分类号_____ 论文选题类型_____

U D C _____ 编号_____

華中師範大學

本科毕业论文（设计）

题 目 基于医疗知识图谱的自动问答系统设计与实现

学 院 计算机学院

专 业 计算机科学与技术

年 级 2016 级

学生姓名 林逸

学 号 2016210830

指导教师 何婷婷

二 〇 二 〇 年 四 月

华中师范大学

学位论文原创性声明

本人郑重声明：所呈交的学位论文是本人在导师指导下独立进行研究工作所取得的研究成果。除了文中特别加以标注引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写的成果作品。本人完全意识到本声明的法律后果由本人承担。

学位论文作者签名：  日期：2020 年 5 月 4 日

学位论文版权使用授权书

本学位论文作者完全了解学校有关保障、使用学位论文的规定，同意学校保留并向有关学位论文管理部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权省级优秀学士学位论文评选机构将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

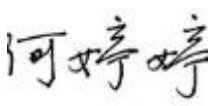
本学位论文属于

1、保密 □ ，在_____年解密后适用本授权书。

2、不保密 □。

（请在以上相应方框内打“ √ ”）

学位论文作者签名：  日期：2020 年 5 月 4 日

导师签名：  日期：2020 年 5 月 4 日

目 录

内容摘要：公共卫生是百姓最关注的话题。而在当前医疗资源分配不均的形势下，一款医疗知识自动问答系统可以有效地缓解资源紧张的局面，提高看病效率。互联网+医疗的尝试也有力地推进了医疗领域的信息化发展。本文研究了基于医疗知识图谱的自动问答系统在开发过程中涉及到的关键技术，并设计实现了一个可以回答十八类不同的医疗方面的问题的系统原型。系统可以“理解”用户输入的问题，并快速、准确地组织答案返回给用户。

论文围绕对基于医疗知识图谱的自动问答系统的研究展开，使用了 Neo4j 图数据库存储知识图谱，使数据的结构更清晰，也更便于查询与管理数据。在自动问答系统的问题分类步骤中，基于 Bi-LSTM 神经网络模型，借助 pytorch 深度学习框架，采用深度学习的方式完成了命名实体识别任务，并获得了较高的正确率。最后用模式匹配的方式生成回答。

本文第一章分析了系统的研究背景与意义。第二章介绍了一些系统在开发过程中涉及到的关键技术，重点介绍了神经网络原理以及 LSTM 模型和 Bi-LSTM 模型的结构和特点。第三章进行了系统功能需求分析。第四章将系统分为知识图谱构建、命名实体识别以及答案生成这三个模块，并分模块详细地介绍了各个模块的功能、设计与实现流程、核心代码和运行效果。论文的最后进行了总结和展望。

关 键 词：知识图谱；图数据库 Neo4j；Bi-LSTM；自动问答系统

Abstract：Public health is the topic that people are most concerned about. In the current situation of unfair distribution of medical resources, an automatic question answering system for medical knowledge can partly solve the resource shortage problem and improve the efficiency of medical treatment. The attempt of medical treatment plus Internet also promotes medical field to be of information to a large extent. This paper studies the key technologies involved in the development of a question answering system based on medical knowledge graph, and designs and implements a system prototype that can answer 18 different kinds of medical questions. The system can "understand" the user's input, organize the answer quickly and accurately and respond to the user.

This paper focuses on the research of an automatic question answering system based on medical knowledge graph. The Neo4j graph database is used to store the knowledge graph, which makes the data structure clearer and makes it more convenient for data query and management. In the question classification steps of the automatic question answering system, with the help of pytorch deep learning framework, the task of named entity recognition is implemented by Bi-LSTM neural network model and the accuracy is high. Finally, the answer is generated by pattern matching.

In this paper, the first chapter analyzes the research background and the significance of the system. The second chapter introduces some key technologies involved in the process of developing the system, with emphasis on the neural network principle and the structure and characteristics of LSTM model and Bi-LSTM model. The third chapter studies the functional requirement of the system. In chapter four, the system is divided into three modules: knowledge graph construction, named entity recognition and answer generation. The text will introduce the function, design and implementation process, core code and operation results of each module in detail. The last part of the paper is conclusion and prospect.

Key words: Knowledge Graph; Neo4j Graph Database; Bi-LSTM; Automatic Question Answering System

第一章 绪论

1.1 研究背景与研究意义

医疗卫生事业是全社会高度关注的话题。然而目前国内的医疗资源主要集中在大型三甲医院，因此，人们都选择去这些医院就诊，这加剧了在这些医院看病难的局面。

随着“互联网+”理念的提出，医疗自动问答系统作为人机交互的一种新方式应运而生。它可以迅速理解患者自然语言的提问，并在短短几秒钟内给用户数百字最相关、最简明扼要的回答，方便、即时、准确地为患者答疑解惑。医疗自动问答系统能有效地缓解医疗资源短缺或是被不合理占用的局面，也能帮助用户快速了解自身健康状况，在去医院就诊时就更有针对性，辅助诊疗的同时带动了医学领域的信息化建设。

自动问答系统作为人工智能的一个实例，需要一个清晰、高效的知识库。而近年来，互联网的迅猛发展带来了呈爆炸式增长的网络数据，且内容呈现出量大、信息类型多元、结构不统一等特点，组织和呈现这些数据需要新的形式。知识图谱将错综复杂的数据转化为清晰的三元组，这不仅使数据更直观地被组织和理解，还聚合整理了大量半结构化知识，实现了知识的快速响应和灵活推理，在精确化的问答服务中起到关键作用。2010 年，苹果公司在 iPhone4S 上推出的个人语音助理 Siri 就是以知识图谱为基础设计并研发成型的[1]。

1.2 研究现状分析

在医学领域，医学信息的传统获取渠道主要是搜索引擎，包括一些专业的健康网站、论坛以及综合搜索引擎中的健康板块等[2]。使用搜索引擎确实可以帮助用户快速地获取大量信息，但是返回的内容存在过量、质量参差不齐且不精准等问题，对于没有专业医疗背景的人来说，筛选出正确且贴切实际的信息困难且费时费力。

1950 年，图灵提出的“图灵测试”被认为是自动问答系统发展的开端 [2]。随后该领域逐渐发展，当前的问答系统主要基于两种技术，一种是基于知识库检索技术，另一种是基于深度学习中的生成技术。

基于知识库检索技术的问答系统常用的方法是先分析提取问题中的关键字，再运用信息检索技术中的倒排索引、文档排序等方式将关键字与定义好的知识库进行模糊匹配，以获取可能相关的答案片段，最后整合语料资源，自动问答系统返回最终答案。1960-1980 年期间的问答系统主要使用此技术，例如 Yahoo 早期的 answer and quora[2]。2011 年，IBM 推出的电脑问答系统 Watson[3-4]也采用了以知识库检索技术为基础，集知识图谱、自动推理、机器学习等技术为一体的研发思路，成为了国外认知计算系统的成功范本[1]。此方法起步早，技术成熟，已经在一些领域有了成熟的应用，但生成的答案有一定的局限性，如：答案必须事先已存在于知识库中，对于答案在知识库以外的问题无法回答。

80 年代以来，问答系统开始结合人工智能并取得突破性进展。基于深度学习中的生成技术的问答系统不依赖于特定的答案库，而是依据从大量语料中习得的语言能力来进行对话，即利用强大的计算和抽象能力自动从海量的数据源中归纳、抽取对解决问题有价值的知识和特征，通过使用生成模型边学习边进行对话。当前基于神经网络的智能回复系统将对话看成是翻译问题[5]，比较主流的生成回答的方法是借助 Sequence to Sequence（Seq2Seq）模型，它分为 Encoder 端和 Decoder 端，即对于句子对<X、Y>,Encoder 对输入句子 X 进行编码,转化为中间语义表示,Decoder 根据中间语义表示生成应答[1]。2015 年，谷歌发表的文献[6]中，采用此模型训练了 2 个 LSTM 中对应的神经网络参数，得到一个对话系统[1]。其后，为了解决生成模型中容易产生通用回复的问题，提出了许多诸如 CopyNet[7]、ECM[8]等的改进模型。基于生成技术的问答系统的回复不再局限于知识库，因此可回答的问题类型更多样。但是由于用生成技术产生的回答存在内容不可控、质量高低不齐等问题，目前使用此方法进行对话的问答系统在医疗方面的应用非常少。

为了兼顾回答的准确性与灵活性，本文设计的系统采用深度学习技术对用户输入的问句进行实体识别，再经过语义分析后，采用知识库检索匹配的方法形成最终的回答。

1.3 研究内容与本文组织结构

本文旨在研究一个基于医疗知识图谱的自动问答系统的设计与实现过程，在此系统中，用户在页面输入想咨询的问题或者问题的关键词，系统分析用户问题，搜

索数据库中存储的数据，并将搜索到的结果组织成回答返回给用户。主要工作围绕以下三个方面展开：

- 将搜集到的数据以知识图谱的形式存入图形数据库 Neo4j 中。
- 基于 Bi-LSTM 模型,对用户输入的问题进行命名实体识别。包括准备数据集、搭建并训练模型、输入测试集测试模型性能、评估测试结果。
- 根据实体识别的结果，完成自动问答系统。包括进行问句类型分类、问句解析并形成 cypher 数据库查询语句、查询数据库并根据查询结果形成最终答案。

