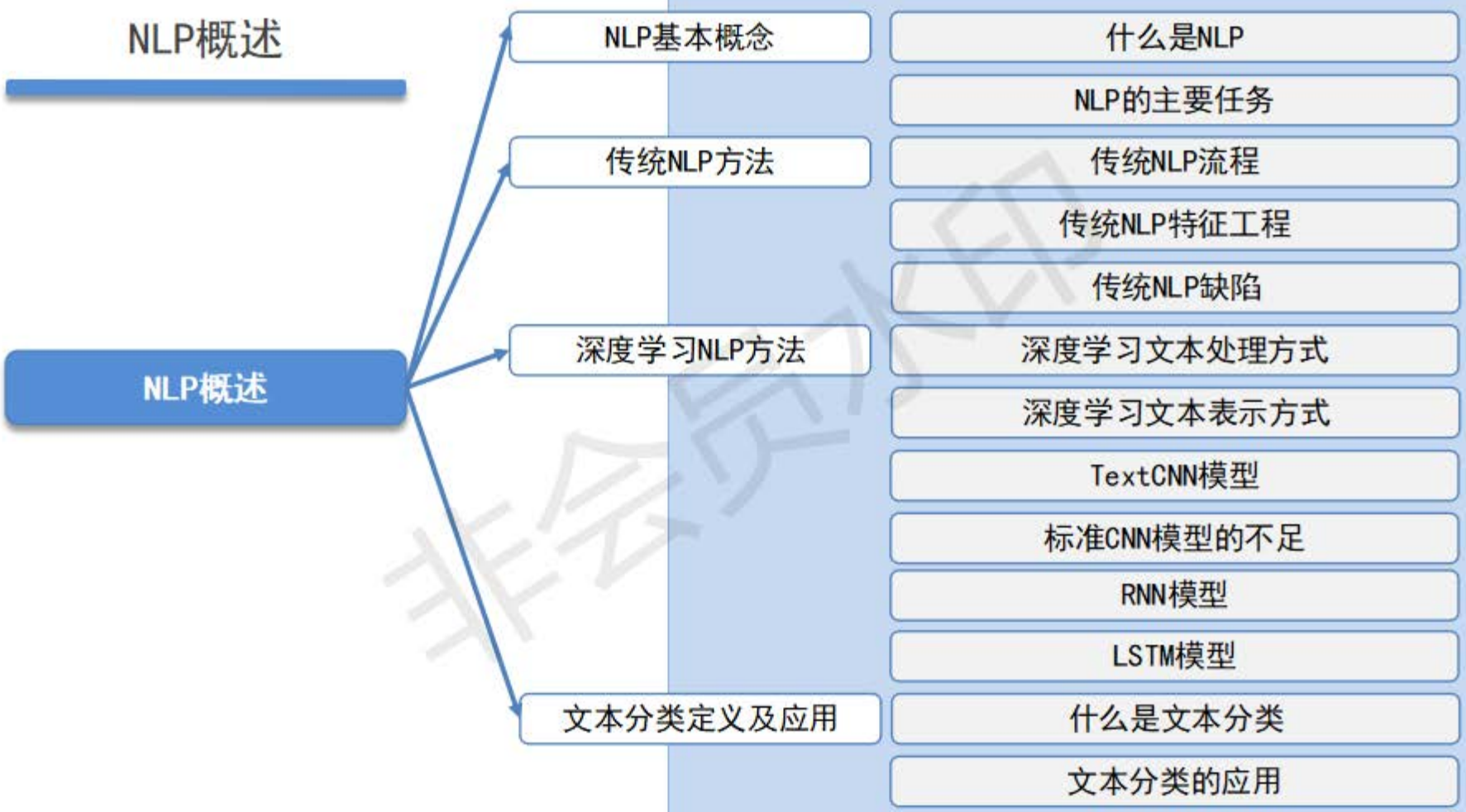
深度学习

PaddlePaddle文本分类 DAY06

NLP概述


NLP概述
NLP基本概念
什么是NLP
NLP的主要任务
传统NLP方法
传统NLP流程
传统NLP特征工程
传统NLP缺陷
深度学习NLP方法
深度学习文本处理方式
深度学习文本表示方式
TextCNN模型
标准CNN模型的不足
RNN模型
LSTM模型
文本分类定义及应用
什么是文本分类
文本分类的应用

NLP基本概念

什么是NLP

知识讲解

- NLP是Natural Language Processing（自然语言处理）简写，NLP常见定义有：
  - ✓ 定义一：自然语言处理是计算机科学与语言中关于计算机与人类语言转换的领域。——中文维基百科
  - ✓ 定义二：自然语言处理是人工智能领域中一个重要的方向。它研究实现人与计算机之间用自然语言进行有效沟通的各种理论和方法。——百度百科
  - ✓ 定义三：研究在人与人交际中及人与计算机交际中的语言问题的一门学科。NLP要研制表示语言能力和语言应用的模型，建立计算机框架来实现这些语言模型，提出响应的方法来不断完善这种模型，并根据语言模型设计各种实用系统，以及对这些系统的评测技术。——Bill Manaris，《从人机交互的角度看自然语言处理》

