华北电力大学

毕 业 设 计（论 文）

论文题目：基于知识图谱的对话系统研究与实现

院　　系：控制与计算机工程学院
计算机系

专　　业：计算机科学与技术

班　　级：1902 班

姓　　名：叶佳庆

学　　号：220191090723

指导教师：闫蕾

二〇二三 年 六 月

摘　　要

本文的主要研究方向是基于医疗领域知识图谱的问答系统，旨在通过构建基于医疗领域知识图谱的对话系统，实现快速准确地获取医疗知识，提高医疗领域的医生和患者的工作效率。

本文对知识图谱和对话系统的相关理论和技术进行了分析。知识图谱是由一系列实体、属性和关系构成的图形化知识表示，具有高效、准确、可扩展和可重用等特点。对话系统是一种能够通过自然语言输入，快速准确地回答问题的人机交互系统。基于此，本文设计了基于医疗领域知识图谱的问答系统的系统结构和功能模块。系统结构主要包括数据抽取和存储、知识图谱查询和分析、自然语言处理等模块。数据抽取和存储模块负责医疗领域实体、属性和关系的抽取和存储。知识图谱查询和分析模块可以通过 Cypher 查询语言和图形数据库工具，对知识图谱中的实体和关系进行查询和分析，并通过可视化展示，呈现知识图谱的结构和属性。用户输入的自然语言问题通过自然语言处理模块被转化为查询语句，从而实现自然语言问答的功能。

总之，本文通过构建和维护医疗领域的知识图谱，能够快速准确地获取医疗知识，并且具有较高的准确性和实用性，在医疗领域具有广泛的应用前景。

关键词：知识图谱，对话系统，医疗领域

ABSTRACT

This paper mainly studies the question-answering system based on the medical field knowledge graph, aiming to quickly and accurately obtain medical knowledge by constructing and maintaining the medical field knowledge graph, thereby improving the work efficiency of doctors and patients in the medical field.

This paper analyzes the related theories and technologies of knowledge graphs and question-answering systems. A knowledge graph is a graphical knowledge representation composed of a series of entities, attributes, and relationships, featuring high efficiency, accuracy, scalability, and reusability. A question-answering system is a human-computer interaction system that can quickly and accurately answer questions through natural language input. Based on this, the system structure and functional modules of the question-answering system based on the medical field knowledge graph are designed in this paper. The system structure mainly includes data extraction and storage, knowledge graph query and analysis, and natural language processing modules. The data extraction and storage module is responsible for the extraction and storage of entities, attributes, and relationships in the medical field. The knowledge graph query and analysis module can query and analyze entities and relationships in the knowledge graph through the Cypher query language and graph database tools, and display the structure and attributes of the knowledge graph through visualization. The natural language processing module can convert users' natural language questions into query statements, realizing the natural language question-answering function.

In conclusion, this paper constructs and maintains the medical field knowledge graph to quickly and accurately obtain medical knowledge. The question-answering system based on the medical foe;d knowledge graph has high accuracy and practicality, and has a wide range of application prospects in the medical field.

KEY WORDS: knowledge graph,dialogue system,medical field

目　　录

第 1 章 绪论

1.1 选题背景及意义

信息时代的快速发展带来了海量的数字化数据和知识，如何高效地获取和管理这些知识成为了一个重要的挑战。传统的搜索引擎在面对复杂的查询和语义理解时存在一定的局限性。因此，人们对于更智能、准确的信息交互方式的需求日益增加。

问答系统作为一种更直接、智能化的信息交互方式可以解决这些问题。问答系统能够通过自然语言提问，并给出准确、简洁的回答。然而，传统的基于关键词匹配的问答系统往往只能提供表面级的答案，无法理解用户查询的真正意图，从而限制了其准确性和智能性。

与此同时，知识图谱的兴起为问答系统的发展提供了新的机遇。知识图谱是一种结构化的知识表示形式，通过将大量实体、属性和关系实现了对知识的有效组织和管理。

知识图谱不仅包含了丰富的领域知识，还能够捕捉实体之间的关系和语义信息，为问答系统提供了更丰富的语境和背景知识。基于知识图谱的问答系统在理解用户的问题方面具有深层次的优势，这得益于它能够结合知识图谱中的语义信息来更好地理解用户的查询意图，提供准确、精准的答案，大大提升了问答系统的准确性和智能化水平。

基于知识图谱的语义搜索在理解用户查询意图方面具有更大的优势。知识图谱中包含丰富的关系和语义信息，可以更好地对用户的问题进行理解，从而提供更加准确的答案。知识图谱中的丰富实体和关系信息可以为问答系统提供更多的上下文，使得系统能够根据用户查询的语义进行推理和分析，提供更具智能化的回答，帮助用户更好地理解和利用知识。此外，知识图谱可以整合不同领域的知识，并建立语义关联，使得不同领域的知识可以相互补充和共享。基于知识图谱的问答系统可以帮助用户跨领域地获取所需知识，推动知识的整合与共享。问答系统作为人机交互的一种形式，具有更直接、自然的交互方式。通过研究基于知识图谱的问答系统，可以探索更高效、智能的人机交互模式，提升用户体验和应用场景的丰富性。总的来说，通过构建更智能、准确的问答系统，可以提升信息检索的效果，推动知识的共享与利用，为人们提供更便捷、准确的知识获取方式。

1.2 国内外研究现状

1.2.1 知识图谱研究现状

在国际上，知识图谱的研究受到了广泛的关注，并取得了许多重要进展。一方面，像谷歌、Facebook 和微软等大型互联网公司积极投入了知识图谱的构建和应用。例如，Google 的 Knowledge Graph 和 Facebook 的 Open Graph 都是大规模知识图谱的代表性应用，为用户提供了丰富的信息检索和知识推荐服务。

另一方面，学术界也开展了大量的研究工作。国际上的研究者提出了许多高效的知识抽取和图谱构建方法。此外，还有基于半监督学习、迁移学习和深度学习等技术的研究。

在知识图谱表示和推理方面，国际上的研究者提出了许多创新的模型和方法。另外，图神经网络等基于图结构的表示方法，通过在图上进行信息传播和特征学习，能够更好地表达知识图谱中的语义信息。

国内的学术界也在知识图谱的研究方面做出了积极贡献。国内研究者在知识图谱构建和表示方法方面开展了广泛的研究。他们提出了一些高效的知识抽取和图谱构建算法，以应对中文语境下的特殊挑战。例如，基于语义依存分析、词义消歧和实体链接等技术手段，可以从大规模的中文文本中提取实体和关系信息，并构建中文知识图谱。此外，国内的研究者也对知识图谱的表示方法进行了深入研究，提出了一些适用于中文知识图谱的表示学习模型和推理算法。

1.2.2 问答系统研究现状

问答系统的研究涵盖了多个方向，包括基于规则的问答系统、基于统计的问答系统和基于深度学习的问答系统等。其中，基于规则的问答系统主要利用规则进行问答，基于统计的问答系统则采用统计模型进行问答，而基于深度学习的问答系统则基于深度学习算法进行问答。

基于规则的问答系统主要通过预定义的规则和模板来实现问题解析和答案生成。然而，由于规则的复杂性和限制性，这种方法在处理复杂的自然语言查询时存在局限性。

基于统计的问答系统则通过分析大量的语料库和语言统计信息来推断问题和答案之间的关联。这种方法通常采用词袋模型、语言模型和统计匹配等技术，能够处理一定程度上的语义理解和答案推断。

