 的第 j 个元素，$\pi_{i,j}$ 是 π_i 的第 j 个元素。对于 n 个样本，总的损失函数为 $\frac{1}{n}\sum_{i=1}^{n} \mathrm{CE}(\boldsymbol{y}_i, \boldsymbol{\pi}_i)$，用随机梯度下降等算法更新参数 $\boldsymbol{W}$、$\boldsymbol{b}$。

3.3 回归任务

回归是另一类机器学习任务，在搜索引擎中也有很广泛的应用。给定特征向量 $\boldsymbol{x}$，目标是预测标签 y。回归与多分类貌似相似，它们的区别在于 y 是否是有序的，即是否可以比较大小。举个例子，搜索中 (q, d) 的相关性分为 5 档，标签为 $y \in \{0, 1, 2, 3, 4\}$，标签是有序的，4 是最高档位，0 是最低档位，因此相关性属于回归任务。另外一个例子是上一节的手写数字识别，标签为 $y \in \{0, 1, \cdots, 9\}$，这些标签只是类别而已，不可比较大小，因此手写数字识别属于多分类任务。

我们用一个简单的函数 $f(\boldsymbol{x}; \boldsymbol{w}, b)$ 拟合 y，其中 $\boldsymbol{w}$ 和 b 是模型参数。首先介绍线性模型，它假设真实标签 y 为 $\boldsymbol{x}$ 的线性函数。因此定义函数 f 为 $\boldsymbol{x}$ 的线性函数：

$$f(\boldsymbol{x}; \boldsymbol{w}, b) \;=\; \boldsymbol{x}^\top \boldsymbol{w} + b.$$

通常用均方误差（mean squared error，MSE）作为损失函数，它等于 $f(\boldsymbol{x}; \boldsymbol{w}, b)$ 与 y 的差的平方。设一共有 n 个样本，把它们的特征向量记作 $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n$，标签记作 $y_1, \cdots, y_n$，那么总的损失函数为：

$$\frac{1}{n}\sum_{i=1}^{n} \big[\, f(\boldsymbol{x}_i; \boldsymbol{w}, b) \;-\; y_i \,\big]^2 .$$

上式中的模型叫做最小二乘回归。用随机梯度下降等算法更新 $\boldsymbol{w}$ 和 b，可以减小损失函数，即让函数的预测更接近真实标签。

之前介绍的 sigmoid 分类器 $f(\boldsymbol{x}; \boldsymbol{w}, b) = \mathrm{sigmoid}(\boldsymbol{x}^\top \boldsymbol{w} + b)$ 也可以用于回归任务。sigmoid 将任意实数映射到 $(0, 1)$ 区间上，因此我们需要对 y 做归一化，将其也映射到

$[0,1]$ 的区间上。例如搜索中的相关性问题，将档位 $y \in \{0,\cdots,4\}$ 除以 4，压缩到 $[0,1]$ 区间上。然后用二分类的交叉熵作为损失函数，最小化交叉熵学习分类器的参数 $\boldsymbol{w}$ 和 b。

既然线性模型和 sigmoid 分类器都可用于回归问题，那么实践中该用哪种方法呢？其实没有特别明确的答案，可以把两种方法都试一试。以搜索相关性为例，如果我们的目标是准确预测相关性分数，让 MSE 误差尽量小，应该使用线性模型。如果我们的目标是在召回海选阶段过滤掉分数低的文档，那么我们并不在乎预测得有多准，而是能区分相关、不相关的文档，那么用 sigmoid 分类器更好。

3.4 排序任务

排序是搜索引擎中常见的一类机器学习任务。每个 batch 中有 m 个样本，把它们的特征向量记作 $\boldsymbol{x}_1,\cdots,\boldsymbol{x}_m$，标签记作 $y_1,\cdots,y_m$。把特征向量输入任意模型 f，模型打分 $f(\boldsymbol{x}_1;\boldsymbol{w}),\cdots,f(\boldsymbol{x}_m;\boldsymbol{w})$，其中 $\boldsymbol{w}$ 是模型的参数。模型的目标不是让打分 $f(\boldsymbol{x}_i;\boldsymbol{w})$ 准确拟合 y_i，而是让模型打分拟合 $y_1,\cdots,y_m$ 的序。我们不妨假设 $y_1 \geq \cdots \geq y_m$，即 m 个样本按降序排列。理想的模型满足

$$f(\boldsymbol{x}_1;\boldsymbol{w}) \;\geq\; \cdots \;\geq\; f(\boldsymbol{x}_m;\boldsymbol{w}),$$

即模型打分与 $y_1 \geq \cdots \geq y_m$ 的序一致。

对于 i 和 j，设标签满足 $y_i \geq y_j$。把模型的打分记作 $p_i = f(\boldsymbol{x}_i;\boldsymbol{w})$。如果 $p_i \geq p_j$，则 (i,j) 被称作一个正序对；如果 $p_i < p_j$，则 (i,j) 被称作一个逆序对。排序的目标是最大化正序对的数量，等价于最小化逆序对的数量。

为了达到最大化正序对数量的目的，对于所有满足 $y_i > y_j$ 的样本对 (i,j)，鼓励 $p_i - p_j$ 尽量大。通常使用下面列出的 pairwise logistic loss：

$$L(\boldsymbol{w}) \;=\; \frac{1}{m^2} \sum_{(i,j):y_i>y_j} \underbrace{\ln\big[1+\exp\big(-(p_i-p_j)\big)\big]}_{\triangleq\, l_{ij}}.$$

$p_i - p_j$ 越大，则损失 l_{ij} 越小。做训练时，对 $L(\boldsymbol{w})$ 关于 $\boldsymbol{w}$ 求梯度，做梯度下降更新 $\boldsymbol{w}$，损失函数会减小。

3.5 知识点小结

- 二分类任务常用 sigmoid 分类器，它的输出介于 0 和 1 之间的一个实数 p，意思是模型认为样本为正的概率是 p。
- 分类任务常用 softmax 分类器，设有 k 个类别，分类器输出 k 个概率值 $\boldsymbol{p} = [p_0,\cdots,p_{k-1}$，意思是模型认为样本属于第 i 类的概率是 p_i。
- 回归问题常用线性函数或 sigmoid 分类器，它们的输出都是实数 $\widehat{y}$，目标是让 $\widehat{y}$ 尽可能接近真实标签 y。
- 排序任务常用线性函数或 sigmoid 分类器，但是训练方式不同于回归和二分类。排序常用 pairwise logistic loss 作为损失函数，目的是让正序对数量尽量大，逆序对数量尽量小。

- 回归与多分类的本质区别在于标签 y 是否是有序的，即两个标签是否可以比较大小。回归与排序的本质区别在于我们在乎的是“值”还是“序”，回归的目标是“保值”，排序的目标是“保序”。

第 4 章　离线评价指标

当算法工程师改进了 QP、召回、排序的模型之后，该如何评价新模型是否优于当前模型？我们在 2 章中讨论了最重要的评价指标，这些指标有提升，则可以说明新策略优于旧策略。获得这些指标的前提是将模型部署在线上，要么用 A/B 测试计算新旧模型线上指标的差异，要么用人工评估两种模型的搜索结果页的 GSB。将模型部署在线上做 A/B 测试相对比较复杂，而且会对用户产生影响，因此在上线之前需要做离线评估。

离线评估的方式与机器学习标准的测试方法无异，就是取一批独立于训练集的样本 $\{(\boldsymbol{x}', y')\}$ 作为测试集，计算模型在测试集上的各种指标。本章以搜索相关性为例讲解离线评价指标。回顾一下，相关性是指查询词 q 与文档 d 的相关性，每对二元组 (q, d) 都有人工标注的档位，分为高、中、低、无这 4 档。离线评价指标可以分为三类，即 pointwise、pairwise、listwise，本章的三节分别讲解这三种评价指标。

4.1　pointwise 评价指标

pointwise 评价的意思是将测试集里的每对 (q, d) 看做独立的样本，判断模型给每对样本的打分是否正确；这种评价方式不考虑样本之间的关系。有三种 pointwise 评估方法。第一种是回归的评价指标，即考察模型估计值 $\hat{y}$ 是否足够接近真实值 y。通常用均方误差（MSE）作为回归的评价指标，本节不再介绍。第二种是二分类的评价指标，即合并高和中两档、合并低和无两档，将相关性视作二分类问题，区分相关与无关，以便过滤掉不相关的文档。第三种是多分类的评价指标，即将高、中、低、无视作 4 个类别，不考虑它们的序。

4.1.1　二分类的评价指标

我们可以将相关性建模为二分类问题，即标签为相关或无关。模型输出一个介于 0 和 1 之间的分数，0 表示不相关，1 表示相关。设置一个阈值 τ，模型输出的分数大于 τ 意味着“模型认为相关”，否则“模型认为无关”。

下面用图 4.1 中的例子解释二分类的评价指标。一共有 100 个样本，其中有 40 个被标注为相关，有 60 个被标注为无关，我们认为人工标注是真实标签。模型不知道人工标注，只根据 q 和 d 的文本预测相关性。对于 40 个标注为相关的样本（正样本），模型认为其中有 30 个相关，10 个无关；实际相关、且预测为相关，称为 true positive（TP）；实际相关、但预测为无关，称为 false negative（FN）。对于 60 个标注为无关的样本（负样本），模型认为其中有 20 个相关，40 个无关；实际无关、但预测为相关，称为 false positive（FP）；实际无关、且预测为无关，称为 true negative（TN）。

图 4.1 的例子中，模型认定一共有 $30 + 20 = 50$ 个样本为相关，其中有 30 个是真正相关的，准确率为

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{30}{30 + 20} = 0.6.$$

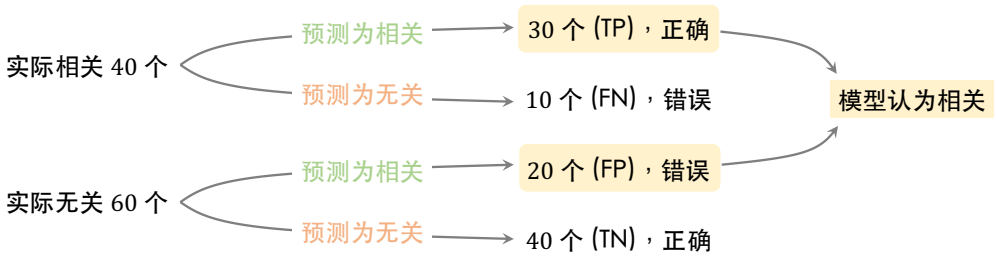

图 4.1: 可以将召回看做二分类问题

实际上一共有 40 个样本为相关，模型只找到其中 30 个，召回率为

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{30}{30 + 10} = 0.75.$$

取准确率和召回率的调和平均数，得到 F1 分数：

$$\text{F1} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} = 2 \times \frac{0.6 \times 0.75}{0.6 + 0.75} = \frac{2}{3}.$$

准确率和召回率两个指标此消彼长。把分类的阈值 τ 调小一些，则模型判定相关的标准更松，模型会将更多样本判定为相关，那么准确率会降低，召回率会增加。反之，如果把 τ 调得更大，则准确率会增加，召回率会降低。我们希望测得的指标只受模型的影响，然而准确率、召回率、F1 都受 τ 的影响。

为了排除 τ 的影响，二分类更常使用 area under curve（AUC）指标。定义真阳性率为 $\text{TPR} = \frac{\text{TP}}{\text{TP+FN}}$，假阳性率为 $\text{FPR} = \frac{\text{FP}}{\text{FP+TN}}$。增大阈值 τ，TPR 和 FPR 都会减小；减小 τ，TPR 和 FPR 都会增大。让 τ 从 0 增长到 1，得到若干对 TPR 和 FPR，以 TPR 作为纵轴，以 FPR 作为横轴，得到的曲线叫作 receiver operating characteristic（ROC）曲线。图 4.2 中画了 3 条 ROC 曲线，曲线越靠上说明模型越准确。曲线与 x 轴之间区域的面积就是 AUC，大小介于 0 和 1 之间，AUC 越大说明模型的预测越准确。如果模型做随机猜测，输出 $[0, 1]$ 区间上随机采样的数值，那么模型的 ROC 曲线是图 4.2 中的对角线，它的 AUC 等于 0.5。如果一个模型的 AUC 在 0.5 上下，说明模型没有泛化能力，只是在测试集上做随机猜测。

图 4.3 是分类结果的混淆矩阵（confusion matrix），记录了 TP、FN、FP、TN，它起到可视化的作用，从中可以看出模型更容易犯什么样的错误。值得注意的是，混淆矩阵只是一种分析工具，而非分类的评价指标，分类应当采用 F1 和 AUC 作为评价指标。二分类的混淆矩阵起到的作用有限，而多分类则非常需要混淆矩阵。

4.1.2 多分类的评价指标

以搜索相关性为例，人工标注 4 个档位，档位之间可以比大小，因此相关性属于回归任务。然而实践中也常将 4 个档位视为 4 个类别，用多分类的评价指标。如果相关性模型是 softmax 分类器，那么模型输出是 4 个档位中的一个。如果相关性模型输出一个实数，那么需要人为设定 3 个阈值，把分数映射到档位。

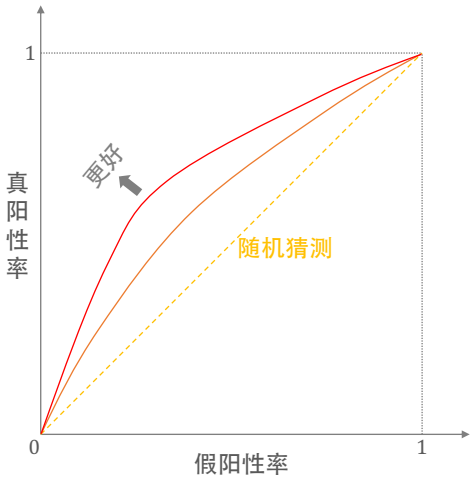

图 4.2: 图中画了 3 条 ROC 曲线，曲线越靠上说明模型越准确

	预测相关 (positive)	预测无关 (negative)
实际相关 (positive)	30 (TP，正确)	10 (FN，错误)
实际无关 (negative)	20 (FP，错误)	40 (TP，正确)

图 4.3: 二分类的混淆矩阵

多分类最常用 macro F1 和 micro F1 作为评价指标，本书用图 4.4 中的例子讲解两种指标。一共有 51 条样本，被人工标注为高、中、低、无的样本数量分别为 20、17、11、3，各对混淆矩阵的一行。混淆矩阵的每一列对应模型预测的一个档位，矩阵对角线元素为预测正确的样本数量。

	预测高	预测中	预测低	预测无	TP	FN	recall
实际高	15	5	0	0	15	5	0.75
实际中	8	8	0	1	8	9	0.47
实际低	5	2	4	0	4	7	0.36
实际无	2	0	0	1	1	2	0.33
TP	15	8	4	1			
FP	15	7	0	1			
precision	0.50	0.53	1.0	0.50			

图 4.4: 多分类的混淆矩阵

独立对待每一类，单独计算每一类自己的二分类 F1，再取 4 个 F1 的均值，得到的均值叫做 macro F1。如图 4.4 所示，混淆矩阵第一列为模型预测高档的样本数量，其中有 $\mathrm{TP}=15$ 个样本是真实高档，其余 $\mathrm{FP}=8+5+2=15$ 个样本并不是高档，这一列对应的准确率为

$$\text{precision} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}} = \frac{15}{15+15} = 0.5.$$

同样的方法可以计算出其余各列的 TP、FP、准确率。混淆矩阵第一行是真实高档的样本数量，其中 $\mathrm{TP}=15$ 个样本被正确预测为高档，其余 $\mathrm{FN}=5+0+0$ 个高档样本被错误

预测为其他档位，这一列对应的召回率为

$$\text{recall} \;=\; \frac{\text{TP}}{\text{TP}+\text{FN}} \;=\; \frac{15}{15+5} \;=\; 0.75.$$

同样的方法可以计算出其余各行的 TP、FP、召回率。取高档位的准确率和召回率的调和平均数，得到高档位的 F1 分数：

$$\text{F1} \;=\; 2\times\frac{\text{precision}\times\text{recall}}{\text{precision}+\text{recall}} \;=\; 2\times\frac{0.5\times 0.75}{0.5+0.75} \;=\; 0.6.$$

用同样的方法，可以计算其余各列的准确率、各行的召回率，取调和平均数得到每个档位的 F1 分数。中、低、无档位的 F1 分数分别是

$$2\times\frac{0.53\times 0.47}{0.53+0.47} \;=\; 0.50, \qquad 2\times\frac{1.0\times 0.36}{1.0+0.36} \;=\; 0.53, \qquad 2\times\frac{0.5\times 0.33}{0.5+0.33} \;=\; 0.40.$$

对各 F1 分数取平均，得到

$$\text{macro F1} \;=\; \frac{1}{4}\big(0.6+0.5+0.53+0.4\big) \;=\; 0.51.$$

实际的数据常存在类别不平衡问题，即有的类样本多、有的类样本少。macro F1 受各类的影响是同等的，不会被大的类别主导。

micro F1 的计算方式更简单。混淆矩阵对角线元素之和为 $15+8+4+1=28$，混淆矩阵所有元素加和（即总样本数）为 51，两者的比值 $\frac{28}{51}$ 就是 micro F1。如果类别不平衡，那么 micro F1 会被大的类别主导，而小的类别对 micro F1 的影响很小。

4.2 pairwise 评价指标

本节介绍正逆序比（positive-negative ratio，PNR），它是排序的一种评价指标。给定一条查询词 q 和 k 篇文档 $d_1,\cdots,d_k$，文档按照模型预估的相关性分数排列，即 $\text{rel}(q,d_1)\geq\cdots\geq\text{rel}(q,d_k)$。此外，文档有人工标注的相关性档位或分数，记作 $y_1,\cdots,y_k$，将其视作真实标签。最理想的情况为 $y_1\geq\cdots\geq y_k$，即模型的排序与真实的序一致。

对于两篇文档 (d_i,d_j)，设 $\text{rel}(q,d_i)\geq\text{rel}(q,d_j)$。如果 $y_i>y_j$，则 (d_i,d_j) 被称作正序对；如果 $y_i<y_j$，则 (d_i,d_j) 被称作逆序对。正逆序比是这样定义的：

$$\text{正逆序比} \;=\; \frac{\text{正序对数量}}{\text{逆序对数量}} \;\in\; (0,+\infty).$$

正逆序比越大，说明模型的预测越准确。

图 4.5 举两个例子说明正序对和逆序对的含义。一共有 $k=6$ 篇文档，$\binom{6}{2}=15$ 个文档二元组，正序对与逆序对的数量相加等于 15。左右两边使用不同的相关性模型，打分 $\text{rel}(q,d_i)$ 有差异，导致左右两边排序不同。左右两边各有 13 个正序对和 2 个逆序对，正逆序比都等于 $\frac{13}{2}$。

正逆序比是一种 pairwise 指标，主要用于考察排序的效果。pairwise 指标与 pointwise 指标有本质的差别。pointwise 指标考察模型预测值 $\text{rel}(q,d_i)$ 对真实相关性 y_i 拟合的是否足够好，而 pairwise 指标考察排序的序是否正确。举个例子，将模型打分 $\text{rel}(q,d_i)$ 替换成 $\sqrt{\text{rel}(q,d_i)}$，那么均方误差会严重变差，但正逆序比不会变化。

如果离线在一个事先标注好的测试集上评估相关性模型，通常会同时用 pointwise 和 pairwise 两类指标。但如果是对线上搜索排序结果做人工满意度评估（见 2.3 节），则使

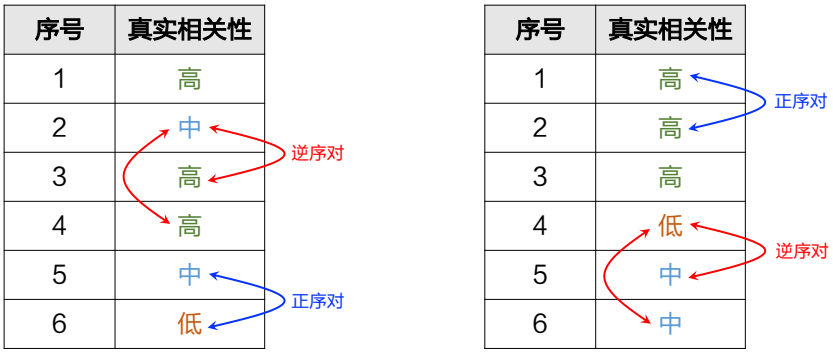

图 4.5: 正序对和逆序对的示意图

用 listwise 指标，而不用 pointwise 和 pairwise 指标，原因如下所述。图 4.5 的例子中，左右两种排序有相同的正逆序比，但我们认为右边的排序结果更好。左边的逆序对出现在 $(2,3)$ 和 $(2,4)$ 位置，排名靠前；而右边的逆序对出现在 $(4,5)$ 和 $(4,6)$ 位置，排名靠后。工业界的共识是用户更在意排名靠前的文档，排名靠前的文档中有 bad case 会非常乍眼；用户对排名靠后的 bad case 有较高的容忍度，甚至不会翻到后面的结果。因此，在做人工满意度评估时，会根据文档所在位置给满意度分数加权。

4.3 listwise 评价指标

给定一条查询词 q，召回若干篇文档，搜索引擎按照模型预测的相关性分数对文档做降序排列，把文档列表中的前 k 篇记作 $d_1, \cdots, d_k$。把人工标注的相关性档位映射成分数 $y_1, \cdots, y_k \in [0,1]$。最理想的情况是所有 k 篇文档均为高相关，即 $y_1 = \cdots y_k = 1$；最差的情况是所有 k 篇文档均为不相关，即 $y_1 = \cdots y_k = 0$。

DCG@k 是评价搜索相关性最常用的指标之一，DCG@k 越大意味着搜索结果的相关性越好。DCG@k 的定义为相关性（或其函数变换）的加权和：

$$\text{DCG@}k = \sum_{i=1}^{k} \frac{y_i}{\log_2(i+1)} \quad \text{或} \quad \text{DCG@}k = \sum_{i=1}^{k} \frac{2^{y_i}-1}{\log_2(i+1)}.$$

设查询词 q 召回 n（$\geq k$）篇文档，按照模型预测的相关性做排序，得到 $d_1, \cdots, d_n$。什么情况下 DCG@k 最大化？直观理解，排在前 k 的文档是真实相关性分数最高的，且模型给前 k 篇的排序与真实分数的序一致，则 DCG@k 最大化。也就是说，最理想的情况为 $y_1 \geq \cdots \geq y_k$ 且 y_k 大于等于所有 $y_{k+1}, \cdots, y_n$。这种最理想情况下的 DCG@k 叫做 ideal DCG@k，缩写 IDCG@k，它是 DCG@k 的上界。

NDCG@k 与 DCG@k 都是评价排序最常用的指标。NDCG@k 的值介于 0 和 1 之间，完美的排序系统的 NDCG@k 等于 1。NDCG@k 是这样定义的：

$$\text{NDCG@}k = \frac{\text{DCG@}k}{\text{IDCG@}k}.$$

计算 NDCG 的时候需要注意实验组和对照组 IDCG@k 的一致性问题，如果两组用的 IDCG@k 不一致，则可能得出与事实不符的结论。举个例子，实验组和对照组各有 10 篇

文档，让人工给文档标注相关性。实验组前 9 篇相关性低（$\text{rel}_1 = \cdots = \text{rel}_9 = 0$），第 10 篇相关性高（$\text{rel}_{10} = 1$）。对照组的 10 篇文档全都是低相关性。如果独立计算两组的 NDCG@10，会发现对照组的 NDCG@10 更高，这显然不符合事实。正确的做法是取两组的文档的并集，然后计算 IDCG@10，保证两组使用的 IDCG@10 相同。这样的话，实验组的 NDCG@10 更高。

4.4 知识点小结

- 模型升级迭代的过程中，需要离线计算模型在测试集上的指标，在观测到离线指标的增长之后才会开线上 A/B 测试。搜索引擎常用 pointwise、pairwise、listwise 评价指标。
- pointwise 的意思是独立看待每个 $(\boldsymbol{x}_i, y_i)$ 样本，不考虑样本之间的关系。pointwise 指标又分为回归、二分类、多分类评价指标。回归通常用均方误差。二分类常用准确率、召回率、F1、AUC 作为评价指标。
- pairwise 的意思是对样本两两组合，判断两个样本是正序对、还是逆序对。常用正逆序比作为作为 pairwise 评价指标。
- listwise 的意思是取一组 k 个样本，根据真实标签、模型排序来打分。常用 DCG@k 和 NDCG@k 作为 listwise 评价指标。

第 5 章 NLP 模型的训练

搜索引擎中有大量的任务需要用到 BERT 模型，比如 QP（分词、命名实体识别、类目识别）、相关性、内容质量、等等。它们的训练流程非常相似，通常包括 4 个步骤——预训练（pretrain）、后预训练（post pretrain）、精调（fine tune）、蒸馏（distill）。预训练是一个公共服务，训练出的模型供下游所有任务使用。后预训练、精调是针对特定任务做的，需要挖掘和标注任务相关的数据。蒸馏是一种通用技术，在不增加线上推理的机器成本的前提下提升准确率。

5.1 预训练任务

BERT 的原始论文提出 masked language model（MLM）与 next sentence prediction（NSP）两个任务。之后 RoBERTa 研究发现 NSP 过于简单，对预训练起到负面作用。后来的 ALBERT 论文提出 sentence order prediction（SOP）任务，目前已经被广泛接受。现在大家公认 MLP 与 SOP 是预训练 BERT 模型必不可少的两个任务，其他任务的作用小于 MLM 与 SOP。

MLM 的意思是遮住句子中若干个字（比如随机选 15% 的字遮住），要求模型预测遮住的字。如图 5.1 所示，输入的句子有 10 个字，随机选中 t_3 与 t_8，将它们遮住。BERT 模型输出 10 个向量 $\boldsymbol{c}_1, \cdots, \boldsymbol{c}_{10}$，我们用遮挡位置上的向量 $\boldsymbol{c}_3$ 与 $\boldsymbol{c}_8$ 预测遮挡的字 t_3 与 t_8。把 $\boldsymbol{c}_3$ 与 $\boldsymbol{c}_8$ 输入 softmax 分类器，把分类器输出记作 $\boldsymbol{p}_3$ 与 $\boldsymbol{p}_8$。如果字典中一共有 d 个字，那么 $\boldsymbol{p}_3$ 与 $\boldsymbol{p}_8$ 都是 d 维向量，每个元素是一个字的概率。把字 t_3 与 t_8 的 one-hot 编码记作 d 维向量 $\boldsymbol{y}_3$ 与 $\boldsymbol{y}_8$，使用交叉熵作为损失函数：

$$\mathrm{CE}(\boldsymbol{y}_3, \boldsymbol{p}_3) \ + \ \mathrm{CE}(\boldsymbol{y}_8, \boldsymbol{p}_8).$$

最小化交叉熵损失，可以让模型更好地预测遮住的字。

NSP 的意思是将两句话拼起来输入 BERT 模型，要求模型判断两句话是原文中真实相邻的两个句子、或是两个无关的句子被拼接起来。举个例子，将两个句子以下面的形式输入 BERT 模型：

“[CLS] 微积分是数学的一个分支 [SEP] 它由牛顿和莱布尼茨发明 [SEP]”。

其中“[CLS]”是分类符，“[SEP]”是分隔符，BERT 中的 embedding 层把它们当做普通的字符对待；你完全可以用“@”代替“[CLS]”，用“#”代替“[SEP]”，不会影响训练。对于上面两个句子，如果 BERT 模型理解牛顿和莱布尼茨是数学家，那么就能判断两句话大概率是真实相邻的。而对于下面两个句子，BERT 模型应当判断为不相邻：

“[CLS] 微积分是数学的一个分支 [SEP] 辣椒原产于拉丁美洲 [SEP]”。

对于 NSP 任务，我们用图 5.2 中的模型结构，把“[CLS]”与“[SEP]”当作普通的字对待。BERT 模型输出多个向量，NSP 任务只用 [CLS] 位置上的向量，把它输入 sigmoid 分类器，分类器输出实数 $p \in (0, 1)$。如果两句话真实相邻，则设标签 $y = 1$；反之，则设 $y = 0$。

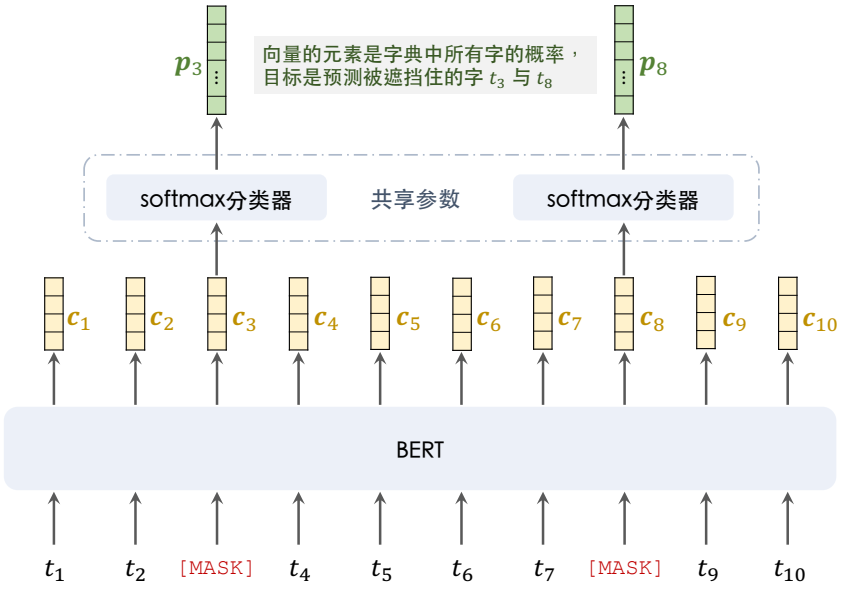

图 5.1: MLM 预训练方法的示意图

用 y 与 p 的交叉熵作为损失函数，让 p 拟合 y。

SOP 与 NSP 非常类似，但是 SOP 更困难，实践证明 SOP 效果优于 NSP。SOP 的正样本是文章中真实的两句话，负样本是将两句话的顺序颠倒。下面例子中，前者是正样本，后者是负样本。

“[CLS] 微积分是数学的一个分支 [SEP] 它由牛顿和莱布尼茨发明 [SEP]”

“[CLS] 它由牛顿和莱布尼茨发明 [SEP] 微积分是数学的一个分支 [SEP]”

SOP 也是个二分类问题，模型结构与 NSP 相同，如图 5.2 所示。SOP 可以与 MLM 等任务同时使用，即遮挡两句话中 15% 的字，在遮挡的位置上预测遮挡的字，在 [CLS] 符号位置上预测句子顺序。

预训练的好处是可以利用海量的文本数据，而无需做任何人工标注，训练用的标签都是 MLM、NSP、SOP 等任务自动抽取的。工业界的预训练有以下几条经验。

- 数据规模是预训练的关键。举个例子，用同样参数量的模型，在 100 亿条样本上训练 1 epoch，效果优于在 10 亿条样本上训练 10 epoch，尽管两者所需的算力相同。
- 数据的质量影响训练的效果。在做训练之前，需要对文档做去重，并清洗数据中无意义的符号、表情。
- 要根据下游任务选取预训练数据。举个例子，如果下游任务是小红书的搜索相关性、文本理解等任务，那么在预训练中使用小红书站内数据会对效果有很大提升。我们的经验是混合公开数据与小红书数据做预训练。

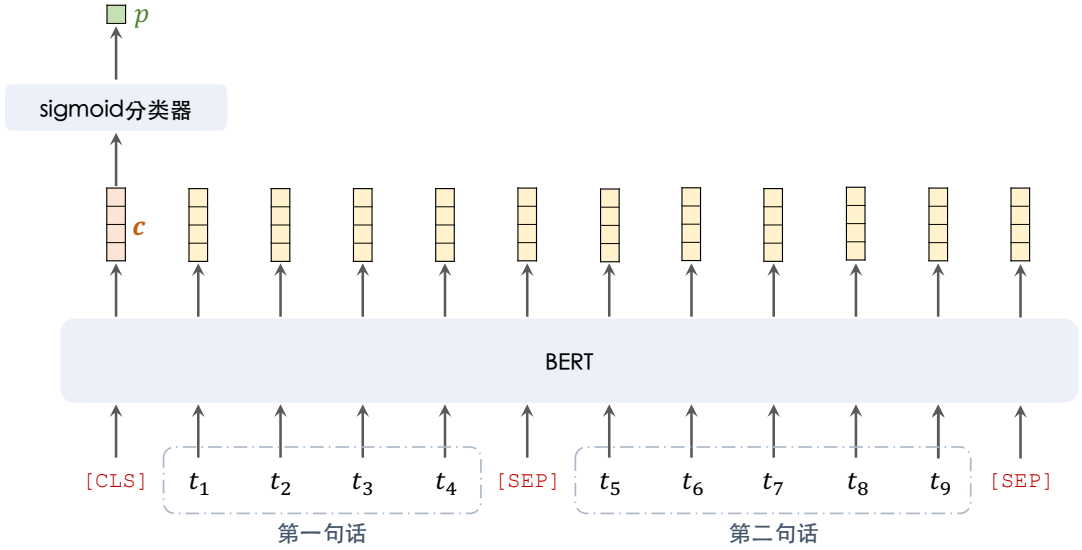

图 5.2: NSP 和 SOP 预训练方法的示意图

5.2 后预训练

学术界训练 NLP 模型的标准方法为公开语料上的预训练 + 人工标注数据上的微调。工业界往往会在预训练与微调之间加入后预训练。可以这么说，大家用的模型大同小异，决定模型模型效果最关键的点就是后预训练。

为什么学术界不做后预训练，而工业界要做呢？以搜索引擎为例，用户在搜索之后，可能会点击文档、点击筛选项，在进入文档之后还可能会发生点赞、收藏、转发、关注、评论等交互行为，这些行为都被记录在日志中。用户个体的行为偶然性很大，但是从海量用户行为中挖掘出的统计值是非常可靠的，是优质的数据。后预训练的方法通常是挖掘搜索日志，自动生成包含标签的数据。这些数据与任务相关，质量低于人工标注，但是数据量远远大于人工标注数据。下面举几个例子说明可以从用户行为中挖掘出与任务相关的标签，后面的章节将做更详细的讲解。

- 文档 d 的文本质量是排序的一个因子，它指的是写作是否认真、事实是否准确、是否对用户有帮助。如果 d 的文本质量高，那么用户更有可能点击和交互 d，反映在 d 的点击率、交互率偏高。
- 查询词 q 的时效性意图是指用户搜 q 时对“新”的需求有多强。如果 q 的时效性意图较强，且搜索结果页显示文档发布时间，则用户倾向于点击和交互较新的文档，且用户会点击“最新”按钮做筛选。
- 相关性指的是 (q, d) 的语义的匹配程度。相关性越高，则用户搜 q 之后越有可能点击和交互 d，反映在 (q, d) 的点击率、交互率偏高。

后预训练的基本流程都相同，下面以 (q, d) 的相关性为例讲解。首先寻找与标签（即相关性分数）有关的信号，比如当查询词为 q 时 d 的点击率与交互率，把这些信号记作向量 $\boldsymbol{x}_{q,d}$。然后从搜索日志中抽取数万对 (q, d) 二元组，以及他们的用户行为统计量 $\boldsymbol{x}_{q,d}$，

用人工根据 q 和 d 的文本标注相关性档位或分数，记作 $y_{q,d}$。接下来训练模型 $t(\boldsymbol{x}_{q,d})$，目标是拟合 $y_{q,d}$；t 被称为教师模型，通常是 GBDT 这样的小模型。再从搜索日志中挖掘数亿对 (q,d)，根据用户行为统计量 $\boldsymbol{x}_{q,d}$，用教师模型打分 $\hat{y}_{q,d}=t(\boldsymbol{x}_{q,d})$。最后，把 (q,d) 的文本作为相关性模型的输入，把 $\hat{y}_{q,d}$ 作为相关性模型拟合的目标，用数亿条数据训练模型，这就叫做后预训练。在后预训练的过程中，保留 MLM 或更多预训练任务，这些预训练任务在损失函数中的权重可以较小，但不能没有。

从上面的例子中不难看出，后预训练与下游任务有关，且需要少量人工标注训练教师模型。后预训练的标注量通常少于微调，比如相关性的应用中，后预训练需要数万人工标注样本训练教师模型，而微调需要数十万、数百万人工标注样本训练相关性模型。值得注意的是，如果我们用教师模型 t 生成数据去训练最终的模型 f，那么就不能用 f 的打分作为 t 使用的特征，否则会形成“反馈回路”，结果不可控。以相关性为例，教师模型 t 只能使用 q、d 的文本特征、用户行为统计量，而不能使用相关性模型 f 的打分。

5.3 微调

微调是训练模型的最后一步，在预训练或后预训练的基础上，在人工标注的数据上进一步训练模型。微调要求高质量的人工标注数据。在工业界的实践中，对效果起决定作用的不是微调的方法，而是数据标注的质量。数据标注需要产品团队、算法团队、标注团队三方合作完成。

产品团队制定标注规则，培训标注团队，解释标注中遇到的困难样本，并验收标注的成果。标注规则非常重要，如果后来大改标注规则，那么之前标注的样本就会被丢弃。比如相关性数据，几乎没有大厂的 (q,d) 样本量能积累到千万量级，原因就是标注标准会中途发生变化。标注规则很难一开始就很完美，为了避免浪费，制定标注规则最好是参考借鉴头部大厂的经验。

算法团队负责选取需要标注的样本，这个听起来简单，但选数据是有讲究的。以相关性数据为例，在选 q 时，应当覆盖头、中、尾部查询词，而不能大多是头部查询词。给定 q 选 d 时，不能只选搜索结果页上排名很高或很低的 d，这些 d 普遍与 q 高相关或低相关；应当想办法选取一定比例困难样本，即档位靠中间的。

标注团队可以是公司长期雇佣的员工，也可以是外包公司。如果是相关性这样对准确性要求高、且需要长年累月持续标注的任务，则使用公司长期雇佣的团队。标注员若能长期稳定从事一项标注工作，那么专业性会持续提升，标注质量会很高。如果是一次性的标注任务，比如标注一批 10 万条样本，然后任务就结束，那么通常会交给外包公司做，这样的标注质量不会太高。

为了保障标注质量，同一条样本至少由两人标注，结果一致算通过，如果不一致会引入第三人，甚至第四、五人。每标注完一批数据，需要考核两人的一致率，按一致率付佣金，如果一致率过低则数据作废。有时，算法和产品团队为了防止标注作弊，需要往标注数据中“埋雷”。比如往 1 万条待标注的数据中掺杂 200 条由产品和算法团队亲自标注的可信样本，在外部团队标注完成之后，计算这 200 条样本的标注准确率，用于评

估标注的质量。

5.4 蒸馏

在训练好 BERT 模型之后，通常需要将它部署到线上做实时推理。以相关性为例，当用户搜索 q 时，有候选文档 $d_1, \cdots, d_k$，需要用相关性 BERT 给所有 (q, d_i) 打分。出于成本的考虑，线上最多只能用 4 到 12 层 BERT 模型，不可能用 24 层或 48 层的大模型。

如果限定线上模型的层数和参数量，比如 4 层小模型，什么样的训练方法可以获得最优的效果呢？一种方法是从头到尾——预训练、后预训练、微调——训练小模型。另一种方法先从头到尾训练大模型，然后做知识蒸馏训练小模型。工业界的共识是如果数据量足够，那么知识蒸馏训练的小模型效果更优。有各种各样的方法做蒸馏，比如蒸馏输出层、自注意力层、embedding 层。在有工业界超大规模数据量的前提下，只需要用最简单的方法即可，只蒸馏输出层。

蒸馏的第一步是训练一个大模型，比如 24 层或 48 层的 BERT 模型。基于 2022 年的算力考量，48 层、十多亿参数的 BERT-Xlarge 是性价比最高的，准确率远高于 12 层的模型，而且所需的算力是工业界普遍能承受的，训练和蒸馏都不会太慢。训练大模型走完整个流程，包括预训练、后预训练、微调，用各种技巧优化大模型的测试准确率。

蒸馏的第二步是准备一个大数据集，用于训练小模型的数据。以相关性为例，可以是包含数亿对 (q, d) 的后预训练数据集，也可以是更大的数据集，数据量越大越好。让训练好的大模型给 (q, d) 打分，也就是说，大模型起到了标注员的作用。

蒸馏的最后一步是在大数据上训练小模型，让小模型的输出拟合大模型的打分。如果是二分类任务，模型最后一个全连接层输出的 logit 记作实数 $l \in \mathbb{R}$，激活函数输出概率 $p = \text{sigmoid}(l)$。如果是多分类任务，类别数量为 k，模型最后一个全连接层输出的 logit 为向量实数 $\boldsymbol{l} \in \mathbb{R}^k$，激活函数输出概率分布 $\boldsymbol{p} = \text{softmax}(\boldsymbol{l})$，它的每个元素是一个类的概率。有两种方法训练小模型，一种是用均方误差损失函数拟合 logit，另一种方法是用交叉损失函数拟合概率。根据我们和同行的经验，似乎是前者效果更好，但大家可以两种方法都尝试。

为了保证推理速度够快、计算成本够小，小模型通常使用 FP16 低精度浮点数，而且限定文档长度为 256。最好是在训练的时候，小模型就使用 FP16，而非先用 FP32 再做量化。相较之下，大模型可以使用 FP32 浮点数，文档长度可以是 512 或 1024，目的是让大模型准确标注训练小模型的样本。

第三部分

什么决定用户体验？

第 6 章 相关性

相关性指的是查询词 q 与文档 d 之间的关系，如果 d 能满足 q 的需求，则 (q,d) 相关。对用户体验影响最大因子就是相关性，它的重要程度远超内容质量、时效性、地域性、个性化。可以这么说，只要能保证相关性，搜索引擎就算是基本可用。

6.1 相关性的定义与分档

每家有搜索引擎业务的公司都需要制定相关性标注标准，然后由人工标注 (q,d) 的相关性档位。相关性的标注标准必须是严谨且可执行的，定义每个档位具体的意义，并给出充足的范例。

6.1.1 相关 VS 不相关

给定 (q,d)，标注员首先需要做二分类，判断相关或不相关。在此基础上，再进一步细分档位。下面我们讨论二分类的基本原则。

第一，相关性是指 d 能否满足 q 的需求或回答 q 提出的问题，而非字面上的匹配。哪怕文档 d 中不包含 q 中的任何字和词，q 与 d 也可以具有高相关性；反之，即便 d 中包含完整的 q，两者也可能不相关。举个例子，用户搜索 q =“谁掌握芯片制造的尖端技术”，搜出一篇介绍 ASML 公司光刻机技术的文档 d，虽然在字面上与 q 不匹配，但 d 是大多数用户想要找的，因此与 q 具有高相关性。另一个例子，一篇文档 d 为“我去马德里旅游，吃到了最好最正宗的西班牙海鲜饭，回来研究了一番，这个视频给大家介绍西班牙海鲜饭的做法”，它与查询词 q =“马德里旅游”不相关，哪怕 d 中包含 q。

第二，相关性的标注只考虑相关性，不考虑内容质量、时效性、地域性、个性化。这句话听起来像是依据废话，但如果不强调，很容易出现错误的标注规则。如果 q 提出一个问题，d 给出一个针对性强、但事实错误（或过时）的回答，应当标注为相关，问题出在了内容质量（或时效性）。在标注 (q,d) 相关性的时候，不要考虑用户 u 的兴趣点；只要大多数具有背景知识的人认为 q 与 d 相关，那就理应认定 q 与 d 相关。

