自然语言处理导论

张奇 桂韬 黄萱菁

2022 年 11 月 4 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
$\bigtriangledown_{\boldsymbol{x}} y$	y 对向量 $\boldsymbol{x}$ 的梯度
$\bigtriangledown_{\boldsymbol{X}} y$	y 对矩阵 $\boldsymbol{X}$ 的梯度
$\bigtriangledown_{\mathbf{X}} y$	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
Var$(f(x))$	$f(x)$ 在分布 $P(x)$ 下的方差
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

$\mathbb{X}$	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

11. 文本摘要

文本摘要（Text Summarization）是一种利用计算机来自动实现文本分析、内容提炼和摘要生成的技术。由于互联网的快速发展，当前时代的信息量出现了指数级的增长，并远超于人类的信息需求、信息处理和信息利用能力，从而导致我们无法准确挑选和运用有效的信息。这种现象被称为**信息过载**（Information Overloading）。而文本摘要技术可以对信息进行简化，帮助人们快速获取和理解新的信息。自 20 世纪 50 年代以来，自动文本摘要技术经过了长远的发展，从早期的基于规则和统计的方法，到如今以数据为驱动的机器学习和深度学习方法，研究人员致力于让自动摘要更接近人类构造的摘要。文本摘要应用的场景十分广泛，且涉及到语义理解和语言生成，是自然语言处理领域中一项重要且具有挑战性的任务。

本章首先介绍文本摘要的基本概念和主要任务，在此基础上介绍抽取式文本摘要方法和生成式文本摘要方法，最后介绍文本摘要的评测方法和文本摘要常见的语料库和评测集合。

11.1 文本摘要概述

从狭义角度来看，文本摘要的目标是为文档提炼关键信息，并构造出代表原文档中最重要或相关内容的子集（摘要）。从广义来看，除了文本形式的信息，我们还可以为图像或视频等多模态的信息进行总结，并产生以自然语言形式描述的摘要。文本摘要的第一份相关文献可以追溯到 1957 年 Hans Peter Luhn 的基于统计方法的研究[1]。随着互联网的高速发展，大量信息以复杂多样的形式存在于人们的日常生活中。文本摘要作为信息甄别和过滤的核心技术，受到学术界和企业界越来越多的关注。以机器学习方法为代表性的抽取式文本摘要技术的研究成果不断涌现，应用落地场景日趋广泛。随着深度学习技术的不断发展，特别是基于大数据和大模型的预训练范式的成功，文本摘要领域的研究仍然非常活跃。文本摘要研究正逐渐转向生成式摘要和实时摘要，并涉及更多基于多模态信息的摘要，例如图像摘要和视频流摘要等。

在本节中，我们将首先介绍文本摘要的发展历程，再分别介绍文本摘要领域的各种任务形式。

11.1.1 文本摘要发展历程

文本摘要相关的研究最早可以追溯到 20 世纪 50 年代，随着计算机技术的普及和应用，自动化的信息提取和摘要逐渐受到人们的关注。1958 年，Hans Peter Luhn 发表了一篇题为“The Automatic Creation of Literature Abstracts”的文章[2]，揭开了通过计算机来实现自动文本摘要的序幕。之后的几十年时间里，文本摘要主要依赖于信息检索中常见的特征，例如词频（Term Frequency，TF）以及词频-逆文档频率（Term Frequency and Inverse Document Frequency，TF-IDF）等，通过统计信息来进行关键信息的提取。在 1990 年之前的方法中，文本摘要都是通过规则和统计信息从原始文本中直接抽取（复制）内容，而不是生成新的内容和抽象摘要。

传统的基于频率和规则的方法使得建立高效和稳健的摘要系统面临很大的挑战。随着机器学习技术的快速发展，有相当多有监督和无监督的自动摘要方法被提出。2002 年，Neto 等人[3] 将自动文本摘要视为一个二分类问题，如果一个句子出现在抽取式的参考摘要中，则认为它是“正确的”，否则就被认为是“不正确的”。在文献 [3] 中，他们使用了两种著名的机器学习分类方法，朴素贝叶斯和 C4.5 算法。对于无监督方法，可以通过定义相关的特征进行关键信息的筛选，从而不需要依赖人工构造的摘要。常见的特征选择包括句子的位置、正负相关词、句子中心度、与标题的相似性等等。对于如何训练模型来选取和组合这些特征，常见的算法包括遗传算法（Genetic Algorithm，GA）和潜在语义分析 (Latent Semantic Analysis，LSA) 等。除此之外，还有针对句子特征进行聚类和排序的无监督文本摘要方法，均展现出了良好的效果。

自 2014 年开始，以深度学习和神经网络为代表的文本摘要技术逐渐兴起，它们与传统技术相比具有更加卓越的性能。首先就是词嵌入表示逐渐取代了原来的特征工程。词嵌入通过神经网络技术，能产生比经典的词袋 (Bag-of-Word，BOW) 方法更加高质量的词表示。2019 年 Alami 等人构建了一个基于词嵌入的文本摘要系统[4]。其次，使用深度自编码器 (Auto-Encoder，AE) 从词频输入计算隐特征也可以为句子或单词产生更高质量的表示。以深度神经网络为支撑的表示学习，为单词、句子、文档提供了高质量的语义表示，并大大促进了文本摘要的发展。

DUC-2003 和 DUC-2004 竞赛为文本摘要任务提供了标准，由此生成式文本摘要技术迎来了快速发展。原文本和摘要都是单词序列，因此可以用序列到序列模型（Sequence-to-Sequence Model，Seq2Seq）处理输入的文本序列和输出的摘要序列。机器翻译（Machine Translation，MT）问题与文本摘要具有明显的相似性，它们的区别在于机器翻译是信息无损的，而文本摘要是有损的。因此，一些早期的生成式文本摘要方法直接使用了机器翻译模型。早期的 Seq2Seq 模型主要是基于循环神经网络（RNN）结构。2017 年，以 Transformer[5] 为代表的自注意力模型凭借强大的文本建模能力和并行效率，成为了生成式文本摘要的主流。与此同时，抽取式文本摘要的建模方法也逐渐转为了以 RNN 或自注意力模块为基本框架的模式。

随着移动互联网和 Web2.0 时代的到来，网络上的信息量增长飞速，数据量越来越大，数据模态越来越多，文本摘要的研究也进入了新时期。在这一阶段，依托大规模预训练模型的文本摘要展现出了卓越的性能，多模态多语言摘要任务逐渐受到大家的关注，文本摘要的应用场景也越来

越广泛。同时，文本摘要的效率和鲁棒性、评估方法的多样性和合理性、场景的可迁移性等，逐渐成为了研究者进一步探索的方向。

11.1.2 文本摘要主要任务

文本摘要被广泛应用于多种任务和场景中，按照任务进行划分，可以将文本摘要分为单文档摘要、多文档摘要、对话摘要、跨语言文本摘要和多模态文本摘要等。本节将介绍各种任务的定义。

1. 单文档摘要

单文档摘要是最基本的摘要任务。给定一篇文档作为输入，模型输出它的核心内容，且长度明显短于输入文档。根据输入文档的内容长度，可以将单文档摘要分为两类：短文本摘要和长文本摘要。

在短文本摘要场景中，文本类型可以是新闻文本、评论文本、邮件文本等短文本。CNN/Daily Mail [6] 是一个被广泛关注和研究的短文本摘要数据集，它包含了时事新闻和其对应的摘要，新闻的数据来源是美国有线电视新闻网和《每日邮报》。长文本摘要则是面向学术论文、专利文档、书籍等长文本的摘要。与短文本摘要相比，长文本的信息量更多，但文档结构更加规范化。我们可以根据文档类型的不同，制定针对性的摘要策略。例如，新闻摘要关注新闻的人物、时间、地点、事件等信息；评论摘要关注产品的特性、用户的情感倾向；学术论文摘要更关注问题的定义、方法的描述、实验的结果等等。

2. 多文档摘要

与单文档摘要不同，**多文档摘**要是对同一主题下的多个文档提炼主要信息并输出摘要的任务。从应用的角度看，对于某一新闻事件，互联网上的某个新闻单位会针对此事件进行系列报道，或者多个新闻单位对同一事件进行同时报道。在这种情况下，对这些相关性很强的多文档提炼出一个覆盖性强、形式简短的摘要就具有重要的意义。另一种应用场景是商品评论的摘要。对于某一种商品，商家分析其售后评论并总结出用户关注的方面和特性，有助于商家针对性地优化产品。对于多文档的学术论文摘要，我们可以通过对多篇文章的研究内容进行归纳总结，得到相关研究主题的主要挑战、主流方法和未来趋势等等。

3. 对话摘要

对话摘要的目标是为多种类型的对话生成摘要，包括会议、通话记录、在线聊天和客服对话等场景。从参与者的角度来看，对话与新闻文档最大的不同点在于前者的信息涉及两个以上的参与者。除此之外，对话摘要会涉及更多的对话语义消歧、指代消解、信息补全等问题和挑战。相比新闻摘要和论文摘要，对话的摘要很难直接从现有的内容中获取，而新闻的标题和论文的摘要可直接作为参考结果并进行大规模的摘要数据收集，因此构建对话摘要的数据集更加困难。除此之外，对话会涉及音频、视频等信息，因此对话摘要还会涉及多模态信息的建模和理解。

4. 跨语言文本摘要

跨语言文本摘要是指输入源语言（如英文）文档，输出目标语言（如中文）摘要的任务。它的主要应用场景在于为非母语者生成摘要。一个常见的跨语言摘要数据集是 Global Voices[7]，它是多语言新闻的英文摘要，主要包含 15 种语言的新闻及其摘要。一般而言，跨语言文本摘要可拆分为文本摘要和机器翻译两个子任务进行处理。

5. 多模态文本摘要

多模态文本摘要的输入和输出不仅仅局限于文本内容，还包括图像、视频、语音等多媒体数据。例如，新闻文档中包含的图像、会议记录中的语音、电影字幕对应的视频等等。在多模态文本摘要中，各个模态的信息可能有重合或互相补充，如何对这些信息进行对齐、识别、筛选，做到摘要内容的不重复和不遗漏，是进行多模态文本摘要的重点。MSMO 数据集[8] 是一个常见的多模态新闻摘要数据集，数据来源是《每日邮报》，该数据集包含了带有图像的新闻文档，它们是内容对齐的，即图像和文本描述了同样的内容。而在一些复杂的场景下，例如教学和演示视频、电影内容等，文字部分无法全面描述视频或图像展示的信息。这种情况下，我们需要进行图像或视频的理解，从而给多模态文本摘要带来了更大的挑战。

11.2 抽取式文本摘要

抽取式文本摘中的关键内容是从原始文本中提取的，同时被提取的内容不会以任何方式进行修改。这里，关键内容包括可用于“标记”或索引文本文档的短语，或共同组成摘要的关键句（包括标题）。关键文本的抽取类似于略读的过程，人们在决定是否要详细阅读一篇文档之前，会倾向于选择阅读其中的摘要（如果有）、标题、副标题、数字、文档的第一段和最后一段，以及段落中的第一句和最后一句。常见的抽取式文本摘要方法可分为两大类：基于排序的方法和基于序列标注的方法。

11.2.1 基于排序的方法

基于排序的抽取式文本摘要的基本思想是对于一篇文档，首先按照一定的规则将其拆分为多个语义单元（如短语、句子、段落等），并通过某种重要性评估方法为每个语义单元进行打分。之后，按照重要性分数的大小将语义单元进行排序，最后选取分值较高的语义单元作为关键内容并组成摘要。一般而言，会选取句子作为内容提取的基本语义单元，而如何衡量句子的重要性以及如何设计合适的打分方法，则是影响排序效果好坏的关键。

1. Lead-3 算法

对于大部分类型的文档，句子在文档中的位置是一个重要的打分依据。一般而言，文档的起始句段会进行总述，结尾句段会进行总结。因此，最简单的启发式排序方法就是按照句子的位置选择关键句作为摘要。Lead-3 算法会选取文档的前三句内容作为摘要。这种方法虽然简单直接，但

在某些场景中（如新闻文本摘要）非常有效。我们常常将 Lead-3 作为一个基线方法，并将其视为文本摘要方法的性能下界。

2. TextRank 算法

TextRank [9] 是一个经典的基于排序的抽取式摘要方法，它的核心思想来自于 PageRank [10] 算法。PageRank 是一个基于图的排序算法，用于将网页进行排序，目的是尽可能地将重要的网页排在靠前的位置。这一类基于图的排序算法，可以递归地从图中提取全局信息，并根据节点间的某种关系来衡量节点的重要性，为节点打分排序。PageRank 算法将网页作为有向图的节点，将网页之间可跳转的超链接作为图的有向边。当页面 A 能跳转到页面 B 时，我们可以将其视为页面 A 为页面 B“投票”。一个页面获得的票数越多，该页面的重要性就越高。此外，PageRank 模型会根据投票页面的重要性，决定其投票的权重。因此，一个页面的重要性得分取决于获得的票数，以及为其投票的节点分数。对于一个 PageRank 模型，我们有如下公式：

$$S(V_i) = (1-d) + d * \sum_{j \in In(V_i)} \frac{1}{|Out(V_j)|} S(V_j) \tag{11.1}$$

其中 V 代表页面节点，$S(V)$ 代表节点的重要性分数，$In(V)$ 为可跳转到页面 V 的页面集合，$Out(V)$ 为页面 V 可跳转到的页面集合，d 为阻尼系数。上述公式为迭代式，节点的分数会经过多轮迭代进行更新直到收敛。一般情况下，会设一个初始值（如 $\frac{1}{N}$，N 为页面数量）对节点分数进行初始化。

受 PageRank 算法的启发，在进行文本摘要任务时，TextRank 算法将文档中的句子作为节点，以句子间的相似度作为边，通过迭代更新节点的重要性分数。相似度表示计算公式如下：

$$\text{Sim}(S_i, S_j) = \frac{|\{w_k | w_k \in S_i, w_k \in S_j\}|}{log(|S_i|) + log(|S_j|)} \tag{11.2}$$

其中，S 和 w 分别代表句子和句子中的单词，$|S|$ 代表句子 S 中的单词数量，分子部分表示同属于两句的单词数量。相应的，对于句子节点的打分，采用如下公式：

$$S(V_i) = (1-d) + d * \sum_{V_j \in In(V_i)} \frac{\text{Sim}(V_j, V_i)}{\sum_{V_k \in Out(V_j)} \text{Sim}(V_j, V_k)} S(V_j) \tag{11.3}$$