什么是NLP（续）

- NLP还有其它一些名称：
  - ✓ - 自然语言理解（Natural Language Understanding）
  - ✓ - 计算机语言学（Computational Linguistics）
  - ✓ - 人类语言技术（Human Language Technology）

知识讲解

NLP的主要任务

知识讲解

- **分词**：该任务将文本语料库分隔成原子单元（例如，单词）。虽然看似微不足道，但是分词是一项重要任务。例如，在日语中，词语不以空格或标点符号分隔
- **词义消歧**：词义消歧是识别单词正确含义的任务。例如，在句子“The dog **barked** at the mailman”（狗对邮递员**吠叫**）和“Tree **bark** is sometimes used as a medicine”（**树皮**有时用作药物）中，单词bark有两种不同的含义。词义消歧对于诸如问答之类的任务至关重要

NLP的主要任务（续1）

知识讲解

- **命名实体识别（NER）：**NER尝试从给定的文本主体或文本语料库中提取实体（例如，人物、位置和组织）。例如，句子：

  John gave Mary two apples at school on Monday

  将转换为：

  $[\text{John}]_{\text{name}}$ gave $[\text{Mary}]_{\text{name}}$ $[\text{two}]_{\text{number}}$ apples at $[\text{school}]_{\text{organization}}$ on $[\text{Monday.}]_{\text{time}}$

  NER在诸如信息检索和知识表示等领域不可或缺

- **词性（PoS）标记：**PoS标记是将单词分配到各自对应词性的任务。它既可以是名词、动词、形容词、副词、介词等基本词、也可以是专有名词、普通名词、短语动词、动词等

NLP的主要任务（续2）

知识讲解

- **句子/概要分类：**句子或概要（例如，电影评论）分类有许多应用场景，例如垃圾邮件检测、新闻文章分类（例如，政治、科技和运动）和产品评论评级（即正向或负向）。我们可以用标记数据（即人工对评论标上正面或负面的标签）训练一个分类模型来实现这项任务
- **语言生成：**在语言生成中，我们使用文本语料库（包含大量文本文档）来训练学习模型（例如，神经网络），以预测后面的新文本。例如，可以通过使用现有的科幻故事训练语言生成模型，来输出一个全新的科幻故事

NLP的主要任务（续3）

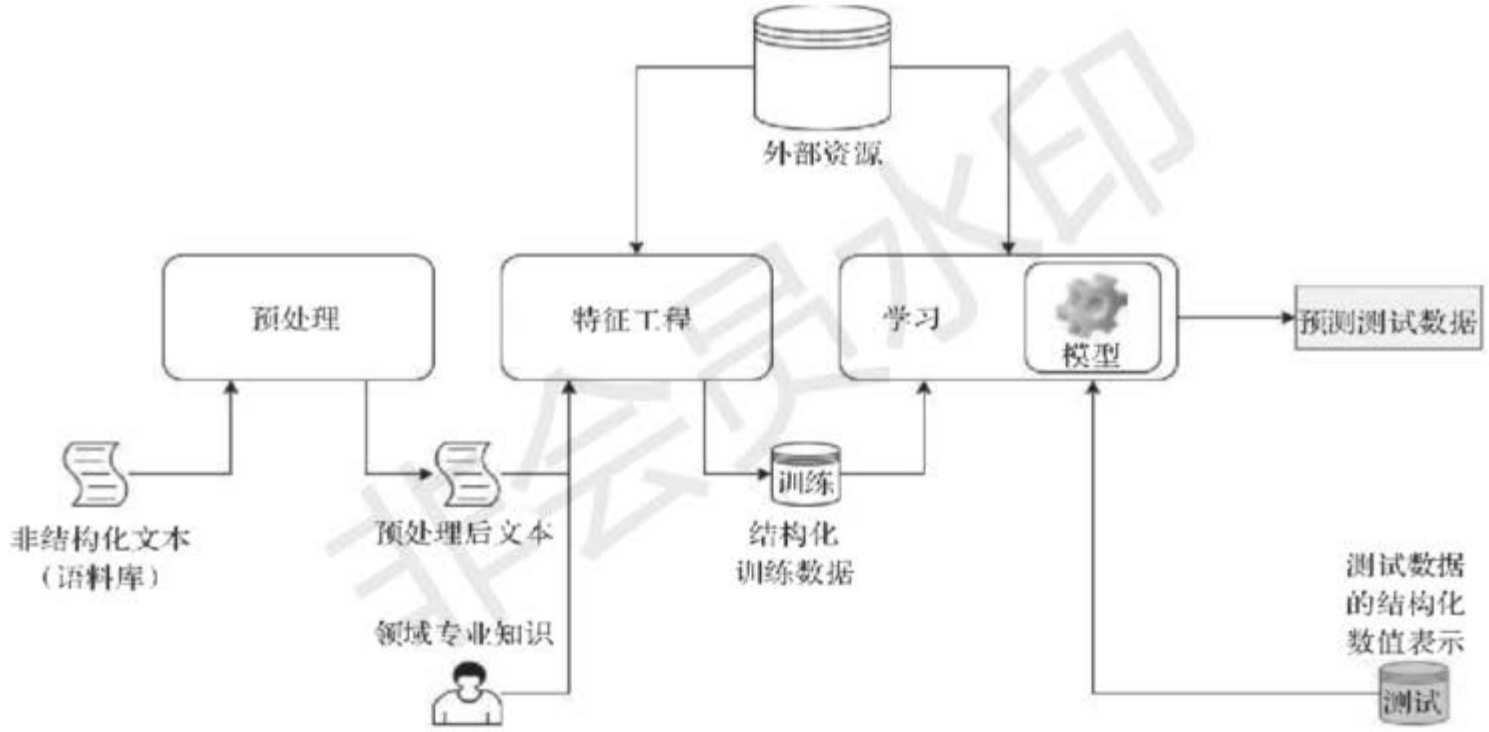
- **问答（QA）：**QA技术具有很高的商业价值，这些技术是聊天机器人和VA（例如，Google Assistant和Apple Siri）的基础。许多公司已经采用聊天机器人来提供客户支持
- **机器翻译（MT）：**MT是将句子/短语从源语言（例如，德语）转换为目标语言（例如，英语）的任务

