文章编号: 1007-5321(2019)06-0001-09 DOI: 10.13190/j.jbupt.2019-111

机器阅读理解的研究进展

王小捷， 白子薇， 李 可， 袁彩霞
(北京邮电大学 计算机学院，北京 100876)

摘要: 为便于厘清机器阅读理解任务的研究现状，按照答案来源，将机器阅读理解分为完形填空、片段选择、多项选择和答案生成 4 类. 在统一的编码器—交互与推理—输出框架下对此 4 类任务的已有研究进行了综述，并描述了 2 种对此框架的可能扩展; 最后讨论了机器阅读理解未来需要解决的问题.

关 键 词: 机器阅读理解; 编码器; 交互; 注意力机制
中图分类号: TP18 **文献标志码:** A **OSID 码:**

Survey on Machine Reading Comprehension

WANG Xiao-jie， BAI Zi-wei， LI Ke， YUAN Cai-xia
(School of Computer Science, Beijing University of Posts and Telecommunications, Beijing 100876, China)

Abstract: In order to make clear the recent work of machine reading comprehension (MRC) tasks, they are divided into four types of subtasks with different sources of answers. They are cloze-style, span selection, multi-choice, and answer generation. Previous work on these four different types of subtasks is investigated under a unified framework of encoder-interaction and reasoning-output. Two types of recent developments for the frame are also given. Several challenges on MRC for future work are discussed at the end of the survey.

Key words: machine reading comprehension; encoder; interaction; attention mechanism

阅读是人类获取新知识的重要方式之一，中国古代有句老话“书读百遍其义自见”，就是强调了阅读的重要性. 教会机器阅读与理解人类的语言是自然语言处理领域的一个重要任务. 因此，人们提出将机器阅读理解(MRC，machine reading comprehension)用于评估机器对人类语言的理解能力，该任务要求机器在阅读自由格式的篇章后回答与篇章相关的问题[1].

机器阅读理解任务针对篇章理解是自然语言处理技术发展的必然. Kamp 等[2]认为，篇章是自然语言理解的基本单位. 篇章中包含的信息丰富多样，基于篇章的问答，不仅需要对问题进行理解，还需要对完整的篇章进行理解并生成答案. 问题类型不同，需要的处理方式也有所不同，有时需要对篇章的情感进行分析，有时需要对主旨进行总结，有时需要对篇章中相关联的事件进行分析，因此，机器阅读理解需要综合运用多种自然语言处理技术来回答各种问题.

1 任务定义

阅读理解任务由一个四元组 $\langle P,Q,C,A\rangle$ 定义. P 为篇章，由一个或多个段落组成; Q 为与 P 相关的

收稿日期: 2019-06-11
基金项目: 国家自然科学基金项目(61906018)
作者简介: 王小捷(1969—)，男，教授.
通信作者: 白子薇(1994—)，女，博士生，E-mail: bestbzw@bupt.edu.cn.

问题; C 是一个集合,包含正确答案和干扰答案,数量一般为4~6个; A 是 Q 的正确答案,是 C 的一个子集. 由四元组 $\langle P,Q,C,A\rangle$ 定义的任务为基于 P 从 C 中挑选出问题 Q 的正确答案. 有时候,任务没有候选答案集 C,此时任务由三元组 $\langle P,Q,A\rangle$ 定义,要求基于 P 生成 Q 的答案 A.

现有的阅读理解任务一般可以概括为4类: 完形填空、多项选择、片段选择和答案生成. 前三类属于四元组 $\langle P,Q,C,A\rangle$ 定义的类型,最后一类属于三元组 $\langle P,Q,A\rangle$ 定义的类型. 以下分别叙述这4类任务.

1.1 完形填空

给定一个篇章 P 以及一个缺失某个词的句子 Q,完形填空的任务是推测出 Q 中的缺失词. 缺失词大多是实体词,不同问题的缺失词来源不同,有的来源于 P,有的来源于候选答案集合 C.

完形填空类数据集是最早出现的大规模阅读理解数据集,因为这类数据可以直接通过规则获取,无需人工标注问题答案,但这类问题比较容易回答.

第1个大规模完形填空数据集是 Hermann 等[3]提出的 CNN/Daily Mail 数据集,其中 CNN 数据集包含9万个篇章和38万个问题,Daily Mail 数据集包含20万个篇章和90万个问题,这使得数据驱动的方法成为可能. 在 CNN/Daily Mail 被提出前,通常使用基于规则或特征工程的方法来完成阅读理解任务. 此类任务通过正确率或准召率进行评估.

1.2 多项选择

多项选择任务是从多个候选答案中为问题选择正确答案. 目前的数据通常来源于试卷中真实的阅读理解题,问题和答案候选集由专家手工进行构建,候选答案通常为4个. 此类任务的目的在于评估人类的阅读理解能力和推理能力,因此在回答问题时会涉及多种不同类型的推理. 代表数据集是由 CMU 提出的 RACE[4](ReAding comprehension dataset from examinations). RACE 数据集来源于中国12~18岁中学生的英文试卷,包括2.8万个篇章和将近10万个由人类专家构建的问题,涵盖多个领域的主题和多种不同类型的篇章,如新闻、故事、广告、传记. 主题与类型的多样性能更好地评估人或机器的阅读理解能力. 多项选择任务一般采用正确率来评估预测结果.

1.3 片段选择

片段选择类阅读理解任务的答案是篇章中的一个片段,可能是一个词、一个词组,也可能是一个句子. 代表性的数据集为 SQuAD[5],被称为机器阅读理解界的 ImageNet. 数据集包含10万+个问题,由众包人员对维基百科标注得到. 在 SQuAD 数据集上的工作较多,也取得了较好的效果,甚至超过了人类在该数据集上的评测结果[6]. Maluuba 研究院提出了一个更具有挑战性的数据集 NewsQA[7],问题由众包人员根据 CNN 的1万篇新闻标注得到,共获得10万个问题-答案对. 此类任务通过精确匹配率(EM 值)[5]和模糊匹配率(F1 值)进行评估.