第二章 关键技术

2.1 神经网络训练的基本过程

训练一个神经网络的过程是：通过前向传播的计算，反向传播的反馈，迭代地调整模型参数，优化模型，最终得到使损失函数（L（w，b））取得全局最小值的参数 w、b。损失函数是用来衡量模型的输出与标准答案之间的偏差的量化指标。图 2.1 显示了一个神经网络的结构。

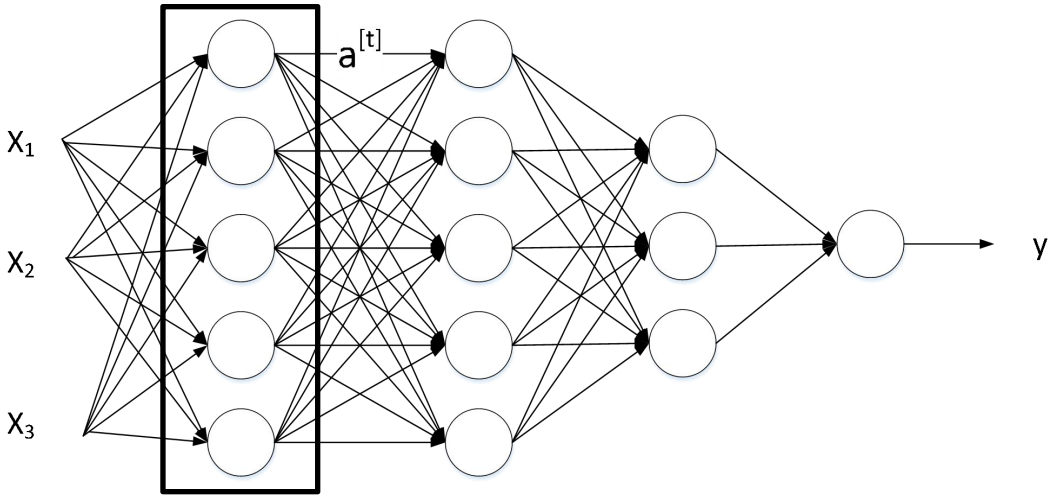


图 2.1 神经网络结构图

图 2.1 框中的一层中的前向传播可描述为（公式 1）（公式 2）。

$$z^{[t]}=w^{[t]}a^{[t-1]}+b^{[t]} \quad \text{（公式 1）}$$

$$a^{[t]}=g^{[t]}(z^{[t]}) \quad \text{（公式 2）}$$

（公式 1）和（公式 2）中，上标[t]意为第 t 层，$w^{[t]}$、$b^{[t]}$为需要调整的第 t 层的参数，$a^{[t-1]}$为第 t-1 层的内部输出，作为第 t 层输入，$a^{[t]}$为第 t 层通过计算产生的内部输出，$g^{[t]}$为第 t 层使用的非线性激活函数，作用于线性计算结果 $z^{[t]}$，以避免神经网络只计算线性激活函数，造成隐藏层不起作用，也可以用于控制输出限于一定范围内。激活函数常用 ReLU 函数，sigmoid 函数，tanh 函数。

反向传播用于传播误差，是寻找损失函数最小值的过程。反向传播的依据是梯度下降法，它是一种解决最优解问题的算法，也是训练神经网络的基础。在训练过程中先选择一组初始的参数值，然后根据当前各参数基于损失函数的偏导数调整各参数，重复前向传播与反向传播两个过程，直至得到合适的参数使得损失函数取得最小值。

2.2 Bi-LSTM 神经网络

一个 Bi-LSTM（Bi-directional long short term memory，双向长短期记忆）神经网络的结构如图 2.2 所示。Bi-LSTM 神经网络是由双向的 LSTM（long short term memory，长短期记忆）神经网络构成的，每个方向都是一个完整的 LSTM 过程，有正向传播和反向传播。LSTM 是 RNN（recurrent neural network，循环神经网络）的一种特殊形式，因此它也适合处理序列问题。而选择 LSTM 代替传统 RNN 模型的主要原因是：传统的 RNN 神经网络模型存在梯度消失的问题，因为随着时间间隔的不断增大，RNN 只能学习到短周期的依赖关系，而会丧失学习到远距离信息的能力[9]。

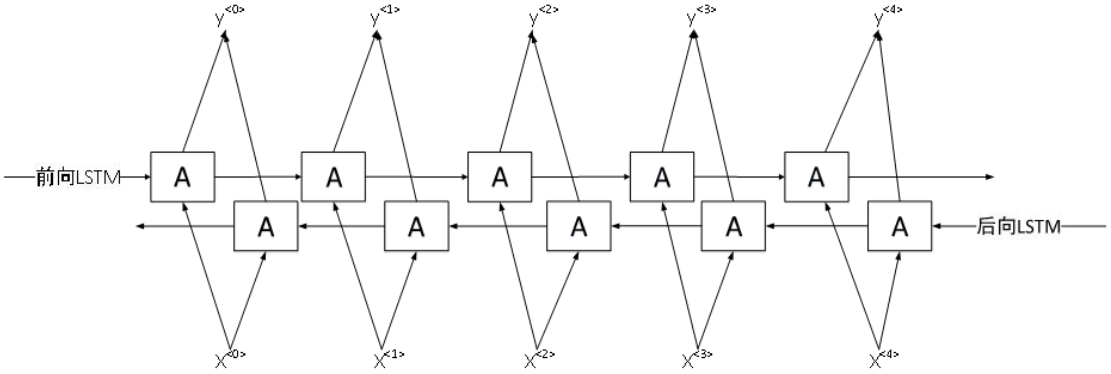


图 2.2 Bi-LSTM 神经网络结构图

LSTM 对 RNN 的改进是通过增加一个记忆细胞结构和三个门控结构实现的，这在一定程度上解决了 RNN 存在的梯度消失问题，用记忆细胞存储输入的历史信息，用门控制历史信息的保留程度，通过训练过程可以学到记忆哪些信息和遗忘哪些信息。这种机制能够记住历史信息，从而能更好地传播输入数据之间的关联性，捕捉到较长距离的依赖关系。

LSTM 单元结构如图 2.3 所示。此单元的输入为：$c^{\langle t-1\rangle}$，$a^{\langle t-1\rangle}$，$x^{\langle t\rangle}$，输出为：$c^{\langle t\rangle}$，$y^{\langle t\rangle}=a^{\langle t\rangle}$，其中，$a^{\langle t\rangle}$为第 t 层输出的激活值，$c^{\langle t\rangle}$为第 t 层的记忆细胞，$G_f$为遗忘门，以一定概率控制是否遗忘上一层的细胞状态，G_u为更新门，以一定概率控制记忆细胞候选值的保留程度，G_f和 G_u共同控制更新记忆细胞，G_o为输出门。各部分具体计算如下：

- **计算遗忘门 G_f** $G_f = \sigma(W_f [a^{\langle t-1\rangle}, x^{\langle t\rangle}] + b_f)$ （公式 3）
- **计算更新门 G_u** $G_u = \sigma(W_u [a^{\langle t-1\rangle}, x^{\langle t\rangle}] + b_u)$ （公式 4）
- **计算记忆细胞的候选值 $c_N^{\langle t\rangle}$** $c_N^{\langle t\rangle} = \tanh(W_c [a^{\langle t-1\rangle}, x^{\langle t\rangle}] + b_c)$ （公式 5）

- **更新记忆细胞 $c^{\langle t\rangle}$** $c^{\langle t\rangle} = G_f \times c^{\langle t-1\rangle} + G_u \times c_N^{\langle t\rangle}$ （公式 6）
- **计算激活值 $a^{\langle t\rangle}$** $a^{\langle t\rangle} = G_o \times \tanh(c^{\langle t\rangle})$ （公式 7）

