班级：计科1503

学号：2015014357

北京化工大学

毕业设计（论文）

**题目 领域问答系统的设计与实现**

**专 业 计算机科学与技术**

**学 生 黄奕达**

**指导教师 万静**

2019**年**5**月**30**日**

**诚信声明**

本人声明：

我所呈交的本科毕业设计论文是本人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢中所罗列的内容以外，论文中不包含其他人已经发表或撰写过的研究成果。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。本人完全意识到本声明的法律结果由本人承担。

申请学位论文与资料若有不实之处，本人承担一切相关责任。

本人签名： 日期： 年 月 日

本科生毕业设计（论文）任务书

设计（论文）题目： 领域问答系统设计与实现

学院： 信息科学与技术学院 专业： 计算机科学与技术 班级：计科1503班

学生： 黄奕达 指导教师（含职称）： 万静(副教授) 专业负责人： 高敬阳

1．设计（论文）的主要任务及目标

（1）通过python爬虫构建一个医疗领域问答数据集，医疗领域知识库，实现系统知识库的构建，为后续词向量模型的训练和seq2seq问答模型的训练做好数据准备。并且建立完备良好的数据集，来确保模型效果。

（2）查阅关于知识图谱与问答系统的文献，了解知识图谱在问答系统的应用。

（3）查阅机器学习的文献，了解机器学习在自然语言处理上的应用

（4）利用所掌握的软件设计方法，设计和实现领域问答系统。

2．设计（论文）的基本要求和内容

（1）查阅领域问答系统，机器学习和知识图谱的相关期刊以及相关文献。至少查阅15篇以上的参考文献，撰写文献综述。翻译一篇与课题相关的外文文献。

（2）研究问答系统中问题分类的方法。实现对于用户输入问题的相应分类功能。

（3）研究问答系统中问句语义分析的方法。实现一个基于encoder-decoder的seq2seq模型，来对于输入的特定问题语句生成相应的回答。

（4）研究问答系统中答案抽取的方法，对于不同类别的答案，使用不同的图算法，选择核心链并生成答案。

（5）为该问答系统设计一个可视化方案，使用springboot框架，amazeui前端框架，实现一个问答系统。

3．主要参考文献

[1] AUER, S. R.; BIZER, C.; KOBILAROV, G.; LEHMANN, J.; CYGANIAK, R.; IVES Z. DBpedia: A Nucleus for a Web of Open Data[C]. //Semantic Web, International Semantic Web Conference, Asian Semantic Web Conference, Iswc + Aswc, Busan, Korea, November. DBLP, 2007: 722-735.

[2] Bollacker K D , Evans C , Paritosh P , et al. Freebase: A collaboratively created graph database for structuring human knowledge[C]// Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008. ACM, 2008: 1247-1249.

4．进度安排

|  |  |  |
| --- | --- | --- |
|  | **设计（论文）各阶段名称** | **起 止 日 期** |
| **1** | 查阅检索相关科技资料，写出文献综述。 | 1月9日——1月30日 |
| **2** | 学习知识图谱的知识，设计网页、建立数据集。 | 2月1日——2月28日 |
| **3** | 完善系统的功能与算法的准确性。 | 3月1日——3月31日 |
| **4** | 改善程序可靠性，最终完成系统的设计工作。 | 4月1日——4月30日 |
| **5** | 撰写符合学校相关规定毕业论文。 | 5月1日——6月1日 |

**摘 要**

在信息量爆炸的当今时代，从大量的数据中提取出重要的信息至关重要，传统的搜索引擎通过对于用户输入的问题，构建回答列表，用户对于回答列表中的答案逐一挑选，才能找到自己真正想要的答案，而这种多个步骤的操作，使得查询时间较长，查询需要逐一查看。针对于这个现状，基于知识图谱的领域问答系统为解决用户的问题提供了新的思路。本文设计并实现了一个中文医疗领域的问答系统。将这个问答系统的实现主要分成了四个部分的算法设计与实现，分别是知识库构建，问题分类，问句语义分析，答案抽取。并且对于问答功能的体现，提供了可视化的解决方案，使用经典的webapp框架Springboot，结合html，js，css等前端页面实现技术，为用户使用系统提供了方便快捷的操作方式。 在知识库构建部分，使用python爬虫，在医疗问答网站上对实体进行爬取，通过爬取相应实体的百度百科确定实体之间的关系，并且爬取医疗问答网站上的医疗问答对，作为模型训练的语料。在问题分类部分，使用卡方方法抽取关键字，对每一个关键字，计算他的tf-idf值，从而得到每一个问句的向量表示，再通过SVM分类器进行分类。在问句语义分析部分，通过使用在医疗网站上爬取到的医疗问答对，进行词向量的训练，在这之后，进行seq2seq模型的训练，使用seq2seq模型，生成相应的回答模板。对于答案抽取部分，根据问句中识别出的实体构建查询图，针对于不同的问题类别，选择不同的图查找算法，选择核心链，并且生成答案。

**关键词：**知识图谱，问答系统，医疗领域，语义分析

**Abstract**

In the era of information explosion, it is very important to extract important information from a large number of data. Traditional search engines can find the answers they really want by building answer lists and selecting the answers one by one. This multi-step operation makes the query time longer and the query needs to be checked one by one. Look. In view of this situation, the domain question answering system based on knowledge map provides a new way to solve users'problems. This paper designs and implements a question-and-answer system in Chinese medical field. The implementation of this question answering system is mainly divided into four parts: knowledge base construction, question classification, question semantics analysis and answer extraction. And for the reflection of question and answer function, it provides a visual solution, using the classic web app framework Springboot, combined with html, js, CSS and other front-end page implementation technology, providing users with a convenient and fast way to use the system. In the part of knowledge base construction, Python crawler is used to crawl entities on the medical Q&A website. The relationship between entities is determined by crawling Baidu Encyclopedia of corresponding entities, and medical Q&A pairs on the medical Q&A website are crawled as corpus for model training. In the part of question classification, we use chi-square method to extract keywords, calculate the TF-IDF value of each keyword, and get the vector representation of each question, and then classify it by SVM classifier. In the part of question semantic analysis, we train the word vector by using the medical question-and-answer pairs which are crawled on the medical website. After that, we train the seq2seq model and use the seq2seq model to generate the corresponding answer template. For the part of answer extraction, query graphs are constructed according to the entities identified in the questions. For different question categories, different graph search algorithms are selected, core chains are selected, and answers are generated.

**Key words：**Knowledge Map, Question Answering System, Medical field , Semantic Analysis

**目录**

[第1章 绪论 1](#_Toc10491752)

[第1.1节 研究背景和意义 1](#_Toc10491753)

[第1.2节 国内外研究现状 2](#_Toc10491754)

[第1.3节 本文的主要研究内容和结构安排 5](#_Toc10491755)

[1.3.1 本文的主要研究内容 5](#_Toc10491756)

[1.3.2 论文的结构和安排 5](#_Toc10491757)

[第1.4节 本章小结 6](#_Toc10491758)

[第2章 需求分析 7](#_Toc10491759)

[第2.1节 系统流程 7](#_Toc10491760)

[第2.2节 核心流程 7](#_Toc10491761)

[2.2.1 知识图谱构建 8](#_Toc10491762)

[2.2.2 问题分类 8](#_Toc10491763)

[2.2.3问句语义分析 8](#_Toc10491764)

[2.2.4答案抽取 8](#_Toc10491765)

[第2.3节 本章小结 9](#_Toc10491766)

[第3章 问答系统设计 10](#_Toc10491767)

[第3.1节 系统结构 10](#_Toc10491768)

[第3.2节 知识图谱构建方案设计 12](#_Toc10491769)

[3.2.1 医疗领域知识图谱数据分析 12](#_Toc10491770)

[3.2.2 数据获取 13](#_Toc10491771)

[3.2.3 知识图谱构建 14](#_Toc10491772)

[3.2.4 知识图谱查询 17](#_Toc10491773)

[第3.3节 问题分类方案设计 18](#_Toc10491774)

[3.3.1 问句关键字提取 18](#_Toc10491775)

[3.3.2 问句向量生成 19](#_Toc10491776)

[3.3.3 基于SVM的问句分类 20](#_Toc10491777)

[第3.4节 问句语义分析方案设计 21](#_Toc10491778)

[3.4.1 基于深度学习的语义分析 21](#_Toc10491779)

[3.4.1.1 词向量训练以及数据准备 22](#_Toc10491780)

[3.4.1.2 encoder-decoder模型 24](#_Toc10491781)

[3.4.1.3 模型训练 25](#_Toc10491782)

[3.4.1.4 模型抽取结果以及评测方法 26](#_Toc10491783)

[第3.5节 答案抽取方案设计 27](#_Toc10491784)

[3.5.1 查询图的构建 27](#_Toc10491785)

[3.5.2 选择核心链 28](#_Toc10491786)

[第3.6节 本章小结 28](#_Toc10491787)

[第4章 系统实现 30](#_Toc10491788)

[第4.1节 系统框架概述 30](#_Toc10491789)

[第4.2节 应用实例 30](#_Toc10491790)

[4.2.1 问答系统 31](#_Toc10491791)

[第4.3节 本章小结 33](#_Toc10491792)

[结 论 34](#_Toc10491793)

[参考文献 36](#_Toc10491794)

[致 谢 37](#_Toc10491795)

# 第1章 绪论

## 第1.1节 研究背景和意义

随着互联网技术的发展与信息资源的爆炸性增长，传统的搜索引擎技术的劣势愈发明显。根据用户的查询而产生的快速排序网页，虽然有着较高的信息检索效率，但是并不意味着能够为用户提供快捷准确的信息资源。当用户查询信息时，要先从查询内容中提取出关键词，但是大部分用户难以从自己的问题中提取出真正准确有效的关键词，因此依照用户给出的关键词，对于搜索引擎反馈的大量结果，还需要进行人工排查和筛选，从而在面对当今爆炸增长的信息资源时，难以满足人们的需求。而知识图谱技术的出现，为解决面对大量数据的信息检索问题提供了新的思路，同时，知识图谱的这些相关特性对于问答系统也提供了新的思路，使得问答系统的准确性与实用性都有了新的提升。问答系统在很早的时候就为计算机领域所关注并研究。而以知识图谱技术为思想，构建的问答系统，则在大约二十年前就已经出现在人们的视野之中。DBPedia[1]就是一种能够为问答系统提供服务的大型知识库。同样的，Freebase[2]也是一个支持基于知识图谱的领域问答系统的大型知识库之一。这些大型知识库，帮助了人们在实现领域问答系统时，构建知识图谱，在分析完问题之后，一个强大的知识库，也便于人们就分析得到的一系列信息，构建查询图，获得更加精确的答案。而对于从知识库查询知识的操作，与人们从数据库中查询信息的方法类似，都是基于查询语句，而对于知识库，查询语句就变成了SPARQL或者CQL等。对于知识图谱的构建环节，则涉及到了相应的图数据结构算法，如在选择核心链时，就会用到最短路径算法，在构建知识图谱时，也会用到相应的图操作。现如今，随着人们对于深度学习的深入研究，其在自然语言处理领域的应用日益丰富，不论是各种分类器，还是生成序列的模型，都可以更加有效地实现问答系统的功能。因此，本文将设计并实现一个医疗领域问答系统，探索领域问答系统的整个实现流程，为实现问答系统提供自己探索的方案与经验。

## 第1.2节 国内外研究现状

对于基于知识图谱的领域问答系统，接下来将针对于其实现的流程，给出其当前研究现状。

1. 知识图谱的构建

知识图谱的构建主要包括实体抽取和关系抽取两部分。对于知识图谱的构建，各种信息处理的方法有所不同，主要区别在于处理的是结构化数据还是非结构化数据。结构化数据可对其直接进行处理，而非结构化数据则需要人们对其再加工，此处我们主要讨论的是非结构化数据的信息抽取。知识图谱是Google公司为了增强搜索引擎的能力，以增加搜索结果的匹配程度和匹配范围为目的，在2012年提出的新理念。知识图谱实际上是在描述各种实体，关系以及由他们形成的概念，将这些信息以图结构的形式描绘出来，形成的图结构的结点代表着相对应的实体，边对应着连接着的实体之间的关系，而对于每一个实体，他还有相应的属性。知识图谱构建指的是从文本数据之中获取知识，包括实体与他们的关系，并按照得到的知识，构建知识图谱。而从互联网上得到的数据，大多是非结构化的数据，这些非结构化数据对于构建知识图谱来说并不足够，将非结构化数据进行处理，从中得到结构化数据，这才是构建知识图谱所用的最终数据。构建一个数据充足且精确的知识图谱对于领域问答系统的实现是非常重要的，他是系统回答的基石。

实体抽取又叫做命名实体识别（named entity recognition, NER），目的是从文本中找出专有名词或者所需要的词。刘峤[3]等人将这一技术分为三类：基于规则，统计机器学习方法，面向开放域。早期研究主要采用基于规则的方法，但是该方法繁琐且有局限，难以适应数据的变化。随后人们将目光放在了统计机器学习方法这一方向上，主要包括RANIBER[4]所述的隐马尔可夫模型（Hidden Markov Model,HMM），HARRIES[5]提出的条件随机场（Condition Random Field，CRF）和LAPTEV[6]所提出的支持向量机（Support Vector Machine，SVM）等方法。Cheng等人[7]提出的BI-LSTM-CRF模型结合了条件随机场与LSTM神经网络，在BI-LSTM层后添加了CRF层，设定矩阵为BI-LSTM的输出，进行训练，该方法结合了上下文的特征并且考虑了句子前后的信息，提高了准确率。基于统计机器学习的方法往往与基于规则的方法结合起来使用，在基于规则的特征构建基础之下，统计机器学习方法可以发现新的规则，并且不需要研究者的干预，会产生良好的效果，这种方法直到现在也在被广泛使用。面向开放域的实体抽取不再限制于单一领域的内容，而是研究和解决面向全互联网的信息抽取。杨等人[8]在文章中指出，面向开放域的问答系统，其问题的类别划分极为重要，对结果的正确性影响最大，问题的分类体系则大多是提前由人工制定好，因此在真正进行计算的时候往往会对结果有所限制，KEHLENBECK[9]提出的bootstrapping方法可以将少量已经限定的信息，作为内容的全部，对其他未进行限定的内容进行处理，来为少量的信息做出补充，此方法常用在构建数据集时的，通过再抽样来扩充样本。

关系抽取的目的是为了得到文本中零散实体之间的对应关系，从而将实体连接起来形成知识结构。早期的关系抽取是对应基于规则的实体抽取，这种关系抽取往往需要人工干预，并且由于技术原因，早期的规则并不全面，这就导致了关系抽取的不准确。随着统计机器学习的发展，在关系抽取方面又有了新的尝试，DU[10]提出了卷积神经网络，杨[11]使用了卷积神经网络[10]，对实体进行处理，再用bootstrapping思想，依照少量的标注样本生成足够多的数据，这种方法可以大量减少人工工作量，并且在训练的时候不需要人力投入。在神经网络的基础上加上注意力机制的思想，可以对于问题不同的部分有所侧重，识别出主要的特征，来增加解决问题的效率。史[12]用双向长短时记忆网络表示问句，用注意力机制获取特征并进行分类，然而该方法仍会受到问句哨声影响。