1.4 答案生成

答案生成类任务中,答案需要模型生成,因为它可能是一个未在篇章中出现的单词、词组或句子. 这类问题不仅需要保证答案正确性,符合逻辑,也需要保证答案连贯,符合语法. 与前面3种类型的任务相比,生成类任务搜索空间更大,更难解决,其中最具有代表性的数据集是微软提出的 MS-MARCO 数据集[8]. MS-MARCO 数据集是基于搜索引擎 Bing 构建的大规模英文阅读理解数据集,包含10万个问题和20万个篇章. MS-MARCO 数据集的答案没有标明来自哪个篇章,其中有一部分问题无法直接在篇章中找到答案. 2017年,百度公司提出了一个类似的中文数据集 Du-Reader[9],这是第1个在中文领域包含生成类问题的大规模机器阅读理解数据集. 此类任务一般采用 BLEU 值或 Rouge-L 值[10]进行评估.

2 传统方法

早期的 MRC 系统采用基于规则的方法. 1999年,Hirschman 等[11]提出 Deep Read 系统,可以自动读取一个故事并回答相应的问题. Deep Read 系统采用了信息抽取的方法,分别将问题和故事中的关键信息抽取出来,然后采用匹配的方式从故事中搜索出问题查询的信息. 该系统使用词袋模型对句子信息进行表示.

2000年,Riloff 等[12]提出一个基于规则的阅读理解问答系统 Quarc,可以接收一个故事并且挑选出一个最合适的句子作为相应问题的答案. Quarc 可以通过启发式规则查找故事和问题中的词汇和语义线索. 在中文领域,2007年,Hao 等[13]提出一个基于规则的中文阅读理解问答系统 Cqarc. 与 Quarc 类似,Cqarc 也是通过启发式规则查找故事和问题中的词汇和语义线索给出答案.

基于规则或手工构建特征的方式依赖人工,模型迁移能力差. 随着大规模阅读理解数据的出现,数据驱动的方法逐渐占据了主导位置,基于深层神经网络的端到端模型成为机器阅读理解研究的主流方法. 下一节将在一个基本模型框架下叙述这些方法,在第 4 节介绍在此基本框架下的扩展.

3　基于深度学习的方法

大多数端到端的深度学习模型都采用"编码器-推理与交互-输出"的框架来解决阅读理解问题,其框架结构如图 1 所示.

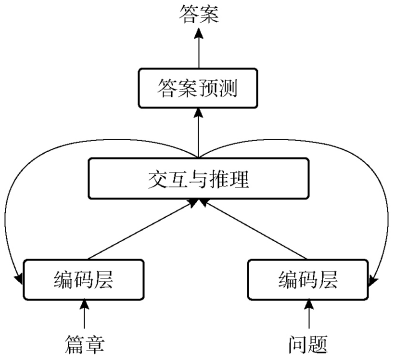

图 1　"编码器-交互与推理-输出"框架

将篇章 P 和问题 Q 看做 2 个词序列,P 是一个长度为 L_p 的序列$\{w_1^p, w_2^p, \cdots, w_{L_p}^p\}$,$Q$ 是一个长度为 L_q 的序列$\{w_1^q, w_2^q, \cdots, w_{L_q}^q\}$. 通过编码器将 P 与 Q 分别进行编码,并将编码后的结果输入"交互与推理"模块,通过问题信息与篇章信息的交互,定位到篇章中与答案相关的片段并进行推理,得到一个综合了问题与篇章信息的表示,最后通过答案预测模块选择或生成一个正确答案.

3.1　编码器

多数机器阅读理解任务通常选择循环神经网络(RNN,recurrent neural network)作为编码器[3,14-17]. 将 P 和 Q 分别通过一个 RNN,得到对应的输出 $\boldsymbol{H}^p = [\boldsymbol{h}_1^p, \boldsymbol{h}_2^p, \cdots, \boldsymbol{h}_{L_p}^p]^{\mathrm{T}}$ 和 $\boldsymbol{H}^q = [\boldsymbol{h}_1^q, \boldsymbol{h}_2^q, \cdots, \boldsymbol{h}_{L_q}^q]^{\mathrm{T}}$. $\boldsymbol{h}_i$(代表 $\boldsymbol{h}_i^p$ 或 $\boldsymbol{h}_i^q$)是第 i 个词对应的隐层状态,$\boldsymbol{h}_i \in \mathbf{R}^d$,$d$ 是 RNN 的隐层大小. 由上一时刻的隐层状态 $\boldsymbol{h}_{i-1}$ 与当前输入 w_i 对应的 embedding 向量 $\boldsymbol{e}_i$ 得到.

$$\boldsymbol{h}_i = \mathrm{RNN}(\boldsymbol{h}_{i-1}, \boldsymbol{e}_i) \tag{1}$$

$\boldsymbol{H}^p$ 是一个 $L_p \times d$ 大小的矩阵,$\boldsymbol{H}^q$ 是一个 $L_q \times d$ 大小的矩阵.

