自然语言处理导论

张奇 桂韬 黄萱菁

2023 年 2 月 6 日

.

数学符号

数与数组

α	标量
$\boldsymbol{\alpha}$	向量
$\boldsymbol{A}$	矩阵
$\mathbf{A}$	张量
$\boldsymbol{I}_n$	n 行 n 列单位矩阵
$\boldsymbol{v}_w$	单词 w 的分布式向量表示
$\boldsymbol{e}_w$	单词 w 的独热向量表示：[0,0,...,1,0,...0]，w 下标处元素为 1

索引

α_i	向量 $\boldsymbol{\alpha}$ 中索引 i 处的元素
α_{-i}	向量 $\boldsymbol{\alpha}$ 中除索引 i 之外的元素
$w_{i:j}$	序列 w 中从第 i 个元素到第 j 个元素组成的片段或子序列
A_{ij}	矩阵 $\boldsymbol{A}$ 中第 i 行、第 j 列处的元素
$\boldsymbol{A}_{i:}$	矩阵 $\boldsymbol{A}$ 中第 i 行
$\boldsymbol{A}_{:j}$	矩阵 $\boldsymbol{A}$ 中第 j 列
$\boldsymbol{A}_{ijk}$	三维张量 $\mathbf{A}$ 中索引为 (i,j,k) 处元素
$\mathbf{A}_{::i}$	三维张量 $\mathbf{A}$ 中的一个二维切片

集合

$\mathbb{A}$	集合
$\mathbb{R}$	实数集合
$0, 1$	含 0 和 1 的二值集合
$0, 1, ..., n$	含 0 和 n 的正整数的集合
$[a, b]$	a 到 b 的实数闭区间
$(a, b]$	a 到 b 的实数左开右闭区间

线性代数

$\boldsymbol{A}^\top$	矩阵 $\boldsymbol{A}$ 的转置
$\boldsymbol{A} \odot \boldsymbol{B}$	矩阵 $\boldsymbol{A}$ 与矩阵 $\boldsymbol{B}$ 的 Hardamard 乘积
$\det \boldsymbol{A}^\top$	矩阵 $\boldsymbol{A}$ 的行列式
$[\boldsymbol{x}; \boldsymbol{y}]$	向量 $\boldsymbol{x}$ 与 $\boldsymbol{y}$ 的拼接
$[\boldsymbol{U}; \boldsymbol{V}]$	矩阵 $\boldsymbol{A}$ 与 $\boldsymbol{V}$ 沿行向量拼接
$\boldsymbol{x} \cdot \boldsymbol{y}$ 或 $\boldsymbol{x}^\top \boldsymbol{y}$	向量 $\boldsymbol{x}$ 与 $\boldsymbol{y}$ 的点积

微积分

$\frac{\mathrm{d}y}{\mathrm{d}x}$	y 对 x 的导数
$\frac{\partial y}{\partial x}$	y 对 x 的偏导数
$\nabla_{\boldsymbol{x}} y$	y 对向量 $\boldsymbol{x}$ 的梯度
$\nabla_{\boldsymbol{X}} y$	y 对矩阵 $\boldsymbol{X}$ 的梯度
$\nabla_{\mathbf{X}} y$	y 对张量 $\mathbf{X}$ 的梯度

概率与信息论

$a \perp b$	随机变量 a 与 b 独立
$a \perp b \mid c$	随机变量 a 与 b 关于 c 条件独立
$P(a)$	离散变量概率分布
$p(a)$	连续变量概率分布
$a \sim P$	随机变量 a 服从分布 P
$\mathbb{E}_{x \sim P}[f(x)]$ 或 $\mathbb{E}[f(x)]$	$f(x)$ 在分布 $P(x)$ 下的期望
$\mathrm{Var}(f(x))$	$f(x)$ 在分布 $P(x)$ 下的方差
$\mathrm{Cov}(f(x), g(x))$	$f(x)$ 与 $g(x)$ 在分布 $P(x)$ 下的协方差
$H(f(x))$	随机变量 x 的信息熵
$D_{KL}(P \parallel Q)$	概率分布 P 与 Q 的 KL 散度
$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	均值为 $\boldsymbol{\mu}$、协方差为 $\boldsymbol{\Sigma}$ 的高斯分布

数据与概率分布

$\mathbb{X}$ 或 $\mathbb{D}$	数据集
$\boldsymbol{x}^{(i)}$	数据集中第 i 个样本（输入）
$\boldsymbol{y}^{(i)}$ 或 $y^{(i)}$	第 i 个样本 $\boldsymbol{x}^{(i)}$ 的标签（输出）

函数

$f : \mathcal{A} \longrightarrow \mathcal{B}$	由定义域 $\mathcal{A}$ 到值域 $\mathcal{B}$ 的函数（映射）f
$f \circ g$	f 与 g 的复合函数
$f(\boldsymbol{x}; \boldsymbol{\theta})$	由参数 $\boldsymbol{\theta}$ 定义的关于 $\boldsymbol{x}$ 的函数（也可以直接写作 $f(\boldsymbol{x})$，省略 $\boldsymbol{\theta}$）
$\log x$	x 的自然对数函数
$\sigma(x)$	Sigmoid 函数 $\dfrac{1}{1+exp(-x)}$
$\|\|\boldsymbol{x}\|\|_p$	$\boldsymbol{x}$ 的 L^p 范数
$\|\|\boldsymbol{x}\|\|$	$\boldsymbol{x}$ 的 L^2 范数
$\mathbf{1}^{\text{condition}}$	条件指示函数：如果 condition 为真，则值为 1；否则值为 0

本书中常用写法

- 给定词表 $\mathbb{V}$，其大小为 $|\mathbb{V}|$
- 序列 $x = x_1, x_2, ..., x_n$ 中第 i 个单词 x_i 的词向量 $\boldsymbol{v}_{x_i}$
- 损失函数 $\mathcal{L}$ 为负对数似然函数：$\mathcal{L}(\boldsymbol{\theta}) = -\sum_{(x,y)} \log P(y|x_1...x_n)$
- 算法的空间复杂度为 $\mathcal{O}(mn)$

目 录

10. 智能问答

智能问答（Questing Answering，QA）旨在自动回答用户以自然语言方式提出的各类问题。自1950 年图灵测试（Turing Test）提出以来[1]，以自然语言进行人机交互就成为人们不断追求和奋斗的目标。这其中如何自动回答用户的各类问题，也成为了自然语言处理领域的研究热点和难点。受到当前技术水平的限制，根据候选答案的来源不同以及问题种类的不同，问答系统所采用的技术手段也不尽相同。近年来，随着深度学习方法的不断进步，特别是超大规模预训练模型的发展，智能问答研究成果不断涌现，各类型问题的回答效果不断提高。智能问答也逐渐成为了对话助手、智能客服、搜索引擎等系统中必不可少的组成部分。

本章首先介绍智能问答的基本概念和发展历程，在此基础上按照问答答案来源的不同，分别介绍阅读理解、表格问答、社区问答以及开放领域问答相关算法。

10.1 智能问答概述

智能问答的目标是针对用户输入的自然语言方式描述的问题自动给出答案。20 世纪 60 年代开始自然语言处理领域就开始了相关研究，BASEBALL 系统[2] 试图通过解析用户自然语言提出的关于美国棒球联赛的问题，通过结构化数据产生答案。1999 年 TREC（Text Retrieval Conference）举办了第一届开放领域问答评测任务 TREC-8[3]，推动了智能问答的快速发展。此后，随着搜索引擎将各种智能问答算法应用于处理用户输入的自然语言查询，使得智能问答相关算法加速发展。如图10.1所示，用户在搜索引擎中输入“珠穆朗玛峰海拔多少米？”，系统可以直接给出“88848.68米”的答案。

图 10.1　搜索引擎中智能回答应用样例（来源：百度搜索）

近年来，随着智能终端的普及以及语音识别技术的进步，智能对话助手也逐渐深入大众生活，这其中也涉及大量用户以自然语言提出的各种类型问题，对智能问答的技术需求也更加急迫。同时，得益于互联网特别是 Web 2.0 的高速发展，大量问答知识以文本、表格、问题答案对等形式存在于互联网中。智能问答作为搜索引擎、对话助手、智能客服等系统的核心模块，受到学术界和企业界越来越多的关注，创新的智能问答研究成果也伴随着深度学习的发展不断涌现。

在本节中，我们将介绍智能问答的发展过程以及各类智能问答系统。

10.1.1 智能问答发展历程

20 世纪 60 年代开始自然语言处理领域就开始从数据库的自然语言接口（Natural Language Interface for Database）角度开启了智能问答的研究。针对特定领域的结构化数据，通过分析用户的自然语言查询输入，并将其转换为结构化数据库查询，从而得到相应的答案。除了针对美国棒球联赛的 BASEBALL 系统之外，LUNAR[4] 系统则针对月球地质学家访问 NASA 数据库需求，构建了自然语言查询接口，可以回答如下问题：

(1) Which samples are breccias?

(2) What is the average analysis of Ir in rock SI0055?

(3) How much Titanium does S10017 contain?

20 世纪 70 年代到 80 年代，绝大部分智能问答研究仍然集中于受限领域。SHRDLU 系统[5] 能够接受自然语言的指令，指挥虚拟积木世界中的机器人移动玩具积木块。Chat-80 系统[6] 将英语转换为 Prolog 语言，根据知识库回答关于国家、城市、河流等世界地理知识问题。UNIX Consultant 系统[7] 可以支持用户使用自然语言，完成对 UNIX 操作系统中特定任务的操作。这期间也有一些工作从自然语言理解的角度，试图理解文本内容，从而直接回答用户问题。1977 年 Lehnert 发布的 QUALM 系统[8] 提出了阅读理解概念，针对不同的问题类型使用不同的策略从文章中寻找答案。

20 世纪 90 年代开始，随着互联网的发展和统计机器学习技术的不断进步，更实用的开放领域问答系统开始兴起。1993 年由麻省理工学院 Boris Katz 教授带领 InfoLab 实验室开发的第一个基于互联网的智能问答系统 START [9–11] 上线，开启了以整个互联网为知识库的开放领域问答时代。1999 年开放领域问答评测任务 Trec-8 的提出，更进一步推动了开放领域问答的研究。TREC-8 评测要求根据给定的数据集合，回答事实性的短答案问题，要求系统返回答案和对应的文档编号。评测的问题相对比较简单，例如：“How many calories are there in a Big Mac?”。IBM 构建的 Watson 系统[12, 13] 参加了多次 TREC 智能问答评测，并于 2011 年参加了美国电视问答比赛界面 Jeopardy!，并战胜了人类冠军。2017 年搜狗问答机器人汪仔在江苏卫视问答节目《一站到底》中也战胜了人类选手取得最终胜利。搜狗问答机器人所使用的问答系统框架如图10.2 所示，这也代表了非端到端问答系统的典型结构。

随着深度学习方法在自然语言处理领域取得突破，阅读理解、社区问答、表格问答、知识图谱问答等研究领域在 2012 年以后也都取得了长足的进步。特别是 2016 年斯坦福大学发布的阅读

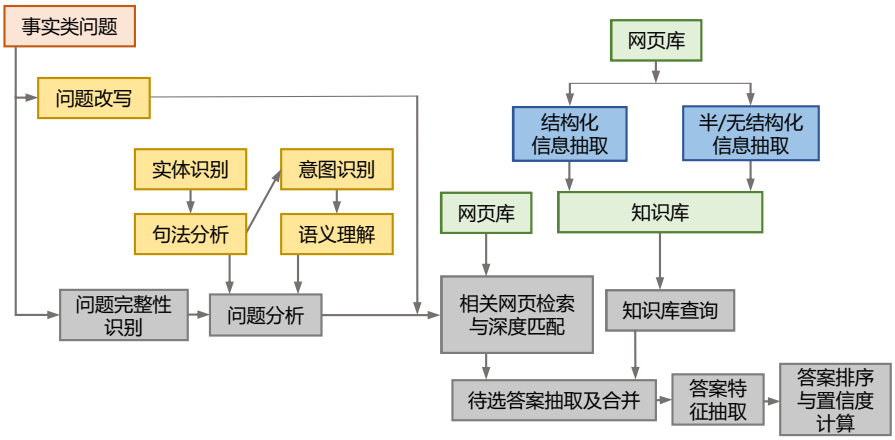

图 10.2 搜狗问答机器人汪仔问答系统结构图[14]

理解数据集 SQuAD[15]，极大的推动了阅读理解的研究进展。SQuAD 包含了 10 万个高质量的问题和对应的答案，为深度学习方法提供了充足的训练语料，同时也初步验证了在训练数据充足的情况下，深度学习算法甚至可以取得超过人类的性能。2017 年哈尔滨工业大学讯飞联合实验室也发布了中文机器阅读理解评测（CMRC）。2018 年百度发布了中文阅读理解数据集 DuReader，包含 20 万个问题、100 万个文档和超过 42 万个人工给出的答案。此后大量智能问答相关评测集合相继发布，包括：多步推理阅读理解评测 HotpotQA[16]、对话问答数据集 CoQA[17]、复杂序列问答 SequenceQA[18]、自由形式表格问答 FeTaQA[19] 等。

10.1.2 智能问答主要类型

受到现有机器学习方法和自然语言处理算法能力的限制，目前还没有通用方法可以回答所有类型的问题，根据问题类型以及答案来源的不同，需要采用不同的方法进行解决。根据参考文献[20]中给出的分类，可以将问题大致分为七类：事实类（Factoid）、是非类（Yes/No）、定义类（Definition）、列表类（List）、比较类（Comparison）、意见类（Opinion）以及指导类（How-to）。表10.1给出了各问题类型的问答样例。根据上述不同类型问题的特点，再结合知识库的来源，可以将智能问答分为五大类：阅读理解、表格问答、社区问答、知识图谱问答和开放领域问答。