知识讲解

传统NLP方法

传统NLP流程

知识讲解



传统NLP特征工程

- 词袋模型。将所有词语装进一个袋子里，不考虑其词法和语序的问题，即每个词语都是独立的，统计并产生每个词出现的频率。词袋方法的一个关键缺陷是，由于不再保留单词的顺序，它会丢失上下文信息

知识讲解

传统NLP特征工程（续1）

知识讲解

- N-Gram模型。N-Gram是一种基于统计语言模型，语言模型是一个基于概率的判别模型，它的输入是个句子（由词构成的顺序序列），输出是这句话的概率，即这些单词的联合概率。
- 常用的有Bi-gram(N=2)和Tri-gram(N=3)。例如：

  句子：L love deep learning

  Bi-gram: **{I, love}**, {love, deep}, **{deep, learning}**

  Tri-gram: {I, love, deep}, {love deep learning}

传统NLP特征工程（续2）

- N-Gram基本思想是将文本里面的内容按照字节进行大小为n的滑动窗口操作，形成了长度是n的字节片段序列。每一个字节片段称为一个gram，对所有gram的出现频度进行统计，并按照事先设置好的频度阈值进行过滤，形成关键gram列表，也就是这个文本向量的特征空间，列表中的每一种gram就是一个特征向量维度

知识讲解

传统NLP特征工程（续3）

知识讲解

- 共现矩阵。共现（co-occurrence）矩阵指通过统计一个事先指定大小的窗口内的word共现次数，以word周边的共现词的次数构成一个矩阵。例如：

I like deep learning.

I like NLP.

I enjoy flying.

counts	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0

- 优缺点
  - ✓ 优点：矩阵定义的词向量在一定程度上缓解了one-hot向量相似度为0的问题；
  - ✓ 缺点：但没有解决数据稀疏性和维度灾难的问题。

传统NLP特征工程（续4）

- 词频-逆文档频率（TF-IDF）。词频-逆文档频率表示词语的语义贡献度，表达式为：

知识讲解

$$TF - IDF = TF * IDF$$

其中：

$$IDF = \log(\frac{n}{docs(w, D) + 1})$$

n表示文档总数，与docs(w, D)表示词w所出现文件数

传统NLP缺陷

- 需要人工手动设计特征工程
- 传统NLP中使用的预处理步骤迫使我们对文本中嵌入的潜在有用信息（例如，标点符号和时态信息）进行取舍权衡，以便通过减少词汇量来使学习成为可能
- 传统方法需要各种外部资源才能表现良好，并且没有多少免费提供的资源
- 精度、准确度低

知识讲解

深度学习NLP方法

深度学习文本处理方式

知识讲解

- 将文本表达成类似图像、语音的的连续稠密数据，利用卷积神经网络提取文本的局部相关性
- 利用CNN/RNN强大的表征能力，自动提取特征，去掉繁杂的人工特征工程

深度学习文本表示方式

- 深度学习使用分布式单词表示技术（也称词嵌入表示），通过查看所使用的单词的周围单词（即上下文）来学习单词表示。这种表示方式将词表示为一个粘稠的序列，在保留词上下文信息同时，避免维度过大导致的计算困难。

