第 3 章

文本预处理与分析

文本预处理与分析是自然语言处理中的一个重要研究方向，目的是通过对原始文本进行处理和分析，提取其中的有用信息。随着互联网和社交媒体的快速发展，人们产生和累积的文本数据呈爆炸式增长，如何有效地处理和分析海量的文本数据，成为各个领域所关注的问题。随着计算机计算能力的大幅提升，机器学习和深度学习都取得了长足的发展，自然语言处理越来越多地通过机器学习和深度学习工具解决问题。本章主要介绍文本预处理中的文本向量化与相似度、文本分析简介及其常用算法。

学习目标

（1）了解文本向量化和相似度的基本概念。

（2）了解文本离散化表示和文本分布式表示的常用方法。

（3）熟悉文本向量化模型 Word2Vec 和 Doc2Vec 的基本原理。

（4）掌握常用的文本相似度算法。

（5）了解结构化分析和语义化分析的常见类型。

（6）掌握文本分析常用的机器学习算法。

（7）了解文本分析常用的深度学习算法。

3.1 文本向量化与相似度

文本向量化是与文本有关的机器学习的常见前置操作。在文本向量化的过程中，根据映射方法的不同，可以将其分为文本离散化表示和文本分布式表示。在自然语言处理中，经常会涉及度量两个文本相似度的问题。在对话系统和信息检索中，度量文本之间的相似度尤为重要。

3.1.1 文本向量化与相似度简介

文本向量化与相似度的计算是文本分析中不可或缺的部分，下面将介绍文本向量化与相似度的概念和定义。

1. 文本向量化

文本向量化是指将文本表示成一系列能够表达文本语义的机读向量，它是文本表示的一种重要方式。在自然语言处理中，文本向量化是一个重要环节，其产出向量的质量将直接影响后续模型的表现。

文本向量化按照向量化的粒度可以分为以字为单位、以词为单位和以句子为单位的向量表达方法，需根据不同的情景选择不同的向量表达方法。目前对文本向量化的大部分研究都是以词为单位的。随着深度学习技术的广泛应用，基于神经网络的文本向量化已经成为自然语言处理领域的研究热点，尤其是以词为单位的文本向量化。Word2Vec 模型是目前以词为单位的文本向量化中最典型的生成词向量的工具，其特点是将所有的词向量化，这样即可度量词与词之间的关系、挖掘词与词之间的联系。也有一部分研究将句子作为文本处理的基本单元，于是产生了 Doc2Vec 和 Str2Vec 等模型。

2. 文本相似度

文本相似度在文本检索、文本分类、文档聚类、主题分析、机器翻译、文本摘要等任务中的研究和应用越来越重要。词与词之间的相似度是文本相似度的重要组成部分，是句子、段落和文档相似度的基础。

文本相似度用来衡量两个文本在语义上的相似程度，定义为

$$\mathrm{Sim}(\boldsymbol{A},\boldsymbol{B})=\frac{\lg P(\mathrm{common}(\boldsymbol{A},\boldsymbol{B}))}{\lg P(\mathrm{description}(\boldsymbol{A},\boldsymbol{B}))} \tag{3-1}$$

其中，common($\boldsymbol{A},\boldsymbol{B}$)是文本 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 的共性信息；description($\boldsymbol{A},\boldsymbol{B}$)是描述文本 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 的全部信息。式（3-1）代表文本相似度与文本共性信息呈正相关。

文本相似度一般可以用[0,1]中的实数表示，该实数可通过计算语义距离获得。文本相似度与语义距离呈负相关，语义距离越小，文本相似度越高；语义距离越大，文本相似度越低。通常用式（3-2）表示相似度与语义距离的关系。

$$\mathrm{Sim}(\boldsymbol{A},\boldsymbol{B})=\frac{\alpha}{\mathrm{Dis}(\boldsymbol{A},\boldsymbol{B})+\alpha} \tag{3-2}$$

其中，Dis($\boldsymbol{A},\boldsymbol{B}$)表示文本 $\boldsymbol{A}$、$\boldsymbol{B}$ 之间的非负语义距离；α 为调节因子，保证当语义距离为 0 时，式（3-2）具有意义。

3.1.2 常用文本向量化方法

文本向量化通过数值向量来表示文本的语义，主要分为文本离散化表示和文本分布式表示。

1. 文本离散化表示

文本离散化表示是一种基于规则和统计的文本向量化方法，常用的方法有词集（Set-Of-Words，SOW）模型和词袋（Bag-Of-Words，BOW）模型，这两种模型都以词与词之间保持独立性、没有关联为前提，以所有文本中的词形成一个词典，然后根据词典统计词的出现频数。但这两种模型也存在不同之处。例如，SOW 模型中的独热表示，只要单个文本中的词出现在词典中，不管出现多少次，都将其向量值置为 1。而 BOW 模型只要文本中一个词出现在词典中，就将其向量值加 1，出现多少次就加多少次。文本离散化表示的特点是忽略文本信息中的语序信息和语境信息，仅将其反映为若干维度的独立概念。由于模型本身存在无法解决的问题，如主语和宾语的顺序问题，会导致模型无法理解文本的原意，如“我为你鼓掌”和“你为我鼓掌”两个句子之间的区别。

1）独热表示

独热表示是指用一个长的向量表示词典中的一个词，向量长度为词典的长度，每个向量只有一个维度为 1，其余维度全部为 0，向量中维度为 1 的位置表示该词在词典中的位置。例如，对于“建设农业强国，利器在科技”和“农业强国，科技助力”，首先对这两个句子进行分词并构造一个词典，词典的键是词，值是 ID，即{"建设":1,"农业":2,"强国":3, "利器":4,"在":5,"科技":6,"助力":7}；然后根据 ID 值对每个词进行向量化，用 0 和 1 代表这个词是否出现，最后得到的独热表示如表 3-1 所示。

表 3-1　独热表示

词语	独热表示
建设	[1,0,0,0,0,0,0]
农业	[0,1,0,0,0,0,0]
强国	[0,0,1,0,0,0,0]
利器	[0,0,0,1,0,0,0]
在	[0,0,0,0,1,0,0]
科技	[0,0,0,0,0,1,0]
助力	[0,0,0,0,0,0,1]

独热表示构造简单，但有如下明显的缺点。

（1）维数过大。上例只有两个句子，每个词是一个 7 维向量，随着语料的增加，维数会越来越大，将导致“维数灾难”。维数灾难是指在高维空间中数据分布变得稀疏、计算复杂度急剧增加的现象。

（2）矩阵稀疏。独热表示的每个词向量只有一个维度是有数值的，其余维度的数值都为 0。

（3）不能保留语义。独热表示的结果不能保留词在句子中的位置信息。

2）BOW 模型

BOW 模型使用一个向量表示一个句子或一个文档。BOW 模型忽略文档的词顺序、语法、句法等要素，将文档看作若干个词的集合，文档中每个词都是独立的。

BOW 模型每个维度上的数值代表每个词在句子中出现的次数。将上例中的两个句子分词后构造词典，词典的键是词，值是 ID，即{"农业":1,"强国":2,"科技":3,"建设":4,"利器":5,"在":6,"助力":7}。然后根据每个词出现的次数对文本进行 BOW 向量化，可得两个句子的 BOW 模型表示，如表 3-2 所示。

表 3-2　BOW 模型表示

文本	BOW 模型表示
建设农业强国，利器在科技	[1,1,1,1,1,1,0]
农业强国，科技助力	[1,1,1,0,0,0,1]

BOW 模型存在如下缺点。

（1）不能保留语义。不能保留词在句子中的位置信息，如“我为你鼓掌”和“你为我鼓掌”的向量化结果没有区别。“我喜欢北京”和“我不喜欢北京”这两个句子语义相反，但利用 BOW 模型得到的结果却可能认为它们是相似的文本。

（2）维数过大和矩阵稀疏。当语料增加时，维数也会增大，一个文本里不出现的词就会增多，导致矩阵稀疏。

3）TF-IDF 表示

TF-IDF 表示与 BOW 模型类似，它是在 BOW 模型的基础上给词的出现频数赋予 TF-IDF 值，对 BOW 模型进行修正，进而表示该词在文本中的重要程度。

仍以“建设农业强国，利器在科技”和“农业强国，科技助力”为例。首先，将句子分词后得到词典，即{农业,强国,科技,建设,利器,在,助力}。然后，计算每个词的 TF-IDF 值，其中“建设农业强国，利器在科技”中“科技”的 TF-IDF 值的具体计算过程如下。

$$\text{TF（科技）}=\frac{1}{6}=0.1667$$

$$\text{IDF（科技）}=\lg\frac{2}{3}=-0.1761$$

$$\text{TF-IDF}=\text{TF}\times\text{IDF}=0.1667\times(-0.1761)=-0.0294$$

“建设农业强国，利器在科技”中所有词的 TF-IDF 值如表 3-3 所示。

表 3-3　TF-IDF 值（1）

建设农业强国，利器在科技	TF-IDF 值
建设	0
农业	−0.0294
强国	−0.0294
利器	0
在	0
科技	−0.0294

“农业强国，科技助力”中所有词的 TF-IDF 值如表 3-4 所示。

表 3-4　TF-IDF 值（2）

农业强国，科技助力	TF-IDF 值
农业	−0.0440
强国	−0.0440
科技	−0.0440
助力	0

2. 文本分布式表示

文本分布式表示是将每个词根据上下文信息从高维空间映射到一个低维度、稠密的向量上。文本分布式表示的思想是，词的语义是通过上下文信息确定的，即相同语境下出现的词，其语义也相近。文本分布式表示的优点是考虑到了词与词之间存在的相似关系，减小了词向量的维度。常用的方法有基于矩阵的分布式表示［如矩阵分解（LSA）模型、概率潜在语义

分析（PLSA）模型和文档生成（LDA）模型］、基于聚类的分布式表示和基于神经网络的分布式表示（如 Word2Vec 模型和 Doc2Vec 模型）。

1）Word2Vec 模型

Word2Vec 模型其实就是简化的神经网络模型。随着深度学习技术的广泛应用，基于神经网络的文本向量化成为自然语言处理领域的研究热点。2013 年，Google 开源了用于词向量建模的工具 Word2Vec 模型，引起了工业界和学术界的广泛关注。首先，Word2Vec 模型可以在百万数量级的词典和上亿数量级的数据集上进行高效的训练；其次，该模型得到的训练结果可以很好地度量词与词之间的相似度。

Word2Vec 模型的输入是独热向量，根据输入和输出模式，可以分为连续词袋（Continuous Bag-Of-Word，CBOW）模型和跳字（Skip-Gram）模型。CBOW 模型的输入是某个特定词的上下文的独热向量，而输出是这个特定词的概率分布。Skip-Gram 模型和 CBOW 模型的思路相反，输入是一个特定词的独热向量，而输出是这个特定词的上下文的概率分布。CBOW 模型适用于小型语料库，而 Skip-Gram 模型在大型语料库中表现更好。

