基于可满足性问题求解的辅助诊断系统的设计与实现

摘　要

近年来，“智慧医疗”概念逐渐兴起。本文主要针对智慧医疗中的辅助诊断场景，即机器可以根据患者的症状信息对其所患疾病自动地做出预测。现在市面上的辅助诊断产品采用的解决方案基本都是大数据 + 深度学习的方案，即采用深度学习的方法从大量的电子病历中学习出疾病和症状的关联模型，然后使用模型根据患者的症状信息对其所患疾病做出预测。这种方案的优势在于数据量大，几乎可以包含所有的常见疾病及症状。但是其问题也很明显，首先就是严重依赖病历，而病历可能无法覆盖罕见疾病。其次就是深度学习的预测结果缺乏可解释性。最后是数据的可靠性问题，病历中的数据可能存在错误，如何甄别这些错误是大数据方案无法解决的。

针对大数据方案存在的问题，本文提出了新的辅助诊断解决方案：首先从可信的医学资源 (比如医书、专业的医学知识库) 中抽取医学知识，并对医学知识进行审核验证保证可靠性。然后将医学知识编码成命题逻辑公式，组合这些逻辑公式构建可满足性问题的实例。最后采用 SMT 求解器去求解该实例，根据求解器的解来进行诊断。因为我们主要从医书等可信的知识源中抽取知识并且有审核过程，所以数据的可靠性和覆盖面都优于病历。而且因为我们从医书中提取包含因果关系的医学知识，并采用逻辑推理的方式模拟诊断过程，所以我们的诊断结果都有医书中的知识作为证据支撑，保证了诊断结果的可解释性。此外我们还提出了知识纠错的方案，可以对知识抽取过程中引入的错误进行纠正。

本文的主要贡献如下：

(1) 医学知识库的构建

为了实现本文的辅助诊断系统，我们首先构建了符合我们需求的

医学知识库。根据预先设计好的医学知识模型，从可信知识源中抽取医学知识，最终表达成医学知识图谱并存储到图数据库 Neo4j 中，供后续的算法使用。

(2) 辅助诊断内核

基于已经构建好的知识库，本文实现了辅助诊断的内核。辅助诊断内核可以根据患者的主述信息从知识库中提取相关的医学知识并编码成逻辑公式，然后组合这些逻辑公式构建可满足性问题的实例，最后使用 SMT 求解器求解其可满足性来进行诊断。

(3) 知识纠错方案

虽然我们从可信知识源抽取知识并进行审核，但由于人工或者机器自动抽取算法的缺陷，所以最终获取的知识还是可能存在错误。因为本文采用的是基于医学知识进行诊断而非大数据方案，错误的知识会很大程度影响最终诊断的准确性，所以必须对知识库中的错误知识进行纠正，本文就基于可满足性问题中的不可满足核 (unsatisfiable core) 技术，提出了知识库中矛盾知识的纠错方案。

(4) 实际应用的辅助诊断系统

本文的所有方案不仅仅是停留在理论原型层面，而是实现出了实际可用的辅助诊断系统，并且已经在合作医院中落地测试。

关键词：智慧医疗, 知识图谱, 知识纠错, 可满足性问题, SMT 求解器

DESIGN AND IMPLEMENTATION OF COMPUTER AIDED DIAGNOSIS SYSTEM BASED ON SATISFIABILITY PROBLEM SOLVING

ABSTRACT

In recent years, the concept of "intelligent healthcare" has gradually emerged. This paper focuses on the computer-aided diagnosis scenario in intelligent healthcare, that is when a computer can automatically diagnose disease based on patient's symptom information. For the time being, the issue of computer-aided diagnosis is solved with big data + deep learning method being involved, that is, deep learning solution is used to learn the association model of diseases and symptoms from a big data of electronic medical records, with further diagnosis based on the patient's symptoms. The distinctive feature of this method is that a large amount of data can contain almost all common diseases and symptoms. But the out coming issues are also very clear. First, this method relies heavily on medical records, which may not cover rare diseases. Second, the results received from the deep learning lack correct interpretation. And finally, there is a problem of data reliability as there may occur errors in the medical records. Unfortunately, big data cannot provide a solution to detect the errors from the correct data.

Focusing on the problems of big data solution, this paper proposes a new computer-aided diagnosis method: first, extract medical knowledge to create a valid knowledge base (KB) from trusted medical resources (such as medical books, professional medical knowledge bases), and verify such a knowledge base to ensure its reliability. Second, encode medical knowledge into propositional logical formulas, and combine these logical formulas to

build instances of satisfiability problem. Finally, the SMT solver is used to find plausible answers for these instances, and on the basis of the found solutions, diagnose a patient. Because we mainly form knowledge base from credible knowledge resources (such as medical books) and have a review process, the reliability and coverage of the received data are better than the one received from medical records. And due to the fact that the medical KB that we possess is based on causal relationships retrieved from the medical books and the fact that we use logical reasoning to simulate diagnosis procedure, our diagnosis results are supported by knowledge from the medical books, ensuring the possibility to interpret the diagnosis results correctly. In addition, we also propose a knowledge error correction scheme which can calibrate the results received in the process of knowledge extraction.

The main contributions of this paper are as follows:

(1) Creation of a medical knowledge base

In order to implement the computer-aided diagnosis system described in this paper, we first build a medical knowledge base that meets our requirements . According to the pre-designed medical knowledge model, the necessary data is extracted from a trusted knowledge source, then expressed as a medical knowledge graph and stored in the graph database Neo4j for subsequent algorithms usage.

(2) Computer-aided diagnosis core

Based on the well-built knowledge base, this paper implements the computer-aided diagnosis core. It can extract relevant medical knowledge from the knowledge base on the basis of the patient's symptoms and encode it into logical formulas. And it then combines these logical formulas to construct instances of the satisfiability problem. Finally, it uses the SMT solver to calibrate its satisfiability in order to perform diagnosis.

(3) Knowledge error correction scheme

Although we extract and review knowledge from trusted sources, due

to the disadvantages of manual or automatic extraction algorithms, there still may be errors in the final acquired knowledge. Because this paper uses medical knowledge-based diagnose methods rather than big data solution, erroneous knowledge will greatly affect the accuracy of the final diagnosis. Therefore, such an instance must be corrected. This paper is based on the unsatisfiable core of satisfiability problem to propose a way to correct the contradictory knowledge in the knowledge base.

(4) Computer-aided diagnosis system for practical application

All the schemes in this paper are not only theoretical, but are also implemented in computer-aided diagnosis systems and have been tested in cooperative hospitals.

KEY WORDS: Intelligent Healthcare, Knowledge Graph, Knowledge Error Correction, Satisfiability Problem, SMT Solver

目 录

插图索引

表格索引

算法索引

第一章　绪论

本章是绪论部分。在本章节中, 首先介绍研究的背景及研究意义, 并对国内外与本文相关的研究现状进行了总结分析，然后详细的说明一下本文主要的研究内容, 最后是本文的章节安排。

1.1　研究背景及意义

多年以来，医疗一直都是困扰民生的痛点问题之一。由于国内医疗卫生体系的不完善，人均医疗资源少、就医渠道窄、医疗网点覆盖面低等问题困扰着大众民生。当前国内的医疗问题主要分为三类:

1) 首先是医疗资源的匮乏。根据《2018 年中国卫生统计年鉴》统计数据，截至 2018 年 11 月底，全国医院数量达 3.2 万个，其中公立医院 12072 个，民营医院 20404 个。医院中总床位数为 612 个，每千人卫生执业医师为 2.44 个，其中城市为 3.97 个，农村为 1.68 个。考虑到我国庞大的人口基数，民众能够享受到的人均医疗资源非常有限。
2) 其次是医疗资源分配严重不均。在医疗资源本来就相对匮乏的情况下，资源分配不均更加重了民众对国内医疗体系的不满意度。在《2018 年中国卫生统计年鉴》的每项数据中，城乡之间的差异都有 2 – 5 倍之多。除了城乡差异，城市与城市之间，医院与医院之间也有很大差异。优质的医疗资源几乎全部集中在发达城市的三甲医院，好医生，好药方，好设备都在这些医院当中。这就造成了很多民众对基层医疗卫生机构不信任，不算严重的病也要去三甲医院就诊，导致大医院人满为患，社区医院无人问津，一定程度上造成了医疗资源的浪费。
3) 最后是各地各医院医疗系统的碎片化。各医院内部都有一套自己的诊疗数据系统，之间的信息格式不统一，医疗资源不共享，导致医疗服务，医疗支付之间数据没有打通，即存在“数据孤岛”现象。

在这样的医疗现状之下，“智慧医疗”的概念应运而生。智慧医疗应用场景包括医疗系统的信息化管理、医学的影像诊断、医疗的辅助诊断、疾病的风险预测和药物的挖掘等。其中医疗系统的信息化管理可以一定程度上解决“数据孤岛”问题，通过统一的医疗系统将各地医院的医疗数据、诊疗流程信息化，提高医院诊疗效率的同时，也方便了各医院之间的数据共享。除此之外，辅助诊断也是智慧医疗中非常重要的应用场景之一。辅助诊断表示机器可以根据患者的症状信息

对其所患疾病自动地做出预测，而无须医生的参与，这就极大程度缩短了患者的就诊时间，也缓解了医疗资源匮乏的压力。本文就是针对辅助诊断的场景，设计并实现了实际可用的辅助诊断系统。对于不算严重的疾病，比如感冒发烧，本文的系统可以直接给出诊断结果，这样患者可以直接开具药品进行治疗，无须再到医院排队等待。对于严重的疾病，比如肺结核，本文的系统可以给出初步的诊断，并推荐相应的检查与就诊科室，患者可以做完检查后到对应科室做更详细的诊断，这样就简化了“问诊-检查-复诊-取药”的一般就诊流程，节省了患者的时间，也减轻了医生的负担。

现在市面上的智慧医疗产品大多是在做医疗系统的信息化管理，而为数不多的辅助诊断系统采用的基本都是大数据 + 深度学习的解决方案：即从大量的电子病历中提取相互关联的疾病和症状，然后采用深度学习的方法训练出一个疾病和症状的关联模型，通过模型就可以根据患者的症状信息推断出其所患疾病。这种方案的优势在于数据量大，几乎可以完全覆盖常见的疾病和症状。但是其问题也很明显，首先是严重依赖电子病历，而病历中一般都是常见疾病，对于罕见疾病难以覆盖。其次就是其诊断结果的可解释性，大数据 + 深度学习方案旨在找到数据之间的关联性，而没法获取数据之间的因果关系。比如大数据方案根据“低热，盗汗，咯血”可能给出正确的诊断结果“肺结核”，但是为什么给出这样的诊断缺乏医学知识的支持，可能引起患者的不信任感。最后是数据的可靠性问题，我国的病历电子化进程尚处于起步阶段，很多医生还不习惯电子病历，书写时敷也了事，这样就造成电子病历中可能存在错误，如何甄别并修改这些错误是大数据方案无法解决的。

综上所述，本文针对大数据 + 深度学习方案所面临的问题，提出了一个新的辅助诊断解决方案：通过从医书等可信医学知识源中提取医学知识，然后将其表达成逻辑公式，通过逻辑推理来进行辅助诊断。具体来说：首先从可信的医学资源中提取医学知识，并对医学知识进行审核及验证保证其可靠性。然后将医学知识表达成命题逻辑公式，组合这些逻辑公式构建可满足性问题的实例。最后采用 SMT 求解器去求解该实例，根据求解器的解来做出诊断。因为本文的所有数据都是系统的医学知识而非病历这样的经验数据，而且采用逻辑推理来模拟诊断过程，所以我们的诊断结果都有医书中医学知识作为证据支撑，保证了结果的可解释性。而且因为本文的辅助诊断系统会从医书中提取医学知识并进行审核，还会有专门的知识纠错模块对已有的知识进行纠错，就在一定程度上保证了数据的可靠性和全面性。本文通过医学知识库，辅助诊断内核以及知识纠错模块搭建了完整的辅助诊断系统，并且已经在合作医院中部署测试。

1.2 国内外研究现状

由于本文采用从医书中提取医学知识构建医学知识库，然后根据知识库中知识进行辅助诊断的方案。而医学知识库的构建，辅助诊断其实是两个相对独立的研究领域，目前工业界和学术界在这两个领域都有了一些较为突出的研究成果和产品。本节会分别对医学知识库构建和辅助诊断两个领域的研究现状进行介绍，同时会总结两个领域目前的研究优势以及仍然存在的挑战和可以进一步优化的地方。

1.2.1 医学知识库

知识库[1]（Knowledge base) 是用于知识管理的一种特殊的数据库，以便于有关领域知识的采集、整理以及提取。知识库起源于早期的专家系统，专家系统依赖知识库中的事实与规则进行推理问答，而后来人工智能技术的火热也进一步推动了知识库研究的发展。广义而言，知识库就是知识工程的结构化，是易操作、易使用、全面有组织的知识集群，是为了适应某一特定领域的需要，采用特定的知识表达方式在计算机中存储的结构化、相互联系的知识集合。而医学知识库[2]，顾名思义，就是针对医学领域人工智能的需要而构建的知识库。其中的知识多为医学知识，比如药物，病症，诊疗等维度的知识。医学知识库是智慧医疗应用的基石，可以为机器阅读理解医学文本、智能咨询、智能诊断提供知识基础。医学知识库的研究在国内外都较为火热，本节会对国内外的研究成果做详细的介绍。

1.2.1.1 SNOMED Clinical Terms

SNOMED CT[3]（Systematized Nomenclature of Medicine – Clinical Terms，医学系统命名法——临床术语），是一部为了便于计算机自动处理而进行严格系统编排的医学术语集，其囊括了大部分的医学临床信息，如疾病、症状、检查、部位、药物等。使用 SNOMED CT 术语集，各专业、各学科、各领域的医学从业者和研究者可以非常方便地进行临床数据的标注、检索和共享。

SNOMED 起源于 1965 年美国病理学院（CAP）开发的 Systematized Nomenclature of Pathology (SNOP)[4]，1975 年 CAP 将 SNOP 和 SNDO[5]（New York Academy of Medicine's Standard Nomenclature of Diseases and Operations）整合成了 SNOMED，SNOMED 起初只在医学学术领域有一定的反响，在实际医学场景中并没有太多应用。但是在 1997 年，SNOMED RT[6]（SNOMED Reference Terminology）横空出示，并且在 2002 年与英国国家卫生服务部的临床术语（National Health Service Clinical Terms）合并成了 SNOMED CT，使其一举成为世界上最全面，最专业，最权威的

医学术语集。这套术语集提供了一套全面统一的医学术语系统，使得全世界各地各医院、各医疗机构之间的信息交换成为可能。例如对于肿瘤医师来说，肝癌、肝肿瘤、liver cancer 可能表达的是同一含义，但是计算机可能把它们当成完全不同的医学词汇来处理，而通过 SNOMED CT 规定的标准术语，各地医疗机构记录的医疗信息可以使用统一的术语集合和数据组织结构，极大程度方便了不同的医疗机构、研究人员以及其他相关方协调一致地交换临床信息。

SNOMED CT 中的逻辑模型如图 1–1所示，其组成要素为：

- 概念：每个全局唯一的数字代码，全局唯一的名称和描述所指定的基本含义单位。概念可以理解为医学中标准的临床术语，每个概念都有全局唯一的概念码，但每个概念可能会有一个或多个描述。比如 “rhinitis”（鼻炎），在 SNOMED CT 术语集中是基本概念。
- 描述：赋予同一概念的不同术语或名称（同义词）。比如对于 “rhinitis”（鼻炎），在实际应用中，它将会有多种不同的术语表达，如 “coryza”、“nasitis” 等，但它们并不是概念，它们只是对于概念”rhinitis” 的描述或者同义词。
- 关系：用于在同一层级结构之内或不同层级结构之间将不同的概念联系起来。在 SNOMED CT 中，主要有两种关系：
  - IS-A 关系，使用在同一个层面中，表示某些概念间的关系。如图 1–2所示，痔疮属于骨盆血管疾病，这样痔疮 → 骨盆血管疾病就形成了一种 IS-A 关系。
  - Attribute 关系，表示跨层面的概念间的关系，如图 1–2所示，从部位来看，“痔疮” 发生在身体部位 “痔丛” 处，所以 “痔疮” 就会有一个 “部位” 属性，属性的值是 “痔丛”。

经过多年的发展，SNOMED CT 目前包括大约 321900 条概念、超过 80 万条临床概念相关的描述，和超过 700 万条进一步描述概念的关系，其已成为世界上最权威，最全面，最专业的多语言医学知识库。SNOMED CT 已经被广泛用于电子病历生成，医学报告，基因数据库等多个领域。但是 SNOMED CT 在中国却没有太大的反响，首先因为是版权问题，中国政府并不是 SNOMED International 的会员，所以大陆机构在使用 SNOMED CT 时需要缴纳高昂的费用。其次是虽然有一些民间组织在做 SNOMED CT 的本地化，但是其质量和速度一直难以保证。所以说现在 SNOMED CT 在中国几乎没有太多实际的应用，国内的医疗科研机构并不会直接使用 SNOMED CT，而更多是参考 SNOMED CT 对知识的组织方式构建自己的医学知识库。

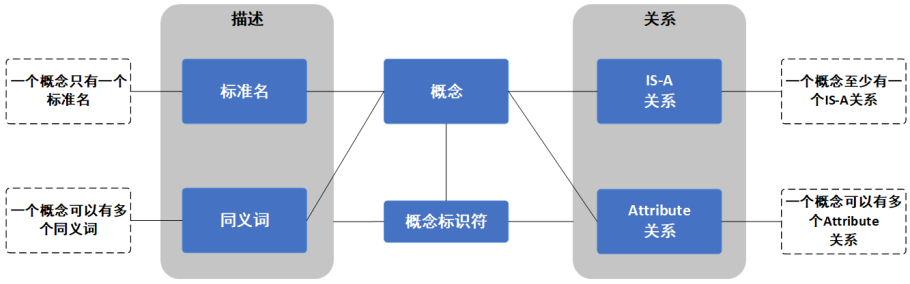


图 1–1 SNOMED CT 逻辑模型

Figure 1–1 The logic model of SNOMED CT

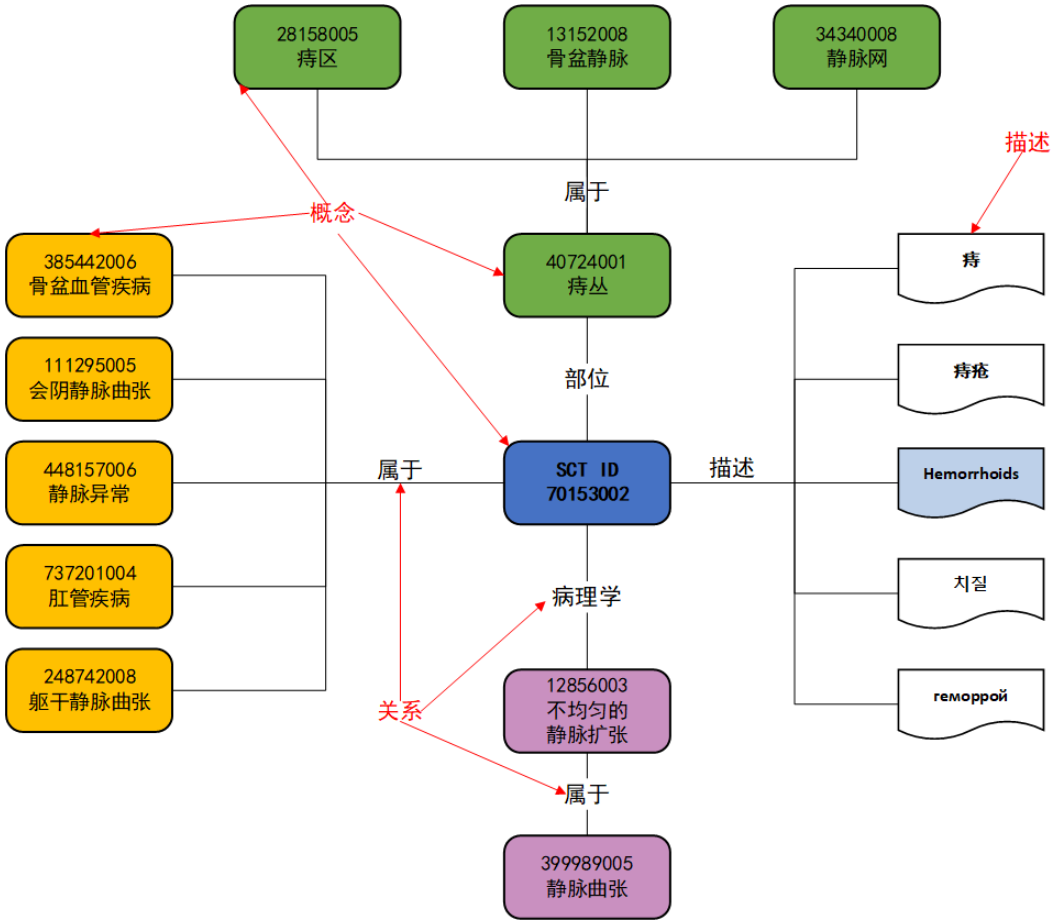


图 1–2 SNOMED CT 中“痔疮”概念的结构关系示例

Figure 1–2 An example of the relationship structure of the concept ”Hemorrhoids” in SNOMED CT

1.2.1.2 BMJ Best Practice

BMJ Best Practice 临床实践（简称 BP）是英国医学杂志 (BMJ) 出版集团于 2009 年 2 月出版的循证医学[7] 数据库资源的升级版。它在 BMJ Clinical Evidence（临床证据）中的治疗研究证据的基础上，增添了由全球知名学者和临床专家执笔撰写的，以个体疾病为主题，涵盖基础知识、诊断方法、治疗步骤和资源等各个关键环节的内容 (如图 1–3所示)。BP 的主要特点及优势有：

a) 收录上千种的临床疾病，80% 以上是常见疾病。
b) 每一种疾病指南都由世界顶尖临床专家撰写，并经过同行评审完成，使得其数据质量非常之高。
c) 收录上万种诊断方法。包括临床鉴别诊断、病史和查体、检查、鉴别做诊断和诊断标准等核心内容。
d) 收录了数千项的国际治疗指南和诊断标准的全文内容；此外还提供了大量的临床彩色图像。
e) 提供最新的药物副反应和多种药物相互作用的最新证据。

然而，虽然 BMJ Best Practice 有中文版，但是其翻译质量却较为一般，很多叙述的翻译甚至类似于机器翻译，当然这个问题在不断的改进当中。除此之外，翻译的速度也无法跟上英文原版的更新速度，导致很多条目只有英文版却没有中文版。另外很关键的一点是，BP 中的知识虽然已经分成了“病史和查体”，“诊断步骤”，“检查”等结构化的条目，但是在每个条目内依然是大篇的叙述，这种非结构化的知识无疑增加了自动化处理的难度。

1.2.1.3 CMeKG

CMeKG[8]（Chinese Medical Knowledge Graph）是由北京大学计算语言研究所，郑州大学自然语言处理实验室和鹏程实验室在 2019 年合作推出的中文医学知识图谱。CMeKG 采用自然语言处理技术，从大规模医学文本中挖掘医学知识而构建的中文医学知识图谱。CMeKG 的构建参考了 ICD[9]、ATC[10]、SNOMED[3]、MeSH[11] 等国际权威的医学术语集及知识库。CMeKG 1.0 包括了近 6000 种疾病、2000 种药物、1200 种治疗方案等大量的医学知识。

如图 1–4所示，CMeKG 中的实体包含“病史”、“发病部位”、“检查”和“临床症状及体征”等多种医学常用关系，但是其采用的 schema 仍然是传统知识图谱扁平的 RDF 结构（主体、谓语、客体）。这种对知识的表示方式虽然简单，也利于机器的自动学习，但是其在医学这样的专业领域却有一定的局限性，很关键的一点是这种扁平的 schema 无法表达嵌套的结构。比如“甲状腺功能亢进症”的临