使用边的权值迭代更新节点的重要性分数直到收敛，最后选取 N 个得分最高的节点句子，作为摘要进行输出。

3. MMR 算法

Maximal Marginal Relevance（MMR）[11] 又叫最大边界相关算法，也是一种基于排序的抽取式文本摘要算法。与 TextRank 相比，MMR 可以得到更加多样化的抽取结果，在保证重要性的同时避免抽取过多相似的句子。例如，以下是一系列描述上海市的短句：“十分现代化”、“有浓厚的

现代化气息”、“现代化中国的代表城市”、“广阔的发展前景”。前三个短句虽然与上海市的相关性很大，但是都表示上海的现代化程度。如果将其作为摘要，就会出现冗余信息。MMR 算法旨在解决这样的问题，在重要性和多样性之间达到平衡。

MMR 算法最初是用来计算搜索引擎中的查询文本与被搜索文档之间的相似度，然后对文档进行排序并召回。具体到文本摘要任务，可以通过计算句子之间的相似度以及句子与整篇文档的相似度，并通过如下公式计算每个句子的分数：

$$MMR \stackrel{\text{def}}{=} \text{Arg} \max_{V_i \in R \setminus S} [\lambda(\text{Sim}_1(V_i, Q) - (1-\lambda) \max_{V_j \in S} \text{Sim}_2(V_i, V_j))] \tag{11.4}$$

其中，Q 代表整篇文档（或文档类别描述）。R 为文档中的句子集合。S 为已被选出作为结果的句子集合，是 R 的子集。$R \setminus S$ 代表 R 中剩余未被选中的句子集合。$\text{Sim}_1(V, Q)$ 表示文档中某个句子和整篇文档（或文档描述）的相似度，$\text{Sim}_2(V_i, V_j)$ 表示文档中两个句子之间的相似度。λ 为范围 $[0, 1]$ 内的常数，用于调整结果的多样性，λ 越小表示算法更倾向于保证抽取结果的多样性。

通过 MMR 算法，可以按照重要性和多样性对句子进行排序，两者的衡量方式都是基于相似性度量。通常认为与文档整体更相似的句子重要性越高，同时与其他句子相似性较低的句子多样性越好。可用的相似性度量方式有很多，常见的包括 TF-iDF、余弦相似度、欧式距离，或者也可以使用神经网络模型进行评判打分。

4. REFRESH 算法

上述讨论的方法均是为单个句子进行打分并选出分数最高的若干句子，除此之外，还可以为选出的句子集合进行整体打分。REFRESH[12] 是一种通过强化学习对抽取的句子集合进行全局优化的方法。在训练阶段，每当模型抽取出若干句子，可以将此句子集合与标准摘要进行比较，计算出两者的相关性分数（一般使用 ROUGE 分数作为度量手段，参考本章的第 4 节）作为奖励分数，通过强化学习训练策略来对模型参数进行优化更新。与一般的极大似然估计法相比，强化学习可以直接使用与摘要任务相关的评估指标来进行模型的优化，从而使得对句子集合的整体打分与最终的评估指标接近。REFRESH 模型主要由三部分组成：句子编码器、文档编码器和句子抽取器。其模型结构如图11.1所示。

在 REFRESH 模型中，句子编码器使用 CNN 中经典的时域卷积网络（Temporal Convolutional Network）[13] 将句子编码成连续型向量表示，$f \in R^{k-h+1}$ 为其得到的时域长度，其中 h 为卷积窗口大小，k 为句子长度，并采用最大池化（max-pooling）将最大值作为最后的特征保留。之后，句子向量经过文档编码器来融合上下文中其他句子的信息。这里，文档编码采用的是经典的 LSTM 长短记忆循环神经网络[14]。

对于核心的句子抽取器，REFRESH 使用了 LSTM 模型外加 Softmax 函数来计算似然概率 $P(y_i|s_i, D, \theta)$。不过,REFRESH 采用的训练策略并非极大化似然函数,而是通过其概率分布 $P(y_i|s_i, D, \theta)$ 为文档 D 中的每一个句子 s_i 进行打分。可以选出分数最高的 n 个句子组成摘要，并将此句子集

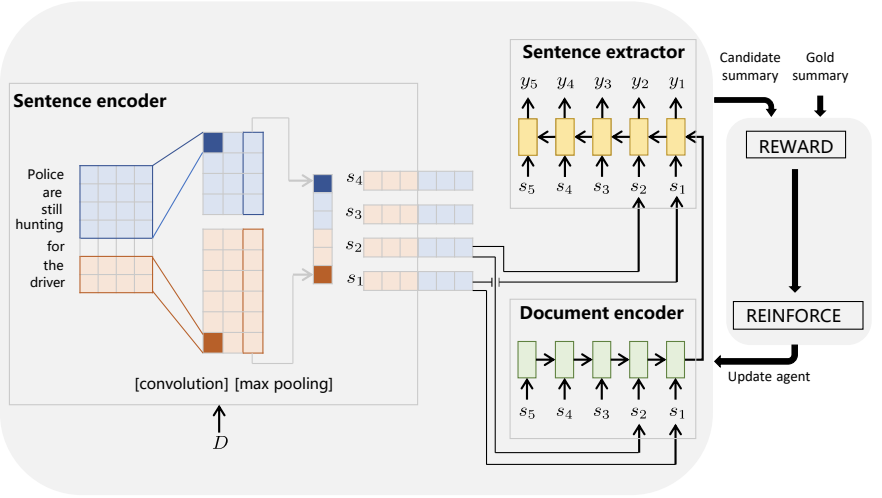

图 11.1 REFRESH 模型结构[12]

合记为 $\hat{y}$，将 $\hat{y}$ 与标准摘要的相关性分数作为奖励函数 $r(\hat{y})$，则强化学习的期望奖励和损失函数可用如下式子表示：

$$\mathcal{L}(\theta) = -E_{\hat{y}\sim p_\theta}[r(\hat{y})] \tag{11.5}$$

$$\nabla\mathcal{L}(\theta) = -E_{\hat{y}\sim p_\theta}[r(\hat{y})\nabla\log(\hat{y}|D,\theta)] \tag{11.6}$$

上式计算的是与句子集合 $\hat{y}$ 相关的期望项。然而，穷举所有可能的句子组合方式是不可行的。在实际的训练过程中，REFRESH 每次从 p_θ 分布中采样单个样本 $\hat{y}$ 进行训练，来近似整个训练批次的奖励期望：

$$\begin{aligned}\nabla\mathcal{L}(\theta) &\approx -r(\hat{y})\nabla\log(\hat{y}|D,\theta)\\ &\approx -r(\hat{y})\sum_{i=1}^{n}\nabla\log(\hat{y}_i|s_i,D,\theta)\end{aligned} \tag{11.7}$$

5. NeuSum 算法

一般而言，基于排序的抽取式摘要方法都需要先对句子进行打分，之后再进行排序和句子选择。与上述方法不同的是，NeuSum[15] 采用了一种将句子打分和句子选择相结合的方式来进行文档摘要。它的训练目标即为学习一个打分函数 g，该函数可以计算某个句子被选为摘要句后，摘要整体的分数增益。在测试阶段，模型在每一个时间步 t 选出能使打分函数 g 结果最高的句子。打分函数 g 如下所示：

$$g(S_i) = r(\boldsymbol{S}_{t-1}\cup\{S_i\}) - r(\boldsymbol{S}_{t-1}) \tag{11.8}$$

上式中，NeuSum 采用了与 REFRESH 相同的评价函数 $r(\cdot)$ 来计算句子集合整体与标准摘要之间的相关性分数。S_i 表示文档中的某个句子，$\boldsymbol{S}_t$ 表示在 t 时刻已经被选择的摘要句组成的集合。在每一个时间步 t，模型会选出能使摘要整体增益最大的句子（即函数 g 达到最大值的句子）直至达到摘要限制的长度。NeuSum 模型结构如图11.2所示，主要由两个模块组成：句子编码器和句子抽取器

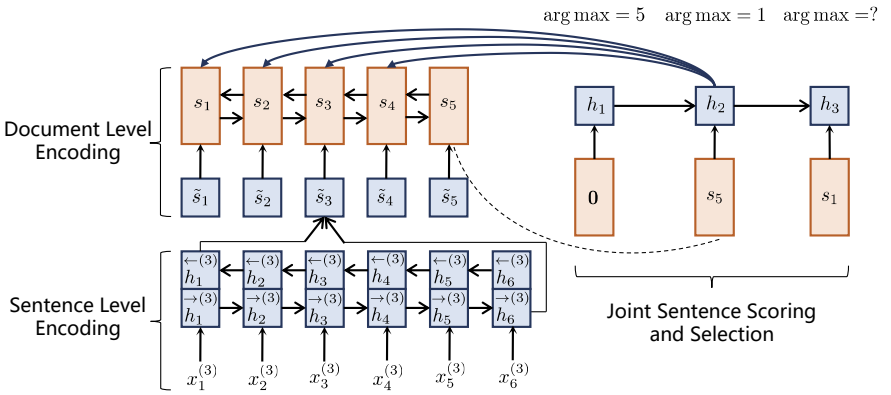

图 11.2　NeuSum 模型结构

NeuSum 模型中句子编码器是一个句子-文档级层次编码器。对于文档 $D = (S_1, S_2 ... S_L)$（S_i 为文档中的句子），模型先将句子 S_i 输入到一个双向的 GRU 网络[16] 中，得到句子级向量表示 $\widetilde{\boldsymbol{s}}_i$，再将得到的句子级向量输入到另一个双向 GRU 网络，可以得到融合上下文信息的句子向量 $\boldsymbol{s}_i$。

对于句子抽取器，我们可以使用如下的公式计算句子 S_i 的分数，并取分数最高的句子作为 t 时刻抽取的摘要句。

$$\delta(S_i) = \boldsymbol{W}_s \tanh(\boldsymbol{W}_q \boldsymbol{h}_t + \boldsymbol{W}_d \boldsymbol{s}_i) \tag{11.9}$$

其中，$\boldsymbol{W}_s$,$\boldsymbol{W}_q$ 和 $\boldsymbol{W}_d$ 为可训练的参数，$\boldsymbol{h}_t$ 为 t 时刻的隐层状态，由一个 GRU 门控循环神经网络[16] 计算得到。基于句子分数 $\delta(S_i), i \in [1, L]$（$L$ 为文档中的句子数量），可以计算模型的预测分布 P，如下所示：

$$P(\hat{S}_t = S_i) = \frac{\exp(\delta(S_i))}{\sum_{k=1}^{L} \exp(\delta(S_k))} \tag{11.10}$$

为了近似式子11.8中的打分函数,NeuSum 计算了用于训练的标签分布 Q,并使用 KL 散度(Kullback-Leibler Divergence）来缩小模型预测分布 P 和训练标签分布 Q 之间的距离。

$$Q(S_i) = \frac{\exp(\tau g(S_i))}{\sum_{k=1}^{L} \exp(\tau g(S_k))} \tag{11.11}$$

$$J = D_{KL}(P||Q) \tag{11.12}$$

上式中，τ 为调整分布方差的温度系数。对于 $r(\cdot)$，我们一般使用 ROUGE 分数来衡量摘要句集与标准摘要之间的相关度。

11.2.2 基于序列标注的方法

抽取式文本摘要的另一种实现方式是序列标注。与基于排序的方法类似，首先要将文档分割成独立的语义单元，并将其组成语义单元序列，然后为每个语义单元附加一个标签来表明该部分是否要被提取出来组成摘要。一般来说，我们会选取句子作为基本语义单元，并使用二元标签（0和 1）来表明句子被抽取的状态。标签“1”表示该句子可以需要被抽取出来作为摘要句，而“0”表示该句子不需要作为摘要句。通过这种思路，我们可以将抽取式文本摘要转化为序列标注任务。与排序式文本摘要的不同点在于，基于序列标注的方法直接对句子进行二元分类，而不需要预先设计并计算句子的重要性分数，所以可以直接使用有监督学习的方法进行训练。

1. 基于特征工程的文本摘要算法

对于早期的抽取式文本摘要，人们通常使用统计机器学习的方法结合特征工程来进行。一个常见的方案是使用朴素贝叶斯模型，需要预先为文档中的句子进行人工标注，标明它是否为摘要句。在附带标签的训练集基础上，我们需要设计文档中句子的特征来训练模型完成序列标注。以下是常见的句子特征：

- **长度**：句子的长度一定程度上可以反映信息量的大小。当句子超过或短于一定长度时都是不适合作为摘要的。
- **位置**：基本上所有类型的文本都有其约定俗成的写作规范。句子的内容和其位置在一定程度上是有关联的。例如，文章结尾通常包含整篇文章的总结综述。
- **关键词**：当某个句子包含该类型文本的关键词，那么这个句子很有可能是摘要的一部分。
- **提示语**：在某些类型的文档中，关键的概括和总述性语句前会有明显的提示语，例如，“综上所述”，“To conclude”一类的短语之后会紧跟重要的句子。

有了设计好的特征值集合和标签后，就可以训练朴素贝叶斯分类器，并用它来计算每个句子属于摘要的概率：

$$P(label|f_1, f_2, f_3, , f_n) \tag{11.13}$$

通常需要人为设定一个阈值，当概率大于该阈值时表示当前句子是摘要的一部分，反之则不是。

但是，文本摘要的人工标注的代价高昂，而且人们在写摘要的过程中通常都会使用到文档中的原词甚至原句，所以研究员们开始思考是否可以使用现有的文本-摘要对，例如：文稿、论文、报告与它们的摘要，来进行分类器的训练。一种思路就是为摘要中的句子在正文中找出与其相似度最高的句子，并用它来作为对应部分的摘要，即**对齐**（alignment）算法。通过对齐算法，能够产生大规模的文档-摘要对，从而可以进行有监督的训练。简单的对齐算法有计算正文和摘要句子的最长公共子序列或编辑距离等，这些方法也可以结合使用。这里，我们介绍一种基于隐马尔可夫模型（Hidden Markov Model，HMM）的对齐算法[17]。

这类算法的思路是将摘要中的句子分解成字或词（后续简称为字词），然后从句首开始往后依次找到这些字词在正文中的出现位置，进而就可以找到相应的句子来构成近似甚至相同的摘要。由于同一个字词可能在文章中多次出现，所以摘要中的单个字词就存在多个可能的出现位置，需要我们去判断其中哪一个是正确的。以某篇文章为例，其摘要中出现了“the communication subcommittee of”，而“the”，“communication”，“subcommittee”，“of”分别在正文中出现了 44，1，2，22 次，也就意味着有 $44 \times 1 \times 2 \times 22$ 共 1936 种可能的位置组合。而隐马尔可夫模型则具备找出这个正确位置组合的能力。