其中，tanh 和 σ（sigmoid）为激活函数，σ 函数多用于二元分类，函数值介于 0-1 之间，（公式 8）（公式 9）分别为 tanh 函数和 σ（sigmoid）函数：

$$y = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \quad \text{（公式 8）}$$

$$y = \frac{1}{1+e^{-x}} \quad \text{（公式 9）}$$

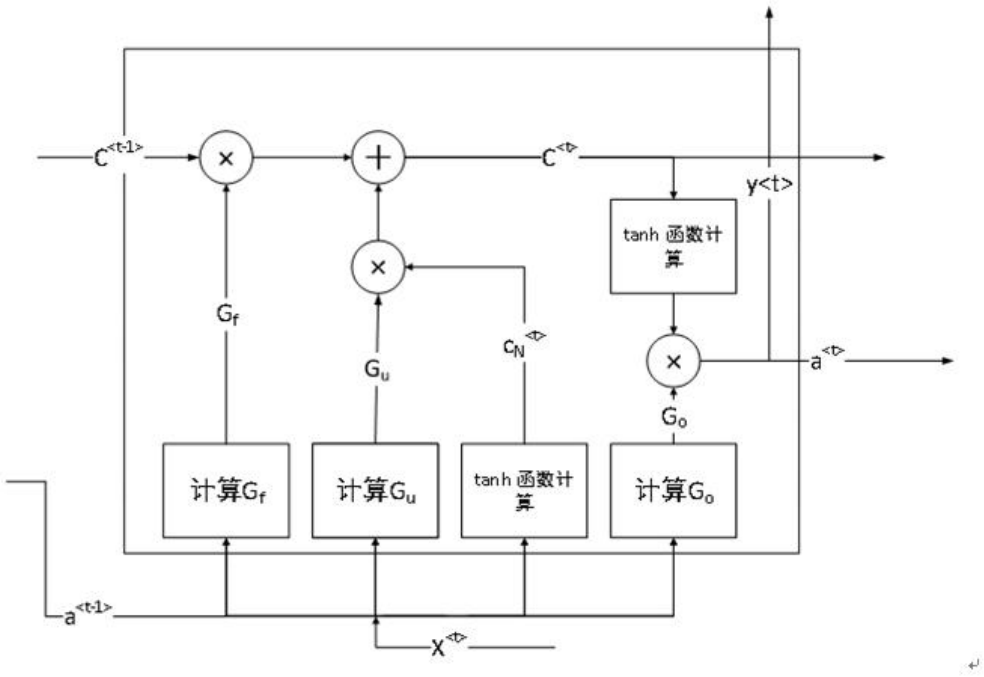


图 2.3 LSTM 结构图

单向 LSTM 中信息的传递方向单一，在分析 t 时刻的输出时，只能利用 t 时刻之前句子的输入信息和激活值，而无法参考后文的信息，这会使命名实体识别的结果不准确。比如：新型冠状病毒肺炎，这是一个疾病实体，如果仅考虑上文信息，可能被识别成为一个病毒实体。而 Bi-LSTM 神经网络分别从 2 个方向分析句子，综合这两个方向的输出得出本神经网络的输出结果，通过这种方式实现了对句子上下文信息的充分利用。所以本文使用 Bi-LSTM 模型以更好地捕捉双向的语义依赖。

2.3 深度学习框架 Pytorch

为了避免深度学习的研究者做实验时重复写大量代码，一些研究员整合了这些基础代码形成了深度学习框架，Pytorch 是被广泛使用的框架之一。

Pytorch 是一个开源的、以 python 优先的机器学习库，它提供两个高级功能：①支持 GPU 加速，提高了神经网络的训练速度，②支持带反馈的动态神经网络，这是一些其他主流框架如 TensorFlow 不支持的。Pytorch 提供的自动求导功能，可以自动求出反向传播中需要的导数，且高效快速，简洁，易入门，适合做小型研究。因此，本文借助 Pytorch 实现 Bi-LSTM 神经网络模型，用于完成命名实体识别任务。

2.4 知识图谱

知识图谱被称为知识域可视化，用可视化技术挖掘、分析、构建和显示知识及他们之间的相互联系。它能把与关键词相关的知识系统化地展示给用户，从而能为学科研究提供有价值的参考。

知识图谱中，知识通常可以表示为一个三元组，如用（实体 1，关系，实体 2）、（概念，属性，属性值）这样的三元组来表示知识。实体表示具体的研究对象，如感冒，关系刻画了连接的两个实体之间的关联，如并发症。概念主要指对象类型、事物种类等，如疾病。属性指概念可能具有的特征、特性及参数，如疾病的名称，概念指定属性的值为属性值，刻画了概念的内在特性[10]。

知识图谱本质上是在用图形的形式组织和呈现知识，用结点表示实体，用边表示关系。本系统中创建了一个的简单医疗知识图谱，图 2.4 展示了一个知识图谱局部示例图，图中包含了“急性鼻炎”、“中耳炎”、“鼓膜穿孔”三个疾病类实体结点和一个“耳鼻喉科”科室类实体结点，两类结点之间存在疾病“属于”科室的关系，而疾病“急性鼻炎”、 “鼓膜穿孔”与疾病“中耳炎”之间是“并发症”的关系。

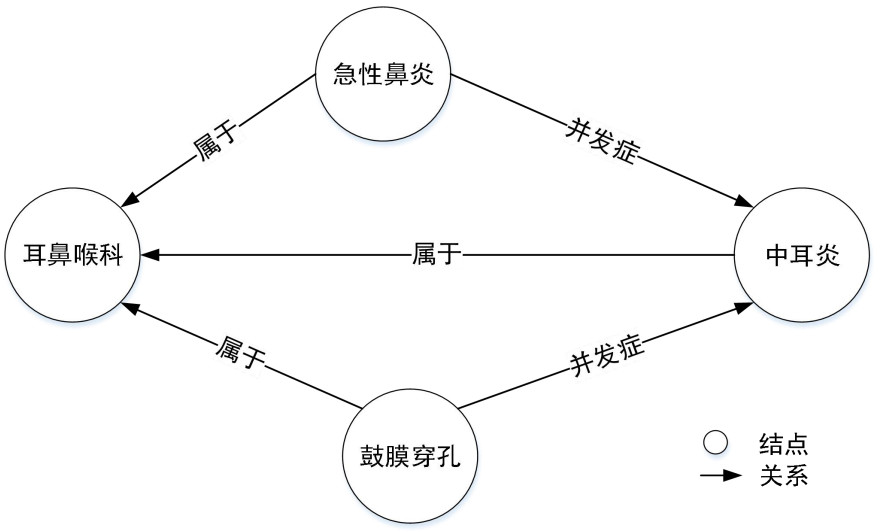


图 2.4 知识图谱局部示例图

2.5 图数据库 Neo4j

知识图谱需使用图数据库进行存储。图数据库用图结构来存储数据，善于处理大量关系数据。Neo4j 是一个高性能的图形数据库，可用 Neo4j 数据浏览器界面执行命令并显示查询结果。其存储的图是由结点（代表实体）和连接结点的边（代表关系）构建的有向图或无向图。每个实体可以用一个全局唯一确定的 ID 来标识，结点和边都可以设置属性。

Neo4j 支持完整的 ACID（原子性，一致性，隔离性和持久性）规则，可以被 Java，Scala 等编程语言访问，也可以将查询到的数据导出成 JSON 或者 XLS 格式。因此本系统使用 Neo4j 存储知识图谱。

Neo4j 使用 Cypher 语句查询数据库。 Cypher 是一个描述性的图形查询语言，功能强大，查询效率高，作用和关系型数据库中的 SQL 语言相当。Cypher 语句可以管理结点、关系和属性，也可以通过模式匹配搜索信息或修改数据。Cypher 查询语句主要包含以下几个常用命令：

- START：在图中的开始点，一般通过元素的 ID 或索引查找获得
- MATCH：图形的匹配模式，束缚于开始点
- WHERE：过滤条件
- RETURN：返回需要的信息

Cypher 语言使用圆括号“()”表示节点，使用“-[]-”表示无方向的关系，使用“-[]->”表示有方向的关系，还提供了聚合函数实现聚类功能，如：求和 sum，平均值 avg，最小/大值 min/max，相异 distinct 去掉值中重复数据等，还可以通过正则表达式进行匹配。

第三章 功能需求分析

本文研究了一个基于医疗知识图谱的自动问答系统设计与实现，该系统中，用户可以在终端以问句或关键词的形式输入想咨询的问题，系统分析用户输入问句的含义、搜索数据库中存储的数据，最终组织出一段回答返回给用户。

通过网络调查和交流访问，思考总结出了七类实体词，如：疾病、药物、症状等，再添加一类否定词以丰富数据，一共形成了八类实体词。再根据实体的属性、实体间的关系、以及患者常见的提问方法，设计了若干类疑问词，比如:“为什么会”、“怎样才”等。综合实体和疑问词两个因素，最终总结了十八类问答系统可回答的问题类型。问题种类涵盖了大部分医疗领域常见的问题类型，若问句不属于任一个具体的问句类型，则可以抽取问句中的疾病或者症状实体，返回相关描述信息。表3.1罗列了系统可回答的问句类型及其含义。