知识讲解

深度学习文本表示方式（续1）

- 词向量示例。有原始语句：

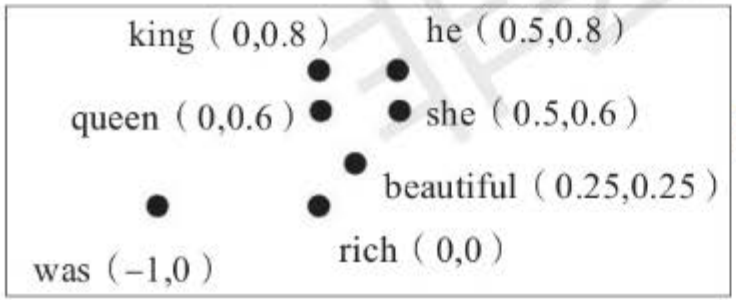
There was a very rich king. He had a beautiful queen. She was very kind.

知识讲解

做一些预处理并删除标点符号和无信息的单词：

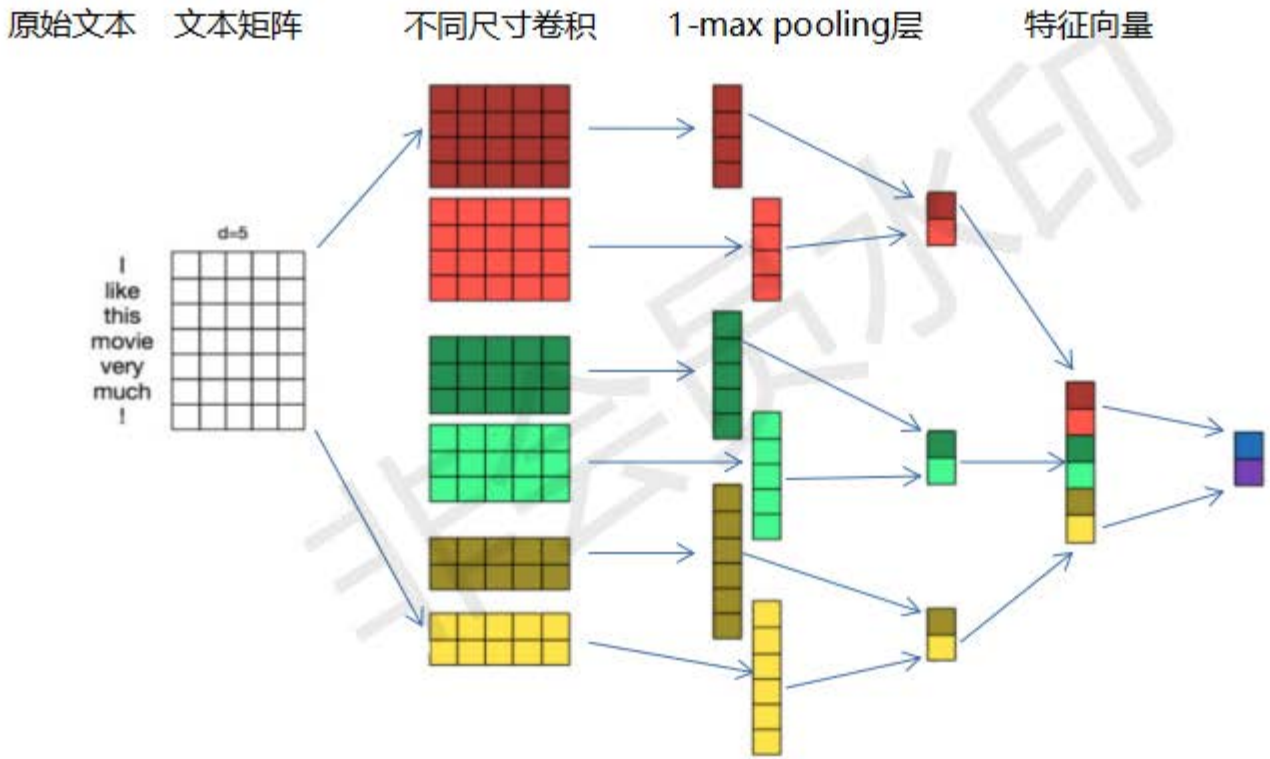
was rich king he had beautiful queen she was kind

表示成词向量：



queen = king - he + she
= [0, 0.8] - [0.5, 0.8] + [0.5, 0.6]
= [0, 0.6]