阅读理解（Machine Reading Comprehension，MRC），也称**机器阅读理解**，是指根据给定的一篇或多篇文本内容回答给定的问题。按照答案类型的不同，还可以进一步细分为完型填空、多项选择、片段抽取和自由作答四种形式。相关工作在 2016 年 SQuAD 语料集发布之前，就已经开始了相关研究。1999 年 Deep Read 阅读理解系统[21] 在 Remedia 语料集上进行了测试。2004 年香港中文大学发布了中英双语阅读理解语料集 ChungHwa[22]。相关算法算法将在本章中进行介绍。

表格问答（Table based Question Answering, TBQA）是指根据给定的表格数据生成问题答案。表格通常由 M 行 N 列的数据组成，第一行中的 N 个单元格是表头信息。2015 年由斯坦福大学发布

表 10.1 各问题类型问答样例

问题类型	问题	答案
事实类	复旦大学成立于那年?	1905 年
是非类	复旦大学在上海吗?	是
定义类	什么是自然语言处理?	实现人与计算机之间用自然语言进行有效通信的各种理论和方法
列表类	复旦大学有哪几个校区?	邯郸、枫林、张江、江湾
比较类	上海和北京哪个人口多?	根据第七次全国人口普查北京市人口数为 2189.31 万，上海市人口数为 2487.09 万，上海人口比北京多
意见类	你觉得上海哪些最值得去的地方?	上海这座被称为“魔都”的城市有很多知名的经典，我觉得最值得去的地方包括外滩、东方明珠、城隍庙、豫园、田子坊等
指导类	如何从虹桥机场到浦东机场?	可以有以下三种方式：1）机场大巴直达：机场一线；2）乘坐地铁 2 号线直达；3）乘坐地铁 2 号线到龙阳路站换乘磁悬浮列车

的表格问答数据集 WikiTableQuestions[23] 中给定了问题和与其对应的数据表格。2022 年 FeTaQA 数据集[19] 进一步升级，要求根据表格和问题生成需要归纳和推理得到的句子形式的答案。相关算法算法将在本章中进行介绍。

社区问答（Community Question Answering，CQA）是指根据社区问答等来源获得的 < 问题，答案 > 对进行问题回答。随着社会媒体的高速发展，以知乎、Quora 等为代表的问答类网站提供了用户发布问题和回答问题的渠道，并提供了用户点赞、关注、评论等多种交互方式。这些问答数据中包含了大量人工凝练和总结的高质量答案，可以很好的回答大量其他方法很难自动回答的比较类、意见类以及指导类问题。社区问答根据用户输入的问题，从已有的问答对中寻找语义最相关的问答，并将所对应的答案返回给用户。社区问答最核心的技术问题就是计算用户输入问题和已有问题之间的语义相关性，以及用户输入问题和答案之间的语义相关性。相关算法算法将在本章中进行介绍。

知识图谱问答（Knowledge based Question Answering，KBQA）是指根据给定知识图谱生成问题的答案。知识图谱采用图的方法对知识进行结构化表示，图的节点表示实体，图的边表示实体之间的关系。利用信息抽取、实体融合等技术可以从大规模的自然语言文本、表格等数据中构建大规模的知识图谱。在此基础上，根据用户的问题，在图谱中根据实体和关系并利用推理机制可以进行问题回答。由于知识图谱问答与知识图谱构建、表示与推理紧密关联，因此相关算法将在本书第 12 章知识图谱部分进行介绍。

开放领域问答（Open-domain Question Answering，ODQA）是通过大规模文档集合回答不限定领域的事实性问题。通常情况下开放领域问答由答案段落检索和答案抽取两个部分组成。IBM Watson

系统[13] 就是采用了开放领域问答架构。通过将问题分解并生成检索词，之后根据检索词得到相关段落，并对段落进行评判，在此基础上通过段落抽取最终答案。目前的开放领域问答系统很多是结合搜索与阅读理解，在通过传统搜索或语义搜索得到候选篇章后，利用阅读理解技术获得最终答案。目前的搜索引擎中也大量使用该项技术，提升用户的搜索体验。相关算法将在本章中进行介绍。

10.2 阅读理解

机器阅读理解目标就是根据给定的一篇或多篇文本内容回答给定的问题。这个任务对于算法的自然语言理解能力和推理能力是很大的考验。虽然阅读理解任务研究在 1999 年就已经开始，但是受到基于特征的机器学习算法能力的限制，以及缺乏大规模的标注数据，阅读理解的研究进展相对较慢。2016 年斯坦福大学推出了 SQuAD 语料集，大幅度推动了人们对阅读理解研究的关注，大量深度神经网络的模型不断提出。2018 年 1 月，微软亚洲研究院提出的 R-Net 算法[24] 率先在 SQuAD 的精准匹配指标上首次超越人类。2018 年基于预训练语言模型 BERT 的阅读理解算法在该任务上取得了大幅度进展。近年来，各类型阅读理解任务不断推出，阅读理解任务也称为了验证大规模预训练模型的标准任务之一。

图10.3给出了阅读理解样例。对于问题“复旦大学江湾校区位于上海哪个区？”，通过选择候选答案以及答案抽取，最终返回从文章中抽取得到的片段“杨浦区”作为答案。

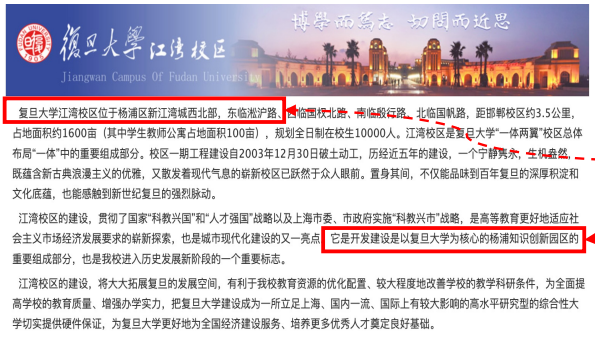

图 10.3　阅读理解样例

目前绝大多数的机器阅读理解算法都采用有监督方法，将阅读理解任务转换为分类问题，因为不同的答案形式需要所采用的方法有所不同，由此可以将阅读理问题分为以下四大类：

(1) 完型填空：对于一段文本中的某个句子，其中缺少某个单词或短语，需要算法根据其他的文本内容预所缺失部分。

(2) 多项选择：给定文本内容，问题及对应的若干选项，需要从选项中选择出一个或多个正确的选项构成答案。

(3) 片段抽取：给定文本内容和问题，从文本中抽取单词、短语、句子或者段落作为答案。

(4) 自由作答：给定文本内容和问题，生成一段可以回答问题的文字。答案可能并没有出现在给定文本中。

本节将从算法类型角度，介绍基于特征的阅读理解和基于深度神经网络两大类阅读理解算法。

10.2.1 基于特征的阅读理解算法

基于特征的阅读理解算法通常是根据人工设计的特征，使用规则或者有监督分类算法，针对问句对篇章中的句子或者短语进行评分。基于该评分，选取得分最高的句子或片段做为答案。

1. 基于规则的 Quarc 阅读理解算法

Quarc（QUestion Answering for Reading Comprehension）[25] 是一个基于规则的阅读理解系统，面向片段抽取式的阅读理解类型，并且抽取的答案限定于句子级别，试图在给定的文章中找到最合适的句子作为问题的答案。Quarc 使用词汇和语义作为基础规则条件，通过启发式的规则来寻找最合适的句子作为答案。由于不同类型的问题所使用的规则有比较大的差别，Quarc 系统将问题分为：WHO、WHAT、WHEN、WHERE、WHY 等 5 个类型，并针对不同类型的问题构建了不同的规则集。

Quarc 算法首先使用浅层句法解析器 Sundance[26] 进行形态分析、词性标注、语义类标注和实体识别，解析问题和文章中的所有句子。为了避免直接使用单词造成的规则的泛化能力不足的问题，Quarc 算法在对单词进行比对时会使用词根，用于消除语法形式的影响。此外还定义了语义类（例如 HUMAN、LOCATION 等），并使用规则方式识别单词的语义类。主要的语义类别包含以下几项：

- HUMAN：包含 2608 个单词，包含常见的名，姓氏和例如“博士”和“女士”等头衔，并且包含 600 个从 WordNet 中抽取的职业词。
- LOCATION：包含 344 个单词，包含 204 个国家名和 50 个美国的州名。
- MONTH：包含一年中的 12 个月。
- TIME：共包含 667 个词语，其中 600 个是 1400-1999 的年份，其他是一些通用的时间表达。

在此基础上，根据问题的类型选择不同的规则集合，将这些规则应用于文章中包括标题在内的所有句子。每条规则都会给予句子一定的分数，分数的高低取决于该规则对于它能找到答案的确定程度。一条规则可以分配四种类型的分数：线索（clue）（+3），好线索（good_clue）（+4），确信（confident）（+6），完全契合（slam_dunk）（+20）。当所有规则都执行完毕后，得到分数最高的句子将被选作答案。以 WHERE 类问题的规则集合为例，Quarc 算法中 WHERE 类部分规则如下：

1. Score(S) += WordMatch(Q,S)
2. If contains(S,LocationPrep) Then Score(S) += good_clue
3. If contains(S, LOCATION) Then Score(S) += confident

规则 1 中 WordMatch 函数是根据同时出现在句子和问题中的单词计算的得分。当两个单词拥有相

同的词根时会认为这两个单词匹配。由于动词对于识别问题和句子的相关性非常重要，所以动词匹配的权重比非动词更高：每个匹配的动词被授予 6 分，其他匹配词仅有 3 分。规则 2 中 LocationPrep 表示地点介词（包括：in、at、near 等），该规则试图寻找句子 S 中包含地点介词，对于包含地点介词的句子增加“好线索”类对应的分数。规则 3 试图寻找包含 LOCATION 语义类的句子，相应的句子增加“确信”类所对应的分数。

当所有规则应用于文章中的每个句子，并分配了分数之后，拥有最高分数的句子就被当成是最佳答案。如果有相同分数的句子，对于原因类的问题将会选择出现更晚的句子，而其他问题都会选择出现最早的句子。如果所有句子都没有得到分数，时间和地点类的问题将返回根据日期线规则集得到的句子，原因类问题将选择文章中的最后一个句子，其它类型都将返回文章的第一个句子。

Quarc 算法是典型的基于规则的自然语言处理方法，在对篇章进行词法、句法和语义分析的基础上，构建人工规则。在阅读理解这种复杂的自然语言处理任务上，基于规则的方法效果并不十分理想。

2. 基于最大熵的阅读理解算法

在阅读理解任务中，通常给定的输入文本很长，但是文章中可以作为答案的句子却只出现一次或很少几次，在这种情况下，想要从输入文本中提炼出一个简短的答案就显得非常困难。上一节中所介绍的基于规则的方法，以词为核心对输入文本进行浅层的分析，但这种浅层的分析很难解决这种答案与文本长度相差悬殊的情况，为了解决这些问题，必须对输入文本进行更深入的分析，捕捉更深层的特征。RCME 算法[27] 在词袋模型的基础上，引入了两类深层特征：词依赖特征和语法关系特征，并采用最大熵算法将两类特征加以利用。

为了更好的提取深层特征，RCME 算法需要对输入文本中的每个句子进行语法分析，得到整个句子的语法结构和每个单词之间的依赖关系和词性。图10.4给出了一个对句子进行语法分析并得到语法解析树的例子，语法解析树中蕴含着丰富的结构信息，为句子中的每个单词标明了词性，并清晰的展示了句子中单词的语法关系。RCME 模型将句子中每个单词的词性抽取出来作为词依赖特征，将词之间的语法关系作为语法关系特征，利用这两个深层特征去判断给定文本中的哪一句话最适合作为问题的答案。

例如，对于下面的问题 Q 和句子 C：

Q：“谁写了《红楼梦》?”

C：“《红楼梦》是曹雪芹写的”

其中部分词和词性的集合如下：

Q = {写/动词，红楼梦/名词}

C = {写/动词，红楼梦/名词，曹雪芹/名词}

为了利用深层特征，RCME 首先定义了标识函数 $f_j(Q,C)$，对于问题 Q 中的每一个特征 f_j，当句子 C 中包含该特征时，$f_j(Q,C)=1$，反之，$f_j(Q,C)=0$。在上面的例子中，$f_{\text{动词}}(Q,C)=$

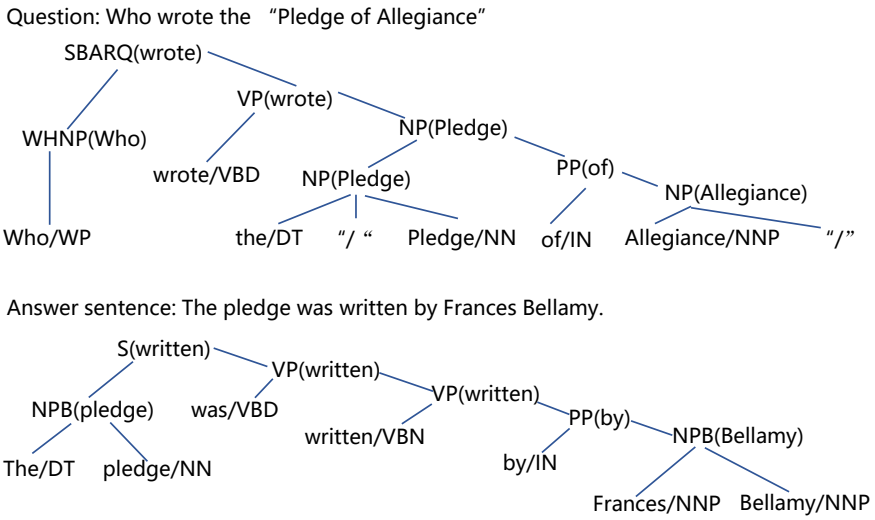

图 10.4 RCME 对问题和答案句语法分析结果示例

$1, f_{\text{名词}}(Q, C) = 1$。RCME 利用最大熵的框架利用这些特征挑选最适合回答问题的句子：对于给定的文本 $S = \{C_1, ..., C_n\}$ 和给定问题 Q, 模型需要从中挑选最适合回答问题的句子：

$$A = \arg\max_{C_i \in S} P(C_i|Q) \tag{10.1}$$

$$P(C_i|Q) = \frac{\exp\left(\sum_j \lambda_j f_j(Q, C_i)\right)}{\sum_C \exp\left(\sum_j \lambda_j f_j(Q, C)\right)} \tag{10.2}$$