图 1–3 BMJ Best Practice “高钠血症”条目截图

Figure 1–3 Screenshot of entry ”Hypernatremia” in BMJ Best Practice

床表现有“心悸”、“失眠”等，但如果患者是女性，其症状还可能有“月经失调”，所以这条知识需要嵌套的 RDF 结构—（（甲状腺功能亢进症，条件为，女性），导致，月经失调）来表达，而传统扁平的 RDF 结构是很难表达这条知识的。除了表达方式的缺陷，CMeKG 的数据大部分来源于百科数据，其数据质量也难以保证。

当然，除了上述的医学知识库，还有很多通用的知识库中也包含了很多医学的内容。比如从维基百科中撷取出结构化的资料而构建的 DBpedia[12]，国内从事知识图谱工作的研究者共同开发的开放中文知识图谱 OpenKG[13]，复旦大学知识工场实验室研发的中文通用百科知识图谱 CN-DBpedia[14]，罗马大学计算语言学实验室创建的 BabelNet[15] 等等，这些通用的知识库中包含了很多的医学知识，但是其专业性和规模要逊色许多。此外，国内也有一些做从事医疗的企业会构建自己的医学知识库，比如 OMAHA 构建的七巧板医学术语集，其构建方式参考 SNOMED CT，具有很强的专业性。但是因为不开源且收费高昂，本文没有对其做过多的探索。

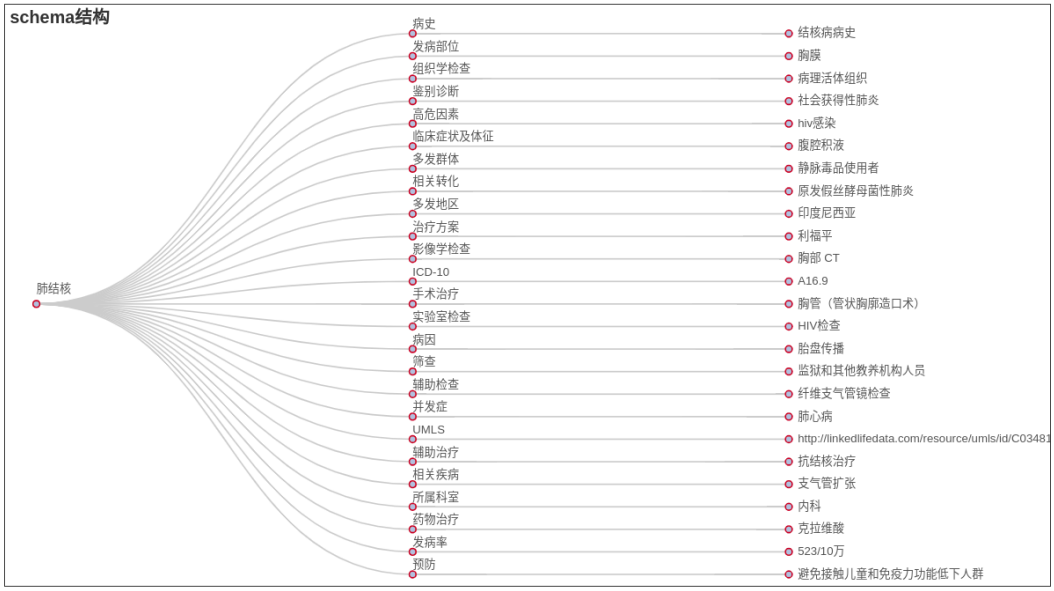


图 1–4 CMeKG schema 示例

Figure 1–4 Example of CMeKG schema

1.2.2 辅助诊断

诊断是医疗中的一个核心环节，当前医疗给出诊断依赖于：1. 患者体征 2. 患者描述 3. 检查数据，其中 1，3 是主要判断依据。而这两项数据完全可以通过机器自动获取，辅助诊断就是自动获取患者体征以及检查数据，再根据其底层的知识

库做自动诊断的过程。辅助诊断如需被医生接受使用，需要满足两个条件：准确以及可解释。而市面上的所有辅助诊断工具都无法满足这两个条件，因此现在所有辅助诊断工具的结果都无法成为患者的确诊结果，而只是作为参考数据提供给医生使用，从而减轻医生的负担。

基于底层数据的类型，可以将辅助诊断模型分为：图像诊断类型[16]，文本诊断类型[17]。图像诊断类型的实现逻辑大体是从医院影像资料库获取医疗影像，然后通过机器学习方法训练出诊断模型，继而可以根据已有模型进行诊断。而文本类型的实现逻辑是从医学文献，病例，医书中获取知识构建医学知识库，然后根据知识库中的知识做出诊断。而知识库中的知识如何表示以及如何根据这些知识做出诊断依赖特定的算法，目前学术界和工业界已经尝试了各种方案，本节会对他们做系统的介绍。

1.2.2.1 基于文本匹配

文本匹配是很多智能问答系统采用的技术手段，通过用户的问题从底层文档库中搜索与之匹配度最高的文档，然后根据文档中的实体进行回答。IBM 发布的 Watson[18] 统就是这样的一个智能问答系统，它曾在电视问答比赛节目击败人类玩家从而引发关注。现在 IBM Watson 已经将重心转移到医疗领域，发布了 IBM Watson Health[19] 主要用于肿瘤的诊断及治疗。它从互联网资源（例如 Wikipedia[20]，DBPedia[12]，WordNet[21] 和 Yago[22]）以及电子病历中收集并结构化医学知识，构建了一个超过 15TB 的庞大知识库。它的诊断策略是首先收集患者的体征和症状信息，然后根据这些患者信息去匹配相似度最高的文档。比如患者的症状是“咳嗽，头晕”，Watson 会使用这两个词条去匹配相似度最高的几篇文档，可能就能匹配到某些医书中的“肺结核”以及“感冒”等章节，然后根据特定的排序策略（比如 TF-IDF[23]，DSSM[24] 等）对结果进行排序筛选，选出相似度最高的文档得出诊断。这种文本匹配方案的优势在于自动化程度高，人类设定好相似度策略以及阈值后就完全可以交给机器自动完成其余的工作，但是其在辅助诊断上的应用缺点也很明显：

- 文本需求量大，从而需要大量计算资源。因为文本匹配只考虑词条在文档中是否出现，而忽视了语义这种包含更多知识量的信息，所以文本匹配算法需要大量的文本来补全这些知识（当然有些缺失掉的知识是无法通过大量文本来补全的）。
- 不适配辅助诊断的流程。辅助诊断需要首先根据患者的主述信息，继而通过与患者的交互获得更多的患者信息，最后根据所有的患者信息进行诊断。

然而文本匹配算法需要获得所有的患者信息去匹配最相似的医学文本，而患者很难一下子说出所有的体征信息，这就对文本匹配方法提出了挑战。

- 文本匹配可能获得错误的医学知识。因为文本匹配只考虑词条在文档中是否出现，而不考虑具体的语义信息，这就可能引来一些错误。比如某本医学文档中记录着“高铁血红蛋白血症虽有明显发绀，但一般无呼吸困难”，如果患者的症状信息为“呼吸困难”，文本匹配算法仍然会匹配到“高铁血红蛋白血症”。但其实文本中“高铁血红蛋白血症”是排斥“呼吸困难”的。本质是因为文本匹配丢失了“一般无”这样的语义信息，所以引入了错误。

1.2.2.2 基于 IF-THEN-ELSE 产生式规则

早期的专家系统[25]都是采用 IF-THEN-ELSE 产生式规则进行推理。比如 1970 年代初期由斯坦福大学开发的辅助诊断工具 Mycin[26]，它主要用于帮助医生对住院的血液感染患者进行诊断和用抗菌素类药物进行治疗。其采用的知识表示方式就是这样的产生式规则，比如“if 化脓菌感染 then 脑膜炎”，“if 脑膜炎 then 脑损伤”，这样就可以根据“化脓菌感染”得出到“脑损伤”的推理路径。这种知识表示的好处在于推理结果的可解释性，因为所有的知识已经表达成了固定的 IF-THEN-ELSE 规则，所以最终的结果一定可以根据这些规则回溯到原因，因此 Mycin 系统中专门有独立的“Explanation System”来为诊断结果做出解释。可以看出，如果这些产生式规则足够多且准确，那么这种方案完全可以胜任医生的角色。但是其弊端就在于这些规则严重依赖专家手动编写，Mycin 刚发布时不过 600 余条规则，导致其只能在很小的领域内使用，很难进行扩展。

1.2.2.3 基于贝叶斯网络

贝叶斯网络[27]（Bayesian network）又称信念网络（belief network），是目前不确定知识性知识表示和推理领域最流行的理论模型之一。一个贝叶斯网络是一个有向无环图 (Directed Acyclic Graph, 简称 DAG)。图中结点代表随机变量，由父结点指向其子结点的有向边代表了结点间的因果联系，并用条件概率来量化关系强度。贝叶斯网络主要用于在已知发生了某些结果，推断出造成结果发生的原因以及发生的概率。这种特性非常适用于医学诊断，在已知患者症状信息（结果）的情况下，推断出其所患疾病（原因）。所以现在很多的辅助诊断系统的底层技术都是基于贝叶斯网络，比如乳腺癌的风险预测系统[28], 糖尿病亚型区分工具[29] 和利用舌头形态进行诊断的中医诊断方法[30] 等。但是绝大多数都是用于专科疾病诊断（比如乳腺癌诊断，肺癌诊断等），因为专科疾病诊断用到的贝叶斯网规模较

小。如果需要分类的疾病增多（达到几百个），贝叶斯网络的结构就会变得非常复杂。有很多边的权重（即疾病与症状之间的条件概率）数据很难获取，在图较小时可以人工赋权，图较大时就很难获得所有的条件概率。其次在网络规模较大时，基于贝叶斯网的推理也会变得很低效。在最坏情况下，贝叶斯网的学习和推理是NP-难（NP-hard）的[31][32]，这就造成了基于贝叶斯网络的辅助诊断工具扩展性不够，只能在疾病规模较小时使用，无法作为全科诊断的工具。

1.2.2.4 基于深度学习

深度学习[33]（deep learning）是近年来最为火热的研究领域，将深度学习技术应用到智能诊断场景非常直观。其具体方案是：首先从大量的电子病历中提取相互关联的疾病和症状，然后将其作为输入训练出一个适配训练集的特定神经网络（CNN[34],RBM[35] 等），最后根据神经网络来对疾病做分类预测[36]。这种方案的优势在于数据量大，几乎可以完全覆盖常见的疾病和症状。但是其问题也很明显，首先是严重依赖电子病历，病历中一般都是常见疾病，对于罕见疾病难以覆盖。其次就是其诊断结果的可解释性，大数据 + 深度学习方案旨在找到数据之间的关联性，而没法获取数据之间的因果联系。比如深度学习方案根据“低热，盗汗，咯血”可以给出正确的诊断结果“肺结核”，但是为什么给出这样的诊断缺乏医学知识的支持，可能会引起患者的不信任感。最后是数据的可靠性问题，我国的电子病历尚处于起步阶段，很多医生还不习惯电子病历，书写时敷也了事，这样就造成电子病历中可能存在错误，如何甄别并修改这些错误是深度学习方案无法解决的。

除了上述的基于文本匹配，产生式规则，贝叶斯网以及深度学习的辅助诊断方案，学术界还提出了一些其他技术来做智能诊断，比如模糊逻辑[37]（Fuzzy logic），概率逻辑[38]（Probabilistic logic）、马尔科夫链逻辑网络[39]（Markov logic network）等，但它们都只停留在理论阶段而没有实际落地的项目。它们的实际落地都都受限于数据的获取，比如模糊逻辑模型难以获取模糊度函数，马尔科夫链逻辑网络难以获取规则的权重等。而本文采用的辅助诊断方案是采用命题逻辑来为诊断过程建模，通过逻辑推理来进行诊断。其主要思想和产生式规则类似，从而保证结果的准确及可解释性。但相比于专家系统，我们采用了自然语言处理[40]（Natural Language Processing）技术自动提取知识，使得我们可以快速的构建大规模的知识库，从而保证了系统的扩展性。

1.3 研究内容

本文针对大数据 + 深度学习方案所面临的问题，提出了一个新的辅助诊断解决方案：通过从医书等可信医学知识源中提取医学知识，然后将其表达成逻辑公式，通过逻辑推理来进行辅助诊断。具体的研究内容如下：

(1) 知识库的构建

为了实现本文的辅助诊断系统，我们首先设计了更符合医学知识表示并且便于自然语言处理的 schema，包括实体的类型以及实体之间的关系类型。然后根据设定好的 schema，从可信知识源中（主要是医书教材）中采用人工标记以及自然语言处理的方法抽取出了大量的医学知识，通过这些医学知识构建了知识库，供后续的推理算法使用。

(2) 辅助诊断内核

基于已经构建好的知识库，本文设计并实现了辅助诊断的内核。其主要思路是首先根据患者主述信息从知识库中提取相关知识并编码成逻辑公式，然后组合这些逻辑公式构建可满足性问题的实例，最后使用 SMT 求解器求解其可满足性来做出诊断。

(3) 知识纠错方案

由于知识源的数据可能存在错误，以及知识抽取的过程中也可能引入错误，所以最终获取的知识可能存在错误。因为本文采用的是基于医学知识进行诊断而非大数据方案，知识的错误会很大程度影响最终诊断的准确率。所以必须对知识库中的知识进行校纠错。本文就从逻辑的角度，提出了纠正知识库中错误的方案。主要基于可满足性问题中的不可满足核 (unsatisfiable core) 技术，完成对知识库中的错误知识（主要是互相矛盾的知识）进行纠正。

(4) 实际应用的辅助诊断系统

本文的所有技术方案不仅仅是停留在理论层面，而是实现出了可用的辅助诊断系统，并且已经在合作医院中落地测试。

1.4 章节安排

本文共有七个章节组成，其中主要内容如下所示：

第一章为绪论部分。绪论中主要阐述了本文的研究背景及意义、国内外的研究现状以及本文主要的研究内容。在最后对本文的章节安排进行了介绍。

第二章是对本文所用到的背景知识以及相关工作的介绍。具体包括知识表示方法，知识存储所用到的数据库以及逻辑推理中的命题逻辑、可满足性问题、SMT

求解器以及不可满足核等知识。

第三章主要阐述了医学知识库的构建流程。我们设计了两版建模医学文本的schema，然后根据 schema 从医书中抽取医学知识，最后将这些医学知识存储到Neo4j 数据库中。

第四章介绍了辅助诊断内核的设计与实现。辅助诊断内核是本文系统的核心，我们详细介绍了其内部各个模并且举例说明了各模块的实现机制和交互流程。

第五章讲解我们如何使用逻辑推理来完成对知识的纠错，这是本文的创新点之一。我们通过求解可满足性问题得出不可满足核完成对错误知识的检测，并对各种错误的模式进行了分析。

第六章是实验部分。在实验章节中，我们会简要介绍整个系统的架构。然后通过实验说明我们知识库的规模，辅助诊断的性能以及修改错误知识对知识库质量的提升。

第七章是总结和展望，对本文的工作进行了总结和未来工作的展望。

第二章　背景知识及相关工作

本文实现的辅助诊断系统使用到了包括知识表示、知识存储，逻辑推理等方面的技术，本章会我们的辅诊系统所依赖的技术做背景以及相关工作的介绍。

2.1　知识表示

根据[41] 的定义，知识表示是将知识表示成符号化、便于计算机自动处理的数据。更直观的来讲，知识的表示就是对知识的一种描述，或者说是对知识的一组约定，一种计算机可以接受的用于描述知识的数据结构，是对知识的符号化、形式化或模型化。它是机器通往智能的基础，使得机器可以像人一样运用知识。经过多年的研究发展，目前比较流行的知识表示方法主要有逻辑表示法[42]、产生式规则表示法[43]、框架表示法[44] 和语义网络表示法[45] 等。本文就是采用语义网络来表示医学知识，并采用 RDF 三元组的形式对其进行具体实现。

2.1.1　语义网络

语义网络[46]（Semantic Network）是一种灵活且表达能力的强的知识表示方式。语义网络是一张带有标识的有向图，它使用用节点来表示对象、概念等实体，用带有标识的边表示实体之间的关系，如图 2–1所示。本文就是使用语义网络来表示医学知识。语义网络的优势在于便于人类理解和展示，但是其也有一些缺点：1) 无法像逻辑或者规则表示法那样直接进行知识推理，所以本文的辅诊系统会把语义网络转换成逻辑公式进行推理。2) 相关概念与关系容易混淆。因为语义网络的中的节点和边可以代表任何的对象、概念、属性和关系。所以在构建语义网络时很容易混淆它们之间的类属关系。比如“熊”可以是“哺乳动物”的一个子类（subclassof），也可以是“哺乳动物”的一个实例（is-a），这样类似的关系很容易混淆。因此本文在构建医学语义网络的时候设计了较为简单的 schema，没有太多的层级结构。当然语义网络只是一个知识表示的概念，其具体的实现方式大多都是采用 RDF 三元组的形式。

2.1.2　资源描述框架

资源描述框架[47](Resource Description Framework，简称 RDF) 一种是用来描述网络资源的框架。RDF 的基本单元是陈述（statement），陈述就是形如 < 主体

(subject)，谓语 (predict)，客体 (object)> 的三元组，每条陈述可以描述一条知识，比如 < 肺结核，导致，咳嗽 > 就可以描述“肺结核导致咳嗽”这条知识。RDF 数据集的本质就是一张有向图（即语义网络）$G = (V, E)$, 其中顶点集 V 代表所有的主体和客体，边集 E 代表所有的谓语，表示主客体之间的关系。此外 RDF 的标准还预定义了一些规则，比如使用 type 这种谓语来表示主客体之间的类属关系。而且嵌套的 RDF 三元组还可以表示较为复杂的知识，比如“肺癌晚期会检查发现肺部阴影”就可以表示成 ((肺癌，条件为，晚期)，检查发现，肺部阴影)，这个三元组的主体不再是一个原子实体，而也是一个 RDF 三元组。如果层层嵌套，那么 RDF 三元组就可以表达非常复杂的知识。现在主流的知识库大多采用 RDF 三元组的方式来实现，比如 WordNet[21]，BabelNet[15]，HowNet[48]。本文的医学知识库其实就是一个 RDF 图，通过预先定义的一些实体和关系类型，然后从知识源中抽取 RDF 三元组集构建知识库。

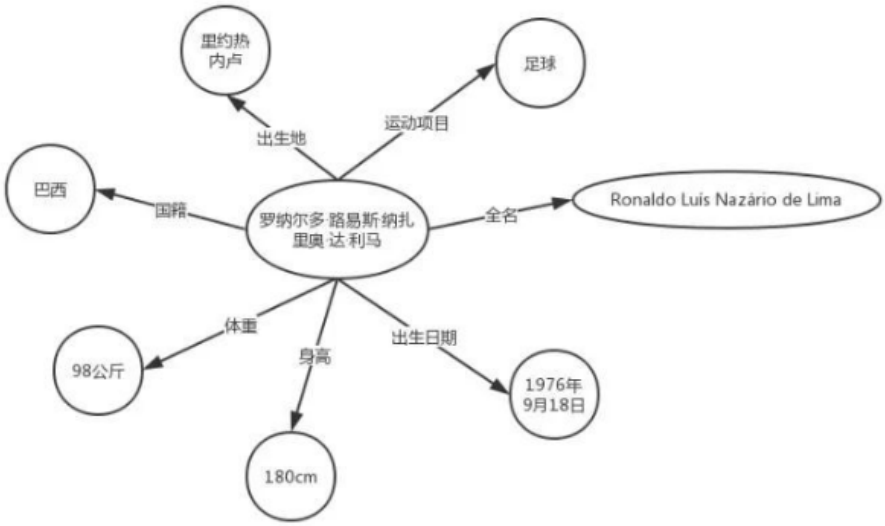


图 2–1 语义网络示例

Figure 2–1 Example of semantic network

2.2 知识存储

知识存储就是如何把抽取出来的知识存储到持久化设备中，本文采用 Neo4j 和 Redis 两种数据库来存储知识。本节会对这两种数据库做详细的介绍。

2.2.1 Neo4j

Neo4j[49] 是一个高性能的,NOSQL 图形数据库。它采用 JAVA 编写，具备完整的数据库事务特性，并且通过使用 Apache Lucence[50] 来支持索引。Neo4j 使用起来非常方便，它的查询语言 Neo4j CQL 是人性化的可读格式，非常容易学习。而且 Neo4j 提供了 REST API，可以被任何编程语言（如 Java，Spring，Scala 等）访问。和传统的 NOSQL 数据库不同的是，它不是基于文档（document）或者键值对（key-value），而是使用网络（从数学角度叫做图）的方式来存储数据。因为是基于图的方式来存储数据，所以 Neo4j 很容易表示连接的数据，也非常容易从一个节点检索、遍历、导航更多的连接数据。因此对于连接的数据操作，Neo4j 的速度要比传统的关系型数据库快近 1000 倍。因为本文的知识采用语义网络的方式来表示，而且经常会在知识网络上做一些查询（比如子图匹配，最短路径等），所以 Neo4j 的图存储方式非常适合存储我们的医学知识。同时，如图 2–2所示，Neo4j 工具栈自带的数据展示工具也非常有助于数据的可视化及调试。

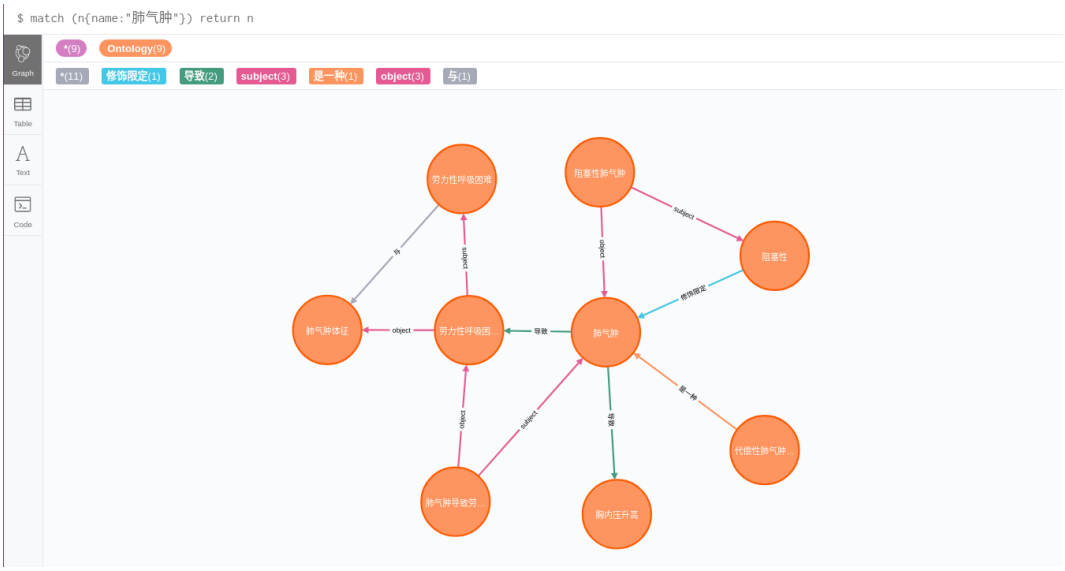


图 2–2 Neo4j 查询界面截图

Figure 2–2 Screenshot of Neo4j query interface

2.2.2 Redis

在实际使用中，本文的系统需要从全体的医学知识网络中提取出一张特定结构的子图。对于本文设计的知识结构，从全图中提取一张特定子图在 Neo4j 中性能一般，所以我们又在 Neo4j 上加了一层 Redis 缓存来提高查询性能。Redis[51]