为了同时获得上下文的信息,目前也常用双向 RNN 作为编码器. $\overrightarrow{\boldsymbol{H}} = [\overrightarrow{\boldsymbol{h}}_1, \overrightarrow{\boldsymbol{h}}_2, \cdots]^{\mathrm{T}}$ 代表前向 RNN 对应的隐藏层状态序列,$\overleftarrow{\boldsymbol{H}} = [\overleftarrow{\boldsymbol{h}}_1, \overleftarrow{\boldsymbol{h}}_2, \cdots]^{\mathrm{T}}$ 代表后向 RNN 对应的隐藏层状态序列. 双向 RNN 的隐藏层状态由 $\overrightarrow{\boldsymbol{H}}$ 和 $\overleftarrow{\boldsymbol{H}}$ 表示为

$$\begin{aligned} \overrightarrow{\boldsymbol{h}}_i &= \mathrm{RNN}(\overrightarrow{\boldsymbol{h}}_{i-1}, \boldsymbol{e}_i) \\ \overleftarrow{\boldsymbol{h}}_i &= \mathrm{RNN}(\overleftarrow{\boldsymbol{h}}_{i+1}, \boldsymbol{e}_i) \\ \boldsymbol{h}_i &= [\overrightarrow{\boldsymbol{h}}_i; \overleftarrow{\boldsymbol{h}}_i] \end{aligned} \tag{2}$$

通过双向 RNN 对篇章 P 和问题 Q 进行编码得到 P 和 Q 的表示:

$$\begin{aligned} \boldsymbol{H}^p &= [\overrightarrow{\boldsymbol{H}}^p; \overleftarrow{\boldsymbol{H}}^p] \\ \boldsymbol{H}^q &= [\overrightarrow{\boldsymbol{H}}^q; \overleftarrow{\boldsymbol{H}}^q] \end{aligned} \tag{3}$$

有的模型中,问题 Q 用 RNN 的最后一个隐藏层状态 $\boldsymbol{h}_{L_q}^q$ 表示,记为 $\boldsymbol{u}$,称为 Q 对应的语义向量.

3.2　交互与推理

交互与推理是一个比较概括形象的说法. 交互主要指在篇章信息与问题信息之间的关联操作,而推理主要指由已有信息产生新信息的过程.

在阅读理解任务中,篇章 P 一般包含上百个词,通常情况下,篇章中只有部分内容与问题相关. 直接使用 RNN 的最后一个隐藏层状态表示整个篇章不能很好地突出与问题相关的信息,因此在选择或生成答案时,可以综合考虑问题与篇章的信息,通过注意力机制生成一个问题感知的篇章信息表示.

假设一个序列$\{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots\}$,每一个 $\boldsymbol{x}_i$ 都有 $\boldsymbol{x}_i \in R^{d_x}$,计算 $\boldsymbol{x}_i$ 与向量 $\boldsymbol{y}$ 的相似度 $m(\boldsymbol{x}_i, \boldsymbol{y})$,并进行 softmax 操作,可以获得 $\boldsymbol{y}$ 对整个序列的注意力权重:

$$s(\boldsymbol{x}_i, \boldsymbol{y}) \propto \exp(m(\boldsymbol{x}_i, \boldsymbol{y})) \tag{4}$$

也可以通过此方法获得向量 $\boldsymbol{x}$ 对序列$\{\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots\}$的注意力权重.

3.2.1　完形填空与多项选择

在完形填空和选择类问题中,一些研究通过注意力机制将篇章编码为一个固定维度的向量,并基于此进行答案的预测[3-4],也有一些完形填空类任务直接使用注意力机制获得的权重作为候选答案的概率分布[14,16]. 一种常见的注意力方法是使用整个问题的向量表示来引导问题与篇章中每个词的相关度[3-4,16-17],如

$$\boldsymbol{r} = \sum_{i=1}^{L_p} s(\boldsymbol{h}_i^p, \boldsymbol{u})\, \boldsymbol{h}_i^p \tag{5}$$

其中:$s(\boldsymbol{h}_i^p, \boldsymbol{u})$ 是问题向量 $\boldsymbol{u}$ 对篇章表示 $\boldsymbol{H}^p$ 中每个向量的注意力权重,$\boldsymbol{r}$ 是对 $\boldsymbol{H}^p$ 进行加权求和后得到的最终篇章向量表示.

这种注意力称为“一维注意力模型”. 一维注意力模型方法关注问题序列的整体对篇章的注意力，忽略了问题中不同词对注意力机制的影响.

为此，一些随后的工作开始关注问题中每个词对篇章的权重. 具体的方法是: 计算问题中每个词对篇章的注意力，得到 L_q 组注意力权重(L_q 为问题长度)，再通过一定的机制综合这 L_q 组权重信息和篇章表示 $\boldsymbol{H}^p$，得到最终的篇章向量. Hermann 等[3]提出的 Impatient Reader 模型使用线性方法计算问题中每个词与篇章中每个词的相似度，得到一个相似度矩阵 $\boldsymbol{M} \in R^{L_p \times L_q}$，$\boldsymbol{M}[i,j]$ 为篇章中第 i 个词与问题中第 j 个词的相似度，$\boldsymbol{M}[i,j] = m(\boldsymbol{h}_i^p, \boldsymbol{h}_j^q)$. 进而通过 softmax 函数得到问题中每个词对篇章的注意力权重 $s(\cdot, \boldsymbol{h}_j^q)$. 问题中第 j 个词对应的篇章向量为 $\boldsymbol{r}(j)$，$\boldsymbol{r}(j)$ 由当前词对应的注意力权重 $s(\cdot, \boldsymbol{h}_j^q)$ 和第 $j-1$ 个词对应的篇章向量 $\boldsymbol{r}(j-1)$ 获得，如

$$\boldsymbol{r}(j) = \begin{cases} r_0, & j = 1 \\ \sum_{i=1}^{l} s(\boldsymbol{h}_i^p, \boldsymbol{h}_j^q)\, \boldsymbol{h}_i^p + \tanh(\boldsymbol{W}_r \boldsymbol{r}(j-1)), & 1 < j \leqslant L_q \end{cases} \tag{6}$$

最终的篇章向量 $\boldsymbol{r}$ 为问题中最后一个词对应的篇章向量 $\boldsymbol{r}(L_q)$.

与 Impatient Reader 模型类似的还有 CAS (consensus attention sum reader) 模型[18] 和 AoA (attention-over-attention) 模型[14].