Word2Vec 模型的特点是，当模型训练好之后，并不会使用训练好的模型处理新的任务，而是使用模型通过训练数据所得的参数进行模型构建，如隐藏层的权重矩阵等参数。

（1）CBOW 模型。

CBOW 模型根据上下文可预测目标词的概率分布。CBOW 模型的神经网络包含输入层、隐藏层和输出层，其网络结构如图 3-1 所示，具体如下。

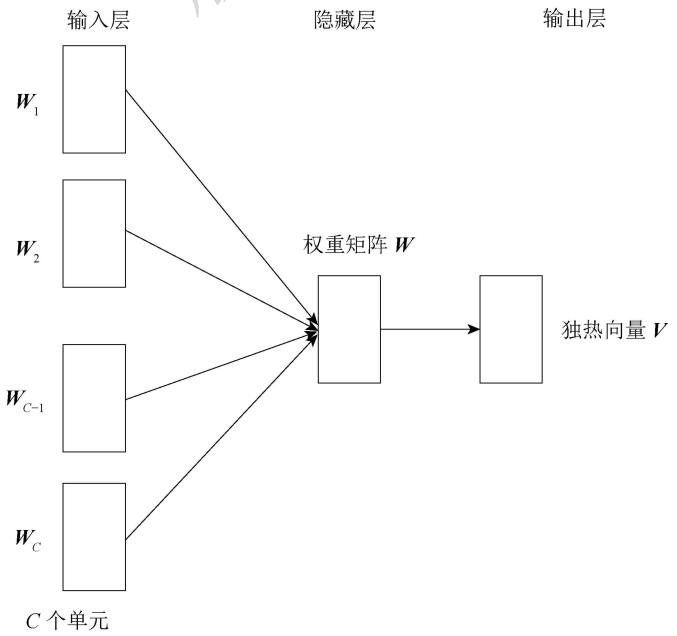

图 3-1　CBOW 模型的网络结构

①输入层含有 C 个单元，每个单元含有 V 个神经元，用于输入 V 维独热向量。

②隐藏层的神经元个数为 N，在输入层中，每个单元到隐藏层的连接权重共享一个 $V \times N$ 维的权重矩阵 $\boldsymbol{W}$。

③输出层含有 V 个神经元，隐藏层到输出层的连接权重为 $N \times V$ 维权重矩阵 $\boldsymbol{W}'$。

④输出层神经元的输出值表示每个词的概率分布，通过 softmax 函数计算每个词的概率分布。

假设词向量空间维数为 V，上下文中词的个数为 C，词典中的所有词都转换为独热向量，CBOW 模型的训练步骤如下。

①初始化权重矩阵 $\boldsymbol{W}$（$V \times N$ 维矩阵，N 为人为设定的隐藏层单元的数量），输入层的所有独热向量分别乘以共享的权重矩阵 $\boldsymbol{W}$，得到隐藏层的输入向量。

②计算隐藏层的所有输入向量的平均值，得到隐藏层的输出向量。

③将隐藏层的输出向量乘以权重矩阵 $\boldsymbol{W}'$（$V \times N$ 维矩阵），得到输出层的输入向量。

④通过激活函数处理输入向量得到输出层的概率分布。

⑤计算损失函数。

⑥更新权重矩阵。

CBOW 模型由权重矩阵 $\boldsymbol{W}$ 和 $\boldsymbol{W}'$ 确定，训练的过程就是确定 $\boldsymbol{W}$ 和 $\boldsymbol{W}'$ 的过程。权重矩阵可以通过随机梯度下降法确定，即初始化时给这些权重赋一个随机值，然后按序训练样本，计算损失函数，并计算这些损失函数的梯度，在梯度方向更新权重矩阵。

（2）Skip-Gram 模型。

Skip-Gram 模型与 CBOW 模型相反，它根据目标词预测其上下文的概率分布。假设词典中词汇量的大小为 V，隐藏层的大小为 N，相邻层的神经元是全连接的，Skip-Gram 模型的网络结构如图 3-2 所示，具体如下。

①输入层含有 V 个神经元，输入是一个 V 维的独热向量。

②输入层到隐藏层的连接权重是一个 $V \times N$ 维的权重矩阵 $\boldsymbol{W}$。

③输出层含有 C 个单元，每个单元含有 V 个神经元，隐藏层到输出层每个单元的连接权重共享一个 $V \times N$ 维的权重矩阵 $\boldsymbol{W}'$。

④对输出层每个单元使用 softmax 激活函数计算上下文的概率分布。

假设词向量空间维数为 V，上下文中词的个数为 C，词典中的所有词都转换为独热向量，Skip-Gram 模型的训练步骤如下。

①输入独热向量 $\boldsymbol{V}$。

②初始化权重矩阵 $\boldsymbol{W}$（$V \times N$ 维矩阵，N 为人为设定的隐藏层单元的数量），输入层的

所有独热向量分别乘以共享的权重矩阵 $\boldsymbol{W}$，得到隐藏层的输入向量。

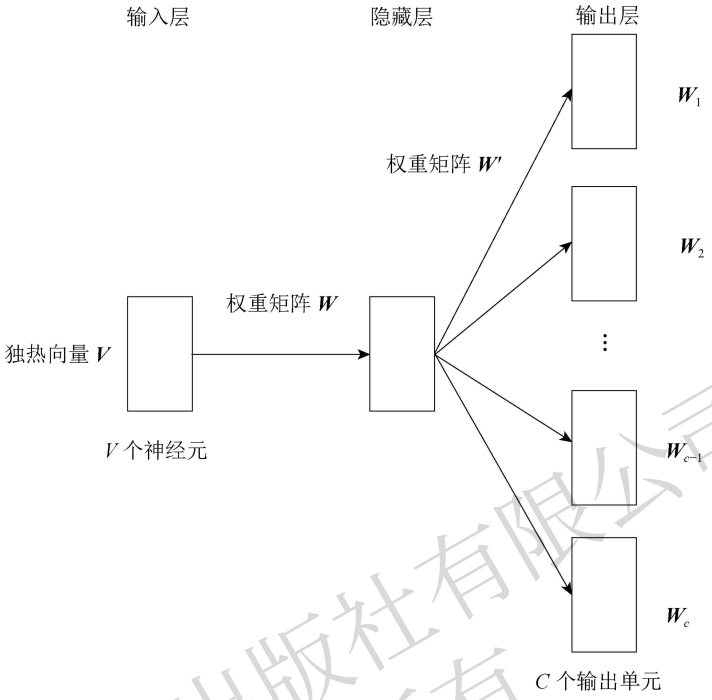

图 3-2　Skip-Gram 模型的网络结构

③将隐藏层的输出向量乘以权重矩阵 $\boldsymbol{W}'$（$V \times N$ 维矩阵），得到输出层的输入向量。

④通过激活函数处理输入向量得到输出层上下文的概率分布。

⑤输出结果。

2）Doc2Vec 模型

通过 Word2Vec 模型可获取文本的向量，一般做法是先进行分词，提取文本的关键词，用 Word2Vec 模型获取这些关键词的词向量，然后计算这些关键词的词向量的平均值，或将这些词向量拼接起来得到一个新的向量，这个新向量可以看作这个文本的向量。然而，这种方法只保留词的信息，会丢失文本中的主题信息。为此，有研究者在 Word2Vec 模型的基础上提出了 Doc2Vec 模型。

Doc2Vec 模型与 Word2Vec 模型类似，只是在 Word2Vec 模型的输入层加了一个与词向量同维度的段落向量，可以将这个段落向量看作另一个词向量。

Doc2Vec 模型有分布式记忆（Distributed Memory，DM）模型和分布式词袋（Distributed Bag-Of-Words，DBOW）模型两种，分别对应 Word2Vec 模型里的 CBOW 模型和 Skip-Gram 模型。

（1）DM 模型。

DM 模型与 CBOW 模型类似，在给定上下文的前提下，试图预测目标词的概率分布，

只不过 DM 模型的输入不仅包括上下文，还包括相应的段落。

假设词典中的词汇量大小为 V，每个词都用独热向量表示，神经网络相邻层的神经元是全连接的，DM 模型的网络结构如图 3-3 所示，具体如下。

①输入层含有 1 个段落单元、C 个上下文单元，每个单元含有 V 个神经元，用于输入 V 维独热向量。

②隐藏层的神经元个数为 N，段落单元到隐藏层的连接权重为 $V \times N$ 维矩阵 $\boldsymbol{D}$，每个上下文单元到隐藏层的连接权重共享一个 $V \times N$ 维的权重矩阵 $\boldsymbol{W}$。

③输出层含有 V 个神经元，隐藏层到输出层的连接权重为 $N \times V$ 维的权重矩阵 $\boldsymbol{W}'$。

④通过 softmax 激活函数计算输出层的神经元输出值。

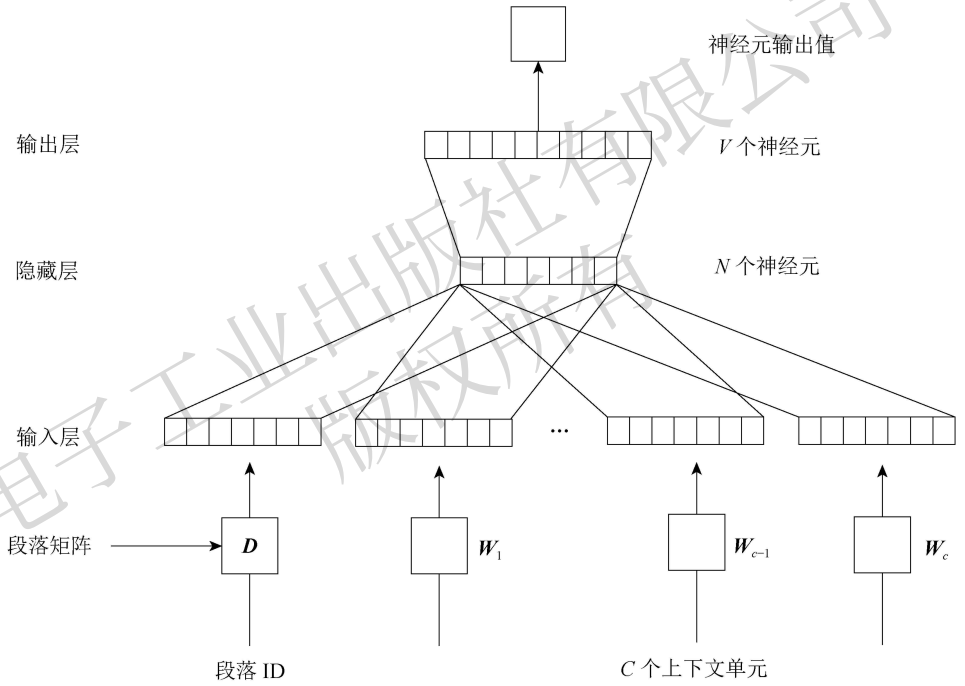

图 3-3　DM 模型的网络结构

DM 模型增加了一个与词向量长度相等的段落向量，即段落（Paragraph）ID，从输入到输出的计算过程如下。

①段落 ID 通过矩阵 $\boldsymbol{D}$ 映射成段落向量。段落向量和词向量的维数虽然一样，但是分别代表两个不同的向量空间。每个段落或句子被映射到向量空间中时，都可以用矩阵 $\boldsymbol{D}$ 的一列表示。