（Remote Dictionary Server）是一个完全开源免费的，遵守 BSD 协议，高性能的 key-value 数据库。Redis 是完全在内存中保存数据的数据库，使用磁盘只是为了持久化。它会将底层的数据加载到内存进行操作，因此其性能非常出色，每秒可以处理近 11 万次读操作，8 万次写操作，是已知读写速度最快的 key-value 数据库。因为 Redis 基于内存操作的特性，很多应用会把其作为数据缓存来使用。本文的辅助诊断系统就是把底层的 Neo4j 中的数据缓存在 Redis 中供上层调用查询，弥补了 Neo4j 在子图查询性能上的不足。

2.3 逻辑推理

本文为了保证辅助诊断的严谨以及可解释性，采用了逻辑推理的方式来进行诊断。主要思想就是从知识库中提取知识编码成命题逻辑公式，然后将逻辑公式组合变换成可满足性问题的实例，最后通过 SMT 求解器求解可满足性问题实例完成诊断。之所以采用求解可满足性问题的方式进行辅助诊断，很重要的一个原因是可满足性问题中的一个重要概念–不可满足核（unsatisfiable core），它可以完成部分知识纠错的功能，这也是本文的一个重点所在。所以本节会对命题逻辑、可满足性问题、SMT 求解器以及不可满足核做详细的介绍。

2.3.1 命题逻辑

命题逻辑[52]（proposition logic) 是应用一套形式化规则对以符号表示的描述性陈述进行推理的系统。命题逻辑的语言由下面的符号组成：

- 联结词：$\vee, \wedge, \neg, \rightarrow, \leftrightarrow$
- 括号：$), ($
- 变量：$a, b, c, ...$

在命题逻辑中，一个真或假的描述性称述成为原子命题公式，通常用变量来表示。而原子命题公式可以通过联结词来构成复合命题公式。因此命题逻辑公式主要分为两种形式：

- 变量是命题（原子命题公式）。
- 如果 α, β 是命题，则 $(\alpha \wedge \beta)$、$(\alpha \vee \beta)$、$(\alpha \rightarrow \beta)$、$(\alpha \leftrightarrow \beta)$、$(\neg\alpha)$ 都是命题（复合命题公式）。

其中命题公式 $\alpha \rightarrow \beta$ 是经常用于逻辑推理，它表示如果 a 为真，则 b 也为真。比如 a 代表 “肺结核”，b 表示 “咳嗽”，则这条公式表示如果患有肺结核，则一定会有咳嗽症状。

变量的真值指派是一个从变量集映射到集合 $\{False, True\}$ 上的函数 f。比如 $a \to b$, 那我们可以定义一个真值指派使得 $f(a) = True, f(b) = True$。

命题逻辑是一种相对简单的演绎系统, 命题公式表示起来简便也易于理解, 而且相比于谓词逻辑等高阶逻辑，命题逻辑是可判定的。也就是说对于任意的命题逻辑公式，一定存在一个算法可以判定其是否是永真、可满足或者是矛盾的。因此本文采用命题逻辑作为我们的推理系统。

2.3.2 可满足性问题

布尔可满足性问题[53]（Boolean Satisfiability Problem）, 有时简称为可满足性问题或者 SAT 问题, 其是第一个被证明的 NP-完全（complete）问题。给定一个命题逻辑公式，可满足性问题是确定是否存在一组公式中变量的真值指派，使得公式为真。后文常提到的可满足性问题实例其实就是一个命题逻辑公式。举例来说，比如公式 $f_1 = a \wedge b$ 就是可满足的（Staisfiable），因为存在一组变量的真值指派 $\{a = True, b = True\}$ 使得 f_1 为真。而公式 $f_2 = a \wedge \neg a$ 就是不可满足的（Unstaisfiable），因为无论变量 a 如何取值，公式 f_2 都不可能为真。在我们的辅助诊断系统中，就可以使用求解可满足性问题来进行诊断，比如我们考虑两个疾病“膀胱炎”和“急性肾盂肾炎”, 根据人卫教材《内科学》对两种疾病的描述, 它们都会有泌尿系统症状, 比如“尿频、尿急、尿痛”, 但是一般“膀胱炎”不会伴有高烧, 而“急性肾盂肾炎”会伴有“高烧”症状。我们可以获取医学知识“膀胱炎会导致尿频、尿急、尿痛, 但无高烧”,“急性肾盂肾炎会导致尿频、尿急、尿痛、高烧”。如果患者的症状是“尿频、尿痛、高烧”, 我们可以编码成 2–1的命题逻辑公式（实体的编码方式为 {膀胱炎：d_1, 肾盂肾炎：d_2, 尿频：s_1, 尿急：s_2, 尿痛：s_3, 高烧：s_4}）。通过求解 f 的可满足性，我们得出解为 $\{d_1 = False, d_2 = True\}$。根据解我们就可以做出患者可能患有“急性肾盂肾炎”但不患有“膀胱炎”的诊断。

$$f = (d_1 \to s_1 \wedge s_2 \wedge s_3 \wedge \neg s_4)(d_2 \to s_1 \wedge s_2 \wedge s_3 \wedge s_4)(s_1 \wedge s_3 \wedge s_4)(d_1 \vee d_2) \quad (2\text{–}1)$$

2.3.3 SMT 求解器

作为 SAT 问题的扩展,SMT 问题[54]（Satisfiability Modulo Theories Problem）处理的对象是一阶逻辑公式, 相比于命题逻辑, 增加了谓词和量词, 其中谓词可以用特定的理论来解释, 比如数组理论[55], 整数算数理论[56] 等, 很大程度增强了 SMT 公式的表达能力. 用以求解 SMT 问题的自动化工具称为 SMT 求解器（SMT Solver）.SMT 求解技术在有界模型检测、基于符号执行的程序分析、线性规划和调度、测试用例生成以及电路设计和验证等领域有非常广泛的应用. 很多科研机构以及公

司都在致力研发正确率高性能优异的 SMT 求解器, 并且已经成功应用到了具体的领域. 目前流行的 SMT 求解器有: Barcelogic[57], Beaver[58],Yices[59] 以及 Z3[60] 等. 其中, 由微软主导开发的 Z3 SMT 求解器所支持的理论最多, 性能也最好, 因此本文使用了 Z3 SMT 求解器作为我们的求解引擎。

2.3.4 不可满足核

定义 2.1 [61](不可满足核). 给定一个公式 φ, φ_c 是 φ 的一个不可满足核当且仅当 φ_c 不可满足且 $\varphi_c \subseteq \varphi$。

定义 2.2 [61](最小不可满足核). 考虑一个公式 φ 和它的所有不可满足核:$\{\varphi_{c_1}, ..., \varphi_{c_j}\}$。则 φ_{c_k} 是一个最小不可满足核当且仅当 $\forall \varphi_{c_i} \in \{\varphi_{c_1}, ..., \varphi_{c_j}\}, 0 < i \leq j : |\varphi_{c_i}| \geq |\varphi_{c_k}|$。

不可满足核[61] (Unsatisfiable Core), 简称为 unsat core。根据定义 2.1, 一个命题逻辑公式的不可满足核是该公式的任意一个不可满足子集。直观来讲, 不可满足核就是原公式中互相矛盾的子句集, 这对知识纠错很有帮助: 如果我们把每条医学知识都编码成一条命题逻辑公式, 合取这些逻辑公式构建可满足性问题的实例并求其可满足性。如果是不可满足的, 那么求解其不可满足核就可以获得互相矛盾的知识, 而互相矛盾的知识中必定有错误的知识, 只要修改不可满足核中的错误知识就完成了部分的知识纠错。因此不可满足核的功能就是帮助我们快速定位错误知识。当然, 因为一个不可满足的公式可能存在很多个不可满足核, 而且它们之间可能存在包含关系, 所以如果在一个不可满足公式的所有不可满足核上都做纠错, 这个过程就会变得很低效。如果我们能找到最小的不可满足核 (Minimum Unsatisfiable Core), 也就是包含子句数量最少的不可满足核 (严格定义见 2.2), 然后在最小不可满足核上做纠错, 就能提高知识纠错的效率。SMT 求解器就有获取不可满足公式最小不可满足核的功能, 这也是我们选择 SMT 求解器作为我们求解引擎的原因之一。

2.4 本章小结

本章主要介绍了知识表示、知识存储以及逻辑推理的相关背景与相关工作。在知识表示中, 虽然有逻辑、规则、框架和语义网络等多种方案, 但考虑到各自的优劣势, 现在用语义网络来表示知识已经成为了主流。在语义网络的基础上, 我们还介绍了其实现方式之一–RDF, RDF 三元组不仅可以简单方便的表示知识, 而且嵌套的 RDF 也可以表示很复杂的知识, 这就是本文选择使用 RDF 三元组表示

知识的原因。在知识存储中，我们介绍了十分适合储存语义网络这种图结构数据的 Neo4j, 同时为了提高算法性能，我们又用 Redis 对 Neo4j 的数据做了一层缓存。最后，在逻辑推理中，我们首先介绍了我们的理论基础–命题逻辑，然后通过举例说明如何通过求解可满足性问题来进行辅助诊断，其次又对可满足性问题的求解器 SMT 求解器做了介绍，最后我们通过介绍不可满足核来说明如何使用它来进行快速的知识纠错。

第三章　医学知识库的构建

本章主要介绍医学知识库的构建。首先会对知识库的构建流程做总体介绍，接着会详细介绍其中的核心部分–医学文本建模，而对于医学知识抽取以及医学知识存储会简要介绍。

3.1　医学知识库构建流程

医学知识库的构建流程如图 3–1所示，主要分为三个步骤：

(1) 医学文本建模：为了获取医学知识，我们首先必须为医学文本建模，也就是说设计一套符合医学知识结构的知识表达方式，然后根据这种表达方式从医学文本中抽取医学知识。

(2) 医学知识抽取：根据设计好的知识表达方式，从医学文本中抽取知识。我们自己开发了数据标记平台，使用标记平台标记少量文本做训练集训练出 NLP 模型，然后使用 NLP 模型对批量医学文本做预标记，对于预标记的文本我们再通过人工审核的方式来保证知识的可靠性，最终生成知识的 RDF 三元组文件。

(3) 医学知识存储：将 RDF 三元组数据存储到图数据库 Neo4j 中。在存储时，我们会对一些特定的关系做特殊处理，并且通过嵌套实体的方式来存储嵌套的 RDF 三元组。

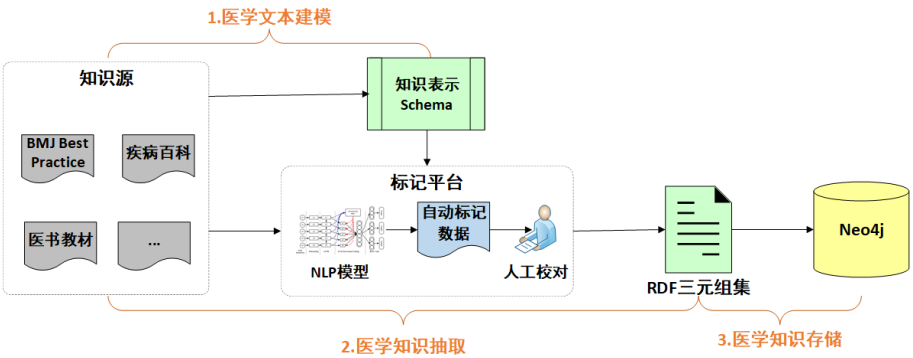


图 3–1 医学知识库构建流程

Figure 3–1 The construction process of Medical knowledge base

3.2 医学文本建模

本节主要介绍医学文本建模的实现方案。首先我们会介绍一下我们的数据源，然后会详细介绍我们设计的医学知识表示的 schema。schema 主要包含实体类型以及实体之间的关系，当然这些实体类型都是医学上的类型（比如病症、检查、病史等），关系也都是医学上一些因果关系（比如导致、检查发现等）。在我们设计知识的 schema 并且实际做知识抽取的过程中发现：医学文本语义复杂，如果要非常准确的给医学文本建模就要设计很复杂的 schema。在我们最初的版本中，我们设计了 30 个实体类型，以及 25 个实体关系。这种复杂的 schema 无论对标记还是 NLP 来说都是很大的负担。我们人工标记的数据有很多错误，NLP 自动标记的数据自然错误更多。而如果设计相对简单的 schema，则很多医学知识难以表达，导致很多医学文本中的知识丢失。在最终的权衡之下，我们设计了相对较为简单，也可以表达大部分问诊层面医学知识的 schema。前后一共经历两个大版本，多个小版本，我们会主要介绍两个大版本的 schema。

3.2.1 医学知识源

我们的医学数据主要有以下几个来源：

- BMJ Best Practice：我们爬取了 BMJ Best Practice 中约 1000 种疾病主题下的诊断步骤的文本数据，然后基于这些文本做医学知识抽取。因为 BMJ Best Practice 中的数据按主题组织，每个主题下有比如诊断、治疗、随访等章节的内容（如图 1–3所示），而只有诊断章节中的诊断步骤子章节包含了较多的症状以及检查信息。而辅助诊断比较关心疾病的症状以及检查指标信息，因此我们主要使用每个疾病主题下诊断步骤章节的文本数据。
- 人卫版医书教材：除了 BMJ Best Practice 中数据以外，我们还预计抽取约 20 本人卫版医书教材中的医学知识。包括《诊断学》、《内科学》、《妇产科学》等等。这个过程一直在进行当中，到目前为止，我们一共抽取完了《诊断学》、《内科学》和《耳鼻咽喉头颈外科学》中的医学知识。
- 疾病百科数据：除了直接从上述较为权威的医学资源中抽取医学知识外，我们另外也爬取一些了医学百科数据，包括百度百科、BabelNet、CMeSH、春雨医生和医脉通等。因为这些数据并不算权威的医学数据源，因此我们没有直接在这些数据上做知识抽取，而是准备根据这些数据构建医学同义词库。同义词库在我们系统中的医学实体对齐[62]，患者意图理解等功能上很有帮助。我们目前的同义词库规模较小，因此尚未实际使用。

3.2.2 知识表示 schema v1.0

3.2.2.1 实体类型

知识表示 schema v1.0 中的实体类型如表 3–1所示。为了较为准确的表达医学实体，我们设计了 23 种实体类型。其中有一些实体类型可能比较令人困惑，我们对其做具体说明：

- 六种修饰语：分别修饰医学实体的状态、性质、程度、频率、方位、数量。我们区分这六种修饰语的目的是为了提高问诊的友好程度以及智能程度。比如我们想要询问患者是否有“持续咳嗽”，我们可以首先询问患者是否有“咳嗽”症状，其次根据“持续”是“频率修饰语”的类型来询问患者咳嗽的频率是多少。这样的询问方式可以很好的优化用户体验。
- 病症：我们在知识表示层不区分疾病和症状，因为这两个概念的界限很模糊，标记人员很容易标错。而在标记平台获得病症实体后，我们将病症实体与 ICD-10[63] 疾病编码匹配来区分疾病和症状。
- 代词：医学文本中有非常多的省略以及指代。比如在《耳鼻咽喉头颈外科》中的“耳气压伤”章节，其只有标题中提到了“耳气压伤”，在之后所有的文本中都是用“本病”指代。比如“本病症状有耳闷、耳鸣等”，所以我们在标记时用“本病”指代“耳气压伤”，然后“本病”会导致“耳闷、耳鸣”。如果直接标记标题中的“耳气压伤”导致正文中的“耳闷、耳鸣”，这种文档级别的关系抽取[64] 对 NLP 自动学习会是很大的负担。
- 否定词：表示对实体的否定。有一些医书中明确说明某种疾病不会有某种症状。比如“膀胱炎不会伴有高烧”，其中“不”就是否定词来修饰“高烧”。

3.2.2.2 关系类型

知识表示 schema v1.0 中的实体关系类型如表 3–2所示。在最初的版本中，我们设计了共 19 种实体间的关系类型，这里节选出其中的一些关系做具体说明：

- 两种导致关系：强导致、弱导致。强导致表示实体之间很强的因果关系，比如如果患有疾病“肺结核”大概率就有“咳嗽”症状，那么就可以表示为（肺结核，强导致，咳嗽）。强导致可以作为推理的依据，具体来说，如果患者患有“肺结核”，我们就断定他有“咳嗽”症状。而如果患者没有“咳嗽”症状，我们就认为其不患有“肺结核”。而弱导致表示较弱的因果关系，比如“吃辣可诱发胃溃疡”，其实“吃辣”不一定就会导致“胃溃疡”，所以他们之间没有很强的因果关系。弱导致无法作为推理的依据，我们无法根据患者“吃辣”推理出“胃溃疡”，只能将“吃辣”作为“胃溃疡”的加分项。

表 3–1 实体类型–schema v1.0

Table 3–1 Entity type–schema v1.0

实体名称	含义	例子
状态修饰语	表示医学实体状态的修饰语	异常（红细胞膜），正常 (体重)
性质修饰语	表示医学实体性质的修饰语	功能性（梗阻），溶血性（黄疸）
程度修饰语	表示医学实体程度的修饰语	中度（肥胖），深度（昏迷）
频率修饰语	表示医学实体频率的修饰语	持续（咳嗽），反复（皮疹）
方位修饰语	表示医学实体位置的修饰语	后（腰椎），下（呼吸道）
数量修饰语	表示医学实体数量的修饰语	大量（脓血），较少（骨关节破坏）
病症	疾病和症状	咳嗽，肺结核
代词	指代其他医学实体的词	者，该病
部位	解剖学部位	腰椎，喉咙
否定词	表示对被修饰实体的否定	否，非，没有
检查	包括体检和化验等所有理化检查	结肠镜检查，内镜检查
病史	包括家族史、既往史、现病史	吸烟史，饮酒史
事件	不可归结为理化因素的外部因素	摔倒，手术
颜色	颜色	红色，绿色
生物	生物	毒蕈
时间	时间	数分钟，数小时
化学物质	表示具体的某一种或某一类化学物质	乙醇，皮质醇
食品	食品	酒，浓茶
年龄	年龄的数值	35 岁
人群分类	区分人群的实体	青壮年，老年人
数据	具体的数据数值（带单位）	80%，10μmol/dl
生理概念	描述生理构造，生理过程	红细胞，骨髓
病理概念	描述病理构造，病理过程	坏死组织，病变瓣膜

- 两种并列关系：与、或。与表示实体之间的合取关系，它们必须都要满足，缺一不可。比如“急性溶血时可有发热、寒战“就表示患有“急性溶血”时会同时有“发热”和“寒战”两个症状，所以可以表示为（急性溶血，强导致，（发热，与，寒战））。而或表示实体之间的析取关系，表示实体之间无须都满足，只要满足其一即可。比如“干咳或刺激性咳嗽常见于急性咽喉炎”表示如果患有“急性咽喉炎”，则会有“干咳”或者“刺激性咳嗽”，可能两者都有，也可能只有两者其一，所以就可以用“或”来表示它们之间的关系。即（急性咽喉炎，强导致，（干咳，或，刺激性咳嗽））。

- 条件为关系：表示某种关系发生的条件，常作为嵌套关系使用。比如“痛风性肾病晚期会出现高血压、水肿等症状”，其中“痛风性肾病”可能有“早期，中期，晚期”等多个阶段，每个阶段的症状都不一样，所以要用“条件为“关系连接不同的阶段，这句话可以用（(痛风性肾病，条件为，晚期)，强导致，(高血压，与，水肿)）这样的嵌套关系来表达。
- 检查发现关系：表示某项检查发现的结果。因为检查的结果都是比如“阳性，阴性”这样通用的用词，其实这里的“阳性，阴性”都是做了省略，其真正的含义是某项检查阳性，某项检查阴性。所以我们对检查发现关系做了规定，如果疾病 X 要执行检查 A,A 的检查结果是 B。则必须标记为 (X, 强导致,(A, 检查发现,B))，我们在后台会将检查结果 B 加上其检查项 A 作为前缀合并成 A 检查发现 B。整个关系可以处理为 (X, 强导致，A 检查发现 B)。
- 不导致：表示对因果关系的否定。如“多数哮喘患者并不会出现发热症状”可以标记为：(哮喘, 不导致, 发热)，其和（哮喘，强导致，(不，修饰限定，发热)）等价。增加不导致关系是为了便于标记人员理解。
- 然后：表示医学实体发生的先后顺序。医学生有一些症状确实是有先后发生顺序的，比如“偏头痛的头痛症状在呕吐后减轻”，这句话可以标记为（偏头痛，弱导致，(呕吐，然后，头痛减轻)）。

我们设计的关系 schema 中最大的特点是嵌套关系的存在，比如“条件为”，“并列”等关系基本都会在嵌套关系中出现。嵌套关系可以较为准确的表达医学知识，比如“痛风性肾病”可能有“早期，中期，晚期”等多个阶段，每个阶段的症状都不一样，这些知识用“条件为”这样的嵌套关系就可以很好的表达出来。当然，嵌套关系的存在也增加了人工标记和 NLP 自动处理的难度。

表 3–2 关系类型–schema v1.0

Table 3–2 Relation type–schema v1.0

关系名称	含义	例子
强导致	表示很强的因果关系	如“凡能引发溶血的疾病都可引发溶血性黄疸”这句话可以表示为：(溶血, 强导致, 溶血性黄疸)
弱导致	表示较弱的因果关系	如“吃辣可诱发胃溃疡”这句话可以提取：(吃辣, 弱导致, 胃溃疡)

续下页

续表 3–2

关系名称	含义	例子
修饰限定	表示某词语对另一个词语的修饰和约束	如“粪胆原随之增加”可以提取：(增加, 修饰限定, 粪胆原)；如“后天性获得性溶血性贫血”中，可以通过“修饰限定”关系标记为 (后天性, 修饰限定,(获得性, 修饰限定,(溶血性, 修饰限定, 贫血)))
与	表示实体之间的与关系	如“急性溶血时可有发热、寒战”可以标记为（急性溶血，强导致，（发热，与，寒战））
或	表示实体之间的或关系	如“干咳或刺激性咳嗽常见于急性咽喉炎”可以标记为（（急性，修饰限定，咽喉炎），强导致，（干咳，或，（刺激性，修饰限定，咳嗽））
条件为	表示某种关系的发生条件，因此常嵌套使用	如“血红蛋白在组织蛋白酶的作用下形成血红素和珠蛋白”可以标记为：((血红蛋白, 条件为, 组织蛋白酶), 转变为,(血红素, 与, 珠蛋白))
调查病史	表示确诊某种病症需要询问某病史	如“肥胖多余家族肥胖史相关”可以标记为：(肥胖，调查病史，家族肥胖史)
执行检查	表示确诊某种病症需要进行的检查，包括查体和化验	如“腹泻患者请执行粪便检查”可以标记为：(腹泻，执行检查，粪便检查)
检查发现	表示某种检查得出特定的结果，如果疾病 X 导致检查 A 的结果是 B，则应该标记为:(X, 导致,(A, 检查发现,B))	如“隐血试验阳性”应标记为：(隐血试验, 检查发现, 阳性)；“血液检查除贫血外尚有网织红细胞增加”应标记为：(血液检查, 检查发现,(贫血, 与,(增加, 修饰限定, 网织红细胞)))
是一种	表示概念的从属关系	如“常见病因有先天性溶血性贫血，如海洋性贫血”应标记为 ((海洋性, 限定修饰, 贫血), 是一种,(先天性, 限定修饰,(溶血性, 修饰限定, 贫血)))
等价	表示数量上的相等，或概念上的相同	如“水肿（edema）”可以标记为（水肿，等价，edema）
不导致	表示对因果关系的否定	如“多数哮喘患者并不会出现发热症状”可以标记为：(哮喘, 不导致, 发热)
转变为	表示物质之间的转化	如“血红蛋白在组织蛋白酶的作用下形成血红素和珠蛋白”可以标记为：((血红蛋白, 条件为, 组织蛋白酶), 转变为,(血红素, 与, 珠蛋白))
定义为	表示显示或隐式的定义说明	如“有不同程度的贫血和血红蛋白尿（尿呈酱油色或茶色）”可以标记为 (血红蛋白尿, 定义为,((酱油色, 与, 茶色), 修饰限定, 尿))