表 3.1 系统可回答问题类型表

问句类型	含义	问句范例
disease_symptom	患某种疾病，会出现哪些症状	感冒的症状有哪些？
symptom_disease	出现某些症状，可能患何种疾病	最近为什么老流鼻涕？
disease_cause	疾病的原因	为什么会感冒？
disease_acompany	疾病的并发症	感冒有哪些并发症？
disease_not_food	疾病需忌口的食物	感冒的人不能吃什么？
disease_do_food	疾病建议吃什么食物	感冒的人吃什么比较好？
food_not_disease	某食物对什么病不好	哪些人最好不要吃冰淇淋？
food_do_disease	某食物对什么病有好处	生姜茶有哪些好处？
disease_drug	某病要吃什么药	感冒要吃什么药？
drug_disease	某药品能治哪种病	新康泰克能治什么病？
disease_check	疾病需要做什么检查	感冒怎么才能查出来？

check_disease	检查能查出什么病	血常规能查出什么？
disease_prevent	预防措施	如何预防感冒？
disease_lasttime	治疗周期	感冒要多久才能好？
disease_cureway	治疗方式	如何治疗感冒？
disease_cureprob	治愈概率	感冒能治好吗？
disease_easyget	易感人群	什么样的人易感冒？
disease_desc	疾病描述	感冒

第四章 系统设计与实现

4.1 系统结构设计

基于医疗知识图谱的自动问答系统分为三个主要模块，分别为：知识图谱构建模块、命名实体识别模块和答案生成模块，系统结构如图 4.1 所示。

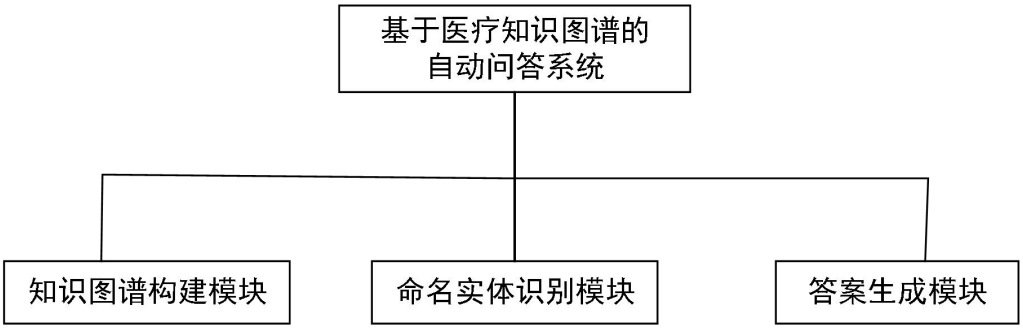


图 4.1 系统总结构图

知识图谱构建模块根据预先定义的知识图谱结构从收集到的数据中提取出相关实体、属性和关系，并将其结构化地存储到图数据库中。

命名实体识别模块主要完成实体识别任务。借助 Bi-LSTM 模型，采用深度学习的方法，识别出用户输入的自然语言问句中包含的实体类别和实体名，这有助于答案生成模块中问句分类步骤的实现。

答案生成模块借助知识图谱和命名实体识别的结果,通过对输入语句进行分类、分析语句含义、形成数据库查询语句、搜索图数据库等步骤，最终生成自动问答系统返回给用户的答案。

4.2 知识图谱构建模块设计与实现

4.2.1 模块功能

知识图谱构建模块负责从结构化数据中提取知识图谱需要的实体、属性、关系信息，并将数据存储到图数据库 neo4j 中。此模块实现后，用户可以在 Neo4j 数据浏览器界面中直观地看到建好的知识图谱。

4.2.2 模块开发环境

neo4j-community-4.0.0

Neo4j 图数据库是全球领先的图数据库平台软件，并同时为开源社区和商业应用提供不同版本。考虑到本文只对图数据库做简单的应用，数据规模不大，功能需

求简单，选择 Neo 4j Community 社区版足以满足本文研究需求，它包含了图数据库引擎、cypher 分析编译器、Neo4j Browser、cyper-shell 命令行工具等，可以运行在多数操作系统上，并且版本安全性高，操作灵活，支持多数据库和分布式查询。

Pycharm Community Edition 2019.3.3

Pycharm 是由 JetBrains 公司打造的一款 Python 集成开发环境，提供编译调试 python 代码、语法高亮、项目管理、代码跳转、智能提示等功能。本原型系统开发的过程中不需要用到一些多余的插件，因此本文选择 Pycharm 社区版。

4.2.3 知识图谱结构设计

初始数据来源于他人收集并公开的疾病数据，数据为 json 格式，图 4.2 展示了一个具体的数据样例格式。


```
{"_id" : { "$oid" : "5bb578b6831b973a137e3ee6" },
"name" : "肺泡蛋白质沉积症",
"desc" : "肺泡蛋白质沉积症(简称PAP)，又称Rosen-Castle-man-Liebow综合征，是一种罕见疾病。
        该病以肺泡和细支气管腔内充满PAS染色阳性，来自肺的富磷脂蛋白质物质为其特征，好发于青中年，男性发病约3倍于女性。",
"category" : [ "疾病百科", "内科", "呼吸内科" ],
"prevent" : "1、避免感染分支杆菌病，卡氏肺囊肿肺炎，巨细胞病毒等。\n2、注意锻炼身体，提高免疫力。",
"cause" : "病因未明，推测与几方面因素有关：如大量粉尘吸入（铝，二氧化硅等），机体免疫功能下降（尤其婴幼儿），
          遗传因素，酗酒，微生物感染等，而对于感染，有时很难确认是原发致病因素还是继发于肺泡蛋白沉着症",
"symptom" : [ "紫绀", "胸痛", "呼吸困难", "乏力", "毓卓" ],
"yibao_status" : "否",
"get_prob" : "0.00002%",
"get_way" : "无传染性",
"acompany" : [ "多重肺部感染" ],
"cure_department" : [ "内科", "呼吸内科" ],
"cure_way" : [ "支气管肺泡灌洗" ],
"cure_lasttime" : "约3个月",
"cured_prob" : "约40%",
"cost_money" : "根据不同医院，收费标准不一致，省市三甲医院约（8000——15000 元）",
"check" : [ "胸部CT检查", "肺活检", "支气管镜检查" ],
"recommand_drug" : [],
"drug_detail" : [] }
```


图 4.2 初始数据结构图

考虑到大部分用户提出的问题都围绕疾病展开，并且数据中，疾病相关的信息最全面，也最容易获得，同时为了更好地存储疾病相关的信息，本文在设计知识图谱的结构时仅设计了疾病相关的属性。最终构建的知识图谱共包含约 4.4 万实体，30 万实体关系，共设计了 7 类实体，9 类关于疾病实体属性以及 11 类实体关系。

实体类型

医疗知识图谱包含 7 种实体类型，表 4.1 列举了实体类型的名称，含义和范例。

表 4.1 实体类型表

实体类型	含义	举例
Check	检查项目	妇科超声检查；糖化血红蛋白

Department	科室	肿瘤外科；儿科；消化内科
Disease	疾病	肺炎；感冒
Drug	药品	青霉胺片；盐酸氯米帕明片
Food	食物	黑木耳粥；小米粉粥； 小麦面粉；鸭脖
Producer	已有厂家生产出的具体在售的药品	康福药业人工牛黄甲硝唑胶囊； 山东仁和堂盐酸地芬尼多片
Symptom	症状	发烧；持物不稳；伤口愈合发痒

疾病实体属性

医疗知识图谱包含 9 种疾病实体属性，表 4.2 列举了疾病类实体的属性名、属性值及其含义。

表 4.2 疾病实体的属性表

属性类型	含义	举例
name	疾病名称	心尖肥厚型心肌病
desc	疾病简介	属于原发性肥厚型心肌病中的特殊类型…
cause	疾病病因	肥厚型心肌病是由于胎儿发育中的心肌对循环中儿茶酚胺反应异常…
prevent	预防措施	在肥厚型心肌病的预后因素中，合理的生活指导是有利的因素…
cure_lasttime	治疗周期	30 天
cure_way	治疗方式	手术治疗
cure_department	治疗科室	心内科
cured_prob	治愈概率	10%
easy_get	易感人群	无特定人群

实体关系

医疗知识图谱中，不同类型的实体间存在 11 类关系，大部分实体间关系应为双向的，例如“common_drug”关系的含义为“某疾病常用某类药”，应该有与其对应的反向关系“某类药适用于某疾病”存在，但是本文的研究中简化了这一步骤，仅设计了以疾病为中心的单向关系。表 4.3 列举了所有关系类型的名称，含义和示例。

表 4.3 实体关系类型表

实体关系类型	含义	举例
belongs_to	疾病所属科室	〈妇科，属于，妇产科〉
common_drug	疾病常用药品	〈脊柱骨折，常用，阿仑膦酸钠片〉
do_eat	疾病宜吃食物	〈卵巢病，宜吃，小麦米粉〉
drugs_of	某类药品对应市面上在售的具体药品	〈克拉霉素胶囊，在售，亚太药业克拉霉素胶囊〉
need_check	疾病需要做的检查	〈贫血，需做检查，血清不饱和铁结合力〉
no_eat	疾病忌口的食物	〈肝结石，忌吃，红辣椒〉
recommand_drug	疾病推荐的药物	〈肩周炎，推荐用药，奥沙普秦肠溶胶囊〉
recommand_eat	疾病推荐的食谱	〈股骨头坏死，推荐食谱，鱿鱼蛋菇汤〉
has_symptom	疾病症状	〈腺性唇炎，疾病症状，上唇肥厚〉
acompany_with	疾病的并发症	〈十二指肠壅积症，并发疾病，营养不良〉
department	科室之间的附属关系	〈心胸外科，属于，外科〉