第三，q 可能具有宽泛的需求，文档只需要解决其中任意一种需求就算相关。例如，q =“黑寡妇”的主要意图包括黑寡妇蜘蛛、黑寡妇电影、黑寡妇演员、车臣黑寡妇组织。科普黑寡妇蜘蛛的文章、黑寡妇电影的预告片、黑寡妇演员（斯嘉丽 · 约翰逊）关于该角色的访谈、车臣黑寡妇组织的介绍全都算与 q 相关。

第四，实际标注的时候，存在很多较难判定的实例，比如 q 与 d 部分匹配，这样实例该如何判定呢？

- 主需求词丢失或发生变化，则无法满足查询词的需求，判定为不相关。例如搜 q =“母亲节礼物”，出 d =“情人节礼物”；搜 q =“黄晓明”，出 d =“杨颖”。
- 如果限定词丢失，则需要进一步判断丢失的限定词是否重要。例如搜 q =“初二下册物理考点”，主需求词是“物理考点”，限定词是“初二”和“下册”。文档 d =“初三下册

物理考点”和“初二上册物理考点”，则无法满足用户需求，判定为不相关。再比如搜 q =“精彩的好莱坞动作片”，主需求词是“动作片”，限定词是“精彩”和“好莱坞”，前者不重要，后者重要。文档 d =“精彩的印度动作片”，丢失重要限定词，无法满足用户需求，判定为不相关；文档 d =“不能错过的十部好莱坞动作片”，丢失不重要限定词，可以满足用户需求，判定为相关。

- 搜上位词出下位词，即 d 是 q 的子集，这种情况判定为相关。例如，搜 q =“粤菜推荐”，出 d =“潮汕美食”；搜 q =“新疆旅游”，出 d =“喀纳斯旅游攻略”。反之，搜下位词出上位词，很可能会无法满足需求，被判定为不相关。例如，搜 q =“潮汕美食”，出 d =“经典广东菜”，其中没有潮汕菜，因此被判定为不相关。
- 一些特殊的限定词，尤其是属性，往往无法精准命中，需要有一定的容忍阈值。例如搜 q =“身高 178 男生穿搭”，出 d_1 =“身高 175 男生穿搭”和 d_2 =“高个男生穿搭”。如果容忍身高误差 ±3 厘米，则 d_1 与 q 相关。如果界定 180 厘米以上为男性高个，则 d_2 与 q 不相关。
- 用虚假标签引流。例如搜 q =“喀纳斯旅游”，文档通篇介绍新疆伊犁旅游，但是带了个引流标签“# 喀纳斯旅游”。

6.1.2 档位细分

上文中讨论如何根据 d 是否满足 q 的需求，对 (q, d) 做二分类，分为相关和不相关。在上述二分类的基础上，还需要进一步细分档位，业界通常使用 4 档或 5 档。以 4 档为例，将“相关”细分为“高”和“中”，将“不相关”细分为“低”和“无”，如图 6.1 所示。

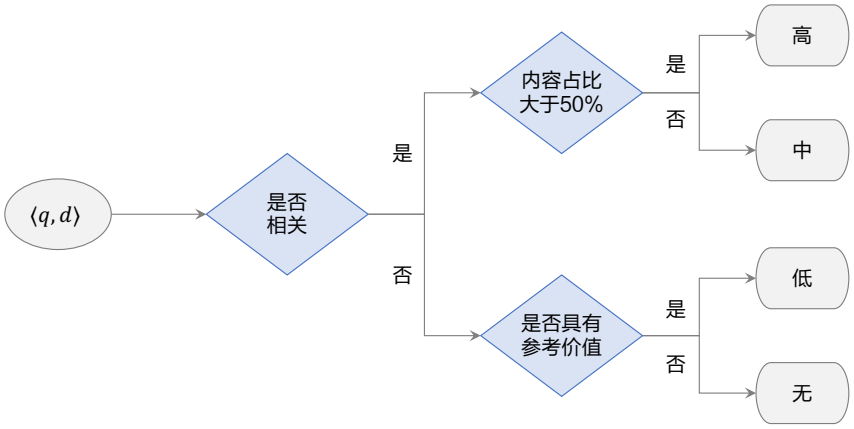

图 6.1: 相关性档位的划分

对于相关的 (q, d)，根据满足需求的内容的篇幅占比是否大于 50% 做分档，下面举例解释。

- 搜索 q =“泰坦尼克号”，d 是演员莱昂纳多的访谈，谈及他的几部代表作，其中重点谈了《泰坦尼克号》电影，因此判定为相关。如果访谈中有 50% 以上的篇幅为《泰坦尼克号》电影，则判定为高档位，否则判定为中档位。
- 搜索 q =“小米手机测评”，d 是安卓手机测评，其中测评了多款手机，包括几款小米

手机，因此判定为相关。如果小米手机所占篇幅超过 50%，则判定为高档位，否则判定为中档位。

不相关的意思是 d 不能满足 q 的需求，对于这种情况，根据文档是否有参考价值，划分为低和无两档。没有参考价值很好理解，比如搜 q =“面霜”出 d =“口红测评”。不相关、但是有参考价值，可以分为以下几种情况。

- d 丢失了 q 的重要限定词，例如搜 q =“初二下册物理考点”，d_1 =“初三中考物理考点”有参考价值。但如果丢失了主需求词“物理考点”，搜出 d_1 =“初二语数外考点”则没有参考价值。
- q 中有多个主需求词，d 丢失了一部分，例如搜 q =“上海落户政策”，d 丢失“上海”或“落户”其中一个，都被判定为不相关，但仍具有参考价值。
- d 顺嘴提一句 q（包括引流标签），但没有满足 q 的任何需求，这样的情况算作不相关，但是有参考价值。
- 如果 q 是提问，而 d 没有回答问题，这种情况算作不相关。例如，搜 q =“双鱼座与摩羯座能结婚吗”，d_1 是 12 星座的命理分析，d_2 是 12 生肖的命理分析。两篇文档都不相关，但 d_1 具有参考价值，而 d_2 没有参考价值。

6.2 文本匹配分数

本节介绍文本匹配（TF-IDF 或 BM25）和词距（OkaTP 或 BM25TP）这两类文本匹配分数，在深度学习兴起之前，它们是计算相关性的重要依据。现在 BERT 等语义模型是相关性的最优选型，但文本匹配分数仍在工业界使用。BERT 模型打分（一个实数）与多种文本匹配分数（多个实数）作为特征输入 GBDT 模型，GBDT 模型输出一个分数，作为预测的相关性分数，过去普遍认为这个分数比 BERT 模型打分更准。但随着 BERT 模型训练得越来越好，BERT+GBDT 并没有显著优于 BERT，单独用 BERT 模型判断相关性就已经足够好了。

6.2.1 词匹配分数

在用户发起搜索请求之后，分词模块将查询词 q 切分为若干个词，记作集合 $\mathcal{Q}$。例如 q =“好莱坞电影推荐”，分词得到 $\mathcal{Q}$ = {好莱坞, 电影, 推荐} 直觉上说，如果 $\mathcal{Q}$ 中的词在文档 d 中出现次数越多，则相关性越高。TF-IDF 和 BM25 都是基于上述想法。

我们首先计算 $\mathcal{Q}$ 中的一个词 t 与文档 d 的相关性。词 t 在 d 中出现的次数叫作 term frequency（TF），记作 $\mathrm{tf}_{t,d}$，TF 越大，则说明 t 与 d 的相关性越高。TF 存在一个缺点，即 TF 与文档长度正相关，如果简单地用 TF 计算相关性，那么长文档会占优势。举个例子，把 d 连接到 d 的后面，得到文档 $d' = d + d$，显然有 $\mathrm{tf}_{t,d'} = 2\mathrm{tf}_{t,d}$ 这样的关系。照理说，d 与 d' 的信息量是相同的，(t, d) 与 (t, d') 的相关性应当相同，但是用 TF 算出的相关性却差了一倍。为了解决文档长度问题，我们对 TF 做归一化。设 $l_d = \sum_t \mathrm{tf}_{t,d}$ 为文档 d 的长度，即 d 中包含的词的数量，相同的词做重复计数。用 $\frac{\mathrm{tf}_{t,d}}{l_d}$ 表示 (t, d) 的相关性。

同等对待 $\mathcal{Q}$ 中所有的词 t，计算加和

$$\sum_{t\in\mathcal{Q}} \frac{\mathrm{tf}_{t,d}}{l_d}, \tag{6.1}$$

用它表示 (q,d) 的相关性。这种方法不受文档长度影响，文档 d 与 $d'=d+d$ 的相关性分数相同。

上述方法同等对待 $\mathcal{Q}$ 中所有的词 t，直接对 $\frac{\mathrm{tf}_{t,d}}{l_d}$ 关于 $t\in\mathcal{Q}$ 求加和，没有对 t 做加权。但事实上，并非所有的词都同等重要。比如用户搜索 $\mathcal{Q}=\{$魔兽, 游戏$\}$，两个词不该被同等对待，因为包含“魔兽”的文档数量较少，而包含“游戏”的文档数量非常多，前者的出现更能说明查询词与文档相关。更极端的例子是停词，比如“你”、“的”、“是”，几乎所有的文档都包含这些词，这些词毫无区分度，这些词出现在文档中不能说明查询词与文档相关。

基于上述想法，式 6.1 中关于 t 取连加时，应当对词做加权。如果词 t 在很多文章中出现，比如“你”、“的”、“是”，这样的词没有区分度，对相关性的贡献小。反之，如果 t 只在很少的文章中出现，比如“魔兽”、“肖申克的救赎”、“强化学习”，这样的词区分度很高，对相关性的贡献大。用 df_t 表示词 t 在多少文档中出现过，它被称作 document frequency（DF）。

设 N 为数据集中文档的总数，inverse document frequency（IDF）定义为 $\mathrm{idf}_t=\log\frac{N}{\mathrm{df}_t}$。TF-IDF 这样计算查询词 $\mathcal{Q}$ 与文档 d 的相关性：

$$\mathrm{TFIDF}(\mathcal{Q},d) \;=\; \sum_{t\in\mathcal{Q}} \frac{\mathrm{tf}_{t,d}}{l_d}\times\mathrm{idf}_t.$$

除此之外，TF-IDF 还有其他变种，比如

$$\mathrm{TFIDF}(\mathcal{Q},d) \;=\; \sum_{t\in\mathcal{Q}} \log(1+\mathrm{tf}_{t,d})\times\mathrm{idf}_t.$$

Okapi BM25 通常被叫做 BM25，其中的 BM 是 best matching 的缩写。BM25 有很多种变体，这里只介绍其中的一种。BM25 这样计算查询词 $\mathcal{Q}$ 与文档 d 的相关性：

$$\mathrm{BM25}(\mathcal{Q},d) \;=\; \sum_{t\in\mathcal{Q}} \frac{\mathrm{tf}_{t,d}\times(k+1)}{\mathrm{tf}_{t,d}+k\times\left(1-b+b\times\frac{l_d}{\mathrm{mean}(l_d)}\right)}\times\ln\left(1+\frac{N-\mathrm{df}_t+0.5}{\mathrm{df}_t+0.5}\right).$$

连加中的第一项类似于 TF，第二项类似于 IDF。上式中的 k 和 b 是两个参数，默认设置 $k\in[1.2,2]$ 和 $b=0.75$。有很多种对 BM25 的解释，此处就不展开讲了，大家只需要把 BM25 看做 TF-IDF 的一种变体即可。

TF-IDF 与 BM25 都是计算相关性的方法，它们都隐含一个假设，即词袋模型（bag of words），它把文档中的词看做集合 $\{(t,\mathrm{tf}_{t,d})\}$。举个例子，如果用词袋模型，$q_1$ =“送 男朋友 的 礼物”与 q_2 =“男朋友 送 的 礼物”完全等价，q_3 =“白 衬衫 灰 裤子 搭配”与 q_4 =“灰 衬衫 白 裤子 搭配”完全等价。词袋模型完全忽略了词的顺序和上下文，因此 TF-IDF、BM25 计算出的相关性分数不准确，远不如 BERT 等先进的语义模型。

6.2.2 词距分数

我们先来看一个例子。查询词 q =“亚马逊雨林”被切分成 $\mathcal{Q}=\{$亚马逊, 雨林$\}$。某篇召回的文档 d =“我在亚马逊上网购了一本书，介绍东南亚热带雨林的植物群落”同时

包含两次词，但显然 (q,d) 不相关。如果用 TF-IDF 或者 BM25 计算相关性，结论很可能是 (q,d) 相关。为了解决这个问题，我们需要引入词距（term proximity）。t 和 t' 是 $\mathcal{Q}$ 中的两个词，它们都出现在文档 d 中，定义距离 $\mathrm{dist}(t,t')$ 为 t 和 t' 之间间隔的词的数量。词距越大，则 t 和 t' 对相关性的贡献越小。

OkaTP 是 2003 年的论文[14] 提出的，用类似于 BM25 的公式计算词距分数。设 t 和 t' 是 $\mathcal{Q}$ 中的两个词，它们都在文档 d 中出现至少 1 次。把 t 出现在 d 中的所有位置记作集合 $\mathcal{O}(t,d)$，比如做分词之后，d 中第 27、84、98 位置上的词都是 t，那么 $\mathcal{O}(t,d)=\{27,84,98\}$。集合的大小 $|\mathcal{O}(t,d)|$ 等于词频 $\mathrm{tf}_{t,d}$。定义

$$\mathrm{tp}(t,t',d) \;=\; \sum_{o\in\mathcal{O}(t,d)} \sum_{o'\in\mathcal{O}(t',d)} \frac{1}{|o-o'|^2},$$

式中的 $|o-o'|$ 就是两个词的距离。两个词 t 和 t' 在文档 d 中出现频次越多、距离越近，则 $\mathrm{tp}(t,t',d)$ 越大。OkaTP 用类似 BM25 的方式计算查询词 $\mathcal{Q}$ 的词距分数：

$$\sum_{t,t'\in\mathcal{Q}} \frac{\mathrm{tp}(t,t',d)\times(k+1)}{\mathrm{tp}(t,t',d)+k\times\left(1-b+b\times\frac{l_d}{\mathrm{mean}(l_d)}\right)} \times \min\left\{\mathrm{idf}_t,\, \mathrm{idf}_{t'}\right\}.$$

这里的 idf_t 与 BM25 中的定义相同：

$$\mathrm{idf}_t \;=\; \ln\left(1+\frac{N-\mathrm{df}_t+0.5}{\mathrm{df}_t+0.5}\right).$$

把上述词距分数与 BM25 分数相加，作为文档 d 与查询词 $\mathcal{Q}$ 的相关性分数。

BM25TP 是 2006 年的论文[3] 提出的，与 OkaTP 非常相似。同样定义集合 $\mathcal{O}(t,d)$ 包含词 t 出现在文档 d 中的位置。定义

$$\mathrm{tp}(t,d) \;=\; \sum_{t':\,t'\neq t} \sum_{o\in\mathcal{O}(t,d)} \frac{\mathrm{idf}_{t'}}{|o-\mathrm{pos}(o,t')|^2}.$$

式中的 $\mathrm{pos}(o,t')$ 是词 t' 出现的一个位置，这个位置必须在 o 之前（即 $o'<o$），且距离 o 最近。式中分母的本质就是 $\min_{o'\in\mathcal{O}(t',d)}|o-o'|^2$，但是要限制 o' 在 o 之前。读者可能会好奇，为什么要限制 $o'<o$ 呢？其实是为了计算方便，只需要从前往后遍历文档 d 中的词，发现位置 o 上是词 t，那么向前回溯词 t' 最近一次出现的位置，得到位置 $\mathrm{pos}(o,t')$。BM25TP 用类似 BM25 的方式计算查询词 $\mathcal{Q}$ 的词距分数：

$$\sum_{t\in\mathcal{Q}} \frac{\mathrm{tp}(t,d)\times(k+1)}{\mathrm{tp}(t,d)+k\times\left(1-b+b\times\frac{l_d}{\mathrm{mean}(l_d)}\right)} \times \min\left\{\mathrm{idf}_t,\, 1\right\}.$$

把上述词距分数与 BM25 分数相加，作为文档 d 与查询词 $\mathcal{Q}$ 的相关性分数。

6.3 相关性 BERT 模型

以 BERT 为代表的语义模型是目前工业界最普遍使用的相关性模型，准确性远超传统的手工特征（比如 BM25、BM25TP）+ GBDT 融合。搜索引擎线上推理通常使用两类 BERT 模型，一类是交叉 BERT，另一类是双塔 BERT。前者准确、但是消耗巨大算力，只能给少量文档打分；后者不准确，但是推理代价很小，可以给海量文档打分。通常将双塔 BERT 模型用于召回海选，将 4 层交叉 BERT 模型用于粗排，将 12 层交叉 BERT 模型用于精排。

6.3.1 交叉 BERT 模型

交叉 BERT 的结构如图 6.2 所示，“交叉”的意思是自注意力层对查询词和文档做了交叉。把查询词、文档标题、文档正文（或摘要）按字粒度（或字词混合粒度）输入 BERT。每个字被表征为三个 embedding 向量：token embedding 作为汉字（或英文单词）的表征，position embedding 用于区分字的序，segment embedding 用于区分查询词、文档标题、文档正文。把三个 embedding 向量相加，作为一个字的表征。BERT 的输出是一个介于 0 和 1 之间的分数，表示查询词与文档的相关性。

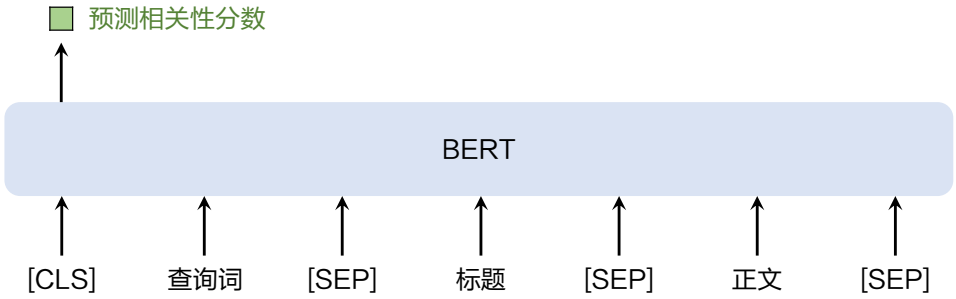

图 6.2: 相关性 BERT 模型的示意图

上述方法是最基础的模型，在此基础上有多种改进，比如 anchor query 方法。给定文档 d，用 Transformer 模型生成高相关性的查询词 $q_1, \cdots, q_k$，它们被称作 anchor query；具体的方法见 18.3 节。将 $q_1, \cdots, q_k$ 附在文档末尾，相当于文档内容的一部分，输入 BERT 模型。我们发现 Transformer 生成的 anchor query 类似于极简的文本摘要，用几个字概括文档内容，起到了给文档“打标签”的作用。如果 anchor query 质量足够高，可以帮助 BERT 模型更好地预测相关性。根据我们的经验，anchor query 对双塔 BERT 模型的帮助较大，对交叉 BERT 模型的帮助较小。

我们在实践中发现字词混合粒度的 WoBERT 模型的准确性优于字粒度的 BERT 模型。给定一个固定的中文词典（大小在 10 万左右），事先用分词器将文档切分成词，没有命中词典的话则被当做字对待。设输入 BERT 模型的序列包含 m 个字（或词），每个字（或词）被嵌入层映射成一个向量，自注意力层推理的时间复杂度为 $\mathcal{O}(m^2)$，全连接层推理的时间复杂度为 $\mathcal{O}(m)$。BERT 推理的代价随 m 超线性增长，因此我们需要设置一个长度上限，比如 128 或 256，截断超出上限的文档正文。与字粒度相比，字词混合粒度可以大幅缩短序列长度 m，这样一来，文档正文更不容易被截断丢弃，相关性的预测因此变得更准确。

交叉 BERT 模型对相关性的预测很准确，但需要付出很大的推理代价。可以这么说，交叉 BERT 模型是搜索链路上计算成本最大的模块，代价远高于其他任何模块。线上有多种降本策略，可以避免绝大部分的推理。一种常用的策略是在 Redis 内存中缓存模型算出的 (q, d) 相关性分数，只要 (q, d) 命中缓存，则避免模型推理。由于 Redis 内存有限（比如容量为 2TB），需要使用 least recently used（LRU）等机制让缓存 (q, d) 退场。其他的降本策略见 18.2 节。

6.3.2 双塔 BERT 模型

双塔 BERT 模型的结构如图 6.3 所示，左右两个塔分别将查询词 q 和文档 d 表征为向量 $\boldsymbol{x}_q$ 和 $\boldsymbol{z}_d$，用 $\text{sigmoid}(\boldsymbol{x}_q^\top \boldsymbol{z}_d)$ 作为相关性分数（介于 0 和 1 之间）的预测。与交叉 BERT 模型不同，此处的两个 BERT 模型独立提取 q 和 d 的向量表征，自注意力层没有对 q 和 d 做交叉。

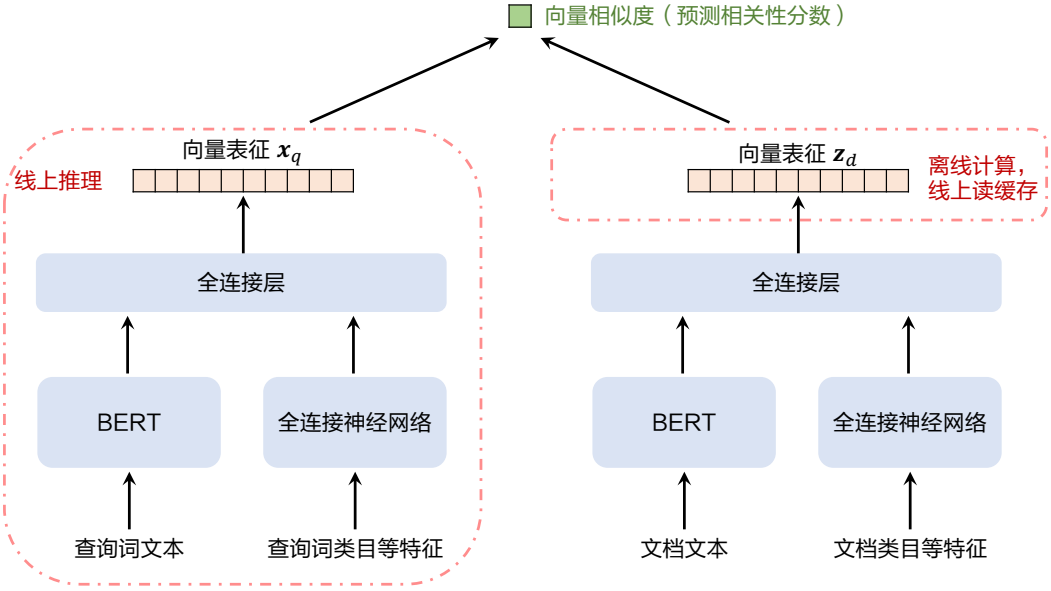

图 6.3: 双塔 BERT 模型的示意图

图 6.3 中的左塔把 q 的文本和其他特征作为输入，输出向量 $\boldsymbol{x}_q$ 作为查询词的表征。每次搜索只有一条 q，而且它的长度通常很短，因此左塔的推理的代价很小，可以在线上实时计算 $\boldsymbol{x}_q$。右塔把文档 d 的文本和其他特征作为输入，输出向量 $\boldsymbol{z}_d$ 作为文档的表征。召回海选每次给数万篇文档打分，而且文档的长度较大，假如在线上做推理，则右塔的计算量非常巨大。好在文档是静态的，可以离线做推理计算 $\boldsymbol{z}_d$，将 $\langle d, \boldsymbol{z}_d \rangle$ 存储在哈希表中，线上不做推理，而是直接读哈希表。

双塔模型在线上推理的代价有多大呢？如果将双塔模型用在召回海选，给 5 万篇文档计算相关性，那么代价包括：

- 用左塔 BERT 计算查询词 q 的向量 $\boldsymbol{x}_q$。
- 给定 5 万篇文档的 ID，从哈希表中读取对应的 5 万个向量 $\{\boldsymbol{z}_d\}$。
- 计算 5 万次向量内积 $\boldsymbol{x}_q^\top \boldsymbol{z}_d$。

如上所述，哪怕候选文档数量很多，双塔模型的计算量也不大。

6.4 相关性模型的训练

工业界训练相关性模型的最佳实践为预训练、后预训练、微调、蒸馏这四个步骤。如果条件允许，最好是做预训练、后预训练、微调这三个步骤训练 48 层 BERT 大模型，取得最优的 AUC 和正逆序比，然后在至少 10 亿对 (q, d) 二元组上蒸馏 BERT 小模型（12

层、6 层、4 层)、双塔 BERT 模型、DNN 模型。前文已经初步讲解过预训练、后预训练、微调、蒸馏，其中预训练和微调是任何 NLP 任务都需要的，没有特别之处；本章重点讲解后预训练与蒸馏，这两部分需要很多特殊的技巧。

6.4.1 评价指标

相关性的离线评估通常使用 AUC 和正逆序比。AUC 是一种 pointwise 指标，独立对待每一对 (q, d)。6.1 节讨论过，高、中两档指 (q, d) 相关，低、无两档指 (q, d) 两档无关。对档位做合并，相关性成为二分类问题。测试集上 (q, d) 的真实标签为 $y_{q,d} \in \{0, 1\}$，相关性模型输出预测值 $p_{q,d} \in (0, 1)$，用 AUC 指标衡量 $p_{q,d}$ 对 $y_{q,d}$ 的预测是否准确。

正逆序比是一种 pairwise 指标，考察模型对文档的排序与真实标签的排序的一致性。给定一条查询词 q 和 k 篇文档 $d_1, \cdots, d_k$，设 $y_1, \cdots, y_k$ 为真实相关性档位（分 4 档），$p_1, \cdots, p_k$ 为相关性模型的打分。k 篇文档可以组成 $\binom{k}{2}$ 个文档二元组，二元组可以是正序对，也可以是逆序对。对于二元组 (d_i, d_j)，设 $y_i \geq y_j$，正序对满足 $p_i \geq p_j$，逆序对满足 $p_i < p_j$。正序对与逆序对数量之比为正逆序比。

6.4.2 损失函数

后预训练、微调、蒸馏都需要用到两种损失函数。一种是 pointwise 损失函数，比如交叉熵和均方误差，它们的作用是“保值”，即让预测的相关性接近真实的相关性标签，有利于提升 AUC 指标。另一种是 pairwise 损失函数，它的作用是“保序”，即让预测的相关性的序与真实标签的序尽量一致，有利于提升正逆序比。

我们首先讨论 pointwise 损失函数。设相关性模型输出的概率为 $p = \text{sigmoid}(l)$，此处的 l 被称作 logit。我们把真实标签（比如 4 个档位）变换到 $[0, 1]$ 区间上，记作 y。可以使用交叉熵损失：

$$\text{CE}(y, p) \;=\; -y \cdot \ln\big(p\big) \;-\; (1-y) \cdot \ln\big(1-p\big).$$

交叉熵损失越小，意味着相关性模型的预测越准确。也可以使用均方误差损失：

$$\text{MSE}(y, l) \;=\; (y - l)^2.$$

使用均方误差损失，让 l 拟合 y。

下面我们讨论 pairwise 损失函数。一条样本包含一个查询词 q 和 k 篇文档 $d_1, \cdots, d_k$，我们希望正序对数量越多越好。如果 $y_i > y_j$，则 $p_i - p_j$ 越大越好。我们使用 pairwise logistic 损失函数，当 $y_i > y_j$ 时，下式越小越好。

$$L(p_i, p_j) \;=\; \ln\Big[1 \;+\; \exp\Big(-\gamma \cdot \big(p_i - p_j\big)\Big)\Big],$$

上式中的 $\gamma > 0$ 是一个超参数，控制 logistic 函数的形状。综合交叉熵和 pairwise logistic 损失函数，我们得到下面的损失函数：

$$\frac{1}{k}\sum_{l=1}^{k} \text{CE}(y_l, p_l) \;+\; \lambda \cdot \frac{1}{k^2} \sum_{(i,j):y_i > y_j} L(p_i, p_j).$$

最小化上式中第一项，可以让相关性模型拟合分数 y_i；最小化上式中的第二项，可以让相关性模型的打分与 $\{y_i\}$ 的序趋于一致；式中的 $\lambda > 0$ 是需要调的超参数。

6.4.3 后预训练

在工业界，人工标注的 (q, d) 二元组数量通常是数十万到数百万，仅用人工标注的数据做监督学习，不足以将 BERT 模型训练到极致。如果条件允许，后预训练可以将监督学习的数据量扩充到 10 亿以上，是提升 AUC 和正逆序比最有效的手段。

获取后预训练的主要途径是挖掘搜索引擎的日志，计算 (q, d) 的多种统计值，包括 d 自身的点击率和交互率，搜索 q 时 d 的点击率和交互率。后预训练用到一个教师模型，通常是 GBDT 这样的小模型，它将 (q, d) 的多种统计值映射到相关性档位。我们使用少量（数万）人工标注的样本训练教师模型，让教师模型拟合人工标注的档位。在完成训练之后，教师模型可以将 (q, d) 的多种统计值映射到相关性档位，记作 $\widehat{y}_{q,d}$。通过这种方法，我们获得海量的 $(q, d, \widehat{y}_{q,d})$，作为训练 BERT 模型的样本。后预训练的数据构造方法可以参考百度 2021、2022 年的论文[25, 23]。

后预训练同时用三种损失函数，对三种损失函数求加权和作为总的损失。第一种是 pointwise 损失，最常用的是均方误差或交叉熵，起到保“值”的作用，有利于提升 AUC。第二种是 pairwise 损失，起到保“序”的作用，有利于提升正逆序比。第三种是 MLM 等预训练的损失，避免预训练的成果在后预训练阶段被“清洗掉”。

6.4.4 蒸馏

在算力允许的情况下，通过预训练、后预训练、微调三个步骤训练出 48 层的 BERT 大模型。但是这样的大模型推理所需的算力过大，无法部署在线上。工业界最优的实践是用 48 层大模型蒸馏 4 层、6 层、或 12 层的小模型，线上用小模型做推理。

蒸馏的方法很简单，用大模型给海量的 (q, d) 二元组打分 $r_{q,d}$，让小模型拟合 $r_{q,d}$。蒸馏小模型用到两种损失函数：一种是 pointwise 损失函数（比如均方误差或交叉熵），另一种是 pairwise 损失函数。与后预训练不同，蒸馏不使用 MLM 等预训练的损失。线上可能同时有不同的小模型，比如粗排用 4 层 BERT 模型，精排用 12 层 BERT 模型。直接用大模型“标注”的数据 $\{(q, d, r_{q,d})\}$ 训练所有的小模型效果最好，不要做多级蒸馏（即先用 48 层模型蒸馏 12 层模型，再用 12 层模型蒸馏 4 层模型），多级蒸馏只会让 4 层模型的指标更差。

工业界做蒸馏有如下几条经验。第一，如果条件允许，蒸馏用的 (q, d) 二元组数量最好在 10 亿以上；蒸馏的数据量越大，则蒸馏的损失越小。第二，大模型的参数量越大，则大模型在测试集上的指标（AUC 和正逆序比）越高，且蒸馏出的小模型的指标也越高。第三，小模型需要“预热”，即先做预训练、后预训练、微调训练小模型（与大模型的训练方法完全相同），再用大模型“标注”的数据 $\{(q, d, r_{q,d})\}$ 训练小模型。

6.5 知识点小结

- 相关性是指文档是否满足查询词的需求、或回答查询词提出的问题。工业界通常将相关性分为 4 档或 5 档。先按照相关、不相关划分为两大档，再将相关细分为 2 档或 3 档，将不相关细分为 2 档。
- 在深度学习兴起之前，工业界普遍采用人工构造的特征计算相关性。人工构造的特征包括文本匹配（比如 BM25 和 BM25TP）、类目匹配、命名实体匹配。将这些特征输入 GBDT 模型，用人工标注的数据训练模型，在线上用模型做推理，预测相关性。
- 以 BERT 为代表的深度语义模型是目前最优的相关性模型。召回海选用双塔 BERT 模型，粗排和精排用交叉 BERT 模型。可以让粗排和精排使用相同大小的 BERT 模型，也可以在粗排使用 4 层 BERT 模型，精排使用 12 层 BERT 模型。
- 即便有了 BERT 模型，人工特征仍然是有用的。将 BERT 模型打分与其他人工特征共同作为 GBDT 模型的输入，用 GBDT 模型的打分作预测相关性，它比 BERT 模型打分更准确。
- 相关性 BERT 模型的训练分为 4 步。第一，在大规模语料上做预训练。使用 MLP 和 SOP 等任务，训练 48 层 BERT 大模型。第二，挖掘搜索日志，构造大规模数据，做后预训练。用小模型将用户行为映射到相关性分数 $\widehat{y}$，自动构造出 10 亿条以上的 $(q, d, \widehat{y})$ 样本，同时用监督学习和 MLM 预训练任务训练 BERT 大模型。第三，用人工标注的数据做微调。人工标注数十万、或数百万条样本 (q, d, y)，用监督学习任务训练 BERT 大模型。最后，用训练好的大模型蒸馏小模型。蒸馏的数据量最好是在 10 亿以上，数据量越大，蒸馏的损失越小。
- 相关性模型的离线评价指标主要是 AUC 和正逆序比。AUC 是一种 pointwise 评价指标，将相关性看作二分类任务，评价分类的准确性。正逆序比是 pairwise 评价指标，将相关性视作排序任务，评价模型的排序与真实的序的一致性。

第 7 章 内容质量

用户做搜索时，在满足相关性的前提下，用户希望搜索结果是高质量的文档，而非劣质、甚至虚假的内容。内容质量主要有两个维度。第一，EAT 分数，主要取决于文档的来源和作者。EAT 在业界俗称“权威性”，早年的 PageRank 就是一种计算权威性的方法。第二，文字和图片质量，主要取决于文档本身，包括文档的意图、写作水平、图片质量。

7.1 EAT 分数

EAT 是谷歌提出的内容质量评价标准，是专业性（expertise）、权威性（authoritativeness）、可信赖（trustworthiness）三个词的首字母缩写。对于谷歌、百度这样的搜索引擎，网页文档是从全网爬取的，主要考察网站、作者、文档自身三者的 EAT。对于小红书这样的 UGC 平台，文档是小红书用户自己创作的，站内的搜索主要考察作者和文档两者的 EAT。下面分别解释 E、A、T 三个维度的含义，但实践中，标注人员综合考虑专业性、权威性、可信赖，给出 1 个 EAT 分数，而不是 3 个分数。本节部分参考文档《Google's Search Quality Evaluator Guidelines》。

专业性是作者的属性。如果文档是新闻，那么只有当作者为认证的记者、新闻媒体机构时，文档才被认为具有专业性。如果文档是医疗知识，那么只有当作者是认证的医生、正规医疗机构时，文档才被认为具有专业性。对于小红书这样的 UGC 平台，需要人工审核作者具有什么样的专业资质，对作者做标记。如果作者具有专业性，而且文档的内容符合作者的专业领域，那么文档就具有专业性。比如，平台认证的医生写的医疗建议被认定为专业性，但他写的旅行攻略不具有专业性。

权威性是网站、作者、网页的属性，谷歌、百度等搜索引擎看中权威性。对于医疗问题，如果网页来源于默沙东诊疗手册、WHO、CDC，则网页具有权威性；如果网页来源于微信公众号，则网页不具有权威性。对于编程类的问题，如果网页来源于 Stack Overflow，则网页具有权威性，因为程序员通常在这个网站提问和回答编程问题。对于地点和 POI，百度地图、高德地图、谷歌地图等网站都具有权威性。对于影星、球星等公众人物，他们的官方推特、微博页面都具有权威性。举个反例，如果某个网站提供医学建议，但是上面有不少违背常识的页面，比如“研究表明吸烟与肺癌没有因果关系”，那么该网站被认定为权威性很差。

可信赖是网站、作者的属性，指信息的来源（作者、网站）的名声如何、是否值得信任。对于饭店、酒店等 POI 的测评，大众点评、Yelp、谷歌地图等网站可信赖，这是因为这些网站上有大量真实用户写的评价，参考价值很高。在很多场景下，权威性与可信赖非常相似。例如，用户搜“哈佛医学院”，哈佛医学院的主页是官方的（权威性），那么它自然就可信赖。例如，用户搜报税问题，国税局官方页面的文档具有最高的权威性，自然也是最可信赖的信息来源。下面举一些反面的例子。如果网页上有不安全的购物链接、或是弹出诱导性的窗口，把广告伪造成正文中的链接骗点击，那么这些网页就不可

信赖。如果某个网站（或博主）把自己做得跟知名网站（或名人）很像，借此骗取用户信任，那么该网站被认定位不可信赖。如果某位测评的博主历史上有软广、甚至收黑公关的钱做恶意抹黑，那么他被认定位可信赖，因为难以判断他是做真诚测评、还是为了利益做虚假测评。如果某位博主宣称自己是律师，但被查出没有律师资格，那么他被认定位不可信赖。

对于某些类型的文档，EAT 是排序最重要的因子之一；而对于其他类型的文档，EAT 在排序中的权重相对较小。有一类文档叫做“金钱或生命”（your money or your life，YMYL），排序需要非常看重 EAT 分数。YMYL 具体包括一下几种类型。

- 健康和安全，包括诊断建议、用药建议、对医院的介绍、减肥、应急准备、风险的评估。高 EAT 分数的文档应该是由专业人士或机构编写，行文严谨和专业，最好是经过专业评审后发表的文档。
- 金融理财，包括报税、投资、退休金、贷款、银行、保险、转账、以及各种涉及金钱的信息。高 EAT 分数的文档应当来源于正规、可信、且声誉良好的作者和网站，比如国税局的官网是税务问题最权威、最可信赖的信息来源。
- 政策和法律，包括选举、政府机构、政府政策、法律问题（例如离婚、收养、诉讼）。高 EAT 分数的文档的评价标准与金融理财相同。
- 新闻和重大事件，包括政治、科技、商业等严肃话题。（体育、娱乐、八卦不被看做 YMYL。）高 EAT 分数的文档应当由专业记者或业内专家撰写，由权威媒体机构发表。
- 购物，比如对商品、店铺、服务的测评，尤其是直接提供购买链接的。高 EAT 分数的文档要求作者可信赖。
- 其他对用户生活有重大影响的信息，比如高考、择校、选专业、出国、就业、买房。高 EAT 分数的文档要求作者具有专业知识，而且作者可信赖。

7.2 文本质量

内容质量的一个维度是“意图”。意图可以是有益的，比如分享有用的攻略、知识、信息、亲身经历，对其他用户有帮助，这样的文档应该扶持。有些文档的意图是有害的，比如传播虚假信息、涉嫌诈骗、散布仇恨、歧视某些人群、引发男女对立情绪，这样的文档会被打压和删除。大多数文档没有特别的意图，不需要特别对待。

内容质量的另一个维度是“文字的质量”。标注人员在判断内容质量高低时，需要考察信息的质量（包括文章是否清晰、全面，事实是否准确，传递的信息是否有价值）和写作是否认真（包括花费的时间、精力、专业程度、写作技巧）。比如一篇介绍柴犬的文章，清晰和详细介绍了柴犬的特点和饲养的注意事项，符合事实（参考权威的宠物网站做比对），对想要养宠物的读者有价值，那么这篇文章的质量为高档。比如一则笑话，标注人员无需考察事实的正确性，只要笑话确实引人发笑，那么它的质量为高档。比如一篇影评，标注人员需要判断写作是否是认真写的、以及影评写得是肤浅还是专业，以此作为档位的判别标准。比如小红书上有很多旅游笔记，有的是认真分享自己的亲身经历、喜

欢的景点、踩过的坑，质量为高档；而有的旅游笔记仅仅描述心情、发个感慨，没有干货，对其他用户没有价值，质量为低档。如果一篇文章有多处语法错误、文笔很差、明显的事实错误、拼凑复制来的内容、或者煽动情绪，内容质量应当被判定为低档。

图文一致性、标题与正文一致性也可以反映出内容质量。比如封面用美女、标题党骗点击，实际文字内容与封面或标题无关，这样的文档点击率高，但对用户没有价值，不利于用户体验。

互联网上存在海量复制、盗用、搬运的网页、博客、视频，甚至在一个 UGC 网站内部也有大量复制和盗用图片、文字、视频的现象。这种行为可以用最小的成本创作大量貌似优质的内容，但这样重复的内容对搜索引擎毫无价值，需要被去重过滤。搜索引擎用人工和算法识别识别复制的内容，给这样的内容打上标签，不会被展示。

在实践中，对意图、文字质量、图片视频质量的判断都采用人工标注 + 模型打分。文本的理解普通使用 BERT 模型，而图片、视频的打分通常采用 CNN 或 Vision Transformer 模型。值得注意的是，使用上述内容质量分数作为排序的依据，可以提升人工体验指标，但未必能提升有点比等指标。

7.3 图片质量

对于小红书这样的图文、视频内容社区，图片和视频的质量与文字的质量同等重要，画质、美学也作为内容质量的一个维度。画质包括是否有模糊、放大是否有锯齿、图片和视频的分辨率，标准比较客观。

- 噪声：高噪声（差）、马赛克（差）、无噪声（中优）、其他。高噪声：可见的边缘锯齿、边缘条纹、阶梯状、栅格化、波浪纹。
- 曝光：过曝（差）、欠曝（差）、正常曝光（中优）、其他。前两个体现为异常的过亮或过暗，没有问题就选正常曝光。
- 景深：虚焦（差）、大光圈（中优）、大景深（中）、其他。虚焦是拍摄问题，体现为核心目标模糊；大光圈主体清晰但周围（焦外）模糊；全局没有明显的焦点模糊都是大景深，占大多数。
- 色彩饱和度、对比度：单调（差）、中等（中）、饱和（优）。
- 清晰度：模糊（差）、中等（中）、高清（优）。

美学分评分指的是个人对图片的视觉吸引力的打分，吸引力的点可以是美景、美食、精彩瞬间等，标准非常主观。一般来说，图片的画质好与坏是第一印象，其次是内容是否符合审美，还有图片的色彩、构图、信息量等也在考虑范畴。

- 颜色与光照：黑白（中）、单调（中差）、多彩（中优）、其他。主要判断画面颜色的丰富和和谐程度，多彩主要考虑视觉上主次颜色的互补性。
- 主体性：主体不突出（中差）、主体中心构图（中优）、主体三分构图（优）、其他。画面是否有明确主体拍摄目标，如果有是否位于好的构图位置（中心或九宫格交叉位）。少数场景允许主体不突出。
- 视觉元素：单主体（中）、有次要元素（中优）、层次丰富（优）、其他。画面除主体

外，是否有辅助构图。少数场景无主体也选单主体。

- 拍摄镜头（全部中性）：鱼眼/广角、全景、中焦、长焦、特写、其他。根据画面视角判定，标签顺序按照画面场景从大到小。鱼眼和全景的区别是画面边缘是否畸变，全景和鱼眼多用于建筑、风光摄影。中和长焦的判断根据背景的虚化和放大程度，日常镜头多为中焦（手机打开的默认画幅焦距）。
- 艺术构图：无构图（差）、普通（中）、设计感（优）。

具体方案可以参考抖音的文章 https://toutiao.io/posts/izme1z1/preview 和腾讯的文章 https://cloud.tencent.com/developer/news/490036。此外，曾经小红书上很多博主大量使用滤镜，照片严重失真，对用户产生误导。识别滤镜图片，并做一定的降权处理，有利于让社区生态变得更真实。

第 8 章 时效性

时效性一方面是指查询词的时效性意图（即查询词对新内容的需求），另一方面是搜索结果能否满足需求。查询词的时效性意图可以划分为突发时效性、一般时效性、周期时效性这三大类。线上由 QP 模块判别查询词的时效性意图，下游根据查询词意图做承接。不论是查询词意图识别错误，还是召回和排序针对时效性意图的承接不好，都会严重影响用户体验。