续下页

续表 3–2

关系名称	含义	例子
指代	表示代词指代的具体实体	如（本病，指代，耳气压伤）
作用于	表示某实体对另一个实体施加作用	如“肝脏处理尿胆原的能力降低”可以标记为：(降低, 修饰限定,(肝脏, 作用于, 尿胆原))
表现为	表示某个实体的具体特点。	如“腹泻表现为每天排便次数增多”可以标记为（腹泻，表现为，排便次数增多）
比例为	表示某实体占另一实体的比例	如“循环血容量 30% 以上...”可以标记为（循环血容量，比例为，30% 以上）
然后	表示时间先后顺序	如“偏头痛的头痛症状在呕吐后减轻”可以标记为（偏头痛，弱导致，（呕吐，然后，头痛减轻））

3.2.2.3 关系约束

在我们用设计好的 schema 做实际的知识抽取过程中发现，三元组（实体 A，关系 R，实体 B）中关系 R 其实对实体 A 和实体 B 的类型有约束。比如（功能性，强导致，梗阻）这种类型的三元组就不是合法的三元组，因为“功能性”属于性质修饰语，而修饰语在语法上都是作为名词的定语出现，而强导致关系的主体必须是名词，所以“功能性”就不能作为强导致关系的主体。又比如（肺结核，检查发现，肺部阴影）也是非法三元组，因为检查发现要求主体的类型必须是检查，而“肺结核”的类型是病症，不能成为检查发现关系的主体。所以对于三元组（实体 A，关系 R，实体 B），我们制作了关系 R 对 A 和 B 类型的约束表，主要分为两种：

- 白名单：规定实体 A，B 合法的类型范围。详细数据见表 3–3。
- 黑名单：规定实体 A，B 非法的类型范围。详细数据见表 3–4。

3.2.3 知识表示 schema v2.0

在我们使用知识表示 schema v2.0 抽取知识的过程中，我们发现标记以及 NLP 抽取的知识错误率很高。究其原因，还是因为 schema 过于复杂，23 种实体类型以及 19 种关系类型对于标记人员已经是很大的负担，对于 NLP 来说更是难上加难。考虑到我们是辅诊系统，主要涉及到的是问诊层面的知识，这个层次的知识从本质上来讲只需要关注疾病和症状之间的相关关系即可，配合上一些基本的患者信息，如性别、年龄、病史等。而对于复杂的检查指标（比如“影像检查发现病灶 >2cm,AFP $>$ 400ng/ml “），深层的病理发生机制（比如“当失水量达体重的 2% ~ 3% 时，渴感中枢兴奋，刺激抗利尿激素释放，水重吸收增加，尿量减少，尿比重

表 3–3 关系约束白名单–schema v1.0

Table 3–3 Whitelist of relationship constraints–schema v1.0

关系类型	主体类型	客体类型
条件为	代词，病史，生理概念，病理概念	病症，代词，病史，事件，时间，数据
调查病史	病症，代词	病史，代词
执行检查	病症，代词	检查，代词
检查发现	检查，代词	病症，代词，化学物质，生理概念，病理概念
指代	代词	*
比例为	病症，代词，病史，生物，化学物质，食品，人群分类，生理概念，病理概念	数据
转变为	化学物质，生物，生理概念，病理概念，代词	化学物质，生物，生理概念，病理概念，代词
作用于	部位，生物，化学物质，食品，生理概念，病理概念，代词	部位，生物，化学物质，食品，生理概念，病理概念，代词
然后	病症，代词	病症，代词

* 为通配符，表示所有实体类型.

增高”）其实无需关注，所以我们简化了我们知识表示的 schema，虽然有一些复杂的医学知识无法表达，但是优化了知识结构，减少了知识抽取的难度。

3.2.3.1 实体类型

知识表示 schema v2.0 中的实体类型如表 3–5所示, 我们主要做了以下修改：

- 合并六种修饰语。在实际标记过程中，标记人员很难区分这六种修饰语，而且修饰语的类别除了优化用户体验之外没有太大的作用。所以我们将其直接合并成一种“修饰语”类型，不再区分。
- 删除病理概念和检查实体类型。因为问诊层不会涉及到太多深层的病理机制和复杂的检查指标，所以我们删除了“病理概念”和“检查”类型。
- 删除“人群分类”实体类型，因为在我们知识抽取的过程中发现，“人群分类”实体出现的非常少，而且很多可以用性别或者年龄类型替代，因此删除该实体类型。
- 增加“阶段”实体类型。医学文本中有很多疾病的分型以及阶段区分，之间一直没有很好的类型来建模这种实体（用“时间”来表示不太妥当），所以我们增加了“阶段”实体类型。

表 3–4 关系约束黑名单–schema v1.0

Table 3–4 Blacklist of relationship constraints–schema v1.0

关系类型	主体类型	客体类型
强导致	性质修饰语，状态修饰语，程度修饰语，频率修饰语，方位修饰语，数量修饰语，部位，否定词，颜色，时间，数据	*
弱导致	性质修饰语，状态修饰语，程度修饰语，频率修饰语，方位修饰语，数量修饰语，部位，否定词，颜色，时间，数据	*

* 为通配符，表示所有实体类型.

3.2.3.2 关系类型

知识表示 schema v2.0 中的实体类型如表 3–6所示，我们主要做了以下修改：

- 去除“执行检查”以及“检查发现”关系，因为我们决定只关注问诊层的知识，所以对复杂的检查指标知识不在抽取。当然，由于我们采用 schema v1.0 抽取过很多医学知识，所以我们现在的医学知识库中依然存在不少“执行检查”和“检查发现”关系。
- 删除“作用于”，“转变为”，“定义为”，“比例为”，这四种关系在抽取时数量较少，而且容易混淆，因此删除。
- 删除“表现为”，因为“A 表现为 B”就是指 A 会引起 B, 因此用“强导致“替换。
- 将“修饰限定”修改为“拼接”关系。修饰限定本质上就是将修饰语和主语拼接，所以修改为“拼接关系”。而且像“后天性获得性溶血性贫血”这种实体之间会把他们标记成（后天性，修饰限定，（获得性，修饰限定，溶血）），现在不在强调修饰关系，直接标记为一个“病症”实体。
- 增加“否定修饰”，因为去除了修饰限定关系，“无发热”在 schema v1.0 中使用（无 [否定词]，修饰限定，发热）在 schema v2.0 中用（无 [否定词]，否定修饰，发热）来标记。
- 将“等价”修改为“同义词”关系，便于理解。
- 增加“并发症”关系。医书中有很多关于并发症的描述，所以我们增加这个关系。

其实我们不在强调修饰语，且只关心问诊层的知识之后，虽然会遗失部分知

表 3–5 实体类型–schema v2.0

Table 3–5 Entity type–schema v2.0

实体名称	含义	例子
修饰语	六种修饰语合并，不再区分	先天性，升高
病症	疾病和症状。这里症状只包含患者可以主观感受的那些，其他的不算在内。	咳嗽，肺结核
代词	指代其他医学实体的词	者，该病
部位	解剖学部位	腰椎，喉咙
否定词	表示对被修饰实体的否定	否，非，没有
病史	包括家族史、既往史、现病史	吸烟史，饮酒史
事件	不可归结为理化因素的外部因素	摔倒，手术，高温环境，月经前
颜色	颜色	红色，绿色
生物	生物	毒蕈
时间	时间	数分钟，数小时
化学物质	表示具体的某一种或某一类化学物质	乙醇，皮质醇
食品	食品	酒，浓茶
性别	男女性别	男性，女性
年龄	年龄的数值	35 岁
生理概念	仅用于拼接实体的时。当短实体类型不能用以上类型表达的时候使用	尿，血压
阶段	用以表示病症的阶段或分型	早期，一型

识，但这些知识对于辅诊来说是无关紧要的，但是知识抽取的复杂度降低了很多。减轻了标记人员的负担，也提高了 NLP 的准确率。

3.2.3.3 关系约束

schema v2.0 中关系约束的设计思路和 schema v1.0 没有太大区别，参考表 3–3和表 3–4，这里不再赘述。

3.3 医学知识抽取

如图 3–1所示，我们自己开发了标记平台来进行知识的标记以及自动抽取。具体来说，就是先使用标记平台标记少量文本做训练集训练出 NLP 模型，然后使用 NLP 模型对批量文本做预标记。在我们的实践过程中，NLP 中命名实体识别 (Named Entity Recognition) 准确率较高，在分词，实体类型判别上都有超过 95%

表 3–6 关系类型–schema v2.0

Table 3–6 Relation type–schema v2.0

关系名称	含义	例子
强导致	同 schema v1.0	同 schema v1.0
弱导致	同 schema v1.0	同 schema v1.0
不导致	同 schema v1.0	同 schema v1.0
拼接	表示组成实体的词素的拼接，在词汇不能直接标记为实体的时候使用。原则上能不用这个关系标记实体的时候就不用它。词素将按照“主 + 宾”的顺序拼接成一个实体。拼接以后实体的类型遵从最右侧优先级规则，取主语和宾语实体中优先级较高的类型。	如“粪胆原随之增加”可以提取：(粪胆原: 生理概念, 拼接, 增加: 修饰语)，则等价于标记实体“粪胆原增加”，合并实体类型为“病症”；又如“肠内的尿胆原增加”可以表示为：((肠: 部位, 拼接, 尿胆原: 生理概念), 拼接, 增加: 修饰语)，等价于标记实体“肠尿胆原增加”，合并实体类型为“病症”。如“后天性获得性溶血性贫血”这种直接连在一起的实体，则不需要拼接关系，直接标记成一整个实体即可。
与	同 schema v1.0	同 schema v1.0
或	同 schema v1.0	同 schema v1.0
条件为	同 schema v1.0	同 schema v1.0
调查病史	同 schema v1.0	同 schema v1.0
是一种	同 schema v1.0	同 schema v1.0
同义词	表示实体之间是同义词, 替换 schema v1.0 中的等价关系	如“水肿（edema）”可以标记为（水肿，同义词，edema）
指代	同 schema v1.0	同 schema v1.0
然后	同 schema v1.0	同 schema v1.0
并发症	表示病症的并发症	如“肠结核的并发症有肠出血”可以标记为 (肠结核，并发症，肠出血)
否定修饰	等价于否定词修饰限定	如“无发热”可以标记为（无，否定修饰，发热）

的准确率，但关系抽取（Relation Extraction）准确率就相对较低，有很多的嵌套关系漏标或者错标，所以 NLP 模型在当前阶段还无法完全代替人直接完成数据标记。所以我们采用通过 NLP 模型预标记数据，然后人工审核（增、删、改）预标记数据的方案来完成医学文本中知识的抽取。因为本文的核心在于知识表示与推理，所以偏重于 NLP 技术的知识抽取不是本文的重点，这里不再详细介绍。

3.4 医学知识存储

因为 RDF 数据集可以看成一个很大的有向图，即语义网络，所以我们直接使用图的方式来存储 RDF 三元组数据。具体来说，RDF 三元组形如 < 主体 (subject)，谓语 (predict)，客体 (object)>，我们把主体和客体分别当成两个节点，谓语当成主体指向客体的有向边存储到 Neo4j 中。具体形式如图 2–2所示。但是在我们的实际使用过程中，我们需要从全体的医学知识网络中提取出一张特定的子图（比如以肺结核为中心的相关知识子图，详见4.2.1节）。而从全图中提取一张特定子图在 Neo4j 中性能一般，又因为我们现在的需求中这张子图不会动态更新，所以我们又在 Neo4j 上加了一层 Redis，将这些特定的子图全部缓存在 Redis 中来提高查询性能。这一块也不是本文的重点，不在详细介绍。

在实际存储时，“拼接”和“指代”关系我们会做特殊处理：1) 会将“拼接”关系的主客体合并成一个实体存储，比如”（典型性，拼接，肺炎）”合并成一个实体“典型性肺炎”存储，因此知识库中不会存在“拼接”关系。2) 会将“指代关系”的主体替换成客体存储，比如“（本病, 指代，肺炎），（本病，强导致，咳嗽）”转化成“（肺炎，强导致，咳嗽）”存储，因此知识库中不会存在“指代”关系。此外，我们在知识存储时引入了嵌套实体的概念。当我们需要存储嵌套关系的时候，比如“（肺癌，导致，（呼吸困难，或，咯血））”，其知识的表示方式如图 3–2a所示，但是 Neo4j 中边只能由实体连接到实体，无法从实体连到边。所以我们额外增加了一个嵌套实体“咯血或呼吸困难”，然后从“肺癌“连接到这个嵌套实体上，如图 3–2b所示。

a) 原知识表示

a) Origin knowledge representation

b) Neo4j 的存储方式

b) Storage method of Neo4j

图 3–2 嵌套实体在 Neo4j 中的存储示例

Figure 3–2 Storage example of nested entities in Neo4j

3.5 本章小结

本章主要介绍了我们医学知识库的构建流程，具体包括医学文本建模，医学知识抽取以及医学知识存储三部分。而医学文本建模是本文的重点，所以我们花了较大篇幅来介绍。医学文本建模的核心就是设计符合医学知识结构的知识表示schema，这是一个 trade-off 的过程：schema 简单，很多医学知识无法表达。schema 复杂，知识抽取的准确率低。我们前后经历了两个大版本的迭代，最终设计了表 3.2.3中的 schema v2.0。schema v2.0 中包括 16 种实体类型，14 种实体关系，而且只关注问诊层面的医学知识，对于复杂的检查指标和病理机制不在关注，这极大程度降低了知识抽取的复杂度。我们的知识 schema 中最核心的点就是使用嵌套关系来表达复杂的医学知识。最后我们还介绍了关系约束的白名单和黑名单，对特定关系的主客体的类型做了约束，来降低知识抽取的错误率。

第四章　辅助诊断内核

本章主要介绍辅助诊断内核的设计与实现。首先会总体介绍诊断内核的架构设计，接着会分别介绍其内部各模块的功能以及实现方式。

4.1　架构设计

辅助诊断内核的主要工作就是利用知识库中的医学知识结合患者的症状信息完成对患者所患疾病的预测。其内部具体的架构设计如图 4–1所示，主要分为四个子模块：

(1) 知识库转换器 (KBConverter): 根据患者的主述信息从医学知识库中抽取相关的医学知识，并将这些医学知识编码成逻辑公式交给推理引擎使用。

(2) 推理引擎 (ReasoningSolver): 将知识库转换器输出的命题逻辑公式组合变换成可满足性问题的实例，并使用 SMT 求解器进行求解。求解过程中会需要更多逻辑变量的赋值信息，推理引擎会将这些变量抛出交给问题生成器处理，问题生成器处理完后会将这些变量赋完值交给推理引擎继续处理。推理引擎会根据医学知识的逻辑规则结合患者的症状信息排除掉很多无关的疾病，最后输出一个候选疾病集。

(3) 问题生成器 (QuestionGenerator)：因为推理引擎处理的对象都是逻辑变量及公式，当推理引擎需要确定某些变量的赋值时，会将这些变量抛给问题生成器处理。问题生成器会根据这些变量解码生成具体的问题（单选，多选等）与患者交互，获得患者的答案后问题生成器会将这些问题答案再编码成赋完值的变量交给推理引擎迭代处理。

(4) 诊断计算器 (DiagnosisCalculator)：因为推理引擎只会排除无关的疾病，对于剩余的疾病都会当成候选疾病。一是这些候选疾病可能数量很多；二是这些疾病没有评分的先后关系。诊断计算器会根据患者的症状信息给这些疾病评分，并最终将得分最高的 5（5 只是阈值，可以调整）个疾病返回。

4.2　知识库转换器

本节会介绍知识库转换器功能及实现方式，其具体的功能主要有两个：1) 根据患者的主述信息从知识库中提取相关的医学知识 2) 将这些医学知识编码成命题逻辑公式。

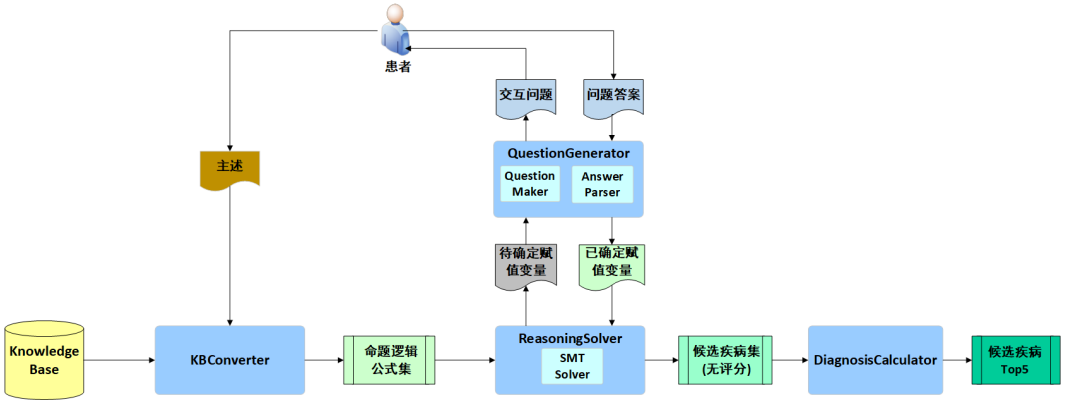


图 4–1 诊断内核架构设计

Figure 4–1 The module architecture design of aided diagnose core

4.2.1 提取相关医学知识

正如前文 2.1.1节提到，我们构建的医学知识库可以看做是一张语义网络图，具体来说，就是一张 RDF 图 G。根据主述信息从医学知识库中提取相关知识就是从 G 中提取出一张子图。提取相关知识子图接收两个参数：一个是患者的主述症状集合 $S_{maininfo}$，另一个是从主述节点延伸出的步数 hop。而根据主述症状集获取相关知识子图为所有主述症状相关知识子图的并集，即

$$\begin{aligned} G_{maininfo,hop} &= \bigcup_{s \in S_{maininfo}} G_{s,hop} \\ &= \bigcup_{s \in S_{maininfo}} DFS(G, s, hop, \Phi) \end{aligned} \tag{4–1}$$

而单一症状 s 的相关知识子图 $G_{s,hop}$ 可以通过深度优先搜索图 G 来构建，其主要思路就是以主述症状 s 为根节点在图 G 上进行深度优先搜索不停的扩展图 $G_{s,hop}$，搜索过程中 hop 的数量逐渐递减（不同类型的边 hop 的递减数量不一样），直到 $hop \leq 0$ 为止。深度优先搜索算法 DFS 如算法4–1所示，其中包含 s 的顶层 Clause 嵌套实体节点表示包含节点 s 且只有“与”、“或”关系的最大嵌套实体。如图4–2所示，节点 a, b, c, t_1, t_2 的顶层 Clause 嵌套实体节点都是 t_2，节点 d 的顶层 Clause 嵌套实体节点是它自己。

图 4–3[1]就是我们从知识库中提取的以‘咳嗽，恶心’为主述，hop=3[2]的子图

[1]实际存储中，图中起点或终点在边上的关系都应该使用如图 3–2示例的嵌套实体的存储方式，本图为了展示方便没有实际画出嵌套实体。

[2]实际应用中，我们一般使用 hop=4 或 hop=5 的子图，其会包含数千个节点及关系，这里为了方便展示取 hop=3。

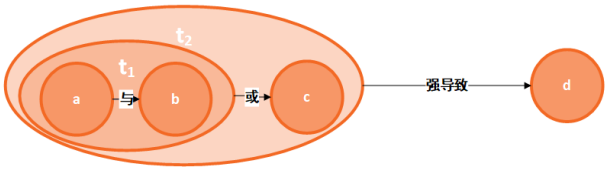


图 4–2 顶层子句嵌套实体示例

Figure 4–2 The example of top clause recursive entity

$G_{\{cough,nausea\},3}$。

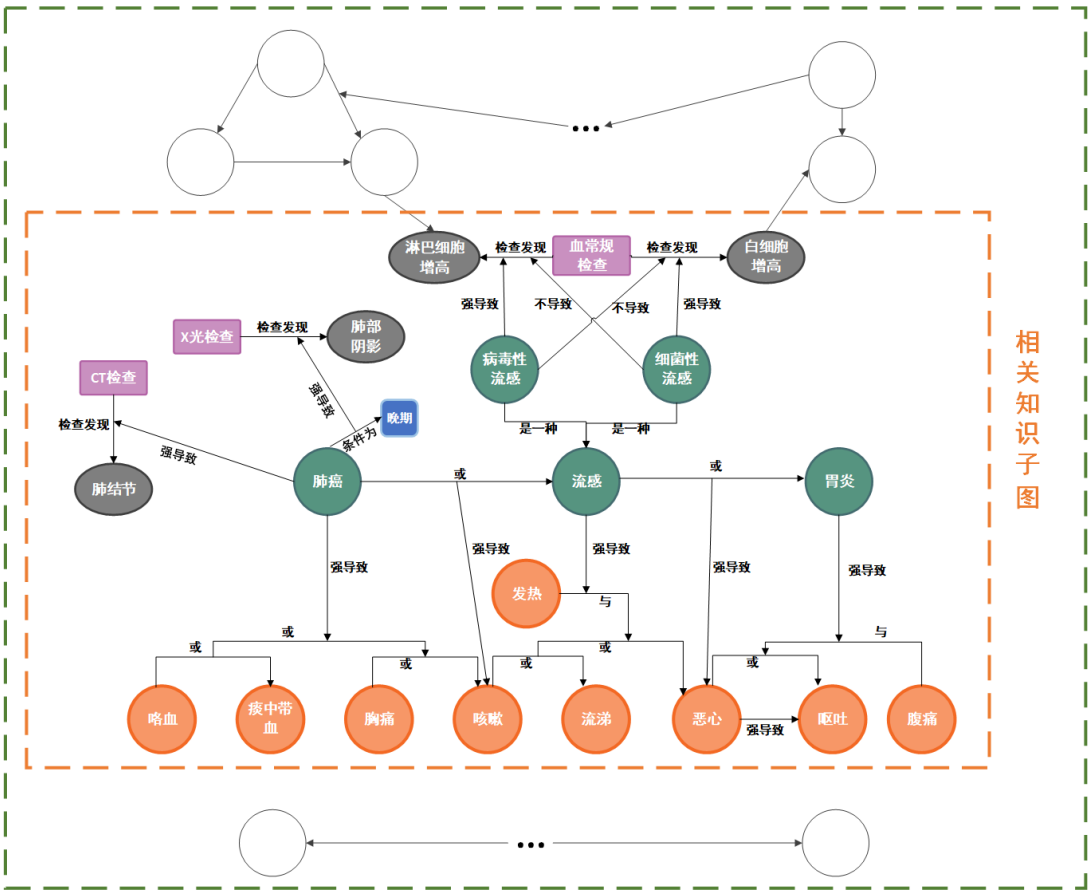


图 4–3 相关医学知识子图示例

Figure 4–3 Example of the related medical knowledge graph

4.2.2 命题逻辑公式编码

提取出子图后，我们要把子图编码成逻辑公式集合，也就是将子图中的 RDF 三元组转换成命题逻辑公式。编码过程分为两步:

(1) 将 RDF 三元组中主客实体编码成逻辑变量。这一步比较简单，就是将不同名称的实体映射成不同的逻辑变量。比如“肺炎, 流感, 胃炎”分别映射成“d_1, d_2, d_3”。

(2) 将 RDF 三元组中谓语编码成命题逻辑联结词。具体编码方式见表 4–2，我们节选一些做具体说明:

(a) ”然后” 关系。“然后”强调的是实体发生的先后顺序，因为我们采用朴素的命题逻辑编码 RDF 三元组，因此无法表示顺序关系。我们采用将”（a, 然后,b）” 合并成一个实体”a 然后 b” 实体的方式，然后直接询问患者是否有 “a 然后 b” 症状，让患者判断先后关系。这样的交互方式显然不太友好，之后我们会考虑加入时序逻辑[65] 解决这个问题。

(b) 非强因果关系。我们认为 “弱导致”，“调查病史”，“并发症” 都不是强因果关系，不会对其编码。

(c) ”不导致” 和 “否定修饰” 关系。这两个关系其实本质一样，比如 “(膀胱炎，不导致，高烧)” 等价于 “(膀胱炎，强导致，(不，否定修饰高烧))”。我们只是为了标记人员理解同时使用了两种关系。

所以，图 4–3所示的相关知识子图 $G_{\{cough,nausea\},3}$ 可以采用如下方式编码成命题逻辑规则集 R。为了处理方便，我们会把所有规则统一转换拆分成 $(v_{l_1} \wedge v_{l_2} \wedge ... \wedge v_{l_m}) \rightarrow (v_{r_1} \vee v_{r_2} \vee ... \vee v_{r_n})$ 的形式，比如”$d_1 \vee d_2 \rightarrow s_4$” 可以拆分成两条规则”$d_1 \rightarrow s_4, d_2 \rightarrow s_4$”（转换拆分算法比较简单，这里不详细介绍）:

(1) 用逻辑变量编码实体，如表 4–1所示

(2) 用逻辑联结词编码关系。基于表 4–2的编码方式，$G_{\{cough,nausea\},3}$，即图4–3可以编码成公式集 4–2。

$$\begin{aligned} R = \{&d_1 \rightarrow s_1 \vee s_2 \vee s_3 \vee s_4,\ d_2 \rightarrow s_4 \vee s_5 \vee s_7,\ d_3 \rightarrow s_7 \vee s_8,\ d_1 \rightarrow s_4,\\ &d_2 \rightarrow s_4,\ d_2 \rightarrow s_6,\ d_2 \rightarrow s_7,\ d_3 \rightarrow s_7,\ d_3 \rightarrow s_9,\ s_7 \rightarrow s_8,\ d_4 \rightarrow d_2,\\ &d_5 \rightarrow d_2,\ d_1 \rightarrow c_1,\ d_1 \wedge t_1 \rightarrow c_2,\ d_4 \rightarrow c_3,\ d_5 \rightarrow c_4,\ d_4 \rightarrow \neg c_4,\ d_5 \rightarrow \neg c_3\} \end{aligned} \tag{4–2}$$

4.3 推理引擎

本节会介绍推理引擎的运行机制。推理引擎的主要功能就是结合知识库转换器输出的公式集以及患者症状信息编码而成的公式集，将它们组合成可满足性问题的实例，然后去求解可满足性问题从而排除无关的疾病，最后输出候选疾病集。

4.3.1 构建可满足性问题实例

在我们构建可满足性问题实例时，需要用到三类信息：

(1) 相关知识子图的知识，即知识库中与患者主述相关的知识。根据公式集 4–2我们可以构建相关知识子公式 $f_1 = (d_1 \rightarrow s_1 \vee s_2 \vee s_3 \vee s_4) \wedge (d_2 \rightarrow s_4 \vee s_5 \vee s_7) \wedge (d_3 \rightarrow s_7 \vee s_8) \wedge (d_1 \rightarrow s_4) \wedge (d_2 \rightarrow s_4) \wedge (d_2 \rightarrow s_6) \wedge (d_2 \rightarrow s_7) \wedge (d_3 \rightarrow s_7) \wedge (d_3 \rightarrow s_9) \wedge (s_7 \rightarrow s_8) \wedge (d_4 \rightarrow d_2) \wedge (d_5 \rightarrow d_2) \wedge (d_1 \rightarrow c_1) \wedge (d_1 \wedge t_1 \rightarrow c_2) \wedge (d_4 \rightarrow c_3) \wedge (d_5 \rightarrow c_4) \wedge (d_4 \rightarrow \neg c_4) \wedge (d_5 \rightarrow \neg c_3)$。

(2) 患者的症状信息，包括患者的主述以及之后通过交互获取的患者症状信息。我们依然用图 4–3为例，患者的主述信息为“咳嗽，恶心”。假设通过与患者的交互了解到患者“发热，无腹痛”。则可以构建患者症状信息的子公式 $f_2 = (s_4) \wedge (s_7) \wedge (s_6) \wedge (\neg s_9)$。当然，我们知道的患者症状信息自然越多越好，但是为了保证交互的友好性，我们会优先选择包含信息量最多的症状去询问患者，使得诊断过程可以尽快收敛（如何选择症状询问我们会在 4.5.2节详细解释）。同时，我们还会设定问题数量的阈值，问题数量达到阈值自动结束诊断，不会因为问题数量太多使得患者厌烦。

(3) 患者必须患有一种疾病的假设。我们认为既然患者前来就诊，并且说出自己的主述症状，那么患者必然患有相关知识子图中的某一个或多个疾病。则我们可以构建子公式 $f_3 = (d_1 \vee d_2 \vee d_3 \vee d_4 \vee d_5)$。

最终合取三个子公式构建了可满足性问题的实例，即命题逻辑公式 $f = f_1 \wedge$

$f_2 \wedge f_3$, 即:

$$
\begin{aligned}
f =& (d_1 \rightarrow s_1 \vee s_2 \vee s_3 \vee s_4) \\
& \wedge (d_2 \rightarrow s_4 \vee s_5 \vee s_7) \\
& \wedge (d_3 \rightarrow s_7 \vee s_8) \\
& \wedge (d_1 \rightarrow s_4) \\
& \wedge (d_2 \rightarrow s_4) \\
& \wedge (d_2 \rightarrow s_6) \\
& \wedge (d_2 \rightarrow s_7) \\
& \wedge (d_3 \rightarrow s_7) \\
& \wedge (d_3 \rightarrow s_9) \\
& \wedge (s_7 \rightarrow s_8) \\
& \wedge (d_4 \rightarrow d_2) \\
& \wedge (d_5 \rightarrow d_2) \\
& \wedge (d_1 \rightarrow c_1) \\
& \wedge (d_1 \wedge t_1 \rightarrow c_2) \\
& \wedge (d_4 \rightarrow c_3) \\
& \wedge (d_5 \rightarrow c_4) \\
& \wedge (d_4 \rightarrow \neg c_4) \\
& \wedge (d_5 \rightarrow \neg c_3) \\
& \wedge (s_4) \\
& \wedge (s_7) \\
& \wedge (s_6) \\
& \wedge (\neg s_9) \\
& \wedge (d_1 \vee d_2 \vee d_3 \vee d_4 \vee d_5)
\end{aligned}
\tag{4–3}
$$

4.3.2 求解候选疾病集

我们通过使用 SMT 求解器求解可满足性来确定候选疾病。具体来说，如果 $f \wedge d_i$ 是可满足的，那么 d_i 就是候选疾病，反之则不是。因为 $f \wedge d_i$ 可满足，意味着存在一组变量的赋值 $\mathcal{V}$ 使得 $\mathcal{V}(f \wedge d_i) = True$，则有 $\mathcal{V}(f \rightarrow d_i) = True$。也就是说在赋值 $\mathcal{V}$ 下, 我们可以从 f 推导出 d_i，即可以从相关医学知识和患

者的症状信息推导出患者患有疾病 d_i，因此患者可能患有疾病 d_i。我们用伪代码 4–2来表示求解候选疾病集的过程。同样基于图 4–3的例子，患者的症状信息为“咳嗽，恶心，发热，无腹痛”，我们可以求得候选疾病集为 $\{d_1, d_2, d_4, d_5\}$，即 {肺癌, 流感, 病毒性流感, 细菌性流感}，疾病“胃炎”被排除。

4.4 问题生成器

问题生成器即问题生成器，它主要做两件事情（如图 4–4）：

(1) 生成问题。因为推理引擎在求解过程中需要知道更多逻辑变量的取值信息，所以它会将它需要知道赋值信息的逻辑变量抛给问题生成器来处理，问题生成器会将逻辑变量解码成症状，然后查看症状是主观可感，还是查体可知或者是检验才能得知。比如“发热”就是查体项目，问题生成器会生成一个查体问题，提示外部硬件设备对患者做体温检测。而“腹痛”是主观可感症状，问题生成器会直接抛出一个单选题“请问您是否腹痛”让患者选择。而“白细胞增高”需要专业的血常规检查才能得知，所以问题生成器不会处理整个症状[1]，并反馈给推理引擎。另外，问题还会处理症状的性别特征，比如患者是男性，而如果推理引擎需要知道“软巢囊肿”的赋值信息，问题生成器则不会为这个症状生成问题。

(2) 分析问题答案。当收到患者的答案（或者是硬件检测的结果），问题生成器会将结果再编码为赋值之后的逻辑变量返回给推理引擎。比如推理引擎需要知道 s_9 的赋值，即需要知道患者是否有“腹痛”症状，问题生成器生成单选题“请问您是否腹痛”让患者作答。假设患者回答没有腹痛，问题生成器则将 s_9 的赋值，即 $\{s_9 : False\}$ 返回给推理引擎迭代求解。

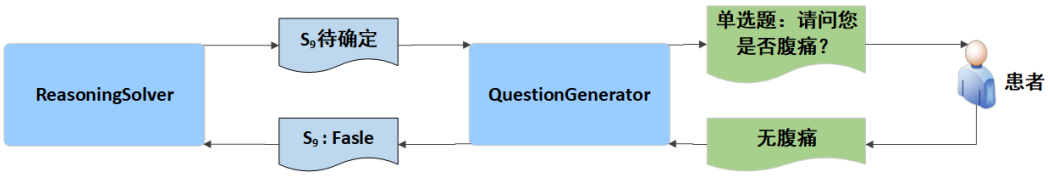


图 4–4 问题生成器运行流程示例

Figure 4–4 Example of running process of QuestionGenerator

[1]我们的系统有已经出厂的硬件诊断舱，可以完成体温、体重、血压等基本特征的检测。问题生成器会根据硬件的能力来决定是否处理特定症状。

4.5 诊断计算器

本节会介绍诊断计算器的主要功能。诊断计算器主要负责症状以及疾病得分的计算：1) 计算症状的得分，帮助推理引擎选择包含信息量最多的交互症状询问患者，使得诊断过程尽快收敛 2) 计算疾病得分，从而为推理引擎输出的候选疾病排序，最终选择得分最高的 5 个疾病（5 是阈值，可以调整），提高诊断结果的区分度。

4.5.1 疾病评分

因为推理引擎采用逻辑求解的方式，其只会排除无关的疾病，对于候选疾病却无先后之分。举个通俗的例子，比如患者说“我头痛，脚不痛”，根据患者的症状信息逻辑只能排除患者没有脚部疾病，而头部以及腹部疾病都是候选疾病。但是既然患者只说明自己“头痛”，并没有说自己任何腹部的症状，我们其实可以认为患者患有头部疾病的可能性要大于腹部疾病（这里并不能直接排除腹部疾病，因为可能真的有一些腹部疾病通过内部生理机制引起头痛）。所以，我们认为“疾病 A 与患者症状的关联度高于疾病 B（患者的症状信息对疾病 A 的贡献度高于 B），则患者患有疾病 A 的可能性高于疾病 B”。我们采用统计的方式来量化症状对疾病的贡献度，具体就是采用了 TF-IDF[23]（term frequency–inverse document frequency）方法来评价贡献度。TF-IDF 是一种统计方法，用以评估一关键词对文件集中一篇文档的重要程度。关键词对某篇文档的重要程度与它在文档中的出现次数成正比，但与它在整个文件集中出现的次数成反比。我们认为每个疾病都是一篇“文档”，这个疾病的症状就是文档中的“关键词”，如果对于患者具有某个症状 s，但是 s 对于疾病 A 的“关键程度”大于 B，则疾病 A 的得分高于疾病 B。

给定所有症状的集合为 S，集合 S 中某个症状 s，所有候选疾病集合 $D_{candidate}$，集合 $D_{candidate}$ 中的某个候选疾病 d，则症状 s 对候选疾病 d 的贡献度可以按如下方式计算：

$$\begin{aligned} w_{s,d} &= tf_{s,d} \times idf_{s,D_{candidate}} \\ &= \frac{n_{s,d}}{\sum\limits_{s_i \in S}(n_{s_i,d} + n_{\neg s_i,d})} \times \frac{|d' \in D_{candidate} : s \in d'|}{|D_{candidate}|} \end{aligned} \tag{4–4}$$

$$\begin{aligned} w_{\neg s,d} &= tf_{\neg s,d} \times idf_{\neg s,D_{candidate}} \\ &= \frac{n_{\neg s,d}}{\sum\limits_{s_i \in S}(n_{s_i,d} + n_{\neg s_i,d})} \times \frac{|d' \in D_{candidate} : \neg s \in d'|}{|D_{candidate}|} \end{aligned} \tag{4–5}$$

其中 $n_{s,d}$ 表示症状 s 在疾病 d 中正出现的次数，而 $n_{\neg s,d}$ 表示示症状 s 在疾病 d 中负出现的次数。比如“膀胱炎会引起尿痛，而不会引起高烧”，则”尿痛”在“膀胱炎”中正出现了一次，“高烧”在“膀胱炎”中负出现了一次。$|d' \in D_{candidate} : s \in d'|$ 表示包含症状 s 正出现的候选疾病数量。$w_{\neg s,d}$ 表示某个症状的负出现对某个疾病的贡献度，比如“膀胱炎会引起尿痛，而不会引起高烧”，则“尿痛”和“不高烧”会对“膀胱炎”有贡献度，而“高烧”对“膀胱炎”没有贡献。

接着我们定义 $n_{\hat{s},d}$（$\hat{s}$ 代表 s 的正出现或者反出现，即 $\hat{s} = s$ 或 $\hat{s} = \neg s$），症状 $\hat{s}$ 在疾病 d 出现的次数自然不能单纯地使用症状 $\hat{s}$ 作为关键词在医书疾病文档 d 出现的次数来定义，因为这种单纯文本匹配的方式（IBM Watson 采用的策略）只考虑了症状关键词是否在疾病文档中出现，而没有考虑它们之间的逻辑关系。比如在《肺结核》文档中有描述，“咳嗽是肺结核的常见症状，咳嗽是一种呼吸道症状”，这里“咳嗽”虽然在“肺结核”文档中出现两次，但是第二次只是对“咳嗽”的描述，其实和肺结核没有任何关系，它的出现不应该对“肺结核”有任何贡献。因此我们采用知识库中的知识来表达症状在疾病中出现的次数，因为知识库中的知识规则只考虑疾病与症状之间的逻辑关系，而没有逻辑关系的无关描述并不会转化成逻辑规则。比如“咳嗽是肺结核的常见症状，咳嗽是一种呼吸道症状”会提取出两条规则“肺结核 → 咳嗽，咳嗽 → 呼吸道症状”，基于这两条规则，“咳嗽”只在“肺结核”中出现了一次。对于相关知识规则集 $\boldsymbol{R}$，其中的某条规则 $r = (\hat{s_{l_1}} \wedge ... \wedge \hat{s_{l_m}}) \wedge (d_{l_1} \wedge ... \wedge d_{l_n}) \wedge (\hat{v_{l_1}} \wedge ... \wedge \hat{v_{l_k}}) \rightarrow (\hat{s_{r_1}} \vee ... \vee \hat{s_{r_p}}) \vee (d_{r_1} \vee ... \vee d_{r_q}) \vee (\hat{v_{r_1}} \vee ... \vee \hat{v_{r_t}})$，规则左边的症状集为 $S_l = \{\hat{s_{l_1}}, ..., \hat{s_{l_m}}\}$，疾病集 $D_l = \{d_{l_1}, ..., d_{l_n}\}$，右边的症状集为 $S_r = \{\hat{s_{r_1}}, ..., \hat{s_{r_p}}\}$，疾病集 $D_r = \{d_{r_1}, ..., d_{r_q}\}$。注意这里 S_l 和 S_r 中的症状可能是正出现也可能是负出现，而 D_l 和 D_r 都是正出现，比如 $s_1 \wedge \neg s_2 \wedge d_1 \rightarrow s_3 \vee \neg s_4 \vee d_2$。从逻辑的角度看，是左边的症状会导致右边的疾病，左边的疾病会导致右边的症状，而左边（右边）的症状和疾病之间其实没有任何逻辑关系。我们认为 $S_l(S_r)$ 中的所有症状在 $D_r(D_l)$ 中的每个疾病中一共出现了一次。即给定某条规则公式 r，S_l, S_r 分别为规则左右两边的症状集，D_l, D_r 分别为规则左右两边的疾病集，则有

$$\begin{aligned} \forall d \in D_r, \sum_{\hat{s} \in S_l} n_{lr\_\hat{s},d}(r) = 1 \\ \forall d \in D_l, \sum_{\hat{s} \in S_r} n_{rl\_\hat{s},d}(r) = 1 \end{aligned} \tag{4–6}$$

其中 $n_{lr\_\hat{s},d}(r)$ 表示在规则 r 中，左边症状在右边疾病中出现的次数,$n_{rl\_\hat{s},d}(r)$ 表示右边症状在左边疾病中出现的次数，然后我们将 $S_l(S_r)$ 中的症状在 $D_r(D_l)$ 中疾病

的出现次数平均，得到：

$$\begin{aligned} \forall d \in D_r, \forall \hat{s} \in S_l, n_{lr\_\hat{s},d}(r) = \frac{1}{|S_l|} \\ \forall d \in D_l, \forall \hat{s} \in S_r, n_{rl\_\hat{s},d}(r) = \frac{1}{|S_r|} \end{aligned} \tag{4–7}$$

然后我们可以定义公式 r 中症状在疾病中出现的次数，即：

$$n_{\hat{s},d}(r) = n_{lr\_\hat{s},d}(r) + n_{rl\_\hat{s},d}(r) \tag{4–8}$$

有了公式集中某一条公式中症状在疾病中出现次数的定义，我们可以定义在相关知识规则集中症状在疾病出现的次数。对于相关知识规则集 R, 有：

$$n_{\hat{s},d} = \sum_{r \in R} n_{\hat{s},d}(r) \tag{4–9}$$

我们可以使用伪代码 4–3来表示计算症状在疾病中出现次数的字典。

基于公式 4–4定义的症状对候选疾病的贡献度，我们可以定义候选疾病的得分。设患者的症状信息集合 $S_{patient}$(里面可能包含症状的正出现和负出现，比如患者“发热，无腹痛”)。注意患者的症状信息除了患者明确说出的症状之外（包括患者的主述以及通过与患者交互获得的症状），还包含我们知识库推断出的症状信息。比如如果知识库中有“咳嗽强导致声嘶”，患者说明自己“咳嗽”，我们可以推断出患者也具有“声嘶”症状。如果我们的知识库构建的全面且准确，这个功能将非常强大，因为它充分发挥了逻辑链条的作用，可以推断出尚未出现但准确的信息。给定某个候选疾病 d，则候选疾病的得分可以如下定义:

$$score_d = \sum_{\hat{s} \in S_{patient}} w_{\hat{s},d} \tag{4–10}$$

我们依然用图 4–2 为例，患者的主述信息为“咳嗽，恶心”。假设通过与患者的交互了解到患者“发热，无腹痛”，即患者说出的症状信息为“咳嗽，恶心，发热，无腹痛“，而通过相关知识的规则集，我们可以推断出患者会有“呕吐”症状（因为相关规则中存在规则“恶心 → 呕吐”），即患者的症状信息集合 $S_{patient} = \{s_4, s_7, s_6, \neg s_9, s_8\}$，可以计算出候选疾病集 $D_{candidate} = \{d_1, d_2, d_4, d_5\}$ 中每个疾病的得分，最后得出 $score_{d_2} > score_{d_1} > score_{d_4} = score_{d_5}$，即得分先后关系为“流感 > 肺癌 > 病毒性感冒 = 流行性感冒”。

4.5.2 症状评分

给症状打分主要是为了选择交互症状询问患者：推理引擎在推理过程中需要知道更多症状变量的信息，最简单的办法自然是把所有症状都向患者询问一遍，

但是这种交互方式显然不友好。所以推理引擎需要找到包含信息量最多的症状变量去询问患者，使得推理过程尽快收敛。症状打分是为了量化症状包含的信息量，使得推理引擎可以问较少的问题就可以做出诊断。这种量化方式业界没有统一的标准，我们主要尝试了两种量化方式：

(1) 最大熵

我们通过计算每个症状变量的熵值，选出熵值最大的变量去询问患者。假设候选疾病集合为 $D_{candidate} = \{d_1, d_2, ..., d_t\}$, 对于某个症状变量 s, 患者回答为 $True$ 的概率为 $p(s|d_1 \vee d_2 \vee ... \vee d_t)$, 我们设为 x。根据香农的信息熵理论[66]，变量 s 的熵为:

$$H(s) = -(xlogx + (1-x)log(1-x)) \tag{4–11}$$

其中 $x = p(s|d_1 \vee d_2 \vee ... \vee d_t)$, 但是这个值我们现在无法获取[1]，所以我们采用如下公式来模拟:

$$p(s|d_1 \vee d_2 \vee ... \vee d_t) \approx \sum_{d_i \in D} p(s \wedge d_i) = \sum_{d_i \in D} p(s|d_i) \times p(d_i) \tag{4–12}$$

其中 $p(s|d_i)$ 表示患者在患有疾病 d_i 的条件下，出现症状 s 的概率，我们可以用 tf_{s,d_i} 来估计，即

$$p(s|d_i) = tf_{s,d_i} = \frac{n_{s,d_i}}{\sum_{s_j \in S}(n_{s_j,d_i} + n_{\neg s_j,d_i})} \tag{4–13}$$

而 $p(d_i)$ 表示患者在当前情况下患有疾病 d_i 的概率，我们可以用疾病得分来估计。用 d_i 的得分除以所有候选疾病的总得分来估计患者患有疾病 d_i 的概率，即:

$$p(d_i) = \frac{score_{d_i}}{\sum_{d_j \in D_{candidate}} score_{d_j}} \tag{4–14}$$

基于公式 4–11- 4–14, 我们可以计算每个尚未确定赋值症状变量的熵值，然后挑选熵值最大的症状询问患者。基于图4–3的例子，患者的主述“咳嗽，恶心”，基于对剩余未确定赋值变量熵值的计算，我们会选择“发热”症状询问患者。

(2) 最大期望疾病贡献度

某个症状 s 的取值会对疾病有不同的贡献度，症状为 $True$ 会对包含 s 正出现的疾病有贡献，症状为 $False$ 会对包含 s 负出现的疾病有贡献。基于

[1]之后的工作我们会通过大量病历提取这个概率值

公式 4–4, 症状会对疾病的贡献度可以量化。对于候选疾病集 $D_{candidate} = \{d_1, d_2, ..., d_t\}$，某个症状 s 对候选疾病集 $D_{candidate}$ 贡献度的期望：

$$\begin{aligned} E(w_{s,D_{candidate}}) &= (\sum_{d_i \in D_{candidate}} w_{s,d_i}) \cdot p(s|d_1 \vee d_2 \vee ... \vee d_t) + (\sum_{d_i \in D} w_{\neg s,d_i}) \cdot p(\neg s|d_1 \vee d_2 \vee ... \vee d_t) \\ &= (\sum_{d_i \in D_{candidate}} w_{s,d_i}) \cdot p(s|d_1 \vee d_2 \vee ... \vee d_t) + (\sum_{d_i \in D} w_{\neg s,d_i}) \cdot (1 - p(s|d_1 \vee d_2 \vee ... \vee d \end{aligned} \tag{4–15}$$