近年来，基于深度学习的问答系统得到了广泛的关注。这些系统利用神经网络模型，通过端到端的训练从原始的自然语言查询中提取特征，并进行问题理解和答案生成。例如，基于循环神经网络（RNN）和长短期记忆网络（LSTM）的模型，能够捕捉句子的语义信息和上下文关系，从而实现更准确的问题解析和答案生成。

在国内，问答系统的研究和应用也取得了一些重要的进展。国内的互联网公司和研究机构积极投入了问答系统的研究和开发。国内的学术界也对问答系统进行了广泛的研究。为了提高问答系统的准确性和智能化水平，研究者们提出了一些创新的模型和方法。其中，基于深度学习的方法得到了广泛应用。研究者们设计了各种神经网络模型，如卷积神经网络（CNN）、循环神经网络（RNN）和注意力机制等，来实现问题解析、语义理解和答案生成。他们通过大规模的语料库和预训练的语言模型，提高了系统对于复杂问题和语义推理的能力。

此外，国内的研究者还关注于特定领域的问答系统研究，如法律问答系统等。他们结合特定领域的知识和规则，设计了定制化的问答系统，提供了领域内准确、专业的问题回

答。

另外，近年来，国内的研究者们开始将知识图谱与问答系统相结合，开展基于知识图谱的问答系统研究，结合自然语言处理和推理技术，实现更准确、智能的问题解析和答案生成。

1.3 本文的主要研究工作

本文的主要研究工作在在于对知识图谱的研究和应用。探究其发展脉络、基本原理和特点，并将构建好的知识图谱应用于医疗领域的问答系统上。本文共分为四章：

第一章介绍了研究背景及意义，简述了知识图谱和问答系统在国内外的研究现状，确定了本文的主要研究内容。

第二章则是相关理论与方法部分，分别介绍了知识图谱和问答系统的定义和特点，探究其发展历程，并简述了知识图谱的构建方法。

第三章详细介绍了医疗领域知识图谱的构建过程，包括知识图谱结构设计、数据收集、数据存储等过程。主要涉及到疾病、药品、医院等相关实体及其关系的构建和管理。

第四章将构建好的医疗领域知识图谱应用于问答系统，并介绍了问答系统的设计架构和实现过程。其中，涉及到各个模块的详细研究和设计，以实现一个高效、准确的医疗问答系统。

最后对本文的研究结果进行总结和分析，并提出了未来的改进方向。

第 2 章 相关技术与理论基础

2.1 知识图谱

2.1.1 知识图谱的定义和特点

知识图谱是一种以结构化形式表示和组织知识，旨在捕捉和表达现实世界中的知识和语义关系的图形化网络结构。它通过将实体、属性和关系以图的形式进行组织和链接，构建了一个语义丰富的知识网络。知识图谱不仅包含了丰富的事实和概念，还能够描述它们之间的关系和语义信息。相比于传统的数据库和搜索引擎，知识图谱具有以下几个特点[1]：

（1）结构化表示：知识图谱采用结构化的方式将知识进行组织和存储。通过使用图的形式，它能够准确地表示实体、属性和关系之间的语义关联，形成一个有机的知识网络。

（2）语义链接：知识图谱通过语义链接将不同实体之间的关系进行表达。这些链接可以是显式的关联关系，也可以是隐含的语义关联。语义链接能够帮助系统理解实体之间的语义关系，从而提供更准确的查询和推理能力。

（3）可扩展性和动态性：知识图谱具有良好的可扩展性和动态性。它可以随着新的知识不断扩展和更新，从而保持其知识库的时效性和准确性。知识图谱的动态性使得它能够适应不断变化的知识环境和用户需求。

2.1.2 知识图谱的发展历程

知识图谱的发展历程可以分为以下几个阶段：

（1）早期阶段（20 世纪 60 年代-90 年代）知识图谱的概念最早由美国计算机科学家 John F. Sowa 提出。他认为，知识图谱是一种表示知识的语言，可以将知识组织成一张有向图，使得不同知识点之间的关系能够清晰地呈现出来[2]。在这个阶段，知识图谱的研究主要集中在知识表示和推理方面。

（2）Web 2.0 时代（2000 年代初期-中期）随着 Web 2.0 时代的到来，人们开始将知识图谱应用于互联网领域，用于构建更加智能化的应用程序[3]。比较典型的例子包括谷歌知识图谱和百度知识库，它们都是基于大规模语义数据的知识图谱系统，可以为用户提供更加准确、可靠的搜索结果。

（3）人工智能时代（2010 年代-至今）随着人工智能技术的不断发展，知识图谱逐渐成为人工智能领域中的重要技术之一。目前，知识图谱已经成为智能化应用的重要支撑技术，例如智能客服、智能问答、智能推荐等。同时，知识图谱的应用场景也不再局限于互联网领域，而是向其他领域延伸，例如医疗、金融、教育等[4]。

2.1.3 知识图谱的组成

知识图谱由实体、属性和关系组成，这些组成部分相互作用形成一个完整的知识网络[5]。

（1）实体（Entities）：实体是知识图谱中的核心元素，代表着现实世界中的具体事物或抽象概念。实体可以是人、地点、组织、产品、事件等。每个实体都有唯一的标识符（ID）和一组属性来描述它的特征。

（2）属性（Properties）：属性用于描述实体的特征和属性。属性可以是实体的名称、描述、时间、地点、数量等。每个属性都与实体之间有明确的关联，用于丰富实体的语义信息。

（3）关系（Relationships）：关系表示实体之间的语义链接和关联。关系描述了实体之间的相互作用、依赖或归属关系。例如，"作者"和"书籍"之间可以有关系"写作"，"演员"和"电影"之间可以有关系"出演"等。关系通过连接不同的实体，形成知识图谱的结构和语义关联。

2.1.4 知识图谱的关键技术

知识图谱的构建和应用涉及多个关键技术

（1）数据采集与处理：知识图谱的构建通常涉及大量的数据采集和处理工作。数据可以来自多种不同的数据源。数据采集包括数据爬取、清洗和预处理等步骤，以获取高质量的数据[6]。

（2）实体识别与链接：实体识别是指从文本中识别出具体的实体，并将其与知识图谱中的相应实体进行链接。实体识别常常利用自然语言处理和机器学习技术，结合命名实体识别（Named Entity Recognition）和实体链接（Entity Linking）方法，将文本中的实体映射到知识图谱中。

（3）属性抽取与关系建模：属性抽取是指从文本或其他数据源中提取出实体的相关属性信息。关系建模是指识别实体之间的关系，并在知识图谱中建立相应的关联。这些技术包括属性抽取、关系抽取、关系推理和关系分类等方法。

2.2 知识图谱的构建

2.2.1 数据采集与处理

知识图谱的构建离不开对数据的采集和处理。数据采集是指从各种数据源中获取知识的过程，而数据处理则包括数据清洗、去重、结构化等操作，以提高数据的质量和一致性。以下是一些常用的数据采集与处理方法：