②上下文通过矩阵 $\boldsymbol{W}$ 映射到向量空间，用矩阵 $\boldsymbol{W}$ 的列表示。

③将对段落向量和词向量求平均后或按顺序拼接后得到的向量输入 softmax 层。

在训练过程中，段落 ID 始终保持不变，共享同一个段落向量，相当于每次预测目标词的概率时，都利用了整个句子的语义。这个段落向量也可以认为是一个词，它的作用相当于上下文的记忆单元，也可作为这个段落的主题。

在预测阶段，为文本新分配一个段落 ID，词向量和输出层的参数保持不变，重新利用随机梯度下降法训练文本，待误差收敛后即可得到新的段落向量。

（2）DBOW 模型。

DBOW 模型与 Skip-Gram 模型只给定一个词预测上下文的概率分布类似，DBOW 模型的输入只有段落向量，通过一个段落向量预测段落中随机词的概率分布。DBOW 模型的网络结构如图 3-4 所示，具体如下。

①输入层含有 1 个段落矩阵、C 个文档或段落，每个文档或段落含有 V 个神经元。隐藏层的神经元个数为 N，每个文档或段落到隐藏层的连接权重共享一个 $V \times N$ 维的权重矩阵 $\boldsymbol{W}$。

②输出层含有 V 个神经元，隐藏层到输出层的连接权重为 $N \times V$ 维的权重矩阵 $\boldsymbol{W}'$。

③通过 softmax 激活函数计算输出层的神经元输出值。

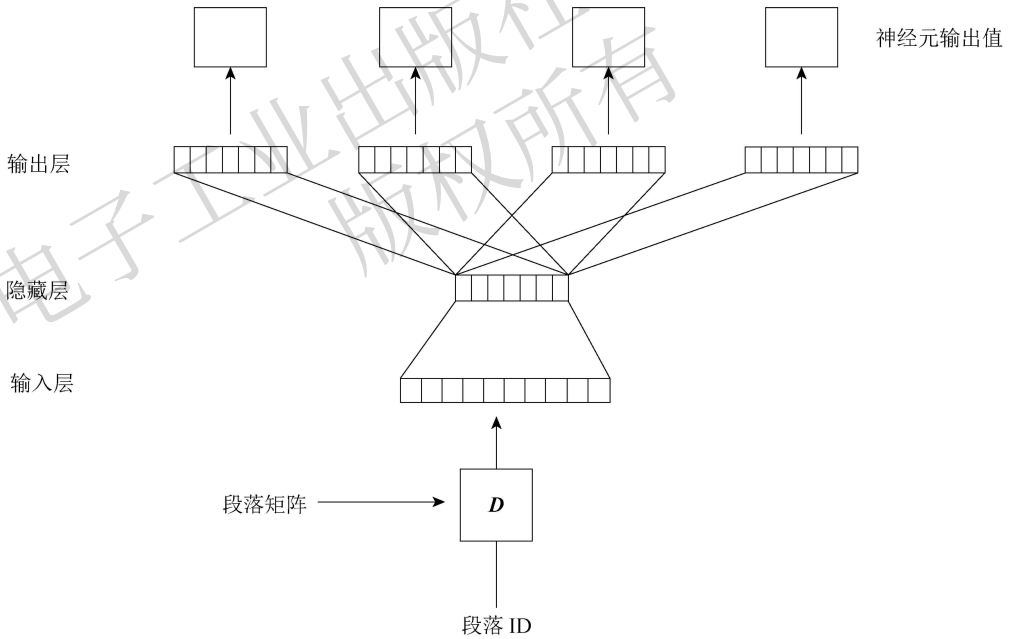

图 3-4　DBOW 模型的网络结构

DBOW 模型忽略了输入的上下文，让模型去预测段落中的随机词，在每次迭代时，从文本中采样得到一个窗口，再从这个窗口中随机采样一个词作为预测任务并让模型去预测，输入就是段落向量。

Doc2Vec 模型从输入到输出的计算过程如下。

①训练模型。在已知的训练数据中得到词矩阵 ***W***、各参数项和段落矩阵 ***D***。

②推断过程。对于新的段落，需要得到它的向量表达。具体做法是在段落矩阵 ***D*** 中添加更多的列，并且在固定参数的情况下利用上述方法进行训练，使用随机梯度下降法得到新的矩阵 ***D***，从而得到新段落的向量表达。

由于 Doc2Vec 模型完全是从 Word2Vec 模型扩展而来的，DM 模型与 CBOW 模型相对应，所以可以根据上下文词向量和段落向量预测目标词的概率分布。DBOW 模型与 Skip-Gram 模型对应，只输入段落向量，可预测从段落中随机抽取的词的概率分布。Doc2Vec 模型不仅可提取文本的语义信息，而且可提取文本的语序信息。

3.1.3 文本向量化实现

文本向量化可以通过 gensim 库实现。下面将介绍基于 Word2Vec 模型和 Doc2Vec 模型实现文本向量化的具体操作过程。

1. Word2Vec 模型文本向量化

Word2Vec 模型可通过调用 gensim 库的 Word2Vec()函数训练词向量，其语法格式如下。

```
Word2Vec(sentences, min_count=1, vector_size=100, window=5, workers=4)
```

Word2Vec()函数的常用参数及其说明如表 3-5 所示。

表 3-5　Word2Vec()函数的常用参数及其说明

参数名称	说明
sentences	接收 str。表示输入的语料库，可以是一个列表或迭代器。无默认值
min_count	接收 int。表示忽略总频率小于这个值的单词。默认为 5
vector_size	接收 int。表示生成的词向量的维度。默认为 100
window	接收 int。表示考虑前后多少个单词作为上下文。默认为 5
workers	接收 int。表示训练模型时使用的工作线程数。默认为 1

现以文本“加快建设制造强国、质量强国、航天强国、交通强国、网络强国、数字中国”为例，使用 Word2Vec 模型进行文本向量化，如代码 3-1 所示。

代码 3-1　Word2Vec 模型文本向量化

```
from gensim.models import Word2Vec
def word2vec_sentence_embedding(sentence):
    # 分词
    words = sentence.split()
    # 训练 Word2Vec 模型
    model = Word2Vec([words], vector_size=100, window=5, min_count=1, workers=4)
    # 文本向量化
```

```
    sentence_vector = model.wv[words].mean(axis=0)
    return sentence_vector
# 文本输入
sentence = '加快建设制造强国、质量强国、航天强国、交通强国、网络强国、数字中国'
# Word2Vec 模型文本向量化
vector = word2vec_sentence_embedding(sentence)
print('Word2Vec 模型文本向量化结果：')
print('sentence 的向量：\n', vector)
```

代码 3-1 的运行结果如下。

```
Word2Vec 模型文本向量化结果：
sentence 的向量：
 [-5.3622725e-04  2.3643136e-04  5.1033497e-03  9.0092728e-03
 -9.3029495e-03 -7.1168090e-03  6.4588725e-03  8.9729885e-03
…            …            …            …
  3.4736372e-03  2.1777749e-04  9.6188262e-03  5.0606038e-03
 -8.9173904e-03 -7.0415605e-03  9.0145587e-04  6.3925339e-03]
```

运行结果以向量的形式展示了文本“加快建设制造强国、质量强国、航天强国、交通强国、网络强国、数字中国”的向量化结果。

2. Doc2Vec 模型文本向量化

Doc2Vec 模型可通过调用 gensim 库的 Doc2Vec()函数训练词向量，其语法格式如下。

```
Doc2Vec(documents, min_count=2, vector_size=50, epochs=40, workers=4)
```

Doc2Vec()函数的常用参数及其说明如表 3-6 所示。

表 3-6　Doc2Vec()函数的常用参数及其说明

参数名称	说明
documents	接收 str。表示输入的文档列表。无默认值
min_count	接收 int。表示忽略总频率小于这个值的单词。默认为 5
vector_size	接收 int。表示生成的词向量的维度。默认为 100
epochs	接收 int。表示训练模型时的迭代次数。默认为 10
workers	接收 int。表示训练模型时使用的工作线程数。默认为 1

现以文本“培育创新文化，弘扬科学家精神，涵养优良学风，营造创新氛围”为例，使用 Doc2Vec 模型进行文本向量化，如代码 3-2 所示。

代码 3-2　Doc2Vec 模型文本向量化

```
from gensim.models import Doc2Vec
from gensim.models.doc2vec import TaggedDocument
def doc2vec_sentence_embedding(sentence):
    # 分词
    words = sentence.split()
    # 转换为 TaggedDocument 格式
```

```
    doc = TaggedDocument(words=words, tags=[0])
    # 训练 Doc2Vec 模型
    model = Doc2Vec([doc], vector_size=100, window=5, min_count=1, workers=4)
    # 文本向量化
    sentence_vector = model.infer_vector(words)
    return sentence_vector
# 文本输入
sentence = '培育创新文化，弘扬科学家精神，涵养优良学风，营造创新氛围'
# Doc2Vec 模型文本向量化
vector = doc2vec_sentence_embedding(sentence)
print('Doc2Vec 模型文本向量化结果：')
print('sentence 的向量：\n', vector)
```

代码 3-2 的运行结果如下。

```
Doc2Vec 模型文本向量化结果：
sentence 的向量：
 [-4.3966193e-03  2.9815948e-03 -3.0292559e-03 -2.9192781e-03
 -2.2427994e-03  2.8775514e-03 -4.0365928e-03  4.0877103e-03
…          …          …          …
-1.6047165e-03  1.2633288e-03  2.3232305e-03  1.1190206e-03
 -1.5022689e-03 -1.8358517e-03  2.0790601e-03  2.9390329e-03]
```

运行结果以向量的形式展示了文本“培育创新文化，弘扬科学家精神，涵养优良学风，营造创新氛围”的向量化结果。

3.1.4 常用文本相似度算法

在自然语言处理的文本预处理与分析任务中，常常需要判断两个文档是否相似。在问答系统中，通常会准备一些经典问题和对应的答案，当用户的问题和经典问题很相似时，系统直接返回准备好的答案；在对文本进行预处理时，需要基于文本的相似度，把重复的文本挑出来删除。文本相似度是一个非常有用的指标，可以帮助解决很多问题。常用的文本相似度算法有欧氏距离、曼哈顿距离、编辑距离、杰卡德相似度、余弦相似度和哈罗距离等。