CAS 在计算最终的篇章向量 $\boldsymbol{r}$ 时，没有采用逐步计算篇章向量的方法，而是先将问题中每个词对篇章的 L_q 个相似度向量 $\boldsymbol{M}[i,j]$ $(j = 1, 2, \cdots, L_q)$ 进行综合得到一个对篇章的权重向量 $\boldsymbol{S} \in R^{L_p}$，并使用 $\boldsymbol{S}$ 对 $\boldsymbol{H}^p$ 进行加权求和，得到篇章向量 $\boldsymbol{r}$. 这里尝试用 3 种不同的模式来获得权重表示 $\boldsymbol{S}$，如

$$\boldsymbol{S} \propto \begin{cases} \exp\left(\sum_{j=1}^{L_q} \boldsymbol{M}[i,j]\right) \\ \exp\left(\frac{1}{L_q}\sum_{j=1}^{L_q} \boldsymbol{M}[i,j]\right) \\ \exp\left(\max_{j=1,2,\cdots,L_q} \boldsymbol{M}[i,j]\right) \end{cases} \tag{7}$$

在一个问题中，不同词所含有的信息量是不同的，有的词对于答案预测有重要的作用，有的词则可有可无; 同时，不同的单词所关注的篇章部分也可能不同. 例如，“What is Wendy's favorite sport?”，“is”所含有的信息量就低于其他单词. “Wendy”与篇章中的人称或人称代词相关度大，“sport”与某些运动名称相关度更大. 显然，上述的几种方法没有建模这种差异. AoA 模型通过两层注意力叠加的方式对现象建模. 首先通过与 CAS 模型相似的方法得到问题与篇章的相似度矩阵 $\boldsymbol{M}$，然后分别计算问题中每个词对篇章的注意力权重向量 $\boldsymbol{\alpha}(j) \in R^{L_p}$，$j \in [1, L_q]$ 和篇章中每个单词对问题的注意力权重向量 $\boldsymbol{\beta}(i) \in R^{L_q}$，$i \in [1, L_p]$:

$$\begin{aligned} \boldsymbol{\alpha}(j) &\propto \exp(\boldsymbol{M}[:,j]) \\ \boldsymbol{\beta}(i) &\propto \exp(\boldsymbol{M}[i,:]) \end{aligned} \tag{8}$$

其中: $\boldsymbol{M}[:,j]$ 是矩阵 $\boldsymbol{M}$ 对应的第 j 个列向量，$\boldsymbol{M}[i,:]$ 是矩阵 $\boldsymbol{M}$ 对应的第 i 个行向量. 令 $\boldsymbol{\beta} = \frac{1}{L_p}\sum_{i=1}^{L_p} \boldsymbol{\beta}(i) \in R^{L_q}$ 为问题层次的注意力权重向量，用于指明问题中哪些单词更加“重要”.

$\boldsymbol{\alpha} = [\boldsymbol{\alpha}(1), \boldsymbol{\alpha}(2), \cdots, \boldsymbol{\alpha}(n)]$ 为篇章层次的注意力权重矩阵，矩阵中每一列代表问题中相应位置的词对篇章的注意力权重; 然后对 $\boldsymbol{\alpha}$ 与 $\boldsymbol{\beta}$ 进行点乘操作，得到加权后的篇章注意力权重 $\boldsymbol{S} \in R^{L_p}$，最后得到篇章表示 $\boldsymbol{r}$.

$$\begin{aligned} \boldsymbol{S} &= \boldsymbol{\alpha}\,\boldsymbol{\beta} \\ \boldsymbol{r} &= \boldsymbol{S}\,\boldsymbol{H}^p \end{aligned} \tag{9}$$

$\boldsymbol{r}$ 是篇章的整体表示，通常使用 $\boldsymbol{r}$ 或篇章的权重信息 $\boldsymbol{S}$ 作为答案预测层的输入.

这种考虑问题中每个词对篇章注意力操作的方法为“二维注意力模型”. 而上述这些问题与篇章只进行一次交互的结构为“单跳结构”，与之相对应的是“多跳结构”.

在阅读理解任务中，许多问题不能直接从篇章中找到答案，而是需要多步推理才能找到正确答案. 端到端记忆网络[19]中给出的例子如图 2 所示.

Brian is a frog.
Lily is gray.
Brian is yellow.
Julius is green
Greg is a frog.

图 2 一个需要多步推理的例子

如果想知道“Greg 是什么颜色的”，需要首先知道“Greg”是一个“frog”，然后知道“Brian”也是一个“frog”，推理出“Greg”和“Brian”具有相同的颜色. 再根据“Brian is yellow”推理出“Greg”是“yellow”的. 在这个推理过程中，要不断变化注意力的对象. 端到端记忆网络用一种多跳结构来实现这种操作.

对于任意第 k 层,模型使用词袋模型作为编码器获得篇章中第 i 个句子的向量 a_i^k 以及每个句子相应的输出向量 c_i^k. 模型一共有 K 层(跳),每一层都采用类似的注意力机制来获取篇章的表示 o^k,如

$$o^k = \sum_i s(a_i^k, u^k)\, c_i^k \tag{10}$$

$s(a_i^k, u^k)$ 由问题向量与篇章中每个句向量通过式(4)得到. 第 1 层的句向量 u^1 由词袋模型获得,第 $k(k \geqslant 2)$ 层的句向量 u^k 由 $k-1$ 层的输出向量 o^{k-1} 和 $k-1$ 层的句向量 u^{k-1} 相加得到:

$$u^k = o^{k-1} + u^{k-1} \tag{11}$$

最后一层的输出 o^K 用于答案的预测.

GA 模型[20] 和 AMRNN 模型[21] 等沿用了端到端记忆网络的多层结构,通过重复编码模块和交互推理模块 K 次来提升模型表现. 从模型结构上看,AMRNN 模型与记忆网络结果更加相似,都是通过注意 力操作不断对问题向量进行更新. 以第 k 层为例,问题表示 u^k 对篇章进行注意力操作,获得加权后的篇章表示 V_s^k. 并将 u^k 与 V_s^k 相加作为下一层的问题表示 u^{k+1}.

$$u^{k+1} = u^k + V_s^k \tag{12}$$

最后将第 K 层的问题向量 u^K 用于选择类任务的答案预测.

