R&S[22] | 搜索系统中的召回

原创 机智的叉烧 CS的陋室 2020-01-12

光

OYRH;Austin Carl - 光

往期回顾：

- NLP.TM[24] | TextCNN的个人理解
- NLP.TM[23] | NLP学习线路推荐
- NLP.TM[22] | 如何修正NLP问题的bad case
- R&S[18] | SIGIR2018：深度学习匹配在搜索与推荐中的应用
- R&S[17] | 手把手搞推荐[6]：回顾整体建模过程

前几天浅梦前辈讨论了有关搜索推荐系统的召回，此处结合我的个人经验给大家分享一下，搜索系统中的召回方法吧。内容可能都是以规则和词典模式为主，大家别嫌弃我low了。浅梦前辈的文章如下：

- 搜索推荐中的召回匹配模型综述(一)--传统方法
- 搜索推荐中的召回匹配模型综述(二)--基于表示学习的深度学习方法

对召回的理解

现在无论是搜索系统还是推荐系统，基本形成了召回+排序的基本结构（当然内部还存在大量过滤逻辑，把一些召回回来的低质东西扔掉），召回的目标在于从海量信息中抽取若干可能可以被展示的信息，而排序则主要负责把更好的内容展示给用户，换言之，正确完整的结构是召回阶段关注召回率，而排序阶段关注准确率，两者结合，最终实现高准确高召回的展示结果。

重申，此时召回的目标在于从海量数据中找到可能可以出的结果，这种情况下，要求的是能找到的东西能尽可能找到，因为召回找不到的东西，是不可能在后面的结果里面出现了。

但值得强调的是，很多系统，尤其是最初刚开始建立的系统，排序其实并不是很完善，而应该是召回，甚至在召回阶段可能就要承担一定的排序压力，此时就需要保证较高程度的准确性了，因此情况还是会有所不同。

召回的操作

对于搜索系统，由于用户有非常明确的用户需求，因此所有操作都应该围绕着这句短短的用户query，但是要理解它的含义，却非常困难的，核心难度还是在于他的短，有的时候会非常模糊。因此和推荐系统不同的是，并非把时间花在用户和ITEM之间的分析上，而是query各个层面的分析。

query的各种分析，其核心原因是，要方便后续在数据库里进行查询，大家可以试想一下，做数据库查询需要知道哪些信息，有了这些信息，我们才能实现真正程度的召回：

- 哪个库哪个表。
- 哪些字段。
- 什么条件。

那么，对于原始query，我们需要做什么处理呢：

- 预处理。这个在之前的文章里面也说过很多次，繁体简体，大写小写，标点符号，数字等等，不赘述了。
- 改写。这个是搜索里面比较复杂的操作，后面会展开写，举个例子吧，招商银行和招行。
- 意图分类。对应是确定哪个库哪个表。
- 实体识别、term weighting等。对应的是哪个字段。

改写

首先是改写，首先需要明确的是，改写这个操作的目标是，要对应到数据库里面的数据，所谓的模糊搜索，能做到多模糊，其实就体现在你的改写能力好不好了，毕竟，**数据库的查询能力是不可能做模糊的**。

首先最简单的改写应该是同义词，这个就非常考验数据挖掘的能力了，怎么构建同义词词表，甚至是垂直领域下的数据挖掘能力，是非常关键的，具体怎么挖掘，这个就需要看大家的智慧啦。补充一下，这个同义词挖掘过程一般都是离线过程，在现阶段一般是直接触发词典来做改写的。

前缀匹配、拼音、拼音前缀、纠错。这个是搜索中的常见操作，用户很多时候不见得会输入所有内容甚至会输入错误，我们需要做补全（自动补全这个也有说法叫做query suggestion）。

- 前缀匹配同样可以通过词典来处理，约束好相似度（例如用编辑距离）即可。“番”直接改成“番禺野生动物园”可就不太合适了。
- 拼音结合拼音转换加词典的方式做改写就行，当然太简单也会有一些bad case，需要慢慢来补充迭代吧。
- 拼音前缀，hdl出海底捞，bd出百度，类似这种，也是可以通过构造实体词典的方式去做。