（1）网络爬虫：通过网络爬虫技术可以自动从互联网上收集数据。爬虫程序可以按照预定的规则和策略抓取网页内容，并提取出需要的信息进行后续处理。

（2）数据清洗：从原始数据中清除噪声、错误和冗余信息，以提高数据的质量。常见的清洗操作包括去除 HTML 标签、处理缺失值、纠正错误等。

（3）数据集成：将来自不同数据源的数据进行整合，消除重复和冲突。数据集成可以通过数据匹配、属性映射和数据融合等技术来实现。

（4）数据转换与结构化：将原始数据转换为结构化的形式，以符合知识图谱的数据模型。数据转换可以包括实体识别、属性抽取、关系抽取等步骤，将文本或非结构化数据转化为结构化的实体、属性和关系。

2.2.2 实体识别与链接

实体识别与链接旨在将文本中的实体识别并链接到知识图谱中的相应实体。以下是常见的方法：

（1）命名实体识别（Named Entity Recognition，NER）：NER 旨在从文本中识别出具体的实体，如人物、地点、组织等。NER 方法可以基于规则、统计模型或深度学习模型进行实现，从而提取出具有语义意义的实体。

（2）实体链接（Entity Linking）：实体链接方法通常利用实体的上下文信息、实体的描述和实体的属性等特征，将文本中的实体与知识图谱中的实体进行匹配和链接。

（3）实体消岐（Entity Disambiguation）：在实体链接过程中，可能会出现同名实体或歧义实体的情况。实体消岐旨在根据上下文和语义信息，确定文本中实体的准确含义和链接目标。

2.2.3 属性抽取与关系建模

属性抽取与关系建模是知识图谱构建过程中的关键环节，它们用于从文本或其他数据源中提取实体的属性信息，并建立实体之间的关系。以下是一些常见的方法[7]：

（1）属性抽取：属性抽取旨在从文本中提取实体的特征和属性信息。它可以通过基于规则的方法、统计模型或深度学习模型来实现。常见的属性包括实体的名称、描述、时间、地点、数量等。

（2）关系抽取：关系抽取可以利用模式匹配、规则推理或机器学习等方法，从句子或段落中提取出实体之间的关联关系，如"作者"、"出版"、"位于"等。

（3）关系推理：关系推理通过利用已知的关系和推理规则，推断出新的关系和语义连接。它可以基于逻辑推理、统计推理或深度学习模型来实现，从而丰富知识图谱中的关系网络。

2.2.4 知识图谱的扩展与更新

知识图谱需要不断地扩展和更新以适应新的知识和需求。以下是一些常见的知识图谱扩

展与更新的方法[8]：

（1）数据补充和扩展：通过不断地从新的数据源中获取知识，将其补充和扩展到已有的知识图谱中。这可以包括数据融合、实体链接、属性抽取等步骤，以保持知识图谱的完整性和一致性。

（2）人工标注与验证：通过人工的标注和验证过程，对知识图谱中的实体、属性和关系进行校对和修正。人工标注可以帮助识别错误、消除歧义，并提高知识图谱的质量和准确性。

（3）自动化更新：借助自动化的方法和技术，定期对知识图谱进行更新和维护。这可以包括自动化的实体识别、属性抽取、关系推理等过程，以减少人工操作和提高更新效率。

综上所述，知识图谱的构建方法涵盖了数据采集与处理、实体识别与链接、属性抽取与关系建模等关键步骤。通过采用合适的方法和技术，可以构建出丰富、准确的知识图谱。

2.3 问答系统

2.3.1 问答系统的定义和分类

问答系统是一种通过与用户进行自然语言交互，从大规模的知识库中获取准确答案的系统。根据不同的特点和功能，分为以下三种：

（1）基于规则的问答系统：这类系统利用预定义的规则和模板来实现问题解析和答案生成。规则可以包括句法规则、语义规则和推理规则等，通过匹配和应用这些规则，系统能够生成准确的答案。然而，由于规则的复杂性和限制性，基于规则的问答系统在处理复杂的自然语言查询时存在局限性。

（2）基于统计的问答系统：这类系统通过分析大量的语料库和语言统计信息来推断问题和答案之间的关联。统计方法可以包括词袋模型、语言模型和统计匹配等技术，通过对问题和候选答案进行相似度计算和匹配，系统能够得出最可能的答案[9]。基于统计的问答系统能够处理一定程度上的语义理解和答案推断，但对于复杂的语义和推理需求仍存在限制。

（3）基于深度学习的问答系统：这类系统利用深度学习模型，通过端到端的训练从原始的自然语言查询中提取特征，并进行问题理解和答案生成。例如，基于循环神经网络（RNN）和注意力机制的模型，能够捕捉句子的语义信息和上下文关系，从而实现更准确的问题解析和答案生成[10]。基于深度学习的问答系统能够处理复杂的语义和推理需求，具有更高的准确性和智能化水平。

2.3.2 问答系统的发展历程

问答系统是人工智能领域的一项重要技术，其发展历史可以追溯到上世纪五十年代。以下是问答系统的发展历史。

第一代问答系统出现于上世纪五十年代末期和六十年代初期，主要应用于数学和计算机科学领域。这些系统主要是基于规则的，通过在系统中编写一系列规则来回答问题。这些系统的问题和回答都是预先定义好的，不能进行自由的问答。第二代问答系统出现于上世纪七十年代，其主要特点是引入了自然语言处理技术。这些系统能够处理自然语言输入，并生成自然语言输出。但是，这些系统的问题和回答仍然是基于规则的，需要手动编写规则。第三代问答系统出现于上世纪八十年代和九十年代，其主要特点是引入了机器学习技术。这些系统能够自动学习规则，并对新的问题进行回答。

近年来，随着人工智能和自然语言处理技术的快速发展，问答系统得到了广泛应用。目前的问答系统主要有两种类型，一种是基于规则的问答系统，另一种是基于机器学习的问答系统。基于规则的问答系统需要手动编写规则，其优点是准确度高，但缺点是需要大量人力和时间来编写规则；基于机器学习的问答系统能够自动学习规则，其优点是效率高，但缺点是准确度可能会受到训练数据的限制[11]。

总之，问答系统的发展历史经历了从规则驱动到自然语言处理，再到机器学习的演进过程。随着人工智能和自然语言处理技术的不断进步，问答系统的应用前景将会越来越广阔。

2.3.3 问答系统的关键要素

问答系统的成功实现依赖于以下关键要素：

(1) 问题理解：问答系统需要能够理解用户提出的问题，并准确抽取问题中的关键信息。问题理解涉及到自然语言处理、语义分析和语境理解等技术，旨在将用户问题转化为可操作的形式。

(2) 知识表示与获取：问答系统需要有一个庞大的知识库作为参考，其中包含了丰富的实体、属性和关系信息。知识表示涉及到知识图谱、本体和语义网络等方法，用于将知识组织起来并提供高效的检索和推理能力。

(3) 答案生成与评估：问答系统需要根据问题和知识库中的信息，生成准确、全面的答案。答案生成涉及到文本生成、推理和逻辑推理等技术，通过综合考虑问题、知识和语境等因素，生成最合适的答案。答案评估则用于确定生成答案的质量和相关性。

(4) 上下文理解与对话管理：对于复杂的问答任务，上下文理解和对话管理至关重要。系统需要能够理解问题和答案之间的语境关系，并能够根据用户的追问或指引进行合理的对话。上下文理解和对话管理涉及到上下文建模、对话状态追踪和对话策略等技术[12]。