GA 模型每层采用了"一维注意力"模型所使用的方法,使用注意力操作对篇章表示不断进行更新. 假设第 k 层的输入为 $\{x_1^k, x_2^k, \cdots, x_{L_p}^k\}$,输出为 $\{x_1^{k+1}, x_2^{k+1}, \cdots, x_{L_p}^{k+1}\}$(下一层的输入). $k=1$ 时,x_i^1 为篇章中第 i 个词对应的 embedding 表示. 在第 k 层,输入 x^k 序列首先通过一个 RNN 得到隐层状态表示 $\{h_1^{p_k}, h_2^{p_k}, \cdots, h_{L_p}^{p_k}\}$,然后计算序列中每个词对问题的注意力权重,并对问题进行加权求和,得到篇章中每个词对应的问题向量 $\hat{u}_i^k$,并进行门操作得到输出 x^{k+1} 序列,

$$x_i^{k+1} = h_i^{p_k} \odot \hat{u}_i^k \tag{13}$$

最后使用第 K 层的问题向量 u_k 对篇章 $\{h_1^{p_K}, h_2^{p_K}, \cdots, h_{L_p}^{p_K}\}$ 的注意力权重 s^K 用于完形填空类任务答案预测.

多项选择任务最初被提出时,采用了完形填空类任务所使用的方法,Guokun Lai 等[4] 将 SAR 模型[17] 和 GA 模型[20] 用于 RACE 数据集. 与完形填空类任务不同的是,多项选择任务中已知信息不仅有篇章和问题,还有候选答案. 在交互与推理时,加入候选答案信息能更好地辅助理解问题和篇章信息,从而提高模型表现. Hierarchical Attention Flow[22] 使用层次的注意力流机制,来充分捕获篇章与问题、篇章与候选项、问题与候选项以及候选项之间的关系,并加以利用. ElimiNet[23] 不是直接选取可能性最大的选项,而是采用"排除法",先进行多次"排除"操作,丢弃不相关答案,然后从剩下的选项中选取正确答案. Dynamic Fusion Networks[24] 使用动态的多策略注意力机制,根据问题类型选取不同的融合机制,将篇章、问题与候选答案信息进行融合,用于答案预测.

3.2.2　片段选择与答案生成

上述几种将篇章最后总结为一个上下文向量的方法在完形填空类任务和多项选择类任务上取得了不错的效果,但在其他更复杂的任务上,如片段选取类任务和生成任务上,单模型效果距离人类评估结果相差很多. 在片段选取类任务上,预测答案在篇章中的开始位置〈start〉和结束位置〈end〉需要关注篇章中不同位置的信息. 在生成任务中,生成每个词时,关注的篇章内容也不完全相同. 因此,上述将篇章通过各种注意力机制转化为固定维度向量用于答案预测的方法就很难获得成功. 因此,研究者们提出了一种新的方式来维护上下文信息. 具有代表性的是 BiDAF (Bi-directional attention flow for machine comprehension) 模型[25]. BiDAF 模型用一种双向注意力机制来获取问题感知的篇章表示,并加入一层基于 RNN 的"建模层"用于捕获在所提问题条件下篇章的上下文关系.

BiDAF 采用了"二维注意力"方法,同时考虑了问题对篇章的注意力 a 与篇章对问题的注意力 b(为保持前后符号一致,使用的符号与 BiDAF 原文不同). 这 2 个方向的注意力来源于同一个共享的相似度矩阵 $M \in R^{L_p \times L_q}$,其中 L_p 是篇章长度,L_q 是问题长度,$M[i,j]$ 表示篇章第 i 个单词与问题第 j 个单词的相似度.

$$M[i,j] = \alpha(h_i^p, h_j^q) = w_{(s)}^{\mathrm{T}}[h_i^p; h_j^q; h_i^p \circ h_j^q] \tag{14}$$

其中: $w_{(s)}$ 是一个可训练的参数,$\circ$ 表示逐点的乘法. 基于 S 可以获得篇章对问题的注意力权重 a.

$$a_i = \mathrm{softmax}(M[i,\cdot]) \in R^{L_q} \tag{15}$$

其中 a_i 表示篇章第 i 个词对问题的注意力权重. 篇章感知的问题表示为 $\tilde{H}^q = [\tilde{h}_1^q, \tilde{h}_2^q, \cdots, \tilde{h}_{L_p}^q]$,$\tilde{h}_i^q$ 为篇章中第 i 个词对应的问题向量:

$$\tilde{h}_i^q = a_i H^q \tag{16}$$

类似地，基于 S 可以获得整个问题序列到篇章的注意力向量 b：

$$b = \mathrm{softmax}(\max_{\mathrm{col}} M) \in R^{L_p} \tag{17}$$

$\max_{\mathrm{col}} M$ 指 M 中每列的最大值，通过 b 得到新的篇章上下文：

$$\tilde{h}^p = \sum_i b_i h_i^p \tag{18}$$

其中 $b_i \in R$ 是篇章中第 i 个词对应的权重. 将篇章上下文向量与经过注意力机制变换后的向量合并到一起，得到一个新的上下文向量 G. 篇章中第 i 个词对应的表示为

$$G_{:,i} = [h_i^p; \tilde{h}_i^q; h_i^p \circ \tilde{h}_i^q; h_i^p \circ \tilde{h}^p] \tag{19}$$

将问题感知的篇章向量 G 再次输入一个 RNN 网络，对篇章进行编码得到新的上下文矩阵 H^r，与首次编码的区别是：此时的编码捕获了问题与篇章的交互信息.

此外，比较常用的还有与 BiDAF 模型结构相似的 Match-LSTM 模型[15] 和 R-Net 模型[26]. Match-LSTM 和 R-Net 同样使用“二维注意力”方法对问题与篇章进行融合，获得结合了问题信息的篇章序列表示，并使用“建模层”进行建模. R-Net 在建模层后又加入了一层篇章自匹配层对篇章本身进行交互，提升了性能表现.

