Arouq - 基于百科的知识搜索引擎

谢兴宇 2017011326
张晨 2017011307

2020 年 6 月

目录

1 问题描述 **3**

2 整体架构 **3**

3 知识检索 **4**
3.1 数据处理 4
3.2 引擎开发 5

4 知识问答 **6**
4.1 基于知识图谱的中文问答 6
4.2 基于神经网络的英文问答 7

5 扩展功能 **8**
5.1 自动纠错 8
5.2 实时补全 9
5.3 相关推荐 9

6 性能评价 **10**
6.1 知识搜索 10
6.2 知识问答 11
6.2.1 基于知识图谱的中文问答 11
6.2.2 基于神经网络的英文问答 11

7 案例分析 **12**
7.1 知识搜索 12

8 总结 **13**

A 参考资料 **13**
A.1 使用的工具 13
A.2 使用的数据 13
A.3 参考实现 14

1 问题描述

我们的目标是实现一个知识搜索引擎，主要包括以下内容：

1. 基于 solr 实现知识实体的搜索与排序
2. 通过知识图谱和神经网络实现简单的知识问答
3. 实现自动纠错、实时补全、相关推荐，使用户有更为良好流畅的搜索体验
4. 实现较为美观的前端页面，分别支持中英文的查询

2 整体架构

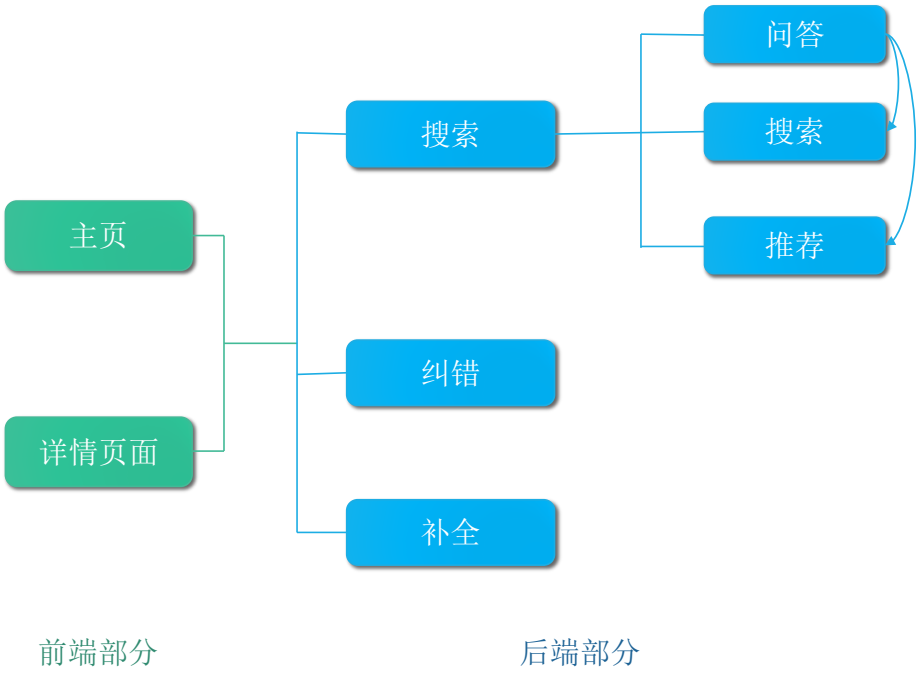

图 1: 整体架构

搜索引的整体架构如图1所示。前端使用 Vue 实现，包括一个主页（图2）和一个搜索结果详情页面（图3）。用户在两个页面的查询框的输入框中输入查询后，都会触发相同的后端动作。我们将后端分为搜索、纠错、补全三个相对独立的模块。在用户进行输入时，会适时触发补全模块。在用户输入完成后，会触发搜索模块与纠错模块。搜索模块分为问答、搜索、推荐三个功能点，其中，问答的结果会引导搜索及推荐的过程。

