信息系统检索

功能设计：大规模预训练模型在信息检索系统的部署

姓名： 蒋浩南　　学号：2012948

一、背景与意义

神经信息检索(Neural Information Retrieval, Neural IR)是信息检索领域的一个重要研究课题。自从谷歌在2018年发布BERT以来，它在11个NLP任务上获得了最先进的结果，一举改变了整个NLP领域的研究范式。2019年1月，Nogueira和Cho在MS MARCO Passage Ranking测试集上首次使用BERT。从时起，人们开始研究神经信息检索的范式，也提出了许多基于BERT的文本排序方法。这些方法在多阶段搜索架构的重排阶段(Re-Ranker)被应用。

1. 对于信息检索的三个层次：

- 从基础的问题角度。查询与文档的相关性度量是信息检索的核心问题。该层次重点关注的是检索结果的准确性，但相关性目前仍无明确定义，是与场景、需求和认知有关的一个概念。即表示为Relevance Estimation:

$$s_{ij} = R(q, d)$$

- 从系统检索框架的角度，需要从大量的文档集合里快速返回相关的文档并排序，除了考虑正确性还要考虑检索效率。可表述为Retrieval Process：

$$r = f(q, [d_0, d_1, \ldots, d_n] \in D)$$

- 从系统的角度来看，不仅要解决排序的问题，还要解决用户意图模糊、用户输入有错、文档结构异质等问题。可表述为Search Engine：

$$\{Q, D, F, R_{ij}(q_i, d_j)\}$$

2. 搜索的常规结构：

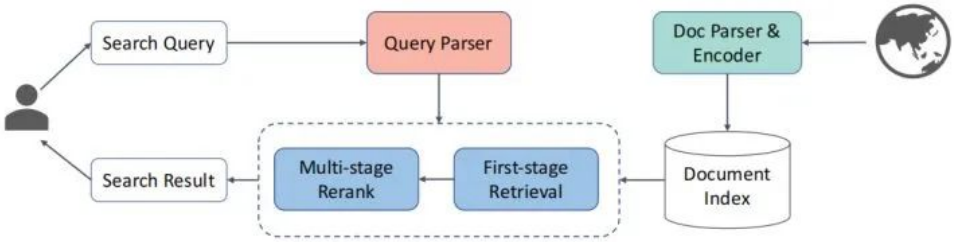


Figure 2.3: The framework of a practical search system.

- Query Parser，即query预处理和理解部分，需要通过算法的方式对query进行解析。
- Doc Parser&Encoder，即对文档处理的部分，什么样的解析和表征能更好地入库检索的同时，更快更准地被Retrieval找到。
- Retrieval，即检索部分，从库里面粗筛出可能相关的文档。
- Rerank，即排序部分，将选出的文档计算相关度得分，排序文档。

3. 相关性度量：

相关性度量的建模范式，经历了传统检索模型、Learning-to-rank （LTR）模型、Neural IR（NeuIR）模型的演进。

- 传统检索模型更多关注查询和文档中重叠的词，在此基础上建模权重，后将不同权重的得分组合，得到相关性得分。代表性的方法包括 PlV、DIR、Language Model、BM25等，都是通过对词频、文档长度、以及词的IDF的不同组合方式，来得到度量的得分。
- LTR模型的建模过程，是从指标函数的计算变成特征学习的过程。通过人工构造不同特征，后采用神经网络或者函数优化的方式，来解决特征组合的问题。代表方法包括RankNet、RankSVM、LambdaMart等。
- NeuIR方法更进一步，通过模型学习构建人工设计特征的过程，采用机器学习优化的方式来抽取特征。大致可以分为3类：一是基于表示的方法，单独学习查询和文档的表示进行打分；二是基于交互的方法，把查询和文档从底层做交互，对交互信号进行抽象得到打分；三是将二者结合起来进行打分。

4. 预训练语言模型

预训练语言模型（如 BERT、GPT 等）已经广泛应用于许多 NLP 任务。这些模型是基于大量的文本语料库（如：Wikipedia）进行自监督训练的，进一步 fine-tune 模型输出的表示可以在各种下游任务上取得很好的成果。

BERT和GPT等大型预训练模型(PTM)取得了巨大的成功，成为人工智能(AI)领域的一个里程碑。由于复杂的训练前目标和庞大的模型参数，大规模PTMs能够有效地从大量有标签和无标签的数据中捕获知识。通过将知识存储到巨大的参数中，并对特定的任务进行微调，隐含在巨大参数中的丰富知识可以使各种下游任务受益，这已通过实验验证和经验分析得到广泛证明。现在AI社区的共识是采用PTMs作为下游任务的骨干，利用丰富的上下文，提高计算效率，以及进行解释和理论分析。最后，我们讨论了PTMs的一系列有待解决的问题和研究方向，希望我们的观点能对PTMs的未来研究有所启发和推动。