TextCNN模型



知识讲解

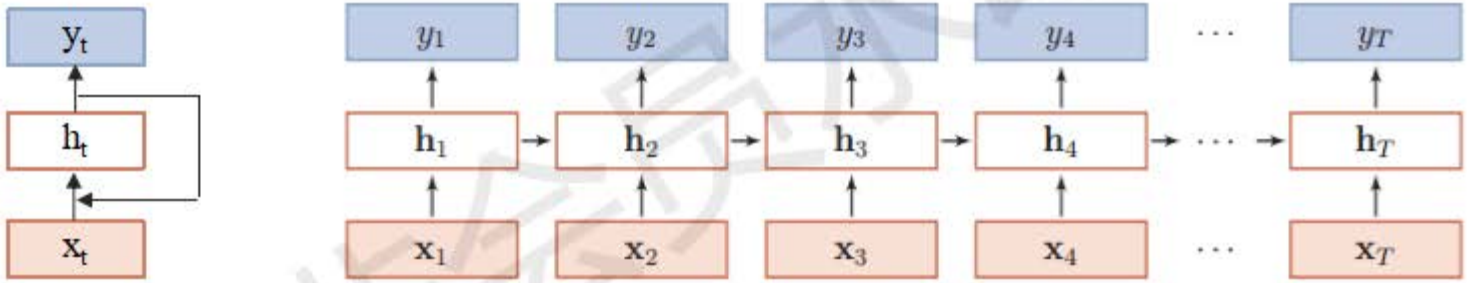
标准CNN模型的不足

知识讲解

- 假设数据之间是独立的。标准CNN假设数据之间是独立的，所以在处理前后依赖、序列问题（如语音、文本、视频）时就显得力不从心。这一类数据（如文本）和图像数据差别非常大，最明显的差别莫过于，文本数据对文字的前后次序非常敏感。所以，需要发展新的理论模型。
- 标准CNN络还存在一个短板，输入都是标准的等长向量，而序列数据长度是可变的。

RNN模型

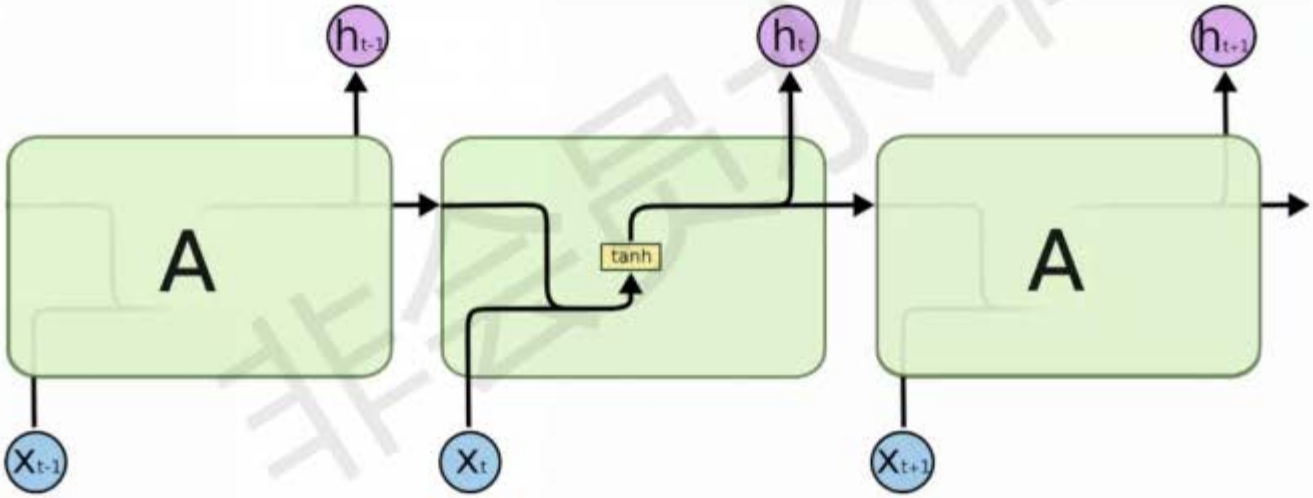
- 循环神经网络（Recurrent Neural Network，RNN）是一类具有短期记忆能力的神经网络，适合用于处理视频、语音、文本等与时序相关的问题



连接不仅存在于相邻的层与层之间（比如输入层-隐藏层），还存在于时间维度上的隐藏层与隐藏层之间（反馈连接，h_1到h_t）。某个时刻t，网络的输入不仅和当前时刻的输入相关，也和上一个时刻的隐状态相关