1. 问句语义分析

问题语义分析主要分为以下三种方法[11]：语义解析法，信息抽取法，向量建模法。

语义解析法是通过将问句转化为某种逻辑形式，使得知识库可以理解问题，并推理，构建相应的查询语句找出答案。其中较为出名的是Berant等人[13]的方法，他们用语义解析法得到了很好的结果，取得了成绩，但是其独创的进行语法树节点合并的桥接方法，最终会生成大量的语法树，复杂度很高，因此需要更对的人力投入，对其进行训练，方可达到效果，其主要思想是通过语义解析法，将句子的语法成分提取出来，其中包括主语，谓语和宾语等语法元素，之后对于语法元素和语法结构进行分析，最后生成答案。

信息抽取法通过提取问题中的实体，依靠在知识库的查询，构建以该实体节点为中心的知识库子图，对于这个图，其中的每一个节点都是候选答案，对于这些候选答案，依照相应的图算法和比较算法进行分析，最终得到相应的答案。通过对于问题的分析得到问题和候选答案的特征，用分类器筛选从而得出结果。Yao等人[14]的文章通过对问题构建语法依存树，将树转化为问题图并进行分类，最终得到了结果，其在F1-score下相对于其他部分方法有所提升，但是该文章中构建的特征比较稀疏，并且涉及到了许多先验知识，因此如果用深度学习的方法，会对这些缺点有所弥补。

向量建模法是将问题与答案向量化，构建相应的模型找出问题中的关键词，通过相应的计算方法来计算问题与答案的匹配程度，从中选出最匹配的答案作为结果，与这个方法息息相关的是词向量的应用。Bordes等人[15]，用Embedding空间对于知识库中的实体与关系映射与问题中的关键词进行处理，用*S*(*q*,*a*)=*f*(*q*)⊤*g*(*a*)作为计算匹配程度的函数来得出结果，该方法的效果在当时是最好的结果，而且不需要借助额外的结构或是人工操作，是很好的方法。Zhang[16]等人，在实现智能问答系统的过程之中，采用了Skip-gram模型进行训练词向量，这种模型适用于数据量较小的情况，相应的CBOW模型则对应于数据量较大的情况。Devlin[17]等人，通过构建一个深层模型，基于transform架构，采用无监督学习的训练方式，来生成文字的语义特征向量，这个模型便是Bert。Bert模型通过对于所有层的前后文进行分析，共同调节来训练深层双向的模型，并且通过另外的一个输出层对预训练得到的Bert模型进行微调，来提供文字的向量表示，其在多项自然语言处理任务上都取得了良好的成果，是今年最热门的方法之一。

1. 知识图谱的存储

由于知识图谱的表示存储方式不同，其存储的使用工具也不尽相同，知识图谱的存储方式中主要的两个是RDF格式和图格式。由xia[18]等人所述，RDF格式往往与三元组相关联，存储内容为主语谓语宾语，是一个句子的主要元素。对于基于关系型数据库的存储方法，常用的有Mysql，Oracle，在支持RDF的数据库中，Jena，Virtuoso为人们所广泛使用。而对于图结构的知识图谱，康[19]做了对于Neo4j的研究，Neo4j作为世界范围内使用量最高的图数据库，其有良好的并发性能，有着高速的图形算法，依赖云计算环境的分布式存储，支持语言丰富，提供了良好的服务。在大量的知识图谱应用的实现中，都能够看到Neo4j的影子。

## 第1.3节 本文的主要研究内容和结构安排

### 1.3.1 本文的主要研究内容

本文将设计并实现一个领域问答系统，通过python爬虫爬取预料网站上的药品，疾病，症状等实体信息，实体关系，医疗问答对，构建本系统的知识库，和相应的问答对语料。利用机器学习，深度学习等方法对各个步骤加以探索并实现，并且对于系统功能的实现，提供了可视化的服务。主要研究内容如下：

(1)构建知识库。使用python爬虫在9939，春雨医生，好大夫等网站上爬取实体，再将实体之间通过百度百科上的详细信息进行匹配，判断实体之间的关系。处理过后，以RDF文件格式存储知识图谱。另外，爬取9939，春雨医生网站上的问答对，作为词向量训练，问句语义分析模型训练的基本数据集。

(2)设计和实现基于知识图谱的领域问答系统。包含问题分类模块，语义分析模块，，答案抽取模块，可视化展示模块。分析并实现问题的分类，将问题主要分为三大类别，是否类，问答类，解释类。使用卡方方法提取问句语料的关键字，生成tf-idf向量，使用SVM分类器进行分类。设计系统语义分析模块，使用tensorflow框架构建基于encoder-decoder的seq2seq模型，使其对于特定的问题输入，产生特定的回答输出，并且对于模型的效果进行相应的实验，来判断模型的效果。设计系统答案抽取模块，构建查询图，选择核心链，最终结合问题分类结果，问句语义分析结果，生成答案。对于可视化展示模块，使用springboot框架实现了一个能够实现用户与系统交互的问答平台。

### 1.3.2 论文的结构和安排

本文包括三章内容，各章节内容安排如下：

第一章是绪论，介绍了领域问答系统的研究背景与意义，国内外研究现状。

第二章是需求分析，介绍了功能实现的思路分析。

第三章是问答系统设计，介绍了对于知识库构建，问题分类，问句语义分析，答案抽取这些关键步骤的方法选择，具体效果，以及测试效果。

第四章是系统实现，介绍了该领域问答系统的系统架构，核心流程以及应用实例。

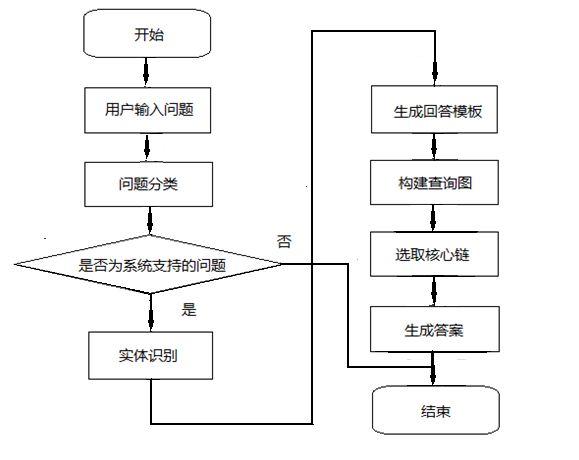
## 第1.4节 本章小结

本章首先介绍了本课题的研究背景和研究意义，之后分析了国内外的研究现状，主要包括知识图谱的构建，问句语义分析，知识图谱的存储，并且对其中方法的优缺点进行分析，最后提出本文的研究内容并安排本文的内容结构。

# 第2章 需求分析

本章对于本系统的基本设计思路进行了叙述，包括系统流程和系统结构。对于用户输入一个问句，系统对其进行分析的过程，被记录在系统流程之中。对于整个系统，起作用的各个部分的信息，则体现在系统结构部分。

## 第2.1节 系统流程



**图2-1 系统流程图**

本系统对于用户输入的问题先进行问题的分类，对于系统能够理解的问题，进行实体识别，识别出之后，对于输入的问题，传入seq2seq模型，生成回答模板。之后再通过问句中识别出的实体，构建查询图，在查询图之中，选择核心链，找出答案中需要的实体，镶嵌到答案模板之中相应的位置，生成最终的答案。

## 第2.2节 核心流程

核心流程是指从问句输入到给出答案的全部流程中的各个重要部分，包括：知识图谱构建，问题分类，问句语义分析，答案抽取。

### 2.2.1 知识图谱构建

知识图谱构建核心流程：

1. python爬虫爬取医疗网站上的实体。
2. python爬虫爬取百度百科上的实体简介，建立不同实体之间的关系。
3. 为不同的关系命名，生成三元组并且存储到RDF文件。
4. 配置apache jena框架，并且在实现功能过程中用sparql操作数据。

### 2.2.2 问题分类

问题分类核心流程：

1. 构建问题分类数据集。
2. 使用卡方方法，获取问题的关键字。
3. 使用tf-idf思想，生成问题句子对应向量。
4. 训练SVM分类器。
5. 对于用户输入的问题，进行分类。

### 2.2.3问句语义分析

问句语义分析核心流程：

1. python爬虫爬取医疗问答对。
2. 对于医疗问答对进行实体识别与实体类别标签赋予，生成训练用数据集。
3. 使用jieba分词处理数据集。
4. 使用word2vec对数据集进行训练，生成词向量。
5. 构建seq2seq模型。
6. 使用词向量与训练用数据集，对于seq2seq模型进行训练。
7. 对于用户输入的问题，生成回答的关系模板。

### 2.2.4答案抽取

答案抽取核心流程：

1. 根据问题中的实体，构建查询图。
2. 根据问题的类别，选择相应的核心链选取方式。
3. 生成最终答案。

## 第2.3节 本章小结

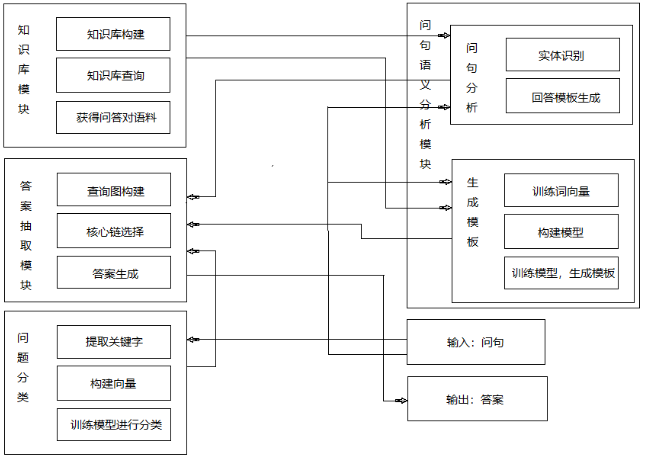
本章介绍了系统流程和系统结构，讲述了系统运行流程大体结构以及功能实现主要部分。将系统实现思路描绘了出来。

第3章 问答系统设计

本章将建立起问答系统的系统结构，并对于系统结构中核心部分的实现思路进行分析与讲解，也就是知识图谱构建，问题分类，问句语义分析，答案抽取这四个模块。

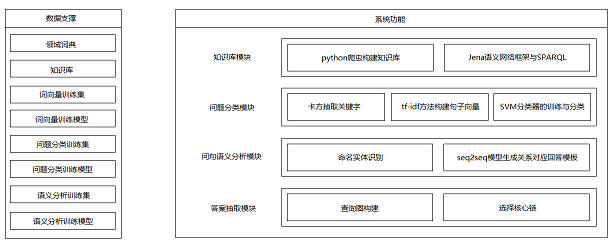
## 第3.1节 系统结构

本系统的功能实现主要由如下四个模块：知识库模块，问题分类模块，问句语义分析模块，答案抽取模块来实现。其中，知识库模块包括了系统所需的数据支持，这些数据包含在了知识库构建和获取问答对语料部分，由python爬虫获得并构建。对于知识库的查询，则在识别实体和构建查询图时提供了帮助。问题分类模块，通过提取关键字，构建句子的向量，训练分类模型，实现问题分类功能，问题的类别与答案抽取的方式紧密联系。对于用户输入的问题，问句语义分析模块，进行实体的抽取，并且将抽取过后的问句，送入生成部分，生成答案的关系模板，这一步骤，是通过seq2seq模型来实现的。生成的关系模板，与识别出的实体，一起作用于答案抽取模块，通过实体构建查询图，根据关系模版上的实体类别，使用最短路径算法查询实体，将得到的答案放入答案模板，最终生成答案。



**图3-1 系统结构图**

上述的四个模块，也都拥有相应的数据支撑。其对应的关系如图3-2。其中左侧为系统的数据支撑，包括：领域词典，知识库，词向量训练集，词向量训练模型，问题分类训练集，问题分类训练模型，语义分析训练集，语义分析训练模型。其中领域词典和知识库主要用来构建知识图谱，词向量训练集和词向量训练模型为问句语义分析模块的训练做好了铺垫。问题分类训练集，问题分类训练模型则是用SVM分类器进行相关工作的基础。语义分析训练集和语义分析训练模型，则是使用训练好的词向量与数据集，构建相应的encoder-decoder模型来进行语义分析的必要准备。



**图3-2 系统架构**

## 第3.2节 知识图谱构建方案设计

本文章使用python爬虫，对于医疗领域的网站进行爬取，以RDF文件的形式进行存储，使用jena框架和sparql对RDF图数据进行操作。

### 3.2.1 医疗领域知识图谱数据分析

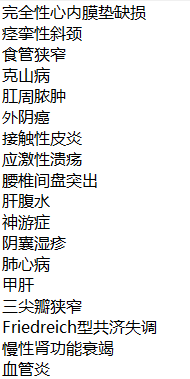
医疗领域的知识主要包括疾病，症状，药品。他们各自存在着相应的的关系，当一个药品可以治疗某一疾病，则这个药品与疾病的关系记载为“药品相关疾病”，当一个疾病有着某个症状，则这个疾病与症状的关系记载为“疾病相关症状”，当一个症状可以被一个药品所治疗，那么这个症状和药品对应的关系为“症状相关药品”，以上的关系，在知识图谱中，是实体与实体之间的单向边。

在本文章实现的医疗领域问答系统之中，其形成的知识图谱的结点，可以是“感冒”，“氧氟沙星胶囊”，等关于疾病，药品，症状的信息，而对于这些实体，他们又有自己的属性，比如对于实体“氧氟沙星胶囊”，它的价格就是它的属性之一，节点之间的边代表着他们的关系，比如连接实体“感冒”，“阿莫西林”的边所记载的关系是“疾病相关药品”。

对于知识图谱的描述以及存储，RDF图是我们的选择之一，RDF是能够描述图结构资源信息的标记语言，一个RDF文件之中可以含有很多资源的相应信息，对于每一种资源，通过语句表示的三元组的形式记载这种资源的一条信息。而对于RDF文件，独特的数据类型需要特有的框架来对他进行操作。

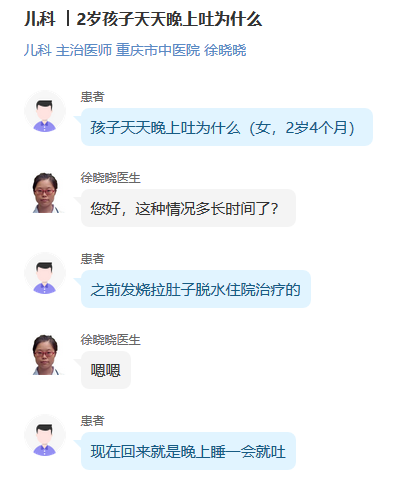
### 3.2.2 数据获取

本文构建的医疗领域知识图谱数据来源包括以下五个部分：1)“好大夫”医疗咨询网站上的疾病实体，药品实体，用来构建知识图谱，最后爬取的，疾病实体共7761个，药品实体共5474个；2)“9939”医疗咨询网站上的医生与患者的问答数据，用来训练词向量和seq2seq模型，“9939”的问答对共有16392个；3)“春雨医生”医疗咨询网站上的医生与患者的问答数据，用来训练词向量和seq2seq模型。“春雨医生”的问答对共有165828个；4)百度百科内容的爬取，用来建立实体与实体之间的关系，本文通过爬取百度百科的相关内容，并将其与已经爬取的实体进行匹配，如果两者有关联，那么建立关系。5)症状实体表，他人整理的症状实体列表，用来构建知识图谱，，症状实体共4637个。