在该隐马尔可夫模型中，字词的每个可能的出现位置都是可以观测到的，且这些位置都各自对应着一个状态 (S, W)，它表示这个字词出现在正文第 S 句话的第 W 个位置。该模型假设摘要中的每个字词 I_{i+1} 在正文中的实际位置都仅受其前一个字词 I_i 在正文中的实际位置的影响，那么每个位置序列的概率就可以表示为：

$$P(I_0(S_0, W_0), ..., I_n(S_n, W_n)) = \prod_{i=0}^{N-1} P(I_{i+1}(S_{i+1}, W_{i+1}) | I_i(S_i, W_i)) \tag{11.14}$$

其中，条件概率 $P(I_{i+1}(S_{i+1}, W_{i+1}) | I_i(S_i, W_i))$ 的值是人工设定的，可以通过多次实验来选择较优值，在处理不同类型的文章时还可以做一些适应性调整。其中一种启发式赋值就是按照当前处理字词 I_{i+1} 和上一个处理的字词 I_i 的相对位置分成 6 种不同的情况来梯度赋值：

- **情况一**：$S_i = S_{i+1}, W_i = W_{i+1} - 1$，即 I_{i+1} 和 I_i 出现在正文的同一个句子中，且 I_i 刚好在 I_{i+1} 的前一个位置。
- **情况二**：$S_i = S_{i+1}, W_i < W_{i+1} - 1$，即 I_{i+1} 和 I_i 出现在正文的同一个句子中，且 I_i 在 I_{i+1} 的前方，但二者不相邻。
- **情况三**：$S_i = S_{i+1}, W_i > W_{i+1}$，即 I_{i+1} 和 I_i 出现在正文的同一个句子中，但 I_i 在 I_{i+1} 的后方。
- **情况四**：$S_{i+1} - \text{CONST} < S_i < S_{i+1}$，即在正文中，$I_{i+1}$ 出现在 I_i 所属句子的后方，且二者之间不会超过 CONST 个句子。
- **情况五**：$S_{i+1} < S_i < S_{i+1} + \text{CONST}$，即在正文中，$I_{i+1}$ 出现在 I_i 所属句子的前方，且二者之间不会超过 CONST 个句子。
- **情况六**：$S_{i+1} - S_i >= \text{CONST}$，即在正文中，$I_{i+1}$ 和 I_i 的位置间隔了 CONST 以上个句子。

其中,CONST 是一个较小的正数。从情况一到情况六,其出现的可能性是递减的,只要对式子11.14中的条件概率项进行合理地赋值，就可以计算出每种位置序列的概率，用概率最大的位置序列去找相应句子组合成摘要就可以取得不错的效果。需要注意的是有些摘要中的字词不一定会出现在原文中，例如：衔接词。需要对这些字词进行特殊化处理，比如说用 $S = -1$ 来表示该字词没有在正文中出现，然后在计算时跳过该字词。

2. SummaRuNNer 算法

循环神经网络（Recurrent Neural Network，RNN）是处理序列标注任务的经典模型之一，被广泛应用于中文分词、命名实体识别等任务中。同样，我们也可以使用 RNN 进行基于序列标注的抽取式摘要。这里，SummaRuNNer[18] 就是基于 RNN 的抽取式摘要模型，该模型是一个采用两层基于门控循环单元（Gated Recurrent Unit，GRU）的双向循环神经网络，其模型结构如图11.3所示。

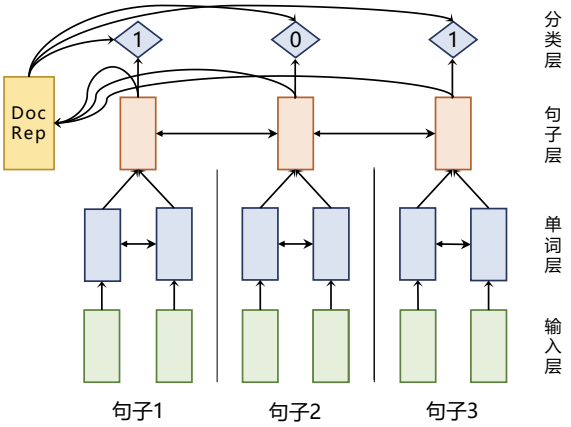

图 11.3　SummaRuNNer 模型结构

SummaRuNNer 算法从文本第一句开始逐句判断各句子是否能够成为摘要的一部分，最终实现摘要的抽取。其中的门控循环单元由两个门组成，分别是更新门 $\boldsymbol{u}$（update gate）和重置门 $\boldsymbol{r}$（reset gate）。门值及隐状态计算公式如下：

$$\boldsymbol{u}_j = \sigma(\boldsymbol{W}_{ux}\boldsymbol{x}_j + \boldsymbol{W}_{uh}\boldsymbol{h}_{j-1} + \boldsymbol{b}_u) \tag{11.15}$$

$$\boldsymbol{r}_j = \sigma(\boldsymbol{W}_{rx}\boldsymbol{x}_j + \boldsymbol{W}_{rh}\boldsymbol{h}_{j_1} + \boldsymbol{b}_r) \tag{11.16}$$

$$\boldsymbol{h}'_j = \tanh(\boldsymbol{W}_{hx}\boldsymbol{x}_j + \boldsymbol{W}_{hh}(\boldsymbol{r}_j \odot \boldsymbol{h}_{j-1}) + \boldsymbol{b}_h) \tag{11.17}$$

$$\boldsymbol{h}_j = (1 - \boldsymbol{u}_j) \odot \boldsymbol{h}'_j + \boldsymbol{u}_j \odot \boldsymbol{h}_{j_1} \tag{11.18}$$

其中，$\boldsymbol{W}, \boldsymbol{b}$ 是模型参数，$\boldsymbol{h}_j, \boldsymbol{x}_j$ 分别是时间步 j 的隐状态向量和输入向量，$\odot$ 表示哈达玛积。

SummaRuNNer 算法逐级对文本进行处理：第一层在词级别对文本进行处理，双向地依次根据每个单词的词嵌入以及历史隐状态来计算每个单词的隐状态。第二层则是在句子级别对文本进行处理，它将第一层输出的隐状态作为其自身的输入，通过平均池化将所有属于同个句子的隐状

态进行融合。然后利用所有句子的隐状态通过下方公式计算出文本的整体表示：

$$\boldsymbol{d} = \tanh(\boldsymbol{W}_d \frac{1}{N_d} \sum_{j=1}^{N_d} [\boldsymbol{h}_j^f, \boldsymbol{h}_j^b] + \boldsymbol{b}) \tag{11.19}$$

其中，$\boldsymbol{h}_j^f$ 和 $\boldsymbol{h}_j^b$ 分别是文本第 j 个句子前向（forward）和后向（backward）处理时的隐状态，N_d 是文本中的句子数量，“[]”表示将两个向量进行拼接。该模型的最后一层就是一个分类器，它从第一句开始，结合句子的隐状态以及文本的整体表示，依次判断各个句子是否需要提取出来作为摘要的一部分：

$$P(y_j = 1|\boldsymbol{h}_j, \boldsymbol{s}_j, \boldsymbol{d}) = \sigma(\boldsymbol{W}_c \boldsymbol{h}_j + \boldsymbol{h}_j^T \boldsymbol{W}_s \boldsymbol{d} - \boldsymbol{h}_j^T \boldsymbol{W}_r \tanh(\boldsymbol{s}_j) + \boldsymbol{W}_{ap} \boldsymbol{p}_j^a + \boldsymbol{W}_{rp} \boldsymbol{p}_j^r + \boldsymbol{b}) \tag{11.20}$$

其中，$\boldsymbol{W}_c \boldsymbol{h}_j$ 反映句子的信息量，$\boldsymbol{h}_j^T \boldsymbol{W}_s \boldsymbol{d}$ 反映句子对于文章的重要性，$\boldsymbol{h}_j^T \boldsymbol{W}_r \tanh(\boldsymbol{s}_j)$ 反映句子对于已提取部分的冗余程度，$\boldsymbol{W}_{ap} \boldsymbol{p}_j^a$ 和 $\boldsymbol{W}_{rp} \boldsymbol{p}_j^r$ 分别考虑了句子的在文本中的绝对位置（实际的句子数）和相对位置（将文本分割成固定数量的段，句子所属的段数就是相对位置），$\boldsymbol{P}^a$ 和 $\boldsymbol{P}^r$ 是需要学习得到的模型参数，分别是句子绝对位置和相对位置的嵌入表示。值得一提的是，这种方式一定程度上提高了模型决策的可解释性。得到模型的预测结果后，再将其代入到损失函数，就能够对模型进行训练来抽取摘要：

$$\begin{aligned} \mathcal{L}(W, b) = -\sum_{d=1}^{N} \sum_{j=1}^{N} (y_j^d \log P(y_j^d = 1|\boldsymbol{h}_j^d, \boldsymbol{s}_j^d, \boldsymbol{d}_d) \\ + (1 - y_j^d) \log(1 - P(y_j^d = 1|\boldsymbol{h}_j^d, \boldsymbol{s}_j^d, \boldsymbol{d}_d))) \end{aligned} \tag{11.21}$$

3. BERTSum-Ext 算法

BERTSum-Ext[19] 是一个基于 BERT 编码器的抽取式摘要模型。由于原始的 BERT 是针对字词进行编码,而基于序列标注的抽取式文本摘要是逐句操作和分类,在做句子级别的序列标注时,要对输入 BERT 的内容做一定调整,需要进一步计算句子的向量表示。图11.4给出了 BERT 与 BERTSum-Ext 在输入输出上的差别。图11.4左边是原始 BERT 模型网络结构，右边是 BERTSum-Ext 模型结构。我们可以观察到两者都是使用了 [SEP] 标记来分隔句子，而不同之处在于两者的 [CLS] 标记的含义。在原始 BERT 中，[CLS] 标记被添加在文本序列的开始，该位置整合了输入序列中的全局信息。在 BERTSum-Ext 中，[CLS] 标记被添加在每一个句子的开始位置，该句子的有效信息将会被整合到对应 [CLS] 标记位置的字段上。此外，BERTSum-Ext 会为位置是奇数和偶数的句子分别赋予不同的段嵌入 (Segment Embeddings)，即第 1，2，3，4... 句将分别被赋予段嵌入 $\boldsymbol{E}_A, \boldsymbol{E}_B, \boldsymbol{E}_A, \boldsymbol{E}_B$...。

在这样调整后，通过结合图11.4的 3 种不同嵌入就可以得到不同位置的 [CLS] 向量 $\boldsymbol{t}_i$ 来表示

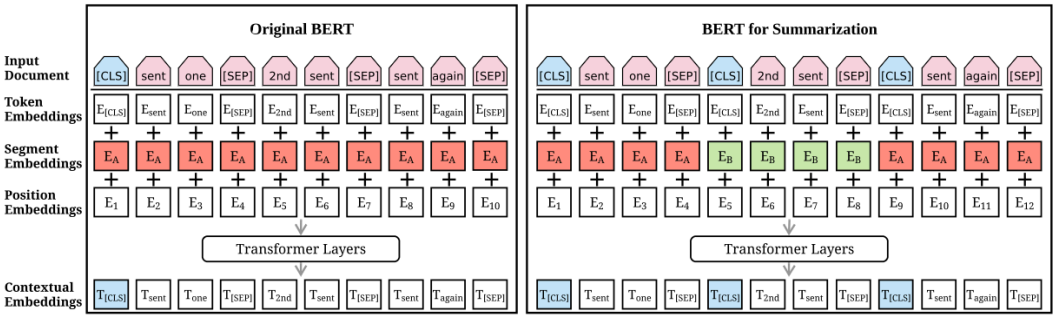

图 11.4 BERTSum 模型结构

各个句子，然后将其输入到后续的 Transformer 模块中进行更新：

$$\tilde{\boldsymbol{h}}^l = \mathrm{LN}(\boldsymbol{h}^{l-1} + \mathrm{MHAtt}(\boldsymbol{h}^{l-1})) \tag{11.22}$$

$$\boldsymbol{h}^l = \mathrm{LN}(\tilde{\boldsymbol{h}}^l + \mathrm{FFN}(\tilde{\boldsymbol{h}}^l)) \tag{11.23}$$

其中，LN（Layer Normalization）表示归一化操作，MHAtt（Multi-head Attention）表示多头注意力操作，FFN（feed-forward neural network）表示前馈神经网络。这 3 个部分共同构成了一个 Transformer 模块，而 l 表示层数，每一层都是一个 Transformer 模块，$\boldsymbol{h}^0 = [\boldsymbol{t}_1, \boldsymbol{t}_2, ..., \boldsymbol{t}_n]$（假设文档一共有 n 个句子）。最终的句子嵌入是逐层学习得到的，越高层的句子嵌入覆盖的信息量越多，我们使用顶层输出的句子嵌入作为最终输入分类器的表示。分类器可以为每个句子表示计算并预测二元分类结果，表示各个句子是否需要被提取出来作为摘要的一部分：

$$\hat{y}_i = \sigma(\boldsymbol{W}_o \boldsymbol{h}_i^L + \boldsymbol{b}_o) \tag{11.24}$$

其中，$\boldsymbol{W}_o$ 为可学习的参数。使用模型预测值与参考标签之间的交叉熵作为损失函数，就可以对模型进行训练。在测试阶段，选择预测概率高于一定阈值的句子，作为最终抽取得到的摘要。

11.3 生成式文本摘要

抽取式文本摘要所获得的结果都来源于原始文本，因此所得到摘要文本中每个单句的语义准确性和语法正确性都可以得到很好的保证。但是由于原始的句子来源于不同的段落甚至篇章，其句子之间的连贯性很难保证，造成摘要整体的可读性较差。生成式文本摘要则可以产生原始文档中不存在的新文本。通常生成式文本摘要算法需要首先构建原始文档的抽象语义表示（编码过程），然后使用此语义表示创建出接近人类表达方式的摘要（解码过程）。该类算法通常能产生更精简凝练并且更具有连贯性的摘要。语义表示过程也可应用于图像和视频，因此生成式摘要方法也可用来处理多模态文本摘要。但是，整体的摘要生成过程比抽取式摘要更具挑战性，它不仅涉及自然