基于公式 4–15, 我们可以计算每个尚未确定赋值症状变量对候疾病集合贡献度的期望值，选出期望值最大的症状来询问患者。

4.6 本章小结

本章主要介绍了辅助诊断内核的内部架构和各模块的实现机制。辅助诊断内核主要包含四个模块，分别是知识库转换器，推理引擎，问题生成器以及诊断计算器。知识库转换器主要负责根据患者的主述信息从医学知识库中提取相关的医学知识并把他们转换成逻辑规则，其中我们介绍了如何将知识中的关系编码成命题逻辑联结词 (表 4–2)。推理引擎是诊断内核的核心模块，它的主要任务是接收知识库转换器输出的逻辑规则，根据其构建可满足性问题实例来求解，从而排除无关疾病得到候选疾病集。因为推理引擎在推理过程中会需要知道更多的逻辑变量的赋值信息，它会将这些逻辑变量抛给问题生成器处理，问题生成器会将这些变量解码成具体症状然后生成相应的问题去询问患者，得到患者的答案后在编码成赋完值的逻辑变量交给推理引擎迭代求解。而诊断计算器可以为症状以及疾病打分，为症状打分可以指导推理引擎选择包含信息量最多的症状去询问，从而可以用较少的问题就可以得出诊断。在症状打分中，我们分别介绍了最大熵、最大期望疾病贡献度两种打分方式。而为疾病打分的目的是增加候选疾病的区分度，因为推理引擎得出的候选疾病可能很多，通过疾病打分就可以给它们排序，从而最后只输可能性最高的几个疾病。

算法 4–1 获取单一症状相关知识子图 DFS 函数伪代码

Input: 知识库中知识图 G，患者的某个主述症状 s，从主述节点延伸出的步数 hop，已经访问过的节点集合 $Visited$

Output: 症状 s 相关的知识子图 $G_{s,hop}$

```
Function DFS(G, s, hop, Visited):
    G_s ← ϕ;
    if hop ≤ 0 Or s in Visited then
        return G_s;
    else
        for e in G_E And (e = (s, s') Or e = (s', s)) do
            if s' in Visited then
                Continue;
            else
                add s' to Visited;
                if type(e) == 是一种 Or 强导致 Or 不导致 then
                    将 s' 加入 G_s 的点集，将 e 加入 G_s 的边集;
                    G_s ← G_s ∪ DFS(G, s', hop − 1, Visited);
                else if type(e) == 并发症 then
                    将 s' 加入 G_s 的点集，将 e 加入 G_s 的边集
                    G_s ← G_s ∪ DFS(G, s', hop − 0.5, Visited);
                else if type(e) == 同义词 then
                    将 s' 加入 G_s 的点集，将 e 加入 G_s 的边集
                    G_s ← G_s ∪ DFS(G, s', hop, Visited);
                else if type(e) == 条件为 Or 否定修饰 Or 然后 then
                    T ← e 对应的上层嵌套实体节点
                    add T to Visited;
                    将 s', T 加入 G_s 的点集，将 e 加入 G_s 的边集
                    G_s ← G_s ∪ DFS(G, T, hop, Visited);
                else if type(e) == 与 Or 或 then
                    T ← s 的顶层 Clause 嵌套实体节点
                    将 s', T 加入 G_s 的点集，将 T 中的所有边都加入 G_s 的边集
                    add T to Visited;
                    G_s ← G_s ∪ DFS(G, T, 1, Visited);
                    for t in T do
                        add t to Visited;
                        G_s ← G_s ∪ DFS(G, t, 1, Visited);
                    end
                else
                    Continue;
                end
            end
        end
        return G_s;
    end
```

表 4–1 实体编码成逻辑变量

Table 4–1 Encoding entities to logic variables

实体类型	实体名称	逻辑变量
疾病	肺癌、流感、胃炎、病毒性流感、细菌性流感	d_1, d_2, d_3, d_4, d_5
症状	咯血、痰中带血、胸痛、咳嗽、流涕、发热、恶心、呕吐、腹痛	$s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8, s_9$
检验指标	肺结节、肺部阴影、淋巴细胞增高、白细胞增高	c_1, c_2, c_3, c_4
阶段	晚期	t_1

算法 4–2 求解候选疾病集伪代码

Input: 可满足性问题实例公式 f, 相关知识子图中所有疾病集合 D

Output: 候选疾病集合 $D_{candidate}$

```
D_candidate ← Φ;
for each d in D do
    if f ∧ d is SATISFIABLE then
        add d to D_candidate;
    end
end
return D_candidate;
```

算法 4–3 求解症状在疾病中出现次数伪代码

Input: 相关知识规则集合 R

Output: 症状在疾病中出现次数的字典

```
D 为 R 中所有疾病集合;
S 为 R 中所有症状集合;
T 初始化为空字典;
for each s in S do
    for each d in D do
        T[s][d] = 0;
        T[¬s][d] = 0;
    end
end
for each r in R do
    S_l 为 r 左边症状集;
    S_r 为 r 右边症状集;
    D_l 为 r 左边疾病集;
    D_l 为 r 右边疾病集;
    for each ŝ in S_l do
        for each d in D_r do
            if ŝ 为 s 正出现 then
                T[s][d] += 1/|S_l|;
            else
                T[¬s][d] += 1/|S_l|;
            end
        end
    end
    for each ŝ in S_r do
        for each d in D_l do
            if ŝ 为 s 正出现 then
                T[s][d] += 1/|S_r|;
            else
                T[¬s][d] += 1/|S_r|;
            end
        end
    end
end
return T;
```

表 4–2 知识关系编码成命题逻辑联结词

Table 4–2 Encoding knowledge relations as propositional logical connectives

关系名称	命题逻辑联结词	例子
强导致	蕴含 (→)	如“（胃炎，强导致，恶心）”可以编码成“胃炎 → 恶心”
弱导致	弱导致不是强因果关系，不会对其编码	——
不导致	导致否 (→ ¬)	如“(膀胱炎，不导致，高烧)”可以编码为“膀胱 → ¬高烧”
拼接	在知识存储时已经将“a 拼接 b”关系合并成一个实体“ab”，所以此关系在知识库中不存在	——
与	与 (∧)	如“(流感，强导致，与（发热，流涕))”可以编码成“流感 → 发热 ∧ 流涕”
或	或 (∨)	如“(肺癌，强导致，或（咯血，痰中有血))”可以编码成“肺癌 → 咯血 ∨ 痰中有血”
条件为	与 (∧)	如“((肺癌，条件为，晚期)，强导致，肺部阴影)”可以编码成“肺癌 ∧ 晚期 → 肺部阴影”
调查病史	调查病史不是强因果关系，不会对其编码	——
是一种	蕴含 (→)	如“(病毒性流感，是一种，流感)”可以编码成“病毒性流感 → 流感”
同义词	等价 (↔)	如“（水肿，同义词，edema）”可以编码为“水肿 ↔ edema”
指代	在知识存储时已经将“a 指代 b”中的 a 用 b 替换，所以此关系在知识库中不存在	——
然后	会将“a 然后 b”关系合并成一个实体“a 然后 b”	如“(糖尿病，导致，（尿微量蛋白，然后，大量尿蛋白））”可以编码成“糖尿病 → 微量尿蛋白然后大量尿蛋白”
并发症	并发症不是强因果关系，不会对其编码	——
否定修饰	否定 (¬)	如“(肺炎，强导致，（不，否定修饰，腹痛）)”可以编码成“肺炎 → ¬腹痛”

第五章　知识纠错

本章主对知识纠错模块进行介绍。因为我们主要通过不满足核技术来检测矛盾规则来获取错误知识，因此本文的错误知识都是知识库中互相矛盾的知识。我们首先会说明错误知识的来源，接着会介绍如何根据可满足性问题中的不可满足核来进行错误知识的纠正。最后我们会分析已经检测出的错误，总结出常见错误遵循的模式。

5.1　错误知识来源

本节主要分析错误知识的来源。如 3.2.1节所述，我们医学文本主要是人卫医书教材、BMJ Best Practice 以及疾病百科，然后通过自然语言处理加上人工审核的方式来抽取知识从这些医学文本中抽取知识，因此我们的错误知识要么来自于这些医学文本，要么就是抽取过程中引入了错误。通过分析，我们发现错误知识主要来源于以下几个方面：

- 医学文本中包含互错误的知识：我们的医学文本主要是医书教材，BMJ Best Practice 以及疾病百科。其中医书教材和 BMJ Best Practice 都较为权威，但是爬取的疾病百科数据质量一般，经常会出现一些错误的医学知识。比如经常混淆疾病的各种亚型，如“二型糖尿病“会有“多饮、多食以、多尿”，但是这些其实是“一型糖尿病”的症状，对于“二型糖尿病”并不明显。
- 不同的医学文本中存在互相矛盾的知识：因为我们从多个医学知识源中抽取知识，而且对于疾病百科这种质量一般的数据源，其中对一些病的描述会于医书中的描述相矛盾，比如医书中对于膀胱炎的描述“膀胱炎会引起泌尿系统正在，比如尿频、尿急、尿痛，但一般不会伴有高烧”，但是我们爬取的百科中会有相矛盾的描述“膀胱炎会引起高烧”。遇到这种情况，我们都是以医书为准。
- 知识抽取过程引入错误知识：因为主要通过自然语言处理加人工审核的方式来抽取知识，在抽取过程中引入错误十分常见，这是我们的主要错误来源。首先是实体类型标记错误，比如将“甲亢性”标记成病症，其应该是修饰语类型，而“甲亢”才是病症。其次对于一些实体关系经常标记错误：
  - 混淆“强导致”和“弱导致”关系。比如“脑膜炎可能会引起咳嗽”，这里“脑膜炎”和“咳嗽”其实是“弱导致”关系，但是标记人员可能会误标成“强导致”关系。

- 混淆“强导致”和“是一种”关系，这种错误十分常见。比如“呕吐、腹泻等消化道中度症状”，这里其实表达的意思是“消化道中毒会导致呕吐、腹泻”，它们之间是“强导致”关系。但是标记人员经常误标记成“是一种”关系。
- 错误的“否定修饰”关系。比如“典型性肺炎”和“非典型性肺炎”，我们标记人员会将“非典型性肺炎”标记成（非，否定修饰，典型性肺炎），也就是说如果用 d 表示“典型性肺炎”，那么“非典型性肺炎”就是 $\neg d$。因为 d 和 $\neg d$ 在逻辑上必为一真一假，也就意味着患者必须患有其中一个疾病，而这显然是错误的。其实“非典型性肺炎”和“典型性肺炎”并不是逻辑上的否定关系，它们只是不能同时为真，而不是必须一真一假。

5.2 知识纠错流程

本节主要介绍如何通过可满足性问题中的不可满足核来进行知识的纠错。我们主要是通过辅助诊断内核来发现错误知识。具体来说，辅助诊断内核通过知识库转换器生成命题逻辑公式，然后推理引擎根据这些公式构建可满足性问题的实例，在推理引擎使用 SMT 求解器求解可满足性问题的时候。如果可满足，那么就结束诊断或者询问更多症状继续推理。而如果不可满足，则说明知识库中的知识存在矛盾，或者患者的症状信息和知识库中的知识存在矛盾。我们可以通过不可满足核来获取这个矛盾，因为不可满足核是不可满足公式的子句集，因此其还是命题逻辑公式的形式，我们还需要把这些命题逻辑公式在解码成医学知识规则。然后通过人工去审核这些规则，继而修改其中的错误知识。

我们通过图 5–1来举例说明我们知识纠错的流程：

(1) 提取主述相关知识并编码

首先根据患者的主述信息从知识库中获取相关知识，通过知识库转换器将相关知识编码成命题逻辑公式集合。如图中步骤 1 所示，我们根据患者的主述“黄疸”获取了相关知识并编码成了逻辑公式集合 $R = \{s_4 \rightarrow s_1, s_1 \rightarrow d_1, d_1 \rightarrow s_5 \vee s_6, d_1 \rightarrow s_2, s_2 \rightarrow \neg s_3, d_1 \rightarrow d_2, d_2 \rightarrow d_3, d_3 \rightarrow d_4, d_4 \rightarrow s_3\}$（其中实体的编码方式为 {黄疸 ： s_1，溶血性黄疸 ： s_2，皮肤瘙痒 ： s_3，肝功能减退 ： s_4，胆汁淤积 ： s_5，肝大 ： s_6，溶血 ： d_1，贫血 ： d_2，多发性骨髓瘤 ： d_3，高钙血症 ： d_4}）。

(2) 构建可满足性问题实例

然后通过推理引擎根据逻辑公式集合构建可满足性问题的实例 f。如图中步骤 2，构建了可满足性问题实例公式 $f = (s_4 \rightarrow s_1) \wedge (s_1 \rightarrow d_1) \wedge (d_1 \rightarrow$

$s_5 \vee s_6) \wedge (d_1 \rightarrow s_2) \wedge (s_2 \rightarrow \neg s_3) \wedge (d_1 \rightarrow d_2) \wedge (d_2 \rightarrow d_3) \wedge (d_3 \rightarrow d_4) \wedge (d_4 \rightarrow s_3) \wedge (s_1) \wedge (d_1 \vee d_2 \vee d_3 \vee d_4)$。

(3) 求解最小不可满足核

如果 f 是不可满足的，那么我们通过 SMT 求解器求解其最小不可满足核（使用 z3 SMT 求解器的 unsat_core 接口），如图中步骤 3，得到 $unsat\_core = \{s_1, s_1 \rightarrow d_1, d_1 \rightarrow s_2, s_2 \rightarrow \neg s_3, d_1 \rightarrow d_2, d_2 \rightarrow d_3, d_3 \rightarrow d_4, d_4 \rightarrow s_3\}$。*unsat_core* 中除了 s_1 是患者的主述信息，即“黄疸”。其余的都是知识库中的知识规则。

(4) 解码最小不可满足核

因为最小不可满足核是 f 的子句集，也是命题逻辑公式的形式。所以我们会把不可满足核中的子句都解码回源知识，如图中步骤 4。不可满足核中的知识可能包括患者的主述信息，患者肯定患有一种疾病的假设和知识库中的规则（详见4.3.1节）。我们认为患者的主述信息不会出错，基于患有一种病的假设也没有问题，所以一定是不可满足核中的知识库知识出现了错误。如图中的 *unsat_core*，我们认为其中的 s_1，即患者主述“黄疸”没有问题，所以我们解码不可满足核中的其他逻辑公式，从这些逻辑公式中找出错误。因为我们采用最小不可满足核，所以不可满足核中的子句数量不会太多，一般在 3-15 条左右。我们通过人工校对的方式去检查不可满足核中的知识库知识。找到错误知识并回溯到知识来源（我们知识库中的每一条知识都记录了来源），重新阅读医学文本来修改错误知识。其实图中示例就是标记时将（多发性骨髓瘤，强导致，贫血）标记成了（贫血，是一种，多发性骨髓瘤）。

5.3 常见错误模式

我们通过知识纠错的方案找到了很多知识库中的错误。通过对这些错误知识的分析，我们总结出一些比较常见的模式（pattern）。符合这些 pattern 的知识，如果根据患者的症状信息加上这些知识构建可满足性问题实例公式，那么这个公式有很大概率是不可满足的。为了说明这些 pattern，我们需要做两点说明：1) 逻辑上的“蕴含”符号，即”→” 代表知识规则中的“是一种”或者“强导致”关系（详见表4–2）。2）后文 pattern 中的英文字母代表知识库中的实体节点，它们可能是原子实体，比如“肺结核”，“咳嗽”等，也可能是嵌套实体，比如“（肺结核，条件为，晚期）”，“（咳嗽，与，发烧）”等。我们通过分析已经发现的不可满足核，发现以下四种 pattern 很容易引起矛盾：

(1) 环：$a \rightarrow^* a$。

这个 pattern 意味着环上所有节点取值必须保持一致。如果从一个节点 a 出发，只考虑“强导致”和“是一种”关系，不停前驱探索，如果最终可以回到节点 a 构成环。那么这条环上的知识和患者的症状信息就很容易产生矛盾。因为构成环意味着环上的节点都是等价的，它们的取值应该保持一致。而如果患者的症状信息不能保证环上的节点取值一致，那么就会产生矛盾。比如知识库中存在环“肺结核 → 咳嗽 → 消瘦 → 肺结核”，而患者的症状信息是“咳嗽，无消瘦”，那么就会产生矛盾。导致根据知识库知识和患者症状信息构建的可满足性问题实例不可满足。在这个环中，“咳嗽 → 消瘦”和“消瘦 → 肺结核”都是错误的知识，它们的关系可能是弱导致，而非强导致。

(2) 自相矛盾 $a \rightarrow^* \neg a$。

从一个节点 a 出发可以推断出自己的否定 $\neg a$，那么无论 a 的取值如何，都会引起矛盾。

(3) 永假推理：$a \rightarrow^* b$, $a \rightarrow^* \neg b$

这个 pattern 意味着 a 的取值只能为 $False$。如果知识库中同时存在这两条推理路径，根据逻辑上“蕴含”的真值表定义，如果 a 的取值为 $True$，那么 b 和 $\neg b$ 的取值都应该为 $True$, 这样就产生的矛盾。所以为了保证可满足性，a 的取值只能为 $False$。但是如果患者的症状信息表明确实患有 a 症状，则会产生矛盾。比如知识库存在“咳嗽 → 咽痛”和“咳嗽 →¬ 咽痛”两条规则，而如果患者确实有“咳嗽”症状，则会产生矛盾。在这个例子中，“咳嗽 →¬ 咽痛”显然是错误知识。

(4) 永真推理：$a \rightarrow^* b$, $\neg a \rightarrow^* b$。

这个 pattern 意味着 b 的取值只能为 $True$。这个 pattern 和第三个 pattern 等价，相当于上一个 pattern 的逆否。根据命题逻辑“蕴含”的定义，无论 a 的取值如何，b 的取值只能为 $True$。比如医学文本中有“典型性肺炎和非典型性肺炎都会引起咳嗽”，知识抽取时抽取到了两条规则“典型性肺炎 → 咳嗽”和“¬ 典型性肺炎 → 咳嗽”。根据这两条规则，“咳嗽”的取值只能为 $True$。而如果患者确实没有“咳嗽”症状，则会产生矛盾。其实这个例子中，非典型性肺炎 $\neq$ ¬ 典型性肺炎。因为在逻辑上，a 和 $\neg a$ 意味着它们必有一真一假。而真实情况是“典型性肺炎”和“非典型性肺炎”并不是必定一真一假，它们只是不能都为真，但是可能都为假。这种情况非常令人困惑，在我们知识抽取的初期有很多这样的错误。

虽然可能产生不可满足核的错误知识有上面四种模式，但是产生这些错误的原因基本都是知识抽取过程中引入了错误。比如标记人员经常会把“弱导致”标记成“强导致”，这会使得很多本来没有蕴含关系的实体间产生了蕴含关系，比如“脑膜炎可能会引起咳嗽”，这里“脑膜炎”和“咳嗽”其实是“弱导致”关系，如果标记成“强导致”则会使得“脑膜炎”和“咳嗽”之间产生蕴含关系；又比如标记人员会混淆“强导致”和“是一种”关系，这会使得很多实体间的蕴含关系方向错误，比如“呕吐、腹泻等消化道中度症状”，正确的知识应该是“消化道中毒强导致呕吐、腹泻”，而标记人员可能会错误的标记成“呕吐、腹泻是一种消化道中毒”，这就会使得本来“消化道中毒蕴含呕吐、腹泻”被标记成“呕吐、腹泻蕴含消化道中毒”，即蕴含的方向发生了错误；还有错误的“否定修饰”关系，比如“非典型肺炎”其实不应该被标记成“(非，否定修饰，典型性肺炎)”，它和“典型性肺炎”并不是逻辑上的否定关系，这就会产生很多模式 4 的错误。当然很多实体类型的标记错误，比如把“病症”标记成“修饰语”，“病症”标记成“生理概念”等等，这些错误都可能导致知识库中的知识产生矛盾。

5.4 本章小结

本章主要介绍了知识纠错模块。我们首先说明了错误知识的来源，它们可能是医书中的错误，多种不同知识源互相矛盾的错误，又或者是在我们知识提取过程中引入的错误；接着我们介绍了知识纠错的方案，主要是通过 SMT 求解器获取不可满足公式的不可满足核，然后将不可满足核解码成知识库中的知识规则，最后通过人工校对的方式来修改错误知识；最后我们分析了所有已经发现的错误知识，并总结出了它们的模式。知识纠错功能可以显著提高知识库的数据质量，可以明显提升辅助诊断的准确率。

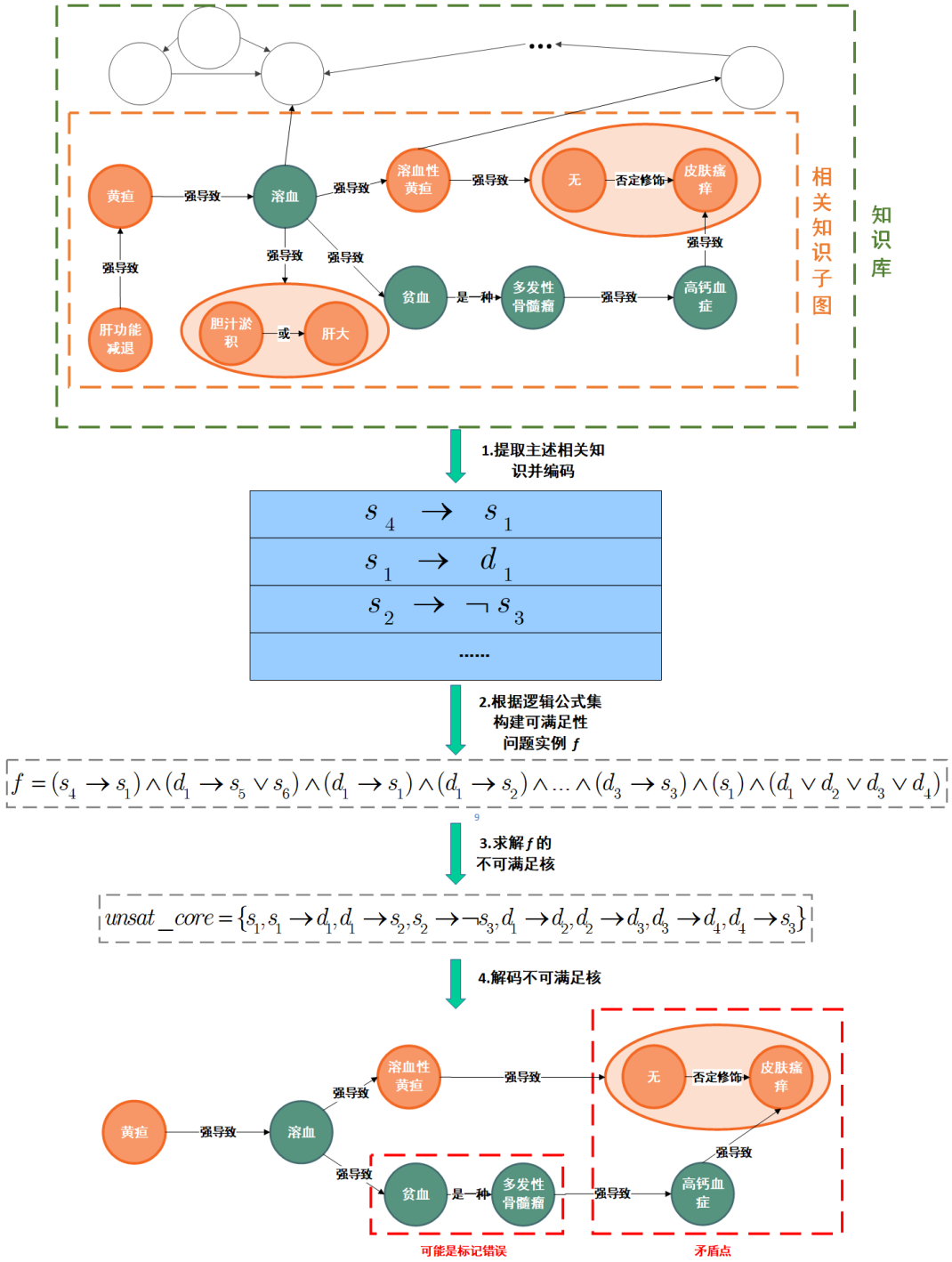


图 5–1 知识纠错流程

Figure 5–1 The process of knowledge correction

第六章　实验及结果分析

本章是实验部分。我们通过医学知识库，辅助诊断内核以及知识纠错模块构建了完整的辅诊诊断系统，所以本章主要通过实验对其进行分析。我们首先会介绍辅助诊断系统的架构，接着会分析医学知识库的规模，然后会通过真实病历来测试辅助诊断的性能，最后会通过实验数据分析错误知识的模式。