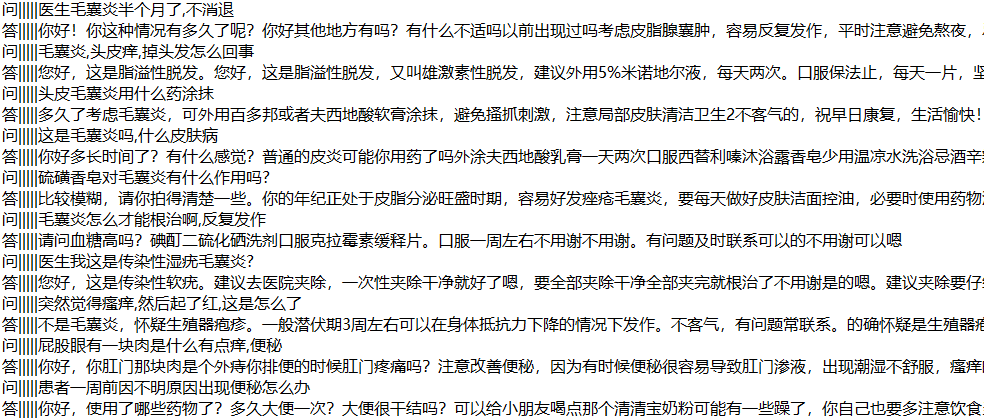


**图3-3 疾病实体部分数据**

对于问答对的爬取工作，是从多家网站进行爬取的，除了每个网站的结构不同之外，其中一个难点便是，一个问答页面里面可能会有多段问答，举例如图3-4 。对于问答对的爬取，这种多段问答不但在爬取上处理难度大，而且对于接下来的训练，也并不是规范的问答对，因此，本爬虫对于问答对的爬取，主要专注于网站整理的经典问答，也就是以一问一答方式整理出的问答语料。其爬取的思路与爬取药品实体的思路大致相同，在问答主界面上，依次访问各科室的url，记录下对于所有疾病的问答界面第一页的url，并根据这些url，生成之后的分页，从而爬取所有的问答对信息。



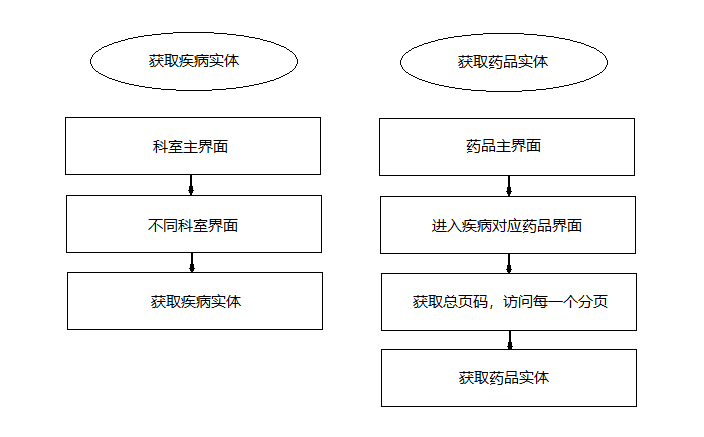
**图3-4 “春雨医生”网站上的多段问答举例**



**图 3-5 医生患者问答对部分数据**

### 3.2.3 知识图谱构建

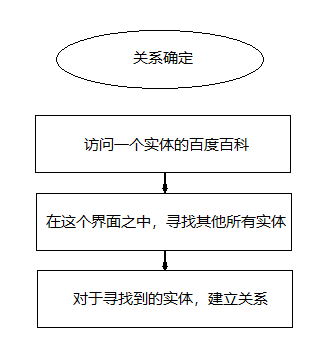
（1）实体抽取



**图3-6 实体爬取流程**

对于实体名称爬虫，实现的基本思路是，对于医疗咨询网站，他的疾病和药品实体，分别对应着不同的网页结构。对于疾病实体，从网站按照科室来划分，因此爬虫先按照科室总的界面，一个个获得到了科室界面，将科室界面的url存储在本地后，按照科室分类获取疾病，其症状，则是在疾病页面的信息中，直接获取即可。对于药品实体，其按照疾病和功能来划分，并且由于药品种类繁多，在爬取的时候，针对于特定功能的药品，会有几十个分页，因此，对于药品实体，在获取到了功能和疾病的url之后，根据第一页，以及页面内信息记录的总的分页数目，生成其他页面的url，再依次访问这些url，获得药品实体的全部名称。

（2）关系抽取



**图3-7 关系确定爬虫流程**

对于关系确定的工作，爬虫先以每个爬取到的一类实体，比如“药品”为例，根据百度百科的界面url格式，访问每个药品在百度百科上的信息，并在之中寻找对应实体的信息，比如“疾病”类实体，对于每一个药品，会有“适应症”这一栏，将这一栏中的信息与已经爬取到的疾病类实体进行匹配，如果发现了疾病，则可以确定，目前这两个实体之间存在“药品相关疾病”和“疾病相关药品”，这样的关系，以此类推。

（3）知识图谱构建

通过python爬虫爬取的各个实体以及关系，建立知识图谱。先从医疗问答网站的药品库和疾病库爬取相应实体，在根据相应的实体，用python访问百度百科，若存在与这个实体相关联的其他实体，在百科的文本中，则，这两个实体存在相应的关系，这些被爬取到的实体与关系，形成三元组。

（4）知识图谱存储

本文用RDF文件的形式存储知识图谱，以三元组为单位，将知识图谱存储下来，其主要格式如下：

声明实体种类

exampleo:疾病 rdf:type owl:Class .

exampleo:疾病 rdfs:subClassof owl:Thing .

声明实体属性

exampleo:症状相关疾病 rdf:type owl:ObjectProperty .

exampleo:症状相关疾病 rdfs:domain exampleo:症状 .

记录三元组

example:二尖瓣关闭不全 exampleo:疾病相关症状 example:端坐呼吸 .

### 3.2.4 知识图谱查询

本文使用jena框架对RDF数据进行操作，使用SPARQL对RDF数据进行增删查改。Jena是一个java框架，用来支持对于，XML，RDF等类型数据的操作。SPARQL是一种专门针对RDF的查询语言和数据获取协议，其以三元组的方式对RDF数据进行相应的操作，而一个SPARQL语句就是一个RDF图。如下面的一个查询语句：

PREFIX owl:< [http://www.w3.org/2002/07/owl#](http://www.w3.org/2002/07/owl)>

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

PREFIX skos: <http://www.w3.org/2004/02/skos/core#>

PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

PREFIX exampleo: <http://www.example.com/ontology/#>

PREFIX example: <http://www.example.com/>

SELECT ?a ?b ?c WHERE {

example: “阿莫西林”?a ?b .

?b rdf:type ?c .

?c rdf:type owl:Class .

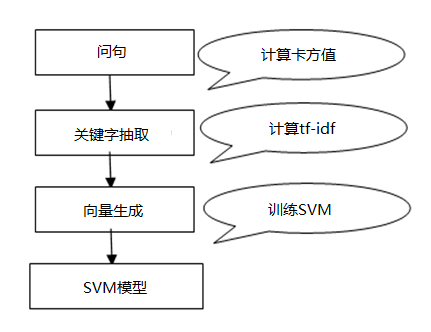
}

由PREFIX对应的语句是对知识库的配置，下面的带有变量a，b，c的语句则是带有变量的查询图，查询的是与实体“阿莫西林”相连接的实体，也就是包含实体“阿莫西林”的三元组。

本文主要使用该框架进行查询图构建操作，通过相应的查询操作构建查询图，是答案抽取的基础。

## 第3.3节 问题分类方案设计

问题分类的目的是对于不同的问题，有不同的答案抽取方式，提前将问题的类别提炼出来，对于生成正确的答案至关重要。本文使用的问题分类的思想如下：对于问题数据集，使用卡方的方法，找出这个问题之中的关键词，而对于这些关键字，更进一步的计算他们对应的tf-idf[20]值，构成相应的向量，对于这些生成的向量，输入到SVM之中，训练相应的模型，这样，对于用户输入的新的问题，便可以用训练好的SVM模型进行分类。对于给出的问题，本文将其分为三种，第一种是是否类问题，第二种是查找类问题，第三种是解释类问题。



**图3-8 基于SVM模型的问题分类流程**

### 3.3.1 问句关键字提取

本文通过计算每一个字的卡方统计量，来提取问句中的关键字。卡方统计量的思想是通过判断该词在相应类别与其他类别出现与不包含的次数，进行运算，得到卡方值，其值大小就表示这这个词对于这个类别的重要性，也就是衡量这个词能否成为这个句子的关键字的因素。在计算出一个字的卡方值之后，通过将这个值与预先设置的阈值进行对比，超过阈值的对应的字，就是一个关键字。

公式如下：

（3·1）

通过计算：

N:训练数据集中句子总数

A:在该类别中包含该字的句子总数

B:在其他类别中包含该字的句子总数

C:在该类别不包含该字的句子总数

D:在其他类别不包含该字的句子总数

本文章选取对于每一个类别之中，卡方值最高的四个字，作为整体关键字的一部分，在对问题中所有的类别的问题处理过之后，便计算得到了关键字表，关键词表中的关键字分别为“药”，“便 ”，“最”，“贵”，“是”，“多”，“怎”，“吗”，“吃”，“能”，“素”，“宜”，“病”，“有”，“中”，“哪”，“些”，“种”，“少”，“什”，“么”，“了”，“做”这23个字。

### 3.3.2 问句向量生成

本文采用TF-IDF方法，对选出的关键字计算权值，从而为每一个问题构成向量，其中TF是词频，IDF是逆文本频率指数。

TF的计算方法是计算这个词在相应的句子中的出现次数与这个句子中所有的词出现的次数之和的比值。也就是，对于句子中的词i和句子j，TF计算公式如下

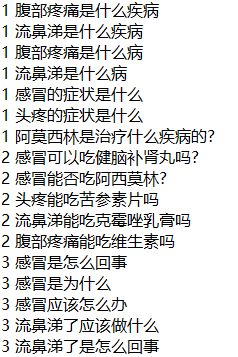
（3·2）

IDF是逆文本频率指数。对于词i，N为句子总数，Ni为含有词i的句子总数，IDF计算公式如下：

（3·3）

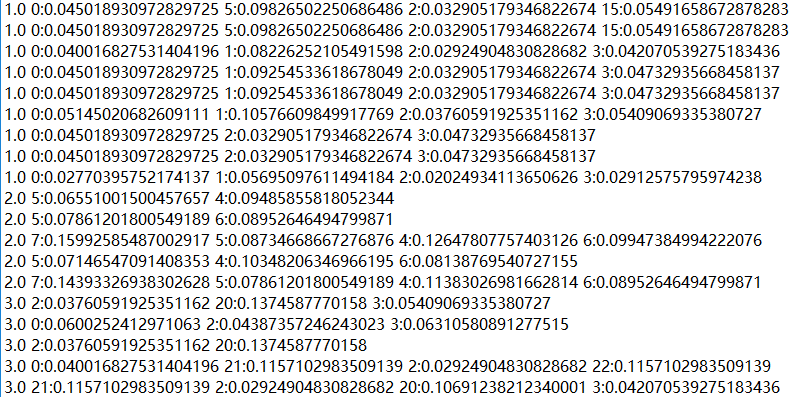
TF-IDF的值便由TF与IDF相乘而获得。

问题数据样例如图3-9所示：



**图3-9 问题样例数据**

得到的问题的TF-IDF词向量如图3-10所示：



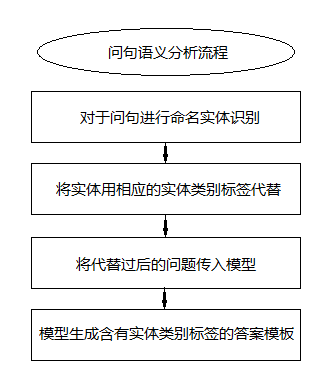
**图3-10 问题样例数据的TF-IDF表示的向量**

### 3.3.3 基于SVM的问句分类

SVM分类器的主要思想是，对于已经给出的，分好类别的一系列样本点，寻找超平面，使得距离这个超平面最近的点到这个超平面的距离最大，当超平面确定完成，分类器也完成了的训练。而当给出的数据是不可分时，SVM的核函数会帮助解决这一问题，通过将数据映射到高维空间，原始空间中的线性不可分问题就得以解决。也就是说，这时候的SVM是现在低维空间下完成计算，然后在高维空间中找到超平面。常用的核函数有多项式核函数，RBF核函数等。本文使用在java开发环境下的工具libsvm，以基于tf-idf的句子向量为基础，对于每一个类别进行分类。

## 第3.4节 问句语义分析方案设计

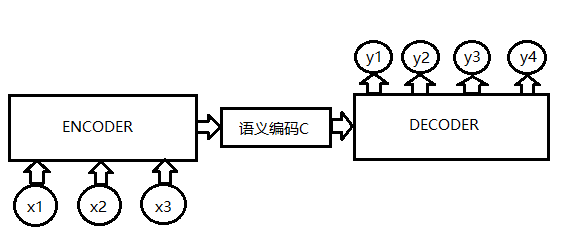
在成功获取了问句的相应类别之后，就要对问句进行语义分析，也就是对于问句的理解，包括识别实体，判断实体间关系等。本文对于实体识别的方法采用的是简单的按照实体字典进行查询匹配的方法。本章的主要介绍实体抽取过后的关系模板生成，通过深度学习的方法理解语义，并且为答案抽取提供了相应的关系限定。问句语义分析流程如图3-11所示：



**图3-11 问题样例数据的TF-IDF表示的向量**

### 3.4.1 基于深度学习的语义分析

本文使用tensorflow框架，通过构建一个应用于seq2seq的encoder-decoder模型，来进行语义分析。Encoder-decoder[21]模型是根据输入的文字序列，输出的文字序列，进行训练之后，能够完成从输入的一段文字获得生成的一段文字的模型，该模型分为两个层，分别是编码层和解码层，编码层将输入的问题进行处理之后，得到语义编码C，语义编码C作为输入，输入到解码层中，解码层对该语义编码进行处理，最终得到应当生成的文字序列。



**图3-12 encoder-decoder 模型结构图**

### 3.4.1.1 词向量训练以及数据准备

要进行对于encoder-decoder模型的训练，首先要将问答对转换成向量的形式。因此，首先要做的是训练词向量。进行词向量模型训练所必须的语料，是由python爬虫在医疗网站上爬取的问答对。分词过后，对问答对进行实体抽取，将问答对中具体的实体，替换成实体类别的相应标记，此时得到的问答语料记录的内容，便是各个类别实体的关系。例如，问题“感冒可以吃阿莫西林吗”在经过实体识别与标签标记之后，就变成了“disease可以吃drug吗”。其中，各个实体类别对应的标记如下表：

**表3-1 实体类别与标记**

|  |  |
| --- | --- |
| 实体类别 | 实体类别标记 |
| 疾病 | disease |
| 药品 | drug |
| 症状 | symptom |