近年来，预训练模型在自然语言处理的不同任务中都取得了极大的成功，在信息检索中也进步不小。

在此，由于预训练模型的发展及其在rerank等各部分上的有效性，可以考虑设计一个模型运用于检索系统。在该系统中可以利用预训练模型的优势，获得更加符合个性化的有效的且高效的检索系统。

二、现状分析

检索组件中预训练方法的应用

1. 第一阶段检索早期的方法是基于 Term-based 的方法，主要是对文档清洗，通过倒排方式索引到引擎中。这类方法面临两个经典的问题：

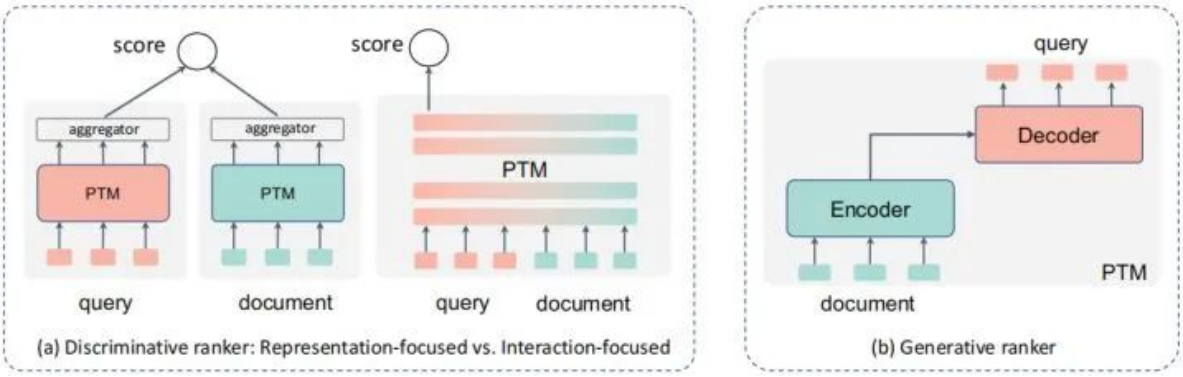
- 一是查询稀疏与文档冗余带来的词失配的问题；
- 二是无序词排列导致丢失语义依赖信息的问题。这两个问题造成语义信息丢失，导致后续建模无法提升性能。

2. 针对以上问题主要有三类不同提升方法，分别是：稀疏检索方法（Sparse Retrieval）、稠密检索方法（Dense Retrieval）、混合方法（Hybrid Retrieval）。

- **Sparse Retrieval**：保持表示的稀疏性，与倒排索引集成以实现高效检索。
- **Dense Retrieval**：从符号空间到语义空间，使用神经网络算法进行语义相似检索
- **Hybrid**：结合基于倒排的检索和语义检索，继承两者的优点。

预训练方法在重排序构件中的应用

- 判别式，即直接用类似分类的方式，直接给出query对各个doc的打分，选下图种中间的那个形式。
- 生成式，假设文档和query中间存在一个生成的过程，通过刻画文档->query或相反的过程来判断两者的相似关系。



大规模预训练模型在信息检索系统中应用于不同组件，包括第一阶段检索组件、重新排序组件和其他组件。当然基于上述模型所设计的检索系统具有较高的有效性。但却不具备真正部署于实际情况的条件。特别是在在线检索系统。上述模型对于文档与查询的匹配需要较长的时间。

故有如下设计，希望将大规模预训练模型运用在检索系统中，并在保证有效性的同时保证有较高的效率。

三、方案设计

对于该系统的设计，我们将其分为离线的预处理阶段和在线的查询阶段。

在线查询阶段：

1. 针对模型的优化所起的效果在在线查询阶段不佳。又需要保证在线查询阶段使用大规模预训练模型的高效。那么在在线查询阶段就不能使用高昂的（查询-文档）的匹配。取而代之，使用相对低廉的（查询-伪查询）的匹配。
2. 既然选择使用查询-伪查询匹配。那么在文档集排序阶段计算相关性得分时，就需要进行相应文档的伪查询与提供查询的相关性计算。该计算由于使用了查询，必须在在线查询阶段实时计算，且该种方式的代价相对低廉。

