《现代信息检索》大作业

November 30, 2020 at 20:51pm

Professor：何苯

李金洋 202028013229069

王凯菲 2020E8013282086

1. BM25

我们所研究的问题是需要给出指定查询的排序式检索，给定一个查询，将查询与它所对应的 1,000 个文档片段进行评分，并按照评分由高到低排序得到结果。在传统信息检索方法中，检索效果最好的是基于概率的 BM25 模型。本节介绍了 BM25 的基本定义及各种优化，并使用该模型解决本次大作业的问题。

1.1 基本概念

TF-IDF 是经典的文本相似度算法，旨在反映单词对集合或语料库中文档的重要性。它通常用作信息检索，文本挖掘和用户建模搜索中的加权因子。搜索引擎经常使用 TF-IDF 加权方案的变体作为在给定用户查询时对文档相关性进行评分和排名的中心工具。假设有文档集合 $D = \{d_1,\ d_2,\ d_3\ \ldots,\ d_n\}$。文档集合中共包含 m 个不同的词记为 $W = \{w_1,\ w_2,\ w_3,\ \ldots,\ w_m\}$。词项 w_t 在文档 d_p 中的 $TF - IDF$ 的权重为，

$$w_{td} = (1 + \log tf_{td}) \cdot idf_t \tag{1}$$

其中 tf_{td} 是指文档词频，即单词 w_t 在文档 d_p 中出现的次数，代表单词出现频次越高，得分越大。idf_t 是词项 w_t 的逆文档频率。

$$idf_t = \log \frac{N}{df_t} \tag{2}$$

上式中的 df_t 是出现词项 w_t 的文档数目，N 是文档集的总文档数，idf 项代表单词越稀有单词的权重越大。因此我们可以将文档中每个单词的 $tf - idf$ 值组成向量来表示该文档，再根据余弦相似度等方法来计算文档之间的相关性。

BM25 模型是基于 TF-IDF 的改进，其在传统 TF-IDF 的基础上增加了几个可调节的参数，使得它在应用上更佳灵活和强大，具有较高的实用性。假设有查询 Q（Q 中词项为 $q_1,\ q_2,\ q_3\ \ldots,\ q_m$）和文档 d，二者的 BM25 相似性得分为，

$$BM25(Q,\ d) = \sum_q w(q, d) \tag{3}$$

$$w(q, d) = \frac{(k_3 + 1) \times qtf}{k_3 + qtf} \times \frac{(k_1 + 1) \times tf_q}{tf_q + k1(1 - b + b \times \frac{l_d}{avg_l})} \times \log_2 \frac{N - df_q + 0.5}{df_q + 0.5} \tag{4}$$

公式 (3) 代表查询 Q 与 d 的得分需要 Q 中每一个词项与 d 计算得分后求和。

公式 (4) 是词项与文档的匹配度，它由三项相乘组成。第一项是词项 q 在查询中的得分，如果某单词在查询 Q 中出现频次很高，则该查询词的权重应该更大。第二项类似于 TF-IDF 算法的 TF 项，代表词项 q 在文档 d 中的得分，如果词项在文档中多次出现，则得分越高。需要注意的是如果文档本身

很长，则每个词项出现的频次都倾向于变大，所以使用该文档长度 l_d 除以平均文档长度 avg_l 来起到归一的作用，以减小文档长度的影响。第三项类似于 TF-IDF 算法的 IDF 项，将函件词的权重调大，常见词权重调小。如“阿尔兹海默症”一词虽然出现的频次很小，但是能够很好地区分不同文档，这种词对文档的贡献分应该更大。

公式 (4) 中的 k_1 k_3 b 均为可调节参数。其中 k_1 k_3 保证了对应项不能无限增加，第一项最多只能到 k_3，第二项最多到 k_1。

1.2 优化策略

对于查询和文档中的每个单词都需要预处理，在本次实验中，将所有单词都映射到了小写形式，并使用 nltk 库对单词进行词干还原。某些没有实际意义的超高频词汇，如“the，to”等，需要根据停用词表去除。

除此之外本实验还采用了查询扩展策略来优化模型。考虑查询 q = “aircraft”，若某篇文档中包含“plane”，但是不包含“aircraft”，显然对于查询 q，一个简单的 IR 系统不会返回文档 d，即使 d 是和 q 最相关的文档。所以我们需要扩展查询，将近义词“plane”也添加到查询中。方法是不考虑查询及其返回文档情况下，对初始查询进行扩展和重构，即进行一次性的全局分析 (比如分析整个文档集) 来产生同/近义词词典。同时词项的扩展可以与词项的权重计算相结合，比如对增加的查询词项赋予一个低于原始查询词项的权重。

本次实验中使用预训练的词向量文档，对每个查询词都提取评分大于 0.75 的前 5 个单词进行扩展，形成新的查询。

1.3 模型实现

代码主要包括 CorpusParser、QueryParser、QueryProcessor 类。其中 QueryParser 类用于读取测试集的全部查询信息，并对每个查询进行查询扩展优化。CorpusParser 类用于读取语料池，对于每一个查询，读取相应的 1000 条段落信息。QueryParser 和 CorpusParser 类都继承于 Parser 类，Parser 类中定义了词项预处理方法。读取测试信息后，使用 QueryProcessor 类按照 BM25 指标进行评分排序。

2. 实验结果分析

对于每个模型输出都为标准 TREC 格式，使用 NDCG@10 作为指标进行评价，结果如表 1 所示。raw 代表原始测试数据排序的得分，BM25，BM25-1，BM25-2 分别代表未使用优化、使用词项预处理（包括标准化和去停用词）、使用词项预处理和查询扩展的结果。可以看到对于 BM25 模型，词项预处理、查询扩展等优化方法的使用，使得检索效果有所提升，但整体上效果不佳。

表 1: NDCG@10 评测结果

模型	NDCG@10
raw	0.0482
BM25	0.1475
BM25-1	0.1590
BM25-2	0.1675

3. 应用

需要依赖两个库，使用命令 pip install gensim 和 pip install nltk 安装。单词预处理还需要使用 nltk 的语料库 wordnet，输入命令

import nltk 和 nltk.download("wordnet") 进行下载。

测试步骤是，首先进入../BM25-qing/src 路径下，输入如下命令:

python main.py parameter1 parameter2 parameter3 parameter4 parameter5

其中参数 1、2、3 都是必须输入的，分别为语料库文件（本实验为 msmarco-passagetest2019-43-sorted-top1000.tsv 文件）、查询文件（本实验为 msmarco-test2019-43-queries.tsv 文件）、输出文件路径。参数 4，5 是可选的，分别为停用词表文件路径和词向量模型路径，如果输入这两个参数，则代表要对模型进行词项预处理和查询扩展的优化。

计算完成后会在参数 3 指定的路径下生成结果文件 BM25.txt 结果文件。