得到的问答语料如图3-13：



**图3-13 处理过后的问答语料**

接下来，再使用word2vec对得到的问答语料进行训练，得到的便是针对该问答的实体关系的词向量，在训练之前，要把问答预料数据进行分词，以空格来分开每一个词，本文章使用的是jieba分词，由于这个中文分词工具，会将一串英文字母，当作一个词来处理，因此，对于实体类别的标签并不会影响到词向量训练的结果。

除此之外，为待训练的语料添加上“<PAD>”，“<EOS>”，“<UNK>”，“<GO>”这四个标签，使得得出的词向量集合能够给出这四个标签的词向量数值。

第一个标签“<PAD>”代表的是填充标签，由于在同一批次中需要输入的训练数据，要有相同的长度，在最大长度得出之后，剩下的长度不够的数据，便由填充标签进行补充，比如将问句[“感冒”，“吃”，“什么”，“药”]填充至长度为10，得到的结果便是[“感冒”，“吃”，“什么”，“药”，“<PAD>” ，“<PAD>” ，“<PAD>” ，“<PAD>” ，“<PAD>” ，“<PAD>”]。

第二个标签“<EOS>”表示问句已经产生答案或者句子末尾。在解码器中，标签“<EOS>”的产生代表着答案生成完毕。

第三个标签“<UNK>”表示未知，意思就是当输入的问句在分词之后得到的结果之中，有不在词表的词，那么这个词，将有标签“<UNK>”所代替，如，对于输入问句[“感冒”，“能”，“吃”，“dis”，“吗”]，此句会被转换为[“感冒”，“能”，“吃”，“<UNK>”，“吗”]。

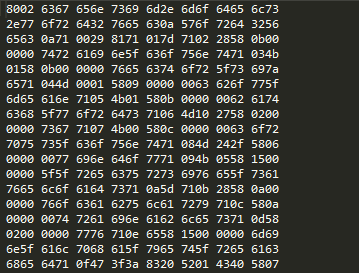
第四个标签“<GO>”代表的是生成答案的开始，是解码器接收到的第一个标记，解码器在接收到标签“<GO>”之后，开始产生答案。

训练过后的词向量以及问答对的基本信息如表2-2：

**表3-2 词向量，问答对基本信息**

|  |  |
| --- | --- |
| 问题词向量维度 | 256 |
| 问题词向量个数 | 630 |
| 训练用问答对个数 | 5167 |
| 测试用问答对个数 | 911 |

得到的部分词向量模型如图3-14所示。



**图3-14 生成的词向量模型部分数据**

### 3.4.1.2 encoder-decoder模型

构建的encoder-decoder模型由LSTM实现，其中编码端为双向双层的LSTM，每层中的神经元数量为256个，并且利用dropout机制来防止过拟合。解码端为一个全连接层，是一个双层的LSTM，每层中的神经元数量为256个，在使用了dropout机制的基础上又使用了注意力机制。

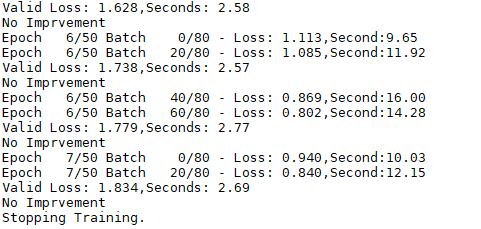
Dropout机制主要是为了防止过拟合现象。在进行深度学习模型的训练过程中，由于训练集不够大，或者是模型参数过多，往往会使得训练出的模型在训练集上效果良好，但是在测试集上的效果却比较差，这种现象就是过拟合现象。Dropout机制则通过使得每个隐层结点有概率p的机会使其隐层结点值为0，通过这种方式来减少隐层结点之间的相互作用，进而防止过拟合。

注意力机制类似于人类观察物体时大脑的理解运作机制，当人眼接收到大量的信息的时候，眼睛会将注意力专注于特定的小范围。注意力机制就是如此，在编码层中，对于文字序列中每一个元素，基于他的隐藏状态分数分配权值参数，这个权值参数代表着每一个词对给定目标的当前元素的影响程度，最后由这些隐藏状态分数得到注意力机制的概率分布，形成一个向量，在解码层中，将该向量与编码层的t时刻的隐藏状态加权求和，加入到全连接层作为编码层的输入。

在进行训练之前，关键的一步是得到词向量的词表，并且将文字表示的问答对，根据文字在词表中的下标，生成数字表示的问答对，这就是模型用来操作的最终语料，在得到数字表示的问答对之后，将词向量表，也输入模型之中。

### 3.4.1.3 模型训练

在进行训练时，设置的学习率是0.001，学习率衰减指数是0.9，设置训练数据的batch为64，epoch为50，也就是每一批的数据中样本数为64，对于所有的数据，要轮流训练50次。在训练过程中，每一经过一定的训练数据量，计算出loss值之后，都会与之前的值进行比较，如果此时得到的loss值小于先前最小值，那么这次训练便得到了新的纪录，若不是，则没有提升，当连续没有提升5次之后，则代表loss值趋近于平稳，训练得到的模型趋近于本次最佳模型。



**图3-15 模型训练进行过程示例**

### 3.4.1.4 模型抽取结果以及评测方法

本文使用准确率，召回率，F值这三个参数，对于训练得到的模型进行测试，得到相应的实验结果，判断模型的可用性。

准确率，反应对于生成的答案，其中正确的实体标签占生成答案中的所有实体标签的比例，计算公式如下：

（3.4）

召回率，用来反应生成答案中正确的实体标签占据整个测试数据集中的实体标签的比例，计算公式如下：

（3.5）

F值，是召回率和正确率的调和平均，主要功能是结合以上两个测试标准，反应模型的整体性能。

（3.6）

准确率是衡量模型运行的准确程度，而召回率则是衡量系统对于给出数据的覆盖能力，在进行实验室，实验者往往想要将这两个指标都有良好的表现，但是当提升其中一个指标的性能的时候，另一个指标往往不能兼顾，这样的情况，对于人们衡量一个系统的好与坏增加了难度，因此，作为这两个指标的调和平均值，F值在一定程度上，反映了两者结合后的整体性能，但是这并不代表准确率和召回值没有了体现的必要。因此，为了让实验结果更加直观，让实验数据，实验结果能够更好的帮助研究者分析，本文采用了准确率，召回值，F值这三者，作为整个模型的度量指标。

本次实验使用测试用的911个问答对，对于每个生成的答案和数据集中的答案，比较两者之中的实体标签，判断答案正确与否。如果在数据集中的答案生成的一个标签在生成的答案之中出现了，那么这就是一个正确识别的实体标签，而如果一个在生成的答案之中出现的实体标签，未能在数据集之中的答案出现，那么这便是一个错误识别的实体标签。而对于每一次识别，每有一种实体标签出现在数据集答案中，那么测试集中的实体标签数就加一，每有一种实体标签出现在生成的答案中，那么识别出的实体标签数就加一。

最终得到的测试集中的实体标签数为1160，识别出的实体标签数为1027，生成答案中正确的实体标签数目为820，得到正确率为0.7984，召回率为0.7069，F1值为0.7498。

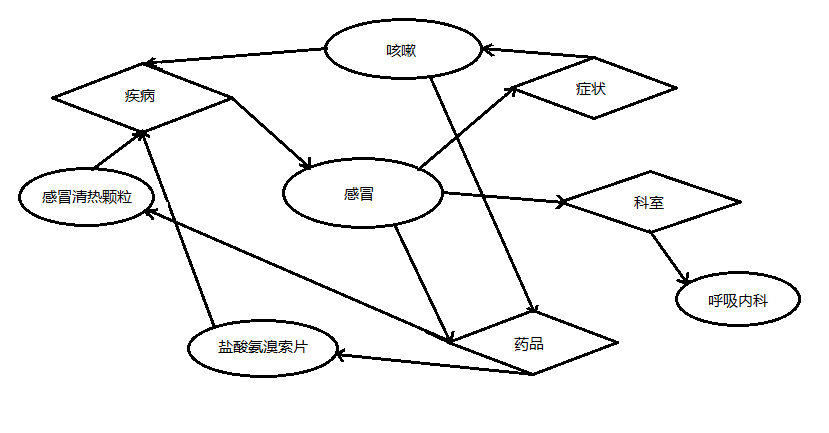
## 第3.5节 答案抽取方案设计

答案抽取是完成系统的最后一步，对于知识图谱，答案抽取的方法主要是关于图数据结构的相关操作，本文的答案抽取方案，包括查询图的构建和选择核心链。本文对于不同的问题类别，有不同的答案抽取方案。根据问句语义分析得到的关系模板，在查询图中采用不同的方式进行搜索，抽取核心链，最终生成最终答案。

### 3.5.1 查询图的构建

本文构建查询图的方法分为以下步骤：对于问题中的实体提取以及问题实体类别标记；针对提取出的每一个实体，构建他们的查询图，并将其合并。

对于得到的问题，先识别出问题中的实体，标记出这个实体的所属类别以及在问题中的位置，之后根据识别出的实体，开始构建查询图。对于每一个识别出的实体，寻找包含他们的最大连通图，以（结点，关系，结点）的方式存储下来，在为每一个实体查询完最大连通图之后，将这些图合并，作为最终得到的查询图。



**图3-16 对于问题“感冒吃什么药”得到的查询图的一部分**

### 3.5.2 选择核心链

对于不同类别的问题，本文选取核心链的方式有所不同。

对于是否类问题，在构建好的查询图中，对于问题中识别出的实体，进行广度优先遍历，如果这些实体都在同一个连通图中，则说明找到了核心链，二者有相应的关联，证明这个问题的答案为肯定。如果在遍历过后实体仍未被识别出，则，有实体与其他实体没有相应的关联，证明这个问题的答案为否定。

对于问答类问题，本文将输入的问题，进行实体抽取过后，在问题相应的位置标记实体的类别，之后将这个问题输入到encoder-decoder模型之中，得到相应的回答模板，作为答案实体之间关系的记录，同时也是对于在知识图谱查询中的束缚。针对回答模板中返回的实体类别和问题给出的实体，在查询图之中用dijkstra最短路径算法，一一查询最优对应实体，再根据他们的类别，安插在回答模板对应类别标签的位置上，从而生成最后的答案。

## 第3.6节 本章小结

本章主要介绍对于医疗领域的基于知识图谱的领域问答系统的实现总思路以及相应的算法的设计与实现。首先将该领域问答系统的实现分为：知识图谱构建方案设计，问题分类方案设计，问句语义分析方案设计和答案抽取方案设计这四个部分。知识图谱的构建，利用python爬虫爬取实体信息与他们相应的关系。构建完成后，训练SVM分类器并且对问题进行分类，其中对于每一个输入的问题，使用卡方方法和tf-idf思想，提取出问句中的关键字并且生成问句向量。再将分好类的问题进行问句语义分析，得到他们的实体与关系的信息，其中进行语义分析的模型经由python爬虫爬取的医疗问答对作为训练数据集，先由问答对训练词向量，再由词向量和问答对作为训练语料，得到最后的seq2seq模型。最后构建查询图，由dijkstra算法和广度遍历算法找到最终的答案，这便是本章的总体思路，也是本系统涉及到的主要实现方法。

# 第4章 系统实现

在第三章中我们将问答功能的实现思路分为四个部分，而本章则是主要介绍了基于前章中的实现思路实现的一个医疗领域问答系统，本系统的实现基于Springboot框架，利用了HTML，CSS，JS等技术，将问答功能可视化。对于前端界面，使用了amazeui框架。对于问句语义分析的seq2seq模型，则是由tensorflow框架实现的。

## 第4.1节 系统框架概述

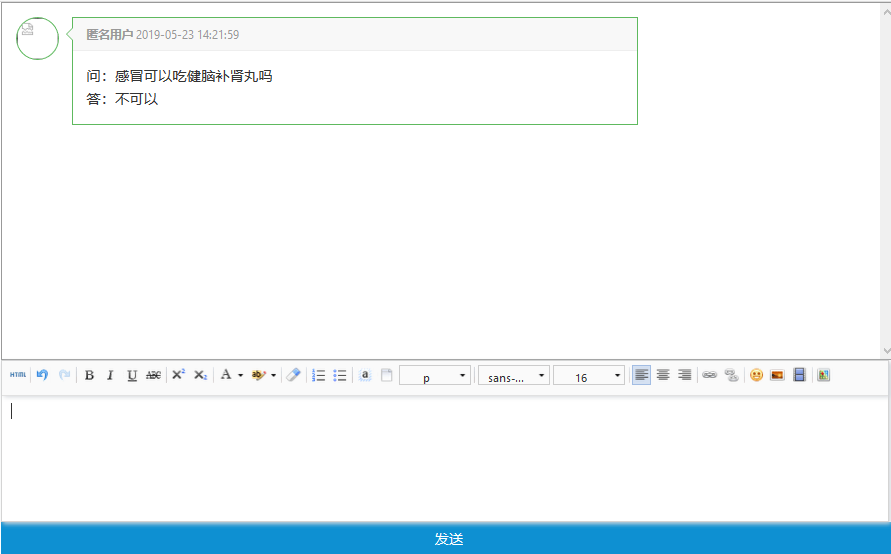
本系统使用Springboot框架，实现了一个问答系统，其前端界面应用了amazeui框架实现了一个问答交互聊天界面。Springboot框架是spring框架在某种意义上的简化，其对于maven的自动配置，不再需要特地部署war文件等特性，使得其在快速应用开发放面独树一帜。前端界面使用的amazeui框架，实现了一个输入问题，输出问答对的界面，其前后端的数据传输采用ajax方法，界面的js功能实现主要基于jquery。这一部分是java的开发环境。

对于问句语义分析的seq2seq模型，由tensorflow框架实现。Tensorflow框架是谷歌公司开发的一套专门针对机器学习领域的算法编程实现，该框架的操作以数据流为基本单位，完全支持C语言和python语言。其有CPU版本和GPU加速版本，本系统选择的是CPU版本。这一部分是python的开发环境。

## 第4.2节 应用实例

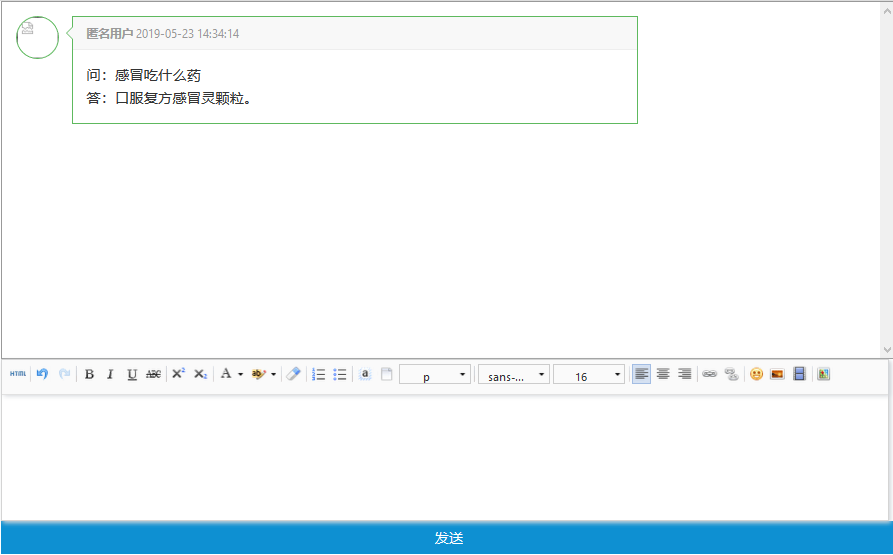
本章主要讲解对于该医疗领域问答系统的应用实例与各部分的实现效果。专注于问答系统的不同问题的问答界面。