RNN模型（续1）

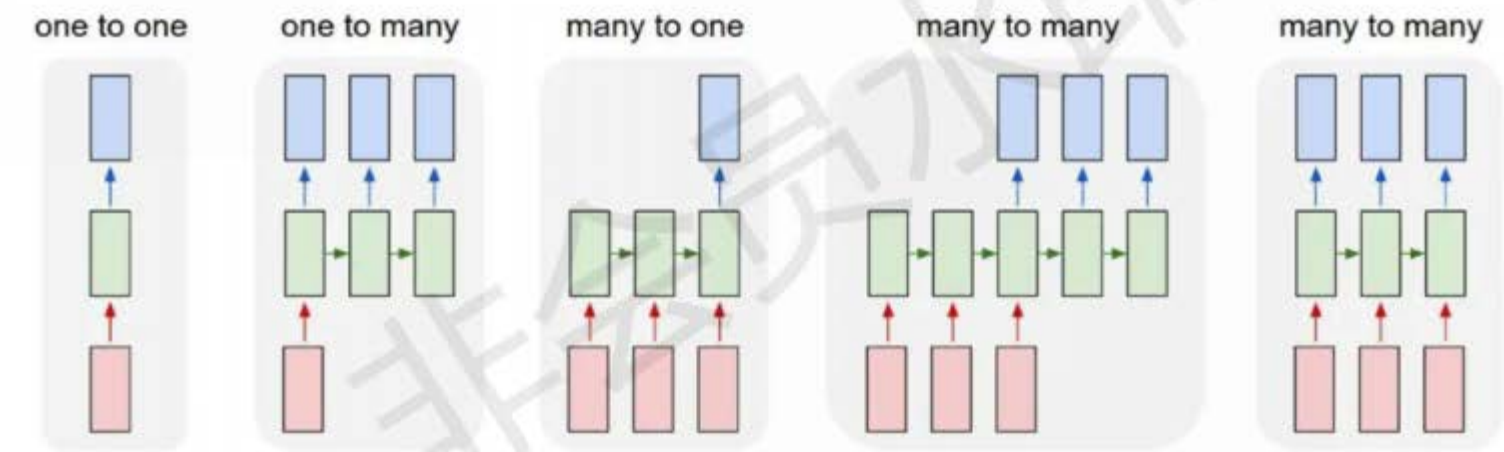
- 循环神经网络内部结构



知识讲解

RNN模型（续2）

- RNN模型输入输出关系对应模式



如：分类问题　　如：情感分析　　如：机器翻译　　如：词性标注

知识讲解

RNN模型（续3）

知识讲解

- RNN善于处理跟序列相关的信息，如：语音识别，语言建模，翻译，图像字幕。它能根据近期的一些信息来执行/判别当前任务。例如：

  白色的云朵漂浮在蓝色的_____

  我和他中午一起吃了个_____

  天空中飞过来一只_____

- RNN不善于处理远期依赖性任务。例如：

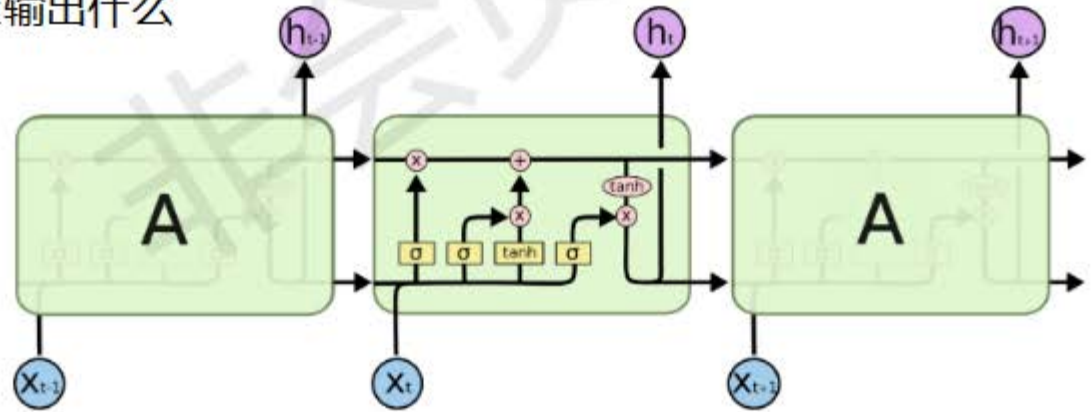
  我生长在中国，我是家里老三。我大哥叫大狗子，二哥叫二狗子，我叫三狗子，我弟弟叫_____。我的母语是_____。

LSTM模型

- 由于RNN具有梯度消失问题，因此很难处理长序列的数据。于是对RNN进行了改进，得到了长短期记忆网络模型（Long Short-Term Memory，简称LSTM）