语言生成，还涉及依赖领域知识对原始文档的深入理解。

11.3.1 序列到序列生成

在 2014 年，谷歌大脑团队首次提出了序列到序列生成结构（Sequence-to-Sequence，Seq2Seq）[20] 来进行文本生成任务的处理。序列到序列生成结构主要由编码器（Encoder）和解码器（Decoder）组成，其中编码器负责将输入文本中所需要的语义信息编码成向量，解码器则负责从编码向量中解码出语义信息并生成特定文本。对应到生成式文本摘要，可以将原始文档输入编码器，编码器进行文档的理解并将其中的关键信息进行编码，解码器则将关键信息进行解码并生成对应的文本摘要。

1. 基于预训练的生成式方法

BERTSum-Abs[19] 是结合了预训练语言模型的 Seq2Seq 摘要生成方法。它的基本框架也是基于 Transformer 结构，其区别在于，编码器是预训练语言模型（如 BERT[21]），而解码器则采用参数随机初始化的多层 Transformer 神经网络结构。这样的设计虽然可以很好的利用了预训练方法，能够对文本内容进行很好的编码，但是会导致编码器和解码器之间存在着参数状态不匹配的问题：前者的模型参数是经过预先训练的，而后者的模型参数是随机初始化的（一般以均匀分布或正太分布初始化参数）。这可能会使训练阶段的微调过程不稳定，因为在相同的优化策略下，解码器相比于编码器更难收敛。为了解决这一问题，BERTSum-Abs 设计了一个新的微调范式，用不同的优化策略对编码器和解码器进行微调。例如，可以将解码器的学习率设置更大，使其参数更新的幅度更大，逐渐缩小两者不匹配的参数状态。

具体地，BERTSum-Abs 对编码器和解码器分别使用了不同的 Adam 优化器[22]，它们具有不同的预热步数（Warm Up Step）和学习率：

$$lr_{\mathcal{E}} = \tilde{lr}_{\mathcal{E}} \cdot \min\left(\text{step}^{-0.5}, \text{step} \cdot \text{warmup}_{\mathcal{E}}^{-1.5}\right) \tag{11.25}$$

$$lr_{\mathcal{D}} = \tilde{\text{lr}}_{\mathcal{D}} \cdot \min\left(\text{step}^{-0.5}, \text{step} \cdot \text{warmup}_{\mathcal{D}}^{-1.5}\right) \tag{11.26}$$

一般来说，预训练的编码器应该以更小的学习率和更平滑的衰减速率进行微调。这样，当解码器参数变得稳定时，编码器就可以得到更精确的梯度进行参数更新。此外，BERTSum-Abs 还提出了一种两阶段的微调方法，首先在抽取式摘要（BERTSum-Ext）上微调编码器，第二阶段再进行生成式摘要的微调。这两个阶段使用的是同一个编码器，从而可以让模型利用这两种模式之间共享的编码信息。实验表明，在抽取式摘要任务上微调编码器，可以进一步提高生成式摘要模型的性能。

上述方法中，BERTSum-Abs 使用的是预训练编码器，并结合参数随机初始化的解码器进行微调。这种解决方案主要是针对早期的预训练语言模型（BERT 等），它们侧重于文本的理解和编码。随着预训练语言模型的不断发展，陆续出现了对编码器和解码器联合预训练的解决方案，其可以直接被应用在适合序列到序列生成架构的文本生成任务中，并在生成式文本摘要任务中展现

了卓越的性能。这里我们介绍一由 Lewis 等人于 2020 年提出的基于序列到序列生成结构的预训练语言模型 BART[23]。BART 的基本模型结构也是 Transformer，但是其预训练目标是学习一个基于 Seq2Seq 结构的去噪自编码器（Denoising Auto-encoder）。BART 的预训练包括以下两个过程：(1) 适用特定的噪声函数破坏原始文本；(2) 将受到破坏的文本输入 Seq2Seq 模型并重建原始文本。其基本的模型结构和预训练任务如图11.5所示。

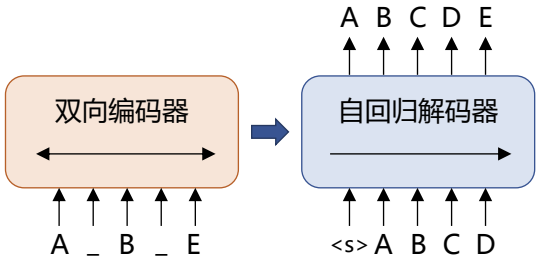

图 11.5　BART 结构图

对于加噪过程，BART 定制了几种噪声函数来“破坏”原始文本。也可以对其进行扩展，设计不同的加噪方案。在极端情况下，可以丢弃所有的原始文本信息，此时 BART 相当于普通的语言模型。BART 的具体加噪方案如下所示，图11.6展示了各种加噪方案的含义。

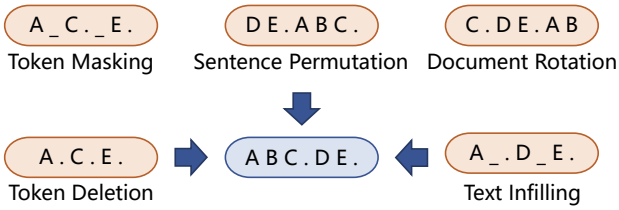

图 11.6　对于输入进行加噪，这些加噪方式可以进行组合

- **单词掩码**（Token Masking）：与 BERT 的预训练任务类似，BART 对文本中的单词进行随机采样，将其替换为 [MASK] 标记。
- **单词删除**（Token Deletion）：随机将文本中的一部分单词删除。与单词掩码相比，模型必须要对被删单词的位置进行判断。
- **文本掩码**（Text Infilling）：从文本中采样一部分短语（短语的长度服从于泊松分布，$\lambda = 3$），每个短语都被替换为一个 [MASK] 标记。对于这种加噪方式，模型需要预测每个 [MASK] 标记的位置丢失了多少信息。值得注意的是，存在长度为 0 的短语，即此处 [MASK] 标记的位置没有丢失任何信息。
- **句子乱序**（Sentence Permutation）：根据句号等句子结束符将文本划分为多个句子，并将这

些句子随机打乱顺序。

- **文档旋转**（Document Rotation）：在文档中均匀随机地选择一个单词，并基于此单词旋转原来的文档序列，使得变换后的文档序列以此单词开始。对于这种加噪方式，模型需要识别出文档的起始位置。

对于重构过程，将上述得到的加噪文本输入编码器，并期望解码器输出原始文本。在预训练阶段，BART 使用重构损失函数进行优化，即优化解码器输出和原始文本之间的交叉熵。用在下游的摘要生成任务时，我们将原始文档输入编码器，从解码器输出摘要，同样使用交叉熵进行微调。

BART 本身是一个去噪自编码器，与摘要任务的形式有一定的关联性，即去除文档中的冗余和无关的信息，只保留关键信息。因为这种上游预训练形式与下游任务相关的特性，BART 在生成式文本摘要中取得了优异的性能，并在多个摘要评测集上取得了很好的效果。

2. 复制与覆盖机制

因为生成式文本摘要方法会对原始文档的内容进行转换和整合，所以如何保证摘要中的关键信息不重复不遗漏是非常重要的问题。一方面，原始文档可能会包含重要的生僻词甚至未登录词，而这些词在解码过程中被生成的概率极低，就会造成关键信息的遗漏问题。另一方面，一些在原始文档中反复出现的关键信息也有可能被解码器生成多次，造成摘要冗余和不通顺的问题。为了解决以上问题，See 等人[24] 针对生成式文本摘要的复制与覆盖机制提出了 PGNet。PGNet 使用了指针解码器，通过指针从原始文本中直接复制单词，同时也可以从词表产生原文档中未出现的新单词。此外，PGNet 还使用了覆盖向量来跟踪和控制摘要对原文档的内容覆盖范围，从而能减少冗余现象。PGNET 的神经网络结构如图11.7所示。

具体地，PGNet 使用了 RNN 文档编码器，将原始文档的所有单词输入一个单层双向 LSTM 模型中，得到编码器隐状态 $\boldsymbol{h}_i$。解码器是一个单层单向 LSTM，在每个时间步 t 上，接收前一个单词的嵌入表示，并得到解码器状态 $\boldsymbol{s}_t$。之后，PGNet 计算了注意力分布[25]$\boldsymbol{a}^t$，来衡量第 t 个解码时间步上每个单词的重要性：

$$\boldsymbol{e}_i^t = \boldsymbol{v}^T \tanh\left(\boldsymbol{W}_h \boldsymbol{h}_i + \boldsymbol{W}_s \boldsymbol{s}_t + \boldsymbol{b}_{\text{attn}}\right) \tag{11.27}$$

$$\boldsymbol{a}^t = \text{softmax}\left(\boldsymbol{e}^t\right) \tag{11.28}$$

其中，$\boldsymbol{v}, \boldsymbol{W}_h, \boldsymbol{W}_s, \boldsymbol{b}_{attn}$ 都是可训练的参数。

接下来，PGNet 利用注意力分布来产生编码器隐状态的加权和，称为上下文向量 $\boldsymbol{h}_t^*$：

$$\boldsymbol{h}_t^* = \sum_i \boldsymbol{a}_i^t \boldsymbol{h}_i \tag{11.29}$$

上下文向量 $\boldsymbol{h}_t^*$ 可以看作是在每个时间步，解码器从原文档中读取的全局内容表示，它与解码器状

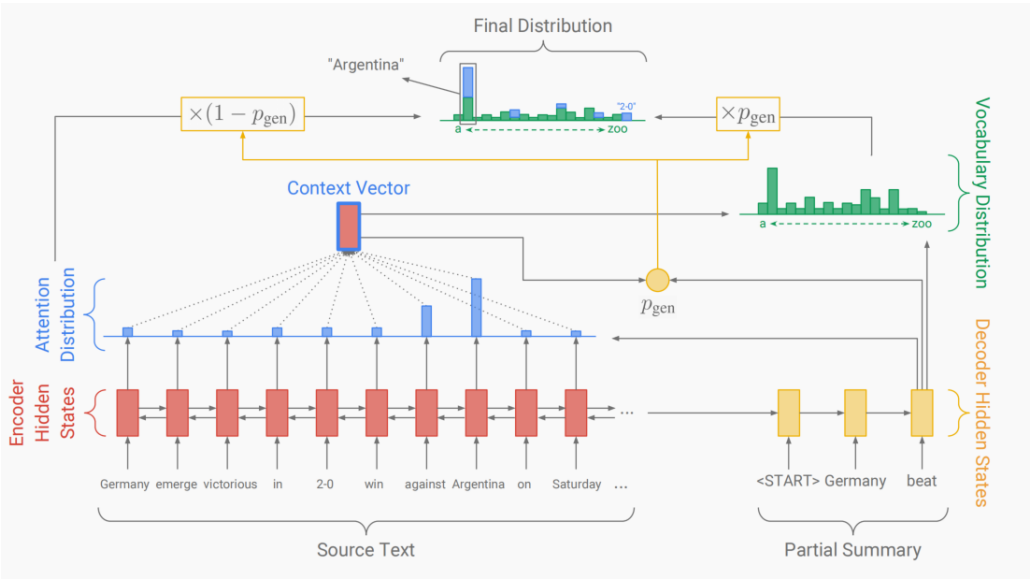

图 11.7 指针生成网络结构[24]

态 $\boldsymbol{s}_t$ 连接，并通过两个线性层来产生词汇表分布 P_{vocab}：

$$P_{\text{vocab}} = \text{softmax}\left(\boldsymbol{V}'\left(\boldsymbol{V}\left[\boldsymbol{s}_t, \boldsymbol{h}_t^*\right] + \boldsymbol{b}\right) + \boldsymbol{b}'\right) \tag{11.30}$$

其中 $\boldsymbol{V}, \boldsymbol{V}', \boldsymbol{b}$ 和 $\boldsymbol{b}'$ 是可学习的参数。P_{vocab} 是词表中所有单词的生成概率分布。

除了从词表单词的概率分布中生成单词，PGNet 还允许通过指针从原文档中复制单词，从而有机会将文档中的生僻词和未登录词写入摘要。PGNet 需要做出以下判断：在 t 时刻，解码器应当从词表中生成单词，还是从原文档中复制单词。这里，PGNet 设计了一个选择器 $p_{\text{gen}} \in (0, 1)$，以 0-1 之间的概率来判断此时间步的操作，应当是从 P_{vocab} 中采样生成一个单词，还是通过从注意力分布 $\boldsymbol{a}^t$ 中采样来复制原文档中的一个单词。时间步 t 的二元选择概率 p_{gen} 基于全局语义向量 $\boldsymbol{h}_t^*$、解码器状态 $\boldsymbol{s}_t$ 和解码器输入 $\boldsymbol{x}_t$，通过如下公式计算：

$$p_{\text{gen}} = \sigma\left(\boldsymbol{w}_{\boldsymbol{h}^*}^T \boldsymbol{h}_t^* + \boldsymbol{w}_{\boldsymbol{s}}^T \boldsymbol{s}_t + \boldsymbol{w}_{\boldsymbol{x}}^T \boldsymbol{x}_t + \boldsymbol{b}_{\text{ptr}}\right) \tag{11.31}$$

最终，可以得到词表与原始文档中单词的总概率分布，即结合了词表分布 P_{vocab} 和注意力分布 $\boldsymbol{a}^t$ 来采样单词。这也可看作是一种扩展的词表，包含了原词表和原始文档中所有出现单词的联合词表。在扩展词表上的概率分布可计算如下：

$$P(w) = p_{\text{gen}} P_{\text{vocab}}(w) + (1 - p_{\text{gen}}) \sum_{i:w_i=w} \boldsymbol{a}_i^t \tag{11.32}$$

如果单词 w 是一个未登录词，那么 $P_{vocab}(w)$ 为零。类似地，如果 w 没有出现在原始文档中，那么 $\sum_{i:w_i=w} a_i^t$ 为零。产生未登录词的能力是 PGNet 模型的主要优势之一。

除了解决生僻词和未登录词的遗漏问题，PGNet 还使用了一个简单的方案解决摘要中单词的重复生成问题。单词的重复生成问题是 Seq2Seq 模型的常见问题，在生成多句文本时尤其明显，一个简单的思路是记录所有已经生成的单词，并且避免在后续的解码时间步生成这些词。PGNet 中的覆盖机制也是基于这一思想。具体地，PGNet 维护一个覆盖向量 $\boldsymbol{c}_t$，它是解码器的所有历史时间步的注意力分布的和：

$$\boldsymbol{c}^t = \sum_{t'=0}^{t-1} \boldsymbol{a}^{t'} \tag{11.33}$$

直观来看，$\boldsymbol{c}_t$ 是基于原始文档单词的一种非归一化的分布，它代表了这些单词到 t 时间步为止被注意力分布覆盖的程度。$\boldsymbol{c}_t$ 一般可以初始化为一个全 0 向量，因为在第一个时间步中，注意力分布还没有覆盖原始文档。