其中，λ_j 是根据语料训练得到的不同特征权重。

类似的，RCME 模型会在语法分析树上抽取单词之间的语法关系（例如：主谓关系等），作为特征计算最适合回答问题的句子，模型的提取特征方式与词性很相似，这里就再过多描述。

10.2.2 基于深度神经网络的阅读理解算法

随着深度学习方法在自然语言处理领域广泛应用，特别是 2016 年斯坦福大学发布的阅读理解数据集 SQuAD[15]，极大的推动了阅读理解的研究进展。大量基于深度学习算法的阅读理解方法也在 2016 年之后不断涌现。

1. 双向注意力流网络阅读理解算法

BiDAF[28] 是基于双向注意力流（Bi-directional Attention Flow）的阅读理解算法。针对的是 SQuAD 评测集合，给定文章和问题，算法需要返回文章中的一个片段作为答案，答案片段通常是短语。BiDAF 将该任务建模为将篇章和问题作为输入，预测开始位置和结束位置的问题。BiDAF

神经网络结构如图10.5所示，由六个神经网络层组成，分别为字符向量模块、词向量模块、注意力流层，上下文向量层、建模层以及输出层。在输入部分显式的区分篇章内容和问题，分别使用 $\{x_1, x_2, ...x_T\}$ 和 $\{q_1, q_2, ...q_J\}$ 表示，其中 T 和 J 分别是问题和文本的长度。

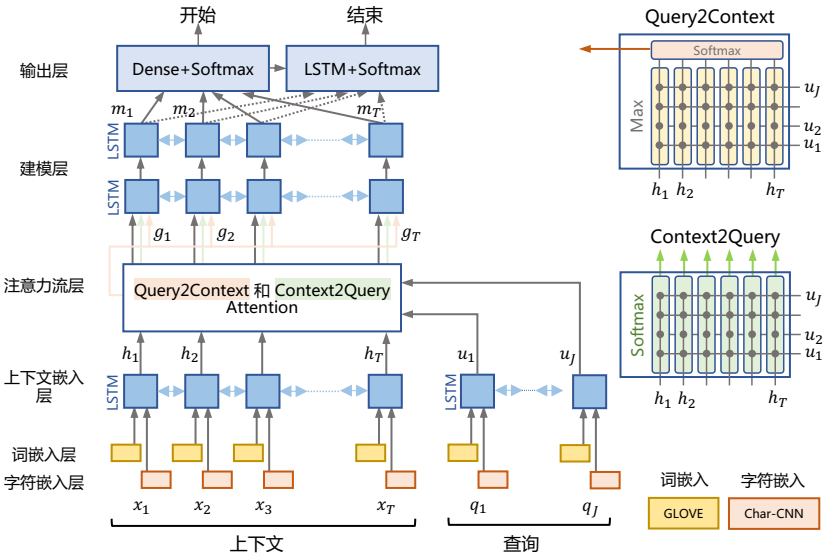

图 10.5　BiDAF 神经网络结构图[28]

字符向量层（Character Embed Layer）作用是利用卷积神经网络对文本和问题中每个单词的字符进行建模，计算得到每个单词相应的字符表示。

词向量层（Word Embed Layer）将问题和文章中的每个单词生成对应的词向量表示，BiDAF 使用了 Glove [29] 生成文本和问题的词向量，并构成文章表示矩阵 $\boldsymbol{X} \in \mathbb{R}^{d \times T}$ 和问题表示矩阵 $\boldsymbol{Q} \in \mathbb{R}^{d \times J}$。

上下文向量层（Contextual Embed Layer）使用 Highway Networks[30] 对前两层中生成的字符向量和词向量进行融合，并将融合后的结果输入给一个 BiLSTM 网络，分别计算得到问题和文章的隐状态序列 $\boldsymbol{U} = [\boldsymbol{u}_1, \boldsymbol{u}_2, ..., \boldsymbol{u}_J] \in \mathbb{R}^{2d \times J}$ 和 $\boldsymbol{H} = [\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_T] \in \mathbb{R}^{2d \times T}$。由于这里使用了 BiLSTM，因此 $\boldsymbol{u}_i$ 和 $\boldsymbol{h}_j$ 都是由前向 LSTM 和后向 LSTM 的 d 维隐变量连接而成的 $2d$ 维列向量。此外还需要注意的是，本层的计算过程中并没有对问题和文章进行信息交互。

注意力流层（Attention Flow layer）对文章和问题进行信息交互。首先根据文章表示矩阵 $\boldsymbol{H}$ 和问题表示矩阵 $\boldsymbol{U}$ 计算相似度矩阵 $\boldsymbol{S} \in \mathbb{R}^{T \times J}$, 其中 S_{ij} 表示输入文本中第 i 个单词和给定问题中第 j 个单词之间的相似度：

$$S_{ij} = \boldsymbol{\alpha}(\boldsymbol{h}_i, \boldsymbol{u}_j) \tag{10.3}$$

$$\boldsymbol{\alpha} = \boldsymbol{W}_S^T[\boldsymbol{h}_i; \boldsymbol{u}_j; \boldsymbol{h}_i \circ \boldsymbol{u}_j] \tag{10.4}$$

其中 $\boldsymbol{W}_S \in \mathbb{R}^{6d}$ 是一个可训练的模型参数矩阵，$\boldsymbol{h}_i \circ \boldsymbol{u}_j$ 是 $\boldsymbol{h}_i$ 和 $\boldsymbol{u}_j$ 之间的元素积（Element-wise Product），$[;]$ 表示向量按行连接。

接下来模型分别计算篇章到问题注意力和问题到篇章注意力。篇章到问题注意力表示对于每一个篇章中的单词来说，哪一个问题中的单词是最相关的。a_t 表示第 t 个单词对所有问题单词的注意力权重，其中 $\sum a_{tj} = 1$, $\boldsymbol{a}_t$ 通过 S 计算而来:

$$\boldsymbol{a}_t = \text{Softmax}(\boldsymbol{S}_{t:}) \tag{10.5}$$

紧接着就可以求得注意力加权后的问题表示向量:

$$\tilde{\boldsymbol{U}}_{:t} = \sum_j \boldsymbol{a}_{tj} \boldsymbol{U}_{:j} \tag{10.6}$$

问题到篇章注意力表示对于每个问题单词哪个文章单词拥有最高的相似度，因此这对回答问题来说非常重要。类似地，模型通过 S 计算问题到篇章注意力:

$$\boldsymbol{b} = \text{Softmax}(max_{col}(\boldsymbol{S})) \tag{10.7}$$

然后可以求得注意力加权后的文章表示向量:

$$\tilde{\boldsymbol{h}} = \sum_t \boldsymbol{b}_t \boldsymbol{H}_{:t} \tag{10.8}$$

表示向量向量通过加权后得到了问题最关注的文章信息。

最后，对上下文向量和注意力向量进行拼接，得到向量 $\boldsymbol{G}$:

$$\boldsymbol{G}_{:t} = \beta(\boldsymbol{H}_{:t}, \tilde{\boldsymbol{U}}_{:t}, \tilde{\boldsymbol{H}}_{:t}) \tag{10.9}$$

其中，$G_{:t}$ 表示第 t 列向量，对应的是第 t 个文章单词，β 是一个用来综合三个输入向量的可训练的向量函数。其中，β 可以是任何一个可训练的神经网络，例如多层感知机或者简单的拼接。

建模层（Modeling Layer），将之前计算出的 G 输入给建模层，输出是建模层获取到的文章向量和问题向量的信息交互。模型采用 BiLSTM 网络，网络输出矩阵 M 用来预测答案。

输出层（Output Layer），利用 G 预测答案的起始位置 p^1 和答案的结束位置 p^2:

$$\boldsymbol{p}^1 = \text{Softmax}(\boldsymbol{W}_{p^1}^T[\boldsymbol{G}, \boldsymbol{M}]) \tag{10.10}$$

$$\boldsymbol{p}^2 = \text{Softmax}(\boldsymbol{W}_{\boldsymbol{p}^2}^T[\boldsymbol{G}, \boldsymbol{M}^2]) \tag{10.11}$$

其中，$\boldsymbol{W}_{p^1}$ 和 $\boldsymbol{W}_{p^2}$ 表示可训练的模型参数，特别地 $\boldsymbol{M}^2$ 是基于 $\boldsymbol{M}$ 再采用一次 BiLSTM 网络生成的另外一个隐向量序列，计算方法类似于建模层所使用的方法。

2. 基于门控自匹配注意力机制的阅读理解算法

R-Net[24] 是基于门控自匹配注意力机制的阅读理解模型，R-Net 在前人工作的基础上进行了改进和提升，取得了明显的效果。R-Net 的创新主要有两点：（1）在 Match-LSTM 网络的基础上进行改进，引入了门控的机制来学习输入文本的哪一部分与问题的相关性最大；（2）自匹配机制，本质是自注意力，充分学习了文本间词和词之间的关系，而不是只关注于文本和问题之间的关系，R-Net 之前的工作，只是利用一个双向 LSTM 对输入文本进行特征提取，而没有进行更深的挖掘。R-Net 的神经网络结构如图10.6所示。

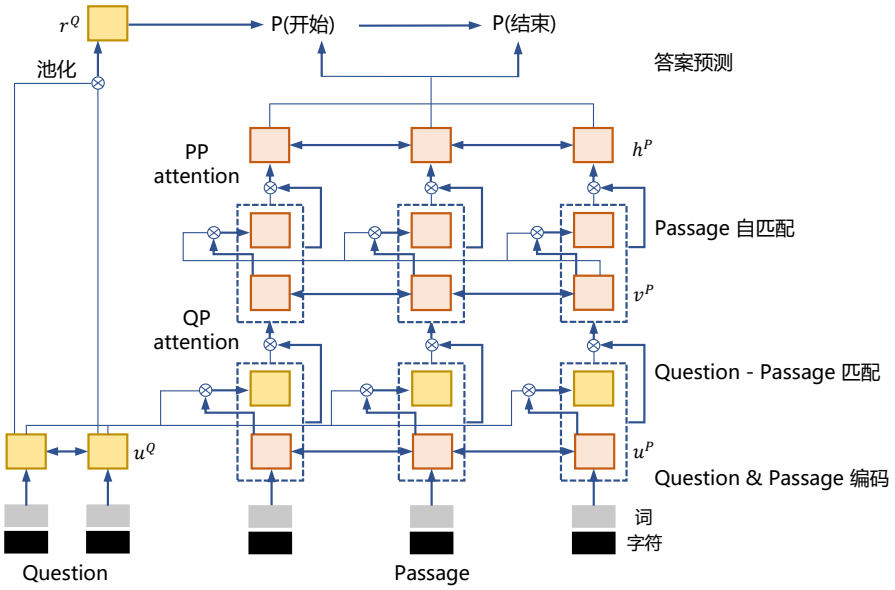

图 10.6　R-Net 神经网络结构图[24]

R-Net 采用两个不同的 BiGRU 网络分别作为问题编码器和文本编码器，对问题 $Q=[w_1^Q,...,W_m^Q]$ 和文本 $P=[w_1^P,...,w_n^P]$ 中每个单词所对应的词向量和字符向量表示进行编码，生成对应的隐状态序列:

$$\boldsymbol{u}_t^Q = \text{BiGRU}(\boldsymbol{u}_{t-1}^Q, [\boldsymbol{e}_t^Q, \boldsymbol{c}_t^Q]) \tag{10.12}$$

$$\boldsymbol{u}_t^P = \text{BiGRU}(\boldsymbol{u}_{t-1}^P, [\boldsymbol{e}_t^P, \boldsymbol{c}_t^P]) \tag{10.13}$$

$\boldsymbol{u}_t^Q$,$\boldsymbol{u}_t^P$ 分别表示问题和文本中第 t 个单词所对应的隐向量表示，$\boldsymbol{e}_t^Q$,$\boldsymbol{e}_t^P$ 分别表示问题和文本中第

t 个单词所对应的词向量，$\boldsymbol{c}_t^Q$,$\boldsymbol{c}_t^P$ 代表字符向量。

为了充分挖掘输入文本中与给定问题相关程度最大的部分，R-Net 对输入文本的特征提取上进行了改进，不仅仅是对整个输入文本进行处理，而是更细粒度的关注文本中的不同部分与问题的关系。为此，R-Net 模型采用了以门控注意力机制为基础的循环网络将给定问题与输入文本对应的向量表示相融合。首先，将字符向量和单词向量连接在一起，输入 RNN 网络，对文章总体进行语义建模，并通过注意力机制对上下文向量进行加强。得到上下文向量 c:

$$\boldsymbol{v}_t^P = \text{RNN}(\boldsymbol{v}_{t-1}^P, [\boldsymbol{u}_t^P, \boldsymbol{c}_t]) \tag{10.14}$$

$$\boldsymbol{c}_t = \sum_{i=1}^{m} \boldsymbol{a}_i^t \boldsymbol{u}_i^Q \tag{10.15}$$

$$\boldsymbol{a}_i^t = \text{Softmax}(\boldsymbol{s}^t) \tag{10.16}$$

$$\boldsymbol{s}_j^t = \boldsymbol{v}^T tanh(\boldsymbol{W}_u^Q \boldsymbol{u}_j^Q + \boldsymbol{W}_u^P \boldsymbol{u}_t^P + \boldsymbol{W}_v^P \boldsymbol{v}_{t-1}^P) \tag{10.17}$$

