技术方案

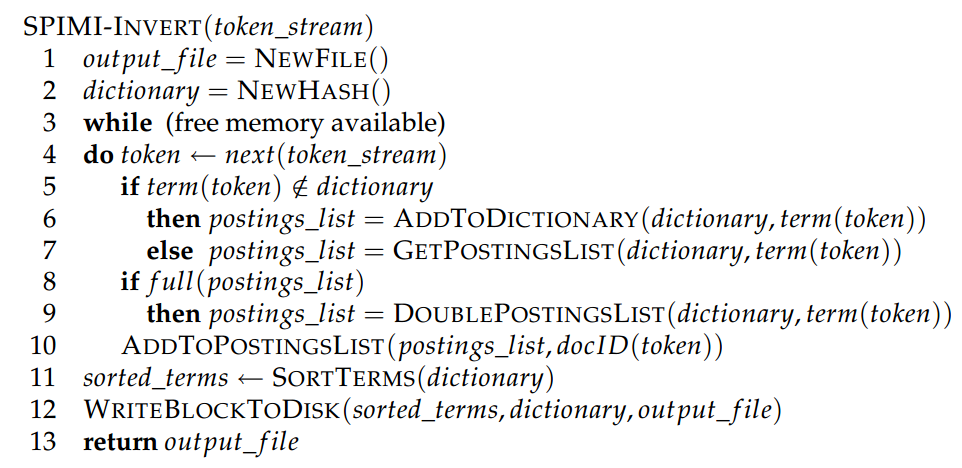
本项目总体采用bs架构，实现了问答式的搜索引擎。提供检索、问答的功能。

整体使用python语言开发，后端使用flask框架，数据库采用sqlite轻型数据库，检索模块为定制（纯python内核，可修改内部检索算法），深度学习框架使用pytorch，预训练语言模型使用bert-base-chinese。

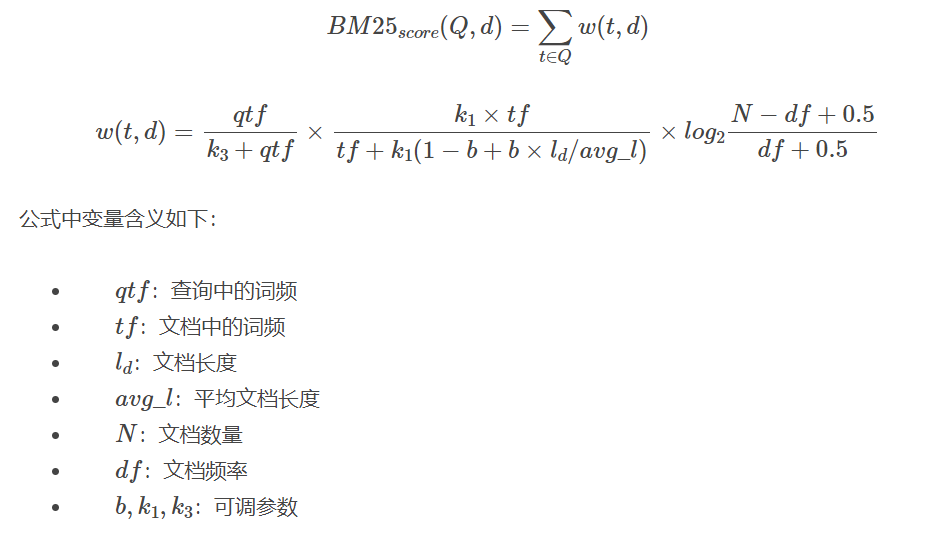
核心硬件配置：nvidia 1080ti显卡1块，intel core i5处理器

检索模块：

倒排索引构建算法使用内存式单遍扫描索引构建方法（SPIMI），其实就是依次对每篇文章进行分词，如果出现新的词项则插入到词典中，否则将该文档的信息追加到词项对应的倒排记录表中。SPIMI的伪代码如下：



采用基于概率的BM25模型。

给定一个查询Q和一篇文档d，d对Q的BM25得分公式为

第一个公式是外部公式，一个查询Q可能包含多个词项，比如“苹果手机”就包含“苹果”和“手机”两个词项，我们需要分别计算“苹果”和“手机”对某个文档d的贡献分数w(t,d)，然后将他们加起来就是整个文档d相对于查询Q的得分。

第二个公式就是计算某个词项t在文档d中的得分，它包括三个部分。第一个部分是词项t在查询Q中的得分，比如查询“中国人说中国话”中“中国”出现了两次，此时qtf=2，说明这个查询希望找到的文档和“中国”更相关，“中国”的权重应该更大，但是通常情况下，查询Q都很短，而且不太可能包含相同的词项，所以这个因子是一个常数，我们在实现的时候可以忽略。

第二部分类似于TFIDF模型中的TF项。也就是说某个词项t在文档d中出现次数越多，则t越重要，但是文档长度越长，tf也倾向于变大，所以使用文档长度除以平均长度起到某种归一化的效果，和是可调参数。

第三部分类似于TFIDF模型中的IDF项。也就是说虽然“的”、“地”、“得”等停用词在某文档d中出现的次数很多，但是他们在很多文档中都出现过，所以这些词对d的贡献分并不高，接近于0；反而那些很稀有的词如”糖尿病“能够很好的区分不同文档，这些词对文档的贡献分应该较高。

所以根据BM25公式，我们可以很快计算出不同文档t对查询Q的得分情况，然后按得分高低排序给出结果。

问答模块：

传统的搜索引擎基于关键字词检索对应的上下文，而想要知道的答案却需要从大量正文中去寻找。为直接返回给用户答案，这里使用机器阅读理解模型，它能够代替用户去阅读搜索结果中的内容，在数十毫秒内将最值得用户去关注的片段挑选出来。因为模型的直接回答是在用户发起搜索时配合语义搜索在最新的语料上实时计算得到的，因此其覆盖面为无限大，不受任何领域限制，时效性也更强。这种做法会导致较大的不确定性，为了避免对用户造成误导，搜索引擎直接回答都会明确给出来源上下文供用户进一步判断。

本项目采用抽取阅读理解的方法。抽取式阅读理解的原理示意图如下图，模型的输入是context和question两个，然后经过BERT进行特征提取，然后获得BERT的输出，连接一个全连接层，利用softmax函数得到开始位置（Output start）和结束位置(Output end)。具体地，BERT输出的序列中，每个token向量的维度是768维，序列的长度是N，经过全连接层之后，得到 ON×2=FC(TN×768)O^{N \times 2} = FC(T^{N \times 768})O N×2 =FC(TN×768 ) ，其中FC表示的是全连接层，ON×2O^{N\times 2}ON×2为每一个token分别作为答案开头和结尾的logit值，再经过Softmax层之后就得到了相应的概率值。