8.1 查询词时效性意图分类

本节讨论查询词的时效性意图。工业界通常把时效性划意图分为突发时效性、一般时效性、周期时效性这四大类，如图 8.1 所示。下面的 3 小节分别讨论这 3 种意图。

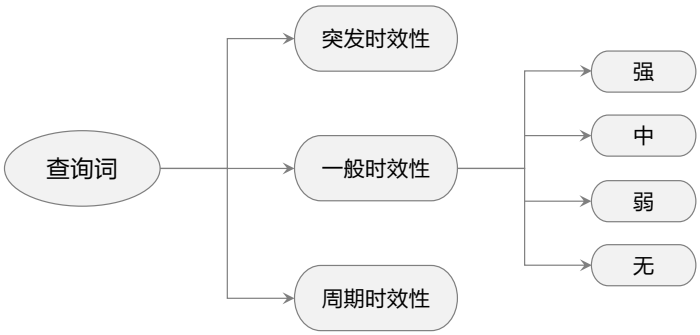

图 8.1: 查询词时效性意图的分类

8.1.1 突发时效性

突发时效性是指突发的热点，包括新闻、灾难、舆论、社会事件、商业活动。例如科比去世、唐山打人、推特裁员、取消行程码、世界杯开幕、阿玛尼打折。在这些例子中，查询词仅在事件发生之后的一两周内有时效性，现在距事件发生已经很久，这些查询词已经失去时效性。

带有突发时效性意图的查询词有如下特点。第一，从某一时刻开始，该查询词的搜索量激增，而且热度会在未来逐渐退却。算法或人工将一个查询词判定为突发时效性，搜索引擎不能永远使用这个设定，而是应当让这种设定在一段时间后过期。第二，如果一个人不看新闻，单从字面上无法看出查询词对时效性有很强的诉求。也就是说，想要判定查询词是否带有突发时效性，不能用 BERT 等语义模型，而是要从站内和站外做数据挖掘。

举一个我自己印象最深刻的案例。2022 年卡塔尔世界杯期间，在比赛中或比赛后搜“国家名”、“国家名 + 世界杯”、“国家名 + 比分”，照理说 QP 应当识别出突发时效性，搜索结果页优先展示最新内容。但是我司的搜索结果非常令我失望，搜到的文档无法满足

我对时效性的需求，而两家主要的竞品的搜索结果都完全满足我的需求。这个案例说明，如果 QP 不能实时识别突发时效性、或提前预判突发时效性，那么用户体验会非常糟糕。

8.1.2 一般时效性

一般时效性是指查询词表达出对新文档的偏好，召回和排序应当扶持新文档。如果查询词带有一般时效性意图，它的字面就能体现出对时效的诉求。一个人只需有基本常识，不需要看新闻，就能判断出一般时效性意图——这是与突发时效性的主要区别。既然人不需要额外知识就能判定一般时效性，那么 BERT 等语义模型也能判断一般时效性，并且给时效性强弱做分档。按照查询词对“新”的诉求的强度，给一般时效性意图分为 4 档——强、中、弱、无。

强时效性意图，是值用户只想要新信息，不接受旧信息。在做召回和排序时，文档年龄的权重很高，甚至只召回年龄小于某个阈值的文档。强时效性的查询词具有某些特点：

- 查询词中有明确的时间节点，比如“新”、“最新”、“最热”、“最近”、“本周”。例如新款口红、最新交规。带有明确时间段的不算强时效性，比如带有“2021”、“2022 年 1 月 9 日”；对于这种查询词，可以通过文本匹配找到用户需要的文档，无需考虑文档的年龄。
- 查询词中带有某些特殊的词，这些词有隐含的时效需求，例如“政策”、“流程”、“时间”、“预约”、“折扣”、“薅羊毛”、“价格”、“天气”、“组队”、“交友”，这类信息更新频率高，旧的信息很快失去价值。例如回国核酸要求、某某基金价格、茅台股价、美元汇率走势、九价疫苗的预约、某某品牌的薅羊毛、李佳琦的直播时间、杭州天气。

中、弱时效性意图的意思是用户并不是只希望获得新信息，旧的有用的信息（例如攻略、评测、经验）也能接受。用户搜本地生活的查询词（例如“附近的餐厅”、“杭州租房攻略”、“洗牙诊所”）和测评、攻略类的查询词（例如“单反相机测评”、“新荣记探店”、“如何导出微信聊天记录”、“上海科技馆攻略”、“喀纳斯旅游”、“强化学习资料推荐”），目的是寻找有用的信息。如果如果文档较旧，则其提供的信息可能部分失效，比如搜到的餐厅已经停业，单反相机型号已经过时、不再销售。一条查询词究竟是归到中档还是弱档，要看信息的更新频率有多高、旧文档提供的信息价值大不大。比如“杭州租房攻略”、“单反相机测评”可以归到中档，因为租房、数码产品的信息刷新较快，很久之前的信息的价值不大。而“喀纳斯旅游”、“强化学习资料推荐”可以归到弱档，这类信息刷新较慢，两三年前的信息仍然有价值，尽管价值低于较新的内容。

很多查询词没有时效性的诉求，文档的新旧不影响文档为用户提供的价值。无时效性的查询词主要有下面三类。

- 用户希望获取客观存在的答案、或者变化频率很低的攻略。例如“怀孕的症状”、“宝宝湿疹怎么办”、“打耳洞注意事项”、“红烧肉怎么做”、“养柴犬的注意事项”。
- 用户想要找长期不变的资源（搞笑、萌宠、文案）和更新已完结或明确指向的资源（例如老的影视、文学作品）。例如“搞笑段子”、“可爱的猫”、“萌娃视频”、“母亲节文案”、“廊桥遗梦解说”、“红楼梦解说”、“武侠小说推荐”、“黄日华版射雕”。

- 查询词带有具体时间点，或带有过去事件的描述，这种查询词依靠相关性可以满足时效性。例如“08 奥运会金牌榜”、“射雕英雄传 83 版”、“汶川地震”、“2020 年的大事件”、“2006 年春晚小品”。

8.1.3 周期时效性

有一些查询词在每年的特定时间段表现为突发时效性，而在其他时间段无时效性，这种查询词被归为周期时效性。举个例子，在除夕夜及之后一段时间，用户搜“春晚小品”，很可能是想看今年的春晚小品，具有时效性诉求。而如果在一年的其他时间搜“春晚小品”，则没有时效性诉求，可能是想看历史上优秀的春晚小品，出《卖拐》这样的老作品也是符合用户需求的。

如果一个查询词及其衍生词（例如“春晚”、“春晚歌曲”、“春晚小品”）定期出现热度，且在出现热度时很多用户只想看新文档，则该查询词具有周期时效性。下面是最典型的周期性意图查询词。

- 节假日，比如圣诞、元旦、跨年、春节、元宵节、中秋节、情人节、母亲节、520、六一、儿童节、七夕、教师节、国庆、开学、寒假。上述节假日会伴随节日活动、购物、促销，导致查询词热度的增加。
- 每年的特殊活动，比如双十一、双十二、618、黑色星期五、各大平台集五福、春运。
- 每年定期的考试，比如高考、考研、国考、省考、CPA 考试、注会考试。
- 比赛和奖项，比如奥运、冬奥、残奥、世界杯、奥斯卡、金曲奖、金鸡奖、金像奖、金马奖。

值得注意的事，有一些查询词搜索的内容不具有时效性、不会过期，这样的查询词算无时效性，而非周期时效性。比如“母亲节文案”、“母亲节折纸”、“母亲节由来”这样的查询词，搜索的是素材、兴趣、爱好、手工、文案、知识，不会过时。

8.2 时效性数据

查询词时效性意图的识别主要有两种方法。突发时效性意图难以从字面上判别，只能做数据挖掘或人工标注。一般时效性意图可以从字面上看出，因此可以使用 BERT 这样的语义模型做判别。本节讨论这两类问题和方法。

8.2.1 突发时效性

上一节讨论过，如果一个人有背景知识，但是不看新闻，他没有能力根据查询词的文字判断出突发时效性意图，因此语义模型不适用于突发时效性。识别突发时效性的主要手段为数据挖掘，具体有如下几种方法。

- 检测查询词的搜索量，如果从某一时刻开始突然激增，则该词很可能带有突发时效性意图。例如，明星王某某离婚，那么“王某某”、“王某某离婚”等查询词的搜索量会暴增，比前一天高很多倍。

- 对于小红书、抖音、微博这样的 UGC 平台，可以监控自己站内发布的文章的关键词的数量，如果某个关键词激增，则该关键词对应一个热点事件。例如，某一天站内新发布文档的关键词“王某某”、“王某某离婚”数量激增，则包含这些词的查询词很可能带有突发时效性。
- 监控站外的热榜，例如百度、微博、抖音、头条，这些热榜上的查询词都带有突发时效性。

此外，某些热点事件可以被预判，在事件发生之前由运营人员手动配置查询词的突发时效性意图。举个例子，在世界杯期间，运营人员可以根据赛事日程做配置，比如让“巴西”和“克罗地亚”及其衍生词在 2022 年 12 月 9 日前后具有突发时效性。

8.2.2 一般时效性

带有一般时效性的查询词具有一些特点，比如显式带有“最新”这样的词根，或隐式包含“特价”、“政策”这样特殊的词。具有背景知识的人无需关注新闻热点，只需根据查询词的文本即可推断出查询词是否带有一般时效性以及强弱。既然人可以做到，BERT 等语义模型也可以根据查询词文本推断一般时效性及其强弱。训练语义模型的数据有两个来源：算法挖掘到的大数据，以及人工标注的小数据。

图 8.2: 搜索结果页示意图

用户的行为反映出查询词是否带有时效性意图、以及意图的强弱。搜索结果页的界面如图 8.2 所示，搜索框的下方有“最新”筛选项，文档下方有文档年龄标签。查询词的时效性意图越强，则用户越有可能点击“最新”筛选项，且越新的文档的点击率越高。通过挖掘搜索日志，可以获得每条查询词 q 下“最新”筛选项的点击率、各年龄段文档的平均点击率，组成数值向量 $\boldsymbol{x}_q$。按照 5.2 节后预训练的方法，用人工标注的少量样本（数千条查询词）训练一个教师模型 $t(\boldsymbol{x}_q)$，它可以根据统计值 $\boldsymbol{x}_q$ 估计时效性档位（强、中、弱、无）。训练好教师模型之后，用它对挖掘到的数据 $\{(q, \boldsymbol{x}_q)\}$ 做标注，获得大规模的数据，用于训练 BERT 模型。最终，用人工标注的小规模数据（数万条查询词）微调 BERT 模型，训练好的 BERT 模型用于线上推理。

8.3 下游链路的承接

搜索的 QP 模块负责分析查询词是否带有时效性意图、以及意图的强弱，下游的召回和排序做相应的承接。如果查询词有时效性意图，那么搜索结果应当偏向新内容；时效性意图越强，则召回和排序越侧重“新”。文档的“新”主要是以发布时间、最近一次更新

的时间为依据。

为了承接突发时效性、强时效性意图的查询词，或是承接“最新”筛选项，文本召回和向量召回需要建单独的新文档索引，比如包含 24 小时、7 天的新文档。假如新文档没有独立的索引，而是在全量的索引上先召回、后筛选，那么效率会低很多。甚至一些新文档可能会因为相关性、内容质量略逊于老文档，无法被召回。举个例子，某关键词 q 热度很高，有数十万篇文档与 q 高相关，但是文本召回限定最多 5 万篇文档送入海选，否则计算量过大。不论查询词是否有强时效性意图，大部分新文档没有机会进入海选阶段。如果新文档有单独的索引，那么就可以优先召回高相关性的新文档，数量不足再用全量索引的召回结果补齐。

在召回之后，会有海选、粗排、精排，它们的依据都是相关性、时效性、内容质量、个性化。查询词的时效性意图越强，文档年龄在排序中的权重就越高。如果查询词有突发时效性、强时效性意图，那么在保证相关性的前提下，文档年龄是排序最重要的因子。如果查询词没有时效性诉求，那么文档年龄不作为排序的因子。

对于周期时效性查询词，在相应的时段按照突发时效性对待，在其他时段按照无时效性对待。如果在 11 月初搜“双十一购物”，用户很可能想看近期各电商平台的优惠活动，算是突发时效性，召回和排序重点出一两周的新文档。如果在 5 月搜“双十一购物”，用户可能是想看各平台往年的双十一数据分析，但不可能是想看近期各电商平台的优惠活动，文档年龄不应当作为召回和排序的主要依据。

8.4 知识点小结

- 时效性可以指查询词的意图，也可以指整条链路对“新”的需求的满足程度。查询词的时效性意图分为突发时效性、一般时效性、周期时效性。搜索引擎时效性的好坏既取决于 QP 对意图识别是否准确，也取决于召回和排序对时效性意图的承接。
- 突发时效性意图主要针对热点事件，反映在相关查询词的搜索次数、相关文档的发布次数激增。人和语义模型都无法根据查询词字面判断突发时效性，只能通过数据挖掘或人工配置的方式识别突发时效性。
- 一般时效性意图分为强、中、弱、无这四档，该意图反映在查询词的字面上，人和语义模型都可以根据文字判断查询词是否具有该意图、以及意图的强弱。
- 周期时效性意图是指一些查询词周期性出现突发时效性的特点，而在其他时间段无时效性。
- 在 QP 模块识别出时效性意图之后，下游的召回和排序需要做相应的承接。在召回模块，独立的新文档索引可以让新文档召回变得更高效。在排序模块，根据时效性的种类和强弱，给文档年龄设置不同的权重。

第 9 章 地域性

用户使用手机 APP 做搜索时，有时会带有搜附近、搜同城的意图。举个例子，用户在小红书上搜“附近的酒店”、“情人节餐厅”、“周边游”、“二手房”，有可能是想看附近或者同城的信息，那么距离的远近、是否同城应当作为召回和排序的依据。假如不对这类查询词做特殊处理，则搜出的结果会严重不符合用户的需求。举个例子，用户在上海，搜“附近的酒店”，搜索结果为“北京前门附近的酒店”，显然不符合用户的需求。

本章介绍我们在小红书落地的“本地内容搜索”项目，从 0 到 1 初步解决这类问题。本章内容分 5 节，分别介绍项目中的几个关键点。小红书的笔记可能带有地理位置，这个信息是地域性意图搜索的关键。查询词处理（QP）是链路上的第一环，QP 判断查询词带有什么样的意图、以及意图的强弱，QP 的结果决定如何调用下游的链路。召回是链路上的第二环，它根据本地意图的类型、查询词文本检索相关的文档，并做初步的筛选。排序是链路上的最后一环，它根据本地意图的类型和强弱、以及文档本身的价值对文档做排序，决定最终的曝光顺序。本章最后一节介绍我们的实验结果与分析。

9.1 POI 识别

point of interest（POI）是指电子地图上的景点、商铺等地标，每个 POI 都会绑定一个经纬度坐标。小红书有相当一部分笔记带有 POI，比如景点攻略的笔记会带有这个景点的 POI，餐厅探店的笔记会带有这家餐厅的 POI。作者在小红书发布笔记的时候，可以自己标记一个 POI，这个 POI 是比较置信的。但作者标的 POI 也存在问题，有一定比例的 POI 是误标的，比方说作者去三亚旅游，回到上海之后写旅游攻略，随手选了自己家的地点作为 POI，这样会误导搜索引擎。作者自己标注的 POI 覆盖率较低，很多作者不知道这个功能，或者懒得自己标记 POI，这会导致搜索的召回量不足。

为了解决上述问题，还需要用 NLP 算法根据笔记文本提取 POI。举个例子，笔记中介绍了三亚的天涯海角，那么算法会提取出这个 POI。假如作者自己没有标记 POI，那么就可以把三亚天涯海角作为笔记的 POI。假如作者自己标记的 POI 在上海，与算法提取的 POI 没有重合，那么可以判定作者标注的 POI 是错的，不予采用。

小红书实际的业务场景中有 9 种 POI，比如用户自己选定的 POI、审核人员强定的 POI、用户发布时 GPS 定位所在的 POI、NLP 算法从文字中提取的 POI、CV 算法做图片识别提取的 POI、等等。笔记内容理解团队需要对 9 种 POI 做融合，输出最置信的一个或多个 POI，作为召回和排序的依据。

9.2 查询词处理

大众点评的本地内容搜索比较容易做，因为用户在大众点评上输入的查询词必然带有地域性意图。而小红书这类通用搜索引擎则有所不同，用户在小红书上输入的查询词带有各种各样的意图，我们需要判断查询词是否带有地域性意图，如果有，则触发相应

的搜索链路。本节的主要内容是如何对地域性意图做细分，以及如何自动挖掘查询词的意图。

9.2.1 附近意图与同城意图

大家思考一个问题：“附近的餐厅”和“周边游”显然都带有本地意图，那么应当用相同的召回和排序策略承接两个查询词吗？答案是否定的。当用户搜“附近的餐厅”时，通常希望找离自己非常近的，最好是步行就能到达的，1 公里跟 3 公里有很大差别。而当用户搜“周边游”时，用户大概率是想开车出游，因此对距离并不敏感，10 公里跟 30 公里对用户来说没有很大差别。不难看出，两个查询词背后的意图有所差异，一个对距离敏感，一个对距离不敏感，因此召回和排序要用不同的方式承接。

基于以上讨论，我们将地域性意图拆分为两个大类，分别是“附近意图”和“同城意图”，它们的区别在于召回和排序是否对距离敏感。附近意图的意思是用户希望寻找定位在自己附近的 POI，因此召回和排序需要把距离作为一个重要因子。同城意图的意思是用户希望寻找某个地区的 POI，只限定市或区，而对距离不敏感，用户在海淀区搜“北京周边游”，可以出石景山区的 POI。由于上述区别，两种意图的下游承接方式有很大差异，无法用统一的召回和排序链路承接。因此，我们搭建了两条搜索链路，一条是附近意图，另一条是同城意图。

9.2.2 意图的细分

我们将地域性意图拆分为“附近意图”和“同城意图”两大类，但它们仍需要细分为“显式”和“隐式”，两种细分意图的处理方式有所不同。两类意图及其细分概括如下。

- 附近意图：用户对距离敏感，召回和排序需要把距离作为重要因子。
  - 显式：用户明确表达了附近意图，例如“我附近的酒店”、“附近的餐厅”、“最近的咖啡店”。对于显式附近意图，仅召回距离小于一定阈值的。如果符合要求的笔记数量超过阈值时，距离近的优先。值得注意的是，QP 需要对这类词做改写，例如把“附近的餐厅”改写成“餐厅”，因为召回的笔记中无需包含文字“附近的”。
  - 隐式：用户没有明确表达附近意图，但是查询词隐含了附近意图，例如“火锅”、“洗牙”、“情人节餐厅”、“去哪遛娃”。但这些词的意图并不唯一，用户搜“火锅”，可能是想看附近的火锅店的探店点评，也可能是想看在家做火锅的教程。隐式附近意图大多是多意图的，既需要附近意图的召回，也需要非地域性的召回，最终对召回的结果做多队列混排。
- 同城意图：用户对距离不敏感，只需要在召回中限定城市，而不需要考虑距离。
  - 显式：用户明确指定了在某个地区做搜索，例如“北京周边游”、“杭州二手房”、“青岛买家具”。对于显式同城意图，即便不特意开发同城意图链路，也能用标准的文本召回较好地解决问题，因为有大量笔记带有“北京周边游”这样的文本。同城意图链路可以作为召回的补充，例如笔记描述介绍了作者在某商场买家具的攻略，笔记中没有明说是在青岛，但是 POI 定位在青岛，那么这篇笔记可

以被同城意图链路搜到，而不会被文本召回搜到。值得注意的是，如果想要调用同城意图链路，需要 QP 做改写，比如把“北京周边游”改写成“周边游”，之后才能根据 POI 定位所在城市做检索。

- 隐式：用户没有明确指定在什么地区做搜索，但是由于查询词隐含了同城意图，例如“周边游”、“租房”、“同城相亲”、“请月嫂”、“本地景点”。对于隐式同城意图，我们把搜索范围确定在用户所在城市、或距离小于某个阈值。比如用户在上海搜“租房”，搜出的笔记要么明确写“上海租房”，要么写“租房”且 POI 定位在上海。对于“本地景点”这样的查询词，需要根据用户所在城市做改写，比如改写成“上海景点”。

9.2.3 意图的强弱

以附近意图为例，查询词存在多重意图，在召回和排序阶段，需要控制附近笔记的占比，占比取决于附近意图的强弱。举个例子，“附近的酒店”只有附近意图，没有其他意图，搜出的结果应当全部是用户附近的笔记。“日料店”和“星巴克”都带有附近意图，但是也有其他意图，搜出的结果应当具有多样性，其中一部分笔记定位在用户附近，其余笔记无需考虑距离和城市。同样是多意图查询词，“日料店”和“星巴克”的附近意图强弱是不同的。“日料店”的附近意图较强，用户有较高概率想找附近的日料店。而“星巴克”的附近意图较弱，用户很可能是想看星巴克最新的杯子和饮品，但如果能出几个定位在用户附近的星巴克店铺，也会给用户带来一些惊喜。在附近笔记供给充足的前提下，附近意图越强，则搜索结果中附近笔记的占比应当越高。

QP 该如何判断查询词意图的强弱呢？“附近的酒店”这种显式附近意图很好处理，它们的附近意图强度是 100%，不存在其他意图。而“日料店”、“星巴克”这样的隐式附近意图不容易处理，它们的附近意图强度介于 0 和 100% 之间，需要人工标注或者用算法自动挖掘。对于高频查询词，可以制定一个分档标准，用人工标注附近意图的强弱。但是搜索中有大量的长尾查询词，它们的数量巨大，但是出现的频次较低，人工标注的 ROI 不高，不如用算法做挖掘。

算法挖掘的基本想法是用小流量做随机试探，判断附近意图的强弱。具体来说，如果我们不确定查询词 q 是否带有附近意图或同城意图，那么可以开一个小流量实验，比方说 10% 的流量带有“附近”筛选项，10% 的流量带有“同城”筛选项。如果用户想要找附近或同城笔记，且搜索结果中没有想要的结果，那么用户倾向于点击“附近”或“同城”筛选项。在实验一段时间后做数据分析，统计筛选项的点击率，即可判断 q 是否带有附近意图或同城意图。

如果我们知道查询词 q 带有附近意图，但不清楚意图的强弱，那么我们可以做小流量实验推断出意图的强弱。比方说 10% 的随机流量把 q 按照附近意图的链路承接，在搜索结果中有一部分附近的笔记，且前端界面给附近笔记打上距离标签。用户的附近意图越强，则带有距离标签的笔记点击率越高，且距离越近则点击率越高。在实验一段时间后做数据分析，根据点击率这样的后验指标，可以推断出 q 的附近意图强弱。

9.3 召回

QP 环节识别查询词意图、并判定意图强弱。QP 将意图类别、意图强弱分数传递给召回。我们新添加了两条召回通道，分别对应附近意图和同城意图。最简单的工程实现方式为先做召回，然后根据距离或城市做过滤，保留 POI 为附近或同城的笔记。新添加的召回通道会被用于解决 4 种细分意图。

- 对于显式附近意图，例如“附近的餐厅”，只需要调用附近意图的召回链路，把“餐厅”作为查询词，检索用户附近的笔记。
- 对于隐式附近意图，例如“餐厅”，需要调用两类召回链路。一类是附近意图的召回，另一类是非地域性的召回（即标准的文本召回和向量召回），这样才能满足多重意图的需求。召回的总量是有上限的，需要根据附近意图的强弱给两类召回通道设定配额。
- 对于显式同城意图，例如“北京周边游”，需要调用两类的召回链路。一类是非地域性的召回，即把“北京周边游”作为查询词，调用标准的文本召回和向量召回。另一类是同城意图的召回链路，即把“周边游”作为查询词，从召回结果中筛选所在城市为北京的笔记。在前者召回量不足的情况下，后者可以起到补充的作用。
- 对于隐式同城意图，例如用户在北京搜“周边游”，也需要调用两类召回链路。一类是非地域性的召回，把“周边游”改写为“北京周边游”，调用标准的文本召回和向量召回。另一类是同城意图的召回链路，即把“周边游”作为查询词，从召回结果中筛选所在城市为北京的笔记。

召回中有很多需要解决的问题，此处列举一些。对于附近意图，召回需要卡一个距离阈值，比如 30 公里，即排除掉 30 公里以外的 POI。但这会导致召回量过多或过少。假如用户在上海市中心搜索“附近的美食”，30 公里的阈值会导致召回量过大；假如在偏远乡村做搜索，30 公里的阈值会导致召回量不足。简单的解决策略是在发现召回量过大之后做进一步过滤，在发现召回量过小之后扩大阈值做二次召回。更优、但是更复杂的策略是在 QP 阶段即预设好阈值，设置阈值的依据是查询词与用户定位。查询词能召回的结果越多则距离阈值越小，举个例子，“附近的美食”需要的距离阈值小，而“附近的电影院”、“附近的日料店”需要的距离阈值大。用户定位到的 GeoHash 越繁华，则距离阈值越小，举个例子，上海市中心的 POI 密集，需要的距离阈值小，而偏僻的地方 POI 少，需要的距离阈值大。

召回中有多样性的问题。举个例子，搜“附近的美食”，恰好附近有一家网红餐厅，召回的结果有一半都是对这家网红店的探店点评。很显然，曝光给用户的笔记应当具有 POI 多样性，不能曝光多篇同一家店的笔记。在召回时，需要根据 POI 做去重，比如同一个 POI 的笔记不能超过 10 篇，否则保留 5 篇最新的，保留 5 篇点赞最多的。

9.4 排序

对于附近意图，排序需要解决两个问题。第一，排序阶段的 LTR 给每篇笔记打了一个分数，这个分数表示笔记的价值。对于附近意图，排序需要综合考虑距离与笔记价值。第二，查询词可能具有多重意图，比如用户搜“火锅”，可能是想找附近的火锅店，也有可能想看在家做火锅的教程。这就涉及到多队列混排的问题，召回的附近笔记是一个队列，其余笔记是另一个队列。两个队列的分数不一致，具体来说，附近笔记的排序分数由价值和距离组成，而另一个队列笔记的分数仅包括价值。

9.4.1 附近意图队列内排序

召回的笔记可以包括用户附近的，也可以包括其他意图的。我们首先讨论附近意图的队列，即其中所有的笔记都位于用户附近。设每篇笔记有两个分数。一个分数是 LTR 给笔记 i 打的，记作 reward_i，它是对相关性、内容质量、时效性、个性化分数的融合，表示笔记本身价值。另一个分数是笔记与用户的距离，记作 dist_i，距离越近，对用户的效用越大。

排序的难点在于如何融合 reward_i 与 dist_i，两者的量纲不同，无法简单地做加减乘除。不同的搜索场景下，距离的尺度的范围很大。搜“附近的餐厅”，一公里范围内通常会有很多 POI；搜“附近的电影院”，十公里之内也没有几家。同样是搜“附近的电影院”，在上海市中心做搜索，一公里之内有好几个 POI；假如在远郊做搜索，最近的 POI 也在几公里之外。假如我们简单地以 $\text{reward}_i - \lambda \cdot \text{dist}_i$ 作为排序依据，并将超参数 λ 调优，会发生什么呢？如果召回的笔记的 dist 都比较小，比如都在 0.1 到 2 公里范围内，则距离没有多少区分度。如果召回的笔记的 dist 偏大，比如都在 5 到 20 公里的范围内，则距离的影响太重，导致 reward 起不到作用。

出于以上的考虑，我们对召回的物品的 dist_i 做升序排列，用序号（记作 rank_i）作为排序的依据，而不是用 dist_i 本身的数值。我们用下面的公式融合 reward_i 和 dist_i：

$$\text{score}_i \;=\; \lambda \cdot \text{reward}_i \;+\; (1-\lambda) \cdot \frac{1+\beta}{\text{rank}_i^{\alpha} + \beta}.$$

上式中的 $\alpha \in (0,1)$、$\beta \geq 1$、$\lambda \in (0,1)$ 是需要调的超参数。用这种方式，我们排除掉了距离的尺度对排序结果的影响，只考虑距离的序。在一期项目中，方便起见，设置 $\alpha = 1$、$\beta = 0$。

9.4.2 多队列混排

查询词可能具有多重意图，这就带来多队列混排问题。召回的附近/同城笔记是一个队列，其余笔记是另一个队列。比如用户搜“火锅”，可能是想找附近的火锅店，也有可能想看在家做火锅的教程。用户搜“博物馆”，可能是想找同城的博物馆，也有可能是想看世界各地的博物馆。两个队列的分数不一致，不容易公平比较分数大小。

- 对于附近意图，附近笔记队列的分数 score 是对价值 reward 和距离 dist 的融合，而另一条队列的分数就是 reward 本身。score 和 reward 的均值、方差可能会有显著

的不一致，导致两条队列不能公平做比较。如果直接用分数做排序，可能排名高的几乎都是其中一个队列的。

- 对于同城笔记，排序无需考虑距离，两个队列的分数都是 reward，可以直接比较大小。但是存在一个问题，与全量的内容池相比，同城的内容池要小得多，那么同城笔记的 reward 会相对偏低。比方说用户在北京搜“博物馆”，北京的博物馆笔记远远少于全网的，如果按照 reward 排序，排名高的笔记中只有很少一部分是位于北京的。

有两种方法解决上述问题。第一种方法是给一条队列的分数乘或加上一个权重，对这个权重做精调，使得两个队列的分数可以公平比较。第二种方法是对分数做变换，比如分别把两个队列的分数转化成序号 rank_i 和 rank'_j，比较 $\frac{1+\beta}{\text{rank}_i^{\alpha}+\beta}$ 和 $\frac{1+\delta}{\text{rank}'^{\gamma}_j+\delta}$ 的大小，这样就可以让对比变得公平。

在设置权重、或是对分数做变换时，还需要考虑附近意图、同城意图的强弱。比方说“附近的餐厅”是单意图，即附近意图的强度是 100%；“情人节餐厅”的附近意图较强，但仍有其他意图；“喜茶”的附近意图较弱，用户搜这个词同城不是为了找附近的喜茶，但出少量附近的笔记没有任何负面影响。附近意图越强，则附近意图队列的分数应当被调得更高，让更多附近的笔记排在前面。在一期项目中，QP 侧尚未推断意图强弱，所以排序中没有使用意图强弱分数。

最后要解决的一个问题是同队列笔记的打散，一期项目还没有做。比方说用打散策略“不能连续出 k 篇附近的笔记”。举个例子，位置 i 到 $i+k-1$ 的都是附近队列的笔记，那么位置 $i+k$ 的必须是另一条队列的笔记。策略中的 k 应当取决于意图的强度，附近意图越强，则 k 设置得越大。一期项目中尚未实现打散策略。

9.4.3 排序策略 vs 排序模型

上文都是对排序策略的讨论，即给定排序模型的打分 reward，结合距离 dist 得到融合分数，根据融合分数对笔记做排序。一种不同的方法是将 dist 作为 LTR 的一个特征，使用用户的点击和交互行为训练模型，让模型学会如何对附近的笔记做排序。理论上，如果训练数据够好，那么模型的效果会优于人工制定的策略（比如融分公式）。

但是我们的一期项目只能使用策略，而不能使用模型。原因是这样的。在一期项目上线之前，不论查询词的附近意图有多强，搜索结果中都不会有附近的笔记，用户不可能点击和交互附近的笔记。如果给 LTR 模型加上距离特征，学到的距离的特征重要性也会是 0。因此，在一期项目中，应当使用策略让附近笔记获得充分的曝光，当用户搜出附近的笔记、且看到距离标签时，可能会发生点击和交互行为，这些行为数据被记录下来。用户熟悉小红书本地内容搜索的产品形态（比如外显距离和城市标签）之后，用户的行为会趋于稳定，届时就可以使用用户行为数据训练 LTR 模型，用模型打分代替策略。

9.5 实验结果

一期的实验取得了显著的业务指标收益。对于显式附近意图、隐式附近意图、隐式同城意图，关键词有效点击率分别增长 8.5pt、1pt、1.4pt。（通常来说，一个有效的策略能带来约 0.1pt 有效点击率的增长。）显式同城意图的有效点击率小幅下跌，故没有推全。

对于附近/同城意图队列的笔记，我们在前端 UI 界面上展示距离和 POI 名，帮助用户决策是否点击进入笔记。实验结果表明外显距离和 POI 标签会导致有效点击率下跌 0.3pt，但这并不能说明用户对外显标签的 UI 界面不满意。有效点击率下跌的原因是用户可以根据距离和 POI 做决策，而无需点击进入距离远的笔记，其实对用户是有帮助的。如何验证这个假设呢？虽然有效点击率下降，但是深度消费占比（有效点击数 / 总点击数）增长 0.2pt，这说明标签外显减少了无效点击。

第 10 章 个性化与点击率预估

搜索引擎与推荐系统有一大共通之处：在实际将文档给用户曝光之前，就能用模型预估出用户点击、交互文档的概率。工业界将这样的模型称作点击率模型（即便它同时预估交互率、完播率等多个目标）。在推荐系统中，用户 u 的历史行为记录越多，文档 d 的曝光、点击、交互次数越多，输入模型的信息就越充分，模型就能越准确地预估 (u, d) 的点击率和交互率。搜索引擎用神经网络根据 (u, q, d) 的特征，预估点击率和交互率，作为排序的因子。

对点击率、交互率做预估，起到两方面的作用。第一，相关性和内容质量的模型主要依据查询词和文档的文本做预测，无法做到完全准确，有一定的概率会犯错。根据我们的实践经验，仅靠相关性和内容质量模型，会出现一定比例的 bad case，而且这类 bad case 很难靠文本特征解决。好在搜索引擎用海量的用户行为，用户用自己的点击和交互投票，选出与查询词相关、且内容质量高的文档。我们利用海量的用户行为训练出点击率模型，它利用多种非文本特征，预估点击率和交互率。结合相关性、内容质量、预估的点击率和交互率，排序结果会好很多，高频查询词搜出的 bad case 极少。

第二，对于“头像”、“耐克”这样相对宽泛的查询词，高相关的文档数量巨大，如果搜索引擎不理解用户的真实需求，很难将令用户满意的文档排在前面。神经网络使用用户画像、用户历史行为记录作为特征，预估用户对文档的点击率、交互率，这些分数可以让符合用户兴趣的文档排名靠前。根据我们的实践经验，在召回和排序中优化个性化，可以提升有点比、首点位置、用户留存指标，不会损害人工体验评估。

精排和粗排必须点击率模型，甚至召回截断也有较弱的点击率模型。对点击率模型的优化是提升有点比、文档点击率、首点位置的最直接有效的途径。本章详细讲解点击率模型的特征、模型结构、训练方法。

10.1 特征

按照来源，特征可以分为查询词特征、文档特征、用户特征、场景特征、统计特征。按照性质，特征可以分为数值特征和离散特征。10.1.1 节按照来源介绍点击率模型使用的特征。10.1.2 节介绍特征交叉的方法。10.1.3 节介绍用户行为序列特征的建模。

10.1.1 特征来源

查询词特征包括文本特征、查询词的类目、查询词的意图（时效性、地域性的诉求）。QP 中的分词器将查询词切分成多个词，嵌入层将每个词表征为一个向量，向量的平均（或加权平均）作为一种特征。文本特征还包括整条查询词的向量表征，它是用 BERT 模型离线计算出的，存储在哈希表中，供线上读取。QP 产出的查询词类目、意图也属于离散特征，经嵌入层表征向量。根据我们的经验，查询词的文本特征是点击率模型中最重要的特征，没有之一。

文档特征包括文档 ID、作者 ID、文档类目、命名实体、地理定位、发布时间（或更新时间）、多种内容质量分数。文档 ID 和作者 ID 经嵌入层表征为向量，由于文档 ID 和作者 ID 数量巨大，对训练和推理的要求都比较高，需要特殊的机器学习系统支持。文档的类目、命名实体、地理位置等离散特征经嵌入层表征为向量，它们与用户画像共同反映用户是否对文档感兴趣。文档的发布时间反映内容有多新，与时效性挂钩，会影响文档对用户的价值。虽然文档发布时间是数值特征，我们通常对其做分桶，转化为离散特征。内容质量分是数值特征，对点击率模型有微弱的贡献。

相关性特征。特征需要链路上游传递，所以精排点击率模型只能用粗排相关性特征，粗排点击率模型只能用召回截断的相关性特征。

用户特征包括用户 ID、用户画像、用户历史行为序列。用户 ID 被嵌入层表征为一个向量，向量是模型从历史数据中学到的，相当于记录了用户的兴趣点。用户画像包括用户性别、年龄、感兴趣的类目和命名实体，这些离散特征同样被嵌入层表征为向量。用户历史行为序列（包括最近搜过的 n 个查询词、点击过的 n 篇文档、交互过的 n 篇文档）反映出用户的兴趣点，被表征为多个向量，具体方法在 10.1.3 节讨论。用户最近搜索过的 n 个查询词、点击和交互过的 n 篇文档被输入 DIN 或 SIM 模型，得到的向量作为模型的输入。并非所有的搜索引擎都有个性化，比如百度网页搜索的用户通常不登录，即便登录，用户历史行为也不够丰富，因此点击率模型不使用用户特征。

场景特征包括当前时刻、是否是周末或节假日、用户所在地点、手机和平板设备的信息。用户在每天不同时间段想搜的内容可能不同，在工作日和周末、节假日想搜的内容也可能不同，所以需要时间特征。用户对自己所在地的内容更感兴趣，所以需要地点特征。我们在实践中发现，iOS 用户与安卓用户的点击率等数据有显著差异，因此手机等设备的品牌、型号、操作系统都被作为特征。

统计特征包括用户统计特征、文档统计特征、查询词—文档统计特征。统计特征被称作“后验特征”，原因是这些特征是用户发生点击、交互行为之后产生的。也就是说，历史上已经发生的点击、交互可以反过来预测尚未发生的点击、交互。下面列出一些典型的用户统计特征。

- 用户统计特征包括最近 30 天、1 天、1 小时的曝光数、点击数、交互数、点击率、交互率。有的用户有频繁交互（点赞、评论、转发）的习惯，他们的交互指标比均值高很多，而有的用户的交互指标会低于均值。这种偏差不利于模型的训练，使用用户统计特征可以让模型更好拟合训练数据。
- 按照文档类目做分桶，比如最近 30 天，该用户对美食文档的点击率、对科技数码文档的点击率。如果用户对科技数码的点击率高于其他类目，可以说明用户偏好科技数码类目。

下面列出一些典型的文档统计特征。

- 文档统计特征包括最近 30 天、1 天、1 小时的曝光数、点击数、交互数、点击率、交互率。文档统计特征反映文档本身的内容质量，内容质量越好，统计指标越高。
- 按照点击文档的用户做分桶，比如最近 30 天，男性用户对这篇文档的点击率、女性用户对这篇文档的点击率。比如一款粉色键盘，女性用户的点击率显著高于男性

用户，统计特征可以说明文档更受哪类用户偏爱。

- 作者粉丝数、近 30 天发布文档数量、文档点击数、文档交互数反映作者的受欢迎程度、以及他的作品的平均质量。如果一位作者已有作品的质量平均很高，他新发布的作品大概率也会质量很高。可以将作者统计特征看做对 EAT 的补充。

查询词—文档统计特征的意思是当查询词为 q 时，文档 d 的点击率和交互率。可以将该特征视作 (q,d) 相关性的补充，通常来说，q 与 d 相关性越高，则该统计特征值越高。查询词—文档统计特征是所有统计特征中重要性最高的，但它也是存储成本最高的。用户数量通常在亿的量级，文档数量在十亿量级，它们的统计特征存储成本不高。而查询词—文档交叉的数量巨大，系统无法存储全量的查询词——文档统计特征，只能存储其中一部分。这会造成排序对新文档不友好，如果文档 d' 以前没有被 q 检索到，那么就没有存储 (q,d') 的统计特征，导致模型预估的点击率、交互率都不太准确。

10.1.2 特征交叉

在上述原始特征的基础上，做特征交叉得到新的特征，可以让点击率模型的预估变得更准确。可以由算法工程师根据经验，选取关联的特征做交叉。以类目特征为例，用户画像中有用户感兴趣的类目 t 及其兴趣分数 a_t，文档画像中也有类目及其置信度 b_t，计算乘积 $a_t b_t$，再关于所有类目取连加 $\sum_t a_t b_t$，得到的实数作为新的特征。

2019 年的论文[7] 使用域间的离散特征交叉方法，将其用于推荐系统的点击率模型。我们团队使用双线性交叉的方法，用在搜索精排点击率模型，取得了有点比等指标的显著收益。每个离散特征被称作一个域（field），它被嵌入层表征为一个向量。举个例子，用户 ID 被表征为 64 维向量，其中 64 个元素属于一个域，而用户 ID 和用户性别属于两个域。图 10.1 中有三种特征交叉的方法，分别是内积（inner product）、哈达玛乘积（Hadamard product）[1]、双线性交叉（bilinear cross）。图中的 $\boldsymbol{x}_i$ 和 $\boldsymbol{x}_j$ 是两个离散特征的向量表征，属于两个域，得到的实数 f_{ij} 或向量 $\boldsymbol{f}_{ij}$ 是特征交叉的结果，作为点击率模型的输入。其中双线性交叉有 m^2 个矩阵 $\{\boldsymbol{W}_{ij}\}$，它们是模型参数，在模型训练的过程中更新。

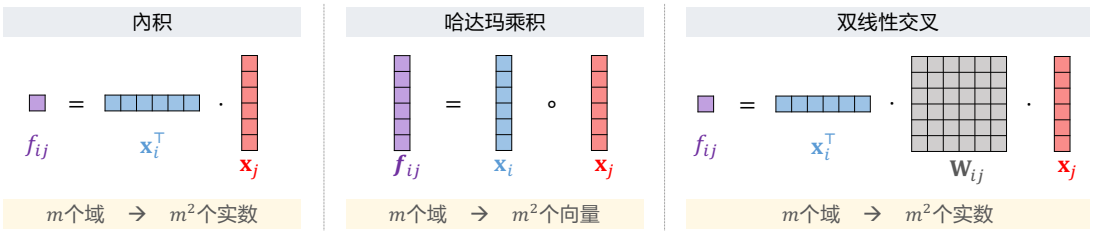

图 10.1: 三种域间特征交叉的方法

2021 年的论文[16] 提出深度交叉网络（deep & cross network，DCN）[2]，这也是一种自动做特征交叉的方法，既可以用于精排点击率模型，也可以用于粗排和召回的双塔点

[1]哈达玛乘积的意思是两个形状相同的向量逐元素相乘，输出相同形状的向量

[2]这篇文章其实是 DCN 的第二版，效果优于 DCN 的第一版[15]

击率模型。DCN 由一个交叉网络和一个全连接网络并联组成，两个神经网络都将 $\boldsymbol{x}_0$ 作为输入，各自输出一个向量，将两个向量拼接起来作为 DCN 的输出。交叉网络的结构如图 10.2 所示，它由若干个交叉层组成。第 i 个交叉层的结构如图 10.3 所示，它的输入是 $\boldsymbol{x}_0$ 与 $\boldsymbol{x}_i$，参数是 $\boldsymbol{W}_i$ 与 $\boldsymbol{b}_i$，输出是 $\boldsymbol{x}_{i+1}$。之所以称之为交叉层，是因为在用全连接层对 $\boldsymbol{x}_i$ 做变化之后，计算输出的向量与 $\boldsymbol{x}_0$ 的哈达玛乘积，即两个向量的交叉。