### 4.2.1 问答系统



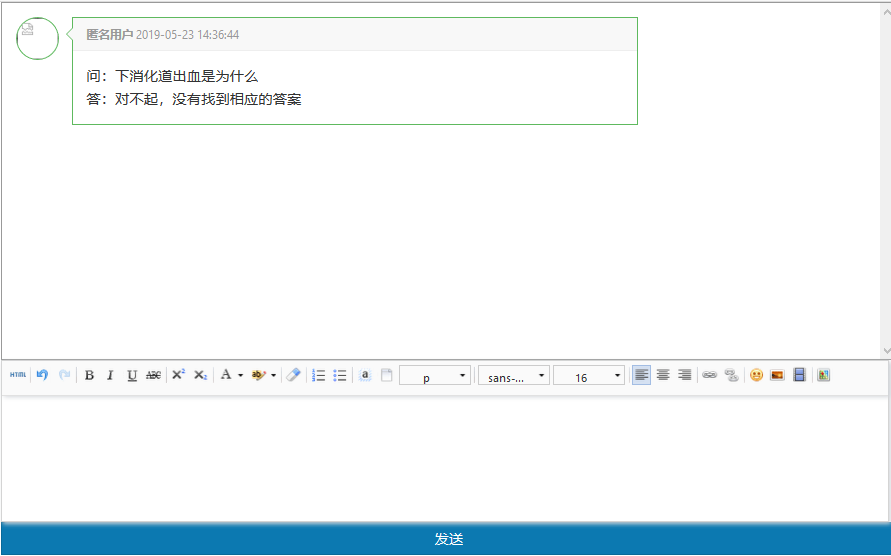
**图4-1 对于是否类问题“感冒可以吃健脑补肾丸吗”的实例**

图4-1展示了对于输入是否类问题“感冒可以吃健脑补肾丸吗”的系统显示结果，用户在输入框输入问题过后，给出的答案包括问题原型与答案，答案的类型包括肯定与否定，其中表示否定的回答是“不可以”。



**图4-2 对于问答类问题“感冒吃什么药”的实例**

图4-2展示了对于问答类问题“感冒吃什么药”的实例，用户在输入框输入问答类问题之后，返回的是含有知识图谱之中包含的信息的回答，也就是根据问题的输入的实体，构建查询图并且查找出答案所需实体的最终回答。



**图4-3 对于解释类问题“下消化道出血是为什么”的实例**

图4-3展示了对于解释类问题“下消化道出血是为什么”的实例，用户输入解释类问题之后，系统将其分类到对应的类别，之后对于解释类问题，系统目前无法给出回答，会有如图3-8的回答。

## 第4.3节 本章小结

本章介绍了该问答系统的界面实现详细信息，其中介绍了问答系统可视化和seq2seq模型实现所需的框架，分别是Springboot框架和基于python语言的Tensorflow框架。对于系统架构，其中包括数据支持和系统功能对于四部分的系统功能，每一部分都有相应的数据支持。系统的核心流程，是对于整个系统构建，与对于一个输入问题进行处理的关键步骤，介绍了这些步骤如何执行。对于系统问答部分实例，主要展现了对于输入的不同类型问题，本系统给出的可视化回答方案和对于不同答案的不同回答方式。

# 结 论

1. 论文总结

本文对实现了一个基于知识图谱的医疗领域问答系统研究。对于设计与实现一个完整的问答系统做出了实践和探索，主要工作如下：

1. 知识图谱构建。从“好大夫”，“春雨医生”，“9939”和“百度百科”网站爬取实体名称，进行相应的关系判断，构建知识库，同时爬取了医疗问答对，为词向量以及seq2seq模型的训练做好了准备。使用apache jena作为知识库操作框架。
2. 使用卡方方法，提取关键字，并且计算tf-idf值，生成问句向量，并且使用SVM分类器进行训练，用生成的模型对问题进行分类。
3. 构建基于encoder-decoder的seq2seq模型，并且对于python爬虫爬取下来的医疗问答对，训练相应的词向量，再使用双层双向LSTM作为编码层，使用双层并且带有注意力机制的LSTM的全连接层作为解码层，使用问答对数据集训练模型。使得得到的模型可以对于用户给出的问题，生成相应的关系模板。
4. 针对不同类别的问题，提供不同的答案抽取方案。通过问题中的实体，在知识库中搜索，在构建查询图，再根据问题的分类，用最短路径算法或广度优先遍历算法对于问题进行核心链选取，得到答案所需要的实体，生成最终答案。
5. 开发医疗问答系统的交互可视界面，让用户有更良好的体验，为系统的交互提供了简明快捷的操作方式。
6. 进一步展望

本文实现了一个基于知识图谱的医疗领域问答系统，在一定程度上能够对于用户输入的问题给出准确答案。但是目前仅支持单句问句的输入比之前的方法有所改进，这些尚且不足的地方将在日后的研究工作中进行更加深层次的研究：

1. 知识库的实体集构建方法比较僵化，如果采用语料标注并结合机器学习方法进行抽取，会有更好的效果。
2. 对于seq2seq的问答对语料，爬取得到的问答对语料，有很多无效信息，比如错别字，问候语。还有一个问题中的多段问答，会对模型训练带来影响，在下一步的研究解决了该问题，seq2seq模型就会有更加丰富，准确的结果。
3. 对于seq2seq模型训练，需要的数据量很大，本文的数据量目前来讲是不够的。在下一步工作中也应当丰富数据集。

# 参考文献

[1] AUER, S. R.; BIZER, C.; KOBILAROV, G.; LEHMANN, J.; CYGANIAK, R.; IVES Z. DBpedia: A Nucleus for a Web of Open Data[C]. Semantic Web, International Semantic Web Conference, Asian Semantic Web Conference, Iswc + Aswc, Busan, Korea. 2007: 722-735.

[2] Bollacker K D , Evans C , Paritosh P , et al. Freebase: A collaboratively created graph database for structuring human knowledge[C]. Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, 1247-1249.

[3] 秦志光刘 峤 李 杨 段 宏 刘 瑶. 知识图谱构建技术综述[J]. 计算机研究与发展, 2016, 53(3): 582-600.

[4] RABINER L. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition[J]. Proceedings of the IEEE, 1989, 77(2): 257–286.

[5] HARRIES M, DRUMMOND C, NIJSSEN S等. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data[J]. Encyclopedia of Machine Learning, 2010: 208–208.

[6] SCHÜLDT C, LAPTEV I, CAPUTO B. Recognizing human actions: A local SVM approach[J]. Proceedings - International Conference on Pattern Recognition, 2004, 3: 32–36.

[7] 程树东, 胡鹰 . 基于 BI-LSTM-CRF 模型的限定领域知识库问答系统[J]. 计算机与现代化, 2018, 275: 53-57.

[8] 杨思春，戴新宇，陈家駿. 面向开放域问答的问题分类技术研究进展[J]. 电子学报, 2015, 8: 1627-1636.

[9] KEHLENBECK ALEX W C P. Web-Scale Named Entity Recognition[J]. ACM Conference on Information and Knowledge Management, 2008, 2: 123–132.

[10] XV C, DU W, HE X等. Survey of Convolutional Neural Network[J]. 2018, 36.

[11] 杨笑然. 基于知识图谱的医疗专家系统[D]. 中国浙江, 浙江大学, 2018.

[12] 史梦飞. 基于中文知识库的问答系统研究与实现[D]. 中国, 华东师范大学, 2018.

[13] BERANT J, CHOU A, FROSTIG R等. Semantic Parsing on Freebase from Question-Answer Pairs[J]. Acl, 2013: 1533–1544.

[14] YAO X, DURME B Van. Information Extraction over Structured DataQuestion Answering with Freebase[J]. 2014: 956–966.

[15] BORDES A, CHOPRA S, WESTON J. Question Answering with Subgraph Embeddings[J]. 2014(1): 615–620.

[16] 张傲, 面向问答系统的问题分类与答案抽取研究D]. 中国, 东北大学, 2013.

[17] DEVLIN J, CHANG M-W, LEE K等. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding[J]. 2018.

[18] 夏宇航, 高大启, 阮 彤, 王昊奋, 殷亦超. 基于知识图谱的医疗病历数据存储研究[J]. 计算机工程. 2019, 45, 9-22.

[19] 康杰华，罗章璇. 基于图形数据库Neo4j的RDF数据存储研究[J]. 信息技术, 2015, 115-117.

[20] JOACHIMS T. A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization.[J]. Fourteenth International Conference on Machine Learning, 1997: 143–151.

[21] MENG F, LU Z, TU Z等. A Deep Memory-based Architecture for Sequence-to-Sequence Learning[J]. 2015: 1–12.

# 致 谢

从最初对于知识图谱，机器学习领域的一无所知，到最后应用所学的知识将问答系统实现，完成论文，内心感到很开心也很充实。在这短时间的对于毕业设计的研究中，我对问答系统的实现流程的相关内容进行了充分的调查，研究和学习，使用自己掌握的专业技能将基于知识图谱的领域问答系统完成，开发了可视化界面，并且爬取构建了一定量的数据集，经过每一次遇到的问题磨砺，我分析问题，解决问题的能力日益提升。

我要感谢我的指导老师万静老师。万静老师对待我们这些本科生非常耐心，负责。每隔一个星期，老师都会召集大家开组会，分享大家遇到的问题，同时汇报各自的毕设进度，在我们遇到了问题，愁眉不展的时候，老师会叫对应的学姐学长与我们沟通，一起发现问题，解决问题。正是因为老师的认真负责，对我们的关爱，我才能够从容不迫地完成这一篇毕业论文。

还要感谢每一个帮助过我的人，四年的相伴，相互搀扶让我们一同前行，一起成长。不论是导员的关心还是朋友的挂念，都是我在面对困难时心中的不灭的光。

# 附 录

Python爬虫代码，实体爬取：

爬取url

# -\*- coding: utf-8 -\*-

"""

Created on Tue Mar 5 23:42:30 2019

@author: admin

"""

from urllib.request import urlopen

import urllib

import requests

import bs4

from bs4 import BeautifulSoup

import re

import time

import random

headers={

'Host':'ptlogin2.qq.com',

'User-Agent': "Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:10.0) Gecko/20100101 Firefox/10.0 ",

}

user\_agents = [

'Mozilla/5.0 (Windows; U; Windows NT 5.1; it; rv:1.8.1.11) Gecko/20071127 Firefox/2.0.0.11',

'Opera/9.25 (Windows NT 5.1; U; en)',

'Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322; .NET CLR 2.0.50727)',

'Mozilla/5.0 (compatible; Konqueror/3.5; Linux) KHTML/3.5.5 (like Gecko) (Kubuntu)',

'Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.8.0.12) Gecko/20070731 Ubuntu/dapper-security Firefox/1.5.0.12',

'Lynx/2.8.5rel.1 libwww-FM/2.14 SSL-MM/1.4.1 GNUTLS/1.2.9',

"Mozilla/5.0 (X11; Linux i686) AppleWebKit/535.7 (KHTML, like Gecko) Ubuntu/11.04 Chromium/16.0.912.77 Chrome/16.0.912.77 Safari/535.7",

"Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:10.0) Gecko/20100101 Firefox/10.0 ",

]

def dealThePage(url,session):

url\_model = "https://www.haodf.com/jibing"

r2=session.get(url)

bsObj = BeautifulSoup(r2.content,"lxml")

title\_inner\_inner = bsObj.body.findAll("div",{"class":"m\_ctt\_green"})

judge\_inner\_inner = bsObj.body.findAll("div",{"class":"m\_title\_green"})

disease\_list = []

introduction\_list = []

disease\_url\_list = []

count=0;

for ele\_inner\_inner in title\_inner\_inner:

print(judge\_inner\_inner[count].text+"---------------------------------")

if '检查及手术' in judge\_inner\_inner[count].text:

count+=1

continue

else:

print(judge\_inner\_inner[count].text+"---------------------------------")

count+=1

mark\_ul=ele\_inner\_inner.findAll("ul")

#ul

for mark\_ul\_ele in mark\_ul:

mark\_li = mark\_ul\_ele.findAll("li")

#li

# print(mark\_li)

for content\_li in mark\_li:

if content\_li.has\_attr("class"):

if content\_li["class"]==["lineli"]:

continue

title\_set.add(content\_li.a.text)

# f.writelines(content\_li.a.text)

print(content\_li.a.text)

# time.sleep(3+random.randint(0,3))

mod = len(content\_li.a['href'])

disease\_url\_list.append(url\_model+content\_li.a['href'][7:mod])

r2=session.get(url\_model+content\_li.a['href'][7:mod-4]+"/jieshao.htm")

bsObj = BeautifulSoup(r2.content,"lxml")

introduction\_con = bsObj.body.findAll("div",{"class":"dis\_description"})[0].p.text

mod = len(introduction\_con)

# introduction\_con=introduction\_con[9:mod-13]

introduction\_con=introduction\_con[0:mod]

fin = open("D://introduction/"+content\_li.a.text+".txt","w+",encoding='utf-8')

fin.write(introduction\_con)

fin.close()

# break #循环疾病单词

# break #循环疾病的ul

# break #div遍历

return ""

def getHTMLText(url):

try:

r = requests.get(url, timeout=30)

r.raise\_for\_status()

r.encoding = r.apparent\_encoding

return r.text

except:

return " "

def main():

depth = 10

session = requests.Session()

session.headers = {

'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_11\_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/56.0.2924.87 Safari/537.36'

}

url\_model = "https://www.haodf.com/jibing"

start\_url = 'https://www.haodf.com/jibing/xiaoerke/list.htm'

fr = open("url.txt","r+")

line = fr.readline()

url\_list=line.split(" ")

fr.close()

deal\_list = []

for ele in url\_list:

if ele == '':

continue

else:

deal\_list.append(ele)

mod=deal\_list.index("https://www.haodf.com/jibing/xiaoeryanke/list.htm")

print(mod)

deal\_list=deal\_list[mod+1:]

for ele in deal\_list:

time.sleep(2)

dealThePage(ele,session)

title\_set = set()

main()

爬取实体

# -\*- coding: utf-8 -\*-

"""

Created on Tue Mar 5 23:42:30 2019

@author: admin

"""

from urllib.request import urlopen

import urllib

import requests

import bs4

from bs4 import BeautifulSoup

import os

import re

import time

import random

headers={

'Host':'ptlogin2.qq.com',

'User-Agent': "Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:10.0) Gecko/20100101 Firefox/10.0 ",

}

user\_agents = [

'Mozilla/5.0 (Windows; U; Windows NT 5.1; it; rv:1.8.1.11) Gecko/20071127 Firefox/2.0.0.11',

'Opera/9.25 (Windows NT 5.1; U; en)',

'Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322; .NET CLR 2.0.50727)',

'Mozilla/5.0 (compatible; Konqueror/3.5; Linux) KHTML/3.5.5 (like Gecko) (Kubuntu)',

'Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.8.0.12) Gecko/20070731 Ubuntu/dapper-security Firefox/1.5.0.12',

'Lynx/2.8.5rel.1 libwww-FM/2.14 SSL-MM/1.4.1 GNUTLS/1.2.9',

"Mozilla/5.0 (X11; Linux i686) AppleWebKit/535.7 (KHTML, like Gecko) Ubuntu/11.04 Chromium/16.0.912.77 Chrome/16.0.912.77 Safari/535.7",

"Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:10.0) Gecko/20100101 Firefox/10.0 ",

]

def dealThePage(url,session,fi\_name,se\_name):

title\_set = set()

r2=session.get(url)

bsObj = BeautifulSoup(r2.content,"lxml")

# print(bsObj)

title\_inner\_inner = bsObj.body.findAll("div",{"class":"m\_ctt\_green"})

judge\_inner\_inner = bsObj.body.findAll("div",{"class":"m\_title\_green"})

print("-------------------------------------------------")

count=0;

for ele\_inner\_inner in title\_inner\_inner:

print(judge\_inner\_inner[count].text+"---------------------------------")

if '检查及手术' in judge\_inner\_inner[count].text:

count+=1

continue

else:

print(judge\_inner\_inner[count].text+"---------------------------------")

count+=1

mark\_ul=ele\_inner\_inner.findAll("ul")

#ul

for mark\_ul\_ele in mark\_ul:

mark\_li = mark\_ul\_ele.findAll("li")

#li

# print(mark\_li)

for content\_li in mark\_li:

if content\_li.has\_attr("class"):

if content\_li["class"]==["lineli"]:

continue

title\_set.add(content\_li.a.text)

print(content\_li.a.text)

# break #循环疾病单词

# break #循环疾病的ul

# break #div遍历

fw = open(root\_dir+"/"+fi\_name+"/"+se\_name+".txt","w+",encoding="utf-8")

for ele in title\_set:

fw.write(ele)

fw.write("\n")

fw.close()

return ""

def getHTMLText(url):

try:

r = requests.get(url, timeout=30)

r.raise\_for\_status()

r.encoding = r.apparent\_encoding

return r.text

except:

return " "

def main():

depth = 10

session = requests.Session()

session.headers = {

'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_11\_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/56.0.2924.87 Safari/537.36'

}

start\_url = "https://www.haodf.com/jibing/xiaoerke/list.htm"

r=session.get(start\_url)

bsObj = BeautifulSoup(r.content,"lxml")

department\_fi = bsObj.body.findAll("div",{"class":"kstl"})

# first

for ele\_depar\_fi in department\_fi:

fi\_name = ele\_depar\_fi.a.text

if not os.path.exists(root\_dir+"/"+fi\_name):

os.makedirs(root\_dir+"/"+fi\_name)

#second

start\_url="https://www.haodf.com"+ele\_depar\_fi.a['href'];

r=session.get(start\_url)

bsObj = BeautifulSoup(r.content,"lxml")

department\_se = bsObj.body.findAll("div",{"class":"ksbd"})

if department\_se == []:

dealThePage(start\_url,session,fi\_name,fi\_name)

else:

department\_se = department\_se[0].ul.findAll("li")

print(department\_se)

for ele\_depar\_se in department\_se:

se\_name=ele\_depar\_se.a.text

url = "https://www.haodf.com"+ele\_depar\_se.a['href']

url = url[:len(url)-1]

print(url)

time.sleep(2)

dealThePage(url,session,fi\_name,se\_name)

root\_dir="all\_dis\_info"

title\_list = []

main()

建立关系：

# -\*- coding: utf-8 -\*-

"""

Created on Wed May 1 14:51:27 2019

@author: admin

"""

import requests

from bs4 import BeautifulSoup

drugs = []

fr = open("drugs.txt","r+",encoding='utf-8')

line = fr.readline()

while(line):

drugs.append(line[:len(line)-1])

line = fr.readline()

fr.close()

drugs\_my = []

fr = open("medicine.txt","r+",encoding='utf-8')

line = fr.readline()

while(line):

med = line[:len(line)-1]

meds = med.split(' ')

for ele in meds:

if ele != '':

drugs\_my.append(ele)

line = fr.readline()

fr.close()

diseases = []

fr = open("diseases.txt","r+",encoding='utf-8')

line = fr.readline()

while(line):

diseases.append(line[:len(line)-1])

line = fr.readline()

fr.close()

new\_drugs = []

def doData():

count = 0

fr = open("count.txt","r+",encoding='utf-8')

line = fr.readline()

count = int(line)

fr.close()

session = requests.Session()

session.headers = {

'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_11\_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/56.0.2924.87 Safari/537.36'

}

url = "https://www.chunyuyisheng.com/pc/search/?query="

for drug\_my in drugs\_my:

if drug\_my not in drugs:

url\_my = url + drug\_my

r=session.get(url\_my)

bsObj = BeautifulSoup(r.content,"lxml")

p = bsObj.body.findAll("p",{"class":"detail"})

if p:

p\_text = p[0].text

exist\_dis = []

for dis in diseases:

if dis in p\_text:

exist\_dis.append(dis)

if len(exist\_dis) >0:

line = "example:"+drug\_my +" rdf:type exampleo:药品 ."

fw = open("kb.ttl","a+",encoding='utf-8')

fw.write(line)

fw.write('\n')

fw.close()

new\_drugs.append(drug\_my)

for dis in exist\_dis:

fw = open("kb.ttl","a+",encoding='utf-8')

line\_r = "example:"+ dis +" exampleo:疾病相关药品 example:"+ drug\_my +" ."

line\_l = "example:"+ drug\_my +" exampleo:药品相关疾病 example:"+ dis +" ."

fw.write(line\_r)

fw.write("\n")

fw.write(line\_l)

fw.write("\n")

fw.close()

count += 1

fw = fr = open("count.txt","w+",encoding='utf-8')

fw.write(str(count))

fw.close()

for new\_drug in new\_drugs:

fw = open("drugs.txt","a+",encoding='utf-8')

fw.write(new\_drug)

fw.write('\n')

fw.close()

doData()

量化问句：

**public** **void** load(String src) {

***LOGGER***.log(Level.***INFO***, "load training file = {0}", src);

BufferedReader br = **null**;

**try** {

//类别，单词，出现次数

//trainDatas 存储label

Map<Double, List<Map<String, Double>>> trainDatas = **new** HashMap<>();

br = **new** BufferedReader(**new** InputStreamReader(**new** FileInputStream(src), "UTF-8"));

String line;

//逐句分析，获得向量。

//tf是每一句的向量

//trainsData是总的向量表

**while** ((line = br.readLine()) != **null**) {

String[] contents = line.split(" ");

**double** label = Double.*parseDouble*(contents[0]);

//contents[1],所有问题的句子，如“阿莫西林是干什么用的”

String sentence = contents[1];

//按字分词

List<String> ws = segment(sentence);

// System.out.println(label);

**if** (!trainDatas.containsKey(label)) {

trainDatas.put(label, **new** ArrayList<>());

}

Map<String, Double> tf = **new** HashMap<>();

//创建对于每一个字的向量，记录出现次数。

ws.forEach(word -> {

**if** (!word.matches(",|\\.|:|\\?|，|。|？")) {

**if** (!tf.containsKey(word)) {

tf.put(word, 0.0);

}

tf.put(word, tf.get(word) + 1.0);

}

});

trainDatas.get(label).add(tf);

}

//处理数据

//(1)卡方统计量 (N\*(AD-CB)2)/((A+C)(B+D)(A+B)(C+D))

//N = 训练数据集句子总数

//A = 在这个类别中，包含这个词的句子数量

//B = 在这个类别中，排除这个类别，其他类别中包含这个词的句子数量

//C = 在这个类别中，不包含这个词的句子数量

//D = 在这个类别中，排除这个类别，其他类别中不包含这个词的句子数量

**double** N = 0.0;

**for** (List<Map<String, Double>> list : trainDatas.values()) {

N += list.size();

}

//类别，词语，在该类别中的句子中出现该词语的句子数目

Map<Double, Map<String, Double>> A = **new** HashMap<>();

//类别，词语，在其他类别中的句子中出现该词语句子数目

Map<Double, Map<String, Double>> B = **new** HashMap<>();

//类别，词语，在该类别中的句子中没有出现该词语的句子数目

Map<Double, Map<String, Double>> C = **new** HashMap<>();

//类别，词语，在其他类别中的句子中未出现该词语句子数目

Map<Double, Map<String, Double>> D = **new** HashMap<>();

//A = 在这个类别中，包含这个词的句子数量。stream生成流，entryset生成键值对。

trainDatas.entrySet().stream().forEach(entry -> {

**double** label = entry.getKey();

List<Map<String, Double>> sentences = entry.getValue();

Map<String, Double> counts = **new** HashMap<>();

sentences.stream().forEach(sentence -> {

sentence.keySet().stream().forEach(word -> {

**if** (!counts.containsKey(word)) {

counts.put(word, 0.0);

}

counts.put(word, counts.get(word) + 1.0);

});

});

A.put(label, counts);

});

//B = 在这个类别中，排除这个类别，其他类别中包含这个词的句子数量

trainDatas.keySet().stream().forEach(label1 -> {

//A中label1的所有词

Set<String> wordsInLabel1 = A.get(label1).keySet();

Map<String, Double> counts = **new** HashMap<>();

wordsInLabel1.stream().forEach(wordInlabel1 -> {

counts.put(wordInlabel1, 0.0);

trainDatas.entrySet().stream().forEach(entry2 -> {

**double** label2 = entry2.getKey();

**if** (label1 != label2) {

List<Map<String, Double>> sentences = entry2.getValue();

sentences.stream().forEach(sentence -> {

**if** (sentence.keySet().contains(wordInlabel1)) {

counts.put(wordInlabel1, counts.get(wordInlabel1) + 1);

}

});

}

});

});

B.put(label1, counts);

});

//C = 在这个类别中，不包含这个词的句子数量

trainDatas.entrySet().stream().forEach(entry1 -> {

**double** label1 = entry1.getKey();

List<Map<String, Double>> sentences = entry1.getValue();

Set<String> wordsInLabel1 = A.get(label1).keySet();

Map<String, Double> counts = **new** HashMap<>();

wordsInLabel1.stream().forEach(wordInlabel1 -> {

counts.put(wordInlabel1, (**double**) entry1.getValue().size());

sentences.stream().forEach(sentence -> {

**if** (sentence.containsKey(wordInlabel1)) {

counts.put(wordInlabel1, counts.get(wordInlabel1) - 1);

}

});

});

C.put(label1, counts);

});

//D = 在这个类别中，排除这个类别，其他类别中不包含这个词的句子数量

trainDatas.entrySet().stream().forEach(entry1 -> {

**double** label1 = entry1.getKey();

Set<String> wordsInLabel1 = A.get(label1).keySet();

Map<String, Double> counts = **new** HashMap<>();

wordsInLabel1.stream().forEach(wordInlabel1 -> {

counts.put(wordInlabel1, 0.0);

trainDatas.entrySet().stream().forEach(entry2 -> {

**double** label2 = entry2.getKey();

**if** (label1 != label2) {

List<Map<String, Double>> sentences = entry2.getValue();

counts.put(wordInlabel1, counts.get(wordInlabel1) + sentences.size());

sentences.stream().forEach(sentence -> {

**if** (sentence.keySet().contains(wordInlabel1)) {

counts.put(wordInlabel1, counts.get(wordInlabel1) - 1);

}

});

}

});

});

D.put(label1, counts);

});

//(N\*(AD-CB)2)/((A+C)(B+D)(A+B)(C+D))

**for** (Map.Entry<Double, Map<String, Double>> entry : A.entrySet()) {

**double** kind = entry.getKey();

Map<String, Double> map = entry.getValue();

Map<String, Double> chis = **new** HashMap<>();

**for** (String word : map.keySet()) {

**double** a = map.get(word);

**double** b = B.get(kind).get(word);

**double** c = C.get(kind).get(word);

**double** d = D.get(kind).get(word);

**double** chi = (N \* Math.*pow*(a \* d - c \* b, 2)) / ((a + c) \* (b + d) \* (a + b) \* (c + d));

chis.put(word, chi);

}

List<Map.Entry<String, Double>> list = **new** ArrayList<>(chis.entrySet());

Collections.*sort*(list, (o1, o2) -> {

**return** Double.*compare*(o2.getValue(), o1.getValue());

});

//Top10

**for** (**int** i = 0; i < ***TOP\_N*** && i < list.size(); i++) {

**if** (!**this**.words.contains(list.get(i).getKey())) {

**this**.words.add(list.get(i).getKey());

}

}

}

***LOGGER***.log(Level.***INFO***, "TOPN = {0}", ***TOP\_N***);

//(2)idf

StringBuilder sb = **new** StringBuilder();

**for** (String word : **this**.words) {

**this**.idfs.put(word, 0.0);

trainDatas.entrySet().forEach(entry -> {

entry.getValue().forEach(sentence -> {

**if** (sentence.containsKey(word)) {

**this**.idfs.put(word, **this**.idfs.get(word) + 1.0);

}

});

});

**double** idf = Math.*log10*(N / **this**.idfs.get(word));

**this**.idfs.put(word, idf);

sb.append(word);

sb.append(" ");

}

// System.out.println("------------------------------------------------------------------");

// System.out.println(this.idfs);

// System.out.println("------------------------------------------------------------------");

***LOGGER***.log(Level.***INFO***, "words = {0}", sb.toString());

} **catch** (IOException | NumberFormatException ex) {

Logger.*getLogger*(Pretreatment.**class**.getName()).log(Level.***SEVERE***, **null**, ex);

System.*exit*(-1);

} **finally** {

**if** (br != **null**) {

**try** {

br.close();

} **catch** (IOException ex) {

Logger.*getLogger*(Pretreatment.**class**.getName()).log(Level.***SEVERE***, **null**, ex);

}

}

}

}

SVM模型训练：

**public** **static** **void** trainSVM() {

Pretreatment p = **new** Pretreatment();

p.load("G:\\毕设\\数据\\问题分类训练数据.txt");

BufferedReader br = **null**;

BufferedWriter bw = **null**;

**try** {

br = **new** BufferedReader(**new** FileReader("G:\\毕设\\数据\\问题分类训练数据.txt"));

bw = **new** BufferedWriter(**new** FileWriter("G:\\毕设\\数据\\train\_data.txt"));

String line;

**while** ((line = br.readLine()) != **null**) {

String[] contents = line.split(" ");

**double** label = Double.*parseDouble*(contents[0]);

String sentence = contents[1];

StringBuilder sb = **new** StringBuilder();

sb.append(label);

sb.append(" ");

//对每个句子的svm节点，写入sb

p.pretreat(sentence).forEach(node -> {

sb.append(node.index);

sb.append(":");

sb.append(node.value);

sb.append(" ");

});

sb.deleteCharAt(sb.length() - 1);

bw.write(sb.toString());

bw.newLine();

}

br.close();

bw.close();

String[] arg = {"-g", "2.0", "-c", "32", "-t", "0", "-m", "500.0", "-h", "0",

"G:\\毕设\\数据\\train\_data.txt", "G:\\毕设\\数据\\model.txt"};