第 3 章 医疗领域知识图谱的构建

医疗领域知识图谱的构建包括知识图谱结构设计、数据采集与预处理、知识存储和知识应用等几个方面。本章从医疗领域知识图谱的应用价值和难点分析入手，探讨了医疗领域知识图谱的构建过程。

(1) 医疗领域知识图谱构建的意义和难点分析：简述知识图谱在医疗领域的应用价值，分析医疗领域知识图谱构建的难点。

(2) 知识图谱结构设计：设计医疗领域知识图谱的结构。

(3) 数据采集：通过对医疗信息网站进行文本分析，利用网络爬虫分析网站，爬取采集医疗领域的数据，

(4) 数据处理：对收集到的数据进行等处理，整理成结构化的数据

(5) 知识存储与可视化：将结构化的数据存储 Neo4j 图数据库中，以实现知识存储和可视化。

3.1 医疗领域知识图谱构建意义及难点

医疗领域知识图谱的构建有以下几个意义：

(1) 促进精准医疗。构建医疗领域知识图谱可以将医学知识、病历数据、医疗资源等进行整合，实现对患者的精准诊治，提高诊疗效率和准确性。

(2) 优化医疗资源配置。医疗领域知识图谱可以对医疗资源进行精准匹配和调配，减少医疗资源的浪费和重复利用，提高医疗资源的利用效率。

(3) 支持医学研究。构建医疗领域知识图谱可以将医学研究中的数据、文献、知识点等进行整合，帮助医学研究人员更好地发现疾病机理、提高临床治疗水平。

总之，医疗领域知识图谱的构建有助于提高医疗质量、优化医疗资源配置、促进医学研究和医疗智能化发展等方面的进步。

医疗领域知识图谱构建的难点主要有以下几个方面：

(1) 知识表示和推理。医学领域的知识点和关系非常繁多复杂，如何将这些知识点和关系进行有效表示和推理是一个难点。在医疗领域知识图谱的构建中，需要对医学知识进行形式化表示，如本体、语义网等，以便进行机器推理和应用。此外，还需要进行关系抽取、实体识别和自然语言处理等技术支持。

(2) 医疗概念的多义性和歧义性。医疗领域的概念存在多义性和歧义性，同一个症状可能由多种疾病引起，同一种药物可能有多种不同的剂量，如何进行有效的概念匹配和概念融合是一个挑战。在医疗领域知识图谱的构建中，需要考虑概念的标准化和规范化，建立术语标准和命名规则，以减少歧义性和多义性[12]。

(3) 专业性和精确度要求高。医学知识专业性高，这需要对医学知识进行深入的研究和理解，针对具体的医学应用场景进行精准的知识表示和推理，以达到高精准度和高可靠

性的要求。

总之，医疗领域知识图谱构建需要克服多个方面的难点，包括知识表示和推理、医疗概念的多义性和歧义性、专业性和精确度要求高等。只有克服这些难点，才能构建出高质量的医疗领域知识图谱，为医疗领域的发展提供更好的支持。

3.2 医疗领域知识图谱结构设计

构建知识图谱的关键是设计知识图谱结构，明确定义知识图谱中的实体、实体属性和实体关系。在设计知识图谱结构时，需要对具体的业务范围有深入的了解，同时要具备向后兼容的能力，以便后续的知识扩展。本文结合对现有多个医疗领域知识图谱的调查研究以及问答系统的需求，设计了以下结构。

(1) 实体：本文设计的医疗领域知识图谱以共包含 6 种实体类别。具体如表 3-1 所示：

表 3-1 知识图谱实体示例

序号	实体类型	示例
1	疾病	肺曲菌病、百日咳
2	症状	腹泻、心源性呼吸窘迫
3	药品	乳酸左氧氟沙星片、氯霉素片
4	食物	韭菜、圆白菜
5	医疗科目	呼吸内科、急诊科
6	诊断检查项目	支气管造影、血常规

(2) 实体关系：本文设计了 7 类实体关系，具体实体关系含义及示例如表 3-2 所示。

表 3-2 知识图谱实体关系示例

序号	实体关系	示例
1	常用药品	<肺气肿，常用药物，氨茶碱片>
2	宜吃食物	<百日咳，宜吃食物，小白菜>
3	所需检查	<肺大疱，所需检查，胸部平片>
4	忌吃食物	<肺气肿，忌吃，带鱼>
5	推荐药品	<肺气肿，推荐药品，白及颗粒>
6	推荐食谱	<肺气肿，推荐食谱，百合汁>
7	疾病症状	<肺曲菌病，症状，胸痛>

(3) 实体属性：本文针设计了 8 类实体属性，具体实体属性含义及示例如表 3-3 所示。

表 3-3 疾病实体属性示例

序号	实体属性	示例
1	疾病名称	感冒

2	疾病简介	感冒是由多种原因导致...
3	疾病病因	病原菌是...
4	预防措施	控制传染源...
5	治疗周期	1-2 个周
6	治疗方式	药物治疗,支持性治疗
7	治愈概率	98%
8	疾病易感人群	多见于老人和小儿

3.3 医疗领域知识图谱数据收集

知识图谱的数据来源多样，包括医院电子病历、医学文献、医药公司数据等等。在进行数据收集时，需要考虑如何对这些数据进行有效整合和融合，如何进行数据清洗和数据质量控制，以确保知识图谱的准确性和完整性。

数据可根据其组织形式分为三类：

（1）结构化的数据、非结构化数据和半结构化数据。结构化数据采用标准化格式，易于处理，如 Excel 表格；非结构化数据缺乏规律且不完整，无预设组织形式，如医生的手记；半结构化数据具备一定结构，组织形式灵活，常见于各种医疗网站。非结构化和半结构化数据往往需要转换成结构化数据才能被有效利用。对于非结构化数据需要大量的自然语言处理技术，实现难度较大。

网络抓取工具，遵循特定规则，自动获取互联网信息的程序或脚本。它能迅速地收集可访问页面的内容数据并将其转化为结构化数据。利用网络抓取技术，可以轻松高效地从网站中获取数据，大幅减少人力和时间成本。

本文使用网络爬虫技术爬取医疗网站上的半结构化数据，具体数据来源于寻医问药网。图 3-1 展示了部分爬虫代码。这段代码定义了一个名为 symptom_spider 的函数，用于从指定的 URL 爬取症状信息。调用 self.get_html(url)方法获取网页的 HTML 内容。html 变量存储 HTML 内容。使用 etree.HTML(html)创建一个 selector 对象，它允许使用 XPath 表达式从 HTML 内容中提取数据。使用 XPath 表达式//a[@class="gre"]/text()从 HTML 内容中提取症状名称。symptoms 变量存储提取到的症状名称列表。首先，使用 XPath 表达式//p 从 HTML 内容中选择所有<p>元素。然后，遍历这些元素，使用 p.xpath('string(.)')提取每个<p>元素中的文本内容，并删除不需要的字符（如换行符、制表符等）。将处理后的文本添加到 detail 列表中。创建一个名为 symptoms_data 的字典，包含两个键：symptoms 和 symptoms_detail。它们分别存储症状名称列表和症状详细信息列表。函数返回两个值：symptoms（症状名称列表）和 detail（症状详细信息列表）。

```
'''症状信息解析'''
def symptom_spider(self, url):
    html = self.get_html(url)
    selector = etree.HTML(html)
    symptoms = selector.xpath('//a[@class="gre" ]/text()')
    ps = selector.xpath('//p')
    detail = []
    for p in ps:
        info = p.xpath('string(.)').replace('\r','').replace('\n','').replace('\xa0', '').replace('    ', '').replace('\t','')
        detail.append(info)
    symptoms_data = {}
    symptoms_data['symptoms'] = symptoms
    symptoms_data['symptoms_detail'] = detail
    return symptoms, detail
```