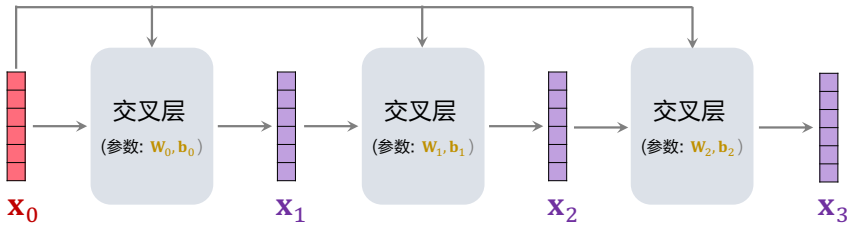

图 10.2: 交叉网络由多个交叉层组成

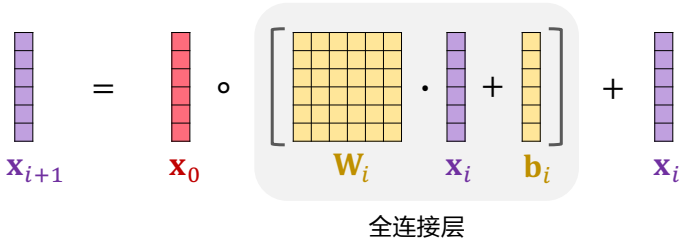

图 10.3: 第 i 个交叉层的结构

10.1.3 用户行为序列建模

系统记录用户的最近的搜索、点击、交互行为，这些行为属于用户个性化特征，反映用户的兴趣点。以用户的点击行为为例，如图 10.4 所示，系统记录用户最近点击过的 n 篇文档，嵌入层将文档 ID 映射到 n 个向量，对它们取平均，作为用户点击行为的向量表征。用同样的方法，对用户的点赞（收藏、转发、评论）的文档 ID 做嵌入、取平均，得到点赞（收藏、转发、评论）行为的向量表征。每种行为（点击、点赞、收藏、评论等）被表征为一个向量，连接这些向量，作为用户用户特征输入神经网络。

对于用户最近搜索过的 n 条查询词，处理方法略有区别。一种方法是用 BERT 模型离线计算头部数千万条查询词的向量表征，存储在哈希表中。如果用户历史记录中的查询词不在哈希表中，直接忽略查询词。另一种方法是固定一个词典，词典中每个词有一个向量表征，存储在哈希表中。对每条查询词做分词，得到若干个词，每个词表征为一个向量，取平均作为整条查询词的向量表征。

提升的方法：加长序列长度、用 ID 以外的多种特征、DIN。同时使用短期行为序列和长期行为序列，比如用最近 1 小时的、最近一年的，每种用户行为序列被表征为两个向量。

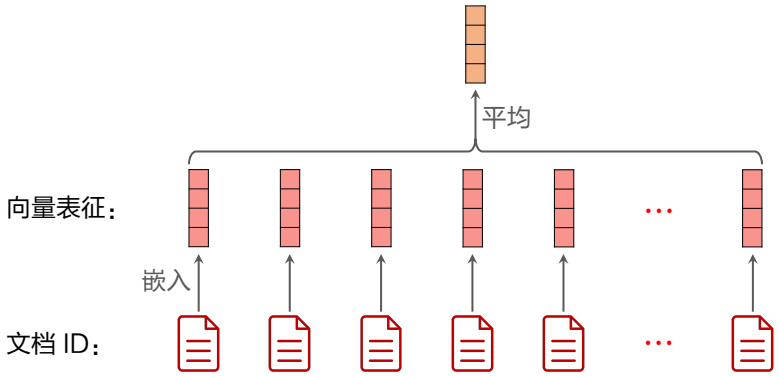

图 10.4: 用户行为序列

Deep Interest Network（DIN）是阿里巴巴在 2018 年发表的论文[21]，已经被广泛用于工业界的推荐系统和广告的排序。类似的思想也可以用于搜索的召回和排序。如图 10.5 所示，DIN 的本质是单头自注意力层。将用户行为序列（比如搜索的 n 个查询词、点击的 n 篇文档）的向量表征作为 K 和 V，将查询词作为 Q，输入单头自注意力层，在 Q 的位置上输出一个向量，作为用户一种行为的表征。

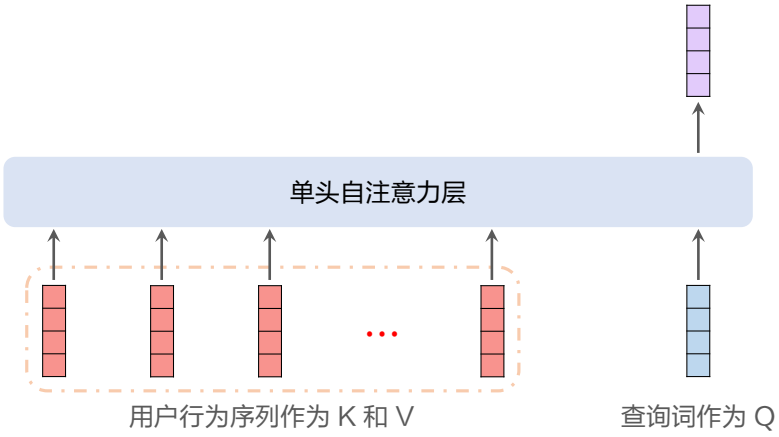

图 10.5: DIN 模型

SIM[13] 长序列建模。

10.2 精排点击率模型

本章介绍精排、粗排、召回截断使用的点击率模型结构，对这方面内容感兴趣的读者，建议阅读推荐系统方面的资料。

10.2.1 多目标深度神经网络模型

图 10.6 中的多目标神经网络是精排最常用的点击率模型。它把查询词特征、文档特征、用户特征、场景特征、统计特征作为输入，输出对点击率和交互率的预估。做训练

时，把用户是否真实点击、点赞、完播等行为作为目标（值为 0 或 1），让神经网络的输出拟合这些目标。损失函数为交叉熵，取多个目标的交叉熵的加权和作为总的损失函数。

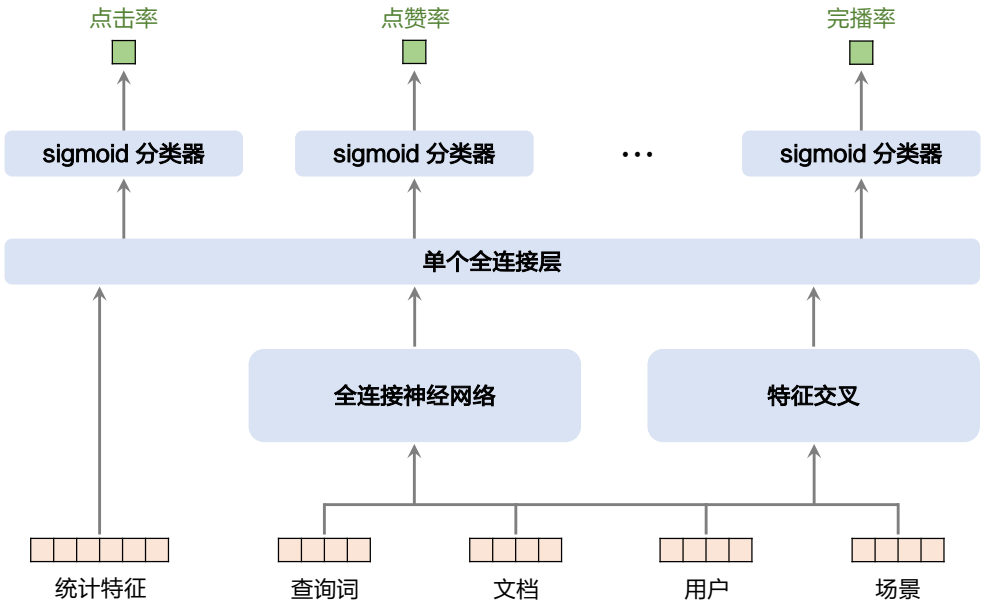

图 10.6: 多目标神经网络模型的示意图

模型拟合的每个目标都可以看做二分类任务，因此常用 AUC 作为离线评价指标。通常来说，如果点击率的离线 AUC 指标提升，则线上的有点比、首点位置、文档点击率都会提升。如果交互率的离线 AUC 指标提升，线上的文档交互率也理应提升。然而交互行为过于稀疏（例如收藏次数比点击次数小 1 到 2 个数量级），线上交互指标波动很大，不容易观测到显著提升。为了便于线上的观测，我们对多种交互指标取加权和，线上只观测该加权和。

有很多种方法让模型的预估变得更准，比如在底层使用 bilinear 交叉特征[7]、在上层使用 multi-gate mixture-of-experts（MMoE）[12]、使用用户行为序列[4, 21, 13]。这些方法最早出现在推荐系统的排序模型中，用在搜索中也同样有效。

图 10.6 所示的模型结构属于“前期融合”，各种离散特征在浅层融合，然后经过 3 到 6 个全连接层的变换，与数值特征融合。前期融合模型的好处是准确性高，坏处是计算量大，因此适用于精排，给数百篇候选文档打分。下面介绍的双塔模型属于“后期融合”，坏处是预估不准确，好处是计算量很小，适用于召回海选和粗排。

10.2.2 MMoE

Multi-gate Mixture-of-Experts（MMoE）模型由 2018 年的论文[12] 提出，用于推荐系统，但也同样适用于搜索引擎的精排点击率模型。MMoE 是对点击率模型上层网络结构的升级，替代图 10.6 中“单个全连接层”及其以上的部分。MMoE 的结构如图 10.7 所示，其底部的红色向量相当于图 10.6 中“单个全连接层”的输入。

图 10.7 的例子中，模型预估 2 个目标（点击率和点赞率），使用 3 个专家神经网络。

[3] 把底层的向量同时输入 3 个专家神经网络，不共享参数，3 个专家神经网络各自输出一个向量，记作 $\boldsymbol{x}_1$、$\boldsymbol{x}_2$、$\boldsymbol{x}_3$。左右两边是门控神经网络，各自对应一个目标，左边门控网络对应点击，右边门控网络对应点赞。如果有 10 个目标，则需要 10 个门控网络。门控网络是一个浅层神经网络，最后一个激活函数是 softmax。两个门控网络各自输出一个 3 维向量：

$$\boldsymbol{p} = [p_1, p_2, p_3] \quad \text{和} \quad \boldsymbol{q} = [q_1, q_2, q_3],$$

3 个元素各对应一个专家。用 $\boldsymbol{p}$ 和 $\boldsymbol{q}$ 对向量 $\boldsymbol{x}_1$、$\boldsymbol{x}_2$、$\boldsymbol{x}_3$ 做加权平均，得到两个向量：

$$\boldsymbol{z}_p = p_1\boldsymbol{x}_1 + p_2\boldsymbol{x}_2 + p_3\boldsymbol{x}_3 \quad \text{和} \quad \boldsymbol{z}_q = q_1\boldsymbol{x}_1 + q_2\boldsymbol{x}_2 + q_3\boldsymbol{x}_3.$$

把 $\boldsymbol{z}_p$ 和 $\boldsymbol{z}_q$ 各自输入一个 sigmoid 分类器，输出点击率和点赞率的预估。

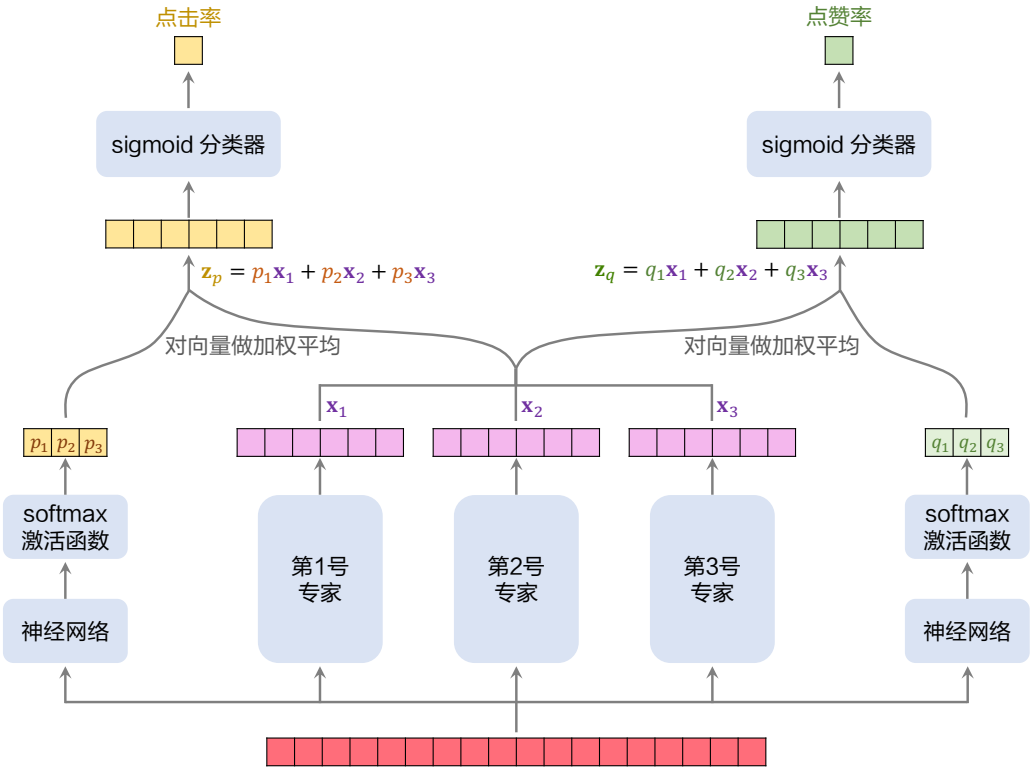

图 10.7: MMoE 模型的示意图

由于向量 $\boldsymbol{p}$ 和 $\boldsymbol{q}$ 是 softmax 激活函数的输出，它们满足两个性质：第一，元素都大于 0，第二，向量中的元素相加等于 1。MMoE 的训练过程中会出现极化现象，即向量 $\boldsymbol{p}$ 和 $\boldsymbol{q}$ 中一个元素接近 1，其余元素接近 0。举个例子，如果

$$\boldsymbol{p} = [0.01, 0.02, 0.97] \quad \text{和} \quad \boldsymbol{q} = [0.94, 0.01, 0.05],$$

那么 $\boldsymbol{z}_p \approx \boldsymbol{x}_3$，$\boldsymbol{z}_q \approx \boldsymbol{x}_1$。这样使得预估点击率只用到第 3 号专家，预估点赞率只用到第 1 号专家。极化现象使得 MMoE 违背了设计的初衷，没有让多个专家融合，而是一个目标只使用一个专家。

[3]此处用 3 个专家神经网络只是为了画图方便，而在实践中，专家神经网络的数量小于预估的目标数量，比如有 10 个目标，用 6 个专家神经网络

为什么这样可以避免极化呢？假如发生极化，学到 $\boldsymbol{p} = [0.01, 0.02, 0.97]$，本该 $\boldsymbol{z}_p \approx \boldsymbol{x}_3$。做 dropout，有 10% 的概率屏蔽第 3 号专家，此时 $\boldsymbol{z}_p = \frac{1}{3}\boldsymbol{x}_1 + \frac{2}{3}\boldsymbol{x}_2$，这会导致对点击率的预估完全错误。在训练的过程中用 dropout，最优的模型参数会尽力避免发生极化，否则会以 10% 的概率出错，导致损失函数过大。

10.3 粗排点击率模型

图 10.8 中是召回海选和粗排常用的点击率模型。模型的左塔将查询词文本、用户特征、场景特征作为数据，输出一个向量表征 $\boldsymbol{x}$。模型的右塔将文档文本和其他特征作为输入，输出一个向量表征 $\boldsymbol{z}$。用 $\text{sigmoid}(\boldsymbol{x}^\top \boldsymbol{z})$ 预估点击率。做训练时，以用户是否点击作为目标 $y \in \{0, 1\}$，让预估点击率拟合目标。

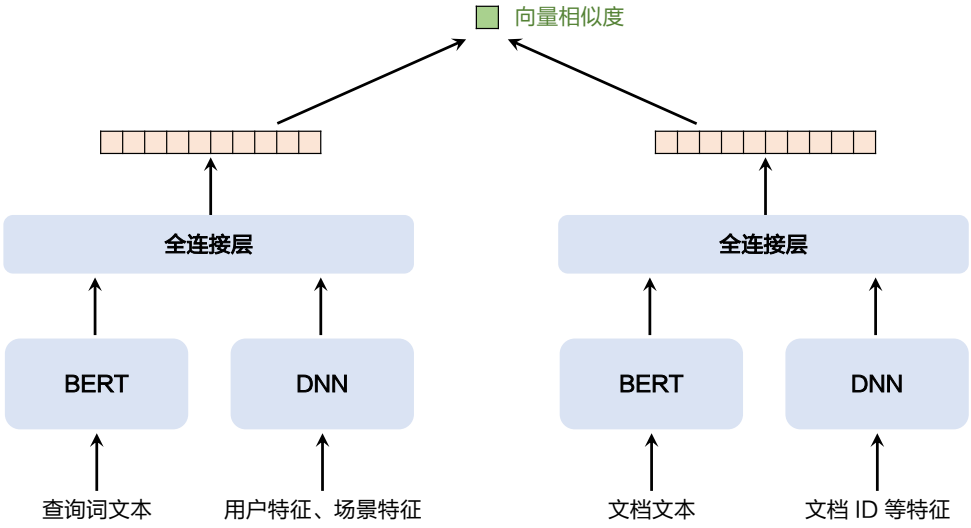

图 10.8: 双塔模型的示意图

应用在线上的召回海选和粗排，不论有多少篇候选文档，每次搜索只有一条查询词、一位用户、一个场景，因此左塔只需要做一次推理。理论上，有多少篇候选文档，右塔就需要做多少次推理，代价非常大。为了降低右塔线上的推理代价，需要离线做推理，将结果以 ⟨文档 ID, 向量表征⟩ 的形式存在哈希表中，线上推理时读取哈希表。通过这种方法，每次搜索用双塔模型给 m 篇候选文档打分，只需要用左塔做一次推理，从哈希表中读取 m 个文档向量表征，计算 m 次向量内积，代价非常小。

双塔模型通常用于召回海选和粗排，离线评价指标是 AUC，线上评价指标是有点比、首点位置、文档点击率。训练双塔模型通常使用知识蒸馏，这样可以提升线上评价指标。把用户真实行为记作 $y \in \{0, 1\}$，把精排模型输出的预估值记作 $\widehat{y} \in (0, 1)$。做训练时，设置权重 $\lambda \in (0, 1)$，以 $\lambda \cdot y + (1 - \lambda) \cdot \widehat{y}$ 代替 y 作为双塔模型拟合的目标。

图 10.9 是双塔模型的一种改进，叫做多目标双塔模型，它可以同时预估点击率与多种交互率。在召回海选和粗排中同时使用点击率和交互率，而非只使用点击率，有利于提升线上的交互率指标。

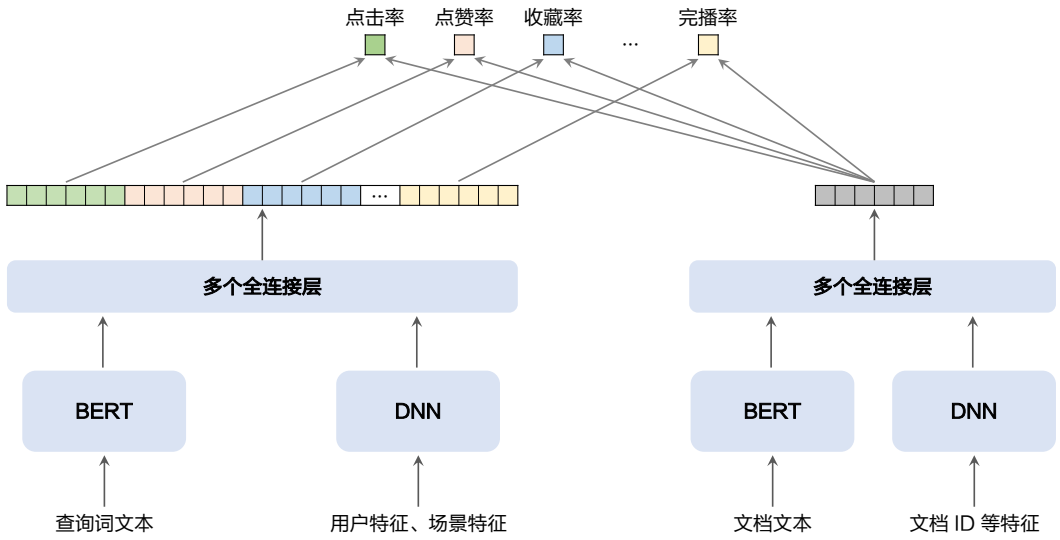

图 10.9: 多目标双塔模型的示意图

双塔模型可以使用用户行为序列特征，用 DIN 或 SIM 将用户行为序列映表征为一个向量。双塔模型可以使用 DCN 代替简单的全连接网络。

10.4 模型训练

10.4.1 训练样本的生成

在讨论生产训练样本的方法之前，我们先讨论精排点击率模型的线上推理。在用模型做推理之前，首先要获取模型所需的特征，特征的数据源主要有以下三个。

- 上游链路传递的特征，包括查询词特征和相关性分数。查询词特征是上游 QP 模块产出的，包括场景特征、查询词特征（类目、命名实体、分词、向量表征）。精排点击率模型使用粗排相关性模型的打分。
- 本地正排表缓存的特征，比如文档类目、命名实体、文本向量表征、多模态向量表征。这些特征相对静态，在较长的时间内不会发生变化、无需更新，存储在本地可以减小特征服务的负载。
- Redis 这样的 KV 索引上存储的特征，比如统计特征、用户画像、用户行为序列。这些特征相对动态，更新频率较高。

在获取和拼接上述特征之后，一方面将特征喂给模型，另一方面将 (搜索 ID, 文档 ID, 特征) 存储在 Hive 表中。

网页、APP 端都有打点服务，记录文档是否获得曝光、点击、交互，作为训练模型使用的标签。网页、APP 端会将打点数据传输给收集日志的服务器，服务器把 (搜索 ID, 文档 ID, 标签) 存储到 Hive 表。在每天凌晨，对两张表做 join，得到 (特征, 标签)，打包成 TFRecord 或 Parquet 格式的文件，作为训练模型的数据。这样的到的数据包含精排点击率模型所需的所有特征，而粗排、召回截断、召回模型只需要其中的部分特征。

10.4.2 降采样和校准

接下来我们讨论训练时的负样本的降采样、推理时的校准。训练精排和粗排的点击率模型，使用曝光的样本，这些样本可能有点击（正样本），也可能没有点击（负样本）。点击率是个较小的数值，也就是说，绝大部分的样本都是负样本。把正样本总数记作 n_+，负样本总数记作 n_-，那么训练所需的计算量正比于 $n = n_+ + n_-$。由于 $n_+ \ll n_-$，使用全部的负样本或是部分负样本，训练出的排序模型效果相当。设置一个负样本降采样率 α（$0 < \alpha < 1$），从 n_- 个负样本中选出 αn_- 个，那么样本总数从 $n = n_+ + n_-$ 降低到 $n' = n_+ + \alpha n_-$，计算量会等比例下降。实践中使用的负样本数量 αn_- 是 n_+ 的 1 到 2 倍。

对负样本做降采样，那么训练数据中的正样本占比会高于真实占比，导致模型预估的点击率 p_{pred} 高于真实点击率 p_{true}。降采样率 α 越小，则高估约严重。模型在线上做完推理之后，需要校准预估的点击率 p_{pred}，得到：

$$p_{\text{true}} = \frac{\alpha \cdot p_{\text{pred}}}{(1 - p_{\text{pred}}) + \alpha \cdot p_{\text{pred}}}. \tag{10.1}$$

p_{true} 的值介于 0 和 1 之间。

下面的内容是对校准公式 (10.1) 的推导。值得注意的是，这个推导并不严谨。训练模型时，用了 n_+ 个正样本和 $\alpha \cdot n_-$ 个负样本。在理想的情况下，如果模型的预估足够准确，那么

$$\mathbb{E}\left[p_{\text{pred}}\right] = \frac{n_+}{n_+ + \alpha \cdot n_-}.$$

设 p_{true} 为校准后的点击率，它应当排除降采样率 α 的影响，并满足性质：

$$\mathbb{E}\left[p_{\text{true}}\right] = \frac{n_+}{n_+ + n_-}.$$

由以上两个公式可得：

$$\frac{1}{\mathbb{E}[p_{\text{pred}}]} = 1 + \alpha \cdot \frac{n_-}{n_+} = 1 + \alpha \cdot \left(\frac{1}{\mathbb{E}[p_{\text{true}}]} - 1\right).$$

调整等式左右两边，得到：

$$\mathbb{E}\left[p_{\text{true}}\right] = \frac{\alpha \cdot \mathbb{E}[p_{\text{pred}}]}{(1 - \mathbb{E}[p_{\text{pred}}]) + \alpha \cdot \mathbb{E}[p_{\text{pred}}]}. \tag{10.2}$$

式 (10.2) 与 (10.1) 并不等价。式 (10.2) 是严谨的，而式 (10.1) 只是个近似的经验公式，并不严谨。

在实践中，对 logit 做校准，比对概率做校准更简单。设 x_{pred} 为 sigmoid 激活函数的输入，即 $p_{\text{pred}} = \text{sigmoid}(x_{\text{pred}})$。直接根据采样率 α 对 logit 做校准：

$$x_{\text{true}} = x_{\text{pred}} + \ln \alpha,$$

那么有

$$\text{sigmoid}\left(x_{\text{true}}\right) = \frac{\alpha \cdot p_{\text{pred}}}{(1 - p_{\text{pred}}) + \alpha \cdot p_{\text{pred}}}.$$

即上式与式 (10.1) 等价。

10.4.3 离线训练与评估

1 epoch

样本的 join

负采样、校准。

老汤模型、热启、冷启 7 天比较。只热启 dense，需要 30 天追平；全部冷启，3 个月没有追平，放弃了。

AUC、GAUC

10.5 知识点小结

- 与相关性、内容质量、时效性、地域性并列，个性化也是用户满意的一个维度。并非所有的搜索引擎都有个性化，只有当平台有用户画像和丰富的行为记录时，才能做到个性化的搜索。
- 不论搜索引擎是否有个性化，都需要点击率模型，它根据文本、非文本的特征预估点击率、交互率等分数，与相关性等分数融合，作为排序的依据。如果搜索引擎有个性化，那么个性化反映在预估的点击率和交互率上，通过这些预估值让排序带有个性化。
- 精排使用多目标神经网络预估点击率和交互率，这种模型叫作前期融合，计算代价大，预估较为准确。召回海选和粗排通常使用双塔模型预估点击率和交互率，这种模型叫作后期融合，计算代价小，预估不准确。
- 前期融合、后期融合模型都可以使用用户行为序列，对 AUC 有非常大的提升。简单的用户行为序列建模方法是取平均，更先进的方法是 DIN[21] 和 SIM[13]。
- 搜索引擎和推荐系统的点击率模型几乎相同，改进推荐系统点击率模型的方法大多适用于搜索引擎。但是搜索与推荐的性质不同，点击率、交互率在搜索排序中起到的作用小于相关性，而在推荐排序中起到决定性的作用。增加模型规模、增加用户序列长度、用在线学习实时更新模型，这些方法可以让点击率模型的预估更准确，但是需要大幅增加成本。用在推荐系统上，这些方法是划算的，能带来较大的业务指标提升；用在搜索引擎上，这些方法未必划算，毕竟点击率和交互率只是搜索排序诸多信号中的一种而已，并不是起决定性作用的信号。

第四部分

查询词处理与文本理解

第 11 章 分词与命名实体识别

搜索引擎的文本召回是以词粒度进行的。举个例子，用户搜索 q =“冬季卫衣推荐”，其中包含“冬季”、“卫衣”、“推荐”这 3 个词，如果文档 d 同时包含这 3 个词，比如 d =“推荐一款冬季穿的卫衣”，则 d 会被召回。当用户搜索 q =“冬季卫衣推荐”，算法如何将它切分成 3 个词？本章讨论分词技术，11.1 节介绍基于词典的分词方法，11.3 节介绍基于深度学习的分词方法，11.2 节介绍新词发现的方法（即如何构造词典）。

11.1 基于词典的分词方法

本节介绍基于词典的分词方法，这种方法需要一个事先准备好的词典，下一节讲解自动构造词典的方法。基于词典的分词方法容易实现和线上部署，对算力的要求非常低，但分词不够准确，而且没有泛化能力，无法识别未登录词（即不在词典中的词）。下一节介绍的深度学习方法代价大，但是更准确。

11.1.1 最大匹配分词

最大匹配法分为**正向最大匹配法**与**反向最大匹配法**。正向最大匹配法的意思是从左至右，以贪心的方式切分出当前位置上长度最大的词。反向最大匹配法方向相反，从右到左寻找长度最大的词。图 11.1 中是两个例子，第一个例子反向匹配正确，第二个例子正向匹配正确。

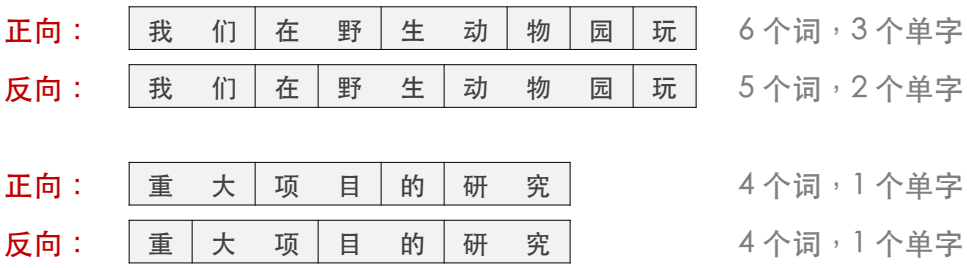

图 11.1: 最大匹配法

双向最大匹配同时执行正向和反向最大匹配。如果两者的词数不同，则返回词数更少的那一个。如果词数相同，返回两者中单字更少的那一个。当单字数也相同时，优先返回反向最长匹配的结果。如果用双向最大匹配，则图 11.1 中第一个例子的结果正确，第二个例子返回反向的结果错误。

正向和反向最大匹配都使用 Trie 数据结构。事先将词典用 Trie 存储，在做分词时，查看 Trie 寻找最长匹配。用这种方法，如果输入的长度是 n 个字，那么只需 $\mathcal{O}(n)$ 的时间就能完成正向和反向最大匹配。这个时间复杂度是理论最优值，因为至少需要 $\mathcal{O}(n)$ 的

时间才能读完 n 个字。虽然最大匹配实现简单、速度快，但它的效果不好，工业界的实践中不用这种方法做分词。

11.1.2 最短路径分词

最短路径分词也是一种基于词典的分词算法。如图 11.2 所示，如果句子有 n 个字，那么就创建 $n+1$ 个节点，n 个字各作为一条边。如果介于节点 i 和 j（$i < j$）之间的字组成一个字典中的词，那么添加一条从 i 到 j 的边。这样构造出一个有向无环图，边的权重都是 1。有很多条从起点到终点的路径，图 11.2 中标出了两条路径，长度分别是 3 和 4。使用最短路算法寻找头、尾两个节点之间的最短路径，作为分词结果。例子中长度为 3 的路径 $0 \to 2 \to 4 \to 6$ 对应分词“研究 / 生命 / 起源”，是正确的分词。最短路分词未必正确，例子中的路径 $0 \to 3 \to 4 \to 6$ 的长度是 3，对应分词“研究生 / 命 / 起源”。

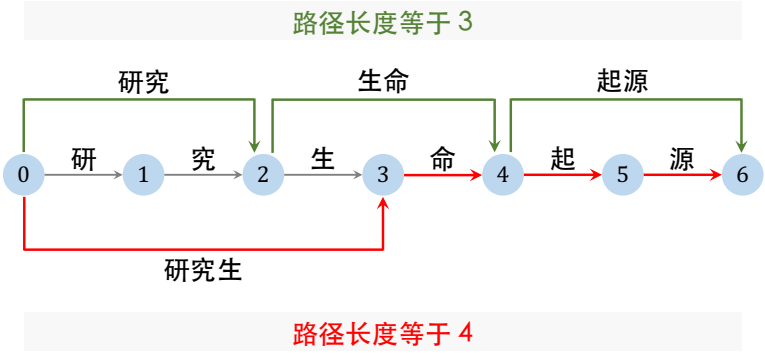

图 11.2: 最短路径分词

11.2 词典的构造

上一节介绍了基于词典的分词方法，这类方法需要一个事先准备好的词典。工业界通常采用自动构造的词典，即根据海量的语料做统计，判断哪些字符串成词，用这些字符串构成词典。词典需要定期更新，否则会不断出现“未登录词”，即不在词典中的词，通常为专有名词、缩写词、流行词汇。

该如何根据统计值判断字符串是否成词呢？最直观的想法，如果字符串成词，那么它在文档中出现的频率会比较高，例如“电影”的频率远远高于“影电”。是否可以计算字符串出现的频率，并根据字符串长度设不同阈值，高于阈值的就算成词呢？很可惜，这种想法是不对的。根据常识，大家知道和“电影院”成词，而“电影的”则不成词，尽管两者频率是同一个数量级。大家还知道“共和国”成词，而“和国”则不成词；但由于“共和国”出现的频率很高，“和国”也顺带成为了高频词。

判断一个字符串是否成词，正确的做法是计算词内部的凝聚度、相邻词间的自由度，如果两者均大于阈值，则被认定成词，加入词典。“电影院”的凝聚度高，而“电影的”凝聚度低；“共和国”的自由度高，而“和国”的自由度低。使用凝聚度、自由度，算法可以判定

“电影的”、“和国”不成词。下面两小节分别讲解凝聚度和自由度。

11.2.1 词内部的凝聚度

算法如何知道“电影院”成词，而“电影的”不成词呢？设 A 和 B 是两个字符串，如果 AB 不成词，则文档中的 AB 只是 A 和 B 碰巧在一起而已，词的频率为

$$p(AB) \approx p(A) \cdot p(B).$$

如果 AB 成词，那么 $p(AB)$ 必然远大于 A 和 B 碰巧撞在一起的概率 $p(A) \cdot p(B)$。我们得出一个结论：$\frac{p(AB)}{p(A) \cdot p(B)}$ 越大，则 AB 越有可能成词。

我们来验证一下这个想法。在社交网络上，字符串的频率为 $p(\text{电影}) = 10^{-4}$，$p(\text{院}) = 2 \times 10^{-4}$。假如两个词无关，即“电影院”只是“电影”、“院”碰巧撞在一起，那么“电影院”的频率应当是 $p(\text{电影}) \cdot p(\text{院}) = 2 \times 10^{-8}$。但实际上，“电影院”出现的频率为 $p(\text{电影院}) = 7 \times 10^{-6}$，计算出比值等于

$$\frac{p(\text{电影院})}{p(\text{电影}) \cdot p(\text{院})} = \frac{7 \times 10^{-6}}{2 \times 10^{-8}} = 350,$$

这说明“电影院”很可能成词。再看一下“电影的”，统计得出 $p(\text{电影}) = 10^{-4}$，$p(\text{的}) = 1.7 \times 10^{-2}$，$p(\text{电影的}) = 1.6 \times 10^{-5}$，计算出比值

$$\frac{p(\text{电影的})}{p(\text{电影}) \cdot p(\text{的})} = \frac{1.6 \times 10^{-5}}{1.7 \times 10^{-6}} = 9.4,$$

说明“电影的”成词的可能性不大。概括一下，“电影”和“院”凝聚到一起的概率远大于随机碰上，而“电影”和“的”凝聚到一起的概率只是略大于随机碰上。以凝聚度作为判断标准，“电影院”远比“电影的”更像一个词。

为了衡量凝聚度，业界普遍使用 pointwise mutual information（PMI），定义为 $\text{PMI}(A, B) = \log_2 \frac{p(AB)}{p(A) \cdot p(B)}$。对于长度为 2 的字符串 ab，凝聚度定义为

$$\text{凝聚度}(ab) = \text{PMI}(a, b),$$

分数越高，则 ab 越有可能成词。对于长度为 3 的字符串 abc，凝聚度定义为

$$\text{凝聚度}(abc) = \min\Big\{\text{PMI}(a, bc),\ \text{PMI}(ab, c)\Big\},$$

对于长度为 4 的字符串 $abcd$，凝聚度定义为

$$\text{凝聚度}(abcd) = \min\Big\{\text{PMI}(a, bcd),\ \text{PMI}(ab, cd),\ \text{PMI}(abc, d)\Big\}.$$

更长的字符串用类似的方法计算凝聚度。这种定义需要枚举所有可能的切词位置，不论在哪里切，PMI 都必须高，否则不被认定为成词。例如“电影院”，前面我们算出 PMI(电影, 院) 分数很高，其实 PMI(电, 影院) 分数也很高，两个 PMI 分数取 min 还是很高，所以“电影院”被判定为高凝聚度。再比如“电影的”，PMI(电影, 的) 分数低，PMI(电, 影的) 分数高，两个 PMI 分数取 min 后很低，所以“电影的”被判定为低凝聚度。

概括一下，对于一个字符串，我们从不同位置做切分，计算每一种切分的 PMI，取 min 得到凝聚度。通常来说，字符串越长，则 PMI 越高。因此需要为不同长度的字符串设定不同阈值，字符串越长则阈值越大。如果凝聚度小于某个阈值，则判定为不成词；但凝聚度大于阈值不意味成词，需要凝聚度和自由度均大于阈值才行（见下一小节）。

11.2.2 相邻词词间的自由度

使用词内部的凝聚度可以判定“电影的”不成词，但无法判定“和国”不成词。“共和国”出现的频率高导致“和国”出现的频率也高，远高于“和”、“国”碰巧撞在一起的概率，算法判定“和国”的凝聚度高。想要判断“和国”不成词，需要用到相邻词间的自由度。

如图 11.3 所示，如果一个字符串成词，则其有丰富的左右的邻词；如果一个字符串是某个词的子串，则其可搭配的左邻字较少、或右邻字较少。信息熵（information entropy）是对不确定性的度量，可以用在这里衡量相邻词间的自由度。设字符串为 A，其左边字为 l，右边字为 r。给定 A，左边出现字 l 的概率记作 $p(l|A) = \frac{p(lA)}{p(A)}$，定义信息熵：

$$H_{\text{left}}(A) \;=\; -\sum_{l\in\mathcal{C}} p(l|A)\cdot\log_2 p(l|A),$$

上式中的 $\mathcal{C}$ 是所有字的集合。“电影院”左邻字非常丰富，不确定性高，因此 H_{left}(电影院) 较大。“和国”左邻字几乎是确定的，大概率是“共”，因此 H_{left}(和国) 很小。

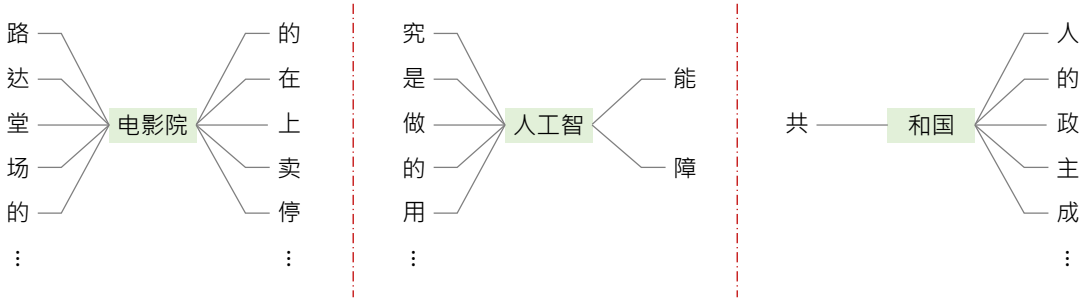

图 11.3: 图中 3 个例子为字符串可搭配的左邻字和右邻字

我们可以类似定义 $H_{\text{right}}(A)$，衡量字符串 A 右邻字的不确定性。不难发现，“电影院”的右邻字也很丰富，因此 H_{right}(电影院) 较大。而“人工智”的右邻字不确定性很小，不是“能”就是“障”，因此 H_{right}(人工智) 很小。取两个信息熵中较小者：

$$H(A) \;=\; \min\Big\{\, H_{\text{left}}(A)\,,\; H_{\text{right}}(A)\,\Big\},$$

用 $H(A)$ 衡量字符串 A 的自由度，自由度越高，说明 A 越不像是词的一个子串。

11.3 基于深度学习的分词方法

查询词较短，对分词准确性要求很高，因此需要更准确的分词方法。基于 BERT 模型的序列标注（sequence tagging）是目前最准确的分词方法。序列标注的意思是输入的句子中包含 n 个字，要求输出每个字的标签。分词任务有 4 种标签，分别是 B（begin）、M（middle）、E（end）、S（single）。下面的例子解释 4 种标签的含义：

B M M E S B M E S B E B E
机 器 学 习 / 在 / 工 业 界 / 有 / 广 泛 / 应 用

最直接的方法是用 BERT 模型预测每个字符的标签。模型的输入是一句话，以字的粒度输入图 11.5 所示的模型。设输入的句子有 n 个汉字，模型输出向量的序列 $\boldsymbol{p}_1, \cdots, \boldsymbol{p}_n$，每个向量 $\boldsymbol{p}_i$ 的形状都是 4 维，对应标签 B、M、E、S 的概率，且 4 个元素相加等于 1。

用这种方法，分词被建模成 4 分类问题，训练时使用交叉熵损失函数，让 $\boldsymbol{p}_i$ 拟合标签的 one-hot 编码。在推理时，独立对每个 $\boldsymbol{p}_i$ 做 argmax，哪个标签的概率最大，就取哪个标签。

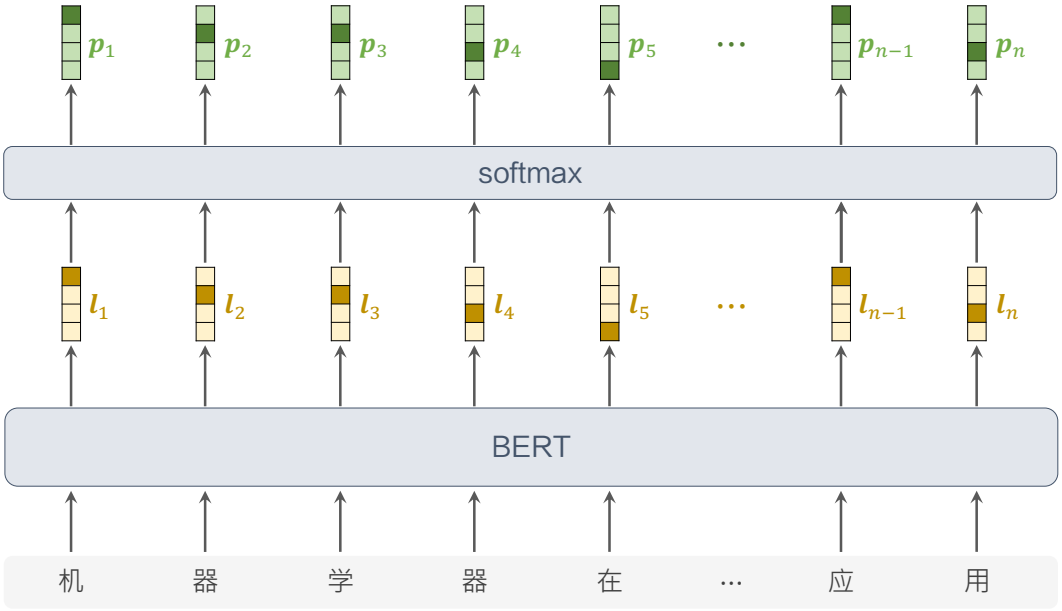

图 11.4: 用 BERT 模型预测每个汉字的标签

上述方法存在一个缺点，举个例子，模型预测前两个汉字的标签分别为 B 和 S，显然不正确，原因是 B 后面只能接 M 和 E。为了保证标签 B 的后面只能接 M 或 E，而不能接 B 和 S，我们用条件随机场（conditional random field，CRF）建模相邻标签之间的关系。把当前位置标签记作 y、下一个位置标签记作 y'，把转移的分数记作 $t_{y\to y'}$。如果 y 是标签 B，y' 是标签 M 或 E，则转移 $y \to y'$ 是合理的，学到的 $t_{y\to y'}$ 值大。如果 y' 是标签 B 或 S，则转移 $y \to y'$ 是不合理的，学到的 $t_{y\to y'}$ 值小。如图 11.5 所示，CRF 会从训练数据中学出一个 4×4 的转移矩阵 $\boldsymbol{T}$，它的元素 $t_{y\to y'}$ 表示当前位置标签为 y、下一个标签为 y' 的分数。