1. 欧氏距离

假设有两个数值向量 $\boldsymbol{A}=(a_1,a_2,\cdots,a_i,\cdots,a_n)$ 和 $\boldsymbol{B}=(b_1,b_2,\cdots,b_i,\cdots,b_n)$，表示两个实例在欧氏空间中的位置，则欧氏距离的定义为

$$d=\sqrt[2]{(\boldsymbol{A}-\boldsymbol{B})\times(\boldsymbol{A}-\boldsymbol{B})^{\mathrm{T}}}=\sqrt[2]{\sum_{i=1}^{n}(a_i-b_i)^2} \tag{3-3}$$

若需要计算两个文本向量间的相似度，则 d 表示欧氏距离，$\boldsymbol{A}$ 和 $\boldsymbol{B}$ 表示两个文本向量，a_i、b_i 分别表示需要计算相似度的两个文本向量中对应位置的元素。

例如，计算“乡村振兴”和“乡村发展旅游业”之间的欧氏距离，具体计算过程如下。

（1）文本向量 $\boldsymbol{A}=$(乡,村,振,兴)，即 $a_1=$乡，$a_2=$村，$a_3=$振，$a_4=$兴，a_5、a_6、a_7均为空；文本向量 $\boldsymbol{B}=$(乡,村,发,展,旅,游,业)，即 $b_1=$乡，$b_2=$村，$b_3=$发，$b_4=$展，$b_5=$旅，$b_6=$游，$b_7=$业。

（2）规定当 $a_i=b_i$ 时，$a_i-b_i=0$；当 $a_i\neq b_i$ 时，$a_i-b_i=1$。

由此可得到文本向量 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 的欧氏距离为

$$d=\sqrt[2]{0^2+0^2+1^2+1^2+1^2+1^2+1^2}=\sqrt{5}$$

欧氏距离主要的适用场景为编码检测，只有两串编码完全一致时，才能通过检测，如果编码中有一个移位或一个错字，可能会造成较大的差异。

2. 曼哈顿距离

曼哈顿距离的计算公式与欧氏距离的计算公式非常相似，相较于欧氏距离，曼哈顿距离的计算公式将平方换成了绝对值，并去除了根号，其定义公式为

$$d=\sum_{i=1}^{n}\left|a_i-b_i\right| \tag{3-4}$$

例如，计算“乡村振兴”和“乡村发展旅游业”之间的曼哈顿距离，具体计算过程如下。

（1）文本向量 $\boldsymbol{A}=$(乡,村,振,兴)，即 $a_1=$乡，$a_2=$村，$a_3=$振，$a_4=$兴，a_5、a_6、a_7均为空；文本向量 $\boldsymbol{B}=$(乡,村,发,展,旅,游,业)，即 $b_1=$乡，$b_2=$村，$b_3=$发，$b_4=$展，$b_5=$旅，$b_6=$游，$b_7=$业。

（2）规定当 $a_i=b_i$ 时，$a_i-b_i=0$；当 $a_i\neq b_i$ 时，$a_i-b_i=1$。

由此可得到文本向量 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 的曼哈顿距离为

$$d=0+0+1+1+1+1+1=5$$

曼哈顿距离的适用场景与欧氏距离类似。

3. 编辑距离

编辑距离又称莱文斯坦（Levenshtein）距离，指的是将文本 A 转换成文本 B 需要的最少改动次数，且每次只能增加、删除或修改一个字。

例如，计算“乡村振兴”和“乡村发展旅游业”之间的编辑距离，具体计算过程如下。

（1）将“乡村振兴”删除“兴”变成“乡村振”。

（2）将“乡村振”删除“振”变成“乡村”。

（3）将“乡村”增加“发”变成“乡村发”。

（4）将“乡村发”增加“展”变成“乡村发展”。

（5）将“乡村发展”增加“旅”变成“乡村发展旅”。

（6）将“乡村发展旅”增加“游”变成“乡村发展旅游”。

（7）将“乡村发展旅游”增加“业”变成“乡村发展旅游业”。

由此可以看出，将“乡村振兴”转换成“乡村发展旅游业”至少需要 7 次改动，所以，“乡村振兴”和“乡村发展旅游业”的编辑距离是 7。

编辑距离是对称的，即将文本 *A* 转换成文本 *B* 的最小变动次数和将文本 *B* 转换成文本 *A* 的最小变动次数是相等的。

编辑距离适用于拼写检查、判断 DNA 相似度等场景中。

4. 杰卡德相似度

杰卡德相似度的计算方式是用文本 *A* 与文本 *B* 相交的字数除以二者并集中的字数，如式（3-5）所示。

$$J(A,B)=\frac{|A\cap B|}{|A\cup B|} \tag{3-5}$$

杰卡德相似度的距离定义为

$$d_J(A,B)=1-\frac{|A\cap B|}{|A\cup B|} \tag{3-6}$$

例如，计算文本 *A*“乡村振兴”和文本 *B*“乡村发展旅游业”的杰卡德相似度，文本 *A* 和文本 *B* 的交集为{乡,村}，并集为{乡,村,振,兴,发,展,旅,游,业}，所以杰卡德相似度为

$$J(A,B)=\frac{2}{9}$$

杰卡德相似度与文本的位置、顺序均无关。只是在某些情况下，会先将文本分词，再对分词结果计算相似度。杰卡德相似度主要适用于对字或词的顺序不敏感的文本，不适用于重复字符较多的文本和对顺序很敏感的文本。

5. 余弦相似度

余弦相似度通过计算两个向量夹角的余弦值来衡量它们的相似度，即假设空间中有两个向量 $\boldsymbol{A}$ 和 $\boldsymbol{B}$，$\boldsymbol{A}=(a_1,a_2,\cdots,a_i,\cdots,a_n)$ 和 $\boldsymbol{B}=(b_1,b_2,\cdots,b_i,\cdots,b_n)$，则 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 的余弦相似度定义为

$$\cos\theta=\frac{\boldsymbol{A}\cdot\boldsymbol{B}}{|\boldsymbol{A}|\times|\boldsymbol{B}|}=\frac{\sum_{i=1}^{n}(a_i\times b_i)}{\sqrt{\sum_{i=1}^{n}(a_i)^2}\times\sqrt{\sum_{i=1}^{n}(b_i)^2}} \tag{3-7}$$

例如，计算文本 *A*“乡村振兴”和文本 *B*“乡村发展旅游业”的余弦相似度，具体计算过程如下。

（1）文本 *A* 和文本 *B* 的并集为{乡,村,振,兴,发,展,旅,游,业}，共 9 个字。

（2）根据并集中每个字在文本 A 和文本 B 中出现的次数可得 $a_1=1$，$a_2=1$，$a_3=1$，$a_4=1$，$a_5=0$，$a_6=0$，$a_7=0$，$a_8=0$，$a_9=0$；$b_1=1$，$b_2=1$，$b_3=0$，$b_4=0$，$b_5=1$，$b_6=1$，$b_7=1$，$b_8=1$，$b_9=1$，即 $\boldsymbol{A}=(1,1,1,1,0,0,0,0,0)$，$\boldsymbol{B}=(1,1,0,0,1,1,1,1,1)$。

（3）将 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 代入式（3-7）中，可得

$$\cos\theta=\frac{1+1+0+0+0+0+0+0+0}{\sqrt{1+1+1+1+0+0+0+0+0}\times\sqrt{1+1+0+0+1+1+1+1+1}}=\frac{1}{\sqrt{7}}$$

余弦相似度和杰卡德相似度虽然计算方式差异较大，但性质很类似，都与文本的并集高度相关，所以它们的适用场景也类似。余弦相似度与杰卡德相似度的不同之处在于余弦相似度考虑到了文本的频次。

6. 哈罗距离

哈罗距离的定义为

$$d=\frac{1}{3}\left(\frac{m}{|S_1|}+\frac{m}{|S_2|}+\frac{m-t}{m}\right) \tag{3-8}$$

其中，m 表示两个文本相互匹配的字符数量；$|S_1|$ 和 $|S_2|$ 表示两个文本的长度（字符数量）；t 表示换位数量。

例如，计算文本 A“乡村振兴”和文本 B“乡村发展旅游业”的哈罗距离，具体计算过程如下。

（1）文本 A 和文本 B 相互匹配的字符为{乡,村}，共两个字符，即 $m=2$；文本 A 和文本 B 的长度分别为 4 和 7，即 $|S_1|=4$，$|S_2|=7$；由于文本 A 和文本 B 相互匹配的字符位置一样，所以换位数量为 0，即 $t=0$。

（2）由此可得到文本 A 和文本 B 的哈罗距离为

$$d=\frac{1}{3}\left(\frac{2}{4}+\frac{2}{7}+1\right)=\frac{25}{42}$$

哈罗距离适用于对位置、顺序敏感的文本。

3.1.5 文本相似度算法实现

文本相似度的计算可以通过 scikit-learn 库实现。下面将介绍基于余弦相似度和欧氏距离两种算法计算文本相似度的具体操作过程。

余弦相似度可通过调用 scikit-learn 库的 cosine_similarity()函数进行计算，其语法格式如下。

```
cosine_similarity(X, Y=None, dense_output=True)
```

cosine_similarity()函数的常用参数及其说明如表 3-7 所示。

表 3-7　cosine_similarity()函数的常用参数及其说明

参数名称	说明
X	接收 str。表示输入文本。无默认值
Y	接收 str。表示输入文本。无默认值
dense_output	接收 bool。表示是否返回密集矩阵。默认为 True

欧氏距离可通过调用 scikit-learn 库的 euclidean_distances()函数进行计算，其语法格式如下。

```
euclidean_distances(X, Y=None, Y_norm_squared=None, squared=False)
```

euclidean_distances()函数的常用参数及其说明如表 3-8 所示。

表 3-8　euclidean_distances()函数的常用参数及其说明

参数名称	说明
X	接收 str。表示输入文本。无默认值
Y	接收 str。表示输入文本。无默认值
Y_norm_squared	接收 int。表示输入数据的平方和。无默认值
squared	接收 bool。表示是否返回欧氏距离的平方。默认为 False

现以文本“多样生态系统构建五彩生态画卷，神州大地处处‘生’机勃勃。”和“丰富生态系统描绘斑斓生态图景，神州大地处处呈现‘生’机盎然。”为例，计算两个文本的余弦相似度和欧氏距离，如代码 3-3 所示。

代码 3-3　余弦相似度和欧氏距离的计算

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity, euclidean_distances
# 文本输入
text1 = "多样生态系统构建五彩生态画卷，神州大地处处“生”机勃勃。"
text2 = "丰富生态系统描绘斑斓生态图景，神州大地处处呈现“生”机盎然。"
# 初始化
vectorizer = TfidfVectorizer()
# 文本向量化
vectorized_text1 = vectorizer.fit_transform([text1])
vectorized_text2 = vectorizer.transform([text2])
# 计算余弦相似度
cosine_similarity_score = cosine_similarity(vectorized_text1, vectorized_
text2)[0][0]
# 计算欧氏距离
euclidean_distance = euclidean_distances(vectorized_text1, vectorized_text2)
[0][0]
# 输出结果
print(f"余弦相似度：{cosine_similarity_score:.2f}")
print(f"欧氏距离：{euclidean_distance:.2f}")
```

代码 3-3 的运行结果如下。