图 3-1 部分爬虫代码

3.4 医疗领域知识图谱数据处理

医疗领域数据的特点是来源复杂，种类繁多，因此在构建中医领域的知识图谱时，需要先对数据进行处理。具体的处理过程包括以下几个步骤：

（1）利用正则表达式对各数据集进行文本分割和无关字符清除，进行比较筛选和去重操作，确保数据的准确性和完整性。

（2）根据设计好的知识图谱框架，移除不必要的数据，抽取所需信息。

（3）对于数据集中有多种表达方式的实体，进行合并，去除冗余数据，以确保知识图谱中的实体是准确的。

3.5 基于 neo4j 的知识表示与存储

知识表示和存储是构建知识图谱的重要环节，本文使用图形数据库 neo4j 实现医疗领域知识图谱的表示与存储。

neo4j 的存储模型基于图形结构，将数据存储在节点和关系上，每个节点和关系都可以有多个属性[13]。节点表示实体，关系表示实体之间的关系，属性表示实体和关系的属性。neo4j 将所有实体和关系都存储在磁盘上，可以通过索引和查询语句来访问和检索数据。由于 neo4j 采用了基于图形的存储模型，因此可以很方便地进行图形数据的可视化和分析。

neo4j 的使用方法比较简单，可以通过图形界面或者命令行来进行操作。在使用 neo4j 时，需要定义节点和关系的类型，以及节点和关系的属性。可以使用图数据库查询语言来进行数据的查询和分析。通过编写 Python 程序，可以将数据导入到 neo4j 数据库[14]。具体步骤包括：（1）定义实体、实体属性以及实体关系变量。（2）将实体和关系添加到图数据库中；（3）编写 Cypher 语句，使用 Cypher 语句查询数据。构建好的知识图谱示例见图 3-2。

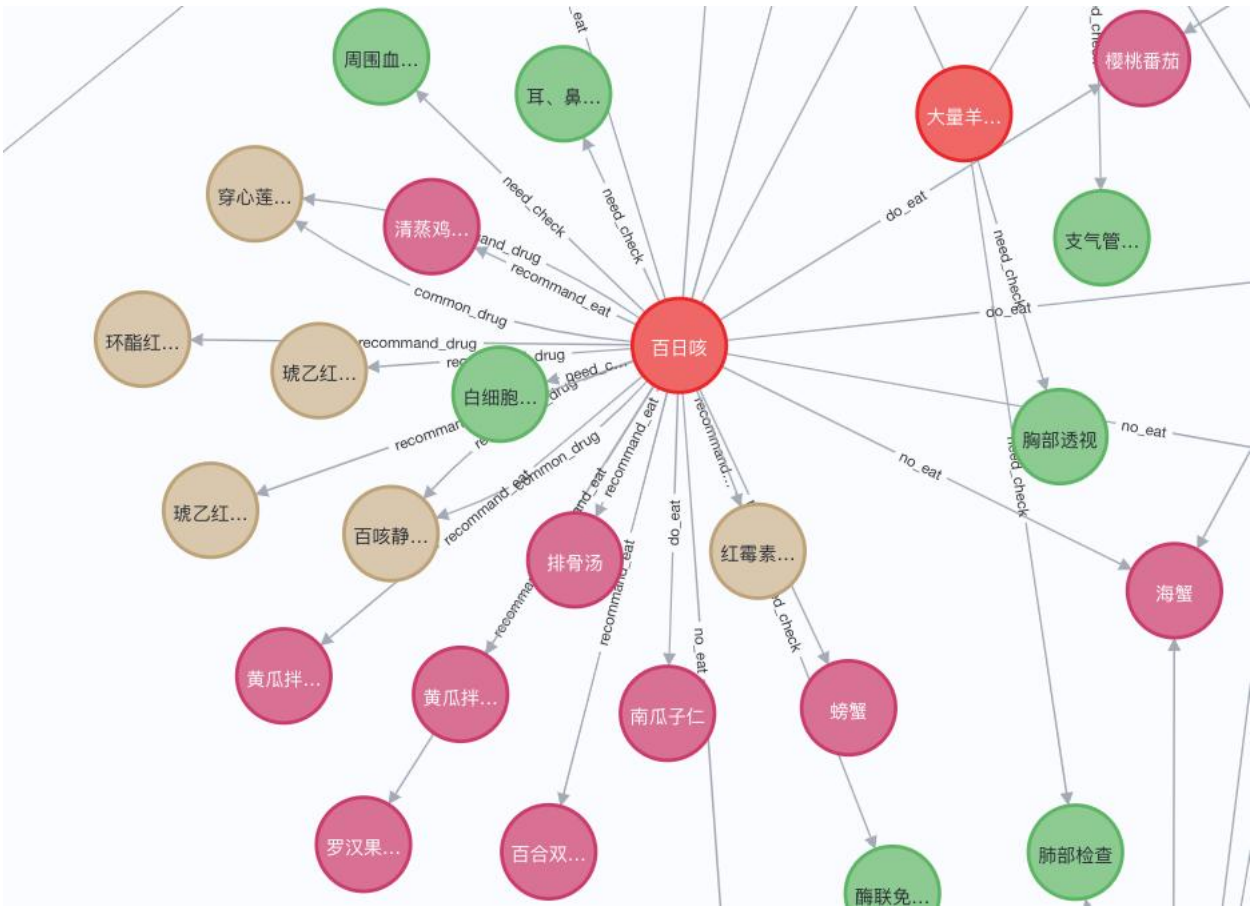

图 3-2 医疗领域知识图谱示例

图 3-3 所示函数用于在 Neo4j 数据库中创建实体节点。label 参数表示节点的标签（如"Disease"或"Symptom"），nodes 参数是一个包含节点名称的列表。对于列表中的每个节点名称，函数会创建一个具有相应标签和名称的节点，并将其添加到 Neo4j 数据库中。

```
'''建立节点'''
def create_node(self, label, nodes):
    count = 0
    for node_name in nodes:
        node = Node(label, name=node_name)
        self.g.create(node)
        count += 1
        print(count, len(nodes))
    return
```

图 3-3 创建实体部分代码

图 3-4 所示函数用于在图数据库中创建实体关系。start_node 和 end_node 分别是两个实体（如"Disease"或"Symptom"），edges 是包含相关联的实体的元组列表，rel_type 是关系的类型（如"has_symptom"），rel_name 是关系的名称（如"症状"），为了避免在创建关系时出现重复的关系，代码首先使用一个 set_edges 列表来存储唯一的关系。通过将每个关系元组中的两个节点名称用###连接成字符串，然后将这些字符串添加到 set_edges 列表中。接下来，将 set_edges 转换为集合以去除重复的关系。遍历去重后的关系集合，将每个关系字