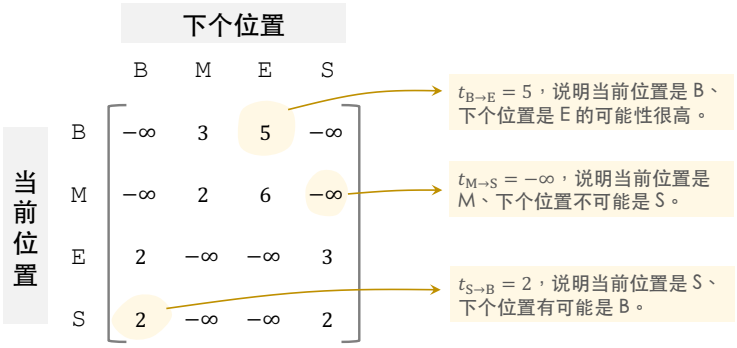

图 11.5: 理想的转移矩阵 $\boldsymbol{T}$

设一句话包含 n 个字，记作 $\boldsymbol{x} = [x_1, \cdots, x_n]$，真实的位置标签记作 $\boldsymbol{y} = [y_1, \cdots, y_n]$。

做训练时，同时学习 BERT 模型参数和 CRF 的转移矩阵 $\boldsymbol{T}$。训练 BERT+CRF 有两个目标。

- CRF 的 4×4 的转移矩阵 $\boldsymbol{T}$ 建模相邻标签的关系，允许 B $\rightarrow$ M 这样的转移，不允许 B $\rightarrow$ B 这样的转移。由于 $y_1, \cdots, y_n$ 是真实的标签序列，$y_i \rightarrow y_{i+1}$ 这样的转移都是合理的，那么应当鼓励 $t_{y_1 \rightarrow y_2}, \cdots, t_{y_{n-1} \rightarrow y_n}$ 越大越好。
- BERT 模型输出 n 个 logit 向量 $\boldsymbol{l}_1, \cdots, \boldsymbol{l}_n$，每个向量都是 4 维。把向量 $\boldsymbol{l}_i$ 的元素记作 $l_{i,\mathrm{B}}$、$l_{i,\mathrm{M}}$、$l_{i,\mathrm{E}}$、$l_{i,\mathrm{S}}$，表示字 x_i 的 4 种标签的分数。由于 y_i 是真实标签，那么 l_{i,y_i} 越大越好，其他 3 个值越小越好。

这样我们就推导出了训练 BERT+CRF 的优化问题。定义

$$\pi(\boldsymbol{y} \,|\, \boldsymbol{L}, \boldsymbol{T}) \;=\; \frac{\mathrm{score}(\boldsymbol{y}, \boldsymbol{L}, \boldsymbol{T})}{\sum_{\boldsymbol{y}' \in \{\mathrm{B,M,E,S}\}^n} \mathrm{score}(\boldsymbol{y}', \boldsymbol{L}, \boldsymbol{T})},$$

其中 $\boldsymbol{L} = [\boldsymbol{l}_1, \cdots, \boldsymbol{l}_n]$ 是 BERT 输出的 logits，

$$\mathrm{score}(\boldsymbol{y}, \boldsymbol{L}, \boldsymbol{T}) \;=\; \exp\left(\sum_{i=1}^{n-1} t_{y_i \rightarrow y_{i+1}} \;+\; \sum_{j=1}^{n} l_{j,y_j} \right).$$

训练时，给定一条样本 $\boldsymbol{x}$ 和 $\boldsymbol{y}$，求 $\ln \pi(\boldsymbol{y} \,|\, \boldsymbol{L}, \boldsymbol{T})$ 关于 BERT 和 CRF 参数求梯度，并做梯度上升更新参数。做梯度上升，会让 $\pi(\boldsymbol{y} \,|\, \boldsymbol{L}, \boldsymbol{T})$ 增大，那么 $\{\, t_{y_i \rightarrow y_{i+1}} \,\}_{i=1}^{n-1}$ 和 $\{\, l_{i,y_i} \,\}_{i=1}^{n}$ 都会增大。

训练好模型之后，可以用模型做推理。给定一句话包含 n 个字 $\boldsymbol{x} = [x_1, \cdots, x_n]$，我们不知道它们的标签，需要用 BERT+CRF 模型推断标签。先用 BERT 做推理，输出 n 个 logits 向量 $\boldsymbol{l}_1, \cdots, \boldsymbol{l}_n$，每个向量都是 4 维的，对应标签 B、M、E、S，如图 11.6 所示。然后求解下面的优化问题：

$$\widehat{\boldsymbol{y}} \;=\; \underset{\boldsymbol{y} \in \{\mathrm{B,M,E,S}\}^n}{\mathrm{argmax}} \;\; \mathrm{score}(\boldsymbol{y}, \boldsymbol{L}, \boldsymbol{T}),$$

上式结合 CRF 的转移矩阵 $\boldsymbol{T}$ 与 BERT 模型算出的 $\boldsymbol{L}$，从集合 $\{\mathrm{B}, \mathrm{M}, \mathrm{E}, \mathrm{S}\}^n$ 选出最优的 $\widehat{\boldsymbol{y}} = \{\widehat{y}_1, \cdots, \widehat{y}_n\}$。如果把每个字符候选标签 B、M、E、S 看做节点，那么上述优化问题就是从 4^n 个节点的图中寻找一条最优的路径。常用维特比（Viterbi）算法求解上式，寻找一条路径。

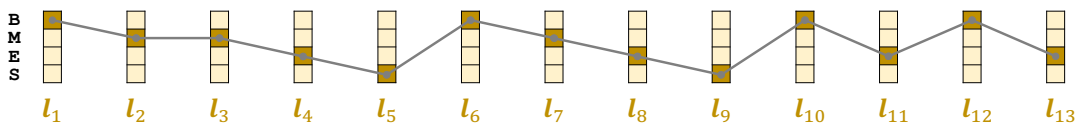

图 11.6: 用 BERT 计算每个字的 logits 向量，然后结合 CRF 寻找最优路径

在 2018 年之前，分词等序列标注的最优方法是线性模型 +CRF、然后是 RNN+CRF，方法与上文的描述基本一致。在 BERT 模型出现之后，序列标注的最优方法是 BERT+CRF。但随着预训练做得越来越好，大家普遍发现 CRF 起到的作用极小，只用 BERT 就能取得几乎同等的效果。因此，只用 BERT 做分词也是完全合理的。

11.4 命名实体识别

命名实体（named entity）指识别文本中具有特定意义的词，比如品牌、品类、人物、地点。命名实体识别（named entity recognition，NER）是一种序列标注任务，给定一句话，从中找出品牌、品类、人物、地点等命名实体。在通用搜索、电商搜索等场景中，通常有数十种命名实体，下面列出 4 种。

- 品牌，比如特斯拉、宝马、尼康、劳力士、布洛芬、雅诗兰黛、Gap……
- 品类，比如汽车、相机、手表、药品、口红、香水、美甲、夹克、童装……
- 人物，比如牛顿、丘吉尔、秦始皇、伊丽莎白二世、杨幂、赫敏、皮卡丘……
- 地点，比如加拿大、纽约、上海、故宫、北京环球影城、黄石公园……

给定输入的查询词“杨幂代言的雅诗兰黛口红”，将杨幂识别成人物，雅诗兰黛识别为品牌，口红识别为品类。从查询词和文档识别出的实体是相关性模型、点击率模型所需的特征。

NER 与分词任务非常相似，都是做序列标注，把一句话以字粒度输入模型，预测每一个字的标签。前文讨论过，分词一共有 4 种位置标签——B（begin）、M（middle）、E（end）、S（single）。NER 使用 2 种位置标签 B（begin）、I（intermediate）和非实体标签 O（other）；NER 的标签不同于分词，但这只是习惯的不同而已。设有 k 种实体，每种实体有 B 和 I 两种位置标签，外加非实体的标签 O。NER 的标签数量较多，如果有 k 种实体，则共有 $2k+1$ 个标签。下面的例子解释标签的含义：

B-人物	I-人物	O	O	O	B-品牌	I-品牌	I-品牌	I-品牌	B-品类	I-品类
杨	幂	代	言	的	雅	诗	兰	黛	口	红

早年的 NER 为基于规则的方法、基于字典的方法。在概率图模型兴起之后，NER 常用 HMM 和 CRF。在深度学习成熟之后，先后出现 RNN+CRF[8]、BERT+RNN+CRF[5, 18]。目前 BERT+CRF 是最常用的 NER 方法，它的原理与分词完全相同，此处就不再赘述。不论是分词还是 NER，想要训练 BERT+CRF 模型，都需要人工标注数据。在预训练 BERT 的基础上，用人工标注数据做微调，得到最终的 BERT+CRF 模型。

2021 年的论文[11] 提出了 LEBERT 这种方法，适用于中文的分词、NER 等序列标注任务。如图 11.7 所示，一句话包含多个字，每个字被多个词覆盖。把第 i 个字表征为向量 $\boldsymbol{h}_i$，把它对应的词表征为向量 $\boldsymbol{v}_{i,1}, \cdots, \boldsymbol{v}_{i,m}$。如图 11.8 所示，将 $\boldsymbol{v}_{i,1}, \cdots, \boldsymbol{v}_{i,m}$ 作为 K 和 V，将 $\boldsymbol{h}_i$ 作为 Q，一同输入单头交叉注意力层。将注意力层输出的向量与 $\boldsymbol{h}_i$ 相加，得到 $\tilde{\boldsymbol{h}}_i$，作为序列中第 i 个字的向量表征。如果句子中有 n 个字，那么得到 $\tilde{\boldsymbol{h}}_1, \cdots, \tilde{\boldsymbol{h}}_n$，作为 BERT 模型的输入。LEBERT 与标准字粒度 BERT 的区别在于使用 $\tilde{\boldsymbol{h}}_i$ 代替 $\boldsymbol{h}_i$ 作为向量表征。

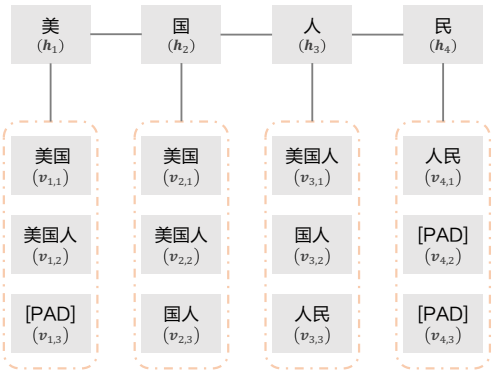

图 11.7: 构建字和词序列

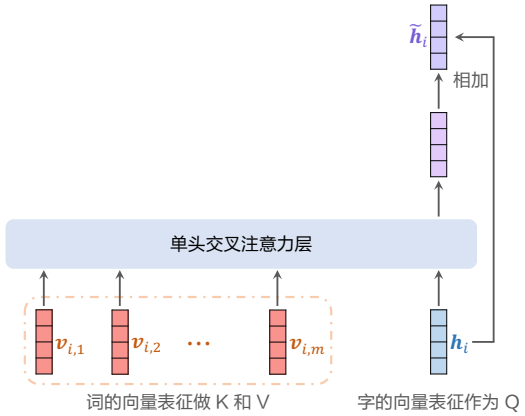

图 11.8: 对字和次序列做向量表征

11.5 评价指标

分词与 NER 的评价指标相同，都是准确率、召回率、F1。下面举个例子讲解分词的评价指标，命名实体识别也是类似的。

人工标注： 深度学习 / 在 / 搜索引擎 / 中 / 的 / 应用，

算法推理： 深度学习 / 在 / 搜索 / 引擎 / 中 / 的 / 应用。

算法推理的结果中，“深度学习”、“在”、“中”、“的”、“应用”这 5 个词命中人工标注的分词结果，而“搜索”和“引擎”这 2 个词没有命中，，因此 TP= 5，FP= 2。算法没有命中人工标注的“搜索引擎”，因此 FN= 1。准确率和召回率分别是

$$\text{precision} = \frac{\text{TP}}{\text{TP}+\text{FP}} = \frac{5}{7}, \qquad \text{recall} = \frac{\text{TP}}{\text{TP}+\text{FN}} = \frac{5}{6}.$$

F1 是准确率和召回率的调和平均数：

$$\text{F1} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} = \frac{10}{13}.$$

测试集中有多个句子，每个句子都有准确率、召回率、F1。对所有句子取平均，得到整个数据集上的指标。

11.6 知识点小结

- 中文的词之间没有空格分隔，需要用算法对查询词和文档做分词。可以根据词典做分词，也可以使用 BERT 等语义模型做分词。前者分词不准，而且没有泛化能力；后者非常准确，有泛化能力，但是推理代价较大。
- 可以从大规模语料中自动挖掘出一个词典。具体方法是将文档切分成字符串，统计字符串的频率，计算词内部的凝聚度和相邻词间的自由度，判断字符串是否成词。
- 分词和 NER 都可以建模成序列标注问题，常用 BERT 或 BERT+CRF 模型。LEBERT 模型结合字与词的向量表征，它是比字粒度 BERT 模型更优的中文序列标注方法。
- 分词、NER 等序列标注任务的离线评价指标为准确率、召回率、F1。

第 12 章 词权重

上一章讲解了分词技术，它可以将查询词 q 切分成若干个词，用于检索文档。这些词的重要性各不相同。丢掉某些词会导致查询词的语义发生变化，检索到的文档与 q 相关性很低；而丢掉某些词则几乎不影响查询词的语义，检索到的文档仍与 q 具有高相关性。词权重（term weight 或 term necessity）是 QP 链路上的一环，以分词的结果作为输入，用模型给每个词赋予一个权重。词权重的主要用途是召回。在文本召回的海选阶段需要计算 BM25 等分数，命中不同重要性的词，对 BM25 分数的贡献有所不同。在召回结果过少时，根据词权重丢弃一部分词，这样可以在尽量少损失相关性的前提下，大幅增加召回量。词权重还有其他用途，比如在训练召回模型时做数据增强，丢弃核心词生成负样本，丢弃非核心词生成正样本。

12.1 词权重的定义与标注方法

经典的词权重计算方法是用人工标注数据，训练 BERT 等语义模型，用在线上做推理。这种思路与训练相关性模型颇为相似。词权重分为 4 档，如下定义档位和分数。

- 核心词，分数为 3，代表查询词的核心意图，去掉后查询词的意思完全改变，难以搜到原查询词的意图。
- 强需求词，分数为 2，查询词核心意图的重要组成部分，去掉后查询词的意思有所改变，但仍可能搜到原查询词的意图。
- 弱需求词，分数为 1，修饰查询词的主要意图，去掉后查询词的主意图基本不变，但整体意思会有所变化。
- 冗余词，分数为 0，去掉后查询词的意思几乎不变。

举个例子，查询词是“冬季 / 风衣 / 推荐”。“风衣”是核心词，如果去掉，则意图发生根本的变化。“冬季”是强需求词，如果去掉，仍然能搜到风衣，其中部分能满足用户意图。“推荐”是弱需求词，去掉之后对意图的影响不大，搜出的结果基本能满足用户意图。

接下来我们讨论人工标注的标准。算法工程师从搜索日志中随机抽出一批查询词，需要覆盖头部和长尾。对查询词做分词之后，交给标注团队。做人工标注时，首先需要判断查询词是否可用，如果不可用，则无需标注词权重。不可用的情况主要有如下两种。

- 查询词拼写错误，比如“红烧肉的做法 1”，用户多打了一个 1。词权重模块位于纠错之后，这种问题应当由纠错解决，词权重模块无需考虑。对于这样写错的查询词，无需做人工标注。
- 分词错误，比如“研究 / 生命 / 起源”被错分成“研究生 / 命 / 起源”。由于分词导致语义变化，词权重不再适用，无需做标注。
- 紧密度错误会导致分词粒度的错误，比如“上海 / 巴黎贝甜”被错分成“上海 / 巴黎 / 贝甜”。这种错误不影响标注，由于“巴黎贝甜”被切开，只需要把“巴黎”和“贝甜”标注相同的权重即可。

对于拼写正确、且分词正确的查询词，根据 4 种档位的定义做人工标注。在实践中会遇到一些特殊情况，特殊情况需要按照以下法则特殊对待。

- 停用词如“的”、“啊”、“呀”、“了”的档位都是 0。表意疑问的词如“怎么”、“如何”、“在哪”、“为什么”，档位为 1 或 0，不同公司的标注标准所有不同。较弱的限定词如“高清”、“大全”、“全集”、“免费版”、“攻略”、“指南”、“推荐”的档位都是 1。
- 如果同时存在上下位词，且下位词是独一无二的，则丢弃上位词毫无影响，可以把上位词标为冗余词。例如“北京市 / 人大附中 / 的 / 学区房”，其中“北京市”是“人大附中”的上位词，且世界上只有一个人大附中，去掉“北京市”不影响搜索结果，因此档位为 0。同理，对于查询词“苹果 / iphone 14”、“兰蔻 / 小黑瓶 / 测评”、“海南 / 三亚 / 景点”，其中的“苹果”、“兰蔻”、“海南”都是上位词，档位为 0。
- 如果同时存在上下位词，但下位词不是独一无二的，那么就不能丢弃上位词，上位词的档位应当为 2 或 1，具体取决于歧义的程度。例如“狄仁杰 / 王者荣耀”，下位词“狄仁杰”更为人所知的是历史人物和影视人物，因此不能去掉上位词“王者荣耀”，王者荣耀的词档位为 2 或 1。同理，“山东 / 德州”、“福特 / 金牛座”中的“山东”和“福特”都是上位词，档位为 2 或 1。
- 对于品牌 + 品类的词，比如“特斯拉 / 电车”、“欧莱雅 / 眼霜”、“LV / 女包”，品牌、品类的档位为 3/3 或 3/2。
- 有重复意思的词，两个词权重相同。例如“适合 / 情侣 / 去 / 的 / 地方”，其中“去”和“地方”意思重复，两者档位应当相同。假如同时删掉两个词，那么查询词变成“适合情侣的”，意图有所改变，因此“去”和“地方”档位均为 2。再例如“一年 / 四季”，两个词含义相同，同时去掉两个词则查询词为空，因此两者档位都是 3。
- 如果分词紧密度有误，把一个词切开，则每个词的档位相同。例如“巴黎 / 贝甜”、“环球 / 影城”、“权力 / 的 / 游戏”、“大 / 钟 / 寺”，每个词的档位都相同。

标注员在工作中常会遇到拿不准的情况。如果无法根据经验做判定，则可以用如下方法辅助判断。如果查询词较短，可以借助百度、谷歌、必应这三大搜索引擎，用排除法逐一判定每个词的权重。假设查询词被分为 4 个词 ABCD，在搜索引擎中分别搜索 BCD、ACD、ABD、ABC，判断转义程度。

- 如果用 BCD 完全搜不到想要的结果，则 A 为核心词。
- 如果用 BCD 搜到的结果中有少部分匹配 ABCD，则 A 为强需求词。
- 如果用 BCD 搜到的结果中大部分匹配 ABCD，则 A 为弱需求词。
- 如果用 BCD 搜到的结果与用 ABCD 搜到的结果基本一致，则 A 为冗余词。

12.2 基于注意力机制的方法

上一节介绍了词权重的定义和人工标注的标准，可以通过人工标注的数据微调 BERT 模型计算词权重。本节介绍一种无需人工标注的方法，它使用相关性数据 $(q, d, y_{q,d})$，其中 y 是查询词 q 与文档 d 的相关性分数。利用相关性数据训练一个注意力层，注意力层中间结果为词向量的权重，可以作为词权重。

我们使用图 12.1 中的双塔模型拟合 q 与 d 的相关性。首先对 q 和 d 做分词，以词粒度输入 BERT 模型，BERT 模型得到若干向量。把查询词的向量表征记作 $\boldsymbol{u}_1, \cdots, \boldsymbol{u}_m$，把文档的向量表征记作 $\boldsymbol{v}_1, \cdots, \boldsymbol{v}_n$。把 [CLS] 符号的向量表征记作 $\boldsymbol{c}$，与 $\boldsymbol{u}_1, \cdots, \boldsymbol{u}_m$ 一同输入单头交叉注意力层，此处把 $\boldsymbol{c}$ 作为 Q，把 $\boldsymbol{q}_1, \cdots, \boldsymbol{q}_m$ 作为 K 和 V，[CLS] 符号位置上的输出向量记作 $\boldsymbol{x}_q$。类似的，把文档 d 的向量表征记作 $\boldsymbol{z}_d$。做训练时，用 $\text{sigmoid}(\boldsymbol{x}_q^\top \boldsymbol{z}_d)$ 拟合 q 与 d 的相关性分数 y。y 可以是人工标注的，也可以是相关性模型（交叉 BERT 模型）的打分。

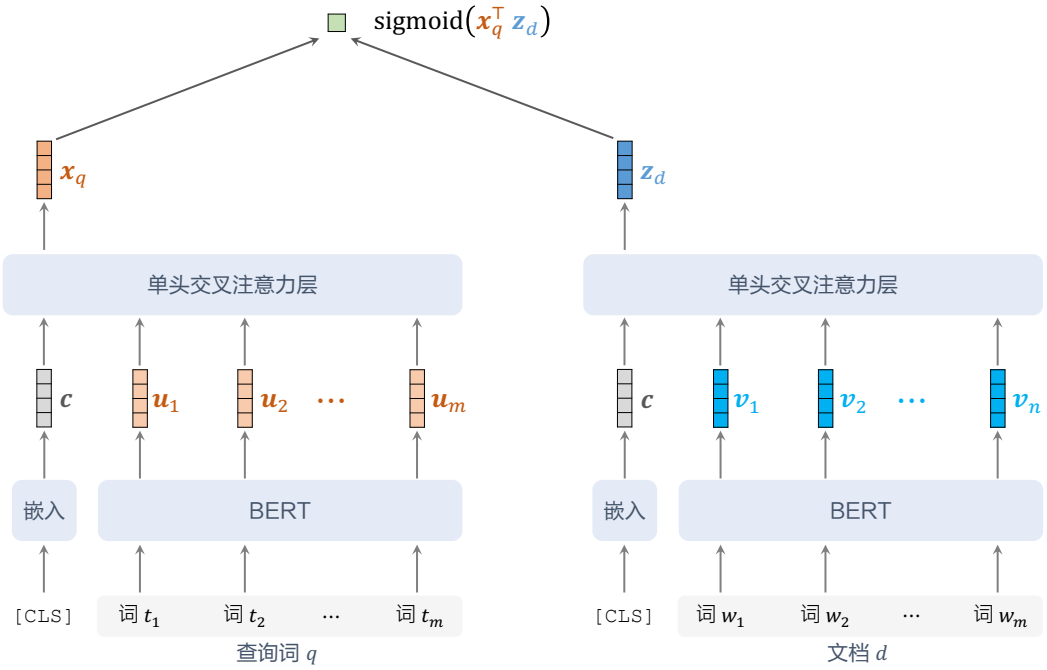

图 12.1: 基于注意力机制的词权重方法

单头交叉注意力层的中间产物为 $\boldsymbol{u}_1, \cdots, \boldsymbol{u}_m$ 的权重，记作 $\alpha_1, \cdots, \alpha_m$。举个例子，设 $\boldsymbol{A}$ 为注意力层中的一个参数矩阵，可以这样计算权重：

$$[\alpha_1, \cdots, \alpha_m] \ = \ \text{softmax}\left[\boldsymbol{c}^\top \boldsymbol{A}\boldsymbol{u}_1, \ \cdots, \ \boldsymbol{c}^\top \boldsymbol{A}\boldsymbol{u}_m\right].$$

可以直接把这些权重作为查询词的词权重。最终输出的向量 $\boldsymbol{x}_q$ 是加权和：

$$\boldsymbol{x}_q \ = \ \alpha_1 \boldsymbol{B}\boldsymbol{u}_1 + \cdots + \alpha_m \boldsymbol{B}\boldsymbol{u}_m,$$

上式中的 $\boldsymbol{B}$ 是注意力层的另一个参数矩阵。上式说明 α_i 决定词 t_i 对查询词的表征向量 $\boldsymbol{x}_q$ 的贡献。如果 α_i 接近 1，说明查询词 q 中只有 t_i 一个词重要。

据说这种方法最初由某大厂发明并申请了专利，后来传到互联网各个大厂。原始方案是用 GRU 这样的循环神经网络从查询词、文档中提取特征，而非使用 BERT 这样的预训练模型。此外，原始方案是以用户点击作为双塔模型拟合的目标 y，而非使用相关性分数。

12.3 知识点小结

- 分词模块将查询词 q 切分成词 $t_1, \cdots, t_m$，这些词的重要性有所不同，它们的重要性被称为词权重。
- 词权重的主要应用是文本召回，包括丢词、计算文本匹配分数、训练召回模型。
- 计算词权重的经典方法是用人工标注的数据微调 BERT 等语义模型。人工标注分为 4 档——核心词、强需求词、弱需求词、冗余词。
- 工业界一种流行的方法是用注意力机制学习词权重。这种方法只需要借助相关性数据，而无需人工标注词权重。

第 13 章 类目识别

搜索引擎中有一套多级类目体系，通常包含数十个一级类目、数百个二级类目、甚至还有三级和四级类目。每篇文档、每条查询词可以属于一个或多个类目。例如查询词 q =“冬季卫衣推荐”，一级类目是“时尚”，二级类目是“穿搭”。再例如 q =“狄仁杰”的一级类目是“影视娱乐”和“游戏”，二级类目是“电视”和“手游”，原因是狄仁杰即是电视剧中的人物形象，也是王者荣耀中的游戏角色，有的用户想搜电视剧，有的用户想搜王者荣耀游戏。

在文档发布和更新时，系统会计算文档类目，将其作为文档特征存储。查询词的类目由 QP 模块在线计算，供下游的召回和排序使用。召回的双塔模型、排序的相关性模型、排序的点击率模型都将文档和查询词的类目作为离散特征。文档和查询词的类目识别都可以看做多标签分类问题，即一条样本有多个标签（类目）。我们在 13.1 节和 13.2 节分别介绍多标签分类的模型和离线评价指标。

13.1 多标签分类模型

文档类目和查询词类目识别都是多标签分类任务。我们首先讨论多标签分类的基线模型，即业界标准的建模方法。如图 13.1 所示，BERT 模型的 [CLS] 符号位置输出一个向量，再经过全连接神经网络和 sigmoid 变换，输出一个 k 维向量 $\boldsymbol{p} = [p_1, \cdots, p_k]$；此处的 k 是类目的数量。注意，由于任务是多标签分类，激活函数是 sigmoid 而不是 softmax，$\boldsymbol{p}$ 的元素都介于 0 和 1 之间，但相加不等于 1。举个例子，如果有 $k = 4$ 个类目，那么 $\boldsymbol{p}$ 可以是 $[0.5, 0.9, 0.7, 0.2]$。做训练时，类目标签被表征为向量 $\boldsymbol{y} \in \{0, 1\}^k$，如果查询词同时属于 t（$\leq k$）个类目，那么 $\boldsymbol{y}$ 中有 t 个元素为 1，其余元素为 0。使用二元交叉熵（binary cross entropy，BCE）作为损失函数：

$$\text{BCE}(\boldsymbol{y}, \boldsymbol{p}) \;=\; -\sum_{i=1}^{k} \Big[y_i \cdot \ln p_i \;+\; (1 - y_i) \cdot \ln(1 - p_i) \Big], \tag{13.1}$$

对损失函数关于模型参数求梯度，做梯度下降更新模型参数。接下来我们讨论几种对训练方法的改进，这些改进方法可以同时用，并不冲突。

第一种改进方法是使用 focal loss[9] 代替交叉熵损失，即将式 (13.1) 替换成下式：

$$\text{FL}(\boldsymbol{y}, \boldsymbol{p}) \;=\; -\sum_{i=1}^{k} \Big[y_i \cdot (1 - p_i)^{\gamma} \cdot \ln p_i \;+\; (1 - y_i) \cdot p_i^{\gamma} \cdot \ln(1 - p_i) \Big]. \tag{13.2}$$

上式中的 γ（≥ 0）是个超参数，如果 $\gamma = 0$，则式 (13.1) 与 (13.2) 等价。可以将 $(1 - p_i)^{\gamma}$ 和 p_i^{γ} 看作对损失函数的加权，预测越准确，则权重越小。γ 越大，则做训练时越关注容易出错的样本，给它们越高的权重。

第二种改进方法是将类目预测看做一个层次分类问题。设类目分为两级，每个一级类目细分为多个二级类目。对于一个二级类目 i，定义集合

$$\mathcal{N}(i) \;=\; \big\{\, j \;\big|\; j \text{ 与 } i \text{ 属于相同一级类目} \,\big\}.$$

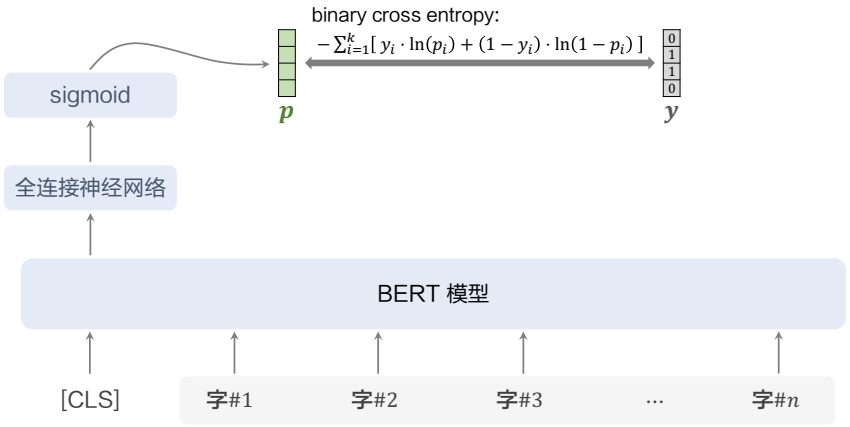

图 13.1: 预测查询词类目、文档类目的模型

举个例子，美妆是一级类目，它下面有 5 个二级类目——彩妆、护肤、美甲、香水、医美，那么

$$\mathcal{N}(\text{彩妆}) = \{\text{护肤, 美甲, 香水, 医美}\},$$
$$\mathcal{N}(\text{护肤}) = \{\text{彩妆, 美甲, 香水, 医美}\}.$$

对于层次分类问题，我们使用递归正则（recursive regularization）[6]，可以参考 2019 年论文[10] 给出的 PyTorch 编程实现。设一共有 k 个二级类目，k 的大小是数百。如图 13.2 所示，图 13.1 中最后一个全连接层的参数矩阵是 $\boldsymbol{W}$，它有 k 行，记作向量 $\boldsymbol{w}_1, \cdots, \boldsymbol{w}_k$，它们各自对应一个二级类目。定义递归正则：

$$R(\boldsymbol{W}) = \sum_{i=1}^{k} \sum_{j \in \mathcal{N}(i)} \|\boldsymbol{w}_i - \boldsymbol{w}_j\|_2^2.$$

我们鼓励 $R(\boldsymbol{W})$ 较小，即如果二级类目 i 和 j 属于相同一级类目，那么它们的参数 $\boldsymbol{w}_i$ 和 $\boldsymbol{w}_j$ 较为接近。做训练时，最小化 BCE（或 focal loss）与递归正则的加权和。

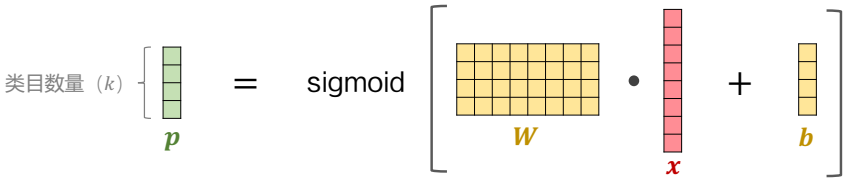

图 13.2: 向量 $\boldsymbol{x}$ 是最后一个全连接层的输入，矩阵 $\boldsymbol{W}$ 和 $\boldsymbol{b}$ 是这一层的参数，向量 $\boldsymbol{p}$ 是经过 sigmoid 变换后的输出

第三种改进方法是针对查询词类目识别的，在预训练和微调中间做后预训练。文档字数多，信息量较大，因此识别文档类目相对比较容易，只需要人工标注数万条标注样本，微调预训练好的模型即可。而查询词很短、字数很少，查询词类目识别比文档类目识别困难很多，最好是做后预训练。根据搜索引擎的曝光日志，挖掘查询词高相关性、高点击率的文档。对于每条查询词，得到数十篇符合要求的文档，统计文档类目，转化成训练所需的标签 $\boldsymbol{y}$。使用数百万条查询词作为后训练样本，基于预训练的 BERT 模型做

多标签分类。只要挖掘的数据质量好，后预训练就可以显著提升查询词类目识别的效果。

13.2 离线评价指标

本节讨论类目识别的离线评价指标。一种是将类目识别看做多标签的二分类问题，使用 micro F1 和 macro F1 作为评价指标。模型输出预测 $\boldsymbol{p} = [p_1, \cdots, p_k]$，其中 p_i 表示查询词（或文档）属于类目 i（比如美甲这个类目）的概率。设置一个阈值 τ，如果 $p_i > \tau$，则判定查询词（或文档）属于类目 i；一条查询词（或文档）可以同时属于多个类目。对于类目 i，计算真阳性率 TP_i、假阳性率 FP_i、假阴性率 FN_i；做计算的时候，独立对待每个类目，忽略其他类目。计算总的真阳性率、假阳性率、假阴性率：

$$\text{TP} = \sum_{i=1}^{k} \text{TP}_i, \qquad \text{FP} = \sum_{i=1}^{k} \text{FP}_i, \qquad \text{FN} = \sum_{i=1}^{k} \text{FN}_i.$$

计算 micro F1：

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \qquad \text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}},$$

$$\text{micro F1} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}.$$

计算 macro F1：

$$\underset{i}{\text{precision}} = \frac{\text{TP}_i}{\text{TP}_i + \text{FP}_i}, \qquad \underset{i}{\text{recall}} = \frac{\text{TP}_i}{\text{TP}_i + \text{FN}_i},$$

$$\text{macro F1} = \frac{1}{k} \sum_{i=1}^{k} \frac{2 \times \text{precision}_i \times \text{recall}_i}{\text{precision}_i + \text{recall}_i}.$$

另一种评价方法是将类目预测看做排序问题，使用 MAP 作为评价指标。模型输出预测 $\boldsymbol{p} = [p_1, \cdots, p_k]$，按照分数给 k 个类目做排序，选择其中分数最高的 t 个类目作为预测。举个例子，一共有 $k = 6$ 种类目，某样本真实的类目是 $\{2, 5\}$，它的 one-hot 编码、模型预测分别是

$$\begin{aligned} \boldsymbol{y} &= [\,0,\ \ 1,\ \ 0,\ \ 0,\ \ 1,\ \ 0\,], \\ \boldsymbol{p} &= [\,0.2,\ 0.7,\ 0.6,\ 0.1,\ 0.5,\ 0.9\,]. \end{aligned}$$

按照模型打分 $\boldsymbol{p}$ 对类目排序，得到 $[6, 2, 3, 5, 1, 4]$。如果只能选 1 个类目，那么选择类目 6，这是个错误，准确率和召回率分别是

$$\text{precision}\,@1 = \tfrac{0}{1}, \qquad \text{recall}\,@1 = \tfrac{0}{2}.$$

如果只能选 2 个类目，那么选择类目 6 和 2，一个错误、一个正确，准确率和召回率分别是

$$\text{precision}\,@2 = \tfrac{1}{2}, \qquad \text{recall}\,@2 = \tfrac{1}{2}.$$

以此类推，得到表 13.1。以召回率作为横轴，准确率作为纵轴绘图，得到准确率—召回率曲线，如图 13.3 所示。曲线与横轴之间区域（图中阴影部分）的面积被称作 average precision（AP）。一个 AP 值对应一条样本（查询词或文档），测试集里有很多条样本，对这些 AP 值取平均，得到 mean average precision（MAP）。

表 13.1: 类目预测的例子

类目排序	是否正确	准确率	召回率
6	×	$\text{precision@1} = \frac{0}{1}$	$\text{recall@1} = \frac{0}{2}$
2	✓	$\text{precision@2} = \frac{1}{2}$	$\text{recall@2} = \frac{1}{2}$
3	×	$\text{precision@3} = \frac{1}{3}$	$\text{recall@3} = \frac{1}{2}$
5	✓	$\text{precision@4} = \frac{2}{4}$	$\text{recall@4} = \frac{2}{2}$
1	×	$\text{precision@5} = \frac{2}{5}$	$\text{recall@5} = \frac{2}{2}$
4	×	$\text{precision@6} = \frac{2}{6}$	$\text{recall@6} = \frac{2}{2}$

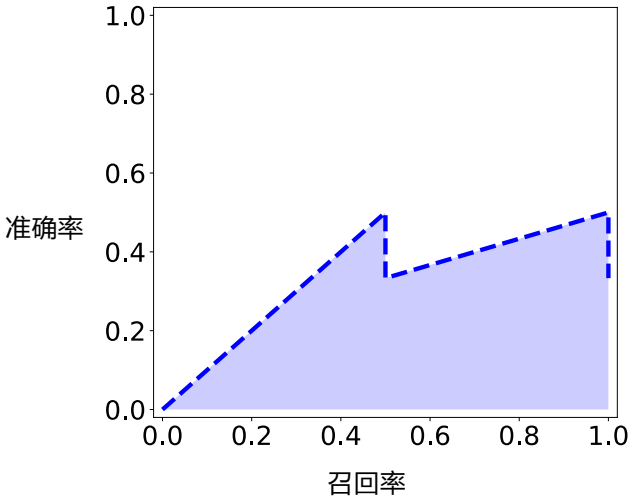

图 13.3: 在表 13.1 的例子中，以召回率作为横轴，以准确率作为纵轴，绘制 precision-recall 曲线，图中的阴影部分为 AP

13.3 知识点小结

- 文档和查询词共享同一个多级类目体系。一篇文档（或一条查询词）可以属于多个类目，且类目是多级的，因此类目识别属于多标签层次分类问题。
- 多标签分类的基线模型以 BERT 为底座，输出层使用 sigmoid 激活函数。做训练时，使用 BCE 或 focal loss 作为损失函数。如果考虑到多级类目属于层次分类问题，可以进一步使用递归正则，将多级类目的信息用于训练。
- 文档的字数多、信息量大，类目识别相对容易。而查询词很短，类目识别很困难，如果不实际搜一下，标注员也无法判定很多查询词的类目。因此，在预训练和微调中间，可以做后预训练，提升查询词类目识别的准确性。
- 如果把类目识别看做多个二分类问题，可以用 micro F1 和 macro F1 作为离线评价指标。如果把类目识别看做排序问题，可以用 MAP 作为评价指标。

第 14 章 意图识别

意图识别是 QP 最重要的任务之一，不止对下游链路有帮助，甚至可以决定调用哪条下游链路。可以这么说，最容易撬动搜索大盘指标增长的，一个是意图识别，另一个是查询词改写。查询词意图分很多种，本章讨论时效性、地域性、用户名这几种意图。

14.1 影响下游链路调用的意图

QP 模块识别出的某些意图会影响下游链路的调用，本节讨论时效性、地域性、用户名、求购这几种意图。我们在 8 章讨论过搜索的时效性问题，这里就不在赘述细节的技术问题，只做简要的概括。QP 需要识别 3 类时效性意图——突发时效性、一般时效性、周期时效性；其中一般时效性又分为强、中、弱、无。识别出的时效性意图对下游的召回和排序起到两方面作用。

- 对于突发时效性、强时效性，应当优先从新文档索引中做召回，召回量不足再从全量索引补齐。
- 召回海选、粗排、精排中，文档年龄在总分中的权重由时效性意图决定，时效性越强，文档年龄的权重越大。

我们在 9 章讨论过搜索的地域性问题，这里简要概括地域性意图、及其对下游链路的影响。“附近的酒店”、“情人节餐厅”、“北京周边游”、“周末好去处”这样的查询词都带有地域性意图，下游链路不能只根据查询词做召回和排序，还需要考虑用户所在城市、甚至用户准确的定位（如果用户运行 APP 获取手机定位）。地域性意图的识别有几个重点。

- 是否对距离敏感。举个例子，“附近的美食”对距离比较敏感，用户很可能是想找步行可到达的餐厅。而“周边游”则是对距离不敏感，驾车能到达的距离通常可以接受。
- 地域性意图的强弱。举个例子，“附近的美食”的地域性意图非常强，用户明确表示只看定位在附近的餐厅。而“日料店”、“星巴克”带有多重意图，有的用户想看附近的店，而有的用户并不想搜附近。
- 显式或隐式意图。举个例子，“附近的美食”、“北京周边游”都是显式意图，即明确表达了搜索的地理范围，可以用规则做识别。而“日料店”、“周边游”则是隐式意图，用户可能想搜附近、同城，但是没有明确表达，需要根据语义做判别。

识别出的时效性意图会影响下游链路的调用，包括召回通道和排序策略。

- 如果识别出查询词带有地域性意图，那么需要根据用户所在地理位置，召回一批距离小于某个阈值的文档。距离的阈值取决于 QP 分析出的对距离的敏感程度。附近文档与全局文档的比例取决于地域性意图的强弱。
- 在排序阶段，需要综合距离与其他因子（相关性、时效性、内容质量、个性化）做排序，QP 认为对距离越敏感，则距离的权重越大。对于“日料店”、“星巴克”这样多意图的查询词，应当采用对队列混排，一个队列为附近的文档，另一个队列为全局的文档，按照地域性意图的强弱对两个队列做融合。

在小红书、抖音、微博、知乎这类社交媒体，用户可能想搜某位用户，查询词是用户 ID、用户名、或用户名的片段。如果不能理解搜用户名的意图，而是简单地匹配文档的标题、正文、关键词，则无法满足用户的需求。比如用户在小红书上搜“李佳琦 Austin”，不能只出他人对李佳琦的讨论和评价，而是应该在显著位置出“李佳琦 Austin”这位 KOL 的账号卡片，并且在搜索结果中出一些这位 KOL 发布的文档。如果查询词是用户 ID，可以根据规则判断查询词符合 ID 的格式，且能在 ID 库中搜到，那么可以将搜到的账号卡片在搜索结果页的显著位置曝光。如果查询词是用户名、甚至用户名片段，则意图的判断比较困难，投入产出比较高的做法是维护一个 KOL 的用户名库，完全命中、或部分命中，则判定带有用户名搜索的意图。

小红书和抖音都有电商业务，是公司的主要营收来源之一。如果 QP 判定查询词有求购意图，那么搜索结果页中的部分结果为商品、店铺、直播，其余结果为普通用户创作的测评等内容。举个例子，在抖音中搜索“阿迪达斯”这样的查询词，搜索结果页中有相关店铺、直播、商品。对于求购意图的查询词，召回和排序都需要特殊的承接。在召回环节，除了使用标准的召回通道之外，还需要额外从商品、店铺、直播的索引中召回一批文档。排序环节更为复杂，需要做多队列混排。普通文档、商品、店铺、直播各自是一个队列，每个队列内部做排序，最终做多队列的融合。商品的排序与普通文档的排序有所不同，商品排序的重要依据是预估的营收，即商品价格乘以转化率。