图 2: 首页

3 知识检索

3.1 数据处理

我们使用的数据集是 xlore，一个中英文跨语言百科知识图谱。

这里，我们将中英文分开，将 xlore 的数据重新整理，每一个实体包含以下五个域：

- 名称
- 链接
- 主体内容
- 类别（可能有多个）
- 属性（包括属性的名称和内容）

最终得到的中文数据集的大小约为 2.52 GiB，英文数据集的大小约为 1.23 GiB。

相比于我们自己写爬虫爬来的数据，xlore 的数据集的质量确实更高。但还是花了较多的时间在数据清洗上，一方面由于 xlore 对自身的格式没有说明文档，另一方面 xlore 的数据集自身依然存在一些问题，比如：

- 主体内容或属性内容中可能有实体 id 或概念 id，作为类似于 URL 的站内跳转。但有不少实体 id 或概念 id 会指向并不存在的概念和实体，这可能是由于实体本身和内容之间的更新时间不一致导致的。
- 在各种域中都可能会意外地出现一些不太正常的字符，比如 " 等。
- 有的百度百科条目会出现两遍（同名，但并不是完全相同的实体）。

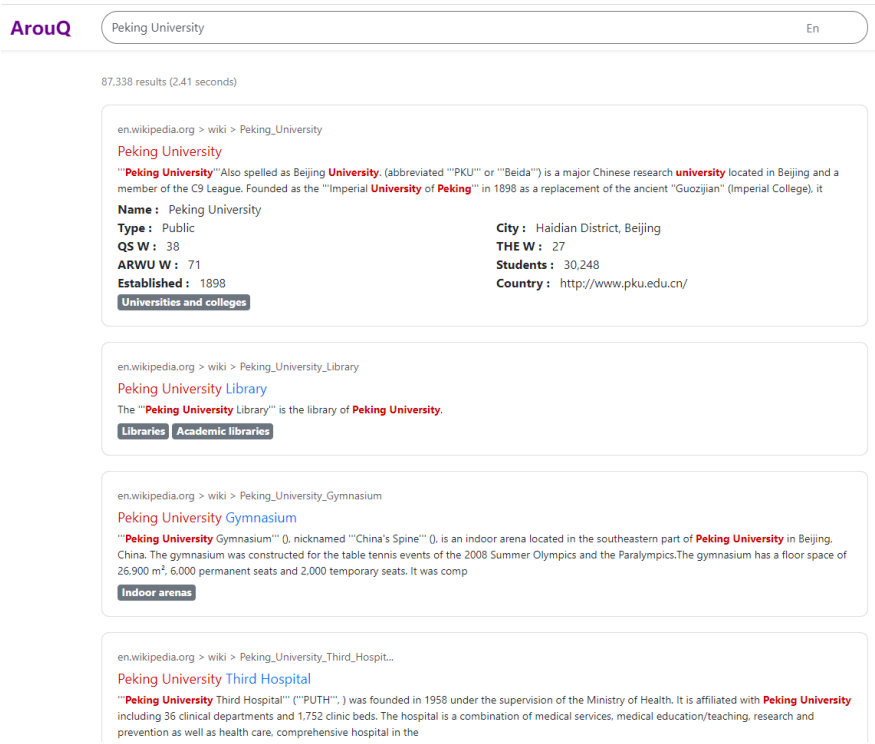

图 3: 检索结果

3.2 引擎开发

我们的知识检索基于搜索引擎框架 Apache Solr 开发。Apache Solr 基于一个更低层的搜索引擎框架 Apache Lucene，Solr 已有大约 16 年的历史，是非常成熟的企业级开源软件，主要特色是分布式易扩展的索引和查询，根据 DB-Engines Ranking，Solr 是目前排名第三的搜索引擎框架。我们使用的 Solr 的架构如图 4 所示，其包含两个节点，倒排索引会被均匀地分配给每个节点，每个节点也都能够独立地处理查询，以减小单一节点面对大量查询的压力。