符串拆分回节点名称。接着构建一个 Cypher 查询语句，该语句使用 MATCH 子句查找具有给定名称的起始节点和终止节点，并使用 CREATE 子句创建它们之间的关系。关系的类型和名称由 rel_type 和 rel_name 参数指定。使用 self.g.run(query)执行 Cypher 查询语句。如果查询成功执行，计数器 count 加 1。如果在执行查询时发生异常，将打印异常信息。通过调用 create_relationship 函数并传递适当的参数，可以在 Neo4j 数据库中创建实体之间的关系。

```
'''创建实体关联边'''
def create_relationship(self, start_node, end_node, edges, rel_type, rel_name):#起点节点，终点节点，边，关系类型，关系名字
    count = 0
    # 去重处理
    set_edges = []
    for edge in edges:
        set_edges.append('###'.join(edge))#使用###作为不同关系之间分隔的标志
    all = len(set(set_edges))
    for edge in set(set_edges):
        edge = edge.split('###')#选取前两个关系，因为两个节点之间一般最多两个关系
        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)#match语法，p，q分别为标签，rel_type关系类别，rel_name 关系名字
        try:
            self.g.run(query)#执行语句
            count += 1
            print(rel_type, count, all)
        except Exception as e:
            print(e)
    return
```

图 3-4 创建实体关系部分代码

第 4 章 基于医疗领域知识图谱的问答系统

4.1 设计背景

医学是一门研究人类健康和疾病的科学，对于人类的健康和生命具有重要意义。医学研究人类健康和疾病的成因、诊断、治疗和预防方法，为人类提供了保障健康的知识和技术。医学的发展和进步使得许多疾病可以得到有效的治疗。这些疾病对人类健康产生的威胁得到了有效控制。医学的发展不仅能够治疗疾病，还能够预防疾病的发生，例如疫苗接种、生活习惯的调整、健康检查等，这些措施对于预防疾病具有重要意义。医学学的发展不仅能够保障人类健康，还能够增强人类的生命质量和福祉。医学对于人类健康和生命具有重要意义，不断的发展和进步，将为人类的健康和生命贡献更多的力量。

医疗领域工具的发展对于医学的发展有着重要的推动作用。随着医疗领域工具的不断发展，医疗技术水平也会不断提高。医疗领域工具的发展能够提高医疗效率。例如，电子病历系统可以减少医生的工作量，提高工作效率，使医生能够更加专注于患者的治疗。而基于医疗领域知识图谱的问答系统是医疗领域发展的一大重要工具。

随着医疗技术的不断发展，医疗领域的知识量不断增加，医生和患者需要快速准确地获取医疗知识。传统的医疗问答方式主要依赖于医生的经验和患者的描述，存在着信息不对称、主观性强、效率低下等问题。通过对知识图谱的构建、维护和查询，实现快速且准确地获取医疗知识。日常生活中的一些简单的医学问题可以通过本文设计的问答系统得到解答。

4.2 设计目标

本文设计的基于医疗领域知识图谱的问答系统，旨在能够实现以下目标：

（1）实现医疗领域知识图谱的构建和维护，包括医疗实体、属性和关系的抽取和存储。这一环节的主要内容是收集医疗网站上的医疗信息，并将其存储到图数据库中。网络上的医疗信息质量良莠不齐，普通病人难以辨别真假，通过构建医疗领域知识图谱，搜集可信度高的信息，

（2）实现医疗领域知识图谱的查询和分析功能，包括实体和关系的查询和分析，以及知识图谱的可视化展示。运用图数据库将众多的医疗信息表示为节点和边之间的关系，是的疾病、药品等实体之间的关系一目了然，方便普通病人进行判断。

（3）实现基于医疗知识图谱的自然语言问答系统，能够通过自然语言输入，快速准确地回答医疗领域的问题。普通病人可以通过问答系统查询自身疾病，病进行自我治疗，可以减轻医院的负担。

4.3 系统结构

本文设计的基于医疗领域知识图谱的问答系统，主要包括以下模块：

（1）数据抽取和存储模块。该模块主要负责医疗领域实体、属性和关系的抽取和存储。通过网络爬虫收集医疗网站上的疾病、药品等医疗信息，对这些数据进行处理后存储到知识图谱数据库中，并使其可视化。

（2）知识图谱查询和分析模块。该模块主要负责医疗领域知识图谱的查询和分析。可以通过 Cypher 查询语言和图形数据库工具，对知识图谱中的实体和关系进行查询和分析，并通过可视化展示，呈现知识图谱的结构和属性。

（3）自然语言处理模块。该模块的主要功能是自然语言处理技术的集成和应用。将用户输入的自然语言问题转化为查询语句，实现自然语言问答功能。

4.4 医疗领域对话系统的实现

医疗问题属于封闭领域场景，本文中的问答系统基于规则匹配实现。首先对领域问题进行详尽分类，穷举出所有可能的问题关键词，再对用户提出的问题进行分析和归类，接着利用 Cypher 语句查询图数据库，根据返回数据构建答案。最后返回结果。问答系统实现流程见图 4-1。

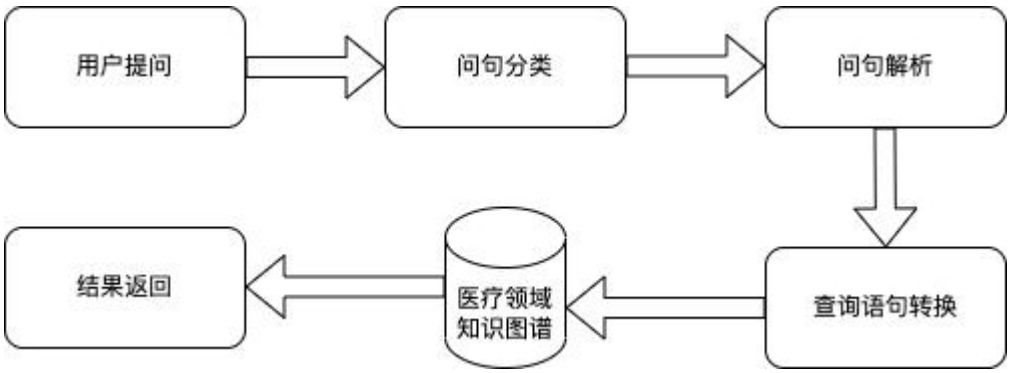

图 4-1 问答系统运行流程图

图 4-2 所示函数是问答系统的核心部分用于处理用户输入的问题并返回相应的答案。answer 变量初始化为一个默认答案，即"没能理解您的问题，我的词汇量有限，请输入更加标准的词语"。调用 self.classifier.classify(sent)方法对用户输入的问题（sent 参数）进行分类。res_classify 变量存储分类结果。如果 res_classify 为空（即问题未被分类），函数将返回默认答案。如果问题被成功分类，调用 self.parser.parser_main(res_classify)方法对分类结果进行解析。res_sql 变量存储解析结果，它是一个包含 Cypher 查询语句的字典。调用 self.searcher.search_main(res_sql)方法在知识图谱中搜索答案。final_answers 变量存储搜索结果，它是一个包含答案的列表。如果 final_answers 为空（即没有找到合适的答案），函数