14.2 知识点小结

- QP 模块需要做查询词的意图识别，包括类目、时效性、地域性、用户名、求购等意图。
- 类目识别可以看做多标签分类任务，主要应用是作为下游链路的特征。
- 时效性、地域性、用户名等意图的识别影响下游链路的调用，包括决定是否使用某些召回通道、以及文档排序的策略。

第 15 章 查询词改写

当搜索引擎功能模块基本就位之后，查询词改写是 QP 模块中撬动大盘指标增长的最有效的手段。基础的改写方法基于词表和规则，高级的改写方法依赖于深度学习模型。15.1 节介绍什么是查询词改写、为什么做查询词改写。15.2 节讲解基于分词的改写，它在链路上位于分词之后，对分词结果做同义词、上下位词替换。15.4 节讲解基于意图的改写，这种改写需要与召回联动，结合起来解决文本召回无法解决的问题。15.3 节详细讲解基于相关性的改写，它对改写和召回联合建模，使原词 q 与改写召回的文档 d 有高相关性。

15.1 改写的目标

查询词改写的意思是把原始的 q 改写成 q'，用 q' 召回的文档 $d_1, \cdots, d_k$ 作为召回的补充。如图 15.1 所示，改写的目标是让 q 与召回的文档 $d_1, \cdots, d_k$ 相关，而非简单地让 q 与 q' 相关。举个例子，q 是“男士夏季穿搭建议”，改写后的 q' 是“夏季穿搭”，两者具有高相关性。由查询词 q' 召回了一篇标题为“夏季女大学生穿搭小技巧”的文档 d，q' 与 d 具有高相关性，然而 q 与 d 不相关。

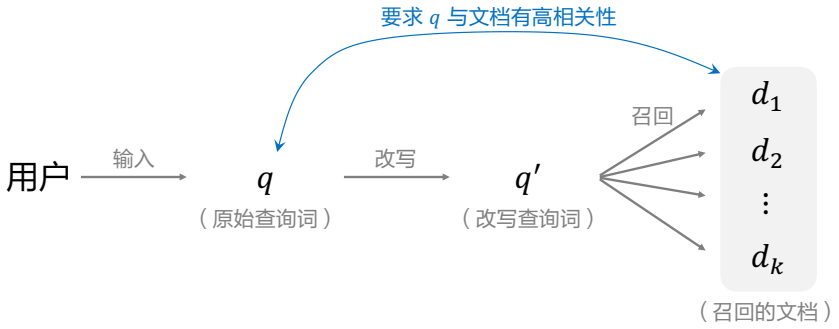

图 15.1: 原始查询词为 q，改写后的查询词为 q'，与 q' 最相关的 k 篇文档为 $d_1, \cdots, d_k$，改写的目标是让 q 与 $d_1, \cdots, d_k$ 相关

查询词改写的目标是让更多相关的文档被召回，增加供给。反映在评价指标上，改写可以降低生僻、低频查询词的无结果、少结果率，提升用户规模、留存、有点比、首点位置、换词率等大盘指标。具体来说，查询词改写起到如下的作用。

- 第一，解决语义鸿沟问题。搜索中的相关性指的是查询词 q 与文档 d 的语义匹配，无需在字面上匹配。如果不做改写，用 q =“粤菜”搜不到 d =“潮汕美食”，用 q =“身高 160 体重 120 女穿搭推荐”搜不到 d =“微胖女孩穿搭技巧”，用 q =“reinforcement learning”搜不到 d =“强化学习教程”，尽管例子中的 q 与 d 具有高相关性。在解决语义鸿沟这一点上，查询词改写与向量召回起到相似的作用，都可以克服文本检索的不足。

- 第二，查询词中经常包含简称、别称、不规范表述，或者查询词过长，都会导致搜索零结果、少结果。把长尾查询词改写成头部查询词，可以有效解决这类问题，大幅增加相关文档的召回量。举个例子，“强化学习”是规范的表达，但也有人用“增强学习”，如果直接检索后者，召回的文档数量远少于检索前者，这个问题可以通过改写解决。

在 QP 的链路上，查询词改写可以在分词之前，也可以在分词之后。如果在分词之前，通常是基于深度学习模型根据 q 召回或生成一批查询词 $\{q'\}$，然后判别 (q, q') 是否是合理的改写。如果在分词之后，通常是基于同义词、上下位词、近义词做替换，或者根据查询词意图做相应的改写。

15.2 基于分词的改写

在分词阶段，一条查询词 q 被分割成为多个词 $t_1, \cdots, t_m$。可以基于词表和规则对 t_i 做替换，包括同义词替换和上下位词替换，从而扩大召回量。词粒度的改写主要有以下几种方式。

- 可以基于同义词表做替换。例如 q =“民族舞 教程”，可以把“民族舞”替换成“民间舞”，把“教程”替换成“教学”。例如 q =“苹果手机 14”，可以把“苹果手机”替换成“iphone”。例如 q =“老谋子 的 电影”，可以把“老谋子”替换成“张艺谋”。
- 可以根据上下位词表做替换，把上位词替换成下位词。例如把“民族舞”替换成“蒙古舞”、“新疆舞”，把“上海”替换成“黄浦区”、“浦东新区”。注意，不能把下位词 q 替换成上位词 q'，否则 q' 召回的文档 d 很可能与 q 不相关。举个例子，q =“上海的餐厅”，q' =“浦东的餐厅”，召回的文档完全符合用户意图。这说明将上位词改写为下位词是合理的。但是反过来不行，把下位词 q' =“浦东的餐厅”改写为上位词 q =“上海的餐厅”，可能召回“闵行区的餐厅”，不符合用户意图。
- 还可以基于规则做词粒度的改写。把阿拉伯数字和中文数字相互替换，例如“双 11”↔“双十一”。把中文和外文相互替换，例如“reinforcement learning 教程”↔“强化学习教程”。

同义词表、上下位词表的来源主要是公开数据，外加一些站内挖掘和人工标注的。常用的开源的中文知识图谱、词表有这几个：

- 1.4 亿中文知识图谱 KnowledgeGraphData: https://github.com/ownthink/KnowledgeGraphData。
- 词林扩展板：https://github.com/yaleimeng/Final_word_Similarity。
- 中文近义词表：https://github.com/Keson96/SynoCN，内含 8 万多个词，33 万多条近义关系。

除了利用公开数据，也需要用算法挖掘同义词，并用人工标注，这样的同义词表、上下位词表更贴近站内的查询词和文档。

查询词改写的目的是召回更多相关的文档。以 q =“民族舞 教程”为例，假如不做改

写，那么召回的逻辑为

$$\text{民族舞} \quad \text{and} \quad \text{教程},$$

它的意义是分别召回包含“民族舞”的文档、包含“教程”的文档，对两个集合取交集，作为最终召回结果。如果做改写，那么先做 or 逻辑，再做 and 逻辑：

$$(\text{民族舞} \ \text{or} \ \text{民间舞} \ \text{or} \ \text{蒙古舞}) \quad \text{and} \quad (\text{教程} \ \text{or} \ \text{教学}).$$

15.3 基于相关性的改写

假设我们已有相关性模型，给定查询词 q 和文档 d，模型给出相关性分数 $\text{rel}(q,d)$。我们挖掘搜索日志，并利用相关性模型打分，构造出海量数据，训练一个改写召回模型和一个改写判别模型。改写召回模型把 q 改写成 q'，改写判别模型判断 $q \to q'$ 是否合理。改写的目标是把任意查询词 q 改写为头部查询词 q'，使得 q' 召回的文档与 q 具有高相关性。本节介绍我们团队设计和落地的方法，对业务指标有大幅的提升。我们完成这个项目之后，发现与阿里巴巴 2019 年的论文[17] 有很多相同之处。

15.3.1 数据的构造

我们准备一个数据集，用于训练改写召回和判别模型。通过挖掘曝光日志，我们选取两个查询词的集合 $\mathcal{T}$ 和 $\mathcal{Q}$。$\mathcal{T}$ 包含搜索频次最高的 500 万查询词。$\mathcal{Q}$ 包含数千万条查询词，对高频和低频查询词都有覆盖。利用已有的相关性模型打分，构造出数据集 $\{q, q', y\}$，其中 $q \in \mathcal{Q}$，$q' \in \mathcal{T}$，$y \in [0,1]$。

构造一个改写的候选集合 $\{q \to q'\}$，其中 $q \in \mathcal{Q}$ 作为原查询词，$q' \in \mathcal{T}$ 作为候选的改写词。构造候选集合的方法有两种，一种是基于语义模型的，另一种是基于用户行为的。

- 使用预训练的 BERT 模型把 $\mathcal{Q}$ 和 $\mathcal{T}$ 中的查询词表征为向量。对于每个查询词 $q \in \mathcal{Q}$，用它的向量表征在 $\mathcal{T}$ 中查找向量相似度最高的查询词 q'，构成 $q \to q'$。
- 构造一个二部图，以查询词和文档作为节点。用户搜索查询词 q，如果文档 d 与 q 高相关、且被用户点击，那么 q 与 d 之间有一条边。在图中，q 与 q' 共同的邻居（文档）越多，则 q 与 q' 的关联越强，以这样的方式挖掘出 $q \to q'$。

值得注意的是，用这些方法得到的 $q \to q'$ 未必是合理的改写，也就是说数据集中有正样本，也有负样本。举个例子，q 可能是 q' 的上位词，也可能是下位词。如果是上位词，则改写 $q \to q'$ 是合理的；如果是下位词，则不合理。

挖掘搜索日志，对于头部查询词 $q' \in \mathcal{T}$，挖掘与 q' 高相关且高点击的文档 $d_1, \cdots, d_k$，并以 $q' \to [d_1, \cdots, d_k]$ 的方式存储。对于候选的改写 $q \to q'$，取回 q' 高相关的文档 $d_1, \cdots, d_k$，利用已有的相关性模型打分：

$$y = \frac{1}{k} \text{rel}(q, d_i).$$

y 越高，说明改写 $q \to q'$ 越合理。通过这种方法，我们构造出数据集 $\{q, q', y\}$。

这种构造训练数据的方法的好处是无需人工标注，只需要要挖掘曝光日志、用相关性模型打分。这种方法的坏处是所需算力较大。如果集合 $\mathcal{Q}$ 中包含 2 千万条查询词 q，每条 q 有 10 条改写的 q'，那么一共有 2 亿条改写 $q \to q'$。设 $k = 10$，那么相关性模型一共要给 20 亿对 (q, d) 打分，才能构造出所需的数据集。

15.3.2 改写召回模型

向量召回的基本思想是用 BERT 把查询词表征为向量，用原始查询词 q 从头部查询词集合 $\mathcal{T}$ 中召回若干查询词 $\{q'\}$。训练召回模型的方法如下。训练，具体方法如下。

- 做训练时使用上文构造的数据集 $\{(q, q', y)\}$。此外，还需要添加一批简单负样本，即随机组合两个无关的查询词 $q_1 \in \mathcal{Q}$ 和 $q_2 \in \mathcal{T}$，将 $(q_1, q_2, 0)$ 添加到数据集。
- 使用图 15.2 所示的双塔模型，把查询词 q 和 q' 的向量表征记作 $\boldsymbol{x}$ 和 $\boldsymbol{x}'$。做 pointwise 训练，使用交叉熵损失函数，让 $\text{sigmoid}(\boldsymbol{x}^\top \boldsymbol{x}')$ 拟合改写分数 y。

在训练结束之后，用 BERT 模型提取所有头部查询词 $q' \in \mathcal{T}$ 的向量表征 $\boldsymbol{x}'$，把 $(q', \boldsymbol{x}')$ 存入 Faiss 这样的向量数据库。做召回时，用 BERT 模型提取原查询词 q 的向量表征 $\boldsymbol{x}$，在向量数据库中查找内积 $\boldsymbol{x}^\top \boldsymbol{x}'$ 最大的若干向量 $\{\boldsymbol{x}'\}$，返回其对应的查询词 $\{q'\}$ 作为 q 的改写。

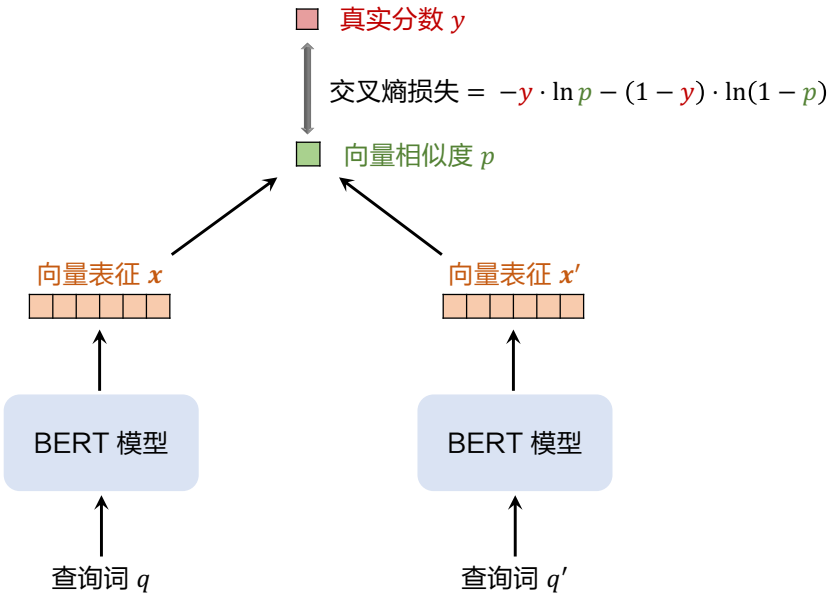

图 15.2: BERT 模型把查询词表征为向量，向量相似度拟合改写分数 y

也可以用改写生成模型替代上述改写召回模型。文本生成使用 Transformer 模型，把原始查询词 q 输入编码器，由解码器生成一批查询词 $\{q'\}$，作为改写结果。做训练的时候只用数据集 $\{(q, q', y)\}$ 中的正样本（即 y 大于某阈值），得到一个子集。用训练机器翻译的方法训练 Transformer 模型，即让解码器的输出逐字拟合 q'。在训练结束之后，用 Transformer 做改写。为了生成很多条改写的查询词 $\{q'\}$，解码器生成文字时做随机抽样或 beam search。

15.3.3 改写判别模型

改写召回和生成模型都容易犯错，即改写 $q \to q'$ 不合理，因此需要在改写召回、生成之后做判别。尤其是改写召回模型，虽然它的推理速度快，但是它的改写结果不可靠。改写召回使用双塔模型，它具有对称性，如果模型认为 $q \to q'$ 合理，那么也会认为 $q' \to q$ 合理。也就是说，它可以正确地将上位词改写成下位词，但也会错误低将下位词改写成上位词。

如图 15.3 所示，我们使用 BERT 模型作为判别器，记作 $f(q, q')$，用于判断改写 $q \to q'$ 的质量。将原查询词 q 与改写查询词 q' 输入 BERT 模型，使用符号、位置、段落三种向量表征的加和作为一个字的向量表征。做训练时，使用上文描述的数据集 $\{(q, q', y)\}$。此外，将查询词 $q' \in \mathcal{T}$ 到它自身的改写作为正样本，把 $(q', q', 1)$ 加入数据集。以交叉熵作为损失函数，$f(q, q')$ 拟合 y。注意，判别器是非对称的，即 $f(q, q') \neq f(q', q)$，因此判别器不会犯上下位词改写的错误。

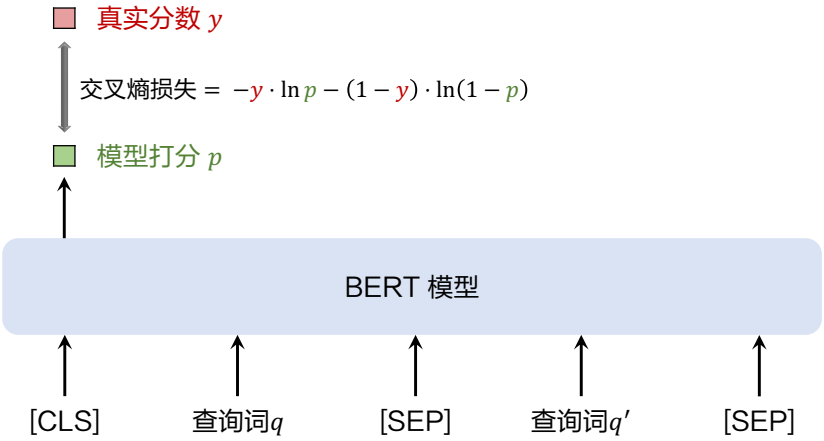

图 15.3: 用 BERT 作为判别器，给改写打分 $p = f(q, q')$，目标是拟合 y

15.3.4 离线的改写

查询词改写模块有一个事先算好的索引 $q \to \text{List}\langle q' \rangle$，比如每个查询词对应 5 个改写的查询词。索引只需很小的存储，就能覆盖线上绝大部分的线上请求。比如，索引上存 1 亿条查询词，每条查询词有 5 条改写查询词，设平均每条查询词有 20 个字符，那么索引只需要 24GB 的存储。如果用户在线上搜索的 q 很生僻、不在索引上，那么在线上用改写召回模型得到改写的候选集，再用判别器做排序，作为最终的改写结果。

在离线构造索引 $q \to \text{List}\langle q' \rangle$ 时，没有响应时间和计算量的约束，可以采用复杂、计算量大的方式获得非常优质的改写结果。可以用类似推荐和搜索的召回、粗排、精排的三级漏斗离线做查询词改写，这种方法比召回 + 判别更准确。

- 对于每个 q，用改写召回和生成得到数百条查询词 $\{q'\}$，作为候选集。
- 用判别器给每对 (q, q') 打分。对于一条原查询词 q，根据分数给 $\{q'\}$ 排序，保留分数最高的数十条改写查询词。

- 对于每对 (q, q')，设 $d_1, \cdots, d_k$ 为 q' 高相关的 k 篇文档，用 $y = \frac{1}{k}\sum_{i=1}^{k} \mathrm{rel}(q, d_i)$ 作为 $q \to q'$ 的打分。对于一条原查询词 q，根据分数给 $\{q'\}$ 做排序，保留分数最高的几条查询词，作为最终的改写结果。

用三级漏斗的方法做离线改写，不但能找到优质的改写词，还有海量的 (q, q', y) 作为副产物，扩充了的数据集，可以训练出质量更好的召回、生成、判别模型。

、

15.4 基于意图的改写

属性检索意图，例如“60 寸 3000 元以下电视机”、“2TB 西部数据移动硬盘”、“身高 161 体重 125 女穿搭建议”这类查询词都指定了一些属性值。用户搜“3000 元以下电视机”，那么一款价格为 2000 的电视机符合用户需求，但是标准的文本检索却搜不到。用户搜“身高 161 体重 125 女穿搭建议”，一篇文档“身高 160 微胖女孩穿搭攻略”符合用户意图，文本检索也找不到。对于属性检索意图的查询词，需要根据物品的品类对查询词做结构化，召回阶段做相应的属性检索。

- 如果品类是“电视机”，那么属性包括品牌、尺寸、分辨率、价格。对于上面例子中的查询词，限定尺寸等于 60，价格小于 3000。
- 如果品类是“移动硬盘”，那么属性包括品牌、容量、价格。对于上面例子中的查询词，限定品牌等于西部数据，容量等于 2TB。
- 如果品类是“穿搭”，那么属性包括性别、年龄、品牌、身高、体重、等等。对于上面例子中的查询词，限定性别为女，身高约等于 161，体重约等于 125。

提问意图，例如“instagram 怎么注册”、“什么车性价比高”、“新生儿得了湿疹用什么药”，这类查询词都是带有找答案的目的。对于提问意图，搜索引擎有特殊的处理方式，比如直接把标准答案排在第一，并在排序阶段给权威性分数有较高的权重。除此之外，查询词改写也可以帮助召回相关的文档。把“instagram 怎么注册”改写成“instagram 注册教程”，把“什么车性价比高”改写成“性价比高的汽车”，把“新生儿得了湿疹用什么药”改写成“新生儿湿疹用药”，这样可以更好地匹配到相关的文档。

地域性意图，例如“附近的火锅店”、“本地的景点”这类查询词的目标是寻找定位在用户附近或同城的 POI。举个例子，上海的用户搜“附近的火锅店”，结果是“北京前门附近的火锅店”，这样的结果显然不符合用户意图。对于地域意图的查询词，需要做特殊的改写和下游链路的承接。

- “附近的火锅店”的意图是寻找距离较近的火锅店，应当把查询词改写为“火锅店”，并把用户的地理定位（GeoHash）传递给召回。召回阶段，检索“火锅店”，并且过滤掉定位与用户距离大于某个阈值的文档。在排序阶段，要结合距离、相关性、内容质量、时效性、个性化做综合排序。
- “本地的景点”的意图是寻找同城的景点，应当把查询词改写为“景点”，并把用户所在城市传递给召回。召回阶段，检索“景点”，并过滤掉非同城的文档。排序阶段与其他意图无异，不需要使用距离，因为用户只是找同城，不在乎距离远近。

15.5 本章小结

- 查询词改写的用途是召回更多相关的文档。把改写和召回记作 $q \to q'$ 和 $q' \to d$，改写的目标是让 (q, d) 具有高相关性，而非让 (q, q') 语义相似。
- 在分词之后，可以对分词结果做同义词、近义词、上下位词替换，这样召回的结果仍然与原查询词相关。
- 在分词之前，可以做整条查询词的改写，使用改写召回模型和改写判别模型。训练改写模型的数据来自搜索日志和相关性模型的打分，无需人工对改写做标注。
- 可以根据查询词的意图做相应的改写。对于属性检索意图、提问意图、地域性意图，有特定的改写方法，下游链路也需要做相应的承接。

第五部分

召回

第 16 章 文本召回

文本召回是搜索引擎最基础、最重要的召回通道，但也是搜索引擎中最成熟的技术，可供算法工程师发挥的空间不大。本书只简要描述倒排索引的原理，具体技术细节可以参考综述论文[22] 和中文教材[26] 。

16.1 倒排索引

在文档发布（或被爬虫抓取）之后，系统会对文档做分词，然后将文档添加到倒排索引上。倒排索引是一种特殊的数据结构，给定一个词，可以用倒排索引找到所有包含这个词的文档。倒排这个名字是与正排相对的。如果索引是“文档 ID → 文档内容”，则索引被称作正排索引。反之，如果索引是“文档内容 → 文档 ID”，则索引被称作倒排索引。建立倒排索引是一项非常成熟的技术，ElasticSearch 等常用的系统都有建立倒排索引的工具。当代的搜索算法工程师几乎无需深入理解倒排索引的技术细节，只需要了解基本原理。

文本召回用到的倒排索引是“词 → List⟨文档 ID⟩”，此处的词被称作 index term，文档 ID 列表被称作 posting list。举个例子，给定两篇文档 a 和 b：

$$a \to [\alpha, \beta, \gamma, \alpha], \qquad b \to [\gamma, \delta, \gamma]. \tag{16.1}$$

那么倒排索引为：

$$\alpha \to \{a\}, \qquad \beta \to \{a\}, \qquad \gamma \to \{a, b\}, \qquad \delta \to \{b\}.$$

文本召回之后，需要计算 TF-IDF、BM25、词距等文本匹配分数，因此倒排索引存储下面的信息：

- $\mathcal{D}_t$：包含词 t 的文档的集合；
- $\mathrm{tf}_{t,d}$：词 t 在文档 d 中出现的次数；
- $\mathcal{L}_{t,d}$：词 t 在文档 d 中出现的位置。

倒排索引是这样的形式：$t \to \left\{ \left(d, \mathrm{tf}_{t,d}, \mathcal{L}_{t,d} \right) \right\}_{d \in \mathcal{D}_t}$。以式 (16.1) 为例，倒排索引包含：

$$\begin{aligned}
\alpha &\to \left\{ \left(a, 2, [1, 4] \right) \right\}, \\
\beta &\to \left\{ \left(a, 1, [2] \right) \right\}, \\
\gamma &\to \left\{ \left(a, 1, [3] \right), \left(b, 2, [1, 3] \right) \right\}, \\
\delta &\to \left\{ \left(b, 1, [2] \right) \right\}.
\end{aligned}$$

16.1.1 倒排索引的创建

有三种创建索引的经典算法，分别是两次遍历算法、排序算法、归并算法。两次遍历算法需要在内存中完成，它的第一轮遍历计算所需内存大小，第二轮遍历真正建索引。第一轮遍历中，挨个对每篇文档做分词，并统计每个词在多少篇文档中出现、以及每个词在数据集中出现的总次数。有了这些统计量，就可以估算出内存开销，为每个词分配

内存空间，用于存储它对应的文档信息。第二轮遍历真正开始建立倒排索引。挨个对每篇文档 d 做分词，得到每个词 t 在文档 d 中出现的频次 $\mathrm{tf}_{t,d}$ 和位置 $\mathcal{L}_{t,d}$。把 $(d, \mathrm{tf}_{t,d}, \mathcal{L}_{t,d})$ 填充到事先为词 t 分配的内存空间中。

排序算法对内存的需求较小，无论内存是否存得下倒排索引，都可以使用排序算法。对每篇文档 d 做分词，得到大量的 $(t, d, \mathrm{tf}_{t,d}, \mathcal{L}_{t,d})$ 四元组。达到内存的配额时，开始做排序，将排序结果写入文件，存入硬盘。具体按照 t 做升序排列；如果 t 相同，则按 d 做升序排列。在处理完全量文档之后，开始对文件做归并，得到最终的倒排索引。排序算法需要自始至终在内存中维护一个词典，用于将词映射到 ID，中途不将词典写入硬盘。

归并算法与排序算法非常类似，都是在内存无法存储全量倒排索引的情况下，使用磁盘存储中间结果。算法以此处理文档，建立倒排索引。当倒排索引达到内存配额时，将倒排索引存入文件，清空内存，然后从零开始建下一个倒排索引。用这种方法，相当于把文档数据集划分为 k 部分，建立 k 个独立的倒排索引。最终对 k 个倒排索引做归并，建立一个总的倒排索引。归并算法不需要自始至终在内存中维护一个词典，这是与排序算法的区别。归并算法天然可并行，适合 MapReduce 等现代的分布式系统。将文档数据集划分成 k 个文件，分布在 k 个节点上。做 map 操作把文档转化为四元组，然后做 reduce 操作按照词 t 做归并，生成倒排索引。

16.1.2 倒排索引的更新

系统中的文档会不断发生变化，包括新增、删除、修改。当站内有新的文档发布、或站外的新文档被爬虫抓取到，需要将这些新文档录入倒排索引，这样才能被文本召回通道检索到。如果文档被删除、或网页链接失效，那么要避免这样的文档被召回，否则用户会无法打开文档。如果文档被修改，那么文档中的词发生变化，文档应当从被删除的词的 posting list 移除、并添加到新增的词的 posting list 上。在实际的系统中，为了保证效率，通常会使用一些策略解决新增、删除、修改带来的问题。

应对新文档的策略是在内存中维护一份临时的倒排索引，把新发布的文档添加到临时倒排索引上。具体来说，对新文档做分词，将文档 ID 和其他信息添加到相应的 posting list 的末尾即可。两份索引的方式可以保障新文档被检索到，但仍需要定时对倒排索引做更新，将增、删、改的变化反映在倒排索引上。有三种更新索引的方式：全量更新、归并、增量更新。全量更新的意思是从零开始重建索引，这种方式最简单粗暴，也最容易实现。如果分词算法和词典有较大变化，则需要做全量更新。归并的意思是定期将临时的新文档索引并入全量索引，可以是简单地把新文档索引的文档列表接在全量索引的文档列表后面，也可以是创建一份新的全量索引。增量更新的意思是每次更新一个词的文档列表。如果临时索引中一个词的文档列表很长，即文档列表中包含的文档数量较多，则将该词的文档列表并入全量索引。

如果文档被删除，照理说应当将文档从多个词的 posting list 中移除。但是这种的操作代价太大，实践中不可行。常用的策略是后置过滤，即实时维护一张哈希表，存储有效文档的 ID。在做完召回之后，用哈希表做过滤，从召回结果中排除无效文档。如果文档有修改，可以认为旧文档被删除，添加一篇新文档。

16.1.3 分布式索引

由于单机的存储和算力有限，文本召回常采用分布式索引，即索引被划分到多机上。有两种常见的索引划分方式：按词划分（term-partitioned index）、按文档划分（document-partitioned index）。按词划分意思是把词典分成 k 份，每一份索引只存一部分词，但是有这些词完整的文档列表。给定一个词 t，同时在 k 份索引上检索 t，只会命中一份索引。按文档划分的意思是把 n 篇文档分成 k 份，每一份索引上几乎包含完整的词典，但是一份索引上只有 n/k 篇文档。

在实践中通常是按文档划分索引，每个节点从一份索引中做检索。这样有很多好处。第一，给定一个查询词，可以独立并行在 k 份索引上做检索，得到 k 份独立的文档列表，整个过程中，节点之间不需要做通信。第二，假如期间有若干节点挂掉，并不会干扰剩余节点的任务，剩余节点仍然能正常返回文档列表。虽然召回的文档数量减少，但用户几乎感知不到。第三，按照文档划分索引，每个节点承担的计算几乎相等，系统的负载均衡很好。

倒排索引的大小与分词粒度相关。如果分词粒度很细，比如把“亚马逊雨林”作为一个词，那么词典会非常大，倒排索引的规模也会非常大。如果分词粒度较粗，比如把“亚马逊雨林”分成“亚马逊”+“雨林”，那么词典会小很多，倒排索引也会相应减小。在实践中会做出取舍，分词粒度不会太粗或太细。

16.2 文本召回

文本召回主要是利用倒排索引做布尔检索。倒排索引上记录“词 $\rightarrow$ List⟨ 文档 ID⟩”。根据分词结果，取回多个文档列表，然后用逻辑 and 和 or 逻辑筛选符合要求的文档。举个例子，查询词“少儿民族舞”被分成 $\mathcal{Q} = \{$少儿, 民族舞$\}$，再加上同义词改写，最终会做这样的布尔检索：

$$\big(\text{少儿 or 儿童 or 幼儿 or 宝宝}\big) \ \text{ and } \ \big(\text{民族舞 or 民间舞}\big).$$

比方说，如果一篇文档同时包含“儿童”和“民族舞”这两个词，则文档会被作为召回结果返回。

值得注意的是，一篇文档符合布尔检索的逻辑，不意味着文档与查询词相关。举个例子，查询词“亚马逊雨林”被分成 $\mathcal{Q} = \{$亚马逊, 雨林$\}$，当用户搜索“亚马逊雨林”时，文本召回会寻找同时包含“亚马逊”和“雨林”的文档。检索到的文档中有很多无关的，举个例子，文档“我在亚马逊上网购了一本书，介绍东南亚热带雨林的植物群落”被命中，它与查询词无关。因此，文本检索还需要考虑到词之间的距离，比如设置距离 ± 5，只有当两个词距离小于阈值时，文档才会被召回。

如果查询词较长或生僻，召回的文档数量可能会很小，需要丢弃一部分的词。这种方法叫做松弛召回，大致思路如下。

- 如果分词结果 $\mathcal{Q}$ 包含 4 个或更多词，那么只需要命中其中 75% 的词，允许丢弃非核心词。举个例子，如果包含 5 个词，那么只需要命中其中 4 个，允许丢弃任意一

个非核心词。这是最低程度的松弛。

- 如果召回的数量小于某个阈值，比如 45，那么就会触发更大程度的松弛，丢弃多个非核心词，甚至丢弃一定比例的核心词，目的是确保能召回足够多的文档。

做上述的松弛召回需要知道词的重要性，哪些是核心词、哪些是非核心词。这个信息是上游 QP 中的词权重模块计算出的，传递给召回模块，供召回使用。

16.3 知识点小结

- 对文档分词之后，构造倒排索引“词 → List⟨文档 ID⟩”。此处的词被称作 index term，文档列表被称作 posting list。为了方便计算 BM25、BM25TP 等文本匹配分数，倒排索引中还记录词在文档中的出现次数和出现位置。
- 构建索引的算法包括两次遍历算法、排序算法、归并算法。其中归并算法适用于 MapReduce 等现代分布式系统。
- 文档会实时发生新增、删除、修改，通常使用高效的策略应对。常用临时倒排索引应对新增，用有效文档 ID 的哈希表过滤掉召回的无效文档，把修改看做删除 + 新增。
- 由于单机存储和算力有限，实践中常用分布式索引。分布式索引有按词划分、按文档划分这两类，实践中常用后者。
- 做文本召回时，先对查询词做分词和改写，每个词检索到一个 posting list，然后做 and 和 or 逻辑运算，得到最终的召回结果。
- 为了应对查询词过长、召回结果不足的问题，召回中根据词权重做丢词，也被称作松弛召回。这样会召回大量中、低相关性的文档，解决数量不足的问题。

第 17 章 向量召回

随着深度学习技术的发展，向量召回成为搜索引擎中一类重要的召回通道，在召回中占据的配额只略低于文本召回。向量召回与查询词改写起到类似的作用，都可以解决语义鸿沟问题，召回文本不匹配、但是语义匹配的文档。向量召回中需要大量的深度学习、NLP 技术，算法工程师的发挥空间很大，对向量召回的优化可以显著提升主要业务指标。搜索引擎中的向量召回主要分为相关性召回、个性化召回这两类，两者的本质区别是拟合的目标不同，分别是相关性分数和用户点击行为。我们在 17.2 和 17.2 节讲解两类向量召回的方法，在 17.3 节讨论模型的线上推理。

17.1 相关性向量召回

相关性召回使用双塔模型，分别提取查询词 q 和文档 d 的向量表征，用向量相似度拟合 (q, d) 的相关性分数。模型结构仍然是 6.3 节介绍的双塔 BERT 模型，与图 6.3 中的模型完全相同，区别在于构造数据与线上推理的方法。本节讨论构造数据与训练模型。

召回的目的是从数亿文档中选出一批可能与 q 相关的文档，重点在于区分无关与可能相关；而海选的目的是从数万可能相关的文档中排除掉低档位的文档。由于模型的用途不同，训练数据应当有差异。训练召回模型的数据中需要掺大量低相关性的样本，而训练海选模型的数据中 4 个档位的数据应当比较均匀。构造负样本的方法主要有以下两种。

- 通过随机匹配 q 和 d，可以构造出不相关的 (q, d)，作为简单负样本。但这样的样本过于简单，绝大多数 q 和 d 甚至没有重叠的词，只用这样的负样本学出的模型太弱。
- 如果 (q, d) 的相关性较高，从 q 中删除一个核心词得到 q'，那么 (q', d) 的相关性通常较低，作为困难负样本。

构造训练样本时还需要注意头、中、尾部查询词的比例不能失衡。假如绝大多数样本都用头部查询词，那么模型在非头部的查询词上表现不好。

构造好训练样本之后，用相关性 BERT 大模型给 (q, d) 打分，记作 y。把 q 和 d 的向量表征记作 $\boldsymbol{x}$ 和 $\boldsymbol{z}$，让 $\text{sigmoid}(\boldsymbol{x}^\top \boldsymbol{z})$ 拟合 y，这种方法叫做知识蒸馏。相关性召回模型是纯基于查询词和文档内容的，既不使用文档 ID embedding，也不使用个性化特征。因此，无需每天更新模型参数、文档的向量表征。仅当当文档发布、修改时，用模型计算文档 d 的向量表征 $\boldsymbol{x}$，把 $(\boldsymbol{x}, d)$ 存入向量数据库，这样文档就可以被检索到。

17.2 个性化向量召回

个性化向量召回也使用双塔模型，左塔提取查询词 q 和用户 u 的向量表征 $\boldsymbol{x}$，右塔提取文档 d 的向量表征 $\boldsymbol{z}$，用向量相似度预测用户是否点击文档。模型结构仍然是 10.2

节中的图 10.8，区别在于构造数据与线上推理的方法。在本节中，我们把双塔模型计算的向量相似度记作 $\text{sim}(u, q, d)$，它可以是向量内积 $\boldsymbol{x}^\top \boldsymbol{z}$，也可以是余弦相似度 $\frac{\boldsymbol{x}^\top \boldsymbol{z}}{\|\boldsymbol{x}\|_2 \cdot \|\boldsymbol{z}\|_2}$。

熟悉推荐系统的读者不难发现，搜索引擎与推荐系统中的个性化向量召回原理基本一致，唯一的区别是此处的左塔额外使用查询词的特征。训练模型的数据分为正样本、简单负样本、困难负样本，需要调各种样本的比例。

- 正样本是从曝光日志中获取的，如果用户点击文档，那么 (u, q, d) 就是一条正样本。曝光给用户的文档，排名通常比较靠前，说明相关性较高；用户点击文档，说明用户对文档感兴趣。
- 简单负样本是用曝光日志中的 (u, q) 与随机的 d 组合成的，q 与 d 大概率不相关，用户 u 大概率对 d 不感兴趣。也可以用 batch 内负样本，即随机选取 b 条正样本组成 batch，记作 $\{(u_i, q_i, d_i)\}_{i=1}^{b}$；对于任意 $i \neq j$，(u_i, q_i, d_j) 是一条简单负样本。
- 困难负样本是被召回、但是没有通过海选的数万条样本，或者通过海选、但是没通过粗排的数千条样本。这样的 q 和 d 多少有一些相关性，而且 u 可能对 d 感兴趣，只是相关性和个性化分数不够高。

17.2.1 pairwise 训练方法

训练双塔模型的方法主要有 pairwise 和 batch 负采样这两种方法。pairwise 训练的目标是让双塔模型能够区分正负样本。给定 u 和 q，设 d^+ 为正样本，d^- 为负样本。训练的目标是让正负样本有区分度，即 $\text{sim}(u, q, d^+) - \text{sim}(u, q, d^-)$ 越大越好。可以使用下式的 logistic 函数，正负样本区分度越高，则下式的值越小：

$$\ln\left[1 + \exp\left(-\gamma \cdot \left(\text{sim}(u, q, d^+) - \text{sim}(u, q, d^-)\right)\right)\right].$$

上式中的 $\gamma > 0$ 是一个超参数，控制 logistic 函数的形状。

一个 batch 的样本包括一对 (u, q)，m 篇正样本文档 $d_1^+, \cdots, d_m^+$，n 篇负样本文档 $d_1^-, \cdots, d_n^-$。负样本是简单负样本与困难负样本的混合，它们的比例需要调。定义损失函数为

$$\frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n} \ln\left[1 + \exp\left(-\gamma \cdot \left(\text{sim}(u, q, d_i^+) - \text{sim}(u, q, d_j^-)\right)\right)\right].$$

计算损失函数关于双塔模型参数的梯度，做梯度下降更新模型参数。

17.2.2 batch 内负采样训练方法

一个 batch 中有 b 条正样本，记作 $\{(u_i, q_i, d_i)\}_{i=1}^{b}$。对于 batch 内任意的 $i \neq j$，将其中的 (u_i, q_i) 与 d_j 组合，得到 $\{(u_i, q_i, d_j)\}_{j \neq i}$ 是一条简单负样本。使用 batch 内负采样的训练方法，目标也是让双塔模型可以区分正负样本，鼓励正样本的 $\text{sim}(u_i, q_i, d_i)$ 尽量大，负样本的 $\text{sim}(u_i, q_i, d_j)$ 尽量小。

如图 17.1 所示，将 (u_i, q_i) 与 $d_1, \cdots, d_b$ 组合，其中有 1 条正样本，$b - 1$ 条负样本。用了 b 条样本，双塔模型计算出 b 个 sim 值。再把它们送入 softmax 函数，函数输出 b 个概率值 $p_{i,1}, \cdots, p_{i,b}$，其中 $p_{i,i}$ 对应正样本。使用交叉熵作为损失函数，最小化交叉熵，

会鼓励 $p_{i,i}$ 接近 1，其余 $b-1$ 个概率值接近零。

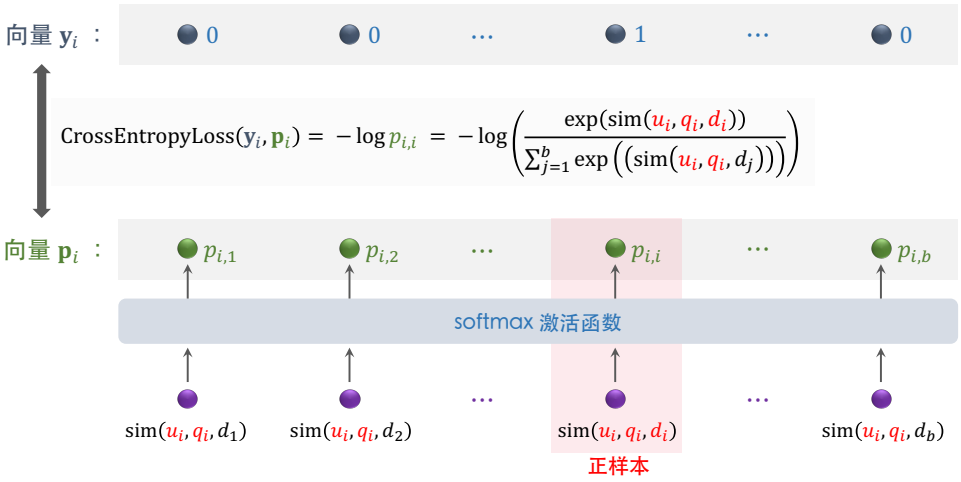

图 17.1: 用 batch 内负样本训练双塔模型

正样本 (u_i, q_i, d_i) 是有点击的样本，那么热门文档成为正样本的概率大。相应的，用 batch 内负样本，那么热门文档成为负样本的概率也大。热门文档成为正负样本的概率大，导致上述 batch 内负样本的训练方法存在偏差。下面介绍谷歌提出的消偏方法[20]。文档 d_j 成为正样本的概率 p_j 正比于它在搜索引擎中的总点击次数。做训练的时候，把图 17.1 交叉熵损失函数中的 $\text{sim}(u_i, q_i, d_j)$ 替换成 $\text{sim}(u_i, q_i, d_j) - \ln p_j$。一个 batch 中有 b 条正样本，对 b 个损失函数求平均得到：

$$-\frac{1}{b}\sum_{i=1}^{b}\ln\left(\frac{\exp\left(\text{sim}(u_i, q_i, d_i) \;-\; \ln p_i\right)}{\sum_{j=1}^{b}\exp\left(\text{sim}(u_i, q_i, d_j) \;-\; \ln p_j\right)}\right).$$

计算损失函数关于双塔模型参数的梯度，做梯度下降更新模型参数。注意，仅在训练的时候使用 p_j，在线上推理的时候不使用 p_j。

17.2.3 模型的更新

个性化向量召回模型既使用查询词和文档的文本，也使用文档和用户的 ID embedding、用户历史行为记录。文档和用户的向量表征从用户和文档的点击记录中学习，每天有新的点击行为发生，因此需要每天更新模型的参数、并重新计算物品的向量表征。最简单的模型训练方法是天级别的增量更新。

1. 在每天凌晨，用 Spark 批量处理搜索曝光日志，生成正样本、负样本，将样本打包成 TFRecord 或 Parquet 格式的文件。
2. 用打包好的样本训练模型。注意，模型不是随机初始化的，而是在昨天的模型参数的基础上，做增量训练。训练只做 1 epoch，即只用每条样本计算一次梯度。
3. 训练完成之后，用模型做离线推理，计算所有文档 d 的向量表征 $\boldsymbol{x}_d$，把 $\langle \boldsymbol{x}_d, d\rangle$ 存入向量数据库。此时向量数据库需要重建索引，需要数小时才能完成。
4. 将模型部署在线上，当用户 u 搜索查询词 q 时，计算 (u, q) 的向量表征 $\boldsymbol{x}_{u,q}$，用它在向量数据库中检索相似度最高的一批文档。