4.2.4 模块实现流程及核心代码

知识图谱构建模块的关键实现步骤为数据预处理、创建疾病实体结点、创建除疾病外其他实体结点和创建实体关系边，它们之间的顺序如图 4.3 所示。

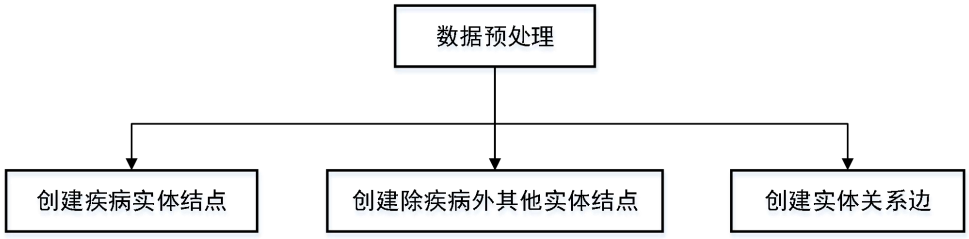


图 4.3 知识图谱建立流程图

数据预处理

读取 json 文件，遍历数据，从每个单元数据结构中分离出实体、属性和关系的信息，将信息分类存入相应数组中，有助于后续创建实体结点和关系边。实体存入以实体名命名的实体数组中，如 diseases，drugs 等，关系存入 rel 关系数组中，如 rels_doeat，rels_acompany 等，代码如图 4.4 所示。

```
if 'symptom' in data_json:
    symptoms += data_json['symptom']
    for symptom in data_json['symptom']:
        rels_symptom.append([disease, symptom])

if 'acompany' in data_json:
    for acompany in data_json['acompany']:
        rels_acompany.append([disease, acompany])
```

图 4.4 实体、关系数据存入数组代码图

疾病的属性存入 disease_dict 结构体中，代码如图 4.5 所示。

```
if 'desc' in data_json:
    disease_dict['desc'] = data_json['desc']

if 'prevent' in data_json:
    disease_dict['prevent'] = data_json['prevent']

if 'cause' in data_json:
    disease_dict['cause'] = data_json['cause']

if 'get_prob' in data_json:
    disease_dict['get_prob'] = data_json['get_prob']

if 'easy_get' in data_json:
    disease_dict['easy_get'] = data_json['easy_get']
```

图 4.5 属性存入结构体代码图

创建疾病类实体结点

因为疾病类实体存在多类属性，所以需要单独创建疾病类实体结点。在创建疾病类实体结点步骤中，先为每个疾病实体创建一个 node 结构预存放 disease_dict 中的所有疾病类实体结点的属性值，然后调用 create 函数在图数据库中创建结点，代码如图 4.6 所示。

```
def create_diseases_nodes(self, disease_infos):
    for disease_dict in disease_infos:
        node = Node("Disease", name=disease_dict['name'], desc=disease_dict['desc'],
                    prevent=disease_dict['prevent'],cause=disease_dict['cause'],
                    easy_get=disease_dict['easy_get'],cure_lasttime=disease_dict['cure_lasttime'],
                    cure_department=disease_dict['cure_department']
                    ,cure_way=disease_dict['cure_way'], cured_prob=disease_dict['cured_prob'])
        self.g.create(node)
    return
```

图 4.6 创建疾病类实体代码图

创建其他实体结点

其他实体结点除实体类别和实体名外，不存在其余属性。因此在创建其他实体结点步骤中，依据事先处理好的实体数组，创建 node 存放结点类型和结点名，再调用 create 函数在图数据库中创建结点，代码如图 4.7 所示。

```
def create_node(self, label, nodes):
    for node_name in nodes:
        node = Node(label, name=node_name)
        self.g.create(node)
    return
```

图 4.7 创建其他实体代码图

创建关系边

按照处理好的实体关系数组，得到要创建的实体边所连接的两个实体，据此得到 cypher 语句 query，调用 run 函数执行 cypher 语句在图数据库中生成关系边，过程如图 4.8 所示。

```
def create_relationship(self, start_node, end_node, edges, rel_type, rel_name):
    set_edges = []
    for edge in edges:
        set_edges.append('###'.join(edge))
    all = len(set(set_edges))
    for edge in set(set_edges):
        edge = edge.split('###')
        p = edge[0]
        q = edge[1]
        query = "match(p:%s),(q:%s) where p.name='%s'and q.name='%s' create (p)-[rel:%s{name:'%s'}]->(q)" % (
            start_node, end_node, p, q, rel_type, rel_name)
        try:
            self.g.run(query)
```

图 4.8 创建关系边代码图

4.2.5 知识图谱展示

输入网址“http://localhost:7474/browser/”，可查看构建好的知识图谱，如图 4.9 为知识图谱结构展示图。图中“Node Labels”一栏是所有实体类型名，“Relationship Types”一栏是所有关系类型名，“Property Keys”一栏是所有疾病类型的属性名。

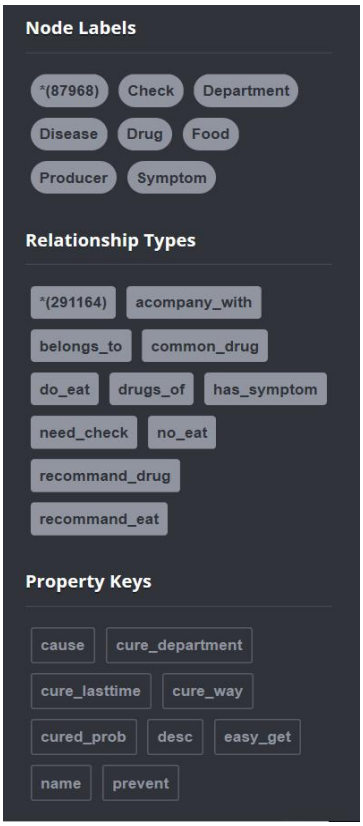


图 4.9 知识图谱结构展示图

以疾病实体类型和并发症关系为例，构建的知识图谱的局部图如图 4.10 所示，图中的每个结点代表疾病实体，每个有向边代表并发症关系（acompany_with）。

图 4.10 知识图谱中并发症关系局部图

4.3 命名实体识别模块设计与实现

4.3.1 系统功能

命名实体识别模块的功能是借助事先训练好的模型，标注出输入模块的一句问句中包含的实体词。比如：“板蓝根可以治疗感冒吗？”，将这句话输入本模块后，需要识别出“板蓝根”为药品实体，“感冒”为疾病实体，识别出的实体，将作为问答系统后续判断输入的问句属于何种问题类型的重要依据。

4.3.2 基于深度学习的命名实体识别的优点

识别句子中的实体有一种传统的方法是将所有实体名词和其所属类型存入一个词典中，每当需要识别一个句子中的实体时，就遍历一遍词典，匹配出句子中存在的实体词。但随着实体的数量增加，这种方式会使得词典占用更多的存储空间，遍历词典所需的时间也更多。除此之外，深度学习可以利用更少的数据识别出更多的实体。匹配时首先需要将所有实体都存入词典，如果句子中出现词典中不存在的实体，则无法识别出新实体。而经过训练，深度学习模型可以根据实体在句中出现的位置，识别出训练数据中不存在的实体。因此本文选择使用深度学习的方式进行命名实体识别。

4.3.3 数据处理与规模

为了获得训练数据，本文采用模板替换的方式将设计的模板句替换成包含实体词的训练数据。设计的句子模板如表 4.4 所示，这些模板以系统可回答的 18 类问题为参考，以实体类型为标准，覆盖到每种实体类型，其中“_”为替换实体词的位置，用在知识图谱中存放的相应类型的实体名称词替换。

表 4.4 扩展句子模板表

Disease	_的症状有哪些？ 为什么会得_？ 患_的人要吃什么药？ _有哪些并发症？ 患_的人不能吃什么？ 患_的人吃什么比较好？ 怎样才能预防_？ _要多久才能好？

	要怎么治疗？ 什么样的人易得？ _治好的概率有多大？ _需要做哪些检查？ _应该去哪个科室看？
Drug	_能治什么病？
Department	_能看什么病？
Food	_有哪些好处？ 哪些人不能吃_？
Producer	_能治什么病？
Check	_能查出什么病？
Symptom	为什么会_？

