NLP.TM[18] | 搜索中的命名实体识别

原创 机智的叉烧 CS的陋室 2019-10-20

Don't

Dont Go - Dont (Ed Sheeran Covers)

【NLP.TM 】

本人有关自然语言处理和文本挖掘方面的学习和笔记，欢迎大家关注。

往期回顾：

- NLP.TM[13] | 命名实体识别基线 BiLSTM+CRF（上）
- NLP.TM[14] | 命名实体识别基线 BiLSTM+CRF（下）
- NLP.TM[15] | 短文本相似度-CNN_SIM
- NLP.TM[16] | SIGIR2019: 深度NLP在搜索系统中的应用
- NLP.TM[17] | 系列阶段总结

最近在做的工作主要是在命名实体识别上，那么在搜索场景，命名实体识别是一个什么样的存在，又是怎么实施落地的，今天来给大家具体讲讲。（额，又是一篇搜索和NLP交叉的文章，由于更偏向NLP的通式通法，所以我放在NLP系列啦）

有关命名实体识别，我在之前的文章已经谈过一次，看这里：

- NLP.TM[13] | 命名实体识别基线 BiLSTM+CRF（上）
- NLP.TM[14] | 命名实体识别基线 BiLSTM+CRF（下）

在此基础上，听我谈搜索中的业务可能就会比较容易了。

在文章前面给出参考文献：

- 美团旅游搜索召回策略演进：https://tech.meituan.com/2017/06/16/travel-search-strategy.html

简述搜索中的NLP应用

日常所谓的搜索，大家最常见的就是类似百度之类的大搜，当然也有像美团、淘宝的那种垂直领域的搜索，在现在互联网的环境下，虽然不如推荐系统热闹，但是却已经成为大家常见的应用中非常重要的一个模块，且所搜是否是所得，其实很大程度体现的就是用户的直接体验，从而一定程度上决定了用户的依赖性，举个例子，哪天百度搜出来的大部分东西都不是我想要的，那我日后基本就不会用百度去做搜索了。

那么，搜索中使用的NLP技术有多少呢，在我的角度下看，非常多。虽然用户搜索的内容各异，文档、商品、图片、视频，但是大部分的输入都只有一种——文字，没错，出了一部分能够输入图片进行搜索的引擎外，大部分的搜索系统都只用了一个东西，那就是文字，由于query的形式局限在文字中，因此对query进行分析的核心技术工具，就是NLP技术，这也造就了NLP在搜索系统中的重要定位。

那么都有哪些应用呢，我简单举几个例子：

- 搜索的意图识别。对于无差别的query，用户具体是什么样的意图，如何将不规范的query和规范化的数据库中的资源映射起来，这是非常困的。
- query改写。搜索引擎底层大都使用的倒排索引，只有映射到对应的倒排，才能够找到对应的资源，然而对于用户而言，某些词可能有很多说法，这些都要映射到对的词，才能够实现查询，例如同义词改写、前缀改写、拼音改写等。
- query-doc相似度。召回可以有很大的灵活，通过改写提升召回，保证该召回的内容被召回，而在排序阶段，为了保证用户的主观感受，必须做文本相似度计算，有些召回内容可能是有关，但是用户感知不明显，肯定不能往前排，因此至少有一个特征体现这两者的相似度。

命名实体识别的适用场景

学过数据库的应该很好理解，要在数据库中检索，必须知道你搜索的时候要在哪个字段里面搜什么，举个例子：“北京的温泉”，即使能有比较好的意图识别，知道是旅游意图，但是，在旅游数据库里，是需要通过字段去搜索的，“北京”是城市，“温泉”是旅游类目，而“的”是一个停止词，这些都是别出来，我们才能在数据库里面搜索，从而得到用户所需的内容推荐。