```
余弦相似度： 0.00
欧氏距离： 1.00
```

运行结果展示了两个文本的余弦相似度为 0.00，欧氏距离为 1.00。

3.2 文本分析简介

文本分析又称文本挖掘，是一种从文本数据中提取有用信息的技术，通过自然语言处理、机器学习和数据挖掘等方法，对文本数据进行结构化和语义化的分析，以揭示其中的潜在模式、趋势和关系。文本分析可以帮助人们从大量的文本数据中获取有用的结构化信息，并为各种任务提供支持，如情感分析、文本分类、关键词提取、命名实体识别等。

3.2.1 结构化分析

结构化分析是对结构化数据进行深入理解和分析的过程，常见类型包括数据分类、数据聚类、关联规则挖掘和预测建模等。

文本向量的结构化分析是一种将无序和非结构化的自然语言文本转换为结构化表示形式的技术。通过将文本数据转换为向量表示，可以将传统的文本处理任务，如分类、聚类和匹配等转换为基于向量空间的数学运算，从而更容易地进行计算和分析。常见的文本结构化分析类型有分词、词性标注、命名实体识别、句法分析和语篇分析。

1. 分词

分词是将连续的文本划分成单个且有意义的词语单元，主要用于处理中文文本。分词的内容主要包括以下四方面。

（1）单词划分：将连续的文本划分为单个的词语单元。

（2）停用词去除：排除常见的无意义词语，如介词、连词等，提高分词结果的准确性和质量。

（3）词语归一化：对于同义词以及词态和词形的变化进行归一化处理，提高文本的一致性。

（4）分词结果标注：给每个词语单元标注其在原文本中的位置。

分词的作用是为后续的文本处理任务提供基本的词语单元，如词频统计、关键词提取和情感分析等。

2. 词性标注

词性标注是为每个分词后的词语单元标注其词性。词性标注的内容主要包括以下三方面。

（1）词性定义：确定不同的词性类别及其含义，如名词、动词、形容词等。

（2）词性归类：将分词后的词语单元分配到不同的词性类别中并进行标注，如对动词进行动词词性标注。

（3）词性标注模型建立：构建机器学习模型或采用基于规则的方法，对分词后的词语单元进行自动标注。

词性标注的作用是对每个词语单元进行语法和语义上的分类，为后续的句法分析和语义分析提供基础。

3. 命名实体识别

命名实体识别是自然语言处理序列标注任务的一种，是指从输入文本中识别出有特定意义或指代性强的实体，是机器翻译、知识图谱、关系抽取、问答系统等的基础。命名实体从学术上通常分为三大类和七小类，三大类指实体类、时间类、数字类，七小类指人名、地名、组织机构名、时间、日期、货币、百分比。语言具有语法，语料遵循一定的语法结构，所以条件随机场、隐马尔可夫模型和最大熵马尔可夫模型等概率图模型被用来分析标签转移概率，深度学习模型一般会加上条件随机场层来进行句子级别的标签预测。

4. 句法分析

句法分析也是自然语言处理中的基础性工作，它分析句子的句法结构（主谓宾结构）和词与词之间的依存关系（并列、从属等）。通过句法分析，可以为语义分析、情感倾向、观点抽取等自然语言处理应用场景打下坚实的基础。

通过句法结构分析，能够识别句子中的主、谓、宾、定、状、补等成分，并分析各成分之间的关系。对于复杂语句，仅通过词性分析，不能得到正确的语句成分关系。

5. 语篇分析

语篇分析是对文本的整体结构和关系进行分析，主要关注句子之间的逻辑关系和上下文的语义。语篇分析的内容主要包括以下三方面。

（1）句子边界检测：确定文本中的句子起始和结束的位置。

（2）句子关联关系分析：分析句子之间的因果关系、逻辑关系、转折关系等。

（3）上下文语义理解：通过推断和关联，理解句子中的指代关系、补全信息等。

语篇分析的作用是对文本的结构和语义进行整体分析，帮助理解文本的上下文信息、隐含语义和逻辑关系。

3.2.2 语义化分析

语义化分析是对文本进行语义层面的分析，关注文本的意义和语义关联。语义化分析的主要作用是更深层次地理解文本的含义和语义关系，有助于理解文本的意图和情感。它可以用于问答系统、文本分类、情感分析和知识图谱构建等任务。下面将介绍语义化分析的五个常见类型。

1. 情感分析

情感分析又称观点挖掘，该任务的目的是从文本中研究人们对实体及其属性所表达的观点、情绪、情感、评价和态度。这些实体可以是各种产品、机构、服务、个人、事件、问题或主题等。情感分析包括很多研究任务，如观点信息抽取、情感挖掘、主观性分析、倾向性分析、情绪分析以及评论挖掘等。基于所处理文本的颗粒度，情感分析可以分成三个级别：篇章级、句子级和属性级。研究情感分析的方法有很多种，如情感词典匹配规则、传统机器学习、深度学习等。

2. 文本分类

文本分类与情感分析是自然语言处理中两个相关但不完全相同的任务。文本分类侧重于将文本数据整理成不同的类别，情感分析则更侧重于分析文本中的情感或情绪倾向。

文本分类的应用包括辨别垃圾信息或恶意评论、对文章进行政治倾向分类等。文本分类的方法有很多种，如传统机器学习的逻辑回归、支持向量机、贝叶斯分类模型、主题模型，再如深度学习的 FastText、基于 CNN/RNN 的分类模型，以及基于预训练模型的 BERT、ELMo、GPT、ULMFiT 等。

3. 文本聚类

文本聚类是将相似的文本分组到同一类别或群组。在语义化分析中，文本聚类的内容主要包括以下三方面。

（1）特征提取：从文本中提取出表达文本含义的特征，如词向量、词频、TF-IDF 值等。

（2）相似度度量：计算文本之间的相似度，常见的度量方法有余弦相似度、欧式距离等。

（3）聚类算法：将文本根据相似度进行分组，常用的聚类算法有 *K*-means 算法、层次聚类算法等。

文本聚类的作用是将相似的文本聚合到一起，便于后续信息检索、文本分类和信息抽取等任务的完成。

4. 问答系统

问答系统是能够针对用户提出的问题给出准确答案或相关信息的系统。在语义化分析中，问答系统的内容主要包括以下三方面。

（1）问题理解：对用户的问题进行自然语言理解，并转化为机器可识别的形式。

（2）信息检索：根据问题中的关键词或句子，从数据库或网络中检索相关信息。

（3）答案生成：根据问题和检索到的信息，生成准确和合理的答案。

问答系统的目标是实现机器对自然语言问题的理解和准确回答，可以应用于智能助手、知识库问答和在线客服等场景。

5. 主题分析

主题分析是一种文本分析技术，旨在从文本数据中提取出隐藏在其中的主题信息。主题分析的目标是将文本聚类成一组具有相关主题的文档集合，从而揭示文本的主题结构和潜在语义。主题分析的内容主要包括以下四方面。

（1）主题模型：主题模型是一种数学模型，能够从大量文本中推断出一组主题，并确定每个文档中的主题分布。最常用的主题模型是潜在狄利克雷分配（Latent Dirichlet Allocation，LDA）模型，它在文本分析领域被广泛应用。

（2）文本预处理：为了能够有效地进行主题分析，需要对文本进行预处理，如分词、去除停用词、词干化或词形还原等。这些预处理步骤有助于提取出文本中有意义的信息，减少噪声对主题分析的干扰。