获得训练数据后，需要对数据分词并加上标签，最终获得的数据格式如表 4.5 所示，其中标签 BMESO 的含义如表 4.6 所示。

表 4.5 数据格式

为	什	么	会	得	感	冒	？
O	O	O	O	O	B-disease	E-disease	O

表 4.6 标签 BOMES 含义表

B	M	E	S	O
实体名的开始	实体名的中间	实体名的结尾	单字实体	其它无意义字

处理后的数据共有 21MB，80%为训练集，10%为验证集，10%为测试集。训练集用于训练模型，选出最优模型；测试集用于评估训练好的模型的性能；为了避免测试模型性能的过程中出现数据提前泄露的问题，即模型在进行测试之前就已经知道了要测试的数据的内容，设置了验证集以调试模型。

4.3.4 命名实体识别流程

对于给定的一句问句 $S=\{s_1,\cdots,s_i,\cdots,s_n\}$，其中$s_i$表示第 i 个字，需要得到这句话对应的标签序列 $T=\{t_1,\cdots,t_i,\cdots,t_n\}$，其中$t_i$表示第 i 个字所对应的标签。本文采用一个 Bi-LSTM 模型完成输入句的实体识别任务，流程如图 4.11 所示。

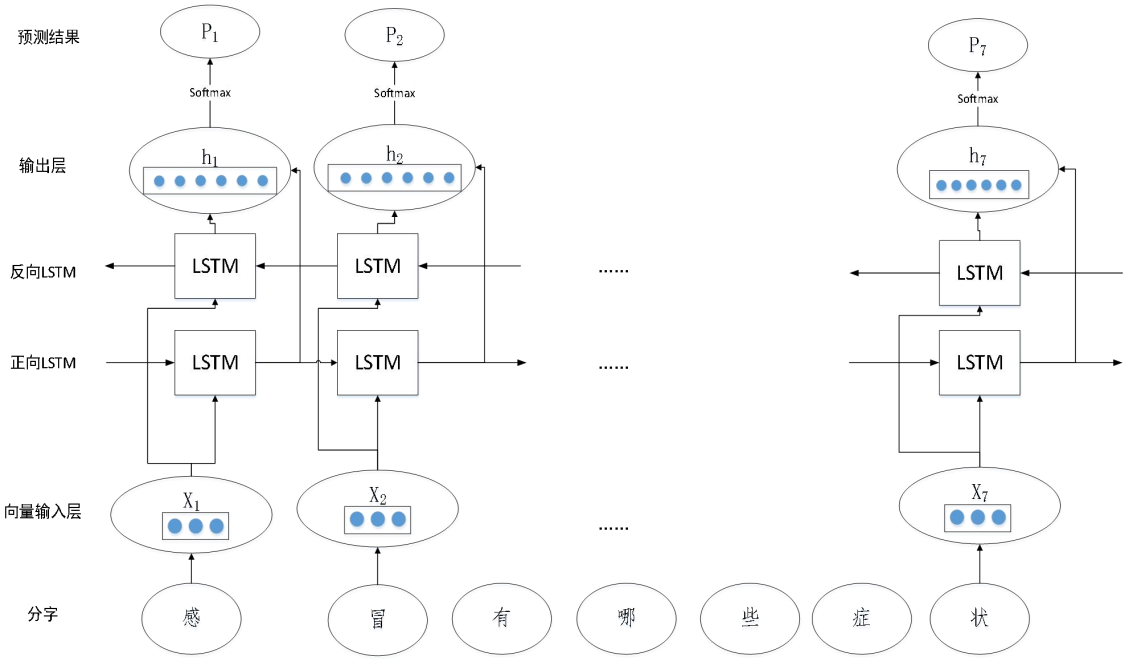


图 4.11 Bi-LSTM 实现命名实体识别流程图

先将句子分成字，再得到每个字对应的向量表示，将向量序列输入到训练好的 Bi-LSTM 模型中，经神经网络处理后，拼接正向 LSTM 的输出向量与反向 LSTM 的输出向量得到模型的输出，将此结果通过 softmax 层处理，计算出每个字对应所有标签的概率，最终为每个字选出它对应的概率最大的标签。

字向量化

系统定义了一个字-向量表，其中的向量是特征向量。当输入的句子包括 n 个字（$S=\{s_1,\cdots,s_i,\cdots,s_n\}$, s_i表示第 i 个字）时，使用映射函数 F(s_i)获取不同字对应的向量，就得到了此句子对应的向量序列 $X=\{x_1,\cdots,x_i,\cdots,x_n\}$，$x_i$表示句中第 i 个字对应的向量。

命名实体识别

得到向量序列后将向量送入训练好的神经网络中进行测试，此过程中使用的公式为：

$$\vec{h}_i=\overrightarrow{LSTM}(x_i) \quad \text{（公式 10）}$$

$$\overleftarrow{h}_i=\overleftarrow{LSTM}(x_i) \quad \text{（公式 11）}$$

$$h_i=[\vec{h}_i;\overleftarrow{h}_i] \quad \text{（公式 12）}$$

其中，x_i表示每个字s_i对应的向量，$\overrightarrow{h_i}$，$\overleftarrow{h_i}$表示在 Bi-LSTM 模型中第 i 个时间步上两个方向上的输出，拼接$\overrightarrow{h_i}$，$\overleftarrow{h_i}$得到h_i作为s_i的语义表示。再将h_i通过 softmax 函数进行计算，

$$P_i = softmax(Wh_i + b) \quad \text{（公式 13）}$$

对神经网络的输出归一化，得到预测概率P_i，W 和 b 是可学习参数。对应到最大概率的那个标签，即为预测出的字s_i对应的标签t_i。

在训练模型时采用交叉熵作为损失函数，该函数为凸函数，求导时能够得到全局最优值，常用于分类问题中。

4.3.5 模型测试结果分析

本文采用3种在自然语言处理领域常用的评价指标评估模型，分别为准确率（P）、召回率（R）和 F1 值（F1）。具体的公式为：

$$P = \frac{T_p}{T_p+F_p} \times 100\% \quad \text{（公式 14）}$$

$$R = \frac{T_p}{T_p+F_a} \times 100\% \quad \text{（公式 15）}$$

$$F1 = \frac{2PR}{P+R} \quad \text{（公式 16）}$$

式中，T_p 为模型识别到的相关的标签的个数，F_p 为模型识别到的不相关的标签的个数，F_a 为相关的标签，但是模型没有识别到的个数[9]。

测试结果如表 4.7 所示，表中标签栏列举出标签类型名，标签名代表实体词中某个字属于的实体类别以及该字在实体词中出现的位置，如：E-disease，意为该字是某疾病类型实体名的最后一个字，类似“感冒”中的“冒”字。

表 4.7 测试结果表

标签	P	R	F1
B-disease	0.9968	1.0000	0.9984
M-disease	1.0000	0.9993	0.9996
E-disease	1.0000	0.9997	0.9999
S-disease	0.9231	1.0000	0.9600
B-department	1.0000	1.0000	1.0000
M-department	1.0000	1.0000	1.0000

E-department	1.0000	1.0000	1.0000
B-food	0.9949	1.0000	0.9974
M-food	1.0000	0.9982	0.9991
E-food	1.0000	1.0000	1.0000
B-drug	0.8980	0.9401	0.9186
M-drug	0.8682	0.9200	0.8933
E-drug	0.8798	0.9531	0.9150
B-check	0.9739	1.0000	0.9868
M-check	1.0000	0.9949	0.9974
E-check	1.0000	1.0000	1.0000
B-symptom	1.0000	1.0000	1.0000
M-symptom	1.0000	0.9995	0.9998
E-symptom	1.0000	1.0000	1.0000
B-producer	0.9865	0.9791	0.9828
M-producer	0.9901	0.9822	0.9861
E-producer	0.9893	0.9704	0.9798
O	1.0000	1.0000	1.0000
avg/total	0.9976	0.9975	0.9975

数据显示，平均正确率、召回率、F1 值均达到了 99%以上。除“drug”相关的标签外，其余标签识别的准确率全达到了 97%以上。“department”实体识别效果最好，准确率和召回率都达到了 100%，而“drug”类实体的识别准确率最低，未达到 90%，可能的原因是包含“drug”类实体的训练数据较少。疾病是本系统中最重要也是最常用到的一类实体，结果显示，除单字“disease”类实体外，其余“disease”相关标签的识别准确率、召回率和 F1 值都达到了 99%以上，这提高了系统在实际应用中的可靠性，单字的“disease”类实体在实际应用中也是不常见的，所以在训练集中这类数据也较少，造成识别率较低。总体来说，该模型在本次命名实体识别任务中表现良好，基本满足本文研究需要。