6.1　系统架构

本文实现了辅诊诊断内核不仅仅是停留在理论原型层面，而是结合知识库，诊断内核，知识纠错模块实现了实际可用的辅助诊断系统。我们将辅诊诊断系统安装在团队自己设计的医疗诊断舱中，并且已经将医疗诊断舱部署在合作医院中落地测试。本节主要介绍辅助诊断系统的架构设计。

如图6–1，整个辅助诊断系统分为三个模块：

(1) 医学知识库：利用自然语言处理结合人工审核的方式，从医书、BMJ Best Practice 等可信知识源中抽取医学知识，生成嵌套 RDF 三元组集并把它们存入 Neo4j 图数据库。为了提高查询性能原因，我们会把 Neo4j 中的数据缓存到 Redis 中供辅助诊断内核使用。详细介绍参考第三章。

(2) 辅助诊断模块：辅助诊断内核主要是结合知识库中的知识和患者的症状信息来对患者的疾病做预测。为了保证系统的可靠性，辅助诊断内核并不直接对外暴露，而是通过另外搭建的主引擎（Main Engine）来对外提供服务。为了模块之间的充分解耦，我们采用了微服务架构[67]，搭建了两个服务器来提供不同的服务：

- 逻辑服务器（LogicServer）：逻辑服务器是整个辅诊系统的核心，它可以完整的完成辅助诊断的功能。它利用辅助诊断内核（详细介绍参考第四章）来提供交互问题生成，疾病预测等功能。逻辑服务器会把所有的诊断记录（Diagnosis Record）保存在 ElasticSearch 数据库中，供数据分析服务器使用。
- 数据分析服务器（DAServer）：数据分析服务器主要是根据逻辑服务器生成的诊断记录进行数据分析。到目前为止，它可以根据逻辑服务器的诊断的疾病来匹配相应的科室或者医生。比如逻辑服务器诊断某个患者患有“肺结核”，那么数据分析服务器可以根据“肺结核”来匹配“呼吸科”或者具体的医生，然后推荐给患者直接到对应科室或者医生那里做后续

的治疗。

逻辑服务器和数据分析服务器都不会直接对外提供服务，而是通过主引擎对外服务。主引擎可以提供统一服务入口，让所有服务对前端透明，还可以提供安全，过滤，流控等 API 管理功能。通过“主引擎 + 功能服务器“的微服务架构可以很好的保证系统的可靠性及易维护性。

(3) 知识纠错模块：知识纠错模块主要是为了消除知识库中互相矛盾的知识，从而提升其数据质量。逻辑服务器在做辅助诊断时，可能会因为知识库中矛盾的知识无法完成诊断，逻辑服务器会把这些矛盾的知识记录在单独的文件中。知识审核人员会定期检查包含错误知识的文件，修改这些错误知识并把新的正确的知识更新到知识库中。详细介绍参考第五章。

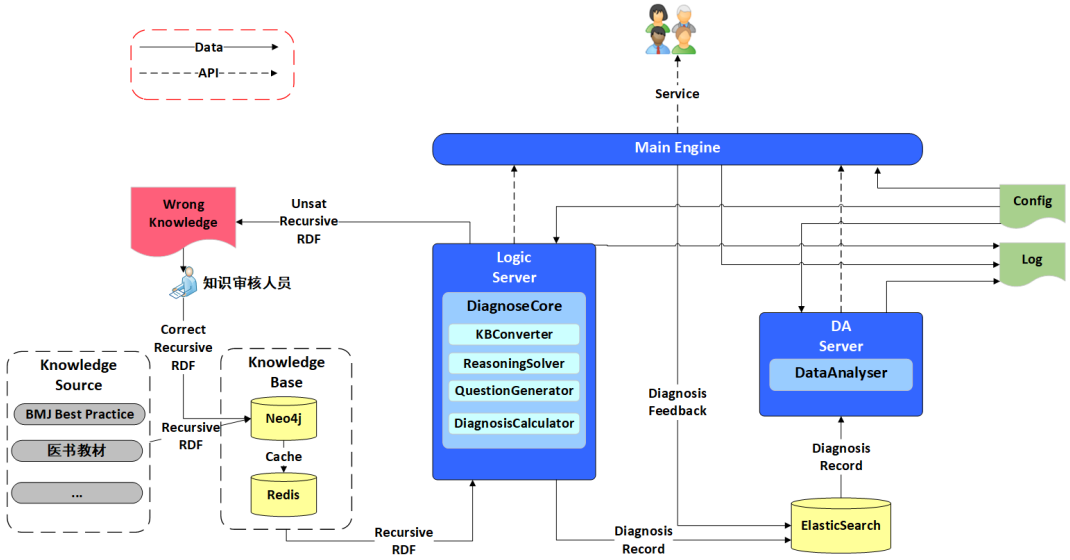


图 6–1 辅助诊断系统架构

Figure 6–1 The system architecture of computer-aided Diagnosis

6.2 医学知识库规模

因为我们的项目在持续更新，所以医学知识库的规模在不断的扩大。截止到 2019 年 12 月 20 日，我们的医学知识库中包含 47856 个实体（实体的个数即 Neo4j 中节点的数量，不是实体的种类数量），33914 个关系（关系的个数即 Neo4j 中边的数量，不是关系的种类数量）。

每种实体类型的数量及百分比如表6–1所示，虽然我们现在使用知识表示的 schema v2.0 进行知识抽取，但是知识库中仍然保存了部分使用 schema v1.0 抽取

的数据，因此现在知识库中仍然部分 schema v1.0 中的定义的实体类型，比如“检查”，“病理概念”等。通过分析实体类型数量的数据，我们发现：

(1) 嵌套实体数量占到了所有实体数量的一半，这说明我们的知识有一半左右是通过嵌套的 RDF 三元组表达的，比如”（肺癌，强导致，（咯血，或，痰中带血））”这个嵌套的三元组就会产生嵌套实体“咯血或痰中带血”。嵌套实体通过多层嵌套可以表达非常复杂的知识，但是会给知识抽取，尤其是自然语言处理的关系抽取增加很多困难。后期我们需要设计更好的知识表示 schema，尽量不使用嵌套结构，或者嵌套层数尽量少，这样才可以提高自然语言处理的准确度，加快知识抽取的速度。

(2) 知识库中有近 3000 多种疾病，10000 种症状。但是现在知识库中有很多冗余的疾病或者症状实体，比如“感冒”、“严重感冒”、“普通感冒”等等，这些疾病实体之间存在冗余，现在知识库中特异性的疾病大概不到 1000 种，不少常见疾病（比如“腰椎间盘突出”）在知识库中并不存在。因此现在的知识库在做全科问诊的时候数据量还稍显不够。后期我们需要抽取更多医学文本中的知识来扩充知识库。

(3) 知识库中有 1600 多个部位实体，而部位实体对于交互方式的优化和同近义词库的构建方面都很有帮助。比如询问患者“上腹部疼痛”，因为我们知识库中存储着“上腹部”是部位类型，则我们可以先询问患者是否腹痛，接着在询问患者腹部的哪个方位疼痛，来优化交互方式。此外，我们也可以利用一些症状的部位属性来训练同义词库。

每种关系类型的数量及百分比如表6–1所示，同样，知识库中存储了部位 schema v1.0 定义的部位关系类型，比如“作用于”，“检查发现”等。通过分析每种关系数量的数据，我们发现：

(1) 知识库中的语义网络图较为稀疏，47856 个实体却只有 33914 个关系，每个节点的度数为 1-3 度左右。这说明很多节点之间的关系还有待挖掘，后期需要更多的医学文本来补充这些缺失的关系。

(2) “强导致“关系数量是“弱导致”关系数量的 5 倍。其实医书中更多的应该是“弱导致”关系，而“强导致“比“弱导致”多说明标记人员混淆了这两种关系，后期需要对这些数据进行纠错。

(3) “与”“或”关系有近 12000 个，“与”“或”是逻辑推理的基础，也是打分的（详见4.5）重要凭证，这两种关系的数量可以决定诊断的准确率。

(4) ”否定修饰”关系只有 387 个，而“否定修饰”关系作用很大：a）通过否定修饰可以快速的排除疾病，比如“（膀胱炎，强导致，无，否定修饰，高

烧)”，通过“高烧”可以直接排除“膀胱炎”。所以“否定修饰”的数量可以决定候选疾病集合收敛的速度。b）“否定修饰”关系可以区分疾病的亚型。比如“一型糖尿病”会有”多饮多尿”症状，而“二型糖尿病”没有这个症状。我们通过“(二型糖尿病，强导致，(无，否定修饰，多尿))”这条知识，患者可能患有糖尿病的其他症状，但是没有“多尿症状”，我们就可以排除“二型糖尿病”这个亚型。因此，后期我们需要挖掘出更多的“否定修饰”关系。

表 6–1 实体类型数量

Table 6–1 The number of entity of different types

实体类型	数量	百分比	实体类型	数量	百分比	实体类型	数量	百分比
嵌套实体	24047	50.25%	检查	870	1.82%	生物	99	0.21%
症状	9976	20.85%	数据	621	1.30%	颜色	62	0.13%
疾病	3317	6.93%	事件	492	1.03%	否定词	58	0.12%
生理概念	2577	5.38%	病理概念	335	0.70%	其他	32	0.07%
修饰语	1962	4.10%	病史	327	0.68%	年龄	23	0.05%
部位	1674	3.50%	时间	259	0.54%	食品	21	0.04%
化学物质	934	1.95%	人群分类	158	0.33%	阶段	12	0.03%
总计				47856				100%

表 6–2 关系类型数量

Table 6–2 The number of relation of different types

关系名称	数量	百分比	关系名称	数量	百分比	关系名称	数量	百分比
修饰限定	9963	29.38%	同义词	683	2.01%	并发症	201	0.59%
或	9581	28.25%	条件为	669	1.97%	表现为	168	0.50%
强导致	5305	15.64%	执行检查	657	1.94%	调查病史	163	0.48%
与	2895	8.54%	检查发现	624	1.84%	然后	81	0.24%
是一种	1145	3.38%	否定修饰	387	1.14%	转变为	15	0.04%
弱导致	1074	3.17%	作用于	291	0.86%	比例为	12	0.04%
总计				33914				100%

6.3 辅助诊断内核诊断性能

6.3.1 测试集

我们抽取了 100 份住院病历来做诊断内核性能的测试。之所以选择住院病历，主要有两个原因：1) 一是因为简单的门诊病历现在大多数医院是通过医生手动书写，导致电子化的门诊病历严重缺乏。2) 二是仅有的电子门诊病历质量较差，因为门诊过程没有详细的问诊过程，也没有长时间的患者体征观察，导致很多门诊病历的诊断结果很模棱两可。比如很多门诊的诊断结果都是“xxx 待查”（如“黄疸原因待查”），而没有一个准确的诊断结果。而住院病历的诊断结果则相对详细且准确，里面包含了患者详细的症状、查体和检查指标的数据以及明确的疾病诊断（如图6–2），因此我们选择使用住院病历作为测试集。同时，为了保证集的测试集的覆盖面，测试集包含了呼吸科，泌尿科，内分泌科、消化内核、骨科、妇科等多个科室的病历。

6.3.2 诊断性能测试

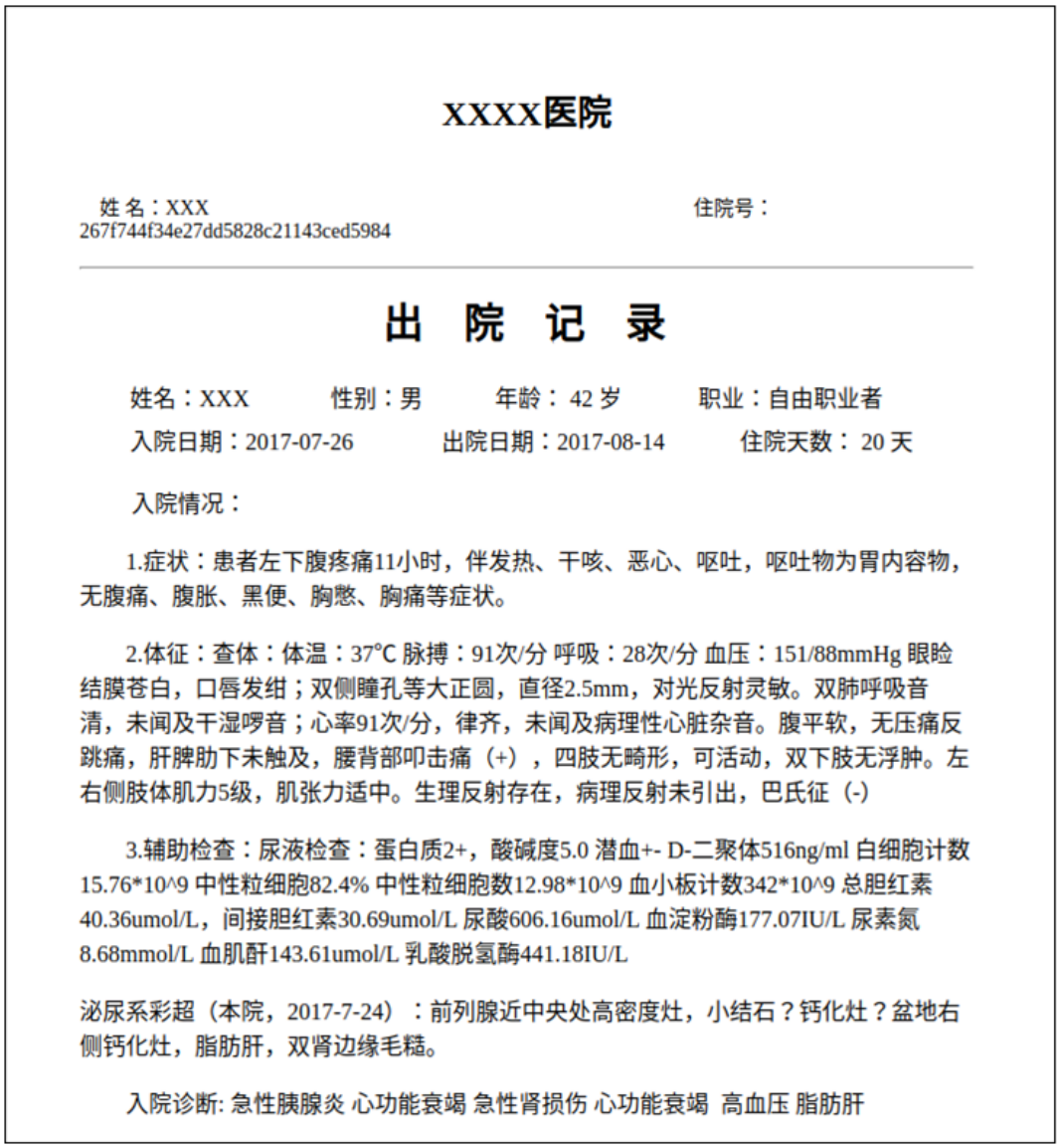
使用一百份覆盖面较广的真实住院病历，我们通过人工的方式对辅助诊断内核的诊断性能进行测试。我们分别使用疾病、部位、科室三个维度的测试数据对辅助诊断内核的性能进行评价。而对于每个维度，性能可以用三个指标来衡量：

(1) 诊断的全面性：可以使用该项指标来衡量辅诊内核的诊断结果是否全面。比如病历内的入院诊断是“{胰腺炎,高血压}”，如果辅诊内核的诊断结果只有“{胰腺炎}”。虽然这个诊断是准确的，但并不全面。我们使用召回率[68](Recall) 来衡量诊断的全面性。

(2) 诊断的准确性：可以使用该项指标来衡量诊断内核的诊断结果是否准确。比如病历内入院诊断是“{胰腺炎.高血压}”，如果辅诊内核的诊断结果为“{胃炎,流感,胰腺炎,高血压}”。虽然这个诊断结果是全面的，但是并不准确，因为其中的“胃炎、流感”是错误的诊断。我们使用准确率[68](Precision) 来衡量诊断的准确性。

(3) 诊断的区分度：因为辅助诊断的候选疾病集是按照可能性排序的，因此正确的疾病排名越靠前诊断的性能越好。比如病历内入院诊断是“{高血压}”，辅助诊断结果 1–“{高血压,流感}”肯定比结果 2–{流感,高血压} 要优异，因为相比于诊断结果 2，结果 1 中正确的诊断排名靠前。我们使用平均倒数排名[69]（Mean reciprocal rank, 简称 MRR ）来衡量诊断的区分度。

假设真实的住院病历中的入院诊断结果集为 D_{mr}, 辅助诊断内核诊断的结果集为 D_{dc}, 对于疾病、科室、部位我们分别用以下的公式来计算各自的 Recall、Precision



XXXX医院

姓 名：XXX　　住院号：
267f744f34e27dd5828c21143ced5984

出　院　记　录

姓名：XXX　　性别：男　　年龄： 42 岁　　职业：自由职业者
入院日期：2017-07-26　　出院日期：2017-08-14　　住院天数： 20 天

入院情况：

1.症状：患者左下腹疼痛11小时，伴发热、干咳、恶心、呕吐，呕吐物为胃内容物，无腹痛、腹胀、黑便、胸憋、胸痛等症状。

2.体征：查体：体温：37℃ 脉搏：91次/分 呼吸：28次/分 血压：151/88mmHg 眼睑结膜苍白，口唇发绀；双侧瞳孔等大正圆，直径2.5mm，对光反射灵敏。双肺呼吸音清，未闻及干湿啰音；心率91次/分，律齐，未闻及病理性心脏杂音。腹平软，无压痛反跳痛，肝脾肋下未触及，腰背部叩击痛（+），四肢无畸形，可活动，双下肢无浮肿。左右侧肢体肌力5级，肌张力适中。生理反射存在，病理反射未引出，巴氏征（-）

3.辅助检查：尿液检查：蛋白质2+，酸碱度5.0 潜血+- D-二聚体516ng/ml 白细胞计数15.76*10^9 中性粒细胞82.4% 中性粒细胞数12.98*10^9 血小板计数342*10^9 总胆红素40.36umol/L，间接胆红素30.69umol/L 尿酸606.16umol/L 血淀粉酶177.07IU/L 尿素氮8.68mmol/L 血肌酐143.61umol/L 乳酸脱氢酶441.18IU/L

泌尿系彩超（本院，2017-7-24）：前列腺近中央处高密度灶，小结石？钙化灶？盆地右侧钙化灶，脂肪肝，双肾边缘毛糙。

入院诊断: 急性胰腺炎 心功能衰竭 急性肾损伤 心功能衰竭 高血压 脂肪肝

图 6–2 一份住院病历样例

Figure 6–2 An example of inpatient medical record

和 MRR 值

- 疾病：

$$Recall_{disease} = \frac{|D_{mr} \cap D_{dc}|}{|D_{mr}|} \tag{6–1}$$

$$Precision_{disease} = \frac{|D_{mr} \cap D_{dc}|}{|D_{dc}|} \tag{6–2}$$

$$MRR_{disease} = \frac{\sum_{d \in D_{mr}} \frac{1}{Rank(d, D_{dc})}}{|D_{mr}|} \tag{6–3}$$

其中，$Rank(d, D_{dc})$ 表示疾病 d 在辅助内核诊断结果集中的序号，序号越低说明越靠前。召回率表示病历诊断的疾病集中被辅助诊断内核诊断正确的疾病的比例，比例越高表示内核诊断的越全面，说明病历中的多数疾病都被内核诊断正确了。准确率表示辅助诊断内核诊断的疾病集中诊断正确的疾病的比例，比例越高表示辅助诊断内核诊断的越准确，说明内核诊断出的大多是正确的疾病。而平均倒数排序表示病历中的疾病在诊断内核的疾病集中的序号倒数的平均值，平均值越高说明内核诊断的区分度越高，正确的诊断排在了诊断结果集合的前面。

- 科室

$$Recall_{department} = \frac{|Dep(D_{mr}) \cap Dep(D_{dc})|}{|Dep(D_{mr})|} \tag{6–4}$$

$$Precision_{department} = \frac{|Dep(D_{mr}) \cap Dep(D_{dc})|}{|Dep(D_{dc})|} \tag{6–5}$$

其中，$Dep(d)$ 表示疾病 d 所在科室，$Dep(D) = \{Dep(d)|d \in D\}$，表示疾病集合 D 内所有疾病所在科室的集合。科室召回率和准确率的定义和疾病的召回率和准确率类似。我们没有定义科室的平均倒数排名，因为一个科室可能包含多个疾病，因此查看病历中疾病的科室在诊断内核结果中的排名意义不大。比如病历结果为"{胃癌}"，内核的诊断结果为"{胃反酸，胃癌}"，虽然"胃反酸"是错误的诊断，但是如果考虑计算 MRR 值，"胃癌"的科室在内核结果集中仍然排在第一位，这样就无法凸显区分度。因此对于科室，我们只使用 Recall 和 Precision 来衡量内核的诊断性能，不使用 MRR 值来衡量。

- 部位

$$Recall_{position} = \frac{|Pos(D_{mr}) \cap Pos(D_{dc})|}{|Pos(D_{mr})|} \tag{6–6}$$

$$Precision_{position} = \frac{|Pos(D_{mr}) \cap Pos(D_{dc})|}{|Pos(D_{dc})|} \tag{6–7}$$

其中，$Pos(d)$ 表示疾病 d 所在部位，$Pos(D) = \{Pos(d)|d \in D\}$，表示疾病集合 D 内所有疾病所在部位的集合。部位召回率和准确率的定义和疾病的召回率和准确率类似。同样，因为一个部位可能对应多个疾病，因此也不去计算部位的 MRR 值。

为了保证诊断内核交互的友好性，我们不能问患者太多的问题。我们规定只能问三个多选问题（三次是普通导诊患者接受度最高的交互轮次），每个多选题中包含 3-5 个选项。因此，选择哪些症状询问患者会很大程度影响诊断的性能，我们在4.5.2节介绍了两种选择交互变量的方式，我们会对这两种症状选择方式对性能的影响进行测试比较。此外，我们还对市面上较为成熟的辅助诊断产品–左手医生进行了测试比较，具体的测试数据如表6–3所示。

表 6–3 辅助诊断性能测试结果

Table 6–3 The performance test result of computer-aided diagnosis core

	疾病			部位		科室	
	Recall	Precision	MRR	Recall	Precision	Recall	Precision
左手医生	0.25	**0.18** [a]	**0.21**	0.53	**0.75**	0.63	**0.92**
最大熵	0.23	0.12	0.12	0.65	0.54	0.60	0.50
最大贡献度	**0.30**	0.16	0.20	**0.65**	0.50	**0.67**	0.58