其中, $\boldsymbol{W}_u^Q$，$\boldsymbol{W}_u^P$ 和 $\boldsymbol{W}_v^P$ 都是可训练的参数矩阵。最后，通过门控注意力的方式，将问题信息和文本信息进行充分融合:

$$\boldsymbol{g}_t = \text{Sigmoid}(\boldsymbol{W}_g[\boldsymbol{u}_t^P, \boldsymbol{c}_t]) \tag{10.18}$$

$$\boldsymbol{u}_t^P = \boldsymbol{g}_t \cdot \boldsymbol{u}_t^P \tag{10.19}$$

$$\boldsymbol{c}_t = \boldsymbol{g}_t \cdot \boldsymbol{c}_t \tag{10.20}$$

$\boldsymbol{W}_g$ 是可训练的参数矩阵。

模型采用自匹配注意机制生成文本向量表示序列 $\boldsymbol{H} = [h_1^P, ..., h_n^P]$ 对于阅读理解任务来说, 输入文本中每个单词的向量表示不仅仅与问题中的单词有关，还应该与该单词的上下文有关，自匹配注意力机制就是要将这两种信息融合起来。

$$\boldsymbol{h}_t^P = \text{BiRNN}(\boldsymbol{h}_{t-1}^P, [\boldsymbol{v}_t^P, \tilde{\boldsymbol{c}}_t]) \tag{10.21}$$

$$\tilde{\boldsymbol{c}}_t = \sum_{i=1}^{n} \boldsymbol{a}_i^t \boldsymbol{v}_i^P \tag{10.22}$$

$$\boldsymbol{a}_i^t = \frac{exp(\tilde{\boldsymbol{s}_i^t})}{\sum_{j=1}^{n} exp(\tilde{\boldsymbol{s}_j^t})} \tag{10.23}$$

$$\tilde{\boldsymbol{s}_j^t} = \tilde{\boldsymbol{v}^t} \tanh(\tilde{\boldsymbol{W}_v^P} \boldsymbol{v_j^P} + \tilde{\boldsymbol{W}_v^P} v_t^P) \tag{10.24}$$

最后 R-Net 利用 Pointer Network 对答案的起始位置 p^1 和结束位置 p^2 进行预测：

$$\boldsymbol{p}^t = \text{argmax}(\boldsymbol{a}_1^t, ..., \boldsymbol{a}_n^t) \tag{10.25}$$

$$\boldsymbol{a}_i^t = \frac{exp(\boldsymbol{s}_i^t)}{\sum_{j=1}^n exp(\boldsymbol{s}_j^t)} \tag{10.26}$$

$$\boldsymbol{s}_j^t = \boldsymbol{v}^T \tanh(\boldsymbol{W}_h^P \boldsymbol{h}_j^P + \boldsymbol{W}_h^a h_{t-1}^a) \tag{10.27}$$

$$\boldsymbol{h}_t^a = \text{RNN}(\boldsymbol{h}_{t-1}^a, \boldsymbol{c}_t) \tag{10.28}$$

$$\boldsymbol{c}_t = \sum_{i=1}^n \boldsymbol{a}_i^t \boldsymbol{h}_i^P \tag{10.29}$$

h_{t-1}^a 表示 Pointer Network 中最后一个隐状态变量，在预测答案其实位置时，其计算方法如下

$$\boldsymbol{h}_{t-1}^a = \sum_{i=1}^m \boldsymbol{a}_i \boldsymbol{u}_i^Q \tag{10.30}$$

$$\boldsymbol{a}_i = \frac{exp(\boldsymbol{s}_i)}{\sum_{j=1}^m exp(\boldsymbol{s}_j)} \tag{10.31}$$

$$\boldsymbol{s}_j = \boldsymbol{v}^T \tanh(\boldsymbol{W}_u^Q u_j^Q + \boldsymbol{W}_v^Q \boldsymbol{V}_r^Q) \tag{10.32}$$

其中，$\boldsymbol{W}_u^Q, \boldsymbol{W}_v^Q, \boldsymbol{W}_h^P, \boldsymbol{W}_h^a$ 都是可训练的参数矩阵。

3. 基于片段掩盖的预训练的阅读理解算法

SpanBERT[31] 是一种可以更好的表示和预测片段的文本的预训练算法，也是对 BERT 原本预训练方法的一种扩展。SpanBERT 对 BERT 模型主要进行了如下改进：

- 不再是随机对单个词进行遮盖操作，而是对相邻的分词进行遮盖操作；
- 加入了 Span Boundary Objective（SBO）作为训练目标。

原始 BERT 算法，在训练时会随机选取句中的子词进行遮盖，但这种训练方式，会让一些原本就连续出现且相关性很强的词组短语，在训练的时候被割裂开。SpanBERT 打破了这种范式，对于给定的文本 $X = x_1...x_n$, SpanBERT 通过迭代的方式从中选择需要遮盖的片段，在每一次迭代中，SpanBERT 通过几何分布对遮盖长度 l 进行随机采样，l 代表要遮盖的单词的数量，紧接着在一个均匀分布中采样出一个遮盖起点 s，对 $\{x_s, ..., x_{s+l}\}$ 进行遮盖，在损失函数方面，这一部分与原本的 BERT 保持一致，采用 MLM 的损失函数进行训练。

另外, SpanBERT 引入了一个新的训练目标, Span Boundary Objective（SBO）。SpanBERT 希望训练好的语言模型可以让每个片段的边缘单词的表示尽可能的包含片段内部内容的所有语义，所以 SBO 的含义是要求在片段边界处的单词表示尽可能可以对遮盖片段内部的单词进行预测。具体来说,对于给定文本 X,对于某个被选中的遮盖片段 $\{x_s, ..., x_{s+l-1}\}$,SpanBERT 希望通过 $\{x_{s-1}, x_{s+l}\}$

以及位置向量 p_{i-s+1} 来对被遮盖的单词 x_i 进行预测：

$$y_i = f(h_{s-1}; h_{x_{s+l}}; p_{i-s+1}) \tag{10.33}$$

所以，最终 SpanBERT 的损失函数为：

$$\mathcal{L}(x_i) = \mathcal{L}_{MLM}(x_i) + \mathcal{L}_{SBO}(x_i) \tag{10.34}$$

$$= -logP(x_i|X) - logP(x_i|x_{s-1}, x_{s+l}, p_i) \tag{10.35}$$

图10.7给出了一个片段掩盖的样例，该例子中“an American football game”全部使用 [MASK] 替代，x_4 和 x_9 以及位置编码用于预测掩盖区域内的单词。在本例中“football”是掩盖片段中的第 3 个单词，因此采用 p_3 的位置编码。“football”所对应的损失函数为：

$$\mathcal{L}(\text{football}) = -logP(\text{football}|x_7) - logP(\text{football}|x_4, x_9, p_3) \tag{10.36}$$

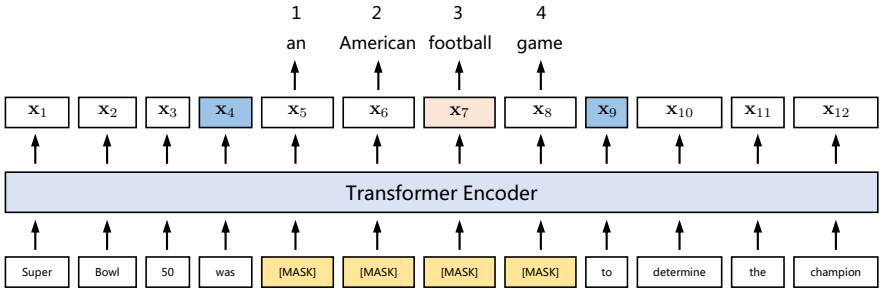

图 10.7 SpanBERT 神经网络片段掩盖样例[31]

在预训练完成的基础上，采用与 BERT[32] 相同的建模方式，将段落 $P = (p_1, p_2, ..., p_m)$ 和问题 $Q = (q_1, q_2, ..., q_n)$ 编码为一个序列 $X = [\text{CLS}]p_1p_2...p_n[\text{SEP}]q_1q_2..q_m$ 作为输入。在段落单词所对应的位置，训练两个线性分类器，分别建模该单词是否为答案片段的开始还是结束位置。为了应对 SQuAD 2.0 中存在不能回答的问题的情况，将答案片段的开头和结尾都标志于 [CLS] 所对应的位置。神经网络架构如图10.8所示。

10.2.3 阅读理解语料库

当前常用的阅读理解数据集如表10.2所示，其中 CNN/Daily Mail、HLF-RC 和 Children’s Book 是完形填空数据集。MCTest 和 RACE 是多项选择任务的数据集，RACE 是从中国初中高中的英语测试题中收集的。SQuAD、TriviaQA 和 NewsQA 是片段抽取的数据集。

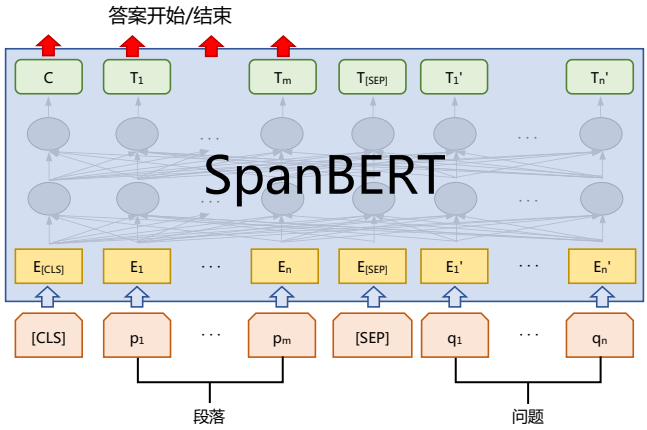

图 10.8　SpanBERT 神经网络阅读理解框架[31]

表 10.2　阅读理解常用数据集

数据集	语言	任务类型	数据来源	问题数量	文档数量
CNN / Daily Mail	英语	完形填空	自动生成	140 万	30 万
Children's Book	英语	完形填空	自动生成	68.8 万	68.8 万
HLF-RC	中文	完形填空	自动生成	10 万	2.8 万
MCTest	英语	多项选择	人工编写	2640	660
RACE	英语	多项选择	英文测试题	87 万	5 万
SQuAD	英语	片段抽取	人工编写	10 万	536
TriviaQA	英语	片段抽取	用户问答界面	4 万	66 万
NewsQA	英语	片段抽取	人工编写	10 万	1 万
MS-MARCO	英语	自由问答	用户日志	10 万	20 万
DuReader	中文	自由问答	用户日志	20 万	100 万

1. SQuAD 数据集

SQuAD[15] 是斯坦福大学于 2016 年推出的阅读理解数据集，给定一篇文章和相应问题，需要模型给出问题的答案，SQuAD 是片段抽取类任务，答案为文章中的片段。数据集的所有文章都选自维基百科，一共有 107,785 个问题，以及配套的 536 篇文章。最近，SQuAD2.0 已发布，其中包含无法回答的问题。SQuAD 的主要缺点是数据集中，问题的答案一般是很简短的一个词组。

2. MS-MARCO 数据集

MS MARCO[33] 又称人类生成的机器阅读理解数据集，由 Microsoft AI & Research 设计和开发。数据集中的所有问题都是从真实的匿名用户查询中获得的，研究者们为这些问题编写了答案。

问题配套的文章是通过 Bing 搜索引擎从真实文档中提取可以获取答案的上下文段落。该数据集有 1,010,916 个数据。

3. TrivialQA 数据集

TrivialQA 数据集是一个难度较高的阅读理解数据集，其中问题较为复杂，并且问题和文章中相应的支持句之间有较大的差距，例如一些句法或者词法的变化，并且相比其他阅读理解数据集，TrivialQA 中包含更多的需要多跳推理能力的问题。

4. RACE 数据集

RACE 数据集是一个从考试中提取的数据集，旨在模拟真实的人类测试，RACE 是第一个基于真实考试的大型数据集。RACE 的数据分为两个部分 RACE-M 和 RACE-H，两个部分的数据分别是从初中和高中的练习题中进行采集而成，RACE-H 中的文本长度和词汇量都比 RACE-M 要长，但总体而言，这些数据的句子长度和复杂度都是比新闻文章或维基百科文章简单。

10.3 表格问答

表格是一种常见的数据存储形式，通常由表头，表格单元和表格标题构成：表格标题概括了表格包含的主要内容，表头一般表示某行或者某列表格单元的内容和类型。除此之外，表格中每个元素都是一个表格单元。图10.9给出了表格问答样例。对于问题“哪个城市举办了 2008 年奥运会？”，通过表格选择和单元格抽取，最终返回“北京”作为答案。

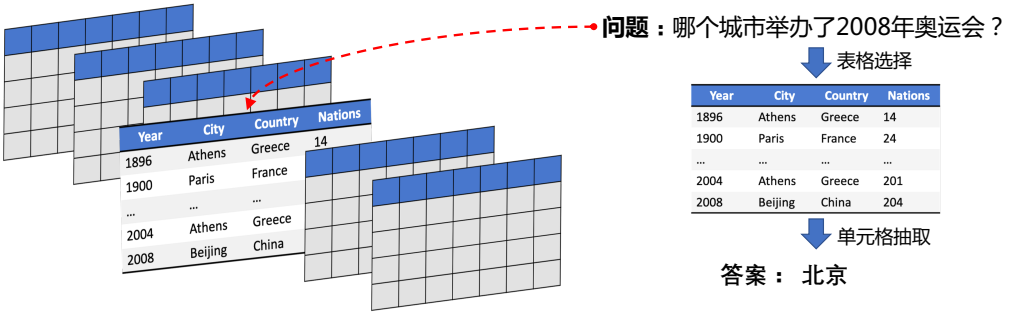

图 10.9 表格问答样例

10.3.1 基于特征的表格问答方法

对于解决表格问答问题，一个基本的想法是将输入的表格视为一个知识库，但这个过程本身包含着很多挑战，因为知识库包含了太多的关系，这令表格数据充满噪音。

CSP 算法[23] 是基于特征的表格问答方法，采用逻辑形式驱动。对于一个给定的表格 t 和一个关于表格的问题 q, 模型需要根据表格内容输出答案 y 来回答 q。对于问题 q 唯一的限制是这个问