PGNet 将覆盖向量作为注意力机制的额外输入作为一种控制冗余的信号。因此，公式11.27被更改为：

$$\boldsymbol{e}_i^t = \boldsymbol{v}^T \tanh\left(\boldsymbol{W}_h \boldsymbol{h}_i + \boldsymbol{W}_s \boldsymbol{s}_t + \boldsymbol{w}_c \boldsymbol{c}_i^t + \boldsymbol{b}_{\text{attn}}\right) \tag{11.34}$$

通过上述公式，PGNet 确保了当前的注意力分布是通过之前时间步的注意力分布得到的，从而可以避免注意力分布在相同位置的单词上赋予过高的权重，避免产生重复的文本。此外，PGNet 还设计了额外的损失函数 covloss_t，以惩罚反复关注同一单词位置的情况：

$$\text{covloss}_t = \sum_i \min\left(\boldsymbol{a}_i^t, \boldsymbol{c}_i^t\right) \tag{11.35}$$

上述公式惩罚了 t 时间步前每个注意力分布和覆盖向量之间的重叠，从而防止了注意力分布的重复。最后，将上述损失函数通过超参数 λ 加权添加到主损失函数中，得到一个新的复合损失函数：

$$\text{loss}_t = -\log P\left(w_t^*\right) + \lambda \sum_i \min\left(\boldsymbol{a}_i^t, \boldsymbol{c}_i^t\right) \tag{11.36}$$

3. 全局优化与强化学习

前面所介绍的两种生成式摘要方法优化目标的计算都是基于摘要序列的极大似然估计，即最小化所有时间步下目标单词的负对数似然函数。然而，这样的优化目标存在以下问题：（1）摘要生成的**曝光偏差问题**（Exposure Bias）[26]，模型在训练阶段计算的是基于标准摘要序列的单词后验分布，即 $p\left(y_t \mid y_1^*, \ldots, y_{t-1}^*, x\right)$，这里 y^* 表示标准摘要。在预测阶段，模型无法获得标准摘要，因

此它只能基于已经预测的结果计算下一个时间步单词的后验概率，即 $p(y_t \mid y_1, \ldots, y_{t-1}, x)$，在这种情况下容易导致错误的累积；（2）摘要是一个较为主观的结果，标准摘要只是一个参考，除此之外还可能存在其它合适的摘要，这些摘要的词语和句子可能以不同的方式进行排列和组合。文本摘要的评测指标（如 ROUGE[27]）考虑了这种灵活性，但最大似然估计的优化目标无法做到这一点。

Paulus 等人提出了一种引入关键的注意力机制并利用强化学习目标，尝试解决上述问题生的成式摘要模型 DeepReinSum[28]。模型使用了双向 LSTM 编码器 $\text{RNN}^{e_{\text{fwd}}}$, $\text{RNN}^{e_{\text{bwd}}}$ 从文档单词 x_i 的嵌入向量计算隐藏状态 $\boldsymbol{h}_i^e = [\boldsymbol{h}_i^{e_{\text{fwd}}}; \boldsymbol{h}_i^{e_{\text{bwd}}}]$ 并建模输入序列。进行编码之后，再使用单向 LSTM 解码器 RNN^d，使用词嵌入向量为 t 时刻的解码输出 y_t 计算隐藏状态 $\boldsymbol{h}_t^d$。输入和输出的单词嵌入都取自同一个嵌入矩阵 $\boldsymbol{W}_{\text{emb}}$。

在每个解码时间步 t，除了解码器此时的隐状态和当前已生成的单词之外，模型还使用了注意力机制来关注编码端得到的输入序列部分。将 e_{ti} 定义为隐状态 $\boldsymbol{h}_i^e$ 在 t 时刻的注意力分数：

$$e_{ti} = f(\boldsymbol{h}_t^d, \boldsymbol{h}_i^e) \tag{11.37}$$

其中 f 可以是从 $\boldsymbol{h}_t^d$ 和 $\boldsymbol{h}_i^e$ 向量返回标量 e_i^t 的任何函数，这里选择使用双线性函数：

$$f(\boldsymbol{h}_t^d, \boldsymbol{h}_i^e) = {\boldsymbol{h}_t^d}^\top \boldsymbol{W}_{\text{attn}}^e \boldsymbol{h}_i^e \tag{11.38}$$

在得到注意力分数之后，该模型使用以下方式在时序维度上对注意力权重进行归一化，惩罚在过去解码步中已获得高注意力分数的单词。它首先定义了一个新的基于时序特征的注意力分数 e'_{ti}：

$$e'_{ti} = \begin{cases} \exp(e_{ti}) & \text{if } t = 1 \\ \frac{\exp(e_{ti})}{\sum_{j=1}^{t-1} \exp(e_{ji})} & \text{otherwise.} \end{cases} \tag{11.39}$$

之后，计算归一化注意力分数 α_{ti}^e 并使用这些权重来获得输入上下文向量 $\boldsymbol{c}_t^e$：

$$\alpha_{ti}^e = \frac{e'_{ti}}{\sum_{j=1}^n e'_{tj}} \tag{11.40}$$

$$\boldsymbol{c}_t^e = \sum_{i=1}^n \alpha_{ti}^e h_i^e \tag{11.41}$$

上述基于时序特征的注意力机制一定程度上可以缓解生成重复短语的问题。然而，解码器仍然会根据其自身的隐状态产生错误累积，尤其在生成长序列时此现象更加严重。为此，模型额外引入了一种解码器内部的注意力机制。具体而言，对于每个解码步 t，模型计算一个解码器上下文向量 $\boldsymbol{c}_t^d$。将 $\boldsymbol{c}_1^d$ 设置为一个零向量，因为在第一个解码步，生成的序列是空序列。对于 $t > 1$，使

用以下公式计算 $\boldsymbol{c}_t^d$:

$$e_{tt'}^d = \boldsymbol{h}_t^{d^\top} \boldsymbol{W}_{\text{attn}}^d \boldsymbol{h}_{t'}^d \tag{11.42}$$

$$\alpha_{tt'}^d = \frac{\exp\left(e_{tt'}^d\right)}{\sum_{j=1}^{t-1} \exp\left(e_{tj}^d\right)} \tag{11.43}$$

$$\boldsymbol{c}_t^d = \sum_{j=1}^{t-1} \alpha_{tj}^d \boldsymbol{h}_j^d \tag{11.44}$$

除了使用注意力机制来缓解可能的错误累积问题，该模型的另外一个核心在于使用了基于强化学习的算法来进行模型参数的优化。如我们在上文提到的，一般的摘要生成模型会在每个解码步使用极大似然估计损失来进行优化，如下所示:

$$L_{ml} = -\sum_{t=1}^{n'} \log p\left(y_t^* \mid y_1^*, \ldots, y_{t-1}^*, x\right) \tag{11.45}$$

这种优化目标与最终的摘要评测指标具有不一致性。而此模型使用了强化学习中的策略学习, 它可以针对某种离散的评估指标直接进行优化，并使用自评判的策略梯度（self-critical policy gradient）算法 [29] 来进行参数的更新。

具体的实现过程如下，首先为每一个训练样本（文档）产生两个单独的解码序列（摘要），分别记为 y^s 和 $\hat{y}$。其中，y^s 是从每个解码步的 $p(y_t^s|y_1^s, ..., y_{t-1}^s, x)$ 概率分布中采样获得的序列。而 $\hat{y}$ 作为一个基线序列，是每一步从上述概率分布中取最大概率的单词而获得的序列，本质上是一种基于贪心策略的采样方案。将 $r(y)$ 定义为输出摘要 y 的奖励函数，它是通过某种评估指标（通常为 ROUGE）来比较 y 与标准摘要 y^* 而得到的分数。由此，强化学习的损失函数可定义如下:

$$L_{rl} = \left(r(\hat{y}) - r\left(y^s\right)\right) \sum_{t=1}^{n'} \log p\left(y_t^s \mid y_1^s, \ldots, y_{t-1}^s, x\right) \tag{11.46}$$

可以看到，最小化 L_{rl} 等价于最大化采样序列 y^s 的条件似然。如果 y^s 获得了比基线 $\hat{y}$ 更高的奖励，模型可以得到正反馈，从而增大 y^s 的概率。反之，模型则会减小 y^s 的生成概率。

然而，这种强化学习目标的一个潜在问题是，针对特定离散指标（如 ROUGE）进行优化并不能保证输出质量和可读性的提高。为此，可以考虑在不影响可读性的情况下对这些离散指标进行策略梯度的优化[30]。一般来说，ROUGE 衡量了生成摘要和标准摘要之间的 n-gram 重叠程度，而对于可读性，一般使用困惑度（Perplexity）来衡量。困惑度可以直接由通过极大似然训练目标训

练的条件语言模型来计算得到。因此，可以定义一个混合的目标函数，它结合了极大似然估计和强化学习目标：

$$L_{\text{mixed}} = \gamma L_{rl} + (1-\gamma) L_{ml} \tag{11.47}$$

其中，超参数 $\gamma \in [0,1]$ 可以用来控制不同训练阶段各个损失函数的比重。一般来说，在训练的初期，将 γ 设为 0，并随着训练轮数的增长逐渐增大 γ 的值，从而先保证模型生成的摘要具有很好的流畅性，再通过优化 ROUGE 指标来使摘要的质量得到整体的提升。

在该模型中，使用了 ROUGE 等常见的摘要评估指标来对摘要模型进行优化。在这个过程中，自动评估指标被用来计算摘要的奖励函数。除了这种方式，还可以使用基于模型的方法来为强化学习模型提供反馈，一种经典的方案便是序列对抗生成网络（Sequence Generative Adversarial Network，SeqGAN）[31]。它的核心思想在于，模型需要同时训练一个生成器 G 和一个鉴别器 D，鉴别器需要尽可能区分真实的文本和 G 生成的文本，而生成器则需要尽可能产生接近真实的文本，从而困扰 D 做出正确的判断。经过多轮的迭代优化，生成器可以产生与真实文本接近的结果。其中，鉴别器的预测结果可以给生成器合适的反馈，并可通过强化学习的方法来优化生成器。

SeqGAN 的思想也可用在生成式文本摘要的全局优化上[32]。此时，生成器就是一个标准的 Seq2Seq 摘要生成模型，输入原文档并输出摘要。鉴别器是一个文本分类器，它的作用就在于试图区分生成的摘要和标准摘要（二分类），并为生成器提供反馈。与标准的 SeqGAN 训练策略类似，我们首先对生成器进行训练。之后，我们将标准摘要作为正例，将生成器产生的摘要作为负例，训练鉴别器。然后我们交替训练生成器和判别器，直到收敛。

在实际训练过程中，当生成器的参数固定之后（记为 G_θ），我们通过以下方式来动态更新鉴别器的参数（记为 D_ϕ）：

$$\min_{\phi} -\boldsymbol{E}_{Y\sim p_{\text{data}}}\left[\log D_\phi(Y)\right] - \boldsymbol{E}_{Y\sim G_\theta}\left[\log\left(1-D_\phi(Y)\right)\right] \tag{11.48}$$

其中，Y 表示摘要结果，$Y \sim p_{\text{data}}$ 表示真实的摘要，$Y \sim G_\theta$ 表示生成的摘要。另一方面，当鉴别器的参数固定时，我们就需要进一步迭代更新生成器。生成器 G 的损失函数由两部分组成：由强化学习的策略梯度计算的损失 J_{pg} 和最大似然损失 J_{ml}。在形式上，G 的目标函数是 $J = \beta J_{pg} + (1-\beta) J_{ml}$，其中 β 是用来计算 J_{pg} 和 J_{ml} 两个损失的比重，与式子11.47类似。J_{pg} 对于参数 θ 的梯度可由以下公式计算：

$$\begin{aligned}\nabla_\theta J_{pg} &= \frac{1}{T}\sum_{l=1}^{T}\sum_{y_t} R_D^{G_\theta}\left(\left(Y_{1:t-1}, X\right), y_t\right)\cdot\nabla_\theta\left(G_\theta\left(y_t \mid Y_{1:t-1}, X\right)\right)\\ &= \frac{1}{T}\sum_{t=1}^{T}\boldsymbol{E}_{y_l\in G_\theta}\left[R_D^{C_\theta}\left(\left(Y_{1:t-1}, X\right), y_t\right)\nabla_\theta \log p\left(y_t \mid Y_{1:t-1}, X\right)\right]\end{aligned} \tag{11.49}$$

其中，$R_D^{G_\theta}\left(\left(Y_{1:t-1}, X\right), y_t\right)$ 是动作价值函数，我们有 $R_D^{G_\theta}\left(\left(Y_{1:t-1}, X\right), y_t\right) = D_\phi(Y_{1:T})$。$T$ 是摘要的长度，$Y_{1:t}$ 是生成到时间步 t 的部分摘要，X 表示原文档。从上式我们可以看出，对解码阶段每

一个时间步决策的奖励都是相同的，都为 $D_\phi(Y_{1:T})$。如果我们把这一项替换为 $Y_{1:T}$ 与标准摘要的 ROUGE 分数，就变回了上文提到的基于指标的强化学习优化方式。

11.3.2 抽取与生成结合式文本摘要

抽取式文本摘要和生成式文本摘要都存在着各自的优势，于是研究员们也尝试将抽取式方法和生成式方法结合，并形成一类新的方法。这一类方法的算法流程为：首先对原文档的内容进行抽取，得到初步的摘要内容；再将抽取的结果输入生成器，经过优化得到最终的摘要。我们可以根据第一步中抽取内容的粒度（如字词、句子）对这一类方法进行区分。

1. 词粒度抽取与生成结合方法

Bottom-Up[33] 是一种以字词为抽取粒度的方法，其思路是将摘要生成分成如图11.8所示两个主要步骤：（1）通过自底向上的注意力机制从文本中抽取可以作为摘要的单词；（2）用第一步抽取出的单词辅助生成摘要。