命名实体识别的常用方案

词典匹配

这个任务虽说是命名实体识别任务，但是却不见得需要建立一个模型才能解决，要进行一个初步的处理，快速上线，其实词典匹配的方法可能是最简单的，而实际上，即使是其他方法，我也很建议大家用这个方式去做一遍，理由后面会谈。

词典匹配的便捷性体现在你真的很容易就能拿到这个词典资源，因为你做搜索，所需要的数据，其实已经在数据库或者底层搜索引擎里面了（没有资源你怎么做搜索推荐？），你可以将数据库内的数据按照字段

提取，然后通过n-gram的方式切词，即可完成一个初步的词典，复杂的，进一步，为了保证词典的可靠性，你可能需要删除一些不适合再次点出现的词汇，举例，酒店名字段中，其实没有必要存“酒店”做为词条，首先召回的时候，大部分酒店都有“酒店”一词，他没有明显地指向性，然后，这种召回也会增加排序的负担。

有了词典之后，就可以通过词典匹配的形式进行命名实体识别。上面给出的例子：“北京的温泉”，就可以快速标记“city-object-type”，然后就可以通过这个实体识别结果，拼好检索语法，完成召回。

机器学习方法

机器学习方法，包括深度学习，是现行的主流方法，我也最建议用这种方法上线。

- 最大熵、HMM、CRF都是轻量级的模型。适合初版功能上线。
- 预训练+RNN系，甚至是transformer模型则适合后续的迭代更新，但当然的随着模型变得复杂，模型体量会上升，响应时间也会上升。

机器学习方法具体实现思路

机器学习方法在这块，很难是无监督学习，顶多也要是半监督学习，当然在搜索场景，我们其实可以跳过半监督学习直接使用监督学习完成，来看看具体的步骤。

数据集的构建和构成

数据集构建是命名实体识别的瓶颈难题，但是在搜索中，其实我们可以轻松解决——词典匹配。词典匹配是一种无监督的方法，而且标注的准确性也相对较高，因此作为有监督学习的准确性。

由于数据集构建是线下的过程，因此使用python脚本比较简单，这里给一个trick，读取字典后建议使用集合 `set` 类型进行存储，主要有下面几个优点：

- `set()` 类型天生具备去重功能，防止你的字典太大。
- `set()` 类型的添加和检索复杂度都是 `O(1)`，计算起来更快。

说到数据集的构成，最近也是想法颇多，简单的总结下来，其实在这块，要保证数据集中尽可能要满足下面的元素：

- 语法结构的完备性和多样性。
- 词汇完备性和多样性。
- 尽可能平衡甚至强化关键实体的出现。

因此，除了日常的query能用，有的时候你甚至可以自己制造一些比较热门的语法结构模板，通过这种模板造一些数据。

训练

训练阶段其实对大家来说就比较简单了，按照模型的开发标准，直接使用即可。

测试

测试这块，在很多文章看来非常简单，但在我最近的经验看来，远远没有想的那么简单，核心原因在于，测试集的构建。

常规思维，测试集是从样本总量里面抽取一部分，但事实上并非如此，核心原因在于整个数据集并不是现实场景应用的数据集，这里的数据集如前面的内容所述，还包括一些拼接的、外部的数据，而这里测试的，其实是需要真正应用场景的那些数据，才更为可靠。

另一方面，测试集认为正确的，其实是词典标注的数据，但是词典标注一定就对吗？其实并不一定吧，原因有几个：

- 词典标注没考虑上下文，只做了匹配
- 词典不一定具有完备性，所有试题都被识别出，这个和词典的构建来源于资源有关

因此，个人建议在通过基本的指标进行总体分析后，再自己抽100-200条进行人工测试（其实这个时间并不长，10来分钟完全足够），看看自己的数据下，有多少是词典标注错误导致的预测错误，这个实体可能模型可以识别对了，类似的问题，真的只能抽case来分析了，而且建议一定要做这种分析，毕竟你的预测正确与否，是靠真实场景用户预测决定的，而不是你词典标注的样本决定的。