题需要仅能够通过表格内容回答，问题可以是任何形式。CSP 算法的整体流程如图10.10所示。

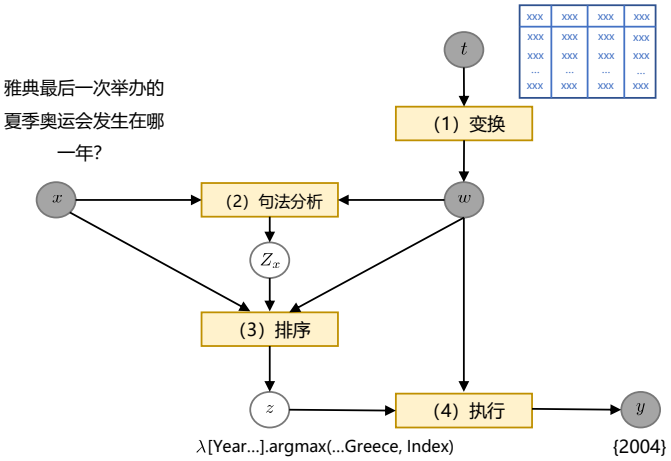

图 10.10　CSP 模型结构图[23]

为了解决表格知识库的噪音问题，CSP 模型首先将表格 t 转化成知识图谱的形式。具体来说，将表格的行视为行节点，表格单元中的字符串被视为实体节点，表格的列被转化成连接行节点和实体节点的有向边，而列表头则成为了这些边的标签。得到知识图谱 w 之后，CSP 模型将问题 q 解析成一些候选的逻辑形式 Z_x。CSP 模型提出了表单检索的方法，对于给定的问题 $q = \{q_1, ..., q_n\}$，模型通过两类规则对问题进行解析：

$$(q, i, j)[s] -> (c, i, j)[f(s)] \tag{10.37}$$

$$(c_1, i, k)[z_1] + (c_2, k+1, j)[z_2] -> (c, i, j)[f(z_1, z_2] \tag{10.38}$$

第一个规则是一类词法匹配规则，可以将问题片段 $q_i...q_j$ 转化为逻辑形式，其中 c 表示类别，s 表示问题片段的文本。例如，s 为“纽约”，c 为“实体”时，根据表格查询内容 $f(s)$ 为“纽约市”。第二个规则将两个相邻片段的逻辑形式 z_1 和 z_2 合并成一个新的逻辑形式 $f(z_1, z_2)$。

CSP 模型将以上特征 $g(x, w, z)$ 输送给逻辑线性模型去捕捉问题 q 和候选 z 之间的关系。CSP 模型利用解析生成出逻辑形式 Z_x，每一个逻辑形式 z_i 都是一个从知识图中抽取出的图问题，利用提取出的特征向量 $g(x, w, z)$，CSP 模型定义了对于每个逻辑线性分布：

$$p(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{w}) = exp(\boldsymbol{\theta}^T g(\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{z})) \tag{10.39}$$

模型的训练方式如下，对于给定的训练样本 $D = \{(x_i, t_i.y_i)\}_{i=1}^N$，CSP 模型在最大似然估计的基

础上增加正则项作为目标函数：

$$L = \frac{1}{N}\sum_{i=1}^{N} logp(\boldsymbol{y}_i|\boldsymbol{q}_i, \boldsymbol{w}_i) - \boldsymbol{L}_1 \tag{10.40}$$

其中 L_1 是模型的 L1 正则，w_i 是根据 t_i 动态生成的：

$$p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{w}) = \sum_{z \in Z_x} p(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{w}) \tag{10.41}$$

在训练完毕之后，模型挑选分数最高的逻辑形式 z 作为答案。

10.3.2 基于深度学习的表格问答模型

CLTR[34] 是基于预训练的端到端表格问答模型，该方法首先从大量表格中筛选得到少量相关表格，再对这些表格进行重新排序，最终回答用户提出的自然语言问题。其模型神经网络结构如图10.11所示。

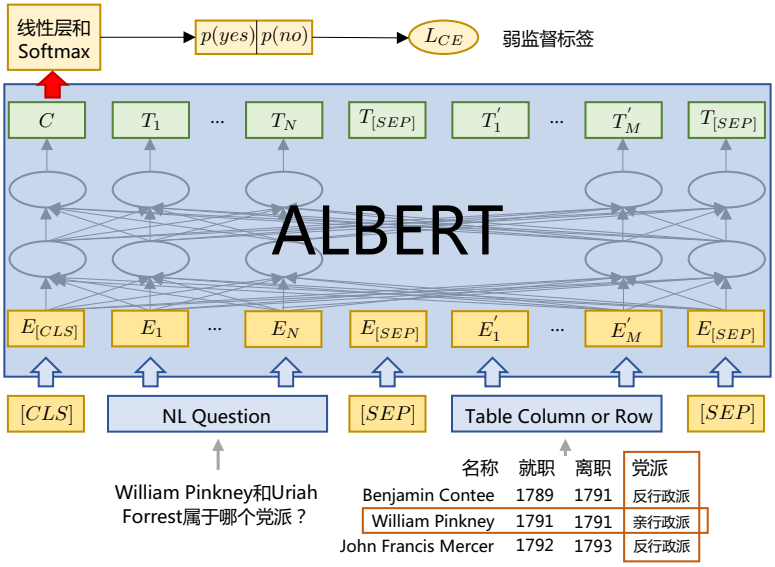

图 10.11 CLTR 模型神经网络结构图[34]

CLTR 模型首先利用 BM25 算法筛选出一个高相关表格池，接着采用行列交互模型对表格池中表格中的行列生成包含答案的概率。行列交互模型将表格问答分为两个基本操作：行选择和列选择，将行列的预测分布进行合并，就能得到答案单元格的目标概率。CLTR 采用文本分配的方式预测答案所在的行列，具体来说就是将每行每列分别与问题 q 相连输入 BERT 模型中，通过二分

类器得到概率分数：

$$\boldsymbol{H}_{r_i} = \text{BERT}(\boldsymbol{q}, \boldsymbol{r}_i) \tag{10.42}$$

$$\boldsymbol{H}_{c_j} = \text{BERT}(\boldsymbol{q}, \boldsymbol{c}_j) \tag{10.43}$$

$$\boldsymbol{P}_{r_i} = \text{Binary Classifier}(\boldsymbol{H}_{r_i}) \tag{10.44}$$

$$\boldsymbol{P}_{c_j} = \text{Binary Classifier}(\boldsymbol{H}_{c_j}) \tag{10.45}$$

至此，在表格池 T 中每张含有 n 行 m 列的表格 t 都含有两个分数集合 $S_c = \{s_{c_1}, ..., s_{c_m}\}$ 和 $S_r = \{s_{r_1}, ..., s_{r_n}\}$。紧接着对整张表格计算最大的单元分数：

$$\boldsymbol{S}_t = max(\boldsymbol{S}_c) + max(\boldsymbol{P}_r) \tag{10.46}$$

CLTR 模型利用最大单元分数对表格池里的候选表格进行重排序，当重排序做完之后，会挑选表格池中得分前 k 的表格返回给用户。最后的答案由行列交互模型所计算的分数，寻找分数最高的交叉行列来得到目标表格单元。

10.3.3 表格问答语料库

当前常用的表格问答语料库如表10.3所示。其中 WikiSQL 和 Spider 数据集合采用用户真实问句，并人工转换为数据库 SQL 查询语句的方式。WikiTableQuestions 则是来源于维基百科的半结构化表格。

表 10.3 表格问答常用数据集

数据集	数据来源	表格数量	问句数量
WikiSQL	用户真实问句转换	24241	80645
Spider	用户真实问句转换	206	10181
WikiTableQuestions	维基百科	2108	22033

1. WikiSQL

WikiSQL 数据集是 2017 年由 Salesforce 提出的大型标注数据集，WikiSQL 是一类 NL2SQL 数据集，该任务要求模型将用户的自然语言询问转换成 SQL 语句并搜索出答案，WikiSQL 是目前规模最大的 NL2SQL 数据集。WikiSQL 包含了 24241 张表，80645 条自然语言问句及相应的 SQL 语句。

2. Spider

Spider 数据集由耶鲁大学于 2018 年提出，Spider 同样是一个 NL2SQL 数据集，其中包含了 10181 条自然语言问句和分布在多个独立数据库中的共 5693 条 SQL 语句，数据集中的内容涉及

138 个不同的领域，相比 WikiSQL，虽然数据量不足，但 Spider 更贴近真实场景，所以难度更大。

3. WikiTableQuestions

WikiTableQuestions 数据集是斯坦福大学提出的一个基于维基百科中半结构化表格问答的数据集，其中包含 22033 条来自真实用户的问句和 2108 张表格，表格中的数据都是取自维基百科的真实数据，表格中的内容往往具有多重含义，并且覆盖了多个领域的数据。特别的是，该数据测试集中的表格主题和实体关系都是训练集中没出现过的。

10.4 社区问答

随着 Web 2.0 的快速发展，社区问答网站自 2008 年以来蓬勃发展，出现了包括 Quora、知乎等在内的众多社区问答网站。用户可以通过这些网站对自己需要解决的问题进行提问，其他用户对这些问题进行回答、关注、评论等。同时，企业内部也大量问答知识，例如人工客服领域，很多高频问题都有相应的知识库。基于这些问答数据，研究人员们提出了社区问答任务，基于收集的 < 问题，答案 > 对数据 $\mathcal{D} = < Q_i, A_i >_{i=1}^{N}$，针对用户自然语言问题 Q 寻找最合适的答案返回。图10.12给出了搜索引擎给出的社区问答样例，对于问题“手机屏碎了怎么办？”，通过与问答库中的问答对语义相关性进行计算，最终返回相关答案。

图 10.12 社区问答样例（来源：搜狗搜索）

社区问答方法可以进一步细分为两个子任务：

(1) 问题-问题匹配：目标是计算输入问题 Q 和问答库 $\mathcal{D}$ 中某个已有问题 Q_i 之间的相似度，如果两者之间相似度较高，显而易见的，Q_i 的答案 A_i 就有很大概率是 Q 的答案。

(2) 问题-答案匹配：目标计算输入问题 Q 与 D 中某个答案 A_i 的相关性，同样的，如果两者之间相关度很高，那么 A_i 有很大概率是 Q 的答案。

本节以问题-问题匹配任务为重点，分别介绍基于特征和基于深度学习的的语义匹配算法。

10.4.1 基于特征的语义匹配

传统的基于特征的语义匹配方法普遍采用一些基于词袋的统计文本表示，通过这些表示去计算两段文本的语义相似度，进而判断两段文本的语义是否匹配。其中，n 元短语匹配度是判断语义相似度的一个重要衡量指标，如果两个问题共同包含的语义片段越多，自然具有更大的可能表达相同的含义。n 元短语匹配度有许多不同的计算方式，最简单的做法可以计算两个问题中重复单词的数量：

$$\text{Sim}(Q_1, Q_2) = \frac{\sum_{w \in Q_1} \text{num}(w, Q_2)}{|Q_1|} \tag{10.47}$$

其中 $\text{num}(w, Q_2)$ 表示单词 w 在 Q_2 中出现的次数，$|Q_1|$ 表示 Q_1 的长度。

类似地，可以将简单的单词匹配扩展到短语匹配，由于短语包含的信息量更多，表达的语义也更加独立，所以短语级匹配特征能更好的表达两个问题之间的语义相似度。常见的短语级匹配特征有 BLEU、编辑距离和最长公共子序列方法都可以将它们用来当作特征来完成问题匹配的任务。

但是，上面介绍的这些语义匹配特征都存在两个缺点：首先，没有处理停用词，例如冠词、介词等，这类词汇虽然在文本中高频出现，但是并不具备实际意义，对语义的贡献程度小。第二，没有处理同义词，由于自然语言的表达非常丰富，同一语义具有非常多的表达方式，这就使得处理语义匹配问题时需要能够对同义词有较好的识别能力。通常采用 FastText、BM25、TF-IDF、SentVec 等方法去解决这些问题，由于大部分方法前面的章节已经介绍过，本章不再赘述，本节主要介绍 BM25 算法。BM25 可以缓解停用词带来的问题，BM25 将基于反转文档频率计算 Q_1 和 Q_2 的相似度:

$$f_{BM25}(Q_1, Q_2) = \sum_{i=1}^{|Q_1|} \text{IDF}(Q_1^i) f req(Q_1^i, Q_2) \tag{10.48}$$

$$\text{freq}(Q_1^i, Q_2) = \frac{\text{num}(Q_1^i, Q_2)(k_1 + 1)}{\text{num}(Q_1^i, Q_2) + k_1(1 - b + b * \frac{|Q_2|}{\text{avglen}(Q_2)})} \tag{10.49}$$

$$\text{IDF}(Q_1^i) = log \frac{N - n(Q_1^i) + 0.5}{n(Q_1^i) + 0.5} \tag{10.50}$$

其中，Q_1^i 表示 Q_1 中第 i 个单词，$\text{IDF}(Q_1^i)$ 表示 Q_1^i 的反转文档频率，$\text{num}(Q_1^i, Q_2)$ 表示单词 Q_1^i 在 Q_2 中出现的次数，$\text{avglen}(Q_2)$ 表示问题及何种问题的平均长度，N 表示文档综述，$n(Q_1^i)$ 表示包含单词 Q_1^i 的文档综述，$k1$ 和 b 事吵参数。最后，调转 Q_1 和 Q_2 的位置可以计算得出 f_{BM25}：

$$\text{BM25}(Q_1, Q_2) = \frac{f_{BM25}(Q_1, Q_2) + f_{BM25}(Q_2, Q_1)}{2} \tag{10.51}$$