将返回默认答案。否则，使用'\n'.join(final_answers)将答案列表连接成一个字符串，并将其作为最终答案返回。

```
def chat_main(self, sent):
    answer = '没能理解您的问题，我的词汇量有限，请输入更加标准的词语'#这是初始答案
    res_classify = self.classifier.classify(sent)#'sent'是用户的输入内容，利用classify函数先对其进行分类
    if not res_classify:
        return answer#没有找到对应分类内容，返回初始答案
    res_sql = self.parser.parser_main(res_classify)#调用parser_main对内容进行解析
    final_answers = self.searcher.search_main(res_sql)#对内容搜索合适的答案
    if not final_answers:
        return answer#如果没有找到合适的最终答案，返回初始答案
    else:
        return '\n'.join(final_answers)#连接字符
```

图 4-2 问答系统流程代码

图 4-3 是问句分类部分代码，这段代码定义了一个名为 classify 的函数，用于对用户输入的问题进行分类。question 表示用户输入的问题。创建一个名为 data 的字典，用于存储问题的分类结果。调用 self.check_medical(question)方法检查问题中出现的实体。medical_dict 变量存储实体检查的结果。如果 medical_dict 为空（即问题中没有实体），函数返回一个空字典。遍历 medical_dict 中的值（实体类型列表），将它们添加到 types 列表中。将问题类型 question_type 初始化为"others"。创建一个名为 question_types 的列表，用于存储问题的所有可能类型。如果问题包含症状关键词（通过调用 self.check_words(self.symptom_qwds, question)检查）并且实体类型列表 types 中包含"disease"，则将问题类型设置为"disease_symptom"，并将其添加到 question_types 列表中。在这段代码中，只展示了一个问题类型的检查（症状）。实际上，classify 函数会检查多种问题类型（如疾病原因、预防措施等），并将所有可能的问题类型添加到 question_types 列表中。通过调用 classify 函数并传递用户输入的问题，可以获取问题的分类结果，包括涉及的实体和问题类型。

```
    '''分类主函数'''
    def classify(self, question):
        data = {}
        medical_dict = self.check_medical(question)#调用下面定义的check_medical问句过滤函数
        if not medical_dict:
            return {}
        data['args'] = medical_dict
        #收集问句当中所涉及到的实体类型
        types = []
        for type_ in medical_dict.values():
            types += type_
        question_type = 'others'#这句话无意义

        question_types = []

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

图 4-3 问句分类部分代码

图 4-4 是问句解析部分代码，这段代码定义了一个名为 sql_transfer 的函数，用于根据问题类型和实体生成相应的查询语句。question_type：表示问题类型的字符串。entities：包含问题中提到的实体名称的列表。如果 entities 列表为空，函数将返回一个空列表。函数使用一系列 elif 语句检查 question_type 参数的值。针对每种问题类型，它都会构建一个 Cypher 查询语句，并将其添加到 sql 列表中。如果问题类型是"disease_cause"，查询语句将返回疾病的原因；如果问题类型是"disease_prevent"，查询语句将返回疾病的防御措施；如果问题类型是"disease_lasttime"，查询语句将返回疾病的持续时间,根据不同的问题类型选择相应的查询语句，函数返回包含 Cypher 查询语句的 sql 列表。

```
'''针对不同的问题，分开进行处理'''
def sql_transfer(self, question_type, entities):
    if not entities:
        return []

    # 查询语句
    sql = []
    # 查询疾病的原因，在debug的时候，运行到对应的elif（说明已经找到合适的关系）会自动停止该函数的执行
    if question_type == 'disease_cause':
        sql = ["MATCH (m:Disease) where m.name = '{0}' return m.name, m.cause".format(i) for i in entities]#调用match语句

    # 查询疾病的防御措施
    elif question_type == 'disease_prevent':
        sql = ["MATCH (m:Disease) where m.name = '{0}' return m.name, m.prevent".format(i) for i in entities]

    # 查询疾病的持续时间
    elif question_type == 'disease_lasttime':
        sql = ["MATCH (m:Disease) where m.name = '{0}' return m.name, m.cure_lasttime".format(i) for i in entities]

    # 查询疾病的治愈概率
    elif question_type == 'disease_cureprob':
        sql = ["MATCH (m:Disease) where m.name = '{0}' return m.name, m.cured_prob".format(i) for i in entities]
```

图 4-4 问句解析部分代码

图 4-5 是答案生成部分代码，这段代码定义了一个名为 answer_prettify 的函数，用于根据问题类型和查询结果生成格式化的回答。question_type：表示问题类型的字符串（如"disease_symptom"、"symptom_disease"等）。answers：包含查询结果的列表。如果 answers 为空（即没有查询结果），函数返回一个空字符串。根据问题类型生成格式化的回答：函数使用一系列 elif 语句检查 question_type 参数的值。针对每种问题类型，它都会从 answers 列表中提取相关信息并生成一个格式化的回答。如果问题类型是"disease_symptom"，函数将提取疾病名称（subject）和症状列表（desc），并生成一个描述疾病症状的回答。函数返回生成的格式化回答（final_answer）。通过调用 answer_prettify 函数并传递问题类型和查询结果，可以生成一个易于理解的、格式化的回答。

```
'''根据对应的qustion_type，调用相应的回复模板'''
def answer_prettify(self, question_type, answers):
    final_answer = []
    if not answers:
        return ''
    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]))

