大作业 问答系统设计与实现

## 实验目的

帮助学生对问答系统的设计与实现过程有一个全面的了解，培养学生使用课堂中介绍过的所有机器学习方法和深度学习方法实现问答系统并将其运用到实际应用中的能力，主要内容包括：对给定的文本集合进行处理；建立文档索引；对问题句进行分类；找出问题的候选答案句；对候选答案句进行排序；对候选答案句进行答案抽取；逐步调优。

## 实验内容

本次实验的语料分为两个部分，文档库passages\_multi\_sentences.json和问答语料train.json来自百度2017年发布的阅读理解数据集DuReader V2.0，经过筛选和处理；问题分类语料来自哈工大信息检索研究室于2004年公开的问答系统问题集。

本次实验中，同学们的任务如下所示：

1. 自行建立一个检索系统，从文本库中检索到与问题最相关的文档（可以是一个或者多个）。
   1. 首先对passages\_multi\_sentences.json中的文档以及train.json中的问题进行分词，并去除停用词。
   2. 接下来为文档建立索引，可以使用开源的检索系统Whoosh，也可以自行构建倒排索引，索引项为文档以及文档编号pid。
   3. 对问题进行预处理，去除停用词，并选择合适的方法（如词性标注、依存句法以及语义角色标注等）抽取问题中的关键词，在检索系统中进行问题查询，从返回的pid中选取前三个作为最后的检索结果。
   4. 结果评估，利用train.json中正确的pid以及检索系统返回的前三个pid来计算检索的准确率（acc），即第一个文档就是目标文档的概率Top1，以及前三个文档中包含目标文档的概率Top3。
2. 构建问题分类器。
   1. 首先对train\_questions.txt和test\_questions.txt中的问题进行预处理，并选择合适的方式将其向量化。可以选择词袋模型搭配TF-IDF，也可以用预训练模型。
   2. 选择课堂中介绍过的机器学习分类器或深度神经网络构建并训练文本分类器。
   3. 在test\_questions.txt上评估细分类以及粗分类准确率（acc），数据中的类别标签形如HUM\_PERSON，下划线前即为粗分类标签。
3. 对文档中的候选答案句进行排序，抽取出最相关的候选答案句。
   1. 将train.json自行划分为训练集（90%）和测试集（10%）
   2. 选择课堂中介绍过的机器学习分类器或深度神经网络构建候选答案句排序系统（Learning to rank），利用2中训练好的问题分类器对问题进行分类来作为候选答案句排序的特征之一，在这里提供两种参考方案：
      1. 将文档与Query的相关程度打上类别标签（有关/无关，或更加细分的相关程度），这样就将问题转为二（多）分类问题；
      2. 直接计算文档与Query的相关性得分，根据得分对候选答案句进行排序。
   3. 在测试集上评估候选答案句排序的性能，采用MRR（Mean Reciprocal Ranking）指标。



1. 最后，在最相关的候选答案句中抽取最精简的答案，这个答案可能是一个词或者几个词。
   1. 选择课堂中介绍过的机器学习分类器或深度神经网络构建答案抽取模块，参考方法为将答案抽取问题转换为序列标注问题，应用词性标注、命名实体识别或事件抽取中的相关方法实现；或利用机器阅读理解课程中介绍片段抽取式MRC使用的边界模型。
   2. 利用精确匹配（EM）、精确率、召回率、F1值和BLEU值作为评价指标，在测试集上进行评价。
2. 由于系统采用pipeline的形式，所以处理需要对每个模块进行单独评测（即使用Ground truth作为上一步输入）以外，还需要对整个系统进行评测（即每一步使用上一步的输出作为输入）。

## 附加实验

附加实验旨在给学生在文档检索上提供另一种思路，该部分实验为选做内容。

对于文件检索方式有两种，分别是 term-based、semantic-based。前者通过建立索引来查询，后者通过双塔网络和 ANN (Approximate Nearest Neighbor) 。但是两者都有缺陷缺陷，前者不能得到 document 的语义信息，不能匹配去有相同语义的不同词汇；后者首先在单一向量尚不能存储所有语义信息，并且重度依赖精确的匹配，第二，模型在深度文献查询交互上不可用（归因于 ANN 算法在理论上不完善）。总的来说，端到端文档检索的召回性能仍有很大的提升空间。

在附加实验中，同学们的任务如下所示：

1、将文档标识为语义标识符（semantic identifiers）：利用层次k-means聚类为identifiers注入文旦信息先验知识，为后续解码部分提供帮助。换句话说就是让相似的文档的标识也更相似。具体操作是对每一簇数量为c的文档集中的每个文档分配一个编号（0，c-1）。如果一簇文档数量超过c，就递归地进行k-means聚类，最终得到一个树状结构每个文档都有有个路径l=[r0,r1,...rm)这个路径就是文档的标识符，相似的文档的标识符应该具有相同的前缀。

2、通过微调生成模型（T5-base），将输入的问题映射到相关的文档标识符，比如在第一步中，教育类的文章被聚类到第1类，而研究生考试类的文章在教育类别中被聚类为第5类，则该文章的文档标识符为1-5，在用户输入研究生考试类别的问题时，生成模型需要先生成1，表示该问题属于教育类问题，然后自回归式的生成5，最终准确的检索到了目标文档。

可参考文章

1、A Neural Corpus Indexer for Document Retrieval，NIPS 2022，

<https://arxiv.org/abs/2206.02743>

2、Transformer Memory as a Differentiable Search Index，NIPS 2022，

<https://arxiv.org/abs/2202.06991>

## 附录：（代码框架建议）

*# 问答系统设计与实现  
  
## Data Preparation*- 将下载下来的原数据放到`data`文件夹中  
  
*## How to Run  
### 1. 文档检索*- 对所有文章分词并去除停用词后，建立word2pid的索引，每个词对应包含这个词的pid以及相应的该词出现的次数(passage\_index.py)  
- 对问题分词之后去除停用词并去重，得到若干关键词  
- 搜索时，对每个问题中的关键词  
 - 累计当前文档命中数量（命中一个关键词加一）  
 - 累计包含该关键词的文档的分数，分数具体计算方法为IDF\*该词在当前文档出现的频率  
- 按照相似度返回排序后的文档  
 - 相似度首先考虑命中数量，再考虑分数累计  
  
```  
python passage\_index.py  
python step1\_search\_passage.py  
```  
  
*### 2. 构建问题分类器*- 使用`DMetaSoul/sbert-chinese-general-v1-distill`预训练模型加上MLP classification head训练  
  
```  
python step2\_question\_classification.py  
```  
  
  
*### 3. 候选答案句*- 将数据集随即划分为9:1的两部分作为训练集和测试集  
- 每篇文档中的正确答案句作为正样本，同一文档中的其他句子作为负样本  
- 使用预训练模型`DMetaSoul/sbert-chinese-general-v1-distill`提取句子特征以及上一步训练好的encoder提取问题的特征  
- 最终问题的embedding为二者逐点相乘的结果  
- 再将问题embedding分别与正负样本候选句子的句子特征concatenate之后得到最终表示  
- 使用InfoNCELoss优化模型  
  
```  
python split\_data.py  
python step3\_sentence\_sim.py  
```  
  
*### 4. MRC抽取*- 训练集测试集划分跟前一步一致  
- 采用边界模型的训练方式，使用`bert-base-chinese`预训练模型对每个token打两个分数，一个分数表示是否是答案起始位置，另一个表示是否为答案终止位置  
- 使用交叉熵损失函数  
  
```  
python step4\_mrc.py  
```