图 4: SolrCloud

Solr 作搜索的核心原理是将文本分析成单词流之后建立倒排索引，倒排索引的主体结构是一棵 B 树。对于询问文本，先在 B 树上查询得到一个结果文档的集合，之后为每一个结果文档计算一个相关度（relevance），再按相关度从大到小排序输出。

相关度的计算是一个布尔模型和向量空间模型的混合模型，向量空间模型以向量来表示文档和查询，向量的每一维对应一个索引值，这一维的权重是 TF-IDF 的值。相关度计算的核心公式如下：

$$\text{relevance}(q,d) = \text{coord}(q,d) \cdot \sum_{t \in q} (\text{tf}(t \in d) \cdot \text{idf}(t^2) \cdot t.\text{getBoost}() \cdot \text{norm}(t), d) \quad (1)$$

式中各项的含义是：

- $\text{tf}(t \in d)$：词 t 在文档 d 中的词频；
- $\text{idf}(t)$：词 t 的逆文件频率；
- $t.\text{getBoost}()$：词 t 的 boost 权重；
- $\text{norms}(t, d)$：索引的 boost 和归一化的长度因子的乘积，旨在让更短的文档相关度更高；
- $\text{coord}(q, d)$：查询 q 在 d 中出现的次数。

经过一些尝试和调整，最终我们选择的 boost 参数如下表中所示：

域	名称	内容	类别	属性
boost	5	1	0.1	0.2

为了更好的理解语义信息，我们在上述模型的基础上又做了进一步的改进。不仅检索原本的查询文本，同时也对问答模块得到的答案进行检索，再将得到的检索结果相混合。

4 知识问答

对于知识问答任务，可以有基于知识图谱、基于神经网络的两种实现方式。我们在中文问答、英文问答中分别对它们进行了尝试。

4.1 基于知识图谱的中文问答

基于知识图谱的问答，核心思想是提取查询中的实体、属性，并在数据库中进行对应。我们使用 xlore 数据集中的“信息框”部分作为数据库，其描述了实体的各个属性对应的属性值。为了能够高效的在这个数据库中进行查询，我们构建了一个三列的 SQL 数据表，分别记录实体的 URI，属性的 URI 及属性值，并在实体 URI 一列上建立索引以提高查询速度。

本模块的核心流程如下

1. 使用 jieba 对查询进行分词，根据停用词表去除停用词，获取若干查询关键词。
2. 使用 xlore API 找到每个查询关键词对应的实体，xlore API 具有一定的模糊匹配能力，因而可返回与查询关键词相关的实体。
3. 在构建的数据库找到每个实体拥有的属性及对应的属性值，
4. 对得到的（实体名，属性名，属性值）三元组进行打分，选择其中得分最高的项。具体评分函数见后文。
5. 进行格式调整，去除结果中的奇怪字符。

对一个（实体名，属性名，属性值）进行打分时，我们同时考虑了实体名、属性名与查询的匹配程度，运算公式为

$$score = score_{ins} * score_{rel} \tag{2}$$

$score_{ins}$ 考虑的是实体名与查询到该实体的查询关键词的对应程度。若实体名与查询关键词为同义词，该值取 1.5，否则取 1。在这里，我们使用了 synonyms 这一近义词库，并将其评分大于 0.75 的均视为同义词。考虑实体名与查询关键词是否为同义词的原因是，xlore api 具有一定的模糊匹配能力，因而返回的实例中，有很多与查询关键词为同一概念。例如，在查询“清华大学”时，会同时返回“清华大学”与“台湾清华大学”的结果，其中，“清华大学”相关的结果应有更大的优先级。

$score_{rel}$ 则体现了属性名与查询的匹配程度。在这里，我们将查询视为原始查询，去除停用词，去除当前查询关键词后得到的字符串。匹配程度的数值为属性名的各个字在这个字符串中的总出现次数。