其他必要的改写，这个就要根据实际业务去做了，举个例子吧，搜索中有一种召回方式是从redis中做召回，这种方式的难度在于当且仅当只有精确匹配才能够找到结果，数据库里面可能没有钢铁侠，只有钢铁

侠1，钢铁侠2，钢铁侠3，此时用户输入钢铁侠的时候是不会出结果的，我们需要设置同义词，这种同义词就不是我们常说意义的同义词，这个就需要结合一些规则去做针对性的操作。

其次，应该就是要上一些比较复杂的模型了，不过因为使用的方式是在线，所以再复杂也不会这么复杂。模型实质上也是去找相似的内容，说到相似、近义词，大家应该想到词向量的最近邻相似了，将某个词汇通过word2vector的方法转为向量，然后找到最接近的几个词（一般用相似度或者个数来截断），就能作为改写词。值得注意的是，如果上述方案已经能覆盖较多结果了，其实并没有太高的优先级，模型常伴有一定的不稳定性，说不定哪天就有一些bad case出现，这种case又不好处理。

意图识别

意图识别如上所述的目标对应下游，是为了知道，你要去哪个数据库找，这样说的会直接一些。当然对于大搜，有比较多的广义内容可以出，这可以当做单个品类，不用做意图识别。

意图识别，说白了就是个文本分类的问题，但是你局限在文本分类本身，那肯定会有问题，我先把文本分类的问题说完。

为什么说是文本分类呢，主要因为针对用户query（就是一段文本），你要分析他有什么意图，日历、电影、地图等等，其实就是个分类问题了吧，所以常用的方式就是文本分类，常用的模型如下：

- fasttext。简单、速度快。
- textcnn，准确召回啥的会提升较多，但是样本依赖也会提升很多。
- bert系列，召准提升明显，速度降低，但是样本依赖由于fine tuning所以会减少很多，但是速度有质的下降。

然后来说为什么不能局限在文本分类。

- 首先，很多东西，不是语义就能处理的，而文本分类本身就是一个语义层面的东西。
- 某些意图是有时效性，模型可不能有这个更新频率。举个例子，“天气之子”在电影上映之前就没有电影意图。
- 模型可能会引入一些数据库不存在或者是无关的数据，这时候会出一些bad case。

那么模型的问题应该怎么解，没错，又是词典+模板，触发规则就是判断为特定意图，这个方式其实我个人非常喜欢，说说优点和缺点。

- 优点是准确率是真的高。
- 缺点是依赖词典挖掘，召回率低。

实体识别

实体识别这块，我曾经画上了比较多的章节讲过，此处就不赘述啦。

- NLP.TM[18] | 搜索中的命名实体识别

召回数据

上面内容的处理都是为了有更规范的信息进入数据库，能够更加精准的找到所需内容。这块写起来我想了超级久不知道怎么说，真的不好总结，我写一些我的理解吧。

数据库

一般会在什么数据库召回呢，首先一个强需求是速度足够快，不能一个查询数据库找一年对吧，因此常用的是这两种查询系统：

- Redis，K-V查询，速度很快。
- Elastic Search，为了搜索为建立的查询数据库。

相关性

虽说是召回，但是召回阶段还是要做一个分析，召回的内容和用户query之间的匹配程度，这个主要结果是为了后续的排序做支持（注意，最后的排序绝对不是文本相似度那么简单）。那么，具体有的方法如下：

- 规则，有些改写，可能会导致相似度很差，例如拼音转文字，这种要认为定相似度。
- BM25这种基于词频类型的相似度计算。
- 文本相似度模型。

相关性截断和过滤

并不是所有的召回都可以直接给下游排序的，有些内容需要在召回阶段干掉，减轻排序压力。主要方法如下：

- 根据相关性，过滤其实相关性不高的数据。
- 不合法数据，可以通过敏感词词典等方式解决。

至此，完成召回过程。

后记

当前，大部分推荐系统和搜索系统的文章，多半会把精力集中在排序阶段，召回阶段多半讲完word2vector和协同过滤就完事了，主要原因是召回阶段的细节操作太多，也不好总结，建议大家还是多