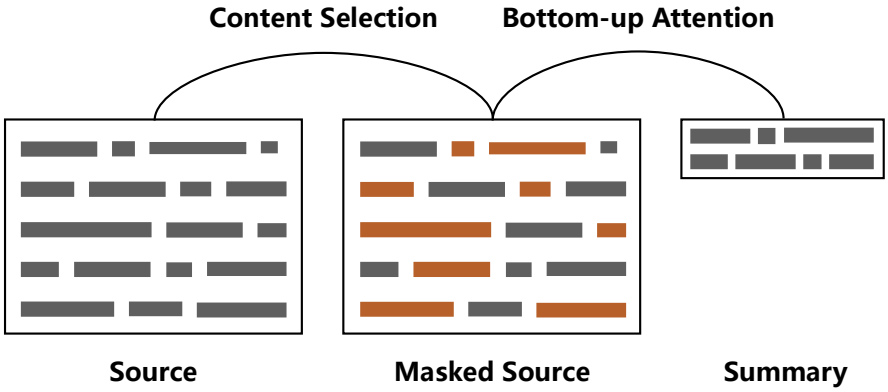

图 11.8 Bottom-Up 模型结构

对于第一步抽取单词，可以直接参考基于序列标注的抽取式方法，让文档中的每个单词都拥有一个二元标签（1 和 0，用于表示该词语能否作为摘要的一部分）以及一个嵌入表示用于完成相应的序列标注。初始嵌入表示 $\boldsymbol{e}_i$ 由两个部分 $\boldsymbol{e}_i^w$ 和 $\boldsymbol{e}_i^w$ 拼接而成，$\boldsymbol{e}_i^w$ 是预训练模型（如 BERT）学习到的词嵌入，$\boldsymbol{e}_i^c$ 则是用一个双层双向长短期记忆网络计算得到：

$$\boldsymbol{e}_i^c = \gamma \times \sum_{l=0}^{2} \boldsymbol{s}_j \times \boldsymbol{h}_i^{(l)} \tag{11.50}$$

其中 l 表示层数，$\gamma, \boldsymbol{s}_{0,1,2}$ 是可以学习的参数。上下文信息通过这种方式被融入到中 $\boldsymbol{e}_i^c$。在计算出 $\boldsymbol{e}_i$ 后，还需要将其输入到另一个单层双向长短期记忆网络中，以此计算出每个单词的最终嵌入 $\boldsymbol{h}_i$，

然后就可以通过 $\boldsymbol{h}_i$ 计算出各单词能用于生成摘要的概率 q_i:

$$q_i = \sigma(\boldsymbol{W}\boldsymbol{h}_i + \boldsymbol{b}) \tag{11.51}$$

其中 $\boldsymbol{W}$,$\boldsymbol{b}$ 为可以学习的参数。最后我们设定阈值 ϵ，当模型选择某个单词的概率高于阈值，我们就可以将其选作关键内容，进入下一个步骤。在步骤二中，摘要生成可以通过一个 Seq2Seq 模型来实现。在每个时间步 j，模型可以选择从词表中生成一个单词或者直接从上一步抽取出的单词中进行复制:

$$\begin{aligned} &p(y_j|y_{1:j-1}, x) = \\ &p(z_j = 1|y_{1:j-1}, x) \times p(y_j|z_j = 1, y_{1:j-1}, x) + \\ &p(z_j = 0|y_{1:j-1}, x) \times p(y_j|z_j = 0, y_{1:j-1}, x) \end{aligned} \tag{11.52}$$

其中的 $z_j = 1$ 表示当前时间步需要从词表中生成一个字词，$z_j = 0$ 表示当前时间步直接从抽取单词集合中复制一个单词。在复制抽取的单词时，我们可以引入注意力机制来提升性能，即把分布 $p(y_j|z_j = 0, y_{1:j-1}, x)$ 替换成:

$$p(\tilde{a}_j^i|x, y_{1:j-1}) = \begin{cases} p(a_j^i|x, y_{1:j-1}) & q_i > \epsilon \\ 0 & \text{other} \end{cases} \tag{11.53}$$

其中 a_j^i 表示时间步 j 时单词 w_i 的注意力权重。

2. 句子粒度抽取与生成结合方法

FastRL[34] 是一种以句子为粒度的抽取与生成方法，其生成摘要的过程也可以大致分为两个步骤: (1) 抽取器提取文章中重要的句子; (2) 生成器对第一步提取出来的句子进行压缩和改述，并生成一段连贯的摘要。与 Bottom-Up 的不同之处在于，FastRL 使用了强化学习的方法对抽取和生成的两阶段模型进行了联合训练，使用了生成模型的摘要结果作为奖励函数来优化抽取模型。FastRL 的模型所使用的抽取器神经网络结构如图11.9所示。

抽取器共由三个部分组成。第一个部分是一个卷积神经网络，通过卷积和池化的方式为文本中的每个句子计算出一个表示 $\boldsymbol{r}_j, j = \{1, 2, ..., n\}$。第二个部分是一个双向长短期记忆网络，它以 $\boldsymbol{r}_j$ 为输入进一步为文本中的每个句子计算出结合全局信息的表示 $\boldsymbol{h}_j, j = \{1, 2, ..., n\}$。第三部分是一个长短期记忆网络，负责完成最终的句子选择，这个部分引入了两跳 (2-hop) 注意力机制，首

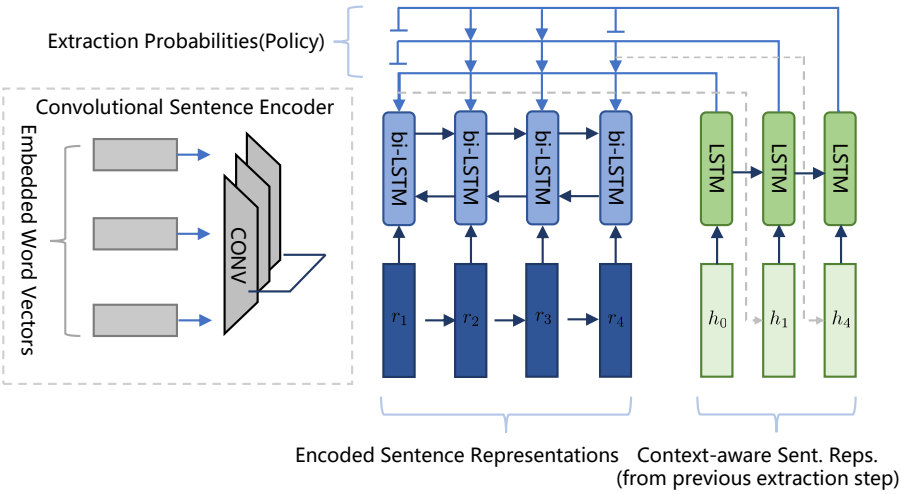

图 11.9 FastRL 模型的抽取模块[34]

先计算 $\boldsymbol{h}_j$ 对应的包含上下文信息的表示 $\boldsymbol{e}_t$，再用其计算每个句子被抽取的概率 $P(j_t|j_1,...,j_{t-1})$：

$$u_j^t = \begin{cases} \boldsymbol{v}_p^T \tanh(\boldsymbol{W}_{p1}\boldsymbol{h}_j + \boldsymbol{W}_{p2}\boldsymbol{e}_t) & \text{if} \quad j_t \neq j_k \\ & \forall k < t \\ -\infty & \text{otherwise} \end{cases} \tag{11.54}$$

$$P(j_t|j_1,...,j_{t-1}) = \text{softmax}(u^t) \tag{11.55}$$

其中 $\boldsymbol{e}_t$ 通过以下公式计算得到:

$$a_j^t = \boldsymbol{v}_g^T \tanh(\boldsymbol{W}_{g1}\boldsymbol{h}_j + \boldsymbol{W}_{g2}\boldsymbol{z}_t) \tag{11.56}$$

$$\alpha^t = \text{softmax}(a^t) \tag{11.57}$$

$$\boldsymbol{e}_t = \sum_j \alpha_j^t \boldsymbol{W}_{g1}\boldsymbol{h}_j \tag{11.58}$$

其中的 $\boldsymbol{z}_t$ 也是由第三部分的长短期记忆网络计算得到。上述公式中所有的 $\boldsymbol{W}$ 和 $\boldsymbol{v}$ 都是可以学习的参数。只要抽取器选择出的句子足够准确，生成器使用简单的 Seq2Seq 模型就可以取得足够好的压缩和改述结果。

因为需要同时训练抽取器和生成器，但抽取器和生成器之间无法直接传递梯度信息，所以 FastRL 使用了强化学习中的策略梯度来协同训练抽取器和生成器：用生成器生成的结果与真实摘要进行比较，然后将比较结果作为反馈来更新抽取器的参数。图11.10给出了强化学习的过程，其中的 $d_1, d_2, ..., d_n, s_t$ 是文本正文以及摘要中的句子。

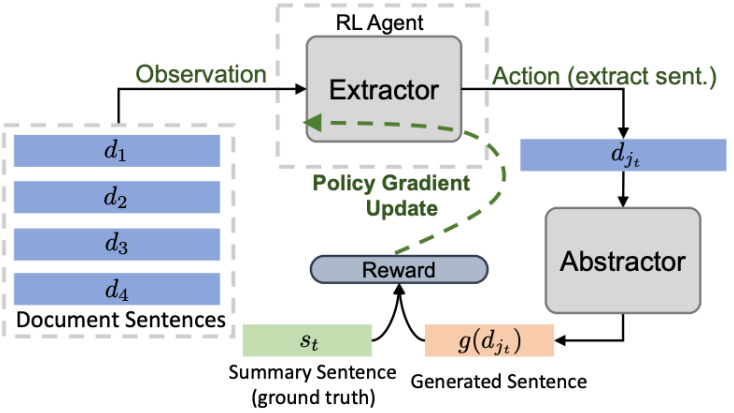

图 11.10　FastRL 利用生成器结果的反馈来更新抽取器[34]

11.4 文本摘要的评测

与文本分类、命名实体等其他自然语言处理任务的评测相比，由于关键信息的选取和文摘的表述没有统一的标准答案，因此文本摘要的人工评测和自动评测都困难很多。与机器翻译、对话系统等面向内容生成（Content Generation）的任务一样，文本摘要也有多维度的评价系统，包括人工评测方案和自动评测指标。

文本摘要的评测按照与任务的相关性可以分为两类，内在评价（Intrinsic Evaluation）方法和外在评价（Extrinsic Evaluation）方法。**内在评价**与摘要任务相关，它通过直接分析摘要的质量来评价摘要系统，一般从总体和细分维度两个粒度进行评估。其中，总体评估是指从总体上评价摘要的质量，并给出一个综合分数。细分维度一般可以从“与参考标准的一致性”和“类人性”两个角度来考虑，多方面地对摘要的质量进行评价。**外在评价**则是一种间接的评价方法，与系统的功能对应，将摘要应用于下游任务或者实际应用系统中，根据摘要在这些任务和系统中产生的实际效果来间接评估摘要的质量。这里我们仅讨论内部评价方法。

一般而言，在评测文本摘要的质量时会关注以下 5 个细分维度：

(1) 信息量（Informativeness），即摘要的内容含量，它可以衡量一段文本所提供的新信息的程度。Peyrard [35] 对信息量作出了直观上的定义：阅读者一般都拥有背景知识和常识，如果一段摘要能使其获得新信息或者产生认知上的变化，则可认为此摘要具有丰富的信息量。

(2) 非冗余性（Non-redundancy），它体现了摘要精简的特性，即不能反复使用重复或相似的文本描述同一个关键点。
(3) 流畅度（Fluency），它包含了摘要结构的连贯性和语法的正确性两个方面。
(4) 忠实度（Faithfulness），也称为相关性（Relevance），即摘要内容是否忠于原文。作为原文的子集，摘要不能包含凭空产生的信息，也不能包含与原文和事实相悖的内容。
(5) 聚焦程度（Focus），也称为显著性（Saliency），它衡量了摘要包含关键信息的程度。除了主要信息，原文档往往具有大量的细节描述和补充内容等次要信息。通过聚焦程度，可以评估一个摘要系统的甄别并捕捉原文主要信息的能力。

以上多数评测维度需要借助人工方式进行。评测者一般会根据预定义好的评测指南和范例对摘要质量进行评估。人工评测能灵活应用于多种场景和标准，适合所有的主观性任务。因此，在目前的文本摘要任务中，人工评测被认为是评价模型优劣的黄金标准。但是，人工评测也存在成本高昂和结果难以复现等问题。相比人工评测，自动评测具有方便快捷、容易复现等优势，因此被广泛应用于摘要评价。尤其是在模型开发的早期阶段，自动评测能帮助开发者快速定位问题。在大多数情况下，自动摘要的评价可以将人工评测方案与自动评测指标相结合，对模型在测试集上的整体效果做自动评测，并随机采样部分测试点进行人工评测，从而兼顾评测的效率和质量。

11.4.1 人工评测

人工评测按照执行方式一般分为两类：**逐点评估**（Point-wise）和**逐对评估**（Pair-wise）。在逐点评估中，评估者对系统产生的每一个结果按照预定义的维度进行评估打分。一个常见的方案是李克特五点量表（Likert Scale）。给定原始文档和模型输出的摘要，评估者会按照 1～5 分对摘要某一方面的评测维度进行打分。假设评测维度是流畅度，则 1～5 的分值依次对应：非常不流畅、不流畅、一般、流畅、非常流畅五种程度。但是，逐点评估具有很强的主观性，导致评估者之间的偏差较大，一致性很低。在逐对评估中，给定相同的原始文档和两个不同系统的输出摘要 A 和 B，评估者需要判断 A 与 B 相比哪个更好。与逐点评估的多选项打分相比，逐对评估采用两两比较的方式，降低了评估难度，可以提高评测结果的一致性。但是，如果存在多个需要评测的摘要系统，总评估次数会随着系统数目的增加而呈 $O(N^2)$ 的复杂度上升，成本较高。

人工评测存在主观性，包括摘要任务本身的主观性和评估者自身的主观性。为了尽可能消除人工评测的主观性偏差，一般会让多个评估者对同一条数据进行独立重复打分。因此，衡量多个评估者之间的评测一致性是一个重要的过程。一致性不仅可以体现人工评测质量的高低，还能反映评测任务的难易程度。**Fleiss 卡帕系数**（Fleiss's κ），也称 Kappa 系数，是度量多个系统一致性的方法，也经常被用于计算人工评测的一致性。定义样本的总数为 N，类别的总数为 K，每一个样本有 n 个标注者进行标注，n_{ij} 是将第 i 个样本标注为类别 c_j 的标注者数量。在第 i 个样本上，标注者之间的一致性可以采用两两标注者之间的一致性进行度量。其中，所有一致的两个标注者的组合数为 $C_{n_{ij}}^2 = \frac{1}{2} n_{ij}(n_{ij}-1)$，而每一个样本有 n 个标注者，则所有可能的两个标注者组合数