由于没有足够好的测试手段，算法中涉及的超参数均为估计，可能存在更为合理的取值方式。

图5 展示了一个中文问答的结果。

图 5: 问答结果（中文）

4.2 基于神经网络的英文问答

首先。我们使用 SQuAD v2.0 [2] 对 bert-base [1] 模型进行 finetune，得到一个具有问答能力的神经网络模型。SQuAD 是斯坦福大学提供的一个问答数据集，其输入为一段文本和一段问题，输出为问题答案在文本中的出现位置或返回答案在文本中未出现。我们训练的模型，在 SQuAD v2.0 的验证集上达到了 $exact\_match = 65.34\%$，$f1 = 67.81\%$ 的准确度。训练的超参数见表1。

batch size	12
learning rate	3×10^{-5}
epoch 数	2

表 1: bert finetune 的超参数

然而，正如上文所说，我们的模型在处理问答的时候，需要提供一段包含问题答案的文本，但在实际的问答任务中，并不存在这一提示文本。我们生成提示文本的方式为，获取查询关键词对应的维基百科文本，并提取出文本的首段，及与查询重叠较大的至多 10 段文本。计算文本与查询的重叠度时，我们使用了一种类似 IDF 的计算方式，以使在维基百科中出现频率较低的查询词具有较高的贡献。

此外，由于问答的准确率远比召回率重要，我们这里将答案的阈值设的较高，只有神经网络在开头位置与结尾位置的评分之和大于 3 时，才视为找到了答案。

本模块的核心流程为：

1. 分词，去停用词，获取若干查询关键词
2. 获取查询关键词对应的文本
3. 将文本中，与查询关系较大的文本作为提示文本，使用神经网络模型进行推断
4. 取评分最大的位置作为最终结果

图6 展示了一个英文问答的结果。

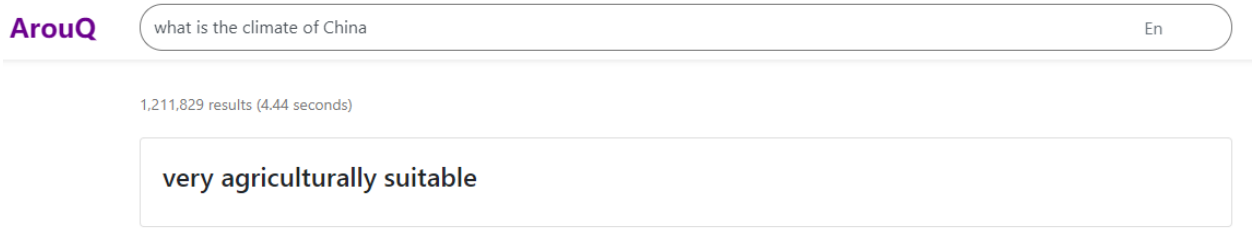

图 6: 问答结果（英文）

5 扩展功能

5.1 自动纠错

自动纠错已经有较为成熟的解决方案，我们使用 pycorrector 作为中文纠错工具，language_tool 作为英文纠错工具。然而，这些工具是为普通文本的纠错而设计的，因而会查出句子中所有不规范的位置，但在搜索时，用户是较为懒惰的，通常不会按照正式文章的书写规则输入查询，例如不会使用正确的大小写。在进行搜索引擎的自动纠错时，需要特别处理这类情况。图7展示了我们的纠错功能。

图 7: 自动纠错

5.2 实时补全

实时补全采用了基于常用词的前缀匹配方式。我们寻找到一个带有词频的中文常用词表，预处理出以各个汉字串为前缀的常用词集合。对集合中的词进行排序，留下得分最高的若干个。排序时，首先考虑的是常用词应尽量出现，其次，较长的词应更容易出现，因为自动补全的字数越多，对用户的方便程度越大。在查询时，直接在预处理结果中寻找当前查询词对应的常用词集合，进行返回。这种直接读取预处理数据的实现方式，降低了补全时的运算时间，提高了补全的实时性。

“实时”在我们的项目中的含义是指，如果用户的查询输入串的上一次改变是在恰好 1 秒之前，这时前端才会向后端发出请求，做出补全操作。预留 1 秒的延时，是为了避免在用户输入查询文本的过程中，给后端造成过大且无意义的压力。