知识讲解

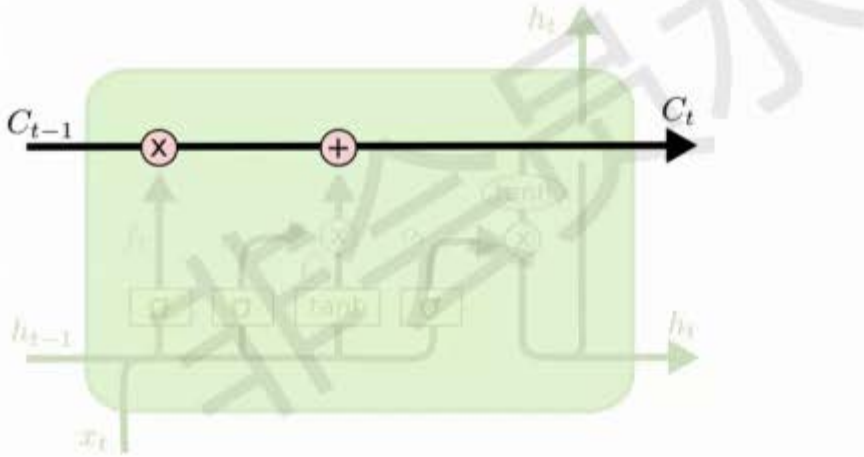
- 输入门：决定什么信息输入进来
- 遗忘门：决定从细胞状态中丢弃什么信息
- 输出门：决定输出什么



LSTM模型（续）

- LSTMs的核心是细胞状态，用贯穿细胞的水平线表示。细胞状态像传送带一样。它贯穿整个细胞却只有很少的分支，这样能保证信息不变的流过整个RNNs。

知识讲解



文本分类定义及应用

什么是文本分类

知识讲解

- 图像分类就是将文本划分到不同类别，例如新闻系统中，每篇新闻报道会划归到不同的类别。本质是找到一个有效的映射函数，实现从文本到类别的映射
- 文本分类主要包括：

文本分类的应用

- 内容分类（新闻分类）
- 邮件过滤（例如垃圾邮件过滤）
- 用户分类（如商城消费级别、喜好）
- 评论、文章、对话的情感分类（正面、负面、中性）

知识讲解

TextCNN实现文本分类

思路及实现

案例目标

- 目标：利用训练数据集，对模型训练，从而实现对中文新闻摘要类别正确划分

知识讲解

数据集介绍

知识讲解

- 来源：从网站上爬取56821条数据中文新闻摘要
- 数据内容：包含10种类别，国际、文化、娱乐、体育、财经、汽车、教育、科技、房产、证券

国际	4354	**汽车**	7469
文化	5110	**教育**	8066
娱乐	6043	**科技**	6017
体育	4818	**证券**	3654
财经	7432	**房产**	3858