通过 BM25 的特征值，可以通过检测句对之间的 bm25 得分是否超过阈值 λ 来判断两个问题之间

是否匹配。此外，也可以利用 WordNet 等知识库来部分解决同义词的问题。

10.4.2 基于深度神经网络的问题匹配

基于深度学习的语义匹配方法主要分为交互式和非交互型两种。非交互型的方法是将两个句子分别编码后进行匹配，而交互式的方法一般是将两个句子一起进行编码，在编码过程中让两个句子进行信息交互。本节中将分别介绍上述两类语义匹配算法。

1. DSSM 语义匹配算法

DSSM（Deep Structured Semantic Model）[35] 是一种非交互式的基于深度网络的文本语义匹配算法，通过深度神经网络对用户问题和题目分别进行建模，将其编码成低维度的语义向量，然后计算两个向量的余弦距离，判断两个句子之间的语义相似度，由于该模型对用户问题和题目分别进行建模，在计算相似度之前两者之间没有交互，这种模型架构又被称作“双塔模型”。DSSM 模型网络结构如图10.13所示。

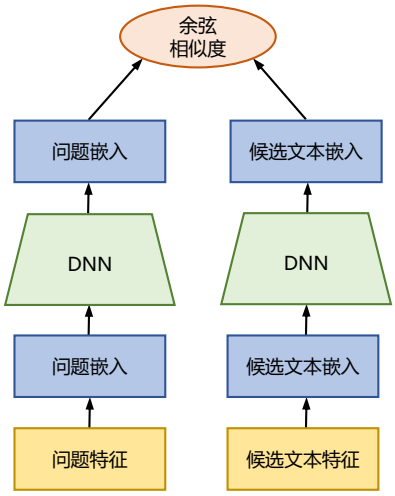

图 10.13　DSSM 模型图[35]

令 $\boldsymbol{x}$ 表示输入向量，$\boldsymbol{y}$ 表示输出向量，$\boldsymbol{y}$ 向量的计算方式如下：

$$\boldsymbol{l}_1 = \boldsymbol{W}_1 \boldsymbol{x} \tag{10.52}$$

$$\boldsymbol{l}_i = f(\boldsymbol{W}_i \boldsymbol{l}_{i-1} + \boldsymbol{b}_i), i = 2, ..., N-1 \tag{10.53}$$

$$\boldsymbol{y} = f(\boldsymbol{W}_N \boldsymbol{l}_{N-1} + \boldsymbol{b}_N) \tag{10.54}$$

其中 l_i 表示隐藏层，共 N 层，W_i 表示第 i 曾的参数矩阵，b_i 表示偏置项，f 是输出层和隐藏层的

激活函数，模型采用 tanh 函数作为激活函数：

$$f(\boldsymbol{x}) = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{10.55}$$

在社区问答任务中使用 Q 表示用户查询，t 表示候选文档题目，两者之间的相关性分数计算方式如下：

$$\text{Score}(Q, t) = \cos(\boldsymbol{y}_Q, \boldsymbol{y}_t) \tag{10.56}$$

DSSM 模型利用点击日志中用户搜索的问题和用户点击的文档作为数据进行训练。其基本假设是：如果用户在当前的搜索问题下点击了某个文档，那么该文档和问题之间一定是相关的。利用该假设通过搜索日志自动构建训练集和测试集。模型在学习过程中利用极大似然估计作为损失函数：

$$\mathcal{L} = -\log \prod_{Q, \mathcal{D}^+} P(\mathcal{D}^+|Q) \tag{10.57}$$

$$P(\mathcal{D}^+|Q) = \frac{\exp(\text{Score}(Q, \mathcal{D}^+))}{\sum_{d \in \mathcal{D}} \exp(\text{Score}(Q, t))} \tag{10.58}$$

其中，$\mathcal{D}$ 表示候选文档集合，$\mathcal{D}^+$ 为正样本。

2. ESIM 语义匹配算法

为解决 DSSM 模型缺乏对用户问题和题目中单词或短语间交互的问题，文献 [36] 提出了 ESIM 模型。该模型基于链式 LSTMs 设计了序列推断模型，并考虑了局部推断和推断组合的问题，在模型中引入了句子间的注意力机制来实现局部推断进而实现全局推断。ESIM 模型结构如图10.14所示，主要分为三个部分：输入编码层、局部推断模块以及推断组合模块。

输入编码层对于输入的两个问题 Q_1 和 Q_2，模型首先通过词嵌入表示得到问题中每个单词的词向量，然后将其输入 BiLSTM 模型，通过 BiLSTM 表达句子的局部信息：

$$\boldsymbol{h}_1^i = \text{BiLSTM}(\boldsymbol{h}_1^{i-1}, \boldsymbol{h}_1^{i+1}, \boldsymbol{q}_1^i) \tag{10.59}$$

$$\boldsymbol{h}_2^i = \text{BiLSTM}(\boldsymbol{h}_2^{i-1}, \boldsymbol{h}_2^{i+1}, \boldsymbol{q}_2^i) \tag{10.60}$$

其中 $\boldsymbol{q}_1^i$ 和 $\boldsymbol{q}_2^i$ 分别表示 Q_1 中的第 i 个单词和 Q_2 中的第 i 个单词的嵌入表示。

局部推断模块在进行局部推断之前，首先模型要对 $\boldsymbol{h}_1$ 和 $\boldsymbol{h}_2$ 进行对齐，首先计算两者的相似度：

$$\boldsymbol{e}_{ij} = \boldsymbol{h}_1^T \boldsymbol{h}_2 \tag{10.61}$$

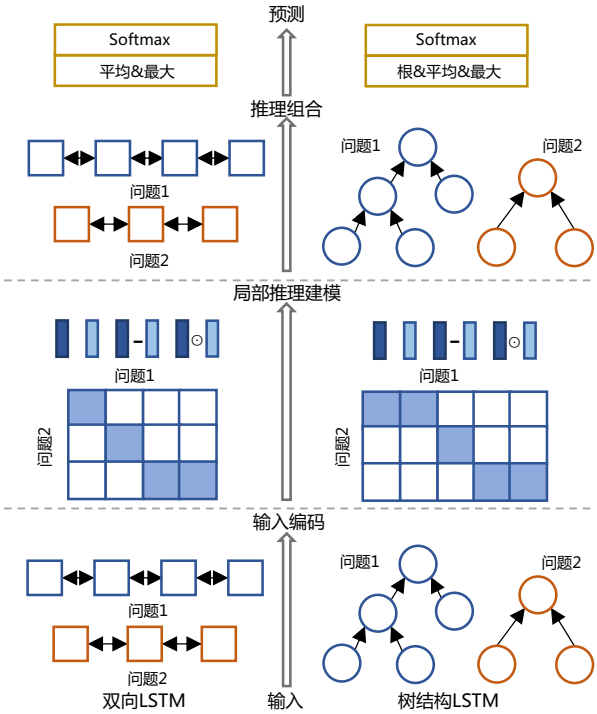

图 10.14 ESIM 模型图[36]

紧接着进行局部推断，结合 $\boldsymbol{h}_1$，$\boldsymbol{h}_2$ 和 $\boldsymbol{h}_{ij}$，得到相似性加权后的向量：

$$\hat{\boldsymbol{h}}_1^i = \sum_{j=1}^{len2} \frac{exp(\boldsymbol{e}_{ij})}{\sum_{k=1}^{len2} \boldsymbol{e}_{ik}} \boldsymbol{h}_2^j \tag{10.62}$$

$$\hat{\boldsymbol{h}}_2^i = \sum_{j=1}^{len1} \frac{exp(\boldsymbol{e}_{ij})}{\sum_{k=1}^{len1} \boldsymbol{e}_{ik}} \boldsymbol{h}_1^j \tag{10.63}$$

在局部推断之后，模型会进行局部信息增强，利用差和点积放大两者的差异性：

$$\boldsymbol{m}_1 = [\boldsymbol{h}_1, \hat{\boldsymbol{h}}_1, \boldsymbol{h}_1 - \hat{\boldsymbol{h}}_1, \boldsymbol{h}_1 \cdot \hat{\boldsymbol{h}}_1] \tag{10.64}$$

$$\boldsymbol{m}_2 = [\boldsymbol{h}_2, \hat{\boldsymbol{h}}_2, \boldsymbol{h}_2 - \hat{\boldsymbol{h}}_2, \boldsymbol{h}_2 \cdot \hat{\boldsymbol{h}}_2] \tag{10.65}$$

推断组合模块首先将 $\boldsymbol{m}_1$ 和 $\boldsymbol{m}_2$ 输入 BiLSTM 提取信息，

$$\boldsymbol{v}_1^i = \text{BiLSTM}(\boldsymbol{m}_1^i) \tag{10.66}$$

$$\boldsymbol{v}_2^i = \text{BiLSTM}(\boldsymbol{m}_2^i) \tag{10.67}$$

$$\tag{10.68}$$

由于两个问题的长度不同导致 $\boldsymbol{m}_1$,$\boldsymbol{m}_2$ 矩阵维度不一致, ESIM 分别采用 MaxPooling 和 AvgPooling 的池化对两个句子进行处理,

$$\boldsymbol{v}_1^{max} = \text{MAXPooling}(\boldsymbol{v}_1) \tag{10.69}$$

$$\boldsymbol{v}_1^{avg} = \text{AvgPooling}(\boldsymbol{v}_1) \tag{10.70}$$

$$\boldsymbol{v}_2^{max} = \text{MAXPooling}(\boldsymbol{v}_2) \tag{10.71}$$

$$\boldsymbol{v}_2^{avg} = \text{AvgPooling}(\boldsymbol{v}_2) \tag{10.72}$$

最后将经过处理的向量拼接起来，连接一个全连接层进行预测:

$$\boldsymbol{v} = [\boldsymbol{v}_1^{max}; \boldsymbol{v}_1^{avg}; \boldsymbol{v}_2^{max}; \boldsymbol{v}_2^{avg}] \tag{10.73}$$

$$\boldsymbol{o} = \text{SoftMax}(\boldsymbol{W}\boldsymbol{v} + \boldsymbol{b}) \tag{10.74}$$

10.4.3 社区问答数据集

当前常用的社区问答数据集如表格10.4所示, 其中 MRPC 和 QQP 是最常用的评价语义匹配模型的基准数据集，LCQMC 是由哈尔滨工业大学构建的中文语义匹配数据集。

表 10.4 社区问答常用数据集

数据集	数据来源	训练集	测试集
MRPC	新闻抽取	3.7k	1.7k
QQP	社区问答抽取	367k	390k
LCQMC	真实对话抽取	239k	12k
PAWS-X	翻译构造	49k	2k

1. MRPC

MRPC(The Microsoft Research Paraphrase Corpus, 微软研究院意译数据集) 数据集是由微软在 2005 年提出的，数据集包含 5800 个句子对，所有句子都是从真实的在线新闻中抽取，由人工注释句对中的句子是否具有相同的语义。该数据集中每个样本的句子长度都很长，并且正负样本的比例不均衡，其中正样本的比例约为 68%。

2. QQP

QQP(The Quora Question Pairs, Quora 问题对数据集) 是基于社区问答网站 Quora 构建的数据集，研究人员在 Quora 上抽取了大量的问题对集合作为数据，由人工标注一对问题是否包含相同的语义。QQP 数据集的数据规模非常大，训练集，开发集和测试集分别包含 363870，40431，390965 个样本。与 MRPC 类似，QQP 数据集也是样本不均衡的，其中负样本占总数的 63%。

3. LCQMC

LCQMC 是哈尔滨工业大学构建的问题语义匹配数据集，其目标是判断两个问题的语义。输入的句子具有高度口语化的特征，这大大提升了语义匹配的难度。输入是两个句子，输出是 0 或 1。其中 0 代表语义不相似，1 代表语义相似。训练集，开发集和测试集分别包含 238766，8802，12500 个样本。

4. PAWS-X

PAWS-X 是由谷歌公司发布的同义句识别数据集，需要模型识别一对句子是否含有相同的语义，其中文本具有高度重叠的词汇，重点考察模型对高层语义的识别能力。训练集，开发集和测试集分别包含 49401，2000，2000 个样本。

10.5 开放领域问答

开放领域问答是与搜索引擎类似的形式，对于一个给定的询问 Q，开放领域问答系统需要先从一个规模非常大的文档集合或知识库（如 Wikipedia 或互联网）$D = \{d_1, ..., d_n\}$ 中选择与问题最相关的文档 d，然后通过文档 d 中包含的信息，给出询问 Q 的答案 A。从任务定义中可以发现，开放问答的结果过程一般分为两个阶段：文档检索和阅读理解。图10.15给出了开放领域问答搜索引擎中的结果，用户输入查询“血压 170 严重吗”，搜索引擎定位了相关文档，并将直接输出最终答案。

图 10.15　开放领域问答样例（来源：百度）

10.5.1 检索-阅读理解架构的开放问答模型

由于开放问答任务面对的是大量的文档或网页，为了兼顾准确率与性能，开放问答一般会采用文档检索方法快速筛选出与询问最相关的文档，在筛选出相关文档之后再通过阅读理解的方式给出问题的答案，这种架构被称为检索-阅读理解架构。该架构的基本流程框架如图10.16所示，主要包含文章检索器和文章阅读器两个模块，此外有些算法中还包含对检索得到的文档的后处理以及对抽取答案的后处理。本节中将介绍两种基于该架构的模型。