为 $C_n^2 = \frac{1}{2}n(n-1)$。因此，第 i 个样本的标注一致性可以用下式表示：

$$P_i = \frac{1}{C_n^2}\sum_{j=1}^{K} C_{n_{ij}}^2 = \frac{1}{n(n-1)}\sum_{j=1}^{K} n_{ij}(n_{ij}-1) = \frac{1}{n(n-1)}(\sum_{j=1}^{K} n_{ij}^2 - n). \tag{11.59}$$

所有样本的平均一致性可以用下式表示：

$$\bar{P} = \frac{1}{N}\sum_{i=1}^{N} P_i = \frac{1}{Nn(n-1)}(\sum_{i=1}^{N}\sum_{j=1}^{K} n_{ij}^2 - Nn). \tag{11.60}$$

上式计算的标注一致性比较直观，但是尚未考虑两个标注者随机一致的情况，即随机地对样本给出一致或不一致的结果。因此需要另外计算标注者之间随机一致的概率 $\bar{P}_e$。首先，计算得到每个类别 c_j 出现的概率为：

$$P_j = \frac{1}{Nn}\sum_{i=1}^{N} n_{ij}, \quad \sum_{j=1}^{K} P_j = 1. \tag{11.61}$$

两个标注者以 P_j 的概率随机标记，将某个样本同时标记为 c_j 的概率为 P_j^2。因此，所有类别上的随机标注概率为：

$$\bar{P}_e = \sum_{j=1}^{K} P_j^2. \tag{11.62}$$

最后，代入 $\bar{P}$ 和 $\bar{P}_e$，Fleiss 卡帕系数计算如下：

$$\kappa = \frac{\bar{P} - \bar{P}_e}{1 - \bar{P}_e}. \tag{11.63}$$

Fleiss 卡帕系数在不同的区间显示不同的一致性程度。系数小于 0 时表示没有一致性，0～0.20 表示轻微（Slight）一致，0.21～0.40 表示一般（Fair）一致，0.41～0.60 表示中等（Moderate）一致，0.61～0.80 表示显著（Substantial）一致，0.81～1.00 表示完美（Perfect）一致。

人工评测的一致性评估主要用来评价标注质量，并体现了标注任务的难易程度。一般而言，标注者需要具备一定的领域专家知识，且需要对原文内容和摘要内容做整体比较，因此摘要任务的人工评测成本高昂，花费时间长。另外，标注者之间的差异会增大标注结果的方差，导致研究结果难以复现。以上问题表明，人工评测也存在着很大挑战。摘要任务的场景和评价维度多种多样，如何实现高质量、易复现的人工评测方案是一个值得深入研究的问题。

11.4.2 自动评测

人工评测虽然可以提供丰富的信息，但是大规模的人工评测耗时长、工作量大、成本高。自动评测则可以提供高效、低成本、一致的评测，在模型开发过程中这些特点都受到研究人员青睐。

随着评测技术的发展，自动评价结果也具有了更好的指导意义。

1. 面向召回率的要点评估

ROUGE [27] (Recall-Oriented Understudy for Gisting Evaluation), 称为**面向召回率的要点评估**, 也是文本摘要中最常用的自动评价指标之一。ROUGE 与机器翻译的评价指标 BLEU 的类似，能根据机器生成的候选摘要和标准摘要（参考答案）之间词级别的匹配来自动为候选摘要评分。ROUGE 包含一系列变种，其中应用最广泛的是 ROUGE-N，它统计了 n-gram 词组的召回率，通过比较标准摘要和候选摘要来计算 n-gram 的结果。给定标准摘要集合 $S = \{Y^1, Y^2, ..., Y^M\}$ 以及候选摘要 $\hat{Y}$，则 ROUGE-N 的计算公式如下：

$$\text{ROUGE-N} = \frac{\sum_{Y \in S} \sum_{n\text{-gram} \in Y} \min[\text{Count}(Y, n\text{-gram}), \text{Count}(\hat{Y}, n\text{-gram})]}{\sum_{Y \in S} \sum_{N\text{-gram} \in Y} \text{Count}(Y, n\text{-gram})}. \tag{11.64}$$

其中 n-gram 是 Y 中所有出现过的长度为 n 的词组，$\text{Count}(Y, n\text{-gram})$ 是 Y 中 n-gram 词组出现的次数。

我们以两段摘要文本为例给出了 ROUGE 分数的计算过程：候选摘要 $\hat{Y} = \{\text{a dog is in the garden}\}$，标准摘要 $Y = \{\text{there is a dog in the garden}\}$。可以按照公式11.64计算 ROUGE-1 和 ROUGE-2 的分数为：

$$\text{ROUGE-1} = \frac{|\text{is, a, dog, in, the, garden}|}{|\text{there, is, a, dog, in, the, garden}|} = \frac{6}{7} \tag{11.65}$$

$$\text{ROUGE-2} = \frac{|\text{(a dog), (in the), (the garden)}|}{|\text{(there is), (is a), (a dog), (dog in), (in the), (the garden)}|} = \frac{1}{2} \tag{11.66}$$

需要注意的是 ROUGE 是一个面向召回率的度量，因为公式11.64的分母是标准摘要中所有 n-gram 数量的总和。相反地，机器翻译的评价指标 BLEU 是一个面向精确率的度量，其分母是候选翻译中 n-gram 的数量总和。因此，ROUGE 体现的是标准摘要中有多少 n-gram 出现在候选摘要中，而 BLEU 体现了候选翻译中有多少 n-gram 出现在标准翻译中。

另一个应用广泛的 ROUGE 变种是 ROUGE-L，它不再使用 n-gram 的匹配，而改为计算标准摘要与候选摘要之间的最长公共子序列，从而支持非连续的匹配情况，因此无需预定义 n-gram 的长度超参数。ROUGE-L 的计算公式如下：

$$R = \frac{\text{LCS}(\hat{Y}, Y)}{|Y|}, \quad P = \frac{\text{LCS}(\hat{Y}, Y)}{|\hat{Y}|}, \tag{11.67}$$

$$\text{ROUGE-L}(\hat{Y}, Y) = \frac{(1 + \beta^2)RP}{R + \beta^2 P}. \tag{11.68}$$

其中，$\hat{Y}$ 表示模型输出的候选摘要，Y 表示标准摘要。$|Y|$ 和 $|\hat{Y}|$ 分别表示摘要 Y 和 $\hat{Y}$ 的长度，$\text{LCS}(\hat{Y}, Y)$ 是 $\hat{Y}$ 与 Y 的最长公共子序列长度，R 和 P 分别为召回率和精确率，ROUGE-L 是两者

的加权调和平均数，β 是召回率的权重。在一般情况下，β 会取很大的数值，因此 ROUGE-L 会更加关注召回率。

还是以上面的两段文本为例，可以计算其 ROUGE-L 如下：

$$\text{ROUGE-L}(\hat{Y}, Y) \approx \frac{\text{LCS}(\hat{Y}, Y)}{\text{Len}(Y)} = \frac{|\text{a, dog, in, the, garden}|}{|\text{there, is, a, dog, in, the, garden}|} = \frac{5}{7} \tag{11.69}$$

2. 基于嵌入表示的度量

ROUGE 是基于候选文本与参考文本之间的精准匹配来评价文本的质量。但是在一些情况下，不同的词语或短语可以表达同一种语义。例如，“立刻”和“马上”，两者的含义相同，但是基于 n-gram 的匹配就会失效。一个常用的解决方案为基于嵌入表示的度量（Embedding-Based Metrics），使用分布式词嵌入来计算词语之间的相似度，具体来说包括静态词向量和上下文相关的词向量两种方案。

对于静态词向量方案，词表中所有的词语都对应一个固定不变的分布式词嵌入向量。静态分布式词嵌入可以通过 Word2Vec [36] 或 Glove [37] 等方法预训练获得。给定标准摘要 $Y = \{y_1, ..., y_{|Y|}\}$ 和候选摘要 $\hat{Y} = \{\hat{y}_1, ..., \hat{y}_{|\hat{Y}|}\}$，其静态词嵌入向量序列为 $\boldsymbol{E} = (\boldsymbol{e}(y_1), ..., \boldsymbol{e}(y_{|Y|}))$ 和 $\hat{\boldsymbol{E}} = (\boldsymbol{e}(\hat{y}_1), ..., \boldsymbol{e}(\hat{y}_{|\hat{Y}|}))$。可以将摘要中每个词的词向量结合起来计算文本级特征向量，进而计算标准摘要和候选摘要的文本级特征向量的余弦相似度。一种常见的计算文本级特征向量的方法为词向量平均（Embedding Average），即把文本中所有词的词向量做平均得到文本级特征向量：

$$\boldsymbol{e}(Y) = \frac{\sum_{y_i \in Y} \boldsymbol{e}(y_i)}{|Y|}. \tag{11.70}$$

得到文本级特征向量后，可以计算摘要的相似度 $S(Y, \hat{Y}) = \frac{\boldsymbol{e}(Y) \cdot \boldsymbol{e}(\hat{Y})}{||\boldsymbol{e}(Y)||\ ||\boldsymbol{e}(\hat{Y})||}$。

除了直接获取文本级特征向量来计算相似度，还可以首先计算参考摘要和候选摘要中的词两两之间的余弦相似度，然后基于这些词级相似度得到文本级相似度。一种常见的方案是贪心匹配（Greedy Matching），其文本级相似度 $S(Y, \hat{Y})$ 可计算如下：

$$S(Y, \hat{Y}) = \frac{1}{2}[GM(Y, \hat{Y}) + GM(\hat{Y}, Y)], \tag{11.71}$$

$$GM(Y, \hat{Y}) = \frac{1}{|Y|} \sum_{y_i \in Y} \max_{\hat{y}_j \in \hat{Y}} S(y_i, \hat{y}_j). \tag{11.72}$$

其中，$S(y_i, \hat{y}_j) = \frac{\boldsymbol{e}(y_i) \cdot \boldsymbol{e}(\hat{y}_j)}{||\boldsymbol{e}(y_i)||\ ||\boldsymbol{e}(\hat{y}_j)||}$。因为 $GM(Y, \hat{Y})$ 具有非对称性，所以需要对 $GM(Y, \hat{Y})$ 和 $GM(\hat{Y}, Y)$ 做平均得到最终的文本级相似度。

由于词语在不同的上下文中可以有不同的语义，而静态词向量难以应对这样的情况，所以还可以使用上下文相关的词向量来计算相似度，比如 BERTScore [38] 使用预训练语言模型 BERT 获得摘要的上下文相关的词向量序列。BERTScore 仍然使用了贪心匹配方式，将 Y 中的每个词与 $\hat{Y}$

中的每个词做匹配来计算召回率 R_{BS}，这里相似度分数简化为计算 $\boldsymbol{e}(y_i) \cdot \boldsymbol{e}(\hat{y}_j)$。同时，用类似的方式可计算精确率 P_{BS}，并得到 F1 值 F_{BS} 如下：

$$R_{BS} = \frac{1}{|Y|} \sum_{y_i \in Y} \max_{\hat{y}_j \in \hat{Y}} \boldsymbol{e}(y_i) \cdot \boldsymbol{e}(\hat{y}_j), \tag{11.73}$$

$$P_{BS} = \frac{1}{|\hat{Y}|} \sum_{\hat{y}_i \in \hat{Y}} \max_{y_j \in Y} \boldsymbol{e}(\hat{y}_i) \cdot \boldsymbol{e}(y_j), \tag{11.74}$$

$$F_{BS} = 2\frac{P_{BS} R_{BS}}{P_{BS} + R_{BS}}. \tag{11.75}$$

有研究表明，稀有词比通用词更能指示句子的相似度，因为通用词频繁出现在大量的文本中，计算它们的词向量相似度会影响到对关键内容的考量。因此，BERTScore 采用整个测试集中的逆文档频率（Inverse Document Frequency, IDF）对相似度进行加权。以召回率 R_{BS} 为例，加权后的公式为：

$$R_{BS} = \frac{\sum_{y_i \in Y} \text{IDF}(y_i) \max_{\hat{y}_j \in \hat{Y}} \boldsymbol{e}(y_i) \cdot \boldsymbol{e}(\hat{y}_j)}{\sum_{y_i \in Y} \text{IDF}(y_i)}. \tag{11.76}$$

实验结果表明，相比 ROUGE 等基于精准词匹配的指标，基于词嵌入的度量与人工评价有更高的统计相关性。但是，预训练词向量的质量会一定程度上影响评估的效果。

11.5 文本摘要语料库

如前所述文本摘要被广泛应用于多种场景中，任务类型包括单文档摘要、多文档摘要、对话摘要、跨语言文本摘要和多模态文本摘要等多种类类型。本节中将按照任务类型对文本摘要常见语料库进行介绍。

11.5.1 单文档摘要语料库

- **CNN/DailyMail** [6]: 该数据集是被广泛关注和研究的短文本摘要数据集，它包含了 311,672 个新闻/摘要对，新闻数据来源是美国有线电视新闻网和《每日邮报》。新闻平均长度为 766 个词（29.74 个句子），摘要的平均长度为 53 个词（3.72 个句子）。
- **LCSTS** [39]: 该数据集是常用的中文短文本摘要数据集，包含 240 万个新闻/摘要对，数据来源为微博认证的官方新闻。
- **Arxiv/PubMed** [40]: 这两个数据集是被广泛应用的长文本数据集,分别来源于 arXiv 和 PubMed 的学术网站上的论文和摘要。ArXiv 数据集包含 21.5 万个论文/摘要对，论文平均长度为 4938 个词，摘要平均长度为 220 个词。PubMed 数据集包含 13.3 万个论文/摘要对，论文平均长度为 3016 个词，摘要平均长度为 203 个词。

11.5.2 多文档摘要语料库

- **Multi-News** [41]: 该数据集是大规模的多文档摘要数据集,场景是新闻文档,数据来源是 Newser 网站。此网站上的新闻稿会引用多个新闻源，这些新闻源被归为输入文档集合。Multi-News 共包含 56216 个文档集合/摘要对，文档集合的平均长度为 2103 个词，摘要的平均长度为 264 个词。
- **WikiSum** [42]: 该数据集是基于英文维基百科的多文档摘要数据集。在每个实例中，输入由维基百科主题（文章标题）和非维基百科参考文档的集合组成，目标是生成维基百科文章的精简文本。数据集以 8：1：1 的比例被划分为训练集、验证集和测试集，分别包含 1,865,750、233,252 和 232,998 个样本。