// svm\_train.main(arg);

}**catch** (IOException | NumberFormatException e) {

} **finally** {

}

}

Seq2seq模型：

# -\*- coding: utf-8 -\*-

"""

Created on Mon May 6 14:19:07 2019

@author: admin

"""

import numpy as np

import tensorflow as tf

from gensim.models.word2vec import Word2Vec

import jieba

import time

from tensorflow.python.layers.core import Dense

def checkList(theList,name):

print("--------------------------the information of "+name+"--------------------")

print("len:"+str(len(theList)))

print("the top 10 elements:")

for index in range(0,10):

print(theList[index])

print("-------------------------- end --------------------------")

def stopWordsList():

stopwords = [line.strip() for line in open('stopwords.txt',encoding='UTF-8').readlines()]

return stopwords

def myReadQAFile():

questions = []

answers = []

fr = open("QADATA\_5\_5\_5\_5.txt","r+",encoding='utf-8')

line = fr.readline()

while(line):

questions.append(line[6:len(line)-1])

line = fr.readline()

answers.append(line[6:len(line)-1])

line = fr.readline()

return questions,answers

def myReplace(oldStr,str1,str2):

length = len(str1)

result = oldStr

while(str1 in result):

index = result.index(str1)

leftStr = result[:index]

midStr = str2

rightStr = result[index+length:len(result)]

result = leftStr+ midStr+rightStr

return result

def stopWordsOperation(questions,stopwords):

for index in range(0,len(questions)):

for stopword in stopwords:

if stopword in questions[index]:

questions[index] = myReplace(questions[index],stopword,"")

return questions

def trainWord2Vec(answers\_jieba):

# model.build\_vocab(answers\_jieba)

# model.train(answers\_jieba,total\_examples = model.corpus\_count,epochs = model.iter)

model=Word2Vec(answers\_jieba, size=100, window=5, min\_count=1, workers=4)

model.save(u'QAvecs\_3.model')

def operation(total\_jieba):

total\_train =[]

for index in range(0,len(total\_jieba)):

tmp = total\_jieba[index].split(' ')

total\_train.append(tmp)

return total\_train

def readDrDiSy():

drugs =[]

diseases = []

symptoms = []

fr = open("drugs.txt","r+",encoding='utf-8')

line = fr.readline()

while(line):

drugs.append(line[:len(line)-1])

line = fr.readline()

fr.close()

diseases = []

fr = open("diseases.txt","r+",encoding='utf-8')

line = fr.readline()

while(line):

diseases.append(line[:len(line)-1])

line = fr.readline()

symptoms = []

fr = open("entities.txt","r+",encoding='utf-8')

line = fr.readline()

while(line):

symptoms.append(line[:len(line)-1])

line = fr.readline()

return drugs,diseases,symptoms

def getLength(questions\_jieba):

min\_q\_length=200

max\_q\_length=0

for question\_jieba in questions\_jieba:

if len(question\_jieba) > max\_q\_length:

max\_q\_length = len(question\_jieba)

elif len(question\_jieba) < min\_q\_length:

min\_q\_length = len(question\_jieba)

return min\_q\_length,max\_q\_length

def QAAddUNK(questions\_jieba,answers\_jieba,QA\_word2vecs):

questions\_UNK = []

answers\_UNK = []

for question\_jieba in questions\_jieba:

words = []

for word in question\_jieba:

if word not in QA\_word2vecs.wv:

words.append('<UNK>')

else:

words.append(word)

questions\_UNK.append(words)

for answer\_jieba in answers\_jieba:

words = []

for word in answer\_jieba:

# print(word)

if word not in QA\_word2vecs.wv:

# print(word)

words.append('<UNK>')

else:

words.append(word)

answers\_UNK.append(words)

return questions\_UNK,answers\_UNK

def calculate\_UNK\_ratio(questions\_UNK,answers\_UNK):

word\_count = 0

unk\_count = 0

for question\_UNK in questions\_UNK:

for word in question\_UNK:

if word == '<UNK>':

unk\_count += 1

word\_count += 1

for answer\_UNK in answers\_UNK:

for word in answer\_UNK:

if word == '<UNK>':

unk\_count += 1

word\_count += 1

unk\_ratio = round(unk\_count/word\_count,4)\*100

print("Number of times <UNK>: {}%".format(round(unk\_ratio,3)))

def sortQA(questions\_UNK,answers\_UNK,min\_q\_length,max\_q\_length):

sorted\_questions=[]

sorted\_answers = []

for length in range(min\_q\_length,max\_q\_length+1):

for index in range(0,len(questions\_UNK)):

if len(questions\_UNK[index]) == length:

sorted\_questions.append(questions\_UNK[index])

sorted\_answers.append(answers\_UNK[index])

return sorted\_questions,sorted\_answers

def vecArray2float32(vecArray):

floatList = []

for num in vecArray:

floatList.append(num)

return floatList

def getIdsWtvWtiItv(word2vecs):

keys\_list = []

keys\_list = word2vecs.wv.vocab.keys()

ids = []

for key in keys\_list:

ids.append(key)

word\_to\_vecs = {}

word\_to\_ids = {}

ids\_to\_vecs = []

for index in range(0,len(ids)):

word\_to\_vecs[ids[index]] = vecArray2float32(word2vecs[ids[index]])

word\_to\_ids[ids[index]] = index

ids\_to\_vecs.append(vecArray2float32(word2vecs[ids[index]]))

return ids,word\_to\_vecs,word\_to\_ids,ids\_to\_vecs

def getQAint(sorted\_questions,sorted\_answers,QA\_word\_to\_ids):

questions\_int = []

for question in sorted\_questions:

ints = []

for word in question:

if word not in QA\_word\_to\_ids:

ints.append(QA\_word\_to\_ids['<UNK>'])

else:

ints.append(QA\_word\_to\_ids[word])

questions\_int.append(ints)

answers\_int = []

for answer in sorted\_answers:

ints = []

for word in answer:

if word not in QA\_word\_to\_ids:

ints.append(QA\_word\_to\_ids['<UNK>'])

else:

ints.append(QA\_word\_to\_ids[word])

answers\_int.append(ints)

return questions\_int,answers\_int

def model\_inputs():

#question batch

input\_data = tf.placeholder(tf.int32,[None,None],name='input')

#answer batch

targets = tf.placeholder(tf.int32,[None,None],name='targets')

lr = tf.placeholder(tf.float32,name='learning\_rate')

keep\_prob = tf.placeholder(tf.float32,name='keep\_prob')

vocab\_table = tf.placeholder(tf.float32,[None,None],name='vocab\_table')

target\_sequence\_length = tf.placeholder(tf.int32, (None,), name='target\_sequence\_length')

max\_target\_sequence\_length = tf.reduce\_max(target\_sequence\_length, name='max\_target\_len')

return input\_data,targets,lr,keep\_prob,vocab\_table,target\_sequence\_length, max\_target\_sequence\_length

#编码层

def encoding\_layer(

rnn\_inputs,

rnn\_size,

num\_layers,

keep\_prob,

sequence\_length

):

'''

#实现多层LSTM循环神经网络

enc\_cell = tf.contrib.rnn.MultiRNNCell([drop]\*num\_layers)

'''

def lstm\_cell():

#每一个单元的输出为 rnn\_size 的向量

lstm = tf.contrib.rnn.LSTMCell(rnn\_size)

#dropout

drop = tf.contrib.rnn.DropoutWrapper(

lstm,

input\_keep\_prob = keep\_prob

)

return drop

fw\_enc\_cell = tf.contrib.rnn.MultiRNNCell([lstm\_cell() for \_ in range(num\_layers)])

bw\_enc\_cell = tf.contrib.rnn.MultiRNNCell([lstm\_cell() for \_ in range(num\_layers)])

#双向RNN

\_,enc\_state = tf.nn.bidirectional\_dynamic\_rnn(

cell\_fw = fw\_enc\_cell,

cell\_bw = bw\_enc\_cell,

sequence\_length = sequence\_length,

inputs = rnn\_inputs,

dtype = tf.float32

)

return enc\_state

def \_embed(self):

"""

词向量矩阵 \* ids向量；不是矩阵相乘，具体操作查看embedding\_lookup（好像是把ids相关的向量在矩阵中提前排列）

The embedding layer, question and passage share embeddings

"""

with tf.device('/cpu:0'), tf.variable\_scope('word\_embedding'):

self.word\_embeddings = tf.get\_variable(

'word\_embeddings',

shape=(self.vocab.size(), self.vocab.embed\_dim),

initializer=tf.constant\_initializer(self.vocab.embeddings),

#initializer=tf.zeros\_initializer(),

trainable=True #加入训练！！

)

#输出还是矩阵，只是行序发生改变

self.p\_emb = tf.nn.embedding\_lookup(self.word\_embeddings, self.p)

#删除每一个批次之中最后一个单词的ID，并且在每个批次开头加入<GO>标记

def pad\_sentence\_batch(sentence\_batch,vocab\_to\_int):

# print(sentence\_batch)

# max\_sentence = len(sentence\_batch)

# print([sentence + [vocab\_to\_int['<PAD>']]\*(max\_sentence - len(sentence)) for sentence in sentence\_batch])

max\_sentence = max([len(sentence) for sentence in sentence\_batch])

return [sentence + [vocab\_to\_int['<PAD>']]\*(max\_sentence - len(sentence)) for sentence in sentence\_batch]

def process\_encoding\_input(

target\_data,

vocab\_to\_int,

batch\_size

):

#tf.strided\_slice是多维切片函数

#前四个参数分别为：输入数据，开始切片处，终止切片处，步长。区间为开区间

#删除每一个批次之中最后一个单词的ID

ending = tf.strided\_slice(target\_data,[0,0],[batch\_size,-1],[1,1])

#用来拼接张量的函数,tf.concat([tensor1, tensor2, tensor3,...], axis)

#axis=1 代表在第1个维度拼接

#tf.fill([batch\_size,1],vocab\_to\_int['<GO>'])

#生成[batch\_size,1]维度的vocab\_to\_int['<GO>']

#tf.concat将ending加到每个batch的<GO>的后面

dec\_input = tf.concat([

tf.fill([batch\_size,1],vocab\_to\_int['<GO>']),ending],

1

)

return dec\_input

def decoding\_layer\_train(

encoder\_state,

dec\_cell,

dec\_embed\_input,

sequence\_length,

decoding\_scope,

output\_layer,

keep\_prob,

batch\_size,

max\_target\_sequence\_length

):

training\_helper = tf.contrib.seq2seq.TrainingHelper(

inputs=dec\_embed\_input,

sequence\_length=sequence\_length,

time\_major=False

)

'''

memory = tf.transpose(encoder\_outputs, [1, 0, 2])

attention\_mechanism = tf.contrib.seq2seq.BahdanauAttention(

num\_units=self.hidden\_size, memory=memory,

memory\_sequence\_length=self.encoder\_inputs\_actual\_length)

'''

'''

#三维数组，每个输入的单元代表着相应维的长度

attention\_states = tf.zeros([batch\_size,1,dec\_cell.output\_size])

#？

attention\_mechanism = tf.contrib.seq2seq.BahdanauAttention(

num\_units=dec\_cell.output\_size, memory=attention\_states,

memory\_sequence\_length=sequence\_length)

'''

training\_decoder = tf.contrib.seq2seq.BasicDecoder(dec\_cell,

training\_helper,

encoder\_state,

output\_layer)

'''

att\_keys,att\_vals,att\_score\_fn,att\_construct\_fn = tf.contrib.seq2seq.prepare\_attention(

attention\_states,

attention\_option="bahdanau",

num\_units = dec\_cell.output\_size

)

'''

#？

'''

train\_decoder\_fn = tf.contrib.seq2seq.attention\_decoder\_fn\_train(

encoder\_state[0],

att\_keys,

att\_vals,

att\_score\_fn,

att\_construct\_fn,

name="attn\_dec\_train"

)

#dynamic的概念，即不需要确定的输入长度，以及batch 大小， 都可以动态。

#但是注意首先每个batch对应所有样本的输入长度还是需要一样的 作为dense数据 否则 不可处理

train\_pred,\_,\_ = tf.contrib.seq2seq.dynamic\_rnn\_decoder(

dec\_cell,

train\_decoder\_fn,

dec\_embed\_input,

sequence\_length,

scope=decoding\_scope

)

'''

training\_decoder\_output, \_ = tf.contrib.seq2seq.dynamic\_decode(

training\_decoder,

impute\_finished=True,

maximum\_iterations=max\_target\_sequence\_length

)

#tf.nn.dropout是TensorFlow里面为了防止或减轻过拟合而使用的函数，它一般用在全连接层。

# train\_pred\_drop = tf.nn.dropout(train\_pred,keep\_prob)

return training\_decoder\_output

def decoding\_layer\_infer(

encoder\_state,

dec\_cell,

dec\_embeddings,

start\_of\_sequence\_id,

end\_of\_sequence\_id,

maximum\_length,

vocab\_size,

decoding\_scope,

output\_layer,

keep\_prob,

batch\_size,

max\_target\_sequence\_length

):

'''

attention\_states = tf.zeros([batch\_size,1,dec\_cell.output\_size])

att\_keys,att\_vals,att\_score\_fn,att\_construct\_fn=tf.contrib.seq2seq.prepare\_attention(

attention\_states,

attention\_option="bahdanau",

num\_units=dec\_cell.output\_size

)

infer\_decoder\_fn = tf.contrib.seq2seq.attention\_decoder\_fn\_inference(

output\_fn,

encoder\_state[0],

att\_keys,

att\_vals,

att\_score\_fn,

att\_construct\_fn,

dec\_embeddings,

start\_of\_sequence\_id,

end\_of\_sequence\_id,

maximum\_length,

vocab\_size,

name="attn\_dec\_inf"

)

infer\_logits,\_,\_ = tf.contrib.seq2seq.dynamic\_rnn\_decoder(

dec\_cell,

infer\_decoder\_fn,

scope=decoding\_scope

)

'''

start\_tokens = tf.tile(tf.constant([start\_of\_sequence\_id],dtype=tf.int32), [batch\_size], name='start\_tokens')

predicting\_helper = tf.contrib.seq2seq.GreedyEmbeddingHelper(dec\_embeddings,

start\_tokens,

end\_of\_sequence\_id)

predicting\_decoder = tf.contrib.seq2seq.BasicDecoder(dec\_cell,

predicting\_helper,

encoder\_state,

output\_layer)

predicting\_decoder\_output, \_ = tf.contrib.seq2seq.dynamic\_decode(predicting\_decoder,

impute\_finished=True,

maximum\_iterations=max\_target\_sequence\_length)

return predicting\_decoder\_output

def decoding\_layer(

dec\_embed\_input,

dec\_embeddings,

encoder\_state,

vocab\_size,

sequence\_length,

rnn\_size,

num\_layers,

vocab\_to\_int,

keep\_prob,

batch\_size,

max\_target\_sequence\_length

):

def get\_decoder\_cell(rnn\_size):

lstm = tf.contrib.rnn.LSTMCell(rnn\_size)

drop = tf.contrib.rnn.DropoutWrapper(lstm,input\_keep\_prob = keep\_prob)

return drop

with tf.variable\_scope("decoding") as decoding\_scope:

dec\_cell = tf.contrib.rnn.MultiRNNCell([get\_decoder\_cell(rnn\_size) for \_ in range(num\_layers)])

#生成截断正态分布的初始化程序.