更先进的训练方法是做 online learning 实时更新模型参数，这种方法比天级别的更新有两点优势。第一，用户兴趣会随时发生变化，online learning 可以学到用户的兴趣点。第二，新发布的文档的点击次数很少，每一次点击都是宝贵的，可以帮助学习文档 ID embedding。总而言之，使用 online learning 势必会让个性好像向量召回更准确，但它的实时数据流（生成训练样本）、实时更新模型参数、实时更新向量数据库索引都需要很多计算资源、工程开发。对于推荐系统，个性化召回至关重要，用 online learning 是划算的；但是对于搜索引擎，相关性的重要性高于个性化，对个性化模型做 online learning 的投入产出比不高。

17.3 线上推理

将双塔模型用于召回、海选，线上推理的方法是不同的。海选从数万文档中选数千个，可以用模型逐一计算 (q, d) 相关性分数和 (u, q, d) 个性化分数。因此，可以事先计算文档 d 的向量表征 $\boldsymbol{z}_d$，以 $\langle d, \boldsymbol{z}_d \rangle$ 的形式存储在哈希表中，线上给定数万文档 ID，取回数万文档向量，送入模型给文档打分。召回需要从数亿文档中选择数万，不可能用模型逐一给数亿文档打分，那么该用什么方法避免枚举呢？

在训练好双塔模型之后，用模型计算全量文档 d 的向量表征 $\boldsymbol{z}_d$，以 $\langle \boldsymbol{z}_d, d \rangle$ 的形式存储在 Faiss 这样的向量数据库中。以相关性双塔模型为例，在线上用户输入 q，模型计算 q 的向量表征 $\boldsymbol{x}_q$，在向量数据库中以 $\boldsymbol{x}_q$ 检索相似度 $\mathrm{sim}(\boldsymbol{x}_q, \boldsymbol{z}_d)$ 最高的数百个 d，作为相关性向量召回的结果。需要从数亿篇文档中选出相似度 $\mathrm{sim}(\boldsymbol{x}_q, \boldsymbol{z}_d)$ 最高的一批 d，但是不能对于所有的 d 都计算 $\mathrm{sim}(\boldsymbol{x}_q, \boldsymbol{z}_d)$，那么向量数据库是如何做到的呢？

想要在检索时避免枚举，向量数据库需要事先对数亿篇文档的向量 $\boldsymbol{z}_d$ 建立索引，被统称为向量 ANN 索引。ANN 的全称是 approximate nearest neighbor，即近似最近邻。有很多种建立向量 ANN 索引的方法，下面以 IVFFLAT 为例讲解如何建立索引、并使用索引做召回。

1. 对数亿个向量 $\{\boldsymbol{z}_d\}$ 做聚类，得到 m 个聚类中心，记作 $\boldsymbol{c}_1, \cdots, \boldsymbol{c}_m$。记录每个向量 $\boldsymbol{z}_d$ 距离最近的聚类中心，建立从聚类中心到文档向量的索引。
2. 给定查询词的向量表征 $\boldsymbol{x}_q$，计算它与聚类中心的相似度 $\mathrm{sim}(\boldsymbol{x}_q, \boldsymbol{c}_1), \cdots, \mathrm{sim}(\boldsymbol{x}_q, \boldsymbol{c}_m)$，选出相似度最高的聚类中心。
3. 给定聚类中心，取回它对应的文档向量。逐一计算 $\boldsymbol{x}_q$ 与这些文档向量的相似度，选出相似度最高的数百篇文档作为召回结果。

假设数据库中一共有 n 个向量，把它们聚类成 $m = \sqrt{n}$ 个簇，如果聚类比较平衡，那么每个簇有 $\frac{n}{m} = \mathcal{O}(\sqrt{n})$ 个向量。如果不建索引，暴力检索需要 $\mathcal{O}(n)$ 的时间；使用索引，只需要 $\mathcal{O}(\sqrt{n})$ 的时间。

17.4 知识点小结

- 搜索引擎中的向量召回主要分为相关性和个性化两类，都使用双塔模型，它们有以下几点区别。第一，相关性召回拟合的目标是相关性分数，而个性化召回拟合的目标是点击行为。第二，相关性召回主要依据查询词和文档的文本特征，而个性化召回额外使用用户 ID、用户画像、用户行为序列、文档 ID 等特征。第三，相关性召回模型无需频繁更新，而个性化召回模型至少需要每天更新。
- 前面章节讨论过，精排、粗排分别使用 12 层、4 层交叉 BERT 模型，海选使用双塔 BERT 模型，它们的训练方法都是蒸馏 48 层交叉 BERT 大模型。训练相关性召回的双塔模型，方法也是蒸馏 48 层大模型；如果条件不允许，则蒸馏 12 层模型。训练召回和排序模型的唯一区别是数据，训练召回模型需要很大比例的负样本，而训练排序模型则让各相关性档位的样本相对均衡。
- 有两种方法训练个性化召回的双塔模型，一种是 pairwise 方法，另一种是 batch 内负采样，两者的目标都是让双塔模型有能力区分正样本和负样本。
- 召回和海选都使用上述两种双塔模型（相关性模型和个性化模型），模型结构完全相同，但是线上推理的方法完全不同。召回的候选文档数量 n 为数亿，需要离线建立向量 ANN 索引，线上推理时才能避免遍历所有文档，用 $o(n)$ 的时间完成召回。海选的候选文档数量 n 为数万，只需要把文档 ID 和向量表征保存在哈希表中，无需向量 ANN 索引，线上推理给所有文档打分，需要 $\mathcal{O}(n)$ 的时间。

第 18 章　离线召回

搜索引擎的查询词有头部效应，每天超过 50% 的搜索集中在一两百万条查询词上。设每篇文档 ID 为 96 位二进制码，每条查询词存储 1000 篇文档，一共存储头部 200 万条查询词，那么所需的存储仅为 25GB。如果召回对 RT 的限制宽松，比如允许耗时数十毫秒，那么可以用硬盘存储，几乎没有成本。我们可以用各种方法做离线计算，为每条查询词 q 找到一批相关性、时效性、内容质量综合分数很高的文档 $\{d\}$，以“$q \to \text{List}\langle d\rangle$”或类似的形式建立索引，作为召回通道。

基于上述想法，我们团队设计和实现了几种离线召回通道，A/B 测试取得了显著的收益，最终全量上线。18.1 节介绍挖掘曝光日志的方法，根据线上曝光、点击、交互行为，给每条查询词找到数百篇文档，构造出“$q \to \text{List}\langle d\rangle$”或类似的索引。18.2 节介绍离线搜索链路，在夜间利用空闲的集群做扩量的召回、排序，把离线排序得到的结果以“$q \to \text{List}\langle d\rangle$”的形式存储，作为一路召回通道。18.3 节介绍我们提出的“反向召回”方法，使用 NLP 模型离线寻找高相关性的 (q, d) 二元组，

18.1　挖掘曝光日志

搜索日志的每一条记录对应一个用户的请求 (u, q)，取搜索结果中排名最高的 k 篇文档 $d_1, \cdots, d_k$，日志中还记录文档是否被点击、交互。我们知道，如果用户搜 q，搜索引擎能把文档 d_i 排在前 k，那么 d_i 大概率与 q 相关，内容质量不错，且符合用户兴趣；用户点击、交互文档，则更能印证上述论点。搜索引擎精排产出的结果必然大幅优于普通召回通道的结果，把精排结果以 $q \to \text{List}\langle d\rangle$ 的形式建立索引，作为一条召回通道，理应远比普通的召回通道高效。我们将这条召回通道叫做“缓存召回”。

读者可能会有疑问：缓存召回的文档来源于其他召回通道，缓存召回会不会与其他召回通道有 100% 的重叠？如果是的话，则缓存召回通道不会提升任何业务指标。但我们实际的落地拿到了有点比和文档点击率的显著收益，比较合理的解释如下。搜索引擎的链路很长，包括二三十路召回、多个相关性和个性化打分模型、以及很多特征服务。每个环节都有一定概率超时，如果不用特殊的策略做保障，同一时刻用同样的 (u, q) 做搜索，两个结果页会有差异。尤其是在流量高峰时段，各环节的超时率都会增加，最终结果页会有较大的差异。缓存召回通道中记录的是非常优质的搜索结果，即便其他召回通道超时，这些优质结果也能被召回，对搜索的稳定性、业务指标都是有利的。

缓存召回可以是非个性化的，也可以是弱个性化的。非个性化的缓存召回用如下方法建立。

1. 搜索日志中的每条记录为 $(u, q) \to [d_{u,1}, \cdots, d_{u,k}]$，这里我们设置阈值 k，取排名最高的 k 篇文档。日志中记录 $d_{u,i}$ 是否被用户 u 点击、交互，并且记录 $d_{u,i}$ 的相关性和内容质量分数。
2. 对于每条查询词 q，如果有 $n = 1000$ 位用户 u 搜过，那么就有 $n = 1000$ 个文档列

表 $[d_{u,1}, \cdots, d_{u,k}]$。对这些文档做去重，得到 $q \to [d_1, \cdots, d_m]$（$m \geq k$），每篇文档至少出现 1 次，至多出现 $n = 1000$ 次。统计每篇文档出现次数、点击次数、交互次数，除以 n 得到频率，频率越高，说明 (q, d_i) 越匹配。

3. 得到 $q \to [d_1, \cdots, d_m]$ 之后，对每篇文档的出现频率、点击频率、交互频率、相关性分数、内容质量分数做融合，得到一个分数。此外，对文档年龄取对数，乘以一个权重，从文档总分中扣除；权重取决于 q 的时效性意图，时效性意图越强，则权重越大。
4. 上述步骤都是在 Hive 表中离线计算的，得到 $q \to [(d_1, s_1), \cdots, (d_m, s_m)]$，其中 s_i 是 d_i 的总分。Hive 表使用磁盘存储，容量非常大，因此无需限制 m 的增长，m 可以是数千、数万。但是线上使用的索引主要存储在内存中，需要做限制，比如缓存召回通道的配额是 200，那么线上索引就存分数 s_i 最高的 200 篇文档。当用户搜索 q 时，缓存召回通道查询线上的索引，返回 200 篇文档。

上述方法使用最近若干天的搜索日志得到 $q \to [(d_1, s_1), \cdots, (d_m, s_m)]$，存储在 Hive 表中。在此之后，需要做天级别的更新，处理当天的搜索日志得到 $q \to [(d'_1, s'_1), (d'_2, s'_2), \cdots]$。对两张表做归并，取新旧分数的加权和 $s_i \leftarrow (1 - \lambda)s_i + \lambda s'_i$，作为文档 d_i 新的分数。更新分数之后，重新做排序，取分数最高的 200 篇文档接入线上的索引。

弱个性化的缓存召回对用户做分组，比如按照性别、年龄分为 10 组，每位用户 u 被映射到用户组 g。那么索引为 $(q, g) \to [d_1, \cdots, d_m]$。建立和更新索引的方法与上面类似。根据我们的经验，用户组数量过多、过少都不利于业务指标。在我们的业务中，用户组数量在 10 左右，对点击率、有点比的提升最大。此外，索引使用的内存正比于用户组数量，因此用户组数量不能设置得过大。

在缓存召回全量上线之后，需要特别注意一个问题，如果忽视这个问题的话，未来其他的召回实验的收益都会有折损。举个例子，在线上有缓存召回的前提下，一个新的召回通道上线做 A/B 测试。新的召回通道能召回一些原本召回不了的高相关性文档，带来业务指标收益。原本这些文档只曝光给实验组用户，但是随着这些文档获得曝光、点击、交互，它们会被缓存召回收录，然后曝光给对照组用户。这样一来，A/B 测试不再准确，新的召回通道带了的业务指标收益有折损。为了避免这种问题，新的召回实验需要设置某种标签，缓存召回在处理搜索日志时，会排除掉带有标签的搜索结果。

18.2 离线搜索链路

上一节介绍的方法从曝光日志中挖掘出每条查询词的优质搜索结果，建立索引，作为缓存召回。作为一路补充召回，它可以提升业务指标；但是它不能取代线上的任何召回通道，因为它没有“活水”。本节介绍我们团队设计的另一种方案，它的开发难度大，但是可以部分取代线上的召回通道，节约很高比例的机器成本，还能提升业务指标。

我们的方案叫做“离线搜索链路”，它在夜间空闲时段调用线上的召回和排序，以 $q \to \text{List}\langle d, r\rangle$ 的形式建索引，作为一路召回通道。索引中的 r 是 d 的精排相关性分数，离线计算，线上不用重算。我们基于如下考量，提出了“离线搜索链路”。

- 前文说过，查询词有很强的头部效应，一两百万条查询词可以覆盖一半以上的搜索请求，一千万条查询词可以覆盖绝大部分的搜索请求。进入排序、最终获得曝光的文档中，大部分来自非个性化召回通道，例如文本召回、相关性向量召回。也就是说，对于给定的 q，不论 u 是谁，大部分的召回结果是相同的；然而用户每发起一次请求，都需要重复做这部分召回，并计算 (q, d) 的相关性。（注意，相关性是非个性化的，不依赖于 u。）如果能事先算好，则可以节省这部分召回和相关性的计算。
- 工业界通常以每秒查询次数（QPS）衡量系统负载，搜索引擎的 QPS 有很严重的峰谷效应。很多业务的 QPS 峰值出现在晚上下班到睡觉之间，整个系统的机器成本取决于峰值 QPS，要保障峰值出现时系统能稳定运行。夜间是 QPS 的低谷，在 2 到 6 点之间的 QPS 几乎降到零。如果在夜间调用线上的召回、排序链路做计算，几乎不会增加机器成本。
- 线上采用多级漏斗，粗排、精排打分的文档数量分别是数千、数百，打分量越大，则业务指标越好。但是出于成本、响应时间（RT）的考虑，线上链路的打分量不能随意增大。离线链路有所不同，使用空闲时段的机器没有额外成本，也没有 RT 的限制，打分量可以大幅扩大。

18.2.1 召回通道

用户在线上搜索查询词 q，如果 q 是头部查询词，则调用图 18.1 中的链路；如果不是头部查询词，则调用图 1.2 中的标准搜索引擎链路。如图 18.1 所示，对于头部查询词，直接从 KV 索引 $q \to \text{List}\langle d, r\rangle$ 中读取离线链路产出的文档 d 及其精排相关性分数 r。即便是头部查询词，也不能用离线链路完全取代线上链路，原因如下。

- 线上需要保留个性化召回通道，否则不利于用户体验，会显著降低有点比、文档点击率等指标。
- 线上需要保留新文档召回通道。如果离线链路需要 t 天能算完一轮头部查询词的搜索结果，那么最近 t 天新发布的新文档无法被离线链路召回。
- 离线链路的头部查询词库需要排除突发时效性、强时效性、地域性意图的查询词。

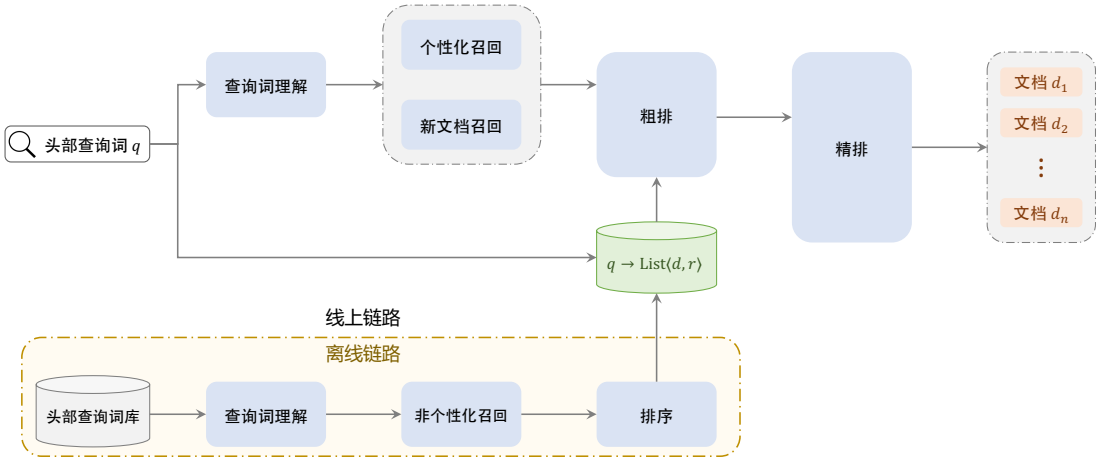

图 18.1: 头部查询词的搜索链路

离线链路可以提升搜索的业务指标，原因主要有两个，一方面是将相关性和点击率算得更准，另一方面是提升链路的稳定性。

- 离线链路的排序取代了线上链路的海选。对于相关性，离线链路使用精排相关性 BERT 模型，线上链路的海选使用双塔 BERT 模型，前者远比后者准确。对于点击率和交互率，离线链路使用非个性化的多目标深度神经网络，线上链路使用双塔点击率模型、甚至不使用点击率模型，前者远比后者准确。综上所述，用离线链路的排序代替线上链路的海选，排序会更准确。
- 离线链路的小部分业务指标收益来自于稳定性的提升。离线链路的结果以 KV 的形式做存储和检索，机器成本远低于文本召回、相关性模型、个性化模型。给 KV 索引多加一点机器成本，线上几乎不会出现超时和失败，稳定性远高于链路上的其他模块。线上链路任何模块的超时都会给业务指标造成或多或少的损失；如果大部分结果来自离线链路的 KV 索引，那么线上链路超时造成的损失会小很多。

18.2.2 降本方案

粗排和精排的相关性用 4 层或 12 的 BERT 模型，分别给数千、数百篇文档打分。文档的长度较长，给每个 (q, d) 二元组打分的代价都比较大。粗排和精排相关性的推理成本是搜索引擎所有模块中最高的，我们希望用离线链路降低线上的推理成本。

用图 18.1 中的方式使用离线链路，可以降低机器成本。离线链路的文档带有精排相关性分数，因此线上的粗排和精排无需重复计算相关性。头部数百万查询词可以覆盖 60% 以上的搜索请求，且每个搜索请求约有 50% 的文档来自非个性化召回通道，那么有 30% 的 (q, d) 相关性已经离线算好，节省了 30% 的线上相关性的请求。

用图 18.1 中的方式使用离线链路，只能事先计算出非个性化召回结果的相关性分数，降低成本的空间有限。如果能覆盖住个性化召回的文档，那么降本的收益可以翻倍。具体方法如下。

- 利用上一节介绍的缓存召回通道，每条查询词存 3000 到 6000 篇文档，可以覆盖个性化召回通道的大部分文档。夜间用离线链路计算这些文档的精排相关性分数 r，不做排序，把结果存储在 KV 索引 $q \to \text{List}\langle d, r\rangle$。
- 当用户搜索 q 时，从 KV 索引上读取列表 $\text{List}\langle d, r\rangle$。只需在粗排开始之前完成读取，对 RT 的限制较为宽松，因此允许 KV 索引建在硬盘上，花数十毫秒时间读取磁盘。判断等待粗排打分的文档是否命中 $\text{List}\langle d, r\rangle$，如果命中，则避免计算粗排和精排相关性。

18.3 反向召回

由于线上链路对 RT、成本的限制，线上召回用的是倒排索引、双塔这样的简单模型，而复杂的模型很难用于线上召回。线上的文本召回、向量召回可以很好地处理头部查询词，而复杂的离线挖掘方法可以提升中、尾部查询词的召回。我们可以在离线用各种方法挖掘高相关性的 (q, d) 二元组，用精排相关性模型打分 r，保留 r 高于某个阈值的 d，

以 $q \to \text{List}\langle d, r\rangle$ 的形式存储在 Hive 表中。根据相关性、内容质量、时效性等因子，在 Hive 表中做简单的规则排序，接入线上的 KV 引，作为召回通道。设每篇文档 ID 为 96 位二进制码，每条查询词存储 200 篇文档，哪怕存储一亿条查询词，所需的存储也就只有 2.5TB，成本开销并不大。

反向召回是我的团队提出和落地的技术。“反向召回”这个名字与“正向”相对，“正向”的意思是线上通用的根据 q 检索相关的 d。“反向召回”只是一种大致的思想，可以用不同的方法实现；但若想要取得召回的收益，必须满足如下条件：

- 给定 d，挖掘出的候选查询词 $\{q\}$ 应当大部分与 d 相关。否则，大部分 q 被相关性模型过滤掉，则效率低下，浪费大量计算资源。
- 挖掘出 (q, d) 二元组要与线上的文本召回、向量召回有足够大的差异。否则，会与线上召回通道的重合度过大，提供不了多少独一无二的召回结果。

我们使用一种叫做 doc2query（D2Q）的方法做反向召回，线上 A/B 测试表明 APP DAU、APP 留存都有很大的涨幅，点击率、有点比有小幅的提升。D2Q 的模型很简单，就是图 18.2 中所示的标准 Transformer 模型，将文档以字粒度输入编码器，解码器逐字生成查询词。训练 Transformer 这样的大模型需要海量的数据，人工标注的相关性数据是不够的。如何挖掘海量、优质的训练数据是提升效果的关键，重要性大于模型本身。获取数据、训练模型的大致流程如下。

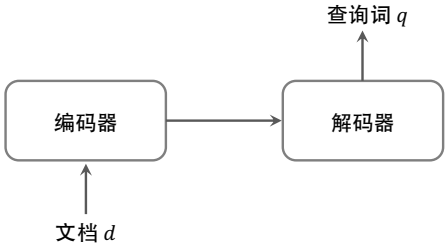

图 18.2: doc2query 使用的 Transformer 模型结构

1. 根据过去数十天的搜索日志，统计最高频的百万查询词。我们只用这样的头部查询词，不用中、尾部查询词，原因是这样的。大量用户都搜的查询词 q 是精炼、且有意义的，如果用户搜 q 且对结果 d 满意，则 q 相当于对 d 很好的概括，类似于文本摘要。
2. 挖掘搜索日志，对于每个头部查询词 q，获取排名高、点击率高、且被精排相关性模型判定为最高档位的文档 d，组成一条训练样本 (q, d)。不能只根据相关性模型预测的档位选文档，因为模型的预测未必精准。但是结合搜索排序高、点击率高这些信号，选出的文档 d 几乎都是高相关性的。
3. 从搜索日志中挖掘数亿对 (q, d) 二元组，以类似机器翻译的方式训练 Transformer 模型，相当于预训练。从人工标注的相关性数据中选出高相关性的 (q, d) 二元组，继续训练 Transformer 模型，相当于精调。预训练起到的作用最大，精调也会带来一些效果的提升。

在训练好模型之后，离线做推理，回刷文档库，为文档生成查询词。

- 过滤掉库中字数少于某个阈值的文档，这样的文档难以生成查询词。过滤掉库中内容质量档位很低的文档，这样的文档没有搜索价值。
- 我们希望一篇文档能生成尽量多的查询词。可以在逐字生成查询词的时候带有一定随机性，重复做一百次生成，可以得到数十个独一无二的查询词。beam search 是更

优的方法，但是计算量很大，我们最终没有采用这种方法。

为了离线评估 D2Q 模型，我们定了两个指标，指标越高越好。一个指标是相关性模型的通过率，举个例子，相关性模型给一亿对生成的 (q, d) 打分，有 6 千万对的分数高于某阈值，那么通过率就是 60%。另一个指标是高相关性查询词生成量，举个例子，平均每篇文档随机做 100 次生成，得到 30 条独一无二的查询词，其中 18 条为高相关性，那么生成量为 18。

图 18.3 用两个真实的例子展示 D2Q 生成的查询词。我们看了大量的实例，发现 D2Q 有两方面非常强大的能力。第一，D2Q 起到生成式文本摘要的作用，对文档做极简的概括，且生成的查询词未必是文档中的原词。第二，D2Q 可以从答案生成极简的问题，这是抽取式的方法无法做到的。

文档	查询词
标题：爸爸衬衫穿搭 衬衫对男士们的重要性大家都很清楚，选择一款合适的衬衫不仅可以让爸爸们展示出成熟男性魅力，而且穿着体验感也很好哦。 纯棉的面料，柔软舒适，贴身穿着也毫无不适感。宽松的版型不会束缚住爸爸们的啤酒肚，经典的格子设计能够更好的衬托出中年男士成熟的魅力，不管是单穿还是内搭，都是很不错的选择。 下半身随意的搭配一条休闲裤，就能满足爸爸们的日常出行了，不管是外出游玩还是上班办公，都是不错的选择。	• 男士衬衫中年 • 中年男士衬衫 • 男士衬衫 • 爸爸衬衫
标题：葱香花卷 小时候的味道 一次发酵 简单快手 今天来做经典好吃的花卷，记得小时候妈妈经常做花卷，刚出锅热腾腾软软的空口都能吃好几个，葱香满满，还有淡淡的五香味儿，当然陪上辣酱或者就着菜吃更是美味，今天这款一次发酵花卷简单需要时间短，也很容易操作蒸出来不容易瘪，次次轻松成功，让你随时随地就可以吃上健康美味的花卷！ 食材： 600g 中筋面粉，320-360g 温水，不同面粉吸水量不同。15g 白糖 1/2tbsp，6g 酵母 1 tsp，葱，油，盐，五香粉或者花椒粉也可以。 醒发30-40分钟左右，然后冷水上锅蒸12分钟关火。 喜欢我的视频记得点赞和关注！谢谢大家观看，我会继续和大家分享健康和简单的各种美食！	• 怎样做花卷 • 花卷怎么做 • 花卷怎么蒸 • 怎么做花卷好吃 • 花卷的做法 • 花卷的制作方法 • 蒸花卷

图 18.3: 用两个真实的例子展示 doc2query 生成的查询词

我们提出的反向召回并不限于 D2Q，它只是我们尝试的第一种方法，且取得了核心业务指标的收益，证明离线挖掘 (q, d) 二元组是有效的。还有很多可能有效的方案给文档寻找匹配的查询词，我们尚未尝试，就不在此处列举了。

18.4 结合查询词改写与缓存召回

我们在 15.3 节讨论过基于相关性的查询词改写方法。对于任意一条查询词 q，将其改写成头部查询词，存储在索引 $q \to [q'_1, \cdots, q'_k]$。由于查询词较短，即便存储一亿条查询词 q 的改写，索引所需的存储空间也不大。线上绝大多数的请求 q 可以命中索引，如果不能命中索引，则线上做改写召回和改写判别。

实际用于召回时，我们结合上述改写方法与缓存召回通道。如图 18.4 所示，对于任意一条查询词 q，将其改写成头部查询词 $q'_1, \cdots, q'_k$。然后用改写后的查询词从缓存召回通道 $q' \to \text{List}\langle d \rangle$ 中取回若干篇文档，作为召回结果。可以把改写和召回记作 $q \to q' \to d$，

缓存召回通道确保 (q', d) 具有高相关性，那么根据基于相关性的改写的性质，(q, d) 很可能具有高相关性。此外，缓存召回通道中排名靠前的文档的内容质量、时效性都比较好，所以图 18.4 中的召回方法的相关性、内容质量、时效性都比较好。

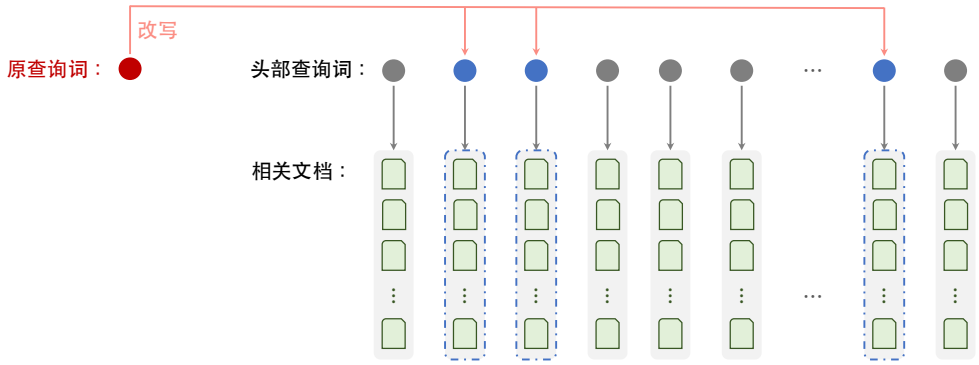

图 18.4: 结合查询词改写与缓存召回，作为一路召回通道

18.5 知识点小结

- 对于头部查询词 q，记录它的搜索结果页上排名靠前的文档，根据文档的曝光、点击、交互、相关性、内容质量、时效性做离线排序，构建 KV 索引 $q \to \text{List}\langle d \rangle$。把索引作为缓存召回通道，只对头部查询词生效。
- 缓存召回通道本身可以提升业务指标，还有其他两个应用。一个应用是在离线搜索链路，事先计算出相关性分数，降低线上相关性模型的计算量，降低成本。另一个应用是与查询词改写结合，增加相关且优质的文档召回量，提升业务指标。
- 离线搜索链路在夜间集群空闲时主动发起搜索，做非个性化的召回和排序，取排名高的文档及其相关性分数，构建 KV 索引 $q \to \text{List}\langle d, r \rangle$，取代线上的非个性化召回通道。离线搜索链路既可以提升业务指标，也可以降低成本。
- 可以用反向召回这样的技术离线挖掘高相关性的 (q, d) 二元组，构建 KV 索引 $q \to \text{List}\langle d \rangle$，作为召回通道，提升业务指标。

第六部分

排序

第 19 章 排序的基本原理

如图 19.1 所示，搜索引擎的链路上有三处需要做排序，它们分别是召回截断、粗排、精排。精排的打分量较小，仅有数百，允许在单篇文档上花费很大的计算量。精排给每篇文档打一个分数，这个分数基本决定了文档的最终曝光顺序。粗排的打分量为数千，在单篇文档上花费的计算量必须远小于精排。粗排起到承上启下的作用，从数千候选文档中选出数百，尽量保证用户最想看的文档不会被过滤掉。召回截断的打分量为数万，在线上只能用向量内积这样的简单运算给数万文档打分，从中选出数千送入粗排。

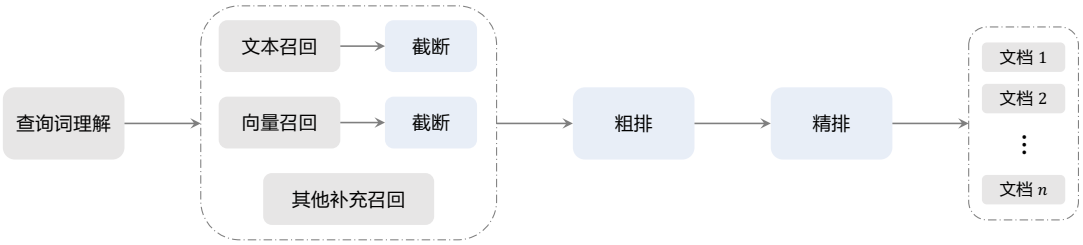

图 19.1: 搜索引擎的链路上有三处需要做排序

精排、粗排、召回截断的原理基本相同。如图 19.2 所示，三种排序都是先用模型预测相关性、点击率，再用模型融合相关性、内容质量、时效性、点击率、其他特征。融合模型输出一个分数，按照这个分数对文档做排序。三处排序的区别在于相关性模型、点击率模型、融合模型的规模与使用的特征。三处排序使用的内容质量和时效性特征是完全相同的。

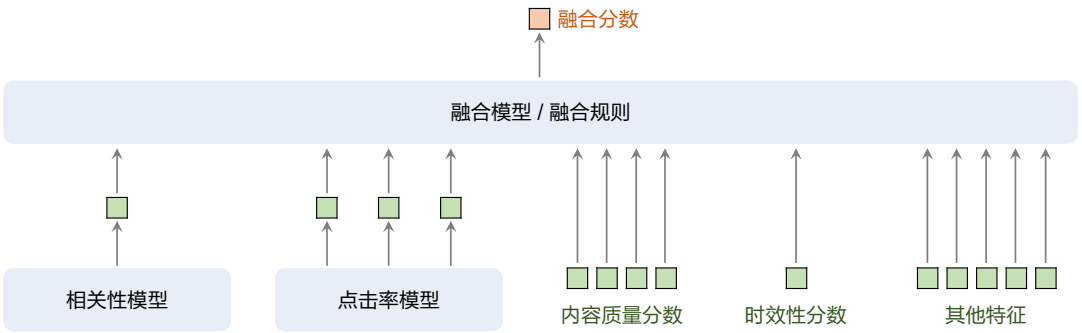

图 19.2: 排序在线上使用 3 个模型做推理，分别是相关性模型、点击率模型、融合模型

19.1 融合模型的特征

融合模型（或融合规则）用到的特征包括相关性、点击率、内容质量、时效性、以及其他多种特征。其中相关性和点击率是由模型在排序模块计算的，而其他特征在离线或链路上游计算好，传输给排序模块。

19.1.1 相关性模型

精排、粗排、召回截断都需要用模型计算 (q, d) 的相关性，它们的基本思路相同。如图 xx 所示，它们都是用 BERT 这样的语义模型计算相关性分数，再用线性模型或 GBDT 融合 BERT 模型打分和其他特征，得到最终的相关性分数。融合模型算出的相关性分数比 BERT 模型有小幅优势，但融合模型的所有特征中，BERT 模型的打分的权重远大于其他特征。

精排和粗排都使用交叉 BERT 模型，即把“[CLS] + 查询词 + [SEP] + 文档 + [SEP]”作为输入，在 [CLS] 的位置上输出相关性分数。如果算力充足，粗排和精排分别用 4 层和 12 层交叉 BERT 模型。如果算力不足，可以只在粗排用 4 层交叉 BERT 模型，精排沿用粗排的相关性分数，不再另外计算相关性。如果算力非常有限，可以在精排用 4 层交叉 BERT 模型，在粗排用双塔 BERT 模型。

召回截断可以使用双塔 BERT 模型，左塔计算查询词的向量表征，右塔计算文档的向量表征，用向量相似度预测相关性。离线事先用右塔计算文档的向量表征，存储在哈希表中。在线上，左塔需要做推理，计算查询词的向量表征；右塔不需要推理，直接从哈希表中读取文档的向量表征即可。如果算力不足或工程基建水平不高，召回阶段可以不用双塔 BERT 模型，只用 BM25 等特征计算相关性。

19.1.2 点击率模型

精排、粗排、召回截断都需要预估点击率和交互率。精排的点击率模型为多目标深度神经网络。它的输入包括查询词特征、文档特征、用户特征、场景特征、以及多种统计特征。模型使用超大规模的嵌入层，数亿用户、数亿文档的 ID 都被 embedding 层表征为 128 或 256 维向量。之所以称之为多目标模型，是因为一个模型同时预估点击率与多种交互率（点赞率、收藏率、转发率、关注率、评论率、完播率），所有这些预估值都作为融合模型的特征。

粗排的点击率模型为图 10.9 中的多目标双塔模型，左塔计算请求（即用户 + 查询词）的向量表征，右塔计算文档的向量表征，用向量相似度预估点击率和交互率。离线事先用右塔计算文档的向量表征，存储在哈希表中。在线上，只需要用左塔做推理，而右塔不需要推理。

召回截断的点击率模型为双塔模型，与粗排类似，但是有两点区别。第一，截断的点击率模型可以不用个性化，或者只用性别这样的弱个性化特征，而粗排使用所有的用户特征。第二，截断可以不对交互率做预估，原因是交互发生在点击之后，这些信号过于稀疏，难以准确预估。如果算力不足或工程基建水平不高，召回截断可以不预估点击率，或者只使用统计特征和其他数值特征，用线性模型融合。

19.1.3 其它特征

除了相关性和点击率，精排、粗排、召回截断还使用下面列出的特征（或部分特征）。其中查询词意图是链路上游的 QP 模块传递给排序模块的，其它特征是在离线计算、线

上读取特征服务。

- 查询词意图，包括查询词类目、时效性意图、地域性意图。
- 用户画像（用户性别、年龄等特征）与场景特征（时间和用户所在地点）。
- 文档特征，包括文档内容质量、文档类型（图文/视频/商品）、文档年龄、文档所属类目。
- 文档统计特征包括文档的曝光次数、点击次数、交互次数、点击率、交互率等统计值。这些后验特征从一个侧面反映出文档质量的高低。
- 查询词—文档统计特征包括点击率等统计值。这里的点击率是指“查询词 q 下文档 d 的点击次数”除以“查询词 q 下文档 d 的曝光次数”。这样的后验特征从一个侧面反映出查询词与文档的相关性。

19.2 融合规则 vs 融合模型

在搜索引擎建设的早期阶段，最好是用融合规则，而不是用融合模型。融合规则主要是根据相关性做分档，档内用人工设定的规则融合点击率、内容质量、时效性等因子，根据融合分数档内做排序。融合规则的好处是简单、可解释、容易调。举个例子，如果发现首图和标题骗点击的文档排名高，那么就可以降低点击率的权重，提升点赞率、收藏率、内容质量分数的权重。再举个例子，召回通道、排序模型升级之后，视频曝光占比显著下跌，可以调整融合规则把视频曝光占比调高。

在搜索引擎的建设发展到一定阶段，可以尝试融合模型代替融合规则。精排和粗排使用树模型，召回截断使用线性模型。融合模型把相关性、点击率、内容质量、时效性等分数作为特征，做 pairwise 或 listwise 训练，拟合某种目标。使用融合模型代替融合规则之后，可以将相关性分档弱化，例如从 4 档松弛到 2 档，根据融合分数在 2 档内做排序。如果不用融合模型，那么相关性分档的弱化会损害相关性，反映在人工体验评估指标变差。我们在松弛相关性分档的同时，用融合模型代替融合规则，人工体验评估指标是持平的，相关性没有变差。由于相关性分档的弱化，点击率、有点比、首点位置等多种点击指标都有巨大的提升，带动留存指标大幅增长。

在松弛相关性分档之后，必须要注意一个问题。如果把线上的点击指标当做优化目标，那么排序团队可以通过弱化相关性的权重来提升点击指标。为了避免这种情况的出现，应当在实验平台上加入对相关性的监控，确保新策略不会损害相关性。取搜索结果页上排名前 k 的文档，统计各档位文档的占比，例如高、中、低、无的占比分别是 50、30、15、5%，将每日的指标显示在监控看板上。监控的相关性分数是模型估算的、而非人工标注的，不是非常准确，且会受相关性模型迭代升级的影响。但这种监控是有意义的，如果排序团队故意降低相关性权重，那么监控看板上的指标肯定发生变化，高、中档位的占比降低，低、无档位的占比升高。

19.3 融合模型训练数据

本节讨论融合模型的训练数据。训练融合模型需要用到人工标注的综合满意度分数、用户的点击和交互行为，两者结合作为训练的目标。由于人工标注的综合满意度规模有限，实践中会训练一个教师模型，用它代替人工做标注，可以无限扩充训练数据的规模。

19.3.1 模型拟合的目标

对于通用搜索引擎，排序的目标是让用户对搜索结果满意。对于电商搜索引擎，排序的目标既包括满意，也包括营收。本书重点讨论通用搜索引擎，以用户满意作为排序的目标。工业界通常从人工标注、用户行为两方面定义“满意”。

人工标注的“综合满意度”只考虑查询词、文档、场景，不考虑用户喜好，因此这是一个客观评价。标注员根据查询词、文档、场景（包括搜索发生的时间、地点），判断文档能在多大程度上满足用户的需求，给出综合的档位。综合满意度评估会考虑相关性、内容质量、时效性等多种因素，但不是对各种分数求简单的加权和。举个例子，对于“糖尿病饮食”、“银行理财产品靠谱吗”这类 your life or your money 查询词，相关性与 EAT 并重。对于“三亚旅游攻略”、“海参怎么泡发”这类寻找攻略的查询词，在满足相关性的前提下，重点看文本质量。对于带时效性意图的查询词，需要根据时效性需求的强弱，结合文档内容是否过时，判断文档能在多大程度上满足用户需求。

用户行为是对满意度的后验，用户的点击、交互行为相当于用户投票，判断文档符合需求。用户行为带有用户的个性化，用户基于自己的兴趣决定是否点击和交互。标注员难以根据用户画像准确推断用户的兴趣点，因此用户行为是对综合满意度的补充。但是用户行为充满随机性，有点击的文档未必有高相关性、高内容质量。

训练融合模型时，该以什么作为模型拟合的目标呢？给定查询词和文档，人工标注综合满意度 s。此外，搜索日志记录用户是否点击和交互过文档。我们用规则结合综合满意度和用户行为，得到分数 y，作为融合模型拟合的目标。一种规则是加权和：

$$y = s + \alpha \times \text{点击} + \beta \times \text{交互}, \tag{19.1}$$

上式中的 α 和 β 是权重。另一种融合方法是优先点击，有点击则 y 为综合满意度与交互的加权和，无点击则 $y = 0$。

$$y = \begin{cases} s + \beta \times \text{交互}, & \text{有点击;} \\ 0, & \text{未点击 } d_i. \end{cases} \tag{19.2}$$

19.3.2 数据的扩充

训练融合模型依赖人工标注的综合满意度分数。由于标注人力的限制，只能标注数万条查询词，数十万组 (q, d)；在项目前期，数据量还要小一个量级。如果搜索引擎是非个性化的，这样的样本量或许足够训练融合模型。但如果搜索引擎有很强的个性化，排序需要做到千人千面，那么融合模型需要更大的数据量。解决方法是做离线数据挖掘，并用人工标注的综合满意度分数训练一个小规模的模型（比如 GBDT），我们称之为“教师

模型”。在训练好教师模型之后，我们用它给海量的 (q,d) 二元组打分，用于训练排序融合模型。这种生成训练数据的方法类似于相关性模型的后预训练。

我们从搜索日志中选出一批 (q,d) 二元组，抽取下面列出的特征，并由人工标注综合满意度分数，用于训练教师模型。由于教师模型是非个性化的，特征中可以有搜索发生的场景（时间、地点），但是不能有用户个性化特征。

- 查询词特征，包括查询词类目、时效性意图。
- 文档特征，包括文档类型（图文/视频/商品）、文档年龄（取搜索发生时文档的年龄）、文档类目、EAT、以及多种文本质量和图片质量分数。
- 相关性特征，包括精排相关性模型的最终打分，还包括词匹配分数、相关性 BERT 打分等中间分数。
- 文档统计特征，包括曝光量、点击量、交互量、点击率、交互率，这些统计值与文档的内容质量相关。
- 查询词—文档统计特征，即 (q,d) 的曝光量、点击量、点击率，这些用户行为后验与 (q,d) 的相关性、d 的内容质量相关。

对离散特征做 one-hot 编码，再对某些特征做交叉（比如交叉查询词类目和文档类目），得到 (q,d) 的特征向量，记作 $\boldsymbol{x}_{q,d}$。把人工标注的综合满意度档位归一化为介于 0 和 1 之间的分数，记作 $s_{q,d}$。因为训练教师模型所用的样本为人工标注，样本数量较小，所以模型的参数量不能过大，不宜使用大规模的嵌入层，离散特征只需做简单的 one-hot 编码。

把教师模型记作 $t(\boldsymbol{x}_{q,d};\boldsymbol{\theta})$，基座可以采用 GBDT（参数记作 $\boldsymbol{\theta}$），输出层用 sigmoid 激活函数。教师模型的目标是准确估计综合满意度 $s_{q,d}$，因此采用 pointwise 训练方法：

$$\min_{\boldsymbol{\theta}} \sum_{(q,d)} s_{q,d} \cdot \ln\big[t(\boldsymbol{x}_{q,d};\boldsymbol{\theta})\big] + (1-s_{q,d}) \cdot \ln\big[1-t(\boldsymbol{x}_{q,d};\boldsymbol{\theta})\big].$$

这样训练出的教师模型可以根据非个性化特征估计综合满意度分数。

训练好教师模型之后，可以自动生成大规模的数据，用于训练排序融合模型。挖掘搜索日志，对于每次搜索请求 (u,q)，取回一批文档 $d_1,\cdots,d_k$，以及非个性化特征、个性化特征、用户的点击和交互行为。我们把 (u,q,d_i) 的非个性化特征记作向量 $\boldsymbol{x}_i$，个性化特征记作向量 $\boldsymbol{z}_i$。用教师模型估计综合满意度分数 $\widehat{s}_i = t(\boldsymbol{x}_i;\boldsymbol{\theta})$，再根据式 (19.1) 或 (19.2)，结合 $\widehat{s}_i$ 和点击、交互行为，得到综合分数 y_i。把融合模型记作 $f(\boldsymbol{x}_i,\boldsymbol{z}_i;\boldsymbol{w})$，模型使用的特征比教师模型更多。我们让融合模型拟合 y_i，学习模型参数 $\boldsymbol{w}$；训练融合模型的具体方法将在下一节中详细讲解。