    elif question_type == 'disease_cause':
        desc = [i['m.cause'] for i in answers]
        subject = answers[0]['m.name']
        final_answer = '{0}可能的成因有：{1}'.format(subject, '；'.join(list(set(desc))[:self.num_limit]))
```

图 4-5 答案生成部分代码

本文实现的对话系统支持 17 类问题的回答，问答类型与示例见表 4-1。

表 4-1 问答系统支持的问答类型

序号	问答类型	问句示例
1	疾病症状	肺气肿的症状是什么
2	已知症状找可能疾病	胸痛是什么病
3	疾病病因	为什么会得放射性肺炎
4	疾病的并发症	大叶性肺炎有哪些并发症
5	疾病不能吃的食物	得了肺气肿不能吃什么
6	疾病建议吃什么食物	得了肺气菌病应该吃什么
7	某食物哪些病不该吃	什么人不该吃蜂蜜
8	某食物哪些病应该吃	香蕉有什么好处
9	某疾病应该吃什么药	肺气肿要吃什么药
10	某药物能治什么病	甲硝锉片能治什么病
11	某病需要做什么检查	肺气肿怎么才能查出来
12	某检查能查什么病	胸部平扫能查出什么病
13	预防措施	怎样才能预防肺大疱
14	治疗周期	放射性肺炎要多久才能好
15	治疗方式	肺气肿怎么治
16	治愈概率	肺气肿能治好吗
17	疾病易感人群	什么人容易的肺气肿

4.5 系统的结果展示

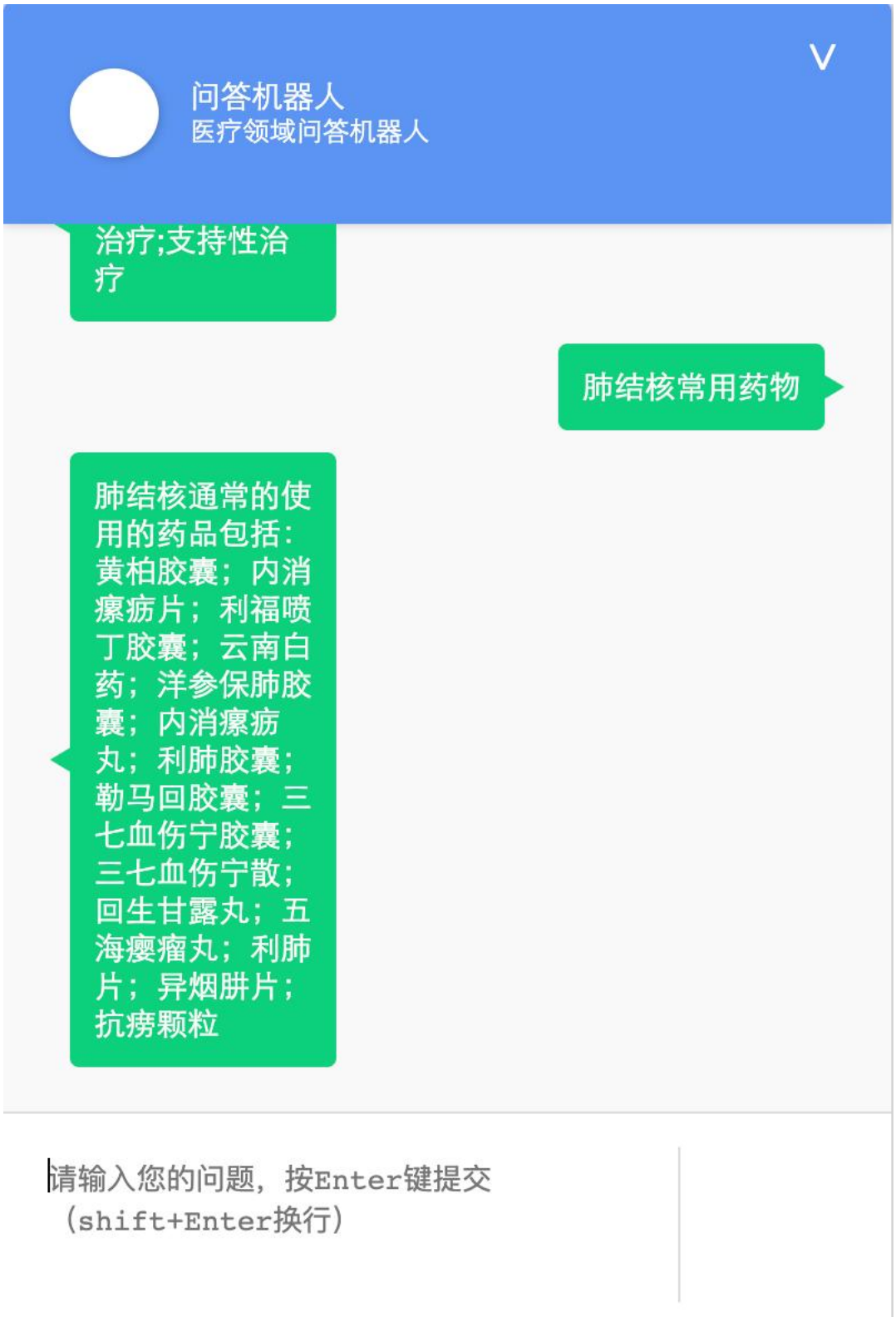

图 4-6 问答系统程序运行结果

结论

近年来，人工智能和自然语言处理技术的发展极大地提高了人机交互的准确性和效率。其中最有前途的应用之一是问答系统。问答系统已经应用于各个领域，如客户服务、教育和医疗保健。特别是在医疗保健行业中，准确获取和处理医疗知识的需求日益增加，医疗问答系统具有广阔的应用前景。本文的主要研究工作是基于医疗领域知识图谱的问答系统。知识图谱是一种新兴的知识表示和管理方法，能够将人类知识以图谱的形式进行表达，具有可扩展性和可查询性等优点。

本文通过构建医疗领域的知识图谱，实现了基于自然语言的医疗问答功能。主要完成了以下工作：

（1）本文构建了医疗领域的知识图谱。知识图谱的构建包括数据采集与处理、实体识别与链接、属性抽取与关系建模等步骤。在医疗领域，本文利用网络爬虫，对医疗网站上的疾病、药品等信息进行收集，并对数据进行处理和分析，提取出实体、属性和关系，并将其在图数据库中进行存储和表示。

（2）本文设计了基于医疗领域知识图谱的问答系统。实现了基于规则的对话功能，能够以自然语言的形式回答用户的问题。系统采用了三层架构，包括自然语言处理层、知识库查询层和应答生成层。用户可以通过提出问题的方式，向系统查询医疗知识，并得到准确的回答。

本文的工作还可以从以下方面进一步改进和完善。首先，可以优化知识图谱的构建过程，提高知识图谱的覆盖范围和准确性。其次，可以进一步完善问答系统的功能，增加多轮对话、推荐、个性化等功能。最后，可以将基于医疗领域知识图谱的问答系统应用于其他领域，如教育、金融等。

参考文献

[1] 钱涛,卢方超,韩梓政,戴文华.基于知识图谱的豆瓣读书问答系统 [J] .湖北科技学院学报,2022 42(245)：06 153-157+162.

[2] 徐涌鑫,赵俊峰,王亚沙,谢冰,杨恺.时序知识图谱表示学习 [J] .计算机科学,2022 49 09 168-177.

[3] 林健,柯清超,黄正华,鲍婷婷.学科知识图谱的动态生成及其在资源智能组织中的应用 [J] .远程教育杂志,2022 40(271)： 04 25-36.

[4] 郑泳智,朱定局,吴惠粦,彭小荣.知识图谱问答领域综述 [J] .计算机系统应用,2022 31 04 5-17.

[5] 周毅,刘峥,粟小青,金体成.融合多层次数据的问答知识图谱本体模型构建 [J] .图书情 报工作,2022 66(690)： 05 127-134.

[6] 曾子玲,张华敏,于彤,刘思鸿,张磊,高宏杰,陈广坤,佟琳.知识图谱及其关键技术在中医 药领域的研究与应用综述 [J] .世界科学技术-中医药现代化,2022 24 02 342-350.

[7] 论兵,王月春,郝晓慧,谷斌,王会勇.知识图谱问答研究进展 [J] .软件导刊,2022 21(233)： 03 232-242.

[8] 马昂,于艳华,杨胜利,石川,李劼,蔡修秀.基于强化学习的知识图谱综述 [J] .计算机研究与发展,2022 59 08 60-88.

[9] 王松,李正钧,杨涛,胡孔法.中医药知识图谱研究现状及发展趋势 [J] .南京中医药大学 学报,2022 38 03 92-98.

[10] 王萌,王昊奋,李博涵,赵翔,王鑫.新一代知识图谱关键技术综述 [J] .计算机研究与发 展,2022 59 09 83-101.

[11] 萨日娜,李艳玲,林民.知识图谱推理问答研究综述 [J] .计算机科学与探索,2022 16(167)： 08 51-65.

[12] 赵京胜,宋梦雪,高祥,朱巧明.自然语言处理中的文本表示研究 [J] .软件学报,2022 33 01 106-132.

[13] 曹亚如,张丽萍,赵乐乐.多轮任务型对话系统研究进展 [J] .计算机应用研究,2022 39(364)：02 17-27.

[14] 王为昌. 基于中医知识图谱的自动问答系统[D].北京交通大学,2022.DOI:10.26944/d.cnki.gbfju.2022.001906.

致谢

感谢闫蕾老师在研究过程中给予了我宝贵的指导以及帮助。