4.4 答案生成模块设计与实现

4.4.1 模块功能

答案生成模块的主要功能是“理解”用户输入的问句或关键词，借助命名实体识别模块及存在图数据库中的知识图谱，用模式匹配的方式形成完整的回答，并在终端显示给用户。

4.4.2 模块流程

答案生成模块分为问句分类，问句解析和合成答案这几个关键步骤，处理流程及各步处理结果如图 4.12 所示。

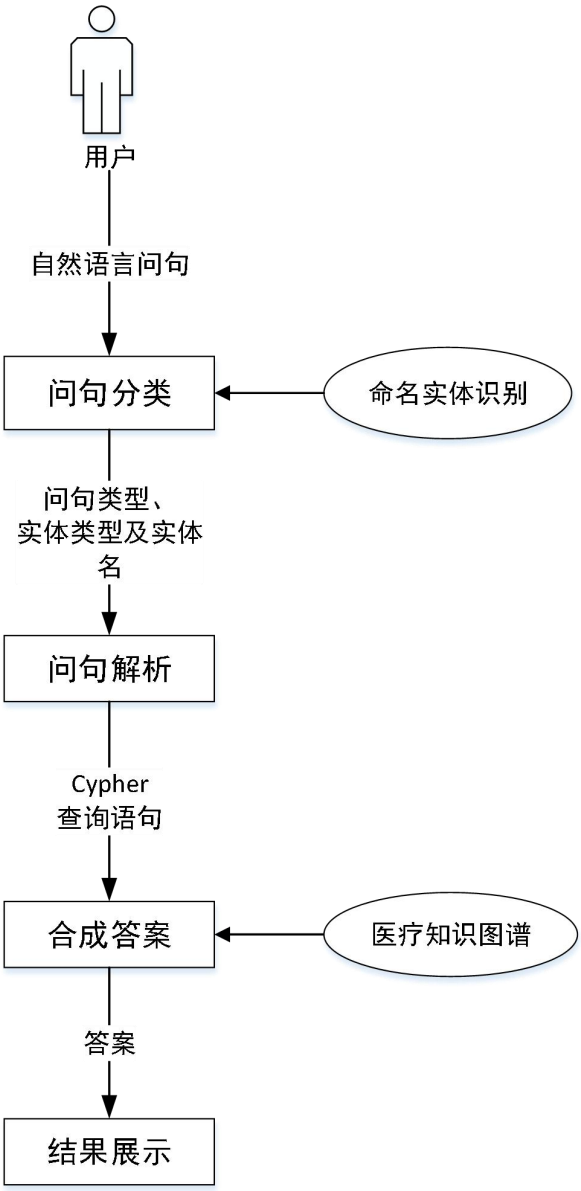


图 4.12 生成答案模块流程图

问句分类

问句分类有两个主要任务，一是通过实体识别标记出问句中含有的实体名称，从而判断他们所属的实体类别，二是通过问题关键词匹配识别出问句中含有的问题词的类别。通过实体识别获得的实体词类别和问题关键词匹配获得的问题词类别最终判断出该问句属于系统支持回答的 18 类问题类型中的哪一类，实现流程如图 4.13 所示。

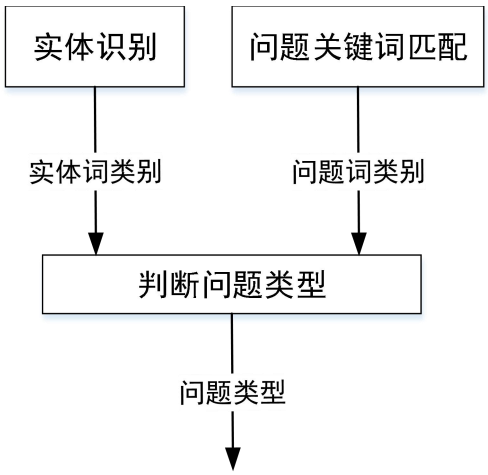


图 4.13 问句分类流程图

实体识别通过命名实体识别模块完成，答案生成模块中只需要将分好字的问句传入实体识别模块中，并接收字-标签格式的输出序列，存在数组中，再从数组中识别出问句中包含的实体名称及类型。具体实体识别过程参考本章 4.3 节。

问题关键词匹配需要借助一系列存储问题词的数组完成。结合系统能回答的问题类型、实体类型、实体关系、疾病属性，设置了一组问题词数组，代码如图 4.14 所示。

```
self.check_qwds = ['检查', '检查项目', '查出', '检查', '测出', '试出']
self.belong_qwds = ['属于什么科', '属于', '什么科', '科室']
self.drug_qwds = ['药', '药品', '用药', '胶囊', '口服液', '炎片']
self.food_qwds = ['饮食', '饮用', '吃', '食', '伙食', '膳食', '喝', '菜' ,'忌口', '补品', '保健品', '食谱', '
self.symptom_qwds = ['症状', '表征', '现象', '症候', '表现']

self.acompany_qwds = ['并发症', '并发', '一起发生', '一并发生', '一起出现', '一并出现', '一同发生', '一同出现

self.cause_qwds = ['原因','成因', '为什么', '怎么会', '怎样才', '咋样才', '怎样会', '如何会', '为啥', '为何',
self.prevent_qwds = ['预防', '防范', '抵制', '抵御', '防止','躲避','逃避','避开','免得','逃开','避开','避掉',
                     '怎样才能不', '怎么才能不', '咋样才能不','咋才能不', '如何才能不',
                     '怎样才不', '怎么才不', '咋样才不','咋才不', '如何才不',
                     '怎样才可以不', '怎么才可以不', '咋样才可以不', '咋才可以不', '如何可以不',
                     '怎样才可不', '怎么才可不', '咋样才可不', '咋才可不', '如何可不']
self.lasttime_qwds = ['周期', '多久', '多长时间', '多少时间', '几天', '几年', '多少天', '多少小时', '几个小时
self.cureway_qwds = ['怎么治疗', '如何医治', '怎么医治', '怎么治', '怎么医', '如何治', '医治方式', '疗法', '
self.cureprob_qwds = ['多大概率能治好', '多大几率能治好', '治好希望大么', '几率', '几成', '比例', '可能性', '
self.easyget_qwds = ['易感人群', '容易感染', '易发人群', '什么人', '哪些人', '感染', '染上', '得上']
```

图 4.14 问题词数组代码图

遍历这些数组的同时判断问句中是否含有某类问题词，就可得到问句中包含的问题词类型，代码如图 4.15 所示。

```
def check_words(self, wds, sent):
    for wd in wds:
        if wd in sent:
            return True
    return False
```

图 4.15 问题关键词匹配代码图

参考系统可回答的问题类型，事先枚举了每一种与某类实体有关的可能的问题类型。因此识别出问句中存在的实体词类别和问题词类别后，最终可以确定该问句的问题类型，此过程的部分代码如图 4.16 所示。

```
if self.check_words(self.symptom_qwds, question) \
        and ('disease' in types):
    question_type = 'disease_symptom'
    question_types.append(question_type)
```

图 4.16 问题分类代码图

问句分类后返回的“data”格式如表 4.8 所示。

表 4.8 data 数据结构表

args		question_types
实体名	实体类型	问题类型

如输入问句“感冒要吃什么药？可以喝小米粥吗？”，则问句分类步骤处理后返回：“{'args': {'感冒': ['disease'], '小米粥': ['food']}, question_types': ['disease_do_food', 'food_do_disease', 'disease_drug']}”，其中有“感冒”和“小米粥”两个实体，最终返回的问题类型有三类。

问句解析

问句解析的目的是根据问题分类处理后传来的包含实体名、实体类型、问题类型的“data”，生成数据库查询所需的 cypher 语句，最终返回一个包含问题类型及其 cypher 语句的数组。

问句解析步骤中，首先需要将“data”中的“args”的格式变换为“实体类型：实体名称”的格式，再将新格式的数据存入“entity_dict”中。例如，“args: { ‘头痛’：[‘disease’,’symptom’] }”转换为“entity_dict：{ ‘disease’:[‘头痛’], ‘symptom’: [‘头痛’] }”。

这便于形成 cypher 语句时找出问句中包含的实体名。此过程是通过“args”中存有的键值对关系实现的，代码如图 4.17 所示。

```
def build_entitydict(self, args):
    entity_dict = {}
    for arg, types in args.items():
        for type in types:
            if type not in entity_dict:
                entity_dict[type] = [arg]
            else:
                entity_dict[type].append(arg)
    return entity_dict
```

图 4.17 arg 格式转换代码图

然后依次处理问句中所含的每一种问题类型，生成 cypher 语句，部分代码如图 4.18 所示。

```
for question_type in question_types:
    sql_ = {}
    sql_['question_type'] = question_type
    sql = []
    if question_type == 'disease_symptom':
        sql = self.sql_transfer\
            (question_type, entity_dict.get('disease'))

    elif question_type == 'symptom_disease':
        sql = self.sql_transfer\
            (question_type, entity_dict.get('symptom'))
```

图 4.18 处理单个问题类型代码图

处理一种问题类型时，根据问题类型以及“entity_dict”中存放的实体名称，生成回答此问题类型需要执行的具体的 cypher 查询语句，实现此过程的部分代码如图 4.19 所示。

```
if question_type == 'disease_cause':
    sql = ["MATCH (m:Disease) where m.name = '{0}' " \
           "return m.name, m.cause"
               .format(i) for i in entities]
```