（3）特征提取：主题分析通常利用词频、TF-IDF 值或词嵌入等提取文本的特征向量。这些特征向量将文本转换为数值表示，便于后续的聚类或分类分析。

（4）聚类分析：基于提取的特征向量，可以使用聚类算法（如 *K*-means 算法、层次聚类算法等）将文本进行聚类，将具有相似主题的文档放在一组。聚类的结果可以帮助理解文本的整体结构并发现潜在的主题。

3.3 文本分析常用算法

在文本分析中，算法的选择是一个尤为关键的影响因素。文本分析的常用算法主要分为两大类：机器学习算法和深度学习算法。这两类算法在文本分析中各有其独特的作用和优势。

3.3.1 常用机器学习算法

机器学习算法需要手工提取特征或构建特征工程来表示文本，如 BOW 模型、TF-IDF

算法等，这些特征表示在机器学习过程中起到了关键作用。机器学习算法在自然语言处理中通常通过监督学习来进行训练和预测，需要大量标记好的数据。文本分析中常用的机器学习算法包括朴素贝叶斯、支持向量机、决策树、随机森林（Random Forest）和 *K*-means 算法等。

1. 朴素贝叶斯

朴素贝叶斯是一种基于贝叶斯定理的分类算法，假设所有特征之间是相互独立的。朴素贝叶斯是分类器中最常用的一种生成式模型，基于贝叶斯定理将联合概率转换为条件概率，利用特征条件及独立假设简化条件的概率进行计算。贝叶斯算法核心为

$$P(A \mid B)=\frac{P(B \mid A)P(A)}{P(B)} \tag{3-9}$$

其中，$P(A)$ 为事件 A 发生的概率；$P(B)$ 为事件 B 发生的概率；$P(A \mid B)$ 为事件 B 发生的条件下 A 发生的概率；$P(B \mid A)$ 为事件 A 发生的条件下 B 发生的概率。

朴素贝叶斯的流程图如图 3-5 所示，主要包括以下步骤。

（1）计算先验概率：计算训练集中每个类别出现的次数，并计算每个类别的概率。

（2）计算条件概率：计算训练集中每个类别下第 i 维特征的第 i 个取值出现的次数，然后计算每个条件概率。

（3）计算特征概率：将待分类项的特征值代入条件概率公式，计算出待分类项在每个类别下的特征概率。

（4）选择特征概率最大的类别作为待分类项的类别。

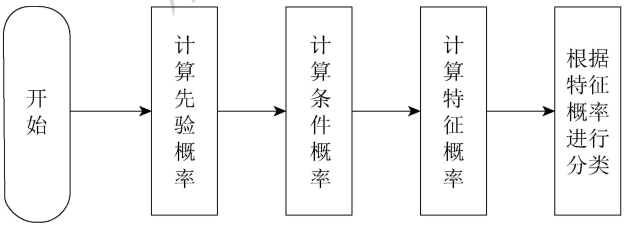

图 3-5　朴素贝叶斯的流程图

2. 支持向量机

支持向量机（Support Vector Machine，SVM）是一种基于统计学习理论的分类和回归算法。它通过在样本空间中构造一个最优划分超平面来实现分类，最大化分类边界与最近训练样本的距离。

对于给定数据集 $D=\{(\boldsymbol{x}_1,y_1),(\boldsymbol{x}_2,y_2),\cdots,(\boldsymbol{x}_n,y_n)\}$，$y_i \in [-1,+1]$，支持向量机的思想是在样本空间中找到一个最优划分超平面，将不同类别的样本分开。能将数据集分开的划分超平面可能有很多，如图 3-6 所示，可以直观地看出应该选择位于两类样本“正中间”的划分超平

面，即图 3-6 中加粗的划分超平面，因为该划分超平面对训练样本的鲁棒性是最强的。

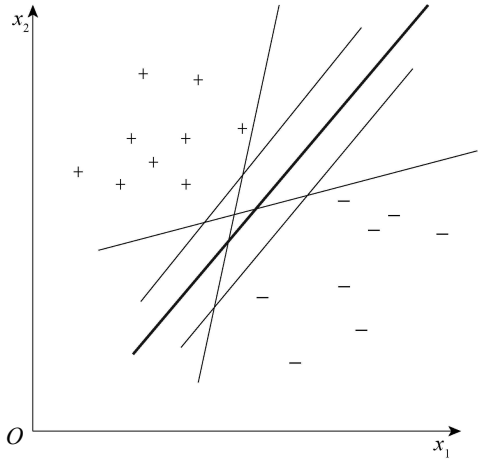

图 3-6　支持向量机超平面划分

在样本空间中，划分超平面可通过线性方程来描述，如式（3-10）所示。

$$\boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{x}+b=0 \tag{3-10}$$

其中，$\boldsymbol{\omega}=(\omega_1;\omega_2;\cdots;\omega_m)$ 为法向量，决定了划分超平面的方向；b 为位移项，决定了划分超平面与原点之间的距离。

支持向量机的流程图如图 3-7 所示，主要包括以下步骤。

（1）选择与目标变量相关性高的特征，并对特征进行归一化或标准化等处理。

（2）使用合适的核函数将数据映射到高维空间中。常用的核函数包括线性核、多项式核和径向基函数核等。

（3）在高维空间中寻找一个最优划分超平面，使得各类数据点到该划分超平面的距离最大。

（4）使用学习到的模型对新的样本进行分类。

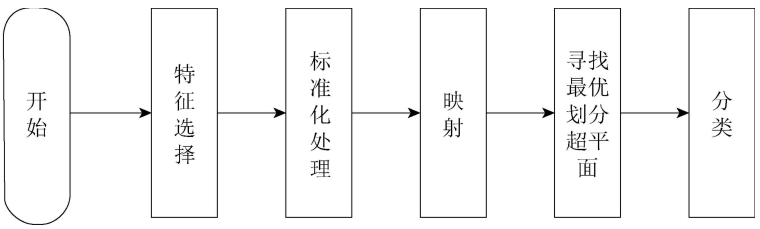

图 3-7　支持向量机的流程图

3. 决策树

决策树是一种基于条件判断的分类算法。通过建立树状结构，将数据集划分为不同的子

集，选择最优划分特征，直到达到预定的终止条件。

决策树是数据反复分组的图形化体现，也是推理规则的图形化展示。决策树中的每个节点都对应一条推理规则，决策树基于叶节点的推理规则实现对新数据的分类预测。对于观测样本 X_0，只需从根节点开始依次根据输入变量取值，沿着决策树的不同分支遍历，直到到达样本量等于 n 的叶节点。其计算公式为

$$\hat{y}_0 = \text{argmax}(n_{y1}, n_{y2}, \cdots, n_{yk}) \tag{3-11}$$

其中，$\hat{y}_0$ 表示观测样本 X_0 的类别值；n_{yk} 表示观测样本 X_0 在每一个叶节点的类别值。

决策树的构造示意图如图 3-8 所示，包括以下步骤。

（1）选择一个特征作为根节点，根据该特征将数据集划分为不同的子集。

（2）对于每个子集，重复步骤（1），选择最优的特征来划分子集，直到满足某个终止条件（如子集中的样本属于同一类别或达到树的最大深度）。

（3）重复步骤（1）和步骤（2）划分子集的过程，最终形成完整的决策树。

（4）使用决策树对新样本进行分类，从根节点开始根据判断条件逐步向下遍历，直到到达叶节点，最终将样本分类到对应的类别。

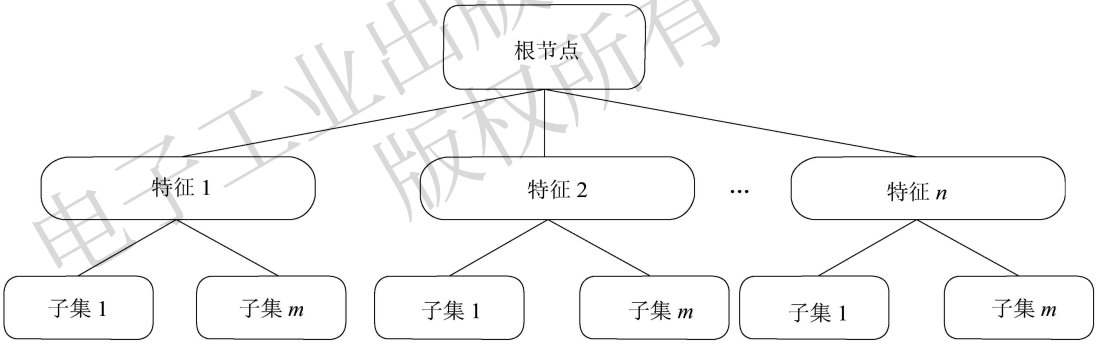

图 3-8　决策树的构造示意图

4. 随机森林

随机森林是一种组合预测模型，随机建立一片森林，森林中包含众多有较高预测精度但弱相关甚至不相关的决策树，并形成组合预测模型，共同参与对新观测输出变量取值的预测。

随机森林的随机性表现在两个方面：第一，训练样本是从原始样本中重复随机选择的，即训练样本具有随机性；第二，在每棵决策树的建立过程中，每个节点的分裂属性集合是随机选择确定的。

随机森林构建变量子集的常见方式有如下两种。

（1）随机选择输入变量。随机选择 k 个输入变量进入候选变量子集，依据变量子集建立一棵充分生长的决策树，无须剪枝以减少预测偏差。

（2）随机组合输入变量。随机选择 L 个输入变量 x 并生成 L 个服从均匀分布的随机数 a，其线性组合如式（3-12）所示。

$$v_j = \sum_{i=1}^{L} a_i x_i \ ,\ \ a_i \in [-1,1] \tag{3-12}$$