#

#这些值与来自 random\_normal\_initializer 的值类似,不同之处在于值超过两个标准偏差值的值被丢弃并重新绘制.这是推荐的用于神经网络权值和过滤器的初始化器.

#stddev：一个 python 标量或一个标量张量,要生成的随机值的标准偏差.

'''

weights = tf.truncated\_normal\_initializer(stddev = 0.1)

'''

#生成张量初始化为0的初始化器.

'''

biases = tf.zeros\_initializer()

'''

#lambda定义一个函数，输入变量为x。

#全连接层,每个输入输出存在连接

'''

output\_fn = lambda x : tf.contrib.layers.fully\_connected(

x,

vocab\_size,

None,

scope = decoding\_scope,

weights\_initializer = weights,

biases\_initializer = biases

)

'''

output\_layer = Dense(vocab\_size,

kernel\_initializer = tf.truncated\_normal\_initializer(mean = 0.0, stddev=0.1))

train\_logits = decoding\_layer\_train(

encoder\_state,

dec\_cell,

dec\_embed\_input,

sequence\_length,

decoding\_scope,

output\_layer,

keep\_prob,

batch\_size,

max\_target\_sequence\_length

)

decoding\_scope.reuse\_variables()

infer\_logits = decoding\_layer\_infer(

encoder\_state,

dec\_cell,

dec\_embeddings,

vocab\_to\_int['<GO>'],

vocab\_to\_int['<EOS>'],

sequence\_length-1,

vocab\_size,

decoding\_scope,

output\_layer,

keep\_prob,

batch\_size,

max\_target\_sequence\_length

)

return train\_logits, infer\_logits

def seq2seq\_model(

input\_data,

target\_data,

keep\_prob,

batch\_size,

sequence\_length,

answers\_vocab\_size,

questions\_vocab\_size,

enc\_embedding\_size,

dec\_embedding\_size,

rnn\_size,

num\_layers,

QA\_word\_to\_ids,

QA\_ids\_to\_vecs,

max\_target\_sequence\_length

):

#一般用于sequence2sequence网络，可完成对输入序列数据的嵌入工作。

'''

enc\_embed\_input = tf.contrib.layers.embed\_sequence(

input\_data,

answers\_vocab\_size+1,

enc\_embedding\_size,

# initializer = tf.random\_uniform\_initializer(0,1)

)

'''

enc\_embed = tf.get\_variable(

'enc\_embed',

shape=(questions\_vocab\_size+1, enc\_embedding\_size),

initializer=tf.constant\_initializer(QA\_ids\_to\_vecs),

#initializer=tf.zeros\_initializer(),

trainable=True #加入训练！！

)

enc\_embed\_input = tf.nn.embedding\_lookup(enc\_embed, input\_data)

'''

##给词向量矩阵 赋值

data = tf.Variable(

tf.zeros([answers\_vocab\_size, enc\_embedding\_size]),

dtype=tf.float32

)

data = tf.nn.embedding\_lookup(ids\_to\_vecs,input\_data)

'''

enc\_state = encoding\_layer(

enc\_embed\_input,

rnn\_size,

num\_layers,

keep\_prob,

sequence\_length

)

dec\_input = process\_encoding\_input(

target\_data,

QA\_word\_to\_ids,

batch\_size

)

#tf.Variable(initializer,name),参数initializer是初始化参数，name是可自定义的变量名称

'''

dec\_embeddings = tf.Variable(

#返回6\*6的矩阵，产生于0和1之间

tf.random\_uniform([questions\_vocab\_size+1,dec\_embedding\_size],0,1)

)

##查找张量dec\_embeddings中的序号为dec\_input的

dec\_embed\_input = tf.nn.embedding\_lookup(dec\_embeddings,dec\_input)

'''

'''

dec\_embeddings = tf.Variable(

tf.zeros([questions\_vocab\_size+1, dec\_embedding\_size]),

dtype=tf.float32

)

dec\_embed\_input = tf.nn.embedding\_lookup(dec\_embeddings,dec\_input)

'''

dec\_embeddings = tf.get\_variable(

'dec\_embed',

shape=(answers\_vocab\_size+1, enc\_embedding\_size),

initializer=tf.constant\_initializer(QA\_ids\_to\_vecs),

#initializer=tf.zeros\_initializer(),

trainable=True #加入训练！！

)

dec\_embed\_input = tf.nn.embedding\_lookup(dec\_embeddings,dec\_input)

train\_logits, infer\_logits = decoding\_layer(

dec\_embed\_input,

dec\_embeddings,

enc\_state,

answers\_vocab\_size,

sequence\_length,

rnn\_size,

num\_layers,

QA\_word\_to\_ids,

keep\_prob,

batch\_size,

max\_target\_sequence\_length

)

return train\_logits,infer\_logits

def question\_to\_seq(question,vocab\_to\_int):

return [vocab\_to\_int.get(word,vocab\_to\_int['<UNK>']) for word in question.split()]

def main():

questions,answers = myReadQAFile()

# stopwords = stopWordsList()

# questions = stopWordsOperation(questions,stopwords)

answers\_jieba = []

questions\_jieba = []

# codes = ['<PAD>','<EOS>','<UNK>','<GO>']

for index in range(0,len(answers)):

answers\_jieba.append(" ".join(jieba.cut(answers[index])))

questions\_jieba.append(" ".join(jieba.cut(questions[index])))

'''

total\_jieba = answers\_jieba + questions\_jieba

total\_train = operation(total\_jieba)

total\_train.append(codes)

total\_train.append(codes)

total\_train.append(codes)

total\_train.append(codes)

total\_train.append(codes)

total\_train.append(codes)

total\_train.append(codes)

total\_train.append(codes)

total\_train.append(codes)

total\_train.append(codes)

'''

answers\_ope = operation(answers\_jieba)

questions\_ope = operation(questions\_jieba)

checkList(questions\_ope,"questions\_ope")

# print(questions\_jieba)

# trainWord2Vec(total\_train)

QAword2vecs = Word2Vec.load(u'allQAvecs\_5\_5\_5\_5.model')

# wordVectors = np.load('QAvecs\_3.model')

# print(wordVectors)

# drugs,diseases,symptoms = readDrDiSy()

#GO与<start>开始标记相同，是第一个提供给解码器的标记，与向量一起，为答案产生标记

#EOS 意思是句尾，一旦解码器产生EOS标记，表示答案已经产生

#UNK 未知标记

#PAD 填充标记

#1, 86

min\_q\_length,max\_q\_length = getLength(questions\_ope)

print(min\_q\_length)

print(max\_q\_length)

questions\_UNK,answers\_UNK = QAAddUNK(questions\_ope,answers\_ope,QAword2vecs)

calculate\_UNK\_ratio(questions\_UNK,answers\_UNK)

sorted\_questions,sorted\_answers = sortQA(questions\_UNK,answers\_UNK,min\_q\_length,max\_q\_length)

QA\_ids,QA\_word\_to\_vecs,QA\_word\_to\_ids,QA\_ids\_to\_vecs = getIdsWtvWtiItv(QAword2vecs)

sorted\_questions\_int,sorted\_answers\_int = getQAint(sorted\_questions,sorted\_answers,QA\_word\_to\_ids)

# checkList(sorted\_questions\_int,"sorted\_questions\_int")

del QA\_word\_to\_vecs

epochs = 50

batch\_size = 64

# length\_of\_vector = 100

rnn\_size = 256

num\_layers = 2

encoding\_embedding\_size = 256

decoding\_embedding\_size = 256

learning\_rate = 0.001

learning\_rate\_decay = 0.9

min\_learning\_rate = 0.0001

keep\_probability = 0.75

tf.reset\_default\_graph()

# start the session

sess = tf.InteractiveSession()

input\_data, targets,lr,keep\_prob,vocab\_table,sequence\_length, max\_target\_sequence\_length = model\_inputs()

'''

sequence\_length = tf.placeholder\_with\_default(

max\_q\_length,

None,

name='sequence\_length'

)

'''

input\_shape = tf.shape(input\_data)

# print(sess.run(enc\_embed\_input))

train\_logits, inference\_logits = seq2seq\_model(

tf.reverse(input\_data,[-1]),

targets,

keep\_prob,

batch\_size,

sequence\_length,

len(QA\_word\_to\_ids),

len(QA\_word\_to\_ids),

encoding\_embedding\_size,

decoding\_embedding\_size,

rnn\_size,

num\_layers,

QA\_word\_to\_ids,

QA\_ids\_to\_vecs,

max\_target\_sequence\_length

)

tf.identity(inference\_logits,'logits')

with tf.name\_scope("optimization"):

#calculating loss function

cost = tf.contrib.seq2seq.sequence\_loss(

train\_logits,

targets,

tf.ones([input\_shape[0],sequence\_length])

)

#using adam optimizer

optimizer = tf.train.AdamOptimizer(learning\_rate)

#performing gradient clipping to handle the vanishing gradient problem

gradients = optimizer.compute\_gradients(cost)

capped\_gradients = [(tf.clip\_by\_value(grad,-5.,5.),var)

for grad, var in gradients if grad is not None]

train\_op = optimizer.apply\_gradients(capped\_gradients)

def batch\_data(

questions,

answers,batch\_size

):

for batch\_i in range(0,len(questions)//batch\_size):

start\_i = batch\_i\*batch\_size

questions\_batch = questions[start\_i:start\_i + batch\_size]

answers\_batch = answers[start\_i:start\_i + batch\_size]

pad\_questions\_batch = np.array(

pad\_sentence\_batch(

questions\_batch,

QA\_word\_to\_ids

)

)

pad\_answers\_batch = np.array(

pad\_sentence\_batch(

answers\_batch,

QA\_word\_to\_ids

)

)

yield pad\_questions\_batch,pad\_answers\_batch

del questions\_UNK,answers\_UNK,sorted\_questions,sorted\_answers,questions\_ope,answers\_ope,answers\_jieba,questions\_jieba,questions,answers

train\_valid\_split = int(len(sorted\_questions\_int)\*0.15)

train\_change = int(len(sorted\_questions\_int)\*0.15)

train\_questions = sorted\_questions\_int[train\_valid\_split:]

print(len(train\_questions))

train\_answers = sorted\_answers\_int[train\_valid\_split:]

valid\_questions = sorted\_questions\_int[:train\_change]

print(len(valid\_questions))

valid\_answers = sorted\_answers\_int[:train\_change]

del sorted\_questions\_int,sorted\_answers\_int

display\_step = 20

stop\_early = 0

stop = 5

validation\_check = ((len(train\_questions))//batch\_size//2)-1

total\_train\_loss = 0

summary\_valid\_loss = []

checkpoint = "./my\_QA\_model.ckpt"

sess.run(tf.global\_variables\_initializer())

for epoch\_i in range(1,epochs +1):

for batch\_i,(questions\_batch,answers\_batch) in enumerate(

batch\_data(train\_questions,train\_answers,batch\_size)):

start\_time = time.time()

\_,loss = sess.run(

[train\_op,cost],

{input\_data:questions\_batch,

targets:answers\_batch,

lr:learning\_rate,

sequence\_length:answers\_batch.shape[1],

keep\_prob:keep\_probability

}

)

total\_train\_loss += loss

end\_time = time.time()

batch\_time = end\_time - start\_time

if batch\_i % display\_step ==0:

print('Epoch {:>3}/{} Batch {:>4}/{} - Loss:{:>6.3f},Second:{:4.2f}'.format(epoch\_i,epochs,batch\_i,len(train\_questions)//batch\_size,total\_train\_loss/display\_step,batch\_time\*display\_step))

total\_train\_loss = 0

#?

if batch\_i % validation\_check == 0 and batch\_i >0:

total\_valid\_loss = 0

start\_time = time.time()

for batch\_ii,(questions\_batch,answers\_batch) in enumerate(

batch\_data(valid\_questions,valid\_answers,batch\_size)):

valid\_loss = sess.run(

cost,

{input\_data:questions\_batch,

targets:answers\_batch,

lr:learning\_rate,

sequence\_length:answers\_batch.shape[1],

keep\_prob:1

}

)

total\_valid\_loss += valid\_loss

end\_time = time.time()

batch\_time = end\_time - start\_time

avg\_valid\_loss = total\_valid\_loss/(len(valid\_questions)/batch\_size)

print('Valid Loss:{:>6.3f},Seconds:{:5.2f}'.format(avg\_valid\_loss,batch\_time))

# reduce learning rate

learning\_rate \*= learning\_rate\_decay

if learning\_rate < min\_learning\_rate:

learning\_rate = min\_learning\_rate

summary\_valid\_loss.append(avg\_valid\_loss)

if avg\_valid\_loss <= min(summary\_valid\_loss):

print("New Record!")

stop\_early = 0

# builder = tf.saved\_model.builder.SaveModelBuilder('./Models/')

# builder.add\_meta\_graph\_and\_variables(sess,[tf.saved\_model.tag\_constants.TRAINING])

# builder.save()

saver = tf.train.Saver()

saver.save(sess,checkpoint)

else:

print("No Imprvement")

stop\_early += 1

if stop\_early == stop:

break

if stop\_early == stop:

print("Stopping Training.")

break

main()

知识图谱构建：

**public** **static** QueryGraph build(List<KbEntity> knownConditions) {

*boundary*.clear();

**final** QueryGraph graph = **new** QueryGraph();

knownConditions.forEach((kbEntity) -> {

//两次查询是因为要得到双向图

//go out

//搜索出与 kbEntity.name 有关的所有三元组的信息

String sql = Config.***KNOWLEDGE\_BASE\_PREFIX***

+ "SELECT ?a ?b ?c WHERE{ example:" + kbEntity.name + " ?a ?b . "

+ "?b rdf:type ?c ."

+ "?c rdf:type owl:Class .}";

KbResults results = KnowledgeBase.*getInstance*().executeQuery(sql);

**while** (results.hasNext()) {

Map<String, String> result = results.next();

KbEntity enode = **new** KbEntity(KnowledgeBase.*handleResult*(result.get("b")),

KnowledgeBase.*handleResultWithOntology*(result.get("c")));

*boundary*.add(enode);

graph.putEdge(kbEntity, enode, KnowledgeBase.*handleResultWithOntology*(result.get("a")));

}

//come back

sql = Config.***KNOWLEDGE\_BASE\_PREFIX***

+ "SELECT ?a ?b ?c WHERE{ ?a ?b example:" + kbEntity.name + " . "

+ "?a rdf:type ?c ."

+ "?c rdf:type owl:Class .}";

results = KnowledgeBase.*getInstance*().executeQuery(sql);

**while** (results.hasNext()) {

Map<String, String> result = results.next();

KbEntity snode = **new** KbEntity(KnowledgeBase.*handleResult*(result.get("a")),

KnowledgeBase.*handleResultWithOntology*(result.get("c")));

*boundary*.add(snode);

graph.putEdge(snode, kbEntity, KnowledgeBase.*handleResultWithOntology*(result.get("b")));

}

});

*expand*(graph, Config.***ALLOW\_QUERY\_STEP\_NUMBER*** - 1);

**return** graph;

}