Contextualized Offline Relevance Weighting for Efficient and Effective Neural Retrieval

蒋浩南_20221021_论文笔记

<table>
<tr><td rowspan="4">基本信息</td><td>发表刊物</td><td>Sigir2021</td><td>发表年份</td><td>2021</td><td>第一完成单位（国内）</td><td>中国科学院大学</td></tr>
<tr><td>作者</td><td colspan="5">Xuanang Chen， Ben He， Kai Hui，Yiran Wang，Le Sun，Yingfei Sun</td></tr>
<tr><td>关键词（中文）</td><td colspan="5">神经信息检索，高效，离线相关性加权，BERT</td></tr>
<tr><td>关键词（英文）</td><td colspan="5">Neural IR; Efficiency; Offline Relevance Weighting; BERT</td></tr>
<tr><td rowspan="3">论文内容</td><td>解决的问题（如有实际应用场景请说明）</td><td colspan="5">在检索应用中部署大规模预训练语言模型有很高的在线搜索延迟。所以如果要部署，就要平衡有效性和延迟。
本论文提出的模型，可以实现利用大规模预训练模型的有效且高效的在线搜素。</td></tr>
<tr><td>解决问题的方法（采用什么模型框架等）</td><td colspan="5">将昂贵（高延时）的在线的查询-文档匹配分解为两部分，即在线的查询-伪查询匹配和离线的伪查询-文档相关性加权。</td></tr>
<tr><td>模型框架实现过程</td><td colspan="5">在语料库 C 中，d 为文档。
离线：
1、伪查询生成，Q_d = doc2query(d)。
2、寻找相邻文档，D_d = doc2doc(d)
3、使用 BERT 预计算相关性得分，对$d \in C$. 每个$\bar{q}_j \in Q_d$ 和每个$d_i \in D_d$，$rel(\bar{q}_j, d_i) = \text{BERT}(\bar{q}_j, d_i)$
在线：
1、生成候选文档，给定查询 q，使用词典匹配功能 BM25 生成种子文档集 Sq = {$d1, d2, \cdots, ds$}，对每个种子文档集中的文档 d 使用 doc2doc(d)获得其相邻文档，所得相交形成候选集 Rq = {$d1, d2, \cdots, dr$}。
2、候选文档相关性计算：
（1）对于每个候选文档集中得文档$di \in R_q$，找一个种子文档$d \in Sq$，该种子文档生成的相邻文档集中包含该文档 d_i。
（2）查询 q 和所选候选文档 di 的相关性可以近似为查询 q 和该候选文档所选的种子文档的伪查询($\bar{q}_j \in Q_d$)的相关性。
（3）由于不止一个种子文档$\in Sq$生成的文档集合包含所选候选文档，所以取最大值。$rel(q, d_i) = \max_{\bar{q}_j \in Q_d} sim(q, \bar{q}_j) \times rel(\bar{q}_j, d_i)$。
（4）查询和伪查询之间的相似性：$sim(q, \bar{q}_j) = \text{BERT}(q, \bar{q}_j)$
（5）以上只考虑了上下文语义相关性，进一步添加了来自词典模型 BM25 的精确匹配信号。
$\widehat{rel}(q, d_i) = \alpha \times rel(q, d_i) + (1 - \alpha) \times \text{BM25}(q, d_i)$</td></tr>
</table>

		其中α权衡语义相关性得分和 BM25 得分之间的权重（归一化为[0,1]）。 3、根据每个候选文档的相关性得分排序。
	仍旧存在的问题（注明论文中说明的问题或自己认为存在的问题）	1、预计算的花费高,需要高昂的计算硬件来减少大量计算所需的计算时间。(论文说明) 2、所需的空间大，按照论文中给出的数据 1 个文件有 5 个伪查询一个伪查询占 34B，共 170B，1000 个相邻文档，加上 5000（1000*5）个分数，每个 4B 共 200170B。 而作者举对一个文件 ColBERT 需要 102, 400 (5×) bytes 的例子，本模型只为 ColBERT 的 0.2 倍做对比。这样看 0.2 看似不大，其实还是占很大空间。（我认为） 3、对于非固定的语料库，需要经常更新离线的相关性得分，而且计算量是庞大且耗时的。（我认为）
实验内容	实验采用的数据集	MS MARCO 数据集，测试查询来自 TREC 2019 and 2020 Deep Learning (DL)Track 与 NIST assessors 的手动多级判断一起使用，以计算不同池深的精确度和召回率。
	数据集内容是否和待解决问题模型对应	是
	实验是否涉及实际应用场景	否
	实验采用的对比方法	将本模型与传统的模型进行比较。 传统模型有 BM25、,BM25+RM3、DeepCT、doc2query 和 docTTTTTquery。
	实验任务	1、分别在 TREC 2019 and 2020DLTrack 上再分别测试检索 passage 或 document 排序任务的实验指标。 2、在将不同模型排序后的段在经过 BERT-Base 或 ColBERT 再次排序后比较 NDCG@10。 3、测试延时。
	实验衡量指标	评估指标为 MRR, NDCG@10 和 MAP 和用来测试候选集质量的召回率 R@100, R@500 和 R@1000。
	实验说明所提出方法的优点	1、对于段排序和文件排序都有在基本所有测试的绝大多数测试指标都优于其他模型。少数也非常接近最优。 2、经过本模型排序过的段，在经过 BERT-Base 或 ColBERT 再次排序后，与其他模型相比排序在重排后得到更好的效果，所以只需少量候选集就可以得到很好的效果。 3、延迟方面，在种子文件不超过 50 的时候有着优于 ColBERT 的延时。由 2 可知只需要少量的候选集，在此情况下需要的延时也少，所以竞争性很高。
思考	论文的主要优	短小精悍，特别是对模型构建的描述，清楚易懂。

内容（阅读论文后自己思考填充）	点是什么	
	论文仍然可以改进的地方是什么	有些重复的东西。可以对涉及的模型的介绍。增加更多的评价指标和实验设置。
	以此论文为出发点，如果需要你做一篇和其相关的顶会论文，你需要的资源是什么？数据，硬件，技术支持等	首先需要了解该领域的各种预训练模型，然后熟悉相关训练集和测试。要有大空间的硬盘存储大量数据集和产生的中间文件，和高性能的 GPU 等硬件来加快大量数据的计算。
	其他想要补充说明的内容	本质来说还是把 BERT 计算文档-查询匹配这种的高延时的计算在离线的时候都算好存起来。但却是迂回实现了端到端和在线搜索的低延时。而且保证了有效且高效。(但离线计算量是恐怖的，实际应用的成本很大估计，可能要等神经网络模型的发展)。