重复得到k个由新变量v组成的输入变量子集，依据变量子集建立一棵充分生长的决策树，且无须剪枝。

假设训练集 T 的大小为 N，特征数目为 M，随机森林的大小为 K，随机森林的流程图如图 3-9 所示，主要包括以下步骤。

（1）从训练集 T 中以有放回抽样的方式取样 N 次，形成一个新的训练集 D。

（2）随机选择 m 个特征，其中 $m < M$ 。

（3）使用新的训练集 D 和 m 个特征，建立一个完整的决策树。

（4）得到随机森林。

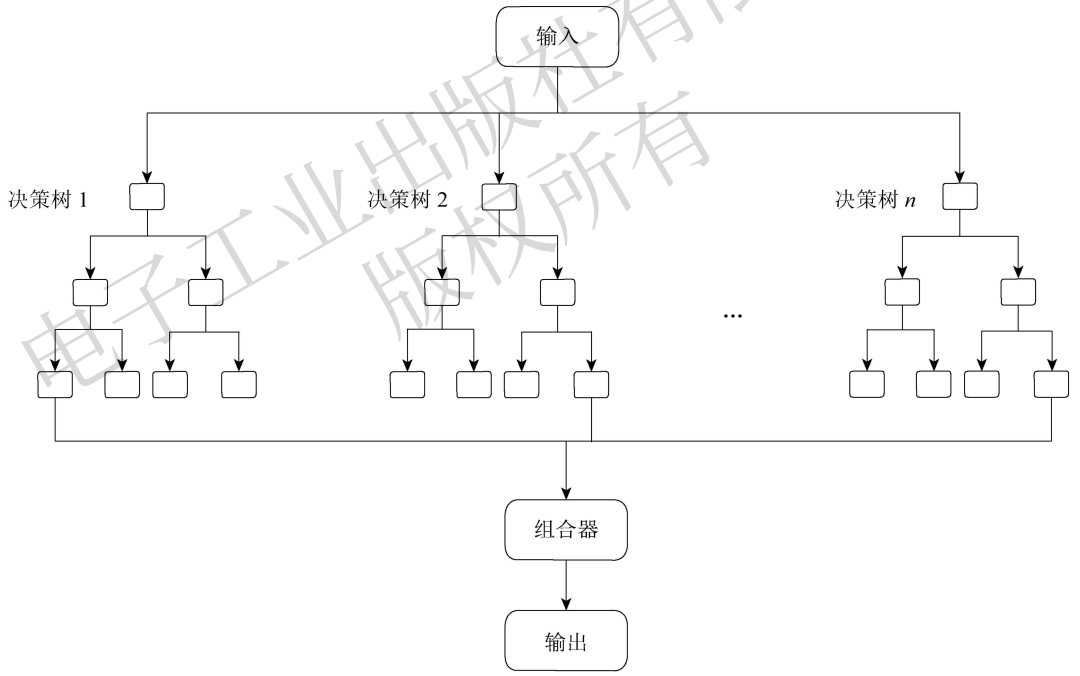

图 3-9　随机森林的流程图

5. *K*-means 算法

K-means 算法是一种无监督的聚类算法，其目的是将 n 个数据点分为 K 个类。每个类都有一个质心，这些质心最小化了其内部数据点与质心之间的距离。*K*-means 算法的基本思想是通过迭代寻找 K 个类的划分方案，使得聚类结果对应的代价函数最小。

K-means 算法的流程图如图 3-10 所示，主要包括以下步骤。

（1）指定聚类数目 K。

（2）确定 K 个小类的初始质心。

（3）根据最近原则进行聚类。依次计算每个样本观测点到 K 个小类质心的距离，并按照距 K 个小类质心最近的原则，将所有样本观测点分配到距离最近的小类中，形成 K 个小类。

（4）重新计算 K 个小类的质心。

（5）判断是否满足终止聚类算法的条件（迭代次数或小类质心偏移程度）。如果没有满足则重复步骤（3）和步骤（4）直至满足迭代终止条件。

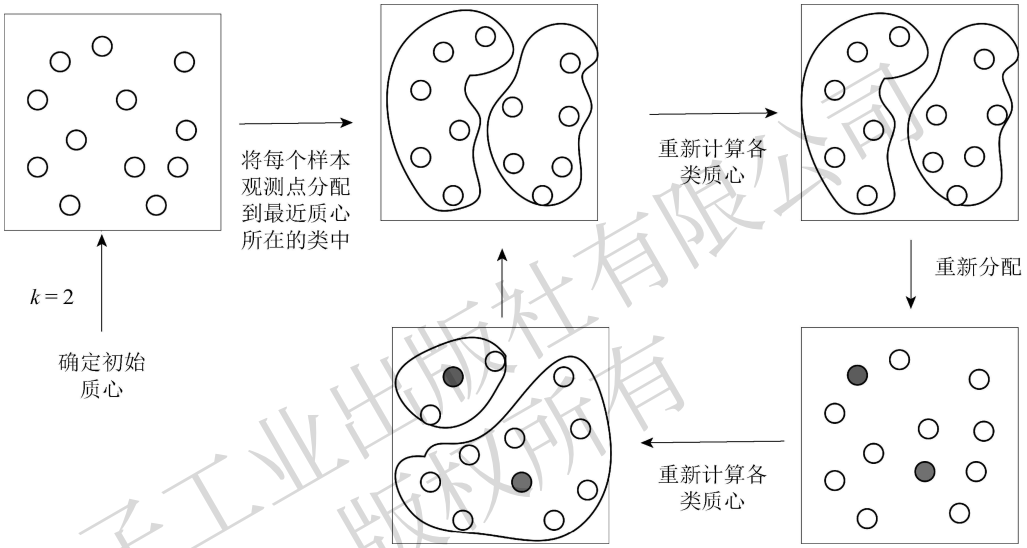

图 3-10　*K*-means 算法的流程图

3.3.2　常用深度学习算法

深度学习算法在自然语言处理中取得了重大突破，这些算法通过构建深层神经网络模型，可以自动学习文本数据中的复杂特征和语义表示，提高了自然语言处理任务的性能。

深度学习算法可以直接从原始文本数据中学习，不需要手工提取特征，这极大地简化了自然语言处理流程。深度学习尤其适用于序列建模任务，如文本分类、命名实体识别、机器翻译、文本生成等。常用的深度学习算法包括卷积神经网络、循环神经网络、注意力机制和 Transformer 等。

1. 卷积神经网络

卷积神经网络（Convolutional Neural Networks，CNN）是包含卷积计算且具有深度结构的前馈神经网络，是深度学习的代表算法之一。

卷积神经网络是一种多层的监督学习神经网络，隐藏层的卷积层和池化层是实现卷积神

经网络特征提取功能的核心模块。该网络模型通过梯度下降法最小化损失函数，从而对网络中的权重参数逐层反向调节，通过频繁的迭代训练提高网络的精度。卷积神经网络主要包括输入层、卷积层、池化层、全连接层、输出层等模块，最后一层的输出层是一个分类器，可以采用逻辑回归、Softmax 回归或支持向量机对输入进行分类。经典的卷积神经网络 LeNet-5 的结构图如图 3-11 所示。

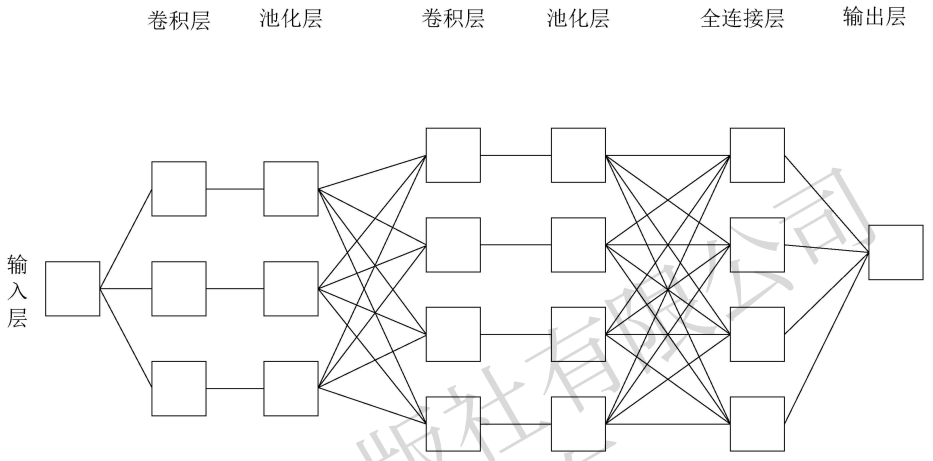

图 3-11　卷积神经网络 LeNet-5 的结构图

卷积神经网络适用于文本分类和信息抽取等任务，其具体算法及其描述如表 3-9 所示。

表 3-9　卷积神经网络的具体算法及其描述

算法	描述
TextCNN	使用卷积层进行文本分类的卷积神经网络模型，可用于对文本进行分类、情感分析等任务
DCNN	扩张卷积（或空洞卷积）的卷积神经网络模型，可用于对文本进行分类、情感分析、文本摘要、机器翻译等任务
VDCNN	深度卷积神经网络模型，可用于对文本进行精细分类
GCNN	应用于图数据分析的卷积神经网络模型，可用于处理非结构化数据和关系型数据，如社交网络、推荐系统等
TCN	应用于时间序列数据的卷积神经网络模型，可用于序列标注、语音识别等任务

2. 循环神经网络

循环神经网络（Recurrent Neural Network，RNN）是一类具有反馈连接的神经网络，特别适合处理序列数据，如自然语言文本。与传统的前馈神经网络不同，循环神经网络在处理序列数据时可以捕捉到数据之间的时序信息。

在循环神经网络中，每个时间步可以看作网络的一个循环单元，它接收输入和前一个时间步的隐藏状态,并输出当前时间步的隐藏状态。这种循环结构使得网络在处理长序列数据时，能够记忆之前的信息并在后续时间步中进行参考。循环神经网络的结构图如图 3-12 所示。

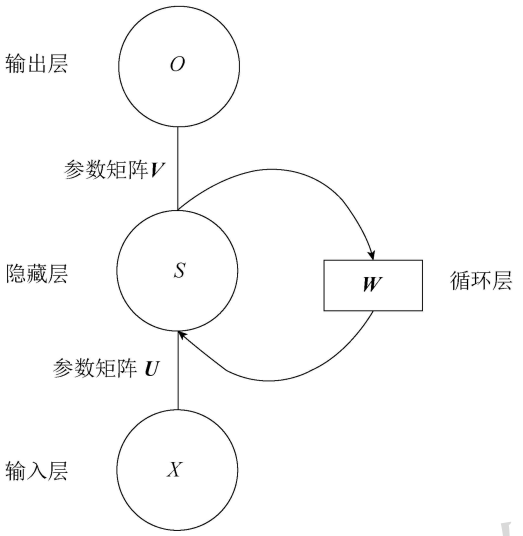

图 3-12　循环神经网络的结构图

按照时间线将图 3-12 的循环层展开可得如图 3-13 所示的结构图。

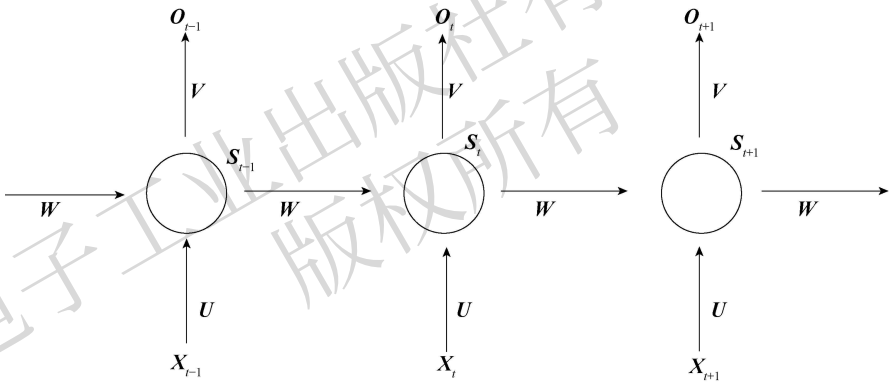

图 3-13　循环层展开结构图

循环神经网络适用于自然语言处理的文本生成、机器翻译和情感分析等任务，其具体算法及其描述如表 3-10 所示。

表 3-10　循环神经网络的具体算法及其描述

算法	描述
Simple RNN	简单循环神经网络，可用于词性标注、情感分析等
LSTM	长短期记忆网络，可用于命名实体识别、情感分析、机器翻译等
GRU	门控循环单元，可用于词性标注、情感分析、自动摘要等
BRNN	双向循环神经网络，可用于命名实体识别、句法分析等

3. 注意力机制

注意力机制（Attention Mechanism）的提出影响了基于深度学习算法的许多人工智能应

用的发展，如自然语言处理、图像检测、语音识别等。

注意力机制是一种让模型重点关注关键信息并提取其特征用于学习分析的策略。最早将注意力机制引进自然语言处理领域的是机器翻译等基于 Encoder-Decoder 框架的场景。Encoder-Decoder 框架的主要作用是将可变长度的输入序列编码成一个固定长度的向量，然后将固定长度的向量解码成一个可变长度的输出序列。Encoder-Decoder 框架的结构图如图 3-14 所示。

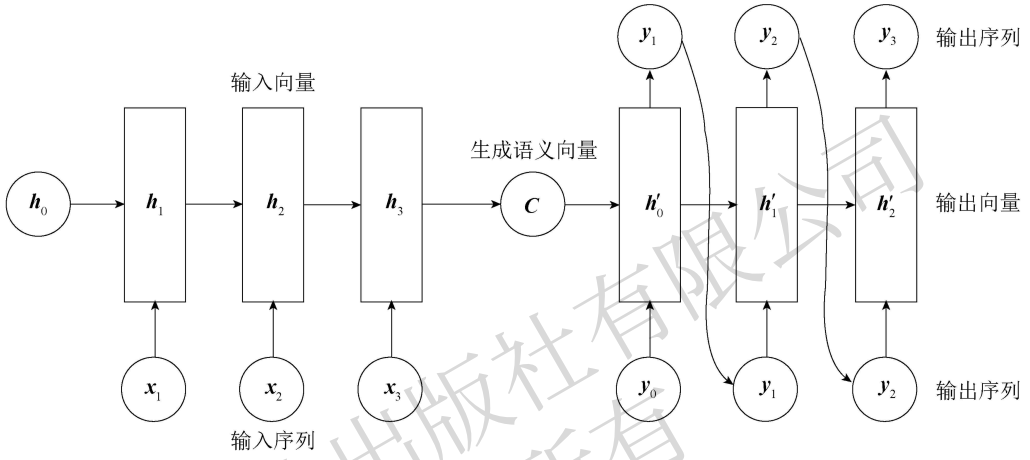

图 3-14　Encoder-Decoder 框架的结构图

传统的 Encoder-Decoder 框架存在一个问题，即一些与当前任务无关的信息都会被编码器强制编码进去，尤其当输入序列很长或信息量很大时，进行选择性编码不是 Encoder-Decoder 框架所能做到的，然而注意力机制刚好可以解决这一问题。注意力机制最早是为了解决 Seq2Seq 问题的，后来研究者尝试将其应用到情感分析、句对关系判别等其他任务场景，如关注 aspect 的情感分析模型 ATAE LSTM、分析句对关系的 ABCNN 等。