图 4.19 生成 cypher 语句代码图

经过问句解析步骤处理后，生成了一个包含问题类型及其对应的 cypher 语句的数组。

合成答案

合成答案步骤根据从问题解析步骤传来的数组，依次处理数组中的每个问题，得到每一个问题的答案，最后连接这些答案得到最终答案。

处理每一个问题时，先执行 cypher 语句得到模板中的关键词，再结合关键词和问题类型生成此问题的答案，最后把生成的答案连接到最终答案的末尾，实现此过程的部分代码如图 4.20 所示。

```
def search_main(self, sqls):
    final_answers = []
    for sql_ in sqls:
        question_type = sql_['question_type']
        queries = sql_['sql']
        answers = []

        for query in queries:
            ress = self.g.run(query).data()
            answers += ress

        final_answer = self.answer_prettify\
            (question_type, answers)
        if final_answer:
            final_answers.append(final_answer)
    return final_answers
```

图 4.20 生成回答代码图

因为事先为每种问题类型定义了一个答案模板，因此结合问题类型及查询结果即可生成回答此问题的答案，部分代码如图 4.21 所示。

```
if question_type == 'disease_symptom':
    desc = [i['n.name'] for i in answers]
    subject = answers[0]['m.name']
    final_answer = '{0}的症状包括：{1}'.format
    (subject, '；'.join(list(set(desc))[:self.num_limit]))

elif question_type == 'symptom_disease':
    desc = [i['m.name'] for i in answers]
    subject = answers[0]['n.name']
    final_answer = '症状{0}可能染上的疾病有：{1}'.format
    (subject, '；'.join(list(set(desc))[:self.num_limit]))
```

图 4.21 生成答案代码图

经过合成答案步骤的处理，可以得到一段最终答案返回给用户。

4.4.3 问答系统结果展示

图 4.22 展示了系统对不同类问句的处理结果。

用户:感冒有哪些症状?

小勇: 感冒的症状包括: 咽痛; 发热伴寒战; 流鼻涕; 咽喉干燥及灼热感; 浑身忽冷忽热; 发烧; 鼻塞; 情绪性感冒; 头痛

用户:感冒需要忌口什么?

小勇: 感冒忌食的食物包括有: 油条; 咸鱼; 猪油(板油); 白扁豆

用户:感冒要多吃什么?

小勇: 感冒宜食的食物包括有: 鸡蛋;芝麻;鹌鹑蛋;南瓜子仁

推荐食谱包括有: 姜丝萝卜汤;赤小豆粥;凉拌香椿;薏米莲子粥;醋熘土豆丝;葱蒜粥;绿豆薏米饭;香椿芽粥

用户:感冒需要做哪些检查?

小勇: 感冒通常可以通过以下方式检查出来: 内科检查; 尿常规; 白细胞计数(WBC); 血常规; 肺和胸膜听诊

用户:感冒的治疗周期?

小勇: 感冒治疗可能持续的周期为: 7-14天

用户:四肢无力是为什么?

小勇: 症状四肢无力可能染上的疾病有: 肾上腺肿瘤; 急性肠炎; 五积; 月劳病; 晕动病; 卡斯钦-贝克病; 痉癖; 老年人重症肌无力; 氰化物中毒; 原发性甲状旁腺功能亢进症与肾病; 糖原贮积病Ⅱ型; 小儿脊髓性肌萎缩; 银杏中毒; 外伤性脾破裂; 暴泻; 脱髓鞘;

图 4.22 系统运行结果图

第五章 总结与展望

5.1 总结

本论文研究了一个基于医疗知识图谱的自动问答系统的研发过程及相关技术。使用知识图谱组织数据，并采用图数据库 Neo4j 存储与管理数据。自动问答系统在问题分类步骤中采用了 Bi-LSTM 神经网络模型来完成识别问句中的实体的任务。生成回答部分通过问题分类、问题解析等步骤，采用模式匹配的方法，结合事先定义好的回答不同问题类型的模板以及问句中的实体关键词得到回答。最终完成了一个可以自动回答用户医疗方面问题的原型对话系统。使用知识图谱组织数据符合联系无处不在的客观规律，也可以使数据的结构更清晰、直观，从而方便数据管理和知识推理。使用神经网络完成实体识别的任务比遍历词典匹配的方法更省空间和时间，有较高的正确率，且可以识别出词典中没有的实体名。

本论文中实现的一个原型对话系统主要达成了以下目标：

• 通过处理医药网站公开的数据，构建起一个包含 7 类共约 4.4 万实体、9 类疾病属性以及 11 类共约 30 万实体关系的知识图谱。

• 训练了一个 Bi-LSTM 神经网络模型以完成命名实体识别任务，测试结果显示实体识别结果良好，平均正确率、召回率都达到了 99%以上。除 drug 类实体外，正确识别其它类实体的概率达到 97%以上。

• 系统支持回答十八类医疗领域相关的问题。

5.2 展望

本文设计并初步实现了一个医疗自动问答系统的原型，但此系统还有许多地方可以完善，我认为以下几点是最迫切需要改进的方面：

• 知识图谱的构建还不够完整。可以加入更多关系类型，这有助于完成一些简单的知识推理，使系统更加智能。可以为除疾病外的实体类型添加更多属性；

• 命名实体识别模块可以使用 Bi-LSTM+CRF 模型或者更好的模型代替单一的 Bi-LSTM 模型，以丰富系统可识别的问句类型；

• 可以为系统创建一个管理员角色，以管理数据库中的数据甚至管理用户。比如可以对用户经常提的问题进行分析与存档，形成用户自己的“电子病历”。

参考文献：

[1] 贾熹滨,李让,胡长建,陈军成.智能对话系统研究综述[J].北京工业大学学报,2017,43(9):1347-1349

[2] 马晨浩.基于甲状腺知识图谱的自动问答系统的设计与实现[J].智能计算机与应用,2018,8(3):102-103

[3]FERRUCCI D A，BROWN E W，CHU-CARROLL J，et al. Building watson: an overview of the DeepQA Project [J]. Ai Magazine，2010，31(3) : 59-79.

[4]FERRUCCI D，LEVAS A，BAGCHI S，et al. Watson: beyond jeopardy [J]. Artificial Intelligence，2013，199 /200(3) : 93-105.

[5] Sutskever, Ilya, Oriol Vinyals. Sequence to Sequence Learning with Neural Networks[M]. Advances in neural information processing systems, 2014.

[6] VINYALS O, LE Q. A neural conversational model[J/OL]. arXiv: 1506. 05869, 2015[2016-08-27]. https://arxiv. org/abs/1506.05869

[7] Gu J , Lu Z , Li H , et al. Incorporating Copying Mechanism in Sequence-to-Sequence Learning[J]. 2016.

[8] Zhou H , Huang M , Zhang T , et al. Emotional Chatting Machine: Emotional Conversation Generation with Internal and External Memory[J]. 2017.

[9] 柏兵,侯霞,石松.基于CRF和BI-LSTM的命名实体识别方法[J].北京信息科技大学学报,2018,33(6):29-32

[10] 曹明宇,李青青,杨志豪.基于知识图谱的原发性肝癌知识问答系统[J].中文信息学报,2019,33(6):90

[11] 李梦洁,董峦.基于 pytorch 的机器翻译算法的实现[J].计算机技术与发展,2018,28(10):161

[12] 张芳芳,马敬东,王小贤.国外医学领域自动问答系统研究现状及启示[J].医学信息学杂志,2017,38(3)

[13] 张聪品,方滔,刘昱良.基于 LSTM-CRF 命名实体识别技术的研究与应用[J].计算机技术与发展,2019,29(2):107-108

[14] 周博通,孙承杰,林磊,刘秉权.InsunKBQA：一个基于知识库的问答系统[J].智能计算机与应用,2017,7(5)

致　　谢

在学习知识、实现系统和撰写论文的过程中，得到了很多人的帮助，我想在此对他们表达感谢。

首先，感谢本次生产实习与毕业论文的指导老师何婷婷教授。何老师总是耐心而明确地指出我的问题所在，和她交谈让我有种如沐春风的感觉。何老师身为院长，尽职尽责，对学生上心、认真，对生产实习和毕业论文的质量严格把控，全程都坚持高标准、严要求。从何老师身上学到的不论是学术知识，还是做人做事的方式，都让我受益终身。

其次，感谢何老师团队里的学长学姐们。感谢及时帮助我、替我答疑解惑的范瑞学长，谢谢学长不厌其烦地帮我处理许多琐碎的细节问题，协助我理顺论文思路、攻克论文难点、调试代码、修改论文、与老师沟通等等，这些帮助我都记在心里。感谢帮我明确研究方向和具体内容的薛昊学长。感谢时常督促我的汤丽学姐。因为有他们，我才能顺利地完成本次毕业设计。

最后，感谢家人的支持和鼓励，感谢同窗四年的陪伴，感谢华中师范大学计算机学院对我的悉心培养，衷心祝愿母校越来越好，计算机学院越来越强！