互联网数据挖掘大作业报告

中文问答系统

周洋 樊乃嘉 李恬

｛zhou.yang, fannaijia, 1400012966｝@pku.edu.cn

**一、背景**

为了加快用户获得信息的速度，提升获取信息的准确性，问答系统一直都受到研究者的关注。而中文问答系统遇到的困难比英文多的多。中文首先需要分词，而且中文的句法、语法相对于英文更加松散、没有规律。我们分别利用中文维基百科的内容和从百度知道上爬取的内容，实现了一个简单的问答系统，取得了不错的效果。

**二、架构和方法综述**

**1、总体架构**

我们的问答系统主要分为三个部分：

1. 问题处理

离线版本中，对获取的中文wiki的xml格式的语料进行解析，经过规范化、繁体转简体等操作之后，形成了约145万个带标题的文件作为段落检索的输入。每个文件包含了某一方面的知识。同时，采用模版匹配和人工规则相结合的方法，确定每个问题的答案的类型。

1. 段落检索

对每个问题，缩小候选文档集的范围。离线版本中，利用Lucene为所有文件建立倒排索引并对每个问题返回top-k的相关文档；在线版本中，从百度知道和百度上爬取前若干个回答的段落。

1. 答案提取

对每个问题，从候选文档集（句子集合）中摘取最终答案。首先利用问题重写进行模版匹配，基于关键词频率进一步摘取相关句子之后，利用word2vec训练词语相似度，结合答案类型以及候选答案和问题关键词的相似度抽取答案。其中还涉及到一些特定问题的特判。

**2、关键方法**

1. 问题关键词提取

经过尝试Stanford NLP、结巴分词、中科大分词、FudanNLP、HanLP等开源软件之后，最终选择了HanLP。著名的结巴分词不支持命名实体识别，Stanford NLP对于中文的支持和兼容性不是很好。

1. 段落检索

离线版本中，首先人工将全部的wiki中文语料分割成约145万个文件，然后采用Apache Lucene建倒排索引，直接将每个问题作为query输入，即可输出前若干个回答这个问题的文档。

在线版本中，我们使用python的urllib2 模块来获得指定URL的html内容（通过该model以及合适的URL实现自动调用主流搜索引擎），并使用一个已有的html/xml内容解释器Beautiful Soup 包提取爬取的网页的snippet。

为了防止由于在一段时间内对某一搜索站点访问次数过多而导致IP被该站点封锁，我们预先定义了12个agent信息，每一个调用搜索引擎的时候都随机选择一个agent信息，并将该agent信息加入到URL中，从而有效地防止IP被该站点封锁。但是即使每一次都选取了随机的agent信息，仍然有一些搜索引擎无法使用程序调用，比如google就对反爬虫做的非常严密，我们尝试了各种方法也没能成功自动调用google搜索引擎。期间，我们还尝试使用了google提供给开发人员的搜索引擎API，但是后来发现该API只能支持一个账户每天调用1000次，根本不能满足我们的需求（7000个问题，每一个问题可能会搜索多次）。除此之外，Bing搜索也提供了相应的搜索API，但是微软官方已经停止维护改API，所以各种借口信息不是很明确，最终放弃使用。

最后，在技术难度和效果评测的双重前提下，我们使用python爬取了百度和百度知道各前10条的搜索结果，并提取了相应的snippet。在实际实验中，我们发现：使用百度进行搜索时采用问题的关键词比较好，而使用百度知道进行搜索是使用整个问题比较好。

这样开放测试中每一个问题都有约20条snippet作为原始数据。

1. 答案提取

开放测试和封闭测试的答案提取过程比较类似，每一个问题的答案抽取流程如下图所示：

QASystem.pdf

a）抽取备选句子

首先是从原始数据中抽取K个sentence，对open测试来说，这里的原始数据就是通过crawl爬取的20条snippet（最终测试表明只使用从百度知道获取的10条snippet效果更好）对close测试来说，这里的原始数据就是从wiki中使用lucence找到的与问题比较相关的一个相对较小的句子集合。目前抽取K个sentence的方法也比较简单：计算出某特定问题对应的原始数据中的每一个句子能够匹配到的问题关键词的个数，取包含问题关键词个数最多的K个sentence。在200个sample question测试集上上(open)，通过这种方法得到的K个sentence中，包含了141个问题的答案。

b）抽取备选词

然后是从该K个sentence中抽取出canditate\_word集合，具体方法是将每一个问题得到的K个sentence做分词和词性标注（使用工具Hanlp ），保留哪些非stopword、不属于stoptype并且符合该问题答案类型的word，这些word成为该问题的candidate\_word集合。在200个sample question测试集上上(open)，通过这种方法得到的candidate\_word集合中，包含了101个问题的答案。