生成类任务更加困难，在获得正确答案的同时还要生成通顺的表达. 早期生成类任务的交互推理模块直接沿用片段抽取类任务的方法，如桑志杰等[27] 采取了 BiDAF 模型所使用的交互推理模块. 之后的生成类任务没有完全依照“编码器-推理与交互-输出”框架的方法，而是进行了一些新的探索，其中最具代表性的是微软提出的 S-Net[28] 模型，它将生成类任务看做是一个“提取-综合”的过程，包括“extraction”和“synthesis”2 个模块. “extraction”模块用于预测文章中与答案相关的片段作为“证据”；“synthesis”模块将“证据”作为问题与文章的附加特征用于答案的生成. “extraction”模块与“synthesis”模块分别进行训练.

3.3 答案预测

不同任务的答案来源不同，因此选取答案的方式也有很大不同. 本节将介绍 4 种任务的输出层实现方式.

3.3.1 完形填空与多项选择

完形填空类任务答案来源于篇章中的所有词，这类任务通常有 3 种答案预测方法.

1) 将篇章上下文向量和问题向量连接为一个向量，通过非线性映射为一个篇章长度向量. 向量中相应位置的值为对应候选词的得分，一般选取得分最高的词作为答案[3]，如

$$\omega = \mathrm{argmax}(W_{\mathrm{rg}} r + W_{\mathrm{ug}} u) \tag{20}$$

其中 W_{rg} 和 W_{ug} 是可以训练的参数.

2) 对篇章中的词进行打分，篇章的注意力权重为每个词的得分. 对出现多次的单词得分进行累加，作为该词的最终得分，选取累加后得分最高的单词为答案[16]，如下式所示.

$$\omega = \mathrm{argmax}\left(\sum_{i \in I(\omega, P)} s_i\right) \tag{21}$$

其中：$I(\omega, P)$ 表示单词 ω 出现在篇章中的位置，s_i 是篇章中第 i 个词对应的注意力权重.

3) 在篇章的所有实体词中选取答案，将篇章上下文向量线性映射为一个篇章长度向量作为每个词的得分，选取得分最高的实体词为答案[17].

$$\omega = \underset{i \in I(\omega, \omega \in P \cap E)}{\mathrm{argmax}} (W^a r)(i) \tag{22}$$

其中：$I(\omega, P \mid \omega \in P \cap E)$ 是所有实体词在篇章中出现的位置，W^a 是可以训练的参数.

第 2 种方法的可解释性较强，并且适用于答案为各种类型词的情况，因此后续很多完形填空任务模型均采用第 2 种方法进行答案预测.

多项任务一般采用相似度比较的方法. 将所有候选答案通过编码层编码为答案向量，比较答案向量与“交互与推理”层得到篇章向量的相似度. 相似度最高的被选为正确答案[4].

3.3.2 片段选择与生成

片段选择类任务有两大类答案预测方法：一种方法是预测答案在篇章中的起始位置与终止位置，称为“边界模型”；另一种是逐字预测答案中每一个词，称为“序列模型”，这种方法也可以用于生成式任务.

早期的边界模型将获得的与问题交互后的篇章向量 H^r 作为输入，通过一个非线性映射层预测起止位置[25].

$$\begin{aligned} p^{\mathrm{start}} &= \mathrm{softmax}(W_{\mathrm{start}} [G; H^r]) \\ p^{\mathrm{end}} &= \mathrm{softmax}(W_{\mathrm{end}} [G; H^r]) \end{aligned} \tag{23}$$

这种方法在预测终止位置时没有考虑起始位置，可能会出现起始位置不匹配的现象. 受指针网络的启发，Shuohang Wang 等[15] 采用一个 RNN 解码器对答案进行预测. 解码器长度设为 2，2 个解码器隐藏状态分别用于答案在篇章中起始位置和终止位

置的预测. 也可以采用“序列模型”,解码器长度可变,每个 RNN 单元顺序预测答案中每个单词的位置. “序列模型”能生成一个不存在于原文中的字符串,生成答案的方式更加灵活. 但对于片段选择任务而言,边界模型“效果更佳,因为片段选择类任务中,答案是篇章中的一个片段,直接预测答案在篇章中的起始位置,降低了不确定性.

在机器翻译等领域已经证明了序列生成任务中注意力机制的有效性,指针网络在解码时也加入了注意力机制. 通过上一时刻的隐层输出状态对解码器的输入进行注意力操作,获得每个输入的权重 γ_t.

$$\boldsymbol{F}_t = \tanh\left(\boldsymbol{V}\boldsymbol{H}^r + \left(\boldsymbol{W}^a\boldsymbol{h}^a_{t-1} + \boldsymbol{b}^a\right) \otimes e_{(L_p+1)}\right)$$

$$\boldsymbol{\gamma}_t = \mathrm{softmax}\left(\boldsymbol{v}^{\mathrm{T}}\boldsymbol{F}_t + \boldsymbol{c} \otimes e_{(L_p+1)}\right) \quad (24)$$

其中: $(\cdot \otimes e_{(L_p+1)})$ 表示重复左边的标量或向量 L_p+1 次. $\boldsymbol{V}$、$\boldsymbol{W}^a$、$\boldsymbol{v}$、$\boldsymbol{b}^a$、$\boldsymbol{c}$ 是可训练的参数. 对输入 $\boldsymbol{H}^r$ 进行加权求和,作为当前时刻解码器的输入:

$$\boldsymbol{h}^a_t = \mathrm{RNN}\left(\boldsymbol{h}^a_{t-1}, \boldsymbol{\gamma}_t\boldsymbol{H}^r\right) \quad (25)$$

在片段选取任务中,答案中每个单词来源于篇章,因此在生成答案序列时,只需要考虑从篇章中选词即可. 按指针网络的思路,使用篇章注意力权重 γ_t 作为 t 时刻生成词的概率分布,最终选择概率最大的词即可. 这种方法的缺点是生成的词都必须来源于原文,不适用于生成类任务. 桑志杰等[27]采用了 Pointer generator[29]作为解码器,令生成的词不仅可以来源于原文,也可以来源于词表,使输出更加多样.