3. 上述，就是使用文档的伪查询作为（文档-查询）相关性计算之间的桥梁。现在已经连接了（查询-伪查询）之间的相关性得分计算。之后便需要再连接（伪查询-文档）之间的相关性得分计算。就可以建立起该桥梁。
4. 但是，由于伪查询-文档之间的相关性得分的计算是耗时的。而且，并不涉及查询，所以将该计算提前至离线时就计算并存储起来。利用空间换时间，利用闲置时间换时。
5. 上述是关于文档的排序计算的问题。此处，需要解决文档集的选择问题。
6. 一种设计思路，根据查询，利用doc2query/docTTTTTquery之类的模型，获得一定规模的文档集合。
7. 另一种设计思路，利用BM25之类的模型获得较小规模的种子文档集合，在种子文档的基础上，使用doc2doc获得其相邻文档，将两个得到的相邻文档集和种子文档集结合，作为要进行排序的文档集。而对于这种思路，由于使用伪查询，或许可以得到更好的效果。

离线预处理阶段：

1. 根据上述在线检索阶段的需求，在离线的预处理阶段，需要提前计算好（文档-伪查询）的相关性得分，并储存起来。这样当在线检索需要时，就可以直接调用，省去时间。
2. 于是，需要对于给定语料库里每一个文档生成其伪查询。
3. 计算文档与其伪查询的相关性得分，存储。

四、技术路线

离线预处理阶段：

对于语料库C：

1. 伪查询生成：

$$Q_d\text{伪查询集}$$
$$Q_d = doc2query(d_i)$$
$$d_i \in C$$

2. 相关性得分计算：以使用BERT为例。$d \in$ C. 每个 $j \in Qd$

$$\text{对于}d_i \in C, \overline{q}_j \in Q_d$$
$$\text{相关性得分}：rel(d_i, \overline{q}_j) = BERT(d_i, \overline{q}_j)$$

在线查询阶段：

给定查询q。

设计思路1：

1. 根据查询q，得到要排序的文档集：

$$\text{文档集}R$$
$$R = query2doc(q)$$

2. 生成文档集中每个文档的伪查询：

$$Q_d, d_i \in R$$
$$Q_d = doc2query(d_i)$$

3. 根据为每个文档生成的伪查询，乘离线计算的伪查询和文档的相关性得分，得到该文档与查询q的得分。

$$d_i \in R, \overline{q}_j \in Q_d, \text{给定查询} q$$
$$rel(d_i, q) = BERT(q, \overline{q}_j) * rel(d_i, \overline{q}_j)$$

设计思路2：

1. 根据查询q，得到种子文档集：

$$\text{种子文档集} S_q = \{d_1, d_2, \ldots, d_s\}$$
$$S_q = BM25(q)$$

2. 种子文档集中每个文档计算其相邻文档，组合形成要排序的文档集合。

$$\text{要排序的文档集合} R_q$$
$$d_i \in S_q$$
$$R_q = S_q \cup doc2doc(d_i)$$

3. 对于每个文档属于要排序的文档集，找到一个种子文档，该种子文档的相邻文档包含所选文档。

$$\text{对于} d_i \in R_q$$
$$\text{选取} d, d_i \in doc2doc(d)$$
$$d\text{可能不止一个}$$

4. 计算d的伪查询：

$$Q_d = doc2query(d)$$

5. 得到文档与查询q的得分。（不止一个d,取最大值）

$$d_i \in R, \overline{q}_j \in Q_d, \text{给定查询} q$$
$$rel(d_i, q) = max \quad BERT(q, \overline{q}_j) * rel(d_i, \overline{q}_j)$$

五、总结

将大规模预训练模型运用在实际的检索系统中，为了保证其是有效且高效的。将高昂的（查询-文档）的匹配拆开。以伪查询作为中间的桥梁。在在线检索时，根据提供的查询，计算（查询-伪查询）之间的得分，而（伪查询-文档）之间的得分，则提前计算并存储起来。

利用空间和无关时间提前计算和存储，保证了系统查询时的高效。利用大规模预训练模型，使用一系列方式，保证拆分计算以及最后的合并不会损失。来保证查询的有效性。

但局限于大规模预训练模型，需要耗费很大的算力，来做好离线时的预处理计算。对于存储空间的需求并不算大。而且当语料库更新时，对于预计算的更新也比较容易。

六、参考文献

[1] Fan Y , Xie X , Cai Y , et al. Pre-training Methods in Information Retrieval[J]. arXiv e-prints, 2021.

[2] Guoa J , Fana Y , Panga L , et al. A Deep Look into Neural Ranking Models for Information Retrieval[J]. Information Processing & Management, 2019:102067.

[3] Han X , Zhang Z , Ding N , et al. Pre-Trained Models: Past, Present and Future[J]. 2021.

[4] Chen X , He B , Hui K , et al. Contextualized Offline Relevance Weighting for Efficient and Effective Neural Retrieval[C]// International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM, 2021.