注意力机制是一种信息处理方法，其核心思想是，在处理信息时，先关注重要的信息，忽略其他不太相关的信息。这种方法能够以高权重聚焦重要信息，以低权重忽略不相关的信息，并且可以不断调整权重，使得在不同的情况下也可以选取重要的信息。注意力机制通过允许解码器访问所有编码器产生的输出来克服传统框架的缺点。其原理是，对编码器的所有输出进行加权组合，然后输入到当前位置的解码器中，影响解码器的输出。通过对编码器的输出进行加权，在实现输入与输出对齐的同时还能够利用更多原始数据的上下文信息。注意力机制的网络结构图如图 3-15 所示。

注意力机制的算法可以分为硬注意力（Hard Attention）和软注意力（Soft Attention）。硬注意力在计算过程中明确需要关注哪些元素，软注意力则对所有元素都有一定的关注度，只是关注度不同。

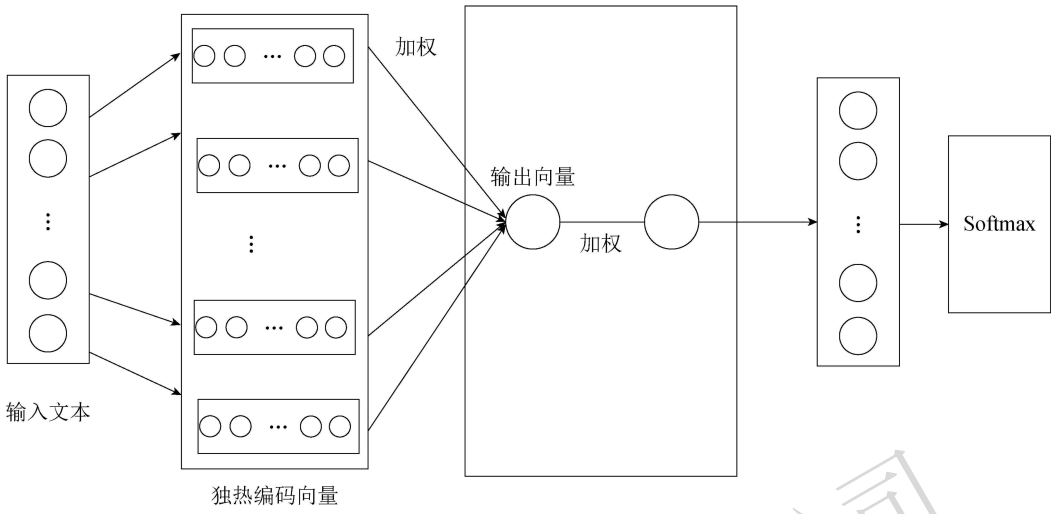

图 3-15　注意力机制的网络结构图

注意力机制的作用主要体现在以下三方面。

（1）提高预测精度：在处理复杂数据时，通过聚焦于重要的信息，可以减少噪声和干扰，从而提高预测的精度。

（2）简化计算：通过忽略一些不重要的信息，可以减少计算量，提高计算效率。

（3）提取关键信息：在处理大量数据时，注意力机制可以帮助快速找到关键信息，从而更好地理解和分析问题。

4. Transformer

Transformer 是一种基于注意力机制的深度学习算法，用于序列数据的处理和生成。Transformer 最初应用于机器翻译领域，由 Google 发布。Transformer 的核心思想是使用多头注意力机制代替传统的循环结构（RNN、LSTM、GRU）来编码序列信息。这种机制允许序列中的每个位置都可以在解码器端上下文有选择地关注一组编码器的位置，而不是像循环结构一样每次仅关注前一个位置。

Transformer 的整体架构分为输入模块、编码器模块、解码器模块和输出模块。输入模块包括源文本嵌入层及其位置编码器、目标文本嵌入层及其位置编码器；编码器模块由 N 个编码器层堆叠而成，每个编码器层由两个子层连接结构组成，第一个子层连接结构包括一个多头自注意力子层、一个规范化层和一个残差连接层，第二个子层连接结构包括一个前馈全连接子层、一个规范化层和一个残差连接层；解码器模块由 N 个解码器层堆叠而成，每个解码器层由三个子层连接结构组成，第一个子层连接结构包括一个多头自注意力子层、一个规范化层和一个残差连接层，第二个子层连接结构包括一个多头注意力子层、一个规范化层和一个残差连接层，第三个子层连接结构包括一个前馈全连接子层、一个规范化层和一个残差连

接层；输出模块包括线性层和 softmax 激活层。

Transformer 的整体架构图如图 3-16 所示。

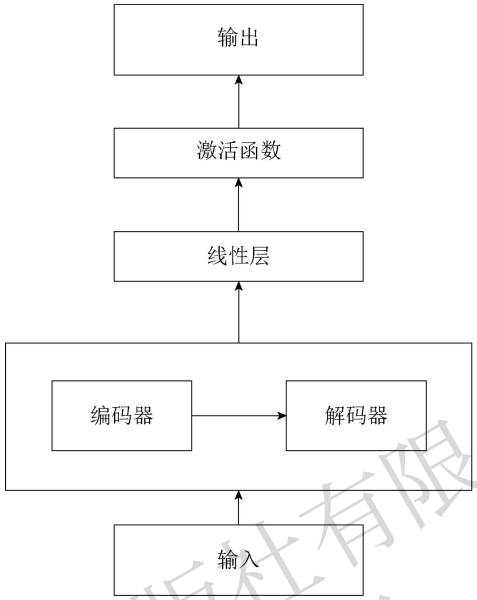

图 3-16　Transformer 的整体架构图

解码器层和编码器层的结构图如图 3-17 所示。

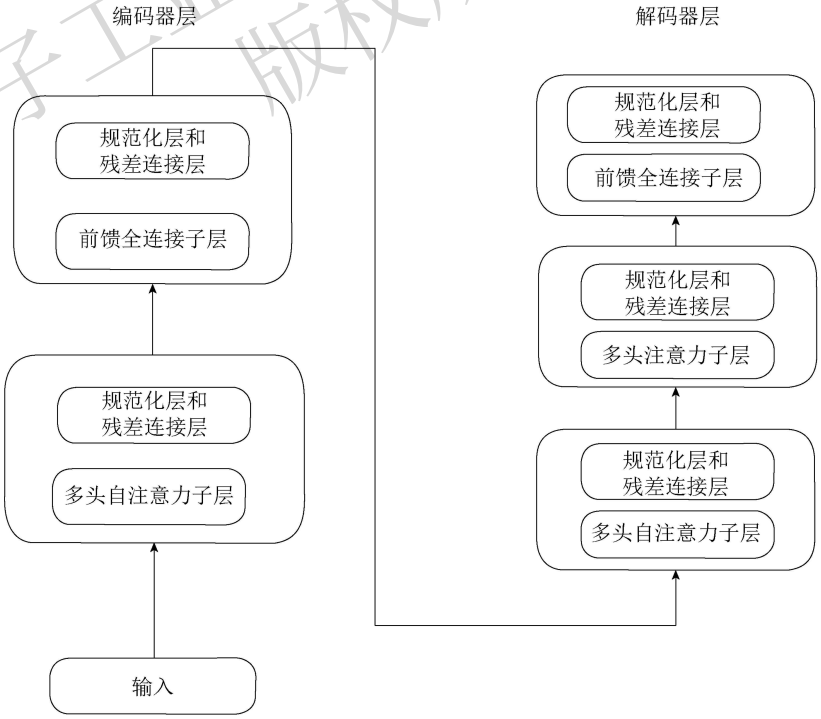

图 3-17　编码器层和解码器层的结构图

Transformer 的具体算法及其描述如表 3-11 所示。

表 3-11　Transformer 的具体算法及其描述

算法	描述
Transformer	Transformer 原始版本，可用于机器翻译
GPT	基于 Transformer 的预训练语言模型，可用于自然语言生成任务，如文本生成、问答、机器翻译等
BERT	一种预训练语言模型，可用于自然语言理解任务，如语言模型、文本分类、问答系统、文本摘要等
XLNet	BERT 和 GPT 的结合体，可用于语言模型、文本分类、问答系统、文本摘要等

本章小结

本章首先介绍了文本向量化与相似度的基本概念，以及常用文本向量化方法和文本相似度算法，并演示了文本向量化和文本相似度计算的实现。其中，文本向量化的方法介绍了文本离散化表示和文本分布式表示；文本相似度算法介绍了欧氏距离、曼哈顿距离、编辑距离、杰卡德相似度、余弦相似度和哈罗距离。接着从结构化和语义化两方面分别介绍了文本分析的方法。最后从机器学习和深度学习两方面介绍了文本分析的常用算法。其中，机器学习算法介绍了朴素贝叶斯、支持向量机、决策树、随机森林和 *K*-means 算法，深度学习算法介绍了卷积神经网络、循环神经网络、注意力机制和 Transformer。

课后习题

1. 选择题

（1）下列不是文本离散化表示的是（　　）。

A. 独热表示　　B. BOW 模型

C. DM 模型　　D. TF-IDF 表示

（2）下列模型中属于 Word2Vec 模型的是（　　）。

A. CBOW 模型　　B. Doc2Vec 模型

C. DM 模型　　D. Skip 模型

（3）以下算法中不是常用的文本相似度算法的是（　　）。

A. 曼哈顿距离　　B. 空间距离

C. 编辑距离　　D. 杰卡德距离

（4）下列不是结构化分析的是（　　）。

A. 主题分析　　B. 分词

C. 词性标注　　D. 句法分析

（5）下列不是语义化分析的是（　　）。

A. 情感分析　　B. 语篇分析

C. 文本聚类　　D. 问答系统

（6）下列机器学习算法中用于文本聚类的是（　　）。

A. 朴素贝叶斯　　B. 决策树

C. 支持向量机　　D. *K*-means 算法

（7）以下关于 Word2Vec 模型的描述，错误的是（　　）。

A. CBOW 模型根据上下文可预测目标词的概率分布

B. Skip-Gram 模型根据目标词预测其上下文的概率分布

C. Word2Vec 模型的输入是独热向量

D. Word2Vec 模型只适用于处理小规模语料库

（8）以下关于 Doc2Vec 模型的描述，错误的是（　　）。

A. Doc2Vec 模型是在 Word2Vec 模型的基础上扩展而来的

B. Doc2Vec 模型可以提取文本的语义信息和语序信息

C. Doc2Vec 模型中，DM 模型与 CBOW 模型类似

D. Doc2Vec 模型中，DBOW 模型只能预测段落中随机词的概率分布

（9）以下关于文本分析算法的描述，错误的是（　　）。

A. 机器学习算法需要手工提取特征或构建特征工程来表示文本

B. 深度学习算法可以直接从原始文本数据中学习，不需要手工提取特征

C. 机器学习算法在 NLP 中通常通过监督学习来进行训练和预测

D. 深度学习算法在 NLP 中通常通过无监督学习来进行训练和预测

2. 操作题

（1）请通过 gensim 库分别使用 Word2Vec 模型和 Doc2Vec 模型对“建设农业强国，利器在科技”和“农业强国，科技助力”两个文本进行向量化，给出最终的向量化结果和实现代码。

（2）请使用 euclidean_distances()函数对“建设农业强国，利器在科技”和“农业强国，科技助力”两个文本进行欧氏距离的计算，给出最终的欧氏距离计算结果和实现代码。