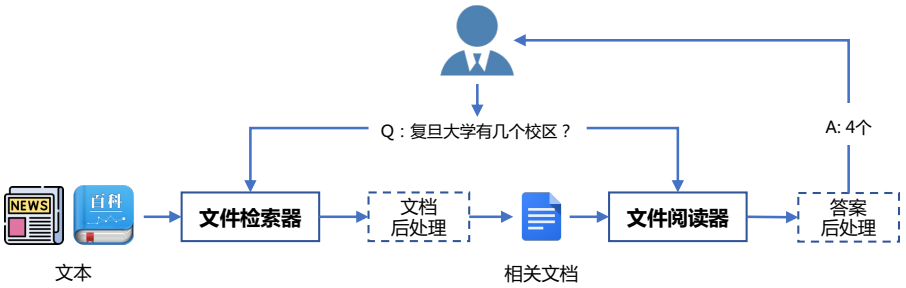

图 10.16　检索-阅读理解架构的开放问答流程框架图[37]

1. DrQA 开放领域问答算法

DrQA[38] 是利用维基百科作为知识库来实现开放式问答系统，DrQA 算法介绍对于任何事实性问题，系统都能从维基百科中找到一个包含答案的文章，在此基础上通过阅读理解算法从相关文章中提取答案。

筛选相关文档的关键是对于一个给定的查询和一个给定的文档，需要定义一个可以衡量该查询与该文档相关性的指标。常用的方法有 TF-IDF 和 BM25，其中，DrQA 采用的是 TF-IDF 的方法。利用 TF-IDF 可以得到文档 d_i 的关键词集合 $Kd = \{kd_1, ..., kd_n\}$ 和问题的关键词集合 $Kq = \{kq_1, ..., kq_m\}$。利用关键词集合可以对问题和文档的相关性进行计算。定义 $K = K_d \cup Kq = \{k_1, ..., k_l\}$，利用 TF-IDF 计算公式可以计算得出 $TF - IDF(k, d)$ 和 $TF - IDF(k, q)$ 的值，从而可以得到向量：

$$\boldsymbol{v}_d = [\text{TF-IDF}(\boldsymbol{k}_1, \boldsymbol{d}), ..., \text{TF-IDF}(\boldsymbol{k}_l, \boldsymbol{d})] \tag{10.75}$$

$$\boldsymbol{v}_q = [\text{TF-IDF}(\boldsymbol{k}_1, \boldsymbol{q}), ..., \text{TF-IDF}(\boldsymbol{k}_l, \boldsymbol{q})] \tag{10.76}$$

DrQA 利用 TF-IDF 的方法表示每个问题，并利用然后利用余弦相似度从大规模语料库中筛选出 5 篇最相关的文章。

答案抽取模块采用多层 RNN 网络，给定一个问题 $q = \{q_1, ..., q_l\}$ 和包含 n 个段落的相关文章，每个段落由 m 个词组成 $p = \{p_1, ..., p_m\}$。对于段落中的每一个词 p_i，DrQA 采用 Glove 词向

量将其映射成一个 300 维的特征向量并把他们输入到 RNN 中并得到上下文向量 $H = h_1, ..., h_m$:

$$\boldsymbol{H} = \text{RNN}(\boldsymbol{p}_1, ..., \boldsymbol{p}_m) \tag{10.77}$$

类似的，DrQA 采用 RNN 网络对问题 q 进行编码得到问题的上下文向量 $H^q = \{h_1^q, ..., h_l^q\}$。

在预测时，DrQA 目标是在 p 中找到一个片段 $\{p_l, ..., p_r\}$ 作为问题的答案。所以，DrQA 将编码好的段落表示向量 H 和问题表示向量 H^q 作为输入，分别训练两个二分类器预测答案的开始位置和结束位置:

$$P_{start}(i) = \exp(\boldsymbol{p}_i \boldsymbol{W}_s \boldsymbol{q}) \tag{10.78}$$

$$P_{end}(i) = \exp(\boldsymbol{p}_i \boldsymbol{W}_e \boldsymbol{q}) \tag{10.79}$$

其中，$\boldsymbol{W}_s$ 和 $\boldsymbol{W}_e$ 是可训练的参数矩阵。最后，DrQA 选择最佳的文章片段 $\{p_l, ..., p_r\}$ 使得 $P_{start}(l) \times P_{end}(r)$ 最大。

2. DPR 开放领域问答算法

上节中介绍的 BM25、TF-IDF 等稀疏向量空间模型，通过词频和逆文档频率，对问题和文档利用关键词进行匹配，无法理解文档和问题中包含的语义信息。例如，对于问题“谁是指环王中的坏人”和文档“萨拉贝克是指环王中最有名的反派角色”，稀疏向量空间模型无法匹配“坏人”和“反派角色”这两个词，但其实在语义角度上，文档和问题是非常相关的。基于密集向量空间的 DPR（Dense Passage Retriever）模型就试图解决上述问题的一种方法。

DPR[39] 模型主要在文章检索模块上做出了改进，DPR 采用了双编码器的结构，可以在 M 个文档中找到与问题 Q 最相关的 K 个文档，DPR 主要分为两个部分：编码和相似度度量模块。DPR 模型的结构如图10.17所示。

在编码部分，DPR 采用了双编码器结构，部署了两个独立的 BERT 模型对问题和文档分别进行编码，将两段文本分别映射到两个 d 维向量中:

$$\boldsymbol{h}_q = \text{BERT}_1(\boldsymbol{q}) \tag{10.80}$$

$$\boldsymbol{h}_p = \text{BERT}_2(\boldsymbol{p}) \tag{10.81}$$

$$\tag{10.82}$$

其中，p 是文档库中的某篇文章，q 是用户提出的问题，两个 BERT 分别独立。紧接着 DPR 采用点积的方式对两个向量的相似度进行度量:

$$\text{sim}(\boldsymbol{q}, \boldsymbol{p}) = \boldsymbol{h}_q^T \cdot \boldsymbol{h}_p \tag{10.83}$$

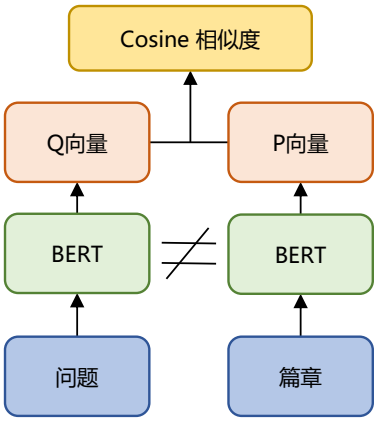

图 10.17 DPR 模型结构图[39]

DPR 的编码模型采用一个正例文档和 n 个负例文档的方式进行训练，整个训练集的定义方式如下：

$$\boldsymbol{D} = < \boldsymbol{q_i}, \boldsymbol{p_i^+}, \boldsymbol{p_{i,1}^-}, ...\boldsymbol{p_{i,n}^-} >_{i=1}^{m} \tag{10.84}$$

其中 $\boldsymbol{q}_i$ 表示第 i 个样本中用户提出的问题，$\boldsymbol{p}_i^+$ 表示该样本中的正确文档，$\boldsymbol{p}_i^-$ 表示第 i 个负例，训练的损失函数如下：

$$\mathcal{L}(\boldsymbol{q}_i, \boldsymbol{p}_i^+, \boldsymbol{p}_{i,1}^-, ..., \boldsymbol{p}_{i,n}^-) = -log\frac{e^{sim(\boldsymbol{q}_i, \boldsymbol{p}_i^+)}}{e^{sim(\boldsymbol{q}_i, \boldsymbol{p}_i^+)} + \sum_{j=1}^{n} e^{sim(\boldsymbol{q}_i, \boldsymbol{p}_{i,j}^-)}} \tag{10.85}$$

在推理阶段，DPR 模型预先将所有文档进行编码并储存，利用 FAISS 向量比较工具对相应的问题进行检索，最终找到与问题最相关的前 K 个文档。之后的答案抽取部分与上文介绍的 DrQA 模型类似，这里就不再赘述。

10.5.2 端到端架构的开放问答模型

随着预训练语言模型的规模越来越大，训练数据越来越多，一些研究人员认为在这些在大规模数据上训练的超大规模语言模型很可能已经建模了网页数据中的知识。基于端到端架构的开放问答模型引起了广泛兴趣。PLSLM[40] 采用小样本提示学习的方式，利用网络数据对超大规模语言模型进行增强，提示学习的方法不需要重新进行预训练或者改变模型结构，可以轻量级的实现模型信息更新。基于小样本提示学习增强大规模语言模型的方法分为三步：

(1) 根据用户提出的问题 q，采用搜索引擎检索一系列相关文档作为增强信息。

(2) 利用检索出的文档，利用提示学习对模型进行调节。

(3) 模型通过每一个检索出的文档生成候选答案，并对这些候选答案进行重排序并选出最合适的

作为最终答案返回。

整体的模型架构如图10.18所示。

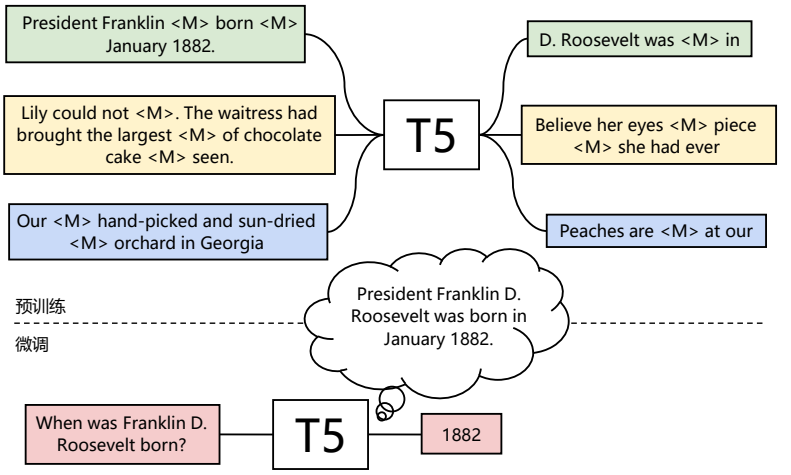

图 10.18 端到端架构开放问答模型（以 T5）为例

对于给定的问题 q, PLSLM 算法利用搜索引擎检索得到前 20 个最相关文本作为模型的增强文档。由于网络上获取的文档会伴随噪音等问题，会增强后续提示学习的难度，所以 PLSLM 对检索文档进行了预处理。将所有检索到的文档分为多个段落，其中每个段落由 6 个句子表示，最后再利用 TF-IDF 算法对所有段落和 q 进行编码，利用余弦相似度进行重排序，将这些排序后的段落作为增强数据。得到经过排序的段落 P 之后，PLSLM 利用小样本提示学习对大规模预训练模型进行调节，PLSLM 针对开放领域问题回答设计了相应的提示模板：

证据: ...

问题: ...

答案: ...

例如：对于训练数据中的问题“复旦大学有几个校区？”，对应的增强文档是复旦大学的百度百科，需要构造如下输入：

证据：复旦大学，简称“复旦”，位于直辖市上海，是中华人民共和国教育部直属的全国重点大学，中央直管高校，综合性研究型大学，由教育部与上海市重点共建。迄 2020 年 4 月，学校有邯郸校区、枫林校区、江湾校区、张江校区四个校区，占地面积约 243.92 万平方米。

问题：复旦大学有几个校区？

答案：

提示学习会通过提示模板引导模型根据问题和证据去生成合适的答案。在对模型进行增强之后，PLSLM 还会利用打分函数对所生成的候选答案在此排序，挑选其中最合适的作为最终答案返

回给用户。针对问题 q 的候选答案 a_i 以及 n 个检索片段 p_i 可以采用如下三种方法选择最合适的答案：

(1) 直接推理：最大化 $P(a|q)$

$$P(a|q) = \sum_{i=1}^{n} P_{\text{TF-IDF}}(p_i|q) \cdot P(a_i|q, p_i) \tag{10.86}$$

(2) 噪声信道模型：最大化 $P(a_i, q|p_i)$

$$P(a_i, q|p_i) = \frac{P(q|a_i, p_i) \cdot P(a_i|p_i)}{P(q|p_i)} \tag{10.87}$$

(3) 专家乘积（Product of experts，PoE）：使用额外的语料训练权重综合上述概率

由于提示学习不需要重新进行预训练以及不改变模型架构的特质，这类方法可以适用于所有语言模型。例如，T5[41] 模型是一个文本-文本的大规模预训练生成模型，提出了一个统一的模型框架，将所有自然语言处理的任务都视为文本到文本的生成任务，即输入是文本，输出也是文本的任务。T5 模型将翻译、分类、回归、摘要生成等任务都统一成了文本到文本任务，使得所有任务在预训练和微调的过程中都能使用同样的目标函数进行训练，使用同样的解码过程进行推理。

T5 模型采用的是编码器-解码器结构，在预训练过程中，采用了噪音扰动的方式，BART 预训练模型同样采用了这种预训练方式，对于这种预训练方式在摘要生成章节进行了详细的描述，这里就不再赘述。对于训练中的输入，用一些特殊符号例如 $< X >$ 来表示原始样本中被随机遮盖的片段或词缀，希望模型生成的目标样本则是没有被遮盖过的原始样本，通过这种方式对 T5 模型进行预训练。

具体到开放问答任务，对每一个输入问题 Q, 可以通过提示学习的方式对 T5 模型进行微调，从而使 T5 模型可以直接生成问题对应的答案 A：

$$\boldsymbol{A} = \text{T5}(\boldsymbol{Q}) \tag{10.88}$$

通过提示学习利用大规模语言模型直接进行开放问答的方式成为近年来开放问答任务上的一个新范式。随着规模更大，预训练更充分的生成式模型（如 GPT-3）不断涌现，这种端到端架构的开放问答模型也逐渐成为研究热点。

10.5.3 开放领域问答语料库

目前常用的开放问答语料库如表格10.5所示。开放问答评测集合的语料规模通常较大，SQuAD$_{open}$、Natural Questions 以及 XQA 利用的是维基百科作为基础资源库，SearchQA 采用搜索引擎的返回文档结果作为支持文本。