图8展示了用户搜索过程中的实时补全。

图 8: 实时补全

5.3 相关推荐

相关推荐基于 xlore 中的 Related Instances 关系实现。在通过 xlore API 获得与查询关键词有关的实体后，仅留下与查询关键词为同义词的实体，汇总它们的 Related Instances。

在实际的查询中，用户对各个查询关键词的好奇程度不同，如在查询“清华大学的地址”的时候，用户一般更为关心“清华大学”相关的内容，而非“地址”相关的内容。但是，无论是分词器，还是 xlore 数据库，都缺乏对查询的语义级理解，因此无法分析出各个查询关键词对用户的重要程度。在这里，我们使用问答结果引导相关推荐，问答结果所对应的实体的相关实体在推荐时具有较高的权重。我们的一个推荐结果见图9。

图 9: 相关推荐

6 性能评价

6.1 知识搜索

在知识实体搜索的任务中,用户一般有明确而单一的目标实体,非常适合使用 RR(Reciprocal Rank)来衡量搜索结果。当然，这里我们需要将 RR 的定义改作：用户的目标实体的排序倒数。

这里我们构建了一个大小为 10 的查询集合，中英文各 5 个：

语言	查询 1	查询 2	查询 3	查询 4	查询 5
中文	武汉	清华现任校长	什么是强降雨	川普	中国的首都
英文	Tsinghua University	the climate of beijing	SpaceX	Trump	algebra group

查询得到的 RR 如下表所示：

语言	查询 1	查询 2	查询 3	查询 4	查询 5	MRR
中文	1.00	0.00	1.00	0.14	1.00	0.63
英文	1.0	0.00	1.00	0.08	0.33	0.48

从测试结果来看，中文的 MRR 要比英文更高。这可能是由于中文的数据集更大，包含了百度百科和中文维基，而英文的数据集仅仅只有英文维基。当然，由于我们的测试集很小，得到的 MRR 的说服力实际上并不是很强。

6.2 知识问答

在日常交流时，人们一般采用疑问句进行提问，如“清华大学在哪里”，而在使用搜索引擎时，人们则倾向于使用陈述句，如“清华大学的位置”。在下文的讨论中，我们将上述两种提问方式分别称为“疑问类”和“陈述类”。我们的两种问答实现方式在这两类提问中的表现是不同的。

基于知识图谱的问答，在陈述类问题上具有较好的表现。因为用户在进行陈述类提问时，会显式提供所需的实体及关系，我们只需要将它们提取出来，并在数据库内进行查找。然而，这种回答问题的方式缺乏对于提问的理解，因而无法知道疑问词和关系的对应方式，因而无法回答这类问题。

基于神经网络的问答，则更适用于疑问类问题。其原因是我们只寻找到疑问类问题的训练数据，因而仅用该类数据进行训练。我们认为如果能够获取高质量的陈述类问题训练数据，神经网络也将有回答陈述类问题的能力。

6.2.1 基于知识图谱的中文问答

我们选取了 5 个问题，其中 3 个陈述类问题，2 个疑问类问题，详见表2。

编号	类型	是否正确	问题	回答
1	陈述类	✗	清华大学的位置	20 个学院、57 个系
2	陈述类	✓	清华大学的地址	北京市海淀区清华大学
3	陈述类	✓	北京的气候	温带季风性气候
4	疑问类	✗	清华大学在哪里	
5	疑问类	✓	清华大学有哪些知名校友	习近平，胡锦涛，杨振宁，邓稼先等

表 2: 中文问答的结果

对于 2、3 两个问答，我们正确定位了实体及关系，得到了正确的答案。问答 5 尽管为疑问类问答，其提问中完整包含了实体“清华大学”与关系“知名校友”，因而也能够正确回答。“清华大学的位置”的错误回答来源于（清华大学，院系设置，20 个学院，57 个系）这一三元组。由于缺乏语义级理解，我们在进行属性的匹配时，只能采用按字的匹配方式，“院系设置”与“位置”均含有“置”字，因而被视为一个候选答案。再加上实体“清华大学”没有“位置”这一关系，上述三元组便成为了最优答案。无法回答问答 4 的原因，更多的在于无法理解“在哪里”的含义，因而无法寻找对应的关系。