19.4 知识点小结

- 搜索引擎的链路上有三处需要对候选文档做排序，分别是精排、粗排、召回截断，三处的打分量分别是数百、数千、数万。
- 精排、粗排、召回截断的原理基本相同，都是先由相关性模型、点击率模型打分，把相关性、点击率、交互率、内容质量、时效性等多个分数作为特征，由融合模型（或融合规则）给出最终的一个分数，作为排序的依据。

- 在搜索引擎建设的前期可以使用融合规则，而非融合模型。首先根据相关性对候选文档分档，比如分为 4 档。再用人工设置的公式对点击率、交互率、内容质量、时效性等分数做融合，比如取加权和，在档内做排序。
- 使用融合模型代替融合规则，可以大幅提升业务指标，且不会显著损害相关性。使用融合模型的坏处是会干扰搜索算法的迭代。
- 想要训练融合模型，需要用人工标注综合满意度分数。结合综合满意度和用户行为，得到融合模型拟合的目标。为了扩大训练数据的规模，可以事先训练一个教师模型，代替人工标注综合满意度分数。

第 20 章 训练融合模型的方法

搜索引擎的精排、粗排、召回截断各需要用到一个融合模型，可以是树模型，也可以是线性模型。将融合模型记作 $f(\boldsymbol{x}, \boldsymbol{z}; \boldsymbol{w})$，其中 $\boldsymbol{x}$ 是非个性化特征，$\boldsymbol{z}$ 是个性化特征，$\boldsymbol{w}$ 是需要学习的参数。本章讲解三种训练融合模型的方法，分别是 pointwise、pairwise、listwise。设每个 batch 的数据包括用户 u、查询词 q、文档 $d_1, \cdots, d_k$、以及结合教师模型打分和用户行为算出的 $y_1, \cdots, y_k$。

20.1 pointwise 训练方法

pointwise 训练方法把 (u, q, d_i) 当成独立的样本，不考虑样本之间的关系，让融合模型打分 $f(\boldsymbol{x}_i, \boldsymbol{z}_i; \boldsymbol{w})$ 拟合 y_i。做 pointwise 训练可以采用均方误差损失或交叉熵损失。不论 y_i 的取值范围是多少，都可以用均方误差损失：

$$\frac{1}{k}\sum_{i=1}^{k}\Big[f(\boldsymbol{x}_i, \boldsymbol{z}_i; \boldsymbol{w}) - y_i\Big]^2.$$

如果 y_i 被归一化到 $[0, 1]$ 区间上，那么可以用交叉熵损失：

$$\frac{1}{k}\sum_{i=1}^{k} y_i \cdot \ln\Big[f(\boldsymbol{x}_i, \boldsymbol{z}_i; \boldsymbol{w})\Big] + (1-y_i)\cdot \ln\Big[1 - f(\boldsymbol{x}_i, \boldsymbol{z}_i; \boldsymbol{w})\Big].$$

不论用哪种，训练时都是取损失函数关于 $\boldsymbol{w}$ 的梯度，做梯度下降更新 $\boldsymbol{w}$，让 $f(\boldsymbol{x}_i, \boldsymbol{z}_i; \boldsymbol{w})$ 更好地拟合 y_i。

从上述训练方式中不难看出，pointwise 的本质是回归，起到“保值”的作用，即尽量让 $f(\boldsymbol{x}_i, \boldsymbol{z}_i; \boldsymbol{w})$ 对 y_i 的估计尽可能准确。但是在排序中，“值”本身不重要，重要的是“序”。举个例子，令 δ 为任意实数，我们用 $f + \delta$ 代替 f 作为排序融合模型，这会导致 pointwise 打分变得非常不准，但是对排序毫无影响，原因在于所有文档分数都同等增加了 δ。综上所述，“保值”的 pointwise 不是最优的训练方法。

虽然 pointwise 不能针对排序的特性去优化模型，但是在项目上线的初期最好是用 pointwise 方法作为基线。pointwise 训练容易实现，不容易出错，并不会比 pairwise、listwise 方法差太多，因此适合作为第一版模型上线。有了 pointwise 训练出的模型作为线上 baseline，可以尝试 pairwise、listwise 训练方法，观察线上的业务指标收益。

20.2 pairwise 训练方法

从此处开始，我们把融合模型给 (u, q, d_i) 打的分数记作 $p_i = f(\boldsymbol{x}_i, \boldsymbol{z}_i; \boldsymbol{w})$。设 $y_i > y_j$，即文档 d_i 理应排在 d_j 前面。如果 $p_i \geq p_j$，那么文档 (d_i, d_j) 是一个正序对；否则，如果 $p_i < p_j$，那么 (d_i, d_j) 是一个逆序对。pairwise 训练方法的目标是最大化正逆序比。

下面介绍 RankNet[2] 这种 pairwise 方法，它的基本思想是这样的。如果 $p_i \geq p_j$，则 RankNet 鼓励 $p_i - p_j$ 尽量大；如果 $y_i < y_j$，则鼓励 $p_i - p_j$ 尽量小。通过学习函数 f，可

以使所有的 (p_i, p_j) 的序趋于与 (y_i, y_j) 一致。RankNet 定义损失函数：

$$c = \frac{1}{n^2} \sum_{(i,j):y_i>y_j} \underbrace{\ln\big[1+\exp\big(-(p_i-p_j)\big)\big]}_{\triangleq c_{ij}}. \tag{20.1}$$

让损失函数 c 下降，会使得所有 (p_i, p_j) 的序趋于与 (y_i, y_j) 一致。

有了损失函数，剩下的任务就是优化 $f(\boldsymbol{x};\boldsymbol{w})$，使得损失函数最小化。对损失函数 c_{ij} 关于模型参数 $\boldsymbol{w}$ 求梯度：

$$\frac{\partial c_{ij}}{\partial \boldsymbol{w}} = \frac{\partial c_{ij}}{\partial p_i}\frac{\partial p_i}{\partial \boldsymbol{w}} + \frac{\partial c_{ij}}{\partial p_j}\frac{\partial p_j}{\partial \boldsymbol{w}}. \tag{20.2}$$

设 $\lambda_{ij} = \frac{\partial c_{ij}}{\partial p_i}$。不难证明，$\lambda_{ij}$ 满足：

$$\lambda_{ij} = \frac{\partial c_{ij}}{\partial p_i} = -\frac{\partial c_{ij}}{\partial p_j} = -\frac{1}{1+\exp(-(p_i-p_j))}. \tag{20.3}$$

由式 (20.2) 可得

$$\frac{\partial c_{ij}}{\partial \boldsymbol{w}} = \lambda_{ij}\left(\frac{\partial p_i}{\partial \boldsymbol{w}} - \frac{\partial p_j}{\partial \boldsymbol{w}}\right). \tag{20.4}$$

对总损失函数 $c = \frac{1}{n^2}\sum_{(i,j):y_i>y_j} c_{ij}$ 关于 $\boldsymbol{w}$ 求梯度，可得

$$\begin{aligned}
\frac{\partial c}{\partial \boldsymbol{w}} &= \frac{1}{n^2}\sum_{(i,j):y_i>y_j} \lambda_{ij}\cdot\left(\frac{\partial p_i}{\partial \boldsymbol{w}} - \frac{\partial p_j}{\partial \boldsymbol{w}}\right) \\
&= \left(\frac{1}{n^2}\sum_{(i,j):y_i>y_j} \lambda_{ij}\cdot\frac{\partial p_i}{\partial \boldsymbol{w}}\right) - \left(\frac{1}{n^2}\sum_{(i,k):y_i<y_k} \lambda_{ik}\cdot\frac{\partial p_i}{\partial \boldsymbol{w}}\right) \\
&= \frac{1}{n^2}\sum_{i=1}^{n}\left[\left(\sum_{j:y_i>y_j}\lambda_{ij}\right) - \left(\sum_{k:y_i<y_k}\lambda_{ik}\right)\right]\cdot\frac{\partial p_i}{\partial \boldsymbol{w}}.
\end{aligned} \tag{20.5}$$

我们可以用梯度更新 $\boldsymbol{w}$，比如做梯度下降：

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta\cdot\frac{\partial c}{\partial \boldsymbol{w}}.$$

上式中的 η 为学习率。用这种方式更新排序的打分函数 $p_i = f(\boldsymbol{x}_i, \boldsymbol{z}_i; \boldsymbol{w})$，会鼓励 (p_i, p_j) 的序与 (y_i, y_j) 的序趋于一致，因为这样可以最小化损失函数 c。

RankNet 的优化目标是增大正序对数量、减小逆序对数量。但 RankNet 并不完全契合搜索排序的目标，下面我们来分析 RankNet 存在的问题。根据损失函数 c_{ij} 的定义，c_{ij} 只取决于 y_i、y_j、p_i、p_j 这四者。如果 $y_i > y_j$，那么 RankNet 鼓励 $p_i - p_j$ 增大。这种定义忽略了两个关键的信息。

- 损失函数的定义没有考虑 y_i 与 y_j 的相对大小。根据 c_{ij} 的定义，不论 y_i 略大于 y_j、还是 y_i 远大于 y_j，都不影响 c_{ij} 的大小。
- 损失函数的定义没有考虑文档 d_i 与 d_j 的排名。根据 NDCG 的定义，排名靠前的文档对 NDCG 的影响大，应当重点保障排名靠前的文档排名的准确性。此外，文档 i 和 j 排名差得越远，损失应当越大。

20.3 listwise 训练方法

LambdaRank[1] 是一种 listwise 训练方法，可以解决 RankNet 存在的问题。LambdaRank 的想法非常简单，就是对损失函数 c_{ij} 做加权，权重是交换文档 i 和 j 造成的 NDCG 之差。DCG 的定义为：

$$\text{DCG@}n \;=\; \sum_{i=1}^{n} \frac{2^{y_i}-1}{\log_2(i+1)}.$$

IDCG 是 DCG 的上界，IDCG 只取决于真实分数 $y_1,\cdots,y_n$。对 $y_1,\cdots,y_n$ 从大到小排序，记作 $\tilde{y}_1,\cdots,\tilde{y}_n$。那么

$$\text{IDCG@}n \;=\; \sum_{i=1}^{n} \frac{2^{\tilde{y}_i}-1}{\log_2(i+1)}.$$

NDCG 定义为 DCG 与 IDCG 的比值，即

$$\text{NDCG@}n \;=\; \frac{1}{\text{IDCG@}n}\sum_{i=1}^{n} \frac{2^{y_i}-1}{\log_2(i+1)}.$$

交换文档 i 与 j 的位置，会影响上式连加中的两项，交换前后 NDCG 之差为

$$\begin{aligned}\Delta_{ij} \;&=\; \frac{1}{\text{IDCG@}n}\cdot\left(\frac{2^{y_i}-1}{\log_2(i+1)}+\frac{2^{y_j}-1}{\log_2(j+1)}-\frac{2^{y_i}-1}{\log_2(j+1)}-\frac{2^{y_j}-1}{\log_2(i+1)}\right)\\ &=\; \frac{1}{\text{IDCG@}n}\cdot\left(2^{y_i}-2^{y_j}\right)\cdot\left(\frac{1}{\log_2(i+1)}-\frac{1}{\log_2(j+1)}\right).\end{aligned}$$

用 $|\Delta_{ij}|$ 对损失函数 c_{ij} 做加权，那么式 (20.1) 就变成

$$c \;=\; \frac{1}{n^2}\sum_{(i,j):y_i>y_j} |\Delta_{ij}|\cdot\ln\left[1+\exp\left(-(s_i-s_j)\right)\right]. \tag{20.6}$$

梯度公式 (20.5) 变成

$$\frac{\partial c}{\partial \boldsymbol{w}} \;=\; \frac{1}{n^2}\sum_{i=1}^{n}\left[\left(\sum_{j:y_i>y_j}\lambda_{ij}\cdot|\Delta_{ij}|\right)-\left(\sum_{k:y_i<y_k}\lambda_{ik}\cdot|\Delta_{ik}|\right)\right]\cdot\frac{\partial s_i}{\partial \boldsymbol{w}}.$$

上式中的 λ_{ij} 在式 (20.3) 中定义。

20.4 知识点小结

- pointwise 训练方法让融合模型 $f(\boldsymbol{x},\boldsymbol{z};\boldsymbol{w})$ 拟合目标 y，而不考虑样本之间的序。pointwise 训练方法简单有效，可以作为初期的基线方法。
- pairwise 训练方法考虑样本之间的序，鼓励正序对尽量多、逆序对尽量少。如果以正逆序比作为评价指标，那么 pairwise 优于其他两种训练方法。
- listwise 训练方法不但考虑样本之间的序，还考虑文档的排名，避免逆序发生在排名靠前的位置。如果以 NDCG 作为评价指标，那么 listwise 优于其他两种训练方法。
- 对排序感兴趣的读者，可以用百度在 2022 年公开的数据[24] 做实验。

第七部分

查询词推荐

第 21 章 查询词推荐的场景

查询词推荐是搜索的一环，但是与搜索链路相对独立，几乎可以自行闭环。前面章节讲的搜索链路决定在用户搜查询词 q 之后搜索结果页面出什么文档，而查询词推荐在很大程度上影响用户搜什么 q。查询词推荐对搜索引擎的影响很重，大部分的搜索请求来自于推荐的查询词、而非用户完整输入的查询词。

查询词推荐主要有四种业务场景。第一，在搜索前根据用户的兴趣做个性化推词，起到激发搜索的作用，让用户从不想搜变成想搜。第二，在搜索中和搜索后做查询建议，实时补全和纠正用户输入的查询词，让搜索变得更方便。第三，在搜索后根据用户输入的查询词建议相关的查询词，增加搜索次数，并且拓宽用户的兴趣。第四，在用户阅读文档时建议相关的查询词，起到搜索后推词同样的作用。

21.1 搜索前推词

在小红书、抖音、快手等 APP 都有搜索功能，点击放大镜样式的搜索按钮，即可打开搜索框。如图 21.1 所示，在用户尚未输入查询词时，搜索框里面、搜索框下方已经有很多候选查询词，包括底纹词、猜你想搜、热榜。用户点击这些查询词，就相当于搜索这些词，会跳转到搜索结果页。

图 21.1: 搜前个性化推词

底纹词、猜你想搜属于个性化推词，根据用户历史上搜过的查询词、点击过的文档，召回与之相关的查询词，对查询词做排序之后推荐给用户。底纹词、猜你想搜是由相同的链路同时产生的，模型打分最高的作为底纹词，其余作为猜你想搜。热榜是非个性化的，所有的用户看到的热榜都相同。热榜中的查询词是统计出来的，但其中可能有运营人员的干预。

搜前的推词起到搜索激发作用，用户原本不想搜 q，但是被推荐的 q 引起兴趣，点击了 q，产生了搜索行为。如果推荐的查询词符合用户兴趣，那么用户点击查询词的概率大，因此可以用推荐的查询词的点击率衡量推词的效果。一些用户本来没想好搜什么，只是随便看看有趣的话题，如果对推荐的查询词感兴趣，就会产生原本不会发生的搜索，会提升搜索日活用户数（Search DAU）与日活用户数（DAU）的比值。因此可以用 $\frac{\text{Search DAU}}{\text{DAU}}$ 作为搜前推词的业务指标。此外，推词还需要为转化负责，即推荐的 q 触发搜索之后，搜索结果页上的点击和交互指标；所有查询词推荐场景都需要为转化指标负责，这部分内容留到 21.5 节讨论。

21.2 查询建议

查询建议发生在搜索中、搜索后。如图 21.2 所示，用户在输入查询词的时候，根据用户已经输入的字，在搜索框下方给出查询建议。这种搜索中的查询建议叫做 suggestion（SUG）。SUG 起到工具的作用，用户输入的字更少、花费的时间更短就能找到想搜的查询词，搜索效率和体验有提升。SUG 占据的流量很大，通过点击 SUG 发起的搜索次数，往往超过用户完整输入后发起的搜索次数。

图 21.2: 左图是搜中的查询建议，中、右图是搜后的纠错建议

SUG 的评价指标包括使用率、点击位置、输入长度、转化指标。转化指标是所有查询词推荐场景都需要负责的指标，留到 21.5 节讨论。

- SUG 使用率（有点比）等于 SUG 点击次数除以搜索次数。使用率高，说明 SUG 精准，找到了用户想搜的查询词。SUG 使用率通常超过 50%，即大多数的搜索来源于点击 SUG、而非用户自己完整输入。
- SUG 中有多条查询词，用户最多只能点击一条，点击位置越靠前，说明 SUG 的排序做得越好。SUG 的点击位置类似于搜索结果页上的首点位置，都是效率指标。
- 搜索框中的输入长度越短越好，越短说明用户使用越方便，不需要自己手动输入完整的查询词。

图 21.2（中、右）是搜后的纠错建议。如果算法对纠错的判断不置信，出原词 q 的搜索结果，并建议用户搜纠错后的词 q'。如果算法对纠错的判断置信，则出 q' 的搜索结果，但仍然询问用户是不是想搜原词 q。纠错占的流量很小，但仍然是有用的，如果不做纠错，搜索结果页很难满足用户需求，转化率偏低。

21.3 搜索结果页推词

如图 21.3 所示，在用户搜索 q 之后，搜索引擎返回的文档 $\{d\}$ 按顺序显示在搜索结果页上，中间会插入若干查询词 $\{q'\}$，这些查询词叫做“相关搜索”。相关搜索的“相关”，是指推荐的 q' 与原查询词 q 相关。比方说搜 $q =$“日料”，可以推荐 $q' =$“上海自助餐”、“寿司之神”，但不适宜推荐 $q' =$“灌篮高手结局”。通过分析搜索日志，可以挖掘出查询词之间的关联，相关搜索带有个性化，根据时间、地点、用户画像、用户历史记录对候选查询词做排序。相关搜索非常类似于搜索引擎，区别在于不是用 q 找 d，而是用 q 找 q'。

图 21.3: 搜索后推词

相关搜索的评价指标有两个：消费深度、用户兴趣宽度。消费深度指的是用户搜索的查询词数量（不重复计数），如果推荐的查询词是用户想搜的，则被点击的概率大，会给搜索带来更多的流量（曝光文档数、点击次数）。用户的兴趣宽度用查询词的类目来衡量，我们希望推荐的查询词可以加宽用户的兴趣。

21.4 文档内推词

如图 21.4 所示，当用户阅读文档 d 时，在文档下方推荐一条查询词 q，这种推词场景也叫“相关搜索”。此处的相关是指 q 与 d 的相关性。根据 d 选 q，需要两者相关，但不需要高度相关性，推荐的目标并不是找出 d 最相关的 q，而是继续深挖和扩展。举个例子，用户在阅读一篇介绍寿司类型的文档 d，查询词 $q =$“寿司种类”是最相关的查询词，但未必是最合适的推荐词；$q =$“上海寿司店”、“寿司的做法”、“寿司醋怎么调”与 d 的相关性差一些，但这些词才是能深挖和扩展用户兴趣的。搜索前推词、搜索结果页推词、文档内推词都有很强的个性化，需要类似于推荐系统的排序模型。

图 21.4: 文档内推词

如果推荐的查询词 q 符合用户兴趣，用户会点击查询词，进入搜索结果页。可以用 q 的点击率、搜索结果页上的有点比作为文档推词的业务指标。对于小红书、抖音、快手这样的 APP，文档的点击主要来源于推荐，而非搜索。在文档内推词，如果获得点击，那么用户会产生搜索行为，提升 $\frac{\text{Search DAU}}{\text{DAU}}$ 比值。

21.5 评价指标总结

前面介绍的搜前推词、SUG、搜索结果页推词、文档内推词是几大主要的推词场景，它们的评价指标有很多共通之处。图 21.5 中列出了有点比、转化率、点击位置、Search DAU 这 4 种评价指标，适用于部分或全部推词场景。

	猜你想搜	SUG	搜索结果页	文档内
有点比	✓	✓	✓	✓
点击位置		✓		
转化率	✓	✓	✓	✓
搜索渗透率	✓			✓

图 21.5: 主要推词场景的评价指标

有点比衡量推荐的查询词是否符合用户兴趣、是否对用户有用。以搜前推词为例，每次列出 6 条查询词，只要用户点击其中一个，那么这次推荐就算有点击。如果今天一共做了 1 千万次推荐，每次推 6 个词，有 1 百万次推荐产生了点击，那么有点比就等于 1百万/1千万 $=10\%$。所有推词场景都可以用有点比作为评价指标，推荐质量高低都能直接反映在有点比上。

点击位置是一个效率指标，考察是否把用户最需要的结果排在最前面。虽然搜前推词、SUG、搜索结果页推词都可以记录点击位置，但只有 SUG 考察点击位置。SUG 是一个工具，目的是让用户能更方便地做搜索，把用户最想要的查询词排在前面可以为用户提供方便。搜前推词、搜索结果页推词都不是工具，只是让用户“逛一逛”。SUG 类似于搜索引擎，有工具属性；而其他两种推词类似于推荐系统，没有工具属性。

推词的场景——搜前推词、SUG、搜索结果页推词、文档内推词——都是搜索的起点，如果发生点击查询词 q，会把用户带到 q 的搜索结果页。有点比、点击位置具有局限性，它们不能评估搜索引擎对 q 的承接。举个例子，推荐 $q=$“教你领取苹果免费赠送的 iPhone”，它很吸引眼球，有点比很高，但是搜索引擎根本找不到这样的文档，用户看到搜索结果页会很失望地离开。从推词到用户点击查询词，这个过程并不完整，完整的转化流程是

$$\text{推词页面曝光查询词} \rightarrow \text{点击查询词} \rightarrow \text{搜索结果页曝光文档} \rightarrow \text{点击文档}。$$

推词不止需要考察推词页面的有点比和点击位置，还需要考察搜索结果页上的转化。因此需要用搜索结果页上的有点比、首屏有点比、文档点击率、首点位置等指标考核推词的转化。

推词和搜索的链路是相对独立的，推词的链路只能控制给用户曝光什么查询词，无法控制搜索结果页曝光什么文档。既然推词无法控制搜索结果页上的曝光文档，那么推词如何能控制转化率呢？给定查询词 q，根据 q 的文本、以及历史上搜索 q 的统计值（比

如搜索结果页的有点比），可以推断出转化率（比如搜索结果页的有点比）。举个例子，如果在小红书上搜 q =“男士冬季卫衣”，搜索结果页的有点比等指标非常高；如果搜 q =“男士瘦高冬季加厚卫衣穿搭建议”，搜索结果页的有点比也还不错；但如果搜同样长度的 q =“拉布拉多感受到职场的黑暗”，有点比非常低，原因是搜生僻查询词没有优质文档供给。其实推词是可以控制转化率的，想要搞转化率，那么就要尽量不推生僻、相关文档少、历史统计有点比低的查询词。

搜索渗透率是搜索引擎的核心业务指标之一，它等于 Search DAU 除以 APP DAU，即今天登录 APP 的用户中有多少人使用搜索功能。搜前推词、文档内推词如果做得好，会产生搜索行为，提升 Search DAU 和搜索渗透率。但是 SUG、搜索结果页推词很难提升搜索渗透率，原因是用户已经在做搜索、或已经搜完。

第 22 章 查询词推荐的召回

上一章重点讨论了 4 种推词场景——搜前推词、SUG、搜索结果页推词、文档内推词。本章讨论推词的召回方法，其中很多召回方法是多种推词场景共用的。图 22.1 列出 4 种推词场景的召回依据。搜前推词是完全基于个性化的推荐，召回的依据是用户的兴趣点。SUG、搜索结果页推词的场景为用户正在输入、或完成输入的查询词 q，q 是召回的依据。文档内推词的场景是当前的文档 d，d 是召回的依据。

	猜你想搜	SUG	搜索结果页	文档内
用户兴趣	✓			
当前查询词		✓	✓	
当前文档				✓

图 22.1: 主要推词场景做召回的依据

词库包括精品词库、站内热门 1 千万查询词、外网抓取的查询词。精品词库要符合小红书的调性，比如从优质笔记的标题和首图中提取文字。在线上做推词时，从词库中做召回。

22.1 SUG 召回

SUG 是推词场景中使用最频繁的，用户在输入查询词的过程中，SUG 根据已经输入的片段实时做召回和排序。SUG 主要用下面 3 种基于拼写的召回，这些都是非常成熟的技术。

- **前缀召回**，即根据用户已经输入的字，补全剩余的字。**通用前缀召回**是最最基础的召回通道，比如“上海 → 上海迪士尼”。**拼音前缀召回**有些类似，利用尚未输全的拼音字母补全查询词，比如“美 j → 美甲”。
- **分词召回**，即对用户已输入的查询词做分词，利用分词结果补全查询词。**通用分词召回**，即分词结果具有包含关系，比如“迪士尼 → 上海迪士尼”。设用户的输入片段被切分为 $\mathcal{Q} = \{w_1, \cdots, w_k\}$，候选查询词的分词结果记作 $\mathcal{Q}'$，两者具有包含关系 $\mathcal{Q} \subset \mathcal{Q}'$ 时，$\mathcal{Q}'$ 可以作为 SUG 召回。在我们的例子中，用户输入 $\mathcal{Q} = \{$迪士尼$\}$，SUG 召回 $\mathcal{Q}' = \{$上海, 迪士尼$\}$。**首字召回**，比如“上迪士尼 → 上海迪士尼”。设用户的输入查询词被切分为 $\mathcal{Q} = \{w_1, \cdots, w_k\}$，建议的查询词的分词结果记作 $\mathcal{Q}' = \{w'_1, \cdots, w'_k\}$，对于所有的 $i \in [k]$，$w_i = w'_i$，或者 w_i 为 w'_i 的首字。**核心词召回**，即在召回的数量低于某个阈值时触发，只保留核心词，然后做通用分词召回，比如“如何制作蛋挞 → 蛋挞做法”，其中“蛋挞”是核心词，其余词被丢弃。
- **拼写召回**，即根据输入法，召回合理的查询词。**拼音简写召回**，比如“shdsn → 上海

迪士尼”。**同音字召回**，比如“喜皮士 → 喜啤士、西皮士、嬉皮士”，仅在召回的数量低于某个阈值时触发。

除了上述三种召回通道，还有生成式召回，比如“日语慢速教学 → 日语慢速教学视频”。这种方法首先丢弃前缀，保留“教学”，以此生成“教学楼”、“教学视频”。这种方法需要用 bigram 关联矩阵校验扩展词的关联性，比如判断“日语慢速”与“教学楼”、“教学视频”的关联性，选取关联性高的扩展词。

22.2 用查询词召回查询词（Q2Q）

本节介绍几种用查询词召回查询词的方法，它们统称为 query2query（Q2Q），用于除 SUG 外的各种推词场景。22.2.1 和 22.2.2 节讲解两种协同过滤方法，它们分别叫做物品协同过滤（item collaborative filtering，ItemCF）和 Swing，它们的区别在于计算查询词相似度的公式。这两种协同过滤方法都需要建立 $q \to \text{List}\langle q' \rangle$ 这样的索引，在线上用一个查询词 q 召回一批查询词 q'。22.2.3 节简要介绍向量召回，用到我们熟悉的双塔模型，把查询词表征为向量，检索向量相似度高的查询词。

22.2.1 ItemCF

ItemCF 利用全体用户的行为判断两个查询词的相似度。如图 22.2 所示，如果有多名用户同时搜索 q_1 和 q_2，则说明 q_1 与 q_2 相似。举个例子，搜索“梅西”的用户，很有可能也搜索“阿根廷国家队”、“巴萨”、“巴黎圣日耳曼”。虽然这些查询词的文本、向量相关性都不高，但它们的兴趣点是一致的，如果一位用户搜“梅西”，推荐“阿根廷国家队”是合理的。

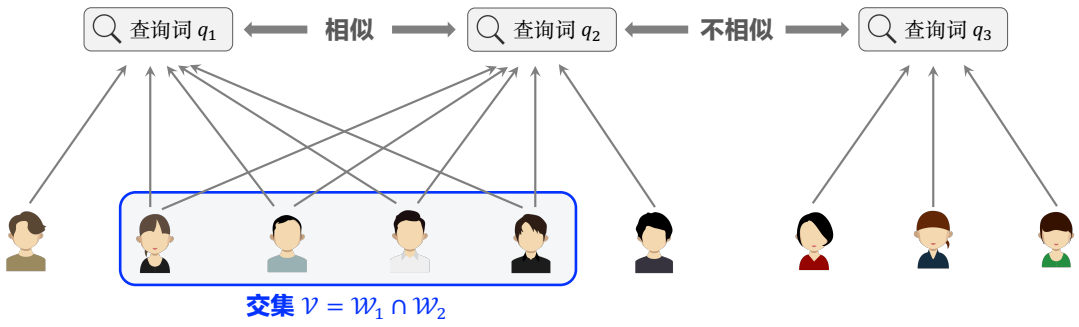

图 22.2: 在 ItemCF 看来，什么样的查询词相似或者不相似？

具体这样计算两个查询词的相似度分数。搜过查询词 q_1 和 q_2 的用户分别记作集合 $\mathcal{W}_1$ 和 $\mathcal{W}_2$。定义交集 $\mathcal{V} = \mathcal{W}_1 \cap \mathcal{W}_2$，集合 $\mathcal{V}$ 中的用户既搜过 q_1 也搜过 q_2。用下式计算两个查询词的相似度：

$$\text{sim}(q_1, q_2) \;=\; \frac{|\mathcal{V}|}{\sqrt{|\mathcal{W}_1| \cdot |\mathcal{W}_2|}}.$$

上式算出的相似度分数介于 0 和 1 之间，原因是 $|\mathcal{V}| \le |\mathcal{W}_1|$，$|\mathcal{V}| \le |\mathcal{W}_2|$。

22.2.2 Swing

Swing 是阿里巴巴 2020 年的论文[19] 提出的，它是 ItemCF 的一种变体，它的原理与 ItemCF 基本一致，但是计算相似度 $\text{sim}(q_1, q_2)$ 的公式有所区别。ItemCF 和 Swing 都根据图 22.2 中的用户交集 $\mathcal{V}$ 的大小计算 q_1 与 q_2 的相似度。ItemCF 只考虑 $\mathcal{V}$ 的大小 $|\mathcal{V}|$，而不考虑 $\mathcal{V}$ 中的用户是一个小圈子、或是广泛的群体。举个例子，一个微信群里每天发薅羊毛的资讯，让群友去某平台搜相应的查询词，于是几百个群友今天搜 q_1，明天搜 q_2 ……按照 ItemCF 的公式，查询词 q_1 和 q_2 有很高的相似度，但其实两个词没有联系。

出于以上考虑，Swing 考察 $\mathcal{V}$ 中用户兴趣的广泛程度，$\mathcal{V}$ 中的用户数量多，且用户两两不相似，则判定 q_1 与 q_2 的相似度高。把用户 u_1 和 u_2 搜过的查询词分别记作集合 $\mathcal{J}_1$ 和 $\mathcal{J}_2$。定义两个用户的重合度为

$$\text{overlap}(u_1, u_2) \ = \ \big|\mathcal{J}_1 \cap \mathcal{J}_2\big|.$$

重合度越高，则两个用户越有可能来自同一个小圈子，应当降低两位用户的权重。Swing 用下式计算两个查询词的相似度：

$$\text{sim}(q_1, q_2) \ = \ \sum_{u_1 \in \mathcal{V}} \sum_{u_2 \in \mathcal{V}} \frac{1}{\alpha + \text{overlap}(u_1, u_2)}. \tag{22.1}$$

上式中的 α（≥ 0）是个需要调的超参数。式中的 $\text{overlap}(u_1, u_2)$ 是两位用户的重合度，重合度越高，则两位用户对物品相似度的贡献越小，如此可以解决小圈子带来的问题。假如忽略用户的重合度，把式 (22.1) 连加中的 $\frac{1}{\alpha+\text{overlap}(u_1,u_2)}$ 换成 1，那么 (22.1) 就变成 $\text{sim}(q_1, q_2) = |\mathcal{V}|^2$，与 ItemCF 计算相似度的公式类似。

22.2.3 向量召回

可以用图 22.3 所示的双塔模型，把查询词的文本、类目、核心词输入神经网络，得到查询词的向量表征。用两个向量表征的相似度衡量查询词的相似度，根据相似度做向量召回，跟前面章节介绍的向量召回方法一致。

训练双塔模型的样本来自搜索结果页推词。用户搜索 q，触发搜索结果页，其中有若干条推荐查询词，用户点击其中一条查询词 q'，那么 (q, q') 是一对正样本。可以用各种方法构造负样本，比如全局随机抽样、batch 内负样本、困难负样本。训练双塔模型的方法是比较成熟的，没有多大改进空间，能提升推荐效果的方法主要是正负样本的构造。

22.3 用文档召回查询词（D2Q）

本节介绍几种用文档召回查询词的方法，它们统称为 doc2query（D2Q），用于搜前推词、文档内推词。各种 D2Q 的方法在文档发布时寻找相关的查询词，将结果以 $d \to \text{List}\langle q \rangle$ 的形式保存。

18.3 节介绍了一种生成式召回的方法，把文档的文本输入 Transformer 模型，模型随机生成若干条查询词，并用相关性 BERT 模型做判别，过滤掉相关性分数低于阈值的。Transformer 模型的需要的算力较大，适用于离线或近线计算。近线的意思是在文档发布

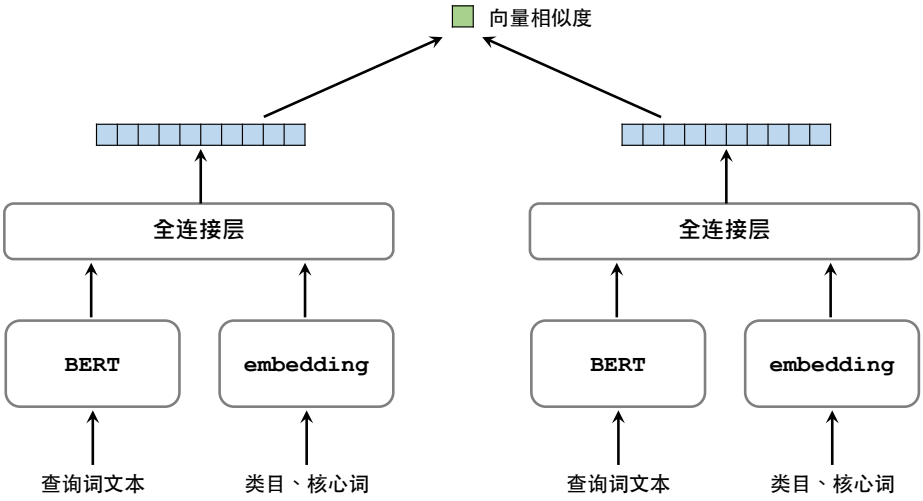

图 22.3: Q2Q 的向量召回

之后，不是实时触发模型做计算，而是进入消息队列等待，模型依次处理队列中的请求，近线计算会有几分钟的延迟。近线计算好处是 QPS 更均衡，比实时在线计算需要的算力更小。

D2Q 的向量召回可以复用相关性双塔模型，无需单独训练模型。只不过这里不是用查询词检索文档，而是用文档检索查询词。做完向量召回之后，还需要用相关性 BERT 模型做判别，排除掉相关性分数低于某个阈值的查询词。

上述方法利用文本相关性给文档召回查询词。此外，还可以根据用户行为做召回，即使用 ItemCF 计算文档与查询词的相似度。把点击过文档 d 的用户记作集合 $\mathcal{U}$，把搜过查询词 q 的用户记作集合 $\mathcal{W}$。把先点击文档 d 之后搜索查询词 q 的用户记作集合 $\mathcal{V}$。此处需要限定时间窗口 t 和查询词序列长度 l，比如点击文档之后 $t = 2$ 天内搜索的前 $l = 3$ 条查询词，这样的查询词可能是受文档激发的。用下式计算文档与查询词的相似度：

$$\text{sim}(d \to q) \;=\; \frac{|\mathcal{V}|}{\sqrt{|\mathcal{U}| \cdot |\mathcal{W}|}}. \tag{22.2}$$

上一节讲解过 ItemCF，此处就不在赘述。

22.4 各推词场景的召回方法

前面几节内容介绍了 SUG、Q2Q、D2Q 召回方法，本节讨论如何将这些方法用于各推词场景。SUG 主要起到补全的作用，它用独有的 SUG 召回方法，与其他推词场景都不同。下面讨论其他 3 种推词场景的召回方法。

搜前推词完全基于用户的兴趣点做召回。下面介绍 U2Q2Q、U2D2Q，它们把用户历史行为记录中的查询词、文档作为种子，用 Q2Q、D2Q 召回查询词。

- U2Q2Q 的意思是“用户 $\to$ 查询词 $\to$ 查询词”的召回方式。系统用索引 $u \to \text{List}\langle q \rangle$ 记录用户搜索过的查询词。使用最近的 n_1 条查询词，并从剩余的查询词中随机抽

取 n_2 条，凑够 n 条不重复的查询词，作为种子。对于每个种子 q，用 Q2Q 召回 m 条相关查询词，那么最多召回 mn 条查询词。

- U2D2Q 的意思是“用户 $\rightarrow$ 文档 $\rightarrow$ 查询词”的召回方式。系统用索引 $u \rightarrow \text{List}\langle d \rangle$ 记录用户交互过的文档。使用最近的 n_1 篇文档，并从剩余的文档中随机抽取 n_2 篇，凑够 n 篇不重复的文档，作为种子。对于每个种子 d，用 D2Q 召回 m 条相关查询词，那么最多召回 mn 条查询词。

搜索结果页推词基于当前的查询词做召回。用户搜 q，跳转到搜索结果页，其中插入一些推荐的查询词。以 q 作为种子，使用 Q2Q 召回一批相关查询词即可。

文档内推词基于用户当前阅读的文档 d 做召回，使用 D2Q 与 D2Q2Q 两种方法。D2Q 把 d 作为种子，直接召回 n 条相关的查询词 $q_1, \cdots, q_n$。D2Q2Q 再多跳转一次，即把 D2Q 召回的查询词 q_i 作为种子，用 Q2Q 召回 m 条查询词 $q'_{i,1}, \cdots, q'_{i,m}$。把 $\{q_i\} \cup \{q'_{i,j}\}$ 作为召回的结果。

第 23 章 查询词推荐的排序

23.1 预估点击和转化

23.2 多样性

参考文献

[1] C. Burges, R. Ragno, Q. Le. Learning to rank with nonsmooth cost functions. Advances in Neural Information Processing Systems (NIPS), 2006

[2] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, G. Hullender. Learning to rank using gradient descent. International Conference on Machine Learning (ICML). 2005 89–96

[3] S. Büttcher, C. L. Clarke, B. Lushman. Term proximity scoring for ad-hoc retrieval on very large text collections. International ACM SIGIR Conference on Research and Development in Information Retrieval. 2006 621–622

[4] P. Covington, J. Adams, E. Sargin. Deep neural networks for YouTube recommendations. ACM Conference on Recommender Systems. 2016 191–198

[5] Z. Dai, X. Wang, P. Ni, Y. Li, G. Li, X. Bai. Named entity recognition using BERT BiLSTM CRF for Chinese electronic health records. 2019 12th international congress on image and signal processing, biomedical engineering and informatics (cisp-bmei). IEEE, 2019 1–5

[6] S. Gopal, Y. Yang. Recursive regularization for large-scale classification with hierarchical and graphical dependencies. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD). 2013 257–265

[7] T. Huang, Z. Zhang, J. Zhang. FiBiNET: combining feature importance and bilinear feature interaction for click-through rate prediction. ACM Conference on Recommender Systems. 2019 169–177

[8] Z. Huang, W. Xu, K. Yu. Bidirectional LSTM-CRF models for sequence tagging. arXiv preprint arXiv:150801991, 2015

[9] T.-Y. Lin, P. Goyal, R. Girshick, K. He, P. Dollár. Focal loss for dense object detection. IEEE International Conference on Computer Vision (ICCV). 2017 2980–2988

[10] L. Liu, F. Mu, P. Li, X. Mu, J. Tang, X. Ai, R. Fu, L. Wang, X. Zhou. NeuralClassifier: an open-source neural hierarchical multi-label text classification toolkit. Annual Meeting of the Association for Computational Linguistics: System Demonstrations. 2019 87–92

[11] W. Liu, X. Fu, Y. Zhang, W. Xiao. Lexicon enhanced chinese sequence labeling using bert adapter. Annual Meeting of the Association for Computational Linguistics (ACL). 2021

[12] J. Ma, Z. Zhao, X. Yi, J. Chen, L. Hong, E. H. Chi. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2018 1930–1939

[13] Q. Pi, G. Zhou, Y. Zhang, Z. Wang, L. Ren, Y. Fan, X. Zhu, K. Gai. Search-based user interest modeling with lifelong sequential behavior data for click-through rate prediction. ACM International Conference on Information & Knowledge Management. 2020 2685–2692

[14] Y. Rasolofo, J. Savoy. Term proximity scoring for keyword-based retrieval systems. European Conference on Information Retrieval. Springer, 2003 207–218

[15] R. Wang, B. Fu, G. Fu, M. Wang. Deep & cross network for Ad click predictions. Proceedings of the ADKDD, 1–7. 2017

[16] R. Wang, R. Shivanna, D. Cheng, S. Jain, D. Lin, L. Hong, E. Chi. DCN v2: Improved deep & cross network and practical lessons for web-scale learning to rank systems. Proceedings of the ACM Web Conference (WWW). 2021 1785–1797

[17] R. Xiao, J. Ji, B. Cui, H. Tang, W. Ou, Y. Xiao, J. Tan, X. Ju. Weakly supervised co-training of query rewriting and semantic matching for e-commerce. ACM International Conference on Web Search and Data Mining (WSDM). 2019 402–410

[18] L. Xu, Z. Jie, W. Lu, L. Bing. Better feature integration for named entity recognition. Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. 2021 3457–3469

[19] X. Yang, Y. Zhu, Y. Zhang, X. Wang, Q. Yuan. Large scale product graph construction for recommendation in e-commerce. arXiv:201005525, 2020

[20] X. Yi, J. Yang, L. Hong, D. Z. Cheng, L. Heldt, A. Kumthekar, Z. Zhao, L. Wei, E. Chi. Sampling-bias-corrected neural modeling for large corpus item recommendations. ACM Conference on Recommender Systems. 2019 269–277

[21] G. Zhou, X. Zhu, C. Song, Y. Fan, H. Zhu, X. Ma, Y. Yan, J. Jin, H. Li, K. Gai. Deep interest network for click-through rate prediction. Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining. 2018 1059–1068

[22] J. Zobel, A. Moffat. Inverted files for text search engines. ACM computing surveys (CSUR), 2006. 38(2): 6–es

[23] L. Zou, W. Lu, Y. Liu, H. Cai, X. Chu, D. Ma, D. Shi, Y. Sun, Z. Cheng, S. Gu, et al. Pre-trained language model-based retrieval and ranking for web search. ACM Transactions on the Web, 2022. 17(1): 1–36

[24] L. Zou, H. Mao, X. Chu, J. Tang, W. Ye, S. Wang, D. Yin. A large scale search dataset for unbiased learning to rank. arXiv preprint arXiv:220703051, 2022

[25] L. Zou, S. Zhang, H. Cai, D. Ma, S. Cheng, S. Wang, D. Shi, Z. Cheng, D. Yin. Pre-trained language model based ranking in baidu search. Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining. 2021 4014–4022

[26] 张俊林. 这就是搜索引擎. 电子工业出版社, 2012