4　扩展方法

随着机器阅读理解的发展,研究者们对基于上述框架的模型进行了进一步扩展. 这些扩展分为两类: 一类是原有结构的改进; 另一类是新结构的引入.

4.1　原有结构改进

1) 递归神经网络虽然能够编码句子的顺序信息,但当前时刻的状态依赖于前一时刻的状态,编码速度非常慢. 因此,一些模型采用卷积神经网络[30]或 transformer 结构[6,31]作为编码器,使得同一序列中不同位置的单词可以并行编码,极大地加快了模型运行速度.

2) 一些预训练的语言模型[32-33]使用 transformer 结构[34]直接对篇章和问题进行编码与交互,将上述框架中的“编码器”和“交互与推理”模块融合为一个部分,即在编码的同时进行交互与推理. 这些预训练语言模型以其强大的背景知识刷新了机器阅读理解领域多个任务上的最佳性能.

3) 在预测生成类答案时,为了令生成的答案风格更加多样,在答案预测层引入了“multi-style”学习用来控制生成答案的风格[31].

4.2　新结构引入

1) 机器回答阅读理解问题时通常需要一些外部知识,而训练集中知识有限. 因此,许多工作通过显式地引入结构化知识库来补充外部知识. 如 Knowledgeable Reader 模型在篇章/问题编码器后引入一个知识编码器,并将编码后的知识与上下文表示相融合,用于预测答案[35].

2) 有的机器阅读理解任务中,回答一个问题需要依赖多个篇章的内容. 完成此类任务要求解决跨篇章实体消歧和跨篇章答案验证等问题. 常用方法一般有 2 种: ① 采用多个编码器分别编码每个篇章; 然后采用对齐、融合等操作进行跨篇章的信息交互[32,36]; 最后基于交互后的篇章信息来预测答案. ② 采用多个编码器分别编码每个篇章,并依据每一个篇章生成一个答案,然后引入 1 个答案验证模块从多个答案中选取 1 个答案[37].

3) 篇章一般会涉及多个实体,这些实体之间存在着一些关系. 为了显式建模实体之间的关系,DFGN 模型将图神经网络引入机器阅读理解模型,并对图采用多跳注意力机制来模拟多步推理[38].

4) 对于“答案预测”模块的扩展,常见的有以下 2 种方法: ① 增加一个答案重排序模块,对原框架提取的 Top K 个答案进行重排序获得最终答案[39]; ② 在预测多项选择类答案时,引入答案排除模块,先根据“排除法”排除若干个无关的答案,再从剩下的候选答案中挑选正确答案[23].

5　结束语

笔者就“编码器-交互与推理-输出”框架及其扩展方法综述了当前的阅读理解研究. 可以看到,现有的模型虽然使用了多种不同的注意力机制与交互方法来寻找篇章中与问题更为相关的片段,并取得了一系列重要进展,但是仍有很多不足之处.

从模型规模来看,现在的机器阅读理解在训练时模型依赖于大规模数据集,并需要多块高显存 GPU,模型参数非常多. 构建一个小而精的模型,并在小样本数据集上取得一个比较好的效果很重要.

从模型结构来说，现有的方法还存在以下2个问题：1）无论是单跳模型还是多跳模型，模型的推理步数都是固定的. 然而在真实场景中，不同的问题需要的推理次数不同，预先固定推理步数的方法是不合理的. 这就要求模型能够动态选择推理步数. 2）此外，已有方法大多采用一种固定的注意力机制，即便是在多跳模型中，每跳采用的注意力方法也十分相似. 不同的注意力计算方法[14-15,17-18,20,22]各有优劣之处，适用于不同类型的问题. 因此，根据任务动态地选取注意力方法十分重要，是今后需要进一步探索的问题.

从解决的任务上看，目前的模型对于事实型问题处理得比较好，但在一些特殊的问题类型上，如列表型问题、计数类问题，或涉及逻辑运算的问题，使用现有的问题与篇章逐词比较的交互方式不能获得很好的效果. 这些问题需要用一些特别设计的结构来解决.

参考文献：

[1] Wang Z, Liu J, Xiao X, et al. Joint training of candidate extraction and answer selection for reading comprehension [C] // ACL 2018. Melbourne: ACL, 2018: 1715-1724.

[2] Heath J, Kamp H, Reyle U. From discourse to logic: introduction to model theoretic semantics of natural language, formal logic and discourse representation theory [J]. Language, 1995, 71(4): 823.

[3] Hermann K M, Kocisky T, Grefenstette E, et al. Teaching machines to read and comprehend [C] // NIPS 2015. Montréal: MIT Press, 2015: 1693-1701.

[4] Lai G, Xie Q, Liu H, et al. Race: large-scale reading comprehension dataset from examinations [C] // EMNLP 2017. Copenhagen: ACL, 2017: 785-794.

[5] Rajpurkar P, Zhang J, Lopyrev K, et al. Squad: 100,000 + questions for machine comprehension of text [C] // EMNLP 2016. Austin: ACL, 2016: 2383-2392.

[6] Devlin J, Chang M W, Lee K, et al. Bert: pre-training of deep bidirectional transformers for language understanding [C] // NAACL-HIT 2019. Minneapolis: ACL, 2019: 4171-4186.

[7] Trischler A, Wang T, Yuan X, et al. Newsqa: a machine comprehension dataset [C] // ACL workshop 2017. Vancouver: ACL, 2017: 191-200.

[8] Nguyen T, Rosenberg M, Song X, et al. Ms-marco: a human generated machine reading comprehension dataset [EB/OL]. (2018-10-31) [2019-05-28]. https://www.microsoft.com/en-us/research/wp-content/uploads/2017/05/r-net.pdf.