由于答案的词性标注本身就有误差，和使用的词性标注字典也有其精度限制，比如许多答案都会涉及不同的数词或数量词。年份（如2016）和一般的数字都是同一词性，因此，我们在抽取candidate\_word后，在模板匹配部分，再次对其进行一次滤杂。通过问题中的关键词，如年份、名次类问题，就结合sentence对candidate直接筛选出年份数字或名次数字；对其他答案类型的问题，粗粒度地筛去混杂其中的年份数字。这部分的进阶处理被包含在了template\_find\_answer(…)函数中。

c）选定最终答案

最后是从该candidate\_word集合中抽取出one answer，这里使用了google的一个开源项目Word2vec ，训练集为从网上获取的一些语料库，该工具通过将word转化为vector，从而进行比较精确地进行word之间相似度的计算。我们使用这个工具对candidate\_word集合中的每一个word进行打分，打分具体规则是：对candidate\_word集合中的每一个word计算它和问题所有关键词之间的相似度之和，该sum作为该candidate\_word的得分（grade or point），得分最高的作为最终该问题的答案。在实际测试中发现，这种打分方式非常容易把问题中的关键词选为最终答案，而一般问题关键词都不会成为答案，所以我们直接从candidate\_word集合中选取得分最高的且不为问题关键词的word作为最终答案。在200个sample question测试集上(open)，通过这种方法得到的最终答案中，回答正确了44个问题。

d）利用模板优化性能

问答系统的一个突出特征是可以通过问题中的特定句式，在语料库中用模板匹配得到较为准确的答案。因此在抽取one answer的阶段，我们还写了一些规则。

例如，专门针对问诗句“下一句”的问题，我们通过检测问题中是否含有关键词“下一句”来识别该种问题，一旦识别出来，便可以从问题中提取诗句上半句，然后在K个sentence中找下半句的方法，轻松获得准确的答案。并且，一旦检测出该问题为问诗句“下一句”这种类型的问题，便不需要使用Word2vec对candidate\_word进行打分排序，可以通过规则直接返回答案。

特别地，针对开放测试，我们的主要数据来源是百度知道。观察备选语料我们发现，有许多爬下来的语句直接包含答案，例如“答：184”，“答：明朝”等等。因此，我们尝试对这些“高能”短句直接识别，并提取答案。我们对语料库中的句子先做分词，分出的词数小于4时，我们就认为该语料直接包含答案。由于答案形式比较规整，我们直接粗放的认为最后一个分标点词就是答案。我们把这些答案收集起来，排序、计数，我们认为超过整个集合半数的就可以认为是正确答案。这部分处理包装为askZhidao(int ques\_id)函数，在template\_get\_answer(…)中调用。实验证明，这种针对百度知道语料的简单的处理，有着不错的效果。

**三、相关困难**

**1、数据处理部分**

（1）预处理

解析xml的时候，我首先采用了java中的DOM库，但是这种方法要求xml文件要全部读入内存，承受不了。如果人为拆分成小文件，比较麻烦；5.6g的文件，通常DOM的索引等overhead需要内存过大， 实验证明，50g的内存也不够。经过一段时间人工的探索，后来发现使用已有的wiki中文的extractor工具效果很好。

（2）建立倒排索引

工程上的问题较多，尤其是Lucene中使用了中文分词的插件IKAnalyzer，Lucene和IKAnalyzer之间的兼容性不是很好。之前下的最新版本的Lucene，相对于最新版本的IKAnalyzer来讲还是过高，于是重新改版本，随之而来的就是基本上写好的API都要再改一遍。IKAnalyzer之前的旧版本也不行，使用过程中换成了新版本。经过探索，充分了解了Lucene的build index, add document, parse query, search top-k 的整个处理流程，提升了一点工程能力。

（3）获得在线文档

爬虫的稳定性不好，需要多次尝试。

**2、答案提取部分**

（1）词语相似度的衡量方法。HanLP的工具包中自带词语距离的计算，但是测试之后发现数值过大，不是很准确，于是采用了word2vec的训练的方法。

（2）中文词法和语法的复杂性。分词和关键词提取的时候，语义的精确度已经丧失一部分了，我们整体的框架是基于统计知识的，所以在语义方面考虑的较少，因此最后的准确度从数值上来讲不是很高。

（3）IDE中，中文编码的问题。需要查阅资料和自己手动改一下IDE的配置。

**3、其他**

（1）代码的管理和共享。

团队协作时，初期使用有道云协作，但是为了方便，后来都在微信中直接传文件和代码，导致需要看代码/文件的时候都要翻聊天记录，而且所有人的代码版本不同，以后可以用git或者svn管理一下。

（2）整个开发过程中，没有建立规范的文档，而是在即时通讯软件中交流，导致关于接口、文件路径、文件格式等需要反复沟通，影响效率。

**四、未来工作**

1、使用贝叶斯概率模型、神经网络训练答案类型；

2、进一步提取特征，并使用决策树（随机森林）等更强的算法学习答案。

**五、其他**

1、编译运行环境

IDE: IntelliJ IDEA, Eclipse

语言：Java, python

2、测试集的结果

按照我们的算法，在200个问题上，离线测试的正确率达到了15％，在线测试的正确率达到了22%。

3、作者信息及分工情况

周洋：负责爬虫的编写、snippet的抽取，和答案提取。

樊乃嘉：负责后期代码的特定模板优化，和选词优化。

李恬：负责前期离线中文wiki数据预处理、问题处理和离线的段落检索。

参考资料

［1］<http://licstar.net/archives/262>

［2］<https://docs.python.org/2/library/urllib2.html>

［3］<https://www.crummy.com/software/BeautifulSoup/bs4/doc.zh/>

［4］<http://hanlp.linrunsoft.com>

［5］<https://code.google.com/archive/p/word2vec/>

［6］基于维基百科的智能答题机器人的设计与实现，陈强