6.2.2 基于神经网络的英文问答

我们选取了 5 个问题，其中 2 个陈述类问题，3 个疑问类问题，详见表3。

为了更好的呈现搜索过程，我们在表4内提供了各查询结果对应的维基百科原文。

编号	类型	问题	回答	是否正确
1	陈述类	✗	the location of Tsinghua University	
2	陈述类	✓	the president of America	Donald Trump
3	疑问类	✓	What is the climate of China	very agriculturally suitable
4	疑问类	✓	What is the earliest form of football	cuju
5	疑问类	✗	Who won world war II	president calderon

表 3: 英文问答的结果

编号	关键词	原文
2	America	Republican **Donald Trump**, the winner of the 2016 presidential election, is serving as the 45th president of the United States.
3	China	China has a **very agriculturally suitable** climate and has been the largest producer of rice, wheat, tomatoes, brinjal, grapes, water melon, spinach in the world.
4	football	The Chinese competitive game **cuju** (蹴鞠), as stated by FIFA, is the earliest form of football for which there is scientific evidence and appears in a military manual dated to the second and third centuries BC.
5	war	The Mexican Drug War, with estimated casualties of 40,000 since December 2006, has recently faced fundamental opposition. In 2011, the movement for peace and justice has started a popular middle-class movement against the war. It won the recognition of **President Calderon**, who began the war.

表 4: 结果对应的维基百科原文

对于正确回答的问答 234，模型均成功的在非常复杂的上下文中，定位到了准确的答案。

对于问答 1，模型实际上得到了 “Beijing” 这一正确答案，但是评分仅有 2.74，低于 3 这一阈值，因而被忽略。我们进一步查询了”Where is Tsinghua University”，发现模型得到了”Beijing” 这一回答的同时，给出了 4.89 的评分。由此可见，模型对于回答疑问类问题更有信心。

对于问答 5，回答错误的核心原因是我们在词级别，而非词组级别进行分词，因而仅查询了 war 的维基百科，而未查询 world war ii 的维基百科。

7 案例分析

7.1 知识搜索

搜索 “北京” 得到的前 10 个结果：

1. 汪峰的歌曲《北京北京》
2. 邓紫棋翻唱的《北京北京》

3. 汪峰的歌曲《北京，北京》
4. 冯唐的著作《北京北京》
5. 歌曲《北京北京我爱北京》
6. 郝云的歌曲《北京北京》
7. 宁城（古称北京）
8. 北京市
9. 北京市
10. 魏郡（北宋时期一行政区划，也称“北京”）

当用户搜索“北京”时，他最有可能想要的搜索是现代的北京市，这在我们得到的前 10 个结果之中，但其出现了两遍，这是因为 xlore 数据集不干净，有时会有一堆同名的百度百科条目；且北京市并没有被排在首位。从这个排名可以看出我们相关度算法的缺陷，当查询词“北京”在实体名字中被重复两遍后，其 tf 和 coord 都会很高，所以会算出很高的相关度；宁城被排在现代北京城之前，是由于其条目很短，$\mathrm{norm}(t, d)$ 便会很高。

8 总结

这次大作业，让我们有机会从零开始一步步实现了一个较为完整的搜索引擎，对搜索引擎的各部分都有了更加深入的认识。感谢在作业完成过程中给予我们帮助的同学，以及老师和助教一学期的辛苦付出，尤其感谢课程组提供的服务器和帮我们管理服务器的张助教！

A 参考资料

A.1 使用的工具

- Solr
- jieba 分词
- bert 的实现
- 中文纠错
- 英文纠错

A.2 使用的数据

- xlore
- SQuAD v2.0

- 停用词表
- 词频表

A.3 参考实现

- SearchTHU
- TaiqiangSearch

参考文献

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[2] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text, 2016.