[9] He W, Liu K, Liu J, et al. Dureader: a chinese machine reading comprehension dataset from real-world applications [C] // ACL workshop 2018. Melbourne: ACL, 2018: 37-46.

[10] Lin C. Rouge: a package for automatic evaluation of summaries [C] // WAS 2004. Barcelona: ACL, 2004: 74-81.

[11] Hirschman L, Light M, Breck E, et al. Deep read: a reading comprehension system [C] // ACL 1999. Maryland: ACL, 1999: 325-332.

[12] Riloff E, Thelen M. A rule-based question answering system for reading comprehension tests [C] // NAACL/ANLP workshop 2000. ACL, 2000: 13-19.

[13] Hao X, Chang X, Liu K. A rule-based chinese question answering system for reading comprehension tests [C] // IIH-MSP 2007. [S. l.]: IEEE, 2007: 325-329.

[14] Cui Y, Chen Z, Wei S, et al. Attention-over-attention neural networks for reading comprehension [C] // ACL 2017. Vancouver: ACL, 2017: 593-602.

[15] Wang S, Jiang J. Machine comprehension using match-lstm and answer pointer [C] // ICLR 2017. Toulon: Proceedings, 2017.

[16] Kadlec R, Schmid M, Bajgar O, et al. Text understanding with the attention sum reader network [C] // ACL 2016. Berlin: ACL, 2016: 908-918.

[17] Chen D, Bolton J, Manning C D. A thorough examination of the cnn/daily mail reading comprehension task [C] // ACL 2016. Berlin: ACL, 2016: 2358-2367.

[18] Cui Y, Liu T, Chen Z, et al. Consensus attention-based neural networks for chinese reading comprehension [C] // COLING 2016. Osaka: ACM Press, 2016: 1777-1786.

[19] Sukhbaatar S, Weston J, Fergus R. End-to-end memory networks [C] // NIPS 2015. Montréal: MIT Press, 2015: 2440-2448.

[20] Dhingra B, Liu H, Yang Z, et al. Gated-attention readers for text comprehension [C] // ACL 2017. Vancouver: ACL, 2017: 1832-1846.

[21] Tseng B H, Shen S S, Lee H Y, et al. Towards machine comprehension of spoken content: initial toefl listening comprehension test by machine [C] // Interspeech 2016. San Francisco: ISCA, 2016: 2731-2735.

[22] Zhu H, Wei F, Qin B, et al. Hierarchical attention flow for multiple-choice reading comprehension [C] // AAAI 2018. New Orleans: AAAI Press, 2018: 6077-6085.

[23] Parikh S, Sai A B, Nema P, et al. Eliminet: a model for eliminating options for reading comprehension with

multiple choice questions [C] // IJCAI 2018. Stockholm: Morgan Kaufmann, 2018: 4272-4278.

[24] Xu Y, Liu J, Gao J, et al. Dynamic fusion networks for machine reading comprehension [EB/OL]. (2018-02-26) [2019-05-28]. https://arxiv.org/pdf/1711.04964.pdf.

[25] Seo M, Kembhavi A, Farhadi A, et al. Bidirectional attention flow for machine comprehension [C] // ICLR 2017. Toulon: Proceedings, 2017.

[26] Wang W, Yang N, Wei F, et al. R-net: machine reading comprehension with self-matching networks [EB/OL]. (2017-05-20) [2019-05-28]. https://www.microsoft.com/en-us/research/wp-content/uploads/2017/05/r-net.pdf.

[27] **桑志杰，袁彩霞. 一种端到端的生成式问答模型** [EB/OL]. 2019 (2019-01-28) [2019-05-28]. http://www.paper.edu.cn/releasepaper/content/201901-186.

[28] Tan C, Wei F, Yang N, et al. S-net: from answer extraction to answer generation for machine reading comprehension [EB/OL]. 2018 (2018-01-02) [2019-05-28]. http://arxiv.org/abs/1706.04815.

[29] See A, Liu P J, Manning C D. Get to the point: summarization with pointer-generator networks [C] // ACL 2017. Vancouver: ACL, 2017: 1073-1083.

[30] Yu A W, Dohan D, Luong M T, et al. Qanet: combining local convolution with global self-attention for reading comprehension [C] // ICLR 2018. Vancouver: Proceedings, 2018.

[31] Nishida K, Saito I, Nishida K, et al. Multi-style generative reading comprehension [C] // ACL 2019, Florence: ACL, 2019: 2273-2284.

[32] Zhuang Y, Wang H. Token-level dynamic self-attention network for multi-passage reading comprehension [C] // ACL 2019, Florence: ACL, 2019: 2252-2262.

[33] Radford A, Narasimhan K, Salimans T, et al. Improving language understanding by generative pre-training [EB/OL]. 2018 (2018-12-03) [2019-05-28]. https://www.cs.ubc.ca/~amuham01/LING530/papers/radford2018improving.pdf.

[34] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need [C] // NIPS 2017. Long Beach: MIT Press, 2017: 5998-6008.

[35] Mihaylov T, Frank A. Knowledgeable reader: enhancing cloze-style reading comprehension with external commonsense knowledge [C] // ACL 2018, Melbourne: ACL, 2018: 821-832.

[36] Hu M, Peng Y, Huang Z, et al. Retrieve, read, rerank: towards end-to-end multi-document reading comprehension [C] // ACL 2019, Florence: ACL, 2019: 2285-2295.

[37] Wang Y, Liu K, Liu J, et al. Multi-passage machine reading comprehension with cross-passage answer verification [C] // ACL 2018, Melbourne: ACL, 2018: 1918-1927.

[38] Xiao Y, Qu Y, Qiu L, et al. Dynamically fused graph network for multi-hop reasoning [C] // ACL 2019, Florence: ACL, 2019: 6140-6150.

[39] Trischler A, Ye Z, Yuan X, et al. Natural language comprehension with the epireader [C] // EMNLP 2016, Austin: ACL, 2016: 128-137.