[a] 粗体表示三种方式的最优结果

考虑到医学同义词的限制，我们通过人工的方式来计算 Recall、Precision 和 MRR 的数值，表格中的数值是一百份住院病历测试结果的平均值，通过分析实验数据，可以得出以下结论：

(1) 辅助诊断内核，如果限制问诊步数，那么最大期望疾病贡献度的交互症状选择方式要优于最大熵：从表格中的数据可以看出，在疾病、部位、科室三个维度上最大期望疾病贡献度的表现都要优于最大熵。究其原因还是因为知识库中的知识比较稀疏，每个节点的平均度数只有 1-3 左右。根据公式4–11，最大熵交互症状选择策略会选择患者回答为 $True$ 的概率最接近 1/2 的症状，即 $p(s|d_1 \vee d_2 \vee ... \vee d_t)$ 最接近 1/2 的症状。我们采用 $p(s|d_1 \vee d_2 \vee ... \vee d_t) \approx \sum_{d_i \in D} p(s|d_i) \times p(d_i)$ 来计算这个概率值，因为知识库是一张很稀疏的图，大部分的症状只有 1-3 个相关的疾病，在其他不相关疾病下的条件概率都为零。因此实际情况下 $p(s|d_1 \vee d_2 \vee ... \vee d_t)$ 的值会远小于 1/2，所以最大熵策略每次都会选择相关疾病最多的症状，导致询问路径会非常固定，根本不会受到患者主述、问题答案的影响。比如，如果知识库中咳嗽、头晕相关的疾病最多，则最大熵策略每次都会选择这两个症状去询问，而不会受到患者实际症状信息的影响，因此造成内核诊断性能的下降。

(2) 辅助诊断内核的诊断全面性要高于左手医生：表格中的数据显示最大贡献度策略下内核诊断召回率在各个维度都是最高的。在测试过程我们发现，

左手医生返回的结果集很小而且很集中，它一般只返回同一个科室或部位中非常相似的 3-5 个疾病，比如“支气管感染、哮喘、肺炎”，而不会考虑其他科室或部位的疾病。而我们的辅助诊断内核因为是从医书中获取医学知识，相对来说较为系统全面，因此考虑的候选疾病会很多。比如患者有“呼吸困难”症状，左手医生只会诊断出肺部疾病，而我们辅助诊断内核中还会考虑高血压等因素，因为在医学上高血压确实会导致呼吸困难，只是概率比较小。对于这样的病历，我们的辅助诊断内核诊断的就会更加全面。

(3) 辅助诊断内核的诊断不如左手医生准确：表格中各个维度左手医生的准确率都是最优异的。主要有两个原因：

(a) 左手医生从大量真实病历出学习出症状和疾病的关联模型，它会根据患者症状从大量病历中匹配相似病历，然后输出可能性最大的疾病。这样就保证了结果的准确性，因为有很多病历已经证明了输出的疾病确实是患者症状的诊断。缺点就是如果和患者症状关联的疾病只在少量的病历中出现，则左手医生无法返回这个疾病。这也是左手诊断诊断准确但不全面的原因。

(b) 左手医生的数据量比我们的知识库的数据量大很多。我们现在的知识主要来自与《内科学》和《诊断学》两本医书，这两本医书中没有覆盖的疾病内核自然无法诊断。在实验过程中我们发现，妇科的病历内核的诊断准确性非常差，因为我们现有的知识库中缺乏妇科的医学知识。后期如果不断扩展知识库，那我们辅助诊断内核的准确率应该可以提高很多。

(4) 左手医生的诊断区分度略优于辅助诊断内核：表格中左手医生疾病的 MRR 值略高于内核。其原因是我们知识库目前缺乏同义词的支持，而且数据也不如左手医生规整。如果加入同义词，那么我们辅诊内核返回的疾病集合会小很多，则疾病排序 $Rank(d, D_{dc})$ 会小很多，最终的 MRR 值会相应提高。

6.4 错误模式分析

我们在第六章介绍了知识库中错误的模式，通过脚本可以从知识库中找出符合这些模式的知识子图，将这些错误子图交付给知识审核人员去修改错误后会提高知识库中知识的质量，从而提升诊断的准确率。我们统计了知识库中各种错误模式子图的数量, 以及每种子图包含规则的平均数量，如表6–4所示。我们人工分析了这些模式的子图，总共发现了近 500 条错误的知识，通过修正这些错误知识，有效提升了知识库中知识的质量。具体错误的分类如下：

- “弱导致”错标成了“强导致”，约 120 条。
- “强导致”错标成了“是一种”，约 100 条。
- “是一种”关系标反，约 70 条。
- “否定修饰”标记错误，约 40 条。
- “与”和“或”关系混淆，约 40 条。
- “条件为”错标成“拼接”关系，约 20 条。
- 其他错误，比如“病症”标记成“修饰语”，“并发症”标记成”同义词“等，约 110 条。

表 6–4 每种错误模式数量及其包含规则平均数量

Table 6–4 Number of each error pattern and average number of rules in it

错误模式	环	自相矛盾	永假推理	永真推理
数量	36	8	41	18
包含规则的平均数量	14	15	10	8

6.5 本章小结

本章是对整个辅助诊断系统各模块的测试及结果分析。我们首先介绍了辅助诊断系统的架构，它包括从医学知识中抽取知识构建的知识库，根据辅助诊断内核搭建的辅助诊断服务和知识纠错模块。然后我们对系统的各模块进了的测试分析，我们首先统计了知识库中各种类型实体和关系的数量进行了统计，发现知识库中有 47856 个实体和 33914 关系，包括近 3000 个疾病和 12000 个“与”“或”关系，这可以很大程度提高辅助诊断的疾病覆盖面和准确率。但是知识库中的疾病实体之间存在大量冗余，比如“感冒”、“严重感冒”和“普通感冒”等，而真正特异性的疾病实体大概不到 1000 个，而且关系中很重要的“否定修饰”关系数量较少，因此我们还需要抽取更多医学文本中的知识来补充知识库。此外从“强导致”关系数量是“弱导致”的 5 倍可以发现，现在知识库中还是存在较多错误，后期需要进行大规模的知识纠错；然后我们对辅助诊断服务的准确率进行了测试，我们通过疾病、部位、科室三个指标来衡量准确率，发现诊断内核的诊断结果非常全面，但不如左手医生准确。因为内核会系统考虑医书中的医学知识，对于罕见疾病也会纳入考虑范围。这样可以保证罕见疾病也可以诊断出来，但同时也会引入一些无关的疾病，造成了准确率的下降。而且因为现在知识库规模较小且缺乏同义词的支持，如果后期补足这两方面，辅助诊断内核诊断的准确率和区分度都会

提高；最后我们对知识库中的错误模式进行了统计，发现并修正了约 500 条错误知识，有效提升了知识库的质量。

第七章　总结与展望

7.1　全文总结

对于智慧医疗中的辅助诊断业务，业界普遍采用的是大数据 + 深度学习的技术路线，但是其可能存在罕见疾病无法覆盖、数据不可靠以及缺乏可解释性等问题。针对大数据 + 深度学习方案存在的问题，本文提出了一个新的辅助诊断解决方案：通过从医书等可信医学知识源中提取医学知识，然后将其表达成逻辑公式，通过逻辑推理来进行辅助诊断。具体来说，本文的主要贡献有：

首先，我们构建了自己的医学知识库。我们首先设计了符合医学文本中的知识结构且便于自然语言处理的知识表示 schema，然后根据 schema 中从可信的知识源（包括医书教材，权威的医学知识库等）中，采用自然语言预标记加人工审核的方式来抽取知识，从而兼顾知识抽取的速度和可靠性。最后通过将抽取的知识存储到图数据库 Neo4j 中，我们就完成了医学知识库的构建。通过实验分析，现有的知识库包含近 48000 个医学实体和 34000 个关系，这样的量级完全可以满足简单的问诊需求，而且关系中有 1/3 是逻辑上的“与”关系和“或”关系，这样就可以充分发挥逻辑推理的作用。

其次，我们设计并实现了辅助诊断的内核。辅助诊断内核可以根据知识库中的知识和患者的症状信息来完成对患者疾病的预测。内核中包括知识库转换器、推理引擎、问题生成器和诊断计算器。它们的工作流程是：首先知识库转换器根据患者的症状信息从知识库中提取相关知识并编码成命题逻辑公式，推理引擎会组合这些逻辑公式来构建可满足性问题的实例并使用 SMT 求解器求解，求解过程中推理引擎可能会需要知道更多患者的症状信息，推理引擎会将这些症状交给问题生成器去生成特定的问题与患者交互，得到答案后推理引擎会进行迭代求解。推理引擎在求解结束后会返回候选疾病集合，诊断计算器会根据患者的症状信息来给候选疾病打分并排序，最后将排好序的候选疾病集合返回。通过真实病历的测试，我们发现辅助诊断内核的诊断结果非常全面，拥有很高的召回率。但也包含了不少错误的诊断，准确性相对较低。但如果后期不断扩充知识库规模并引入同义词库，内核诊断的准确性会提升很多，完全可以满足实际应用的需求。

然后，我们提出了知识纠错的方案。因为知识抽取的过程不可避免的会引入错误，所以必须对知识库中的知识进行纠错才能保证知识库的可靠性。我们采用可满足性问题中的不可满足核来发现知识库中互相矛盾的知识，具体就是辅助诊断内核在使用 SMT 求解可满足性问题的过程中，如果遇到不可满足的情况则会记

录最小不可满足核。我们会将不可满足核解码回知识库中的知识，然后知识审核人员会分析这些互相矛盾的知识找到错误并修改，从而完成错误知识的纠正。通过分析已有的错误我们发现了错误的模式，并通过这些模式发现并修正了约 500 条错误知识。

最后，我们根据知识库，辅助诊断内核，知识纠错模块搭建了完整的辅助诊断系统。我们将辅助诊断系统安装在团队设计并生产的诊断舱中，已经在合作医院中部署并测试。

7.2 未来展望

本文提出了新的辅助诊断解决方案，并且已经设计并实现了完整的辅助诊断系统，但是由于研究时间的限制，本文在各方面仍存在一些未来需要进行的工作：

首先是知识层次的深入。当前阶段，我们只考虑问诊层面的知识，并根据该层面的知识结构设计了知识表示的 schema。它的优点在于知识结构简单而且便于抽取，缺点就是该层次的医学知识距离疾病的核心机制很远，因此表达的信息比较模糊，只能大概提高诊断对某一类疾病的怀疑程度，极少数可以直接确诊或者排除疾病。这也是为什么目前我们疾病预测准确度较低的原因。后期我们会考虑抽取体格检查、辅助检查和实验室检查的知识，通过严格的检验指标来完成对疾病更为准确的预测。

其次是知识库规模的扩充。目前知识库中包含了 48000 个医学实体和 34000 个关系，但是实体之间包含大量冗余，核心的关系（比如“否定修饰”关系）较少，所以在做全科的辅助诊断时会显得力不从心。因此我们之后需要不断扩充知识库的规模，从而提高辅助诊断的覆盖面。同时需要引入同义词库，删除知识库中的冗余知识，让知识库中的知识更加规整。

然后是诊断模型的优化。当前阶段，我们主要使用命题逻辑来建模医学诊断，并使用求解布尔可满足性问题的方式来进行辅助诊断。但是强逻辑的特性与医学诊断存在一定的偏差。比如在命题逻辑中，如果“胃炎导致呕吐”，那么没有“呕吐”就一定没有“胃炎”，但是真实情况是有一些“慢性胃炎”确实不会引起”呕吐”。因此强逻辑的种非真即假的推理方式在医学诊断上稍显武断。之后我们会考虑使用概率逻辑、模糊逻辑、马尔科夫逻辑网等对强逻辑进行“软化”的方式为医学诊断建模。

最后是知识的进一步纠错。因为我们现在主要是对知识库中互相矛盾的知识进行纠错，但是有一些知识虽然错误但是并没有与其他知识矛盾，之后我们需要大量优质的病历测试来完成这部分错误知识的纠正。

参考文献

[1] TROCHIM W M, DONNELLY J P. Research methods knowledge base[M]. [S.l.]: Atomic Dog Publishing Cincinnati, OH, 2001.

[2] MASARIE JR F E, MILLER R A, MYERS J D. INTERNIST-I properties: Representing common sense and good medical practice in a computerized medical knowledge base[J]. Computers and Biomedical Research, 1985, 18(5): 458-479.

[3] DONNELLY K. SNOMED-CT: The advanced terminology and coding system for eHealth[J]. Studies in health technology and informatics, 2006, 121: 279.

[4] COTE R A, ROBBOY S. Progress in medical information management: Systematized Nomenclature of Medicine (SNOMED)[J]. Jama, 1980, 243(8): 756-762.

[5] PLUNKETT R J. Standard nomenclature of diseases and operations[M]. [S.l.]: American Medical Association, 1952.

[6] SPACKMAN K A, CAMPBELL K E, CÔTÉ R A. SNOMED RT: a reference terminology for health care.[C]//Proceedings of the AMIA annual fall symposium. [S.l. : s.n.], 1997: 640.

[7] SACKETT D L. Evidence-based medicine[C]//Seminars in perinatology: vol. 21: 1. [S.l. : s.n.], 1997: 3-5.

[8] YANG W, JUN L I, JINGLI L I. The Preliminary Study on the Construction of Medical Device Naming System in China[J]. Chinese Journal of Medical Instrumentation, 2015.

[9] ORGANIZATION W H, et al. Manual of the international statistical classification of disease, injuries, and causes of death. Based on the recommendations of the eighth revision conference, 1965, and adopted by the Nineteenth World Health Assembly[J]., 1967.

[10] HUTCHINSON J M, PATRICK D M, MARRA F, et al. Measurement of antibiotic consumption: A practical guide to the use of the Anatomical Therapeutic Chemical classification and Defined Daily Dose system methodology in Canada[J]. Canadian Journal of Infectious Diseases and Medical Microbiology, 2004, 15(1): 29-35.

[11] LIPSCOMB C E. Medical subject headings (MeSH)[J]. Bulletin of the Medical Library Association, 2000, 88(3): 265.

[12] AUER S, BIZER C, KOBILAROV G, et al. Dbpedia: A nucleus for a web of open data[G]//The semantic web. [S.l.]: Springer, 2007: 722-735.

[13] YUANZHUO W, YANTAO J, ZEYA Z, et al. OpenKG-Knowledge Computing Engine in the Era of Network Big Data[J]. Communications of the Chinese Computer Federation, 2014, 10(11): 30-35.

[14] XU B, XU Y, LIANG J, et al. Cn-dbpedia: A never-ending chinese knowledge extraction system[C]//International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems. [S.l. : s.n.], 2017: 428-438.

[15] NAVIGLI R, PONZETTO S P. BabelNet: Building a very large multilingual semantic network[C]//Proceedings of the 48th annual meeting of the association for computational linguistics. [S.l. : s.n.], 2010: 216-225.

[16] DOI K. Computer-aided diagnosis in medical imaging: historical review, current status and future potential[J]. Computerized medical imaging and graphics, 2007, 31(4-5): 198-211.

[17] DE DOMBAL F, LEAPER D, STANILAND J R, et al. Computer-aided diagnosis of acute abdominal pain[J]. Br Med J, 1972, 2(5804): 9-13.

[18] CHEN Y, ARGENTINIS J E, WEBER G. IBM Watson: how cognitive computing can be applied to big data challenges in life sciences research[J]. Clinical therapeutics, 2016, 38(4): 688-701.

[19] ZAUDERER M G, GUCALP A, EPSTEIN A S, et al. Piloting IBM Watson Oncology within Memorial Sloan Kettering' s regional network.[Z]. 2014.

[20] VÖLKEL M, KRÖTZSCH M, VRANDECIC D, et al. Semantic wikipedia[C]//Proceedings of the 15th international conference on World Wide Web. [S.l. : s.n.], 2006: 585-594.

[21] MILLER G A. WordNet: An electronic lexical database[M]. [S.l.]: MIT press, 1998.

[22] SUCHANEK F M, KASNECI G, WEIKUM G. Yago: A large ontology from wikipedia and wordnet[J]. Web Semantics: Science, Services and Agents on the World Wide Web, 2008, 6(3): 203-217.

[23] RAMOS J, et al. Using tf-idf to determine word relevance in document queries[C]//Proceedings of the first instructional conference on machine learning: vol. 242. [S.l. : s.n.], 2003: 133-142.

[24] HUANG P S, HE X, GAO J, et al. Learning deep structured semantic models for web search using clickthrough data[C]//Proceedings of the 22nd ACM international conference on Information & Knowledge Management. [S.l. : s.n.], 2013: 2333-2338.

[25] GIARRATANO J C, RILEY G. Expert systems[M]. [S.l.]: PWS publishing co., 1998.

[26] SHORTLIFFE E. Computer-based medical consultations: MYCIN[M]. [S.l.]: Elsevier, 2012.

[27] FRIEDMAN N, GEIGER D, GOLDSZMIDT M. Bayesian network classifiers[J]. Machine learning, 1997, 29(2-3): 131-163.

[28] GEVAERT O, SMET F D, TIMMERMAN D, et al. Predicting the prognosis of breast cancer by integrating clinical and microarray data with Bayesian networks[J]. Bioinformatics, 2006, 22(14): e184-e190.

[29] GUO Y, BAI G, HU Y. Using bayes network for prediction of type-2 diabetes[C]// 2012 International Conference for Internet Technology and Secured Transactions. [S.l. : s.n.], 2012: 471-472.

[30] PANG B, ZHANG D, LI N, et al. Computerized tongue diagnosis based on Bayesian networks[J]. IEEE Transactions on biomedical engineering, 2004, 51(10): 1803-1810.

[31] COOPER G F. The computational complexity of probabilistic inference using Bayesian belief networks[J]. Artificial intelligence, 1990, 42(2-3): 393-405.

[32] CHICKERING D M, GEIGER D, HECKERMAN D, et al. Learning Bayesian networks is NP-hard[R]. [S.l.]: Citeseer, 1994.

[33] BAKATOR M, RADOSAV D. Deep learning and medical diagnosis: A review of literature[J]. Multimodal Technologies and Interaction, 2018, 2(3): 47.

[34] KRIZHEVSKY A, SUTSKEVER I, HINTON G E. Imagenet classification with deep convolutional neural networks[C]//Advances in neural information processing systems. [S.l. : s.n.], 2012: 1097-1105.

[35] SALAKHUTDINOV R, HINTON G. Deep boltzmann machines[C]//Artificial intelligence and statistics. [S.l. : s.n.], 2009: 448-455.

[36] LIANG Z, ZHANG G, HUANG J X, et al. Deep learning for healthcare decision making with EMRs[C]//2014 IEEE International Conference on Bioinformatics and Biomedicine (BIBM). [S.l. : s.n.], 2014: 556-559.

[37] PHUONG N H, KREINOVICH V. Fuzzy logic and its applications in medicine[J]. International journal of medical informatics, 2001, 62(2-3): 165-173.

[38] NILSSON N J. Probabilistic logic[J]. Artificial intelligence, 1986, 28(1): 71-87.

[39] RICHARDSON M, DOMINGOS P. Markov logic networks[J]. Machine learning, 2006, 62(1-2): 107-136.

[40] COLLOBERT R, WESTON J, BOTTOU L, et al. Natural language processing (almost) from scratch[J]. Journal of machine learning research, 2011, 12(Aug): 2493-2537.

[41] LEVESQUE H J. Knowledge representation and reasoning[J]. Annual review of computer science, 1986, 1(1): 255-287.

[42] SOWA J F, et al. Knowledge representation: logical, philosophical, and computational foundations[M]. [S.l.]: Brooks/Cole Pacific Grove, CA, 2000.

[43] AIKINS J S. Prototypes and Production Rules: A Knowledge Representation for Computer Consultations.[R]. [S.l.]: STANFORD UNIV CA DEPT OF COMPUTER SCIENCE, 1980.

[44] DAVIS R, SHROBE H, SZOLOVITS P. What is a knowledge representation?[J]. AI magazine, 1993, 14(1): 17-17.

[45] PETERS S, SHROBE H E. Using semantic networks for knowledge representation in an intelligent environment[C]//Proceedings of the First IEEE International Conference on Pervasive Computing and Communications, 2003.(PerCom 2003). [S.l. : s.n.], 2003: 323-329.

[46] SOWA J F. Semantic networks[J]., 1987.

[47] MILLER E. An introduction to the resource description framework[J]. Bulletin of the American Society for Information Science and Technology, 1998, 25(1): 15-19.

[48] DONG Z, DONG Q, HAO C. HowNet and the Computation of Meaning[J]., 2006.

[49] VAN BRUGGEN R. Learning Neo4j[M]. [S.l.]: Packt Publishing Ltd, 2014.

[50] MCCANDLESS M, HATCHER E, GOSPODNETIC O. Lucene in action: covers Apache Lucene 3.0[M]. [S.l.]: Manning Publications Co., 2010.

[51] CARLSON J L. Redis in action[M]. [S.l.]: Manning Shelter Island, 2013.

[52] BÜNING H K, LETTMANN T. Propositional logic: deduction and algorithms[M]. [S.l.]: Cambridge University Press, 1999.

[53] SCHAEFER T J. The complexity of satisfiability problems[C]//Proceedings of the tenth annual ACM symposium on Theory of computing. [S.l. : s.n.], 1978: 216-226.

[54] BARRETT C, TINELLI C. Satisfiability modulo theories[G]//Handbook of Model Checking. [S.l.]: Springer, 2018: 305-343.

[55] OPPEN D C. Complexity, convexity and combinations of theories[J]. Theoretical computer science, 1980, 12(3): 291-302.

[56] DEDEKIND R. Theory of algebraic integers[M]. [S.l.]: Cambridge University Press, 1996.

[57] BOFILL M, NIEUWENHUIS R, OLIVERAS A, et al. The barcelogic SMT solver[C]//International Conference on Computer Aided Verification. [S.l. : s.n.], 2008: 294-298.

[58] JHA S, LIMAYE R, SESHIA S A. Beaver: Engineering an efficient smt solver for bit-vector arithmetic[C]//International Conference on Computer Aided Verification. [S.l. : s.n.], 2009: 668-674.

[59] DUTERTRE B, DE MOURA L. The yices smt solver[J]. Tool paper at http://yices. csl. sri. com/tool-paper. pdf, 2006, 2(2): 1-2.

[60] DE MOURA L, BJØRNER N. Z3: An efficient SMT solver[C]//International conference on Tools and Algorithms for the Construction and Analysis of Systems. [S.l. : s.n.], 2008: 337-340.

[61] LYNCE I, MARQUES-SILVA J P. On computing minimum unsatisfiable cores[J]., 2004.

[62] ZHU H, XIE R, LIU Z, et al. Iterative Entity Alignment via Joint Knowledge Embeddings.[C]//IJCAI. [S.l. : s.n.], 2017: 4258-4264.

[63] QUAN H, SUNDARARAJAN V, HALFON P, et al. Coding algorithms for defining comorbidities in ICD-9-CM and ICD-10 administrative data[J]. Medical care, 2005: 1130-1139.

[64] ZELENKO D, AONE C, RICHARDELLA A. Kernel methods for relation extraction[J]. Journal of machine learning research, 2003, 3(Feb): 1083-1106.

[65] PNUELI A. The temporal logic of programs[C]//18th Annual Symposium on Foundations of Computer Science (sfcs 1977). [S.l. : s.n.], 1977: 46-57.

[66] SHANNON C E. A mathematical theory of communication[J]. Bell system technical journal, 1948, 27(3): 379-423.

[67] NADAREISHVILI I, MITRA R, MCLARTY M, et al. Microservice architecture: aligning principles, practices, and culture[M]. [S.l.]: " O'Reilly Media, Inc.", 2016.

[68] POWERS D M. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation[J]., 2011.

[69] CRASWELL N. Mean reciprocal rank[J]. Encyclopedia of Database Systems, 2009: 1703-1703.