表 10.5 开放领域问答常用语料库

数据集	知识来源	数据集规模
SQuAD$_{open}$	维基百科	97K
Natural Questions	维基百科	323K
XQA	维基百科	90K
SearchQA	搜索引擎	140K

1. SQuAD$_{open}$

由斯坦福大学构建的 SQuAD 数据集极大的阅读理解的发展，在很多应用中也采用其训练和验证开放问答。SQuAD$_{open}$[38] 扩展自 SQuAD 数据集，在应用于开放问答时，使用整个维基百科数据集合作为问答资源，不再提供问题所对应的文章段落。由于 SQuAD 评测不公开测试集合，因此在应用于开放问答场景评测时，使用开发集用于测试。与 SQuAD$_{open}$ 扩展自 SQuAD 类似，包括 MS-MARCO、HotpotQA 等在内的用于阅读理解评测的语料集合常通过不提供答案所在文章，从而应用于开放领域问答评测。

2. Natural Questions

Natural Questions[42] 数据集由谷歌公司发布，旨在解决目前开放问答方法的局限性。Natural Questions 提供 307,373 条训练数据，7830 条验证集数据以及 7842 条测试数据。对于每一条数据，包含一个问题和一个简短答案以及包含答案的维基百科界面。Natural Questions 数据集是开放问答领域重要的数据集。

3. XQA

XQA[43] 是清华大学构建的跨语言开放领域问答评测集合，提供了包含 56279 英语问题答案对的训练集合，测试集合包含中文、英文、法语、德语等在内的 9 种语言开放领域问题和答案，开发集合规模 17358 问答对，测试集合规模为 16973 个问答对。

4. SearchQA

为了模拟人们在回答问题时的一般流程，SearchQA[44] 采用了与之前问答评测集合构建不同的方法，首先收集问题答案对，在此基础利用 J! Archive 以及谷歌搜索引擎返回片段构建支持文本。针对收集的 14 万问题答案对，每个收集了约 49.6 个支持文本片段。

10.6 延伸阅读

从基于规则和基于统计的模型到如今基于神经网络的模型，智能问答任务经历多年的发展取得了很大的进步，随着预训练语言模型和深度学习的兴起，在很多智能问答问题上都已经能实现很好的效果。尽管如此，智能问答领域中还有很多问题没有解决，包括多跳问答问答、智能问答

系统解析生成、多模态智能问答等。

在多跳问答方面，虽然现在的深度学习模型在传统的抽取式阅读理解数据集上取得了很好的表现，但现有模型在多跳问答数据集上还有所欠缺。多跳问答即数据集中的问题，需要多次“跳转”的阅读理解才能回答。具体来说，给定一个问题，系统只通过一个文档是无法正确回答问题的，需要系统根据多篇文档回答一个问题，所以需要多跳推理。现有的工作基于图神经网络[45, 46]，通过对抗学习[47], 证据检索 [48] 等方式试图解决多跳问答任务。

智能问答系统解析生成方面，虽然面对大部分数据集，基于预训练的智能问答模型都可以给出正确的答案，但是这些答案都不可解释，不能给出一个思维链去解释得到答案的过程，而这种可解释性的缺失也制约了智能问答系统的进一步发展。研究人员通过构建模型选择支持事实[16]，生成蕴含树[49–51] 等方式进行了一些研究。如何利用潜在的常识，充分挖掘文档信息，生成推导逻辑链，是智能问答乃至人工智能的重要发展方向之一。

在多模态智能问答方面，目前的智能问答主要针对的是纯文本数据，但真实场景中的文档具有多样性的布局和丰富的信息。此外，很多网页以及社交媒体包含丰富的多模态信息。针对多模态智能问答，研究人员提出了多种方案试图对不同形式的信息进行融合，包括图融合，图卷机，结构化等方面进行了一系列研究 [52–55]。如何充分利用图片、视频、文本中包含的丰富信息，是智能问答的重要发展方向之一。

10.7 习题

（1）阅读理解模型有哪些信息提取方式？你认为基于深度学习的阅读理解模型包含哪些缺陷？

（2）基于 SQuAD 数据集，复现 R-Net 模型，正确率差距不能超过 5%。

（3）试比较 R-Net 模型和 BiDAF 模型的异同点。

（4）你认为 SpanBERT 模型解决了什么样的问题，除了本章节所介绍的阅读理解任务，你认为 SpanBERT 还适用于哪些自然语言处理任务，并说出原因。

（5）你认为端到端架构的开放问答模型相比传统的基于检索的开放问答模型有哪些可能的优势，为什么当前端到端架构的模型效果不如基于检索的模型。

（6）你认为 ChatGPT 模型可以看作是一种开放问答模型吗？它与传统的开放问答模型相比有哪些优点及其原因？

参考文献

[1] Turing A M. Computing machinery and intelligence[M]//Parsing the turing test. Springer, 2009: 23-65.

[2] Green Jr B F, Wolf A K, Chomsky C, et al. Baseball: an automatic question-answerer[C]//Papers presented at the May 9-11, 1961, western joint IRE-AIEE-ACM computer conference. 1961: 219-224.

[3] Voorhees E M, et al. The trec-8 question answering track report[C]//Trec: volume 99. 1999: 77-82.

[4] Woods W A. Progress in natural language understanding: an application to lunar geology[C]// Proceedings of the June 4-8, 1973, national computer conference and exposition. 1973: 441-450.

[5] Winograd T. Procedures as a representation for data in a computer program for understanding natural language[R]. MASSACHUSETTS INST OF TECH CAMBRIDGE PROJECT MAC, 1971.

[6] Warren D H, Pereira F C. An efficient easily adaptable system for interpreting natural language queries[J]. American journal of computational linguistics, 1982, 8(3-4):110-122.

[7] Wilensky R. The berkeley unix consultant project[M]//Wissensbasierte Systeme. Springer, 1987: 286-296.

[8] Lehnert W G. A conceptual theory of question answering[C]//Proceedings of the 5th international joint conference on Artificial intelligence-Volume 1. 1977: 158-164.

[9] Katz B. Using english for indexing and retrieving[R]. MASSACHUSETTS INST OF TECH CAMBRIDGE ARTIFICIAL INTELLIGENCE LAB, 1988.

[10] Katz B. From sentence processing to information access on the world wide web[C]//AAAI Spring Symposium on Natural Language Processing for the World Wide Web: volume 1. Stanford University Stanford, CA, USA, 1997: 997.

[11] Katz B, Felshin S, Lin J, et al. Viewing the web as a virtual database for question answering.[C]// New Directions in Question Answering. 2004: 215-226.

[12] Ittycheriah A, Franz M, Roukos S. Ibm's statistical question answering system - trec-10[C]//TREC. 2001.

[13] Ferrucci D, Brown E, Chu-Carroll J, et al. Building watson: An overview of the deepqa project[J]. AI magazine, 2010, 31(3):59-79.

[14] 许静芳. 搜狗汪仔的"大梦想"[J]. 中国计算机学会通讯, 2007, 13(5).

[15] Rajpurkar P, Zhang J, Lopyrev K, et al. Squad: 100,000+ questions for machine comprehension of text[C]//Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing. 2016: 2383-2392.

[16] Yang Z, Qi P, Zhang S, et al. Hotpotqa: A dataset for diverse, explainable multi-hop question answering[C]//Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. 2018: 2369-2380.

[17] Reddy S, Chen D, Manning C D. Coqa: A conversational question answering challenge[J]. Transactions of the Association for Computational Linguistics, 2019, 7:249-266.

[18] Saha A, Pahuja V, Khapra M, et al. Complex sequential question answering: Towards learning to converse over linked question answer pairs with a knowledge graph[C]//Proceedings of the AAAI Conference on Artificial Intelligence: volume 32. 2018.

[19] Nan L, Hsieh C, Mao Z, et al. Fetaqa: Free-form table question answering[J]. Transactions of the Association for Computational Linguistics, 2022, 10:35-49.

[20] 段楠, 周明. 智能问答[M]. 北京: 高等教育出版社, 2018.

[21] Hirschman L, Light M, Breck E, et al. Deep read: A reading comprehension system[C]//Proceedings of the 37th annual meeting of the Association for Computational Linguistics. 1999: 325-332.

[22] Xu K, Meng H. Using verb dependency matching in a reading comprehension system[C]//Asia Information Retrieval Symposium. Springer, 2004: 190-201.

[23] Pasupat P, Liang P. Compositional semantic parsing on semi-structured tables[C]//Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). 2015: 1470-1480.

[24] Wang W, Yang N, Wei F, et al. Gated self-matching networks for reading comprehension and question answering[C]//Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2017: 189-198.

[25] Riloff E, Thelen M. A rule-based question answering system for reading comprehension tests[C]// ANLP-NAACL 2000 Workshop: Reading Comprehension Tests as Evaluation for Computer-Based Language Understanding Systems. 2000.

[26] Riloff E, Phillips W. An introduction to the sundance and autoslog systems[R]. Technical Report UUCS-04-015, School of Computing, University of Utah, 2004.

[27] Xu K, Meng H, Weng F. A maximum entropy framework that integrates word dependencies and grammatical relations for reading comprehension[C]//Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers. 2006: 185-188.

[28] Seo M, Kembhavi A, Farhadi A, et al. Bidirectional attention flow for machine comprehension[J]. arXiv preprint arXiv:1611.01603, 2016.

[29] Pennington J, Socher R, Manning C. Glove: Global vectors for word representation[C]//Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). 2014: 1532-1543.

[30] Srivastava R K, Greff K, Schmidhuber J. Highway networks[J]. arXiv preprint arXiv:1505.00387, 2015.

[31] Joshi M, Chen D, Liu Y, et al. Spanbert: Improving pre-training by representing and predicting spans [J]. Transactions of the Association for Computational Linguistics, 2020, 8:64-77.

[32] Devlin J, Chang M W, Lee K, et al. Bert: Pre-training of deep bidirectional transformers for language understanding[C]//Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). 2019: 4171-4186.

[33] Nguyen T, Rosenberg M, Song X, et al. Ms marco: A human generated machine reading comprehension dataset[C]//CoCo@ NIPs. 2016.

[34] Pan F, Canim M, Glass M, et al. Cltr: An end-to-end, transformer-based system for cell level table retrieval and table question answering[J]. arXiv preprint arXiv:2106.04441, 2021.

[35] 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. 2013: 2333-2338.

[36] Chen Q, Zhu X, Ling Z H, et al. Enhanced lstm for natural language inference[C]//Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2017: 1657-1668.

[37] Zhu F, Lei W, Wang C, et al. Retrieving and reading: A comprehensive survey on open-domain question answering[J]. arXiv preprint arXiv:2101.00774, 2021.

[38] Chen D, Fisch A, Weston J, et al. Reading wikipedia to answer open-domain questions[C]// Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2017: 1870-1879.

[39] Karpukhin V, Ouguz B, Min S, et al. Dense passage retrieval for open-domain question answering [J]. arXiv preprint arXiv:2004.04906, 2020.

[40] Lazaridou A, Gribovskaya E, Stokowiec W, et al. Internet-augmented language models through few-shot prompting for open-domain question answering[J]. arXiv preprint arXiv:2203.05115, 2022.

[41] Roberts A, Raffel C, Shazeer N. How much knowledge can you pack into the parameters of a language model?[J]. arXiv preprint arXiv:2002.08910, 2020.

[42] Kwiatkowski T, Palomaki J, Redfield O, et al. Natural questions: a benchmark for question answering research[J]. Transactions of the Association for Computational Linguistics, 2019, 7:453-466.

[43] Liu J, Lin Y, Liu Z, et al. Xqa: A cross-lingual open-domain question answering dataset[C]// Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. 2019: 2358-2368.

[44] Dunn M, Sagun L, Higgins M, et al. Searchqa: A new q&a dataset augmented with context from a search engine[J]. arXiv preprint arXiv:1704.05179, 2017.

[45] Fang Y, Sun S, Gan Z, et al. Hierarchical graph network for multi-hop question answering[J]. arXiv preprint arXiv:1911.03631, 2019.

[46] Ding M, Zhou C, Chen Q, et al. Cognitive graph for multi-hop reading comprehension at scale[J]. arXiv preprint arXiv:1905.05460, 2019.

[47] Jiang Y, Bansal M. Avoiding reasoning shortcuts: Adversarial evaluation, training, and model development for multi-hop qa[J]. arXiv preprint arXiv:1906.07132, 2019.

[48] Nishida K, Nishida K, Nagata M, et al. Answering while summarizing: Multi-task learning for multi-hop qa with evidence extraction[J]. arXiv preprint arXiv:1905.08511, 2019.

[49] Dalvi B, Jansen P, Tafjord O, et al. Explaining answers with entailment trees[J]. arXiv preprint arXiv:2104.08661, 2021.

[50] Ribeiro D, Wang S, Ma X, et al. Entailment tree explanations via iterative retrieval-generation reasoner[J]. arXiv preprint arXiv:2205.09224, 2022.

[51] Hong R, Zhang H, Yu X, et al. Metgen: A module-based entailment tree generation framework for answer explanation[J]. arXiv preprint arXiv:2205.02593, 2022.

[52] Kembhavi A, Seo M, Schwenk D, et al. Are you smarter than a sixth grader? textbook question answering for multimodal machine comprehension[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern recognition. 2017: 4999-5007.

[53] Huang Z, Liu F, Wu X, et al. Audio-oriented multimodal machine comprehension via dynamic inter- and intra-modality attention[C]//Proceedings of the AAAI Conference on Artificial Intelligence: volume 35. 2021: 13098-13106.

[54] Yagcioglu S, Erdem A, Erdem E, et al. Recipeqa: A challenge dataset for multimodal comprehension of cooking recipes[J]. arXiv preprint arXiv:1809.00812, 2018.

[55] Mai S, Hu H, Xu J, et al. Multi-fusion residual memory network for multimodal human sentiment comprehension[J]. IEEE Transactions on Affective Computing, 2020, 13(1):320-334.

索引