11.5.3 对话摘要语料库

- **SAMSum** [43]: 该数据集是一个常用的在线聊天的摘要数据集，构造方式比较特殊，是由语言专家模拟人类实际交流的特征构建的虚拟对话和对应的摘要。它包含 16,369 个对话段，大多数为双人对话，对话的平均长度为 120 个词，摘要的平均长度为 23 个词。
- **DialogSum** [44]: 该数据集是一个包含现实生活场景对话的摘要数据集，这些对话涵盖了广泛的日常生活主题和面对面交流场景。它包含 13,460 个对话，对话的平均长度为 190 个词，摘要的平均长度为 30 个词。
- **AMI/ICSI** [45, 46]: AMI 和 ICSI 是经典的会议摘要数据集。AMI 是关于工业环境中产品设计的会议数据集。它由 137 场会议组成，包含会议记录及其相应的会议摘要。ICSI 数据集是由一个学术会议数据集组成，来自于伯克利的国际计算机科学研究所 (ICSI) 举行的 59 次每周小组会议，以及对应的摘要。与 AMI 不同的是，ICSI 会议的内容是专门针对学生之间关于研究的讨论。

11.5.4 多模态文本摘要语料库

- **MSMO** [8]: 该数据集一个公开的多模态新闻摘要数据集，数据来源是《每日邮报》。数据集包含了 31.4 万条带有图像的新闻文档，每篇新闻文档平均包含 6.6 张图片，文档和摘要的平均长度分别是 720 和 70。
- **How2** [47]: 该数据集包含大约 8 万个教学视频（约 2,000 小时）以及相关的英文字幕和视频摘要文本。其中，大约 300 小时的视频内容还通过众包的方式翻译成葡萄牙语。

11.5.5 跨语言文本摘要语料库

- **Global Voices** [7]: 该数据集关注多语言新闻的英文摘要，主要包含 15 种语言的新闻及其摘要。Global Voices 网站采用众包的方式人工标注了高质量的英文摘要，对于非英语的新闻文档，其均有对应的英文翻译。
- **En2ZhSum/Zh2EnSum** [48]: 该数据集使用了自动机器翻译系统将常见的基准文本摘要数据

集进行了翻译。其中，En2ZhSum 将 CNN/Dailymail 和 MSMO 两个数据集的摘要内容从英文翻译到中文。而 Zh2EnSum 则是将 LCSTS 数据集的摘要内容从中文翻译到英文。数据集中的其他信息与原始数据集相比保持不变。

- **WikiLingua** [49]: 该数据集包括来自 WikiHow 的 18 种语言的文档和摘要，总计约 77 万条文章和摘要对。在 WikiHow 中，文章包含有描述操作步骤的图像。通过对齐这些图像的文本，可以将跨语言的文章-摘要对进行对齐和抽取。

11.6 延伸阅读

尽管基于深度学习尤其是预训练技术的文本摘要已经取得了显著的进展，但在真实场景大规模应用文本摘要仍面临着诸多挑战。首先是数据资源的问题，取得高质量的摘要数据往往成本高昂，有时候我们必须要面临低资源情况下训练数据缺乏的问题。其次是摘要的场景丰富多样，不仅包括文本的类别（新闻、评论、学术论文等），还包括多模态数据（对话、图像、视频等），这些场景领域差异大，导致训练好的摘要模型难以迁移。最后，文本摘要的评估也是亟待解决的难题，设计合理的高效的评估方法能促进模型的迭代，但其具有很大的挑战性。

得益于大规模预训练模型的成功，自动文本摘要系统的性能有了巨大的提升。预训练模型以自监督任务在海量文本语料库上进行预训练，然后在下游摘要任务上进行微调，实验结果表明只需经过有限的训练样本即可在各种摘要基准数据集上取得最先进的性能[19, 23, 50, 51]。这表明预训练模型是解决低资源摘要问题的十分有前景的方向。但是，预训练模型也存在着推理速度较低，微调优化难度较高等问题。最近基于提示学习的预训练微调策略则聚焦于提高预训练模型的效率[52]。

强化学习（Reinforcement Learning, RL）的训练策略可以面向任何用户自定义的指标对模型进行全局优化，包括许多不可微分的指标（ROUGE 等）。这些指标可作为训练摘要模型的反馈[28, 31, 53]。通过这种方式，我们可以通过利用外部资源和不同数据集的特征来改进当前的强化学习模型。但是，强化学习也有训练困难，摘要流畅度损失等问题。目前，基于重排序的后处理方法受到了摘要领域的广泛关注，它通过模型的不同解码采样策略得到多个候选摘要，再通过排序模型对候选摘要进行全局的评估和选择[54, 55]。通过这种方式，可以一定程度上同时满足摘要的流畅度和自定义的全局指标。

目前摘要模型在新闻语料库上[6, 41] 得到了广泛的迭代更新和评估。然而，新闻的写作风格决定了大多数新闻文章的前导段落和句子即可视为摘要。这导致模型倾向于直接依据句子的位置决定摘要的内容，且往往直接进行内容的复制，而不是得到高度抽象的摘要[33]。为了让模型适应更加多样化的场景，研究人员构造了更加抽象的摘要数据集[56]，以及探索了在其他领域的摘要任务，如对话摘要任务[43]。此外，多模态文本摘要、跨语言文本摘要也受到了越来越广泛的关注。未来的趋势是更多摘要场景的数据集会不断涌现，以构建更好更具有迁移能力的摘要系统。

目前文本摘要的大多数自动评估指标，例如 ROUGE 和 BERTScore 等，不足以全面合理地评估生成摘要的整体质量[57]。我们仍然需要依赖人类专家对摘要的一些关键特征进行评估，例如事

实正确性、流畅性和相关性。因此，设计一个更好的摘要评估指标或评价系统是一个非常重要且具有挑战的方向。这些指标或系统需要更高效地更准确地捕捉与人类一致的评估特征。

11.7 习题

参考文献

[1] Luhn H P. A statistical approach to mechanized encoding and searching of literary information[J]. IBM Journal of research and development, 1957, 1(4):309-317.

[2] Luhn H P. The automatic creation of literature abstracts[J]. IBM Journal of research and development, 1958, 2(2):159-165.

[3] Neto J L, Freitas A A, Kaestner C A. Automatic text summarization using a machine learning approach[C]//Brazilian symposium on artificial intelligence. Springer, 2002: 205-215.

[4] Alami N, Meknassi M, En-nahnahi N. Enhancing unsupervised neural networks based text summarization with word embedding and ensemble learning[J]. Expert systems with applications, 2019, 123:195-211.

[5] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[C]//Advances in Neural Information Processing Systems. 2017: 5998-6008.

[6] Hermann K M, Kocisky T, Grefenstette E, et al. Teaching machines to read and comprehend[C]//Advances in Neural Information Processing Systems. 2015: 1693-1701.

[7] Nguyen K, Daumé III H. Global voices: Crossing borders in automatic news summarization[C]//Proceedings of the 2nd Workshop on New Frontiers in Summarization. 2019: 90-97.

[8] Zhu J, Li H, Liu T, et al. Msmo: Multimodal summarization with multimodal output[C]//Proceedings of the 2018 conference on empirical methods in natural language processing. 2018: 4154-4164.

[9] Mihalcea R, Tarau P. TextRank: Bringing order into text[C/OL]//Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing. Barcelona, Spain: Association for Computational Linguistics, 2004: 404-411. https://aclanthology.org/W04-3252.

[10] Brin S, Page L. The anatomy of a large-scale hypertextual web search engine[J]. Computer networks and ISDN systems, 1998, 30(1-7):107-117.

[11] Carbonell J, Goldstein J. The use of mmr, diversity-based reranking for reordering documents and producing summaries[C]//Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval. 1998: 335-336.

[12] Narayan S, Cohen S B, Lapata M. Ranking sentences for extractive summarization with reinforcement learning[C]//Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers). 2018: 1747-1759.

[13] Kim Y. Convolutional neural networks for sentence classification[J]. arXiv preprint arXiv:1408.5882, 2014.

[14] Hochreiter S, Schmidhuber J. Long short-term memory[J]. Neural computation, 1997, 9(8):1735-1780.

[15] Zhou Q, Yang N, Wei F, et al. Neural document summarization by jointly learning to score and select sentences[C]//Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2018: 654-663.

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

[17] Jing H. Using hidden markov modeling to decompose human-written summaries[J]. Computational Linguistics, 2002, 28(4):527-543.

[18] Nallapati R, Zhai F, Zhou B. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents[C]//Thirty-first AAAI conference on artificial intelligence. 2017.

[19] Liu Y, Lapata M. Text summarization with pretrained encoders[C]//Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). 2019: 3730-3740.

[20] Sutskever I, Vinyals O, Le Q V. Sequence to sequence learning with neural networks[J]. Advances in neural information processing systems, 2014, 27.

[21] 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.

[22] Kingma D P, Ba J. Adam: A method for stochastic optimization[C]//ICLR (Poster). 2015.

[23] Lewis M, Liu Y, Goyal N, et al. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension[C]//Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. 2020: 7871-7880.

[24] See A, Liu P J, Manning C D. Get to the point: Summarization with pointer-generator networks[C]// Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2017: 1073-1083.

[25] Bahdanau D, Cho K, Bengio Y. Neural machine translation by jointly learning to align and translate [J]. arXiv preprint arXiv:1409.0473, 2014.

[26] Ranzato M, Chopra S, Auli M, et al. Sequence level training with recurrent neural networks[J]. arXiv preprint arXiv:1511.06732, 2015.

[27] Lin C Y. Rouge: A package for automatic evaluation of summaries[C]//Text summarization branches out. 2004: 74-81.

[28] Paulus R, Xiong C, Socher R. A deep reinforced model for abstractive summarization[C]// International Conference on Learning Representations. 2018.

[29] Rennie S J, Marcheret E, Mroueh Y, et al. Self-critical sequence training for image captioning[J]. computer vision and pattern recognition, 2017.

[30] Liu C W, Lowe R, Serban I V, et al. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation[J]. empirical methods in natural language processing, 2016.

[31] Yu L, Zhang W, Wang J, et al. Seqgan: Sequence generative adversarial nets with policy gradient [C]//Proceedings of the AAAI conference on artificial intelligence: volume 31. 2017.

[32] Liu L, Lu Y, Yang M, et al. Generative adversarial network for abstractive text summarization[C]// Proceedings of the AAAI Conference on Artificial Intelligence: volume 32. 2018.

[33] Gehrmann S, Deng Y, Rush A M. Bottom-up abstractive summarization[C]//Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. 2018: 4098-4109.

[34] Chen Y C, Bansal M. Fast abstractive summarization with reinforce-selected sentence rewriting[C]// Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2018: 675-686.

[35] Peyrard M. A simple theoretical model of importance for summarization[C]//Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. 2019: 1059-1073.

[36] Mikolov T, Sutskever I, Chen K, et al. Distributed representations of words and phrases and their compositionality[C]//Advances in neural information processing systems. 2013: 3111-3119.

[37] 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.

[38] Zhang T, Kishore V, Wu F, et al. Bertscore: Evaluating text generation with bert[C]//International Conference on Learning Representations. 2019.

[39] Hu B, Chen Q, Zhu F. Lcsts: A large scale chinese short text summarization dataset[C]//Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. 2015: 1967-1972.

[40] Cohan A, Dernoncourt F, Kim D S, et al. A discourse-aware attention model for abstractive summarization of long documents[C]//NAACL-HLT (2). 2018.

[41] Fabbri A R, Li I, She T, et al. Multi-news: A large-scale multi-document summarization dataset and abstractive hierarchical model[C]//Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. 2019: 1074-1084.

[42] Liu P J, Saleh M, Pot E, et al. Generating wikipedia by summarizing long sequences[C]//International Conference on Learning Representations. 2018.

[43] Gliwa B, Mochol I, Biesek M, et al. Samsum corpus: A human-annotated dialogue dataset for abstractive summarization[C]//Proceedings of the 2nd Workshop on New Frontiers in Summarization. 2019: 70-79.

[44] Chen Y, Liu Y, Chen L, et al. Dialogsum: A real-life scenario dialogue summarization dataset[C]//Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021. 2021: 5062-5074.

[45] Carletta J, Ashby S, Bourban S, et al. The ami meeting corpus: A pre-announcement[C]//International workshop on machine learning for multimodal interaction. Springer, 2005: 28-39.

[46] Janin A, Baron D, Edwards J, et al. The icsi meeting corpus[C]//2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03).: volume 1. IEEE, 2003: I-I.

[47] Sanabria R, Caglayan O, Palaskar S, et al. How2: A large-scale dataset for multimodal language understanding[C]//NeurIPS. 2018.

[48] Zhu J, Wang Q, Wang Y, et al. Ncls: Neural cross-lingual summarization[C]//Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). 2019: 3054-3064.

[49] Ladhak F, Durmus E, Cardie C, et al. Wikilingua: A new benchmark dataset for cross-lingual abstractive summarization[C]//Findings of the Association for Computational Linguistics: EMNLP 2020. 2020: 4034-4048.

[50] Dong L, Yang N, Wang W, et al. Unified language model pre-training for natural language understanding and generation[J]. Advances in Neural Information Processing Systems, 2019, 32.

[51] Raffel C, Shazeer N, Roberts A, et al. Exploring the limits of transfer learning with a unified text-to-text transformer.[J]. J. Mach. Learn. Res., 2020, 21(140):1-67.

[52] Liu P, Yuan W, Fu J, et al. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing[J]. arXiv preprint arXiv:2107.13586, 2021.

[53] Pasunuru R, Bansal M. Multi-reward reinforced summarization with saliency and entailment[C]//Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers). 2018: 646-653.

[54] Zhong M, Liu P, Chen Y, et al. Extractive summarization as text matching[C]//Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. 2020: 6197-6208.

[55] Liu Y, Liu P. Simcls: A simple framework for contrastive learning of abstractive summarization [C]//Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers). 2021: 1065-1072.

[56] Narayan S, Cohen S B, Lapata M. Don’t give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization[C]//Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. 2018: 1797-1807.

[57] Kryściński W, Keskar N S, McCann B, et al. Neural text summarization: A critical evaluation[C]//Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). 2019: 540-551.

索引