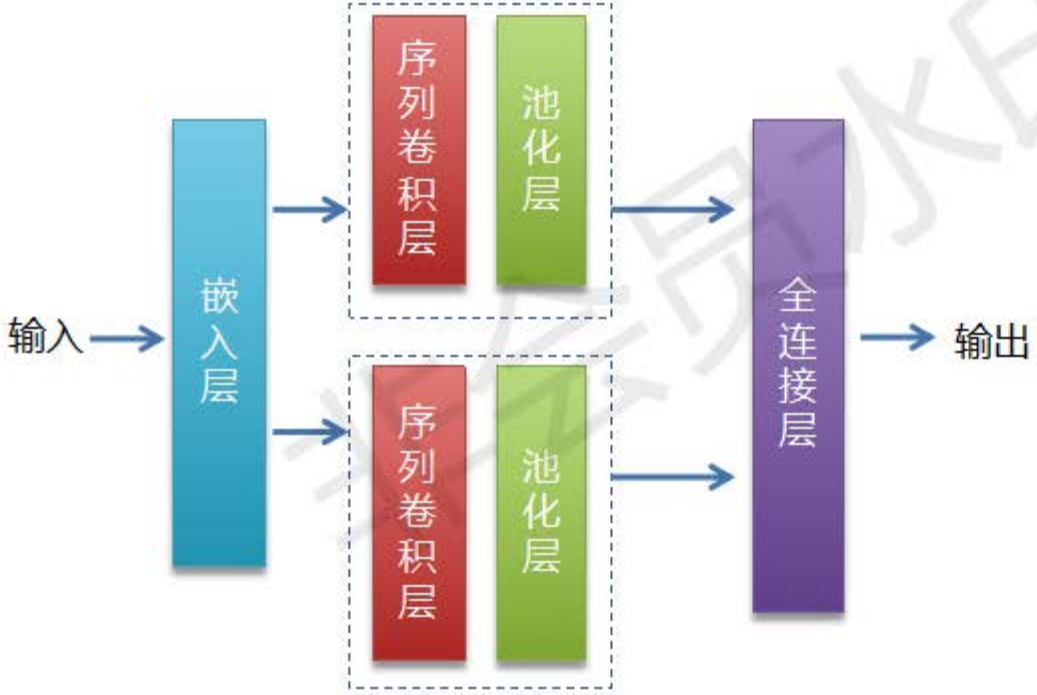
原始数据格式

知识讲解

```
6691168960903119367_!_3_!_财经_!_“擦鞋匠”注册公司当法人，83名投资人被骗108万！
6691211641503089159_!_3_!_财经_!_姜建清：做好“一带一路”投融资 互利是关键、互信是基石
6691460738675900941_!_3_!_财经_!_习近平向国际人工智能与教育大会致贺信
6691447881779380747_!_3_!_财经_!_西门子将在北京人工智能实验室 为德国本土以外的首个
6691205589617345031_!_3_!_财经_!_黑龙江省成品油零售资格审批大“瘦身” 申办加油站“不求人”
6691135110684606984_!_3_!_财经_!_所得税优惠确定延续 软件业如何提升“硬实力”
6691244489131950595_!_3_!_财经_!_国台办：两岸经济合作大潮大势是挡不住的
6691429876404060685_!_3_!_财经_!_刘长民调研市级重点项目
6691485644243337735_!_8_!_国际_!_外媒：苏丹军队开枪驱散示威者致9伤，军方和抗议者互相指责
6691479002646839822_!_8_!_国际_!_特朗普说中国经济不好，外交部：他们是中国经济主管部门？
6691483762988941835_!_8_!_国际_!_美伊局势升级，美国会要求白宫作出解释
6691481494600286727_!_8_!_国际_!_中国女留学生在澳失踪9个月，警方再次呼吁公众协助寻人
6691480635921400323_!_8_!_国际_!_只要自带桌子来店就免单！你是英国这家餐厅的目标用户吗？
```

网络模型介绍

知识讲解



总体步骤

- 数据预处理：解析数据文件，编码，建立训练集、测试集
- 训练与模型评估
- 输入测试数据，进行预测

知识讲解

数据预处理

- 字典编码

{'字': 0, '龙': 1, '重': 2, '骏': 3, '夏': 4, '袍': 5, '拔': 6, '余': 7, '餐': 8, '潼': 9, '驳': 10, '髓': 11, '羊': 12, '找': 13, '概': 14, '终': 15, '革': 16, '睇': 17, '哗': 18, '龄': 19, '务': 20, '启': 21, '企': 22, '干': 23, '魔': 24, '接': 25, '咽': 26, '滨': 27, '朔': 28, '助': 29, '沈': 30, '掺': 31, '阪': 32, '旅': 33, '梢': 34, '煦': 35, '龍': 36, 'ㄩ': 37, '或': 38, '槐': 39, '禛': 40, '岜': 41, '劭': 42, '符': 43, '"': 44, '镑': 45, '肌': 46, '纡': 47, '溧': 48, '问': 49, '搦': 50, '雁': 51, '监': 52, '靖': 53, '伢': 54, '幔': 55, '饺': 56, '屉': 57, '丨': 58, '函': 59, '鱿': 60, '唐': 61, '错': 62, '癌': 63, '惯': 64, '滋': 65, '都': 66, '圣': 67, '脊': 68, '史': 69, '磺': 70, '】': 71, '性': 72, '鸢': 73, '玲': 74, '肝': 75, '珐': 76, '落': 77, '意': 78, '犷': 79, '屁': 80, '珙': 81, '靓': 82, '珏': 83, '伯': 84, '俏': 85, '检': 86, '谷': 87, '栓': 88, '甄': 89, '钏': 90, '戍': 91, '硒': 92, '享': 93, '琳': 94, '农': 95, '电': 96, '芸': 97, '调': 98, '讽': 99, '筑': 100, '疮': 101, '睛': 102, '尬': 103, '铮': 104, '邻': 105, '棠': 106, '湖': 107, '·': 108, '聆': 109, '石': 110, '煽': 111, '莎': 112, '转': 113, '盹': 114, '示': 115, '葫': 116, '坂': 117, '苹': 118, '这': 119, '楠': 120, '猪': 121, '尾': 122, '慕': 123, '麒': 124, '骗': 125, '椎': 126, '恐': 127, '拆': 128, '辰': 129, '惬': 130, '橡': 131, '

知识讲解

数据预处理（续）

- 文本编码

```
4424,3559,1702,4216,1538,1248,4710,4337,530,472,145,1499,4424,1682,3763,1763,4239,3290,1795,3365,4419    5
4048,157,2714,4364,1619,1073,267,3327,1499,3658,2085,1706,4197,2614,4662,1292,2882,638,3397,2629,2064    2
1451,2467,2318,4211,2548,1676,2796,4556,514,984,1162,405,2,1495,4471,717,3548,536       7
1400,1493,334,1641,3935,2290,4406,1499,1876,3124,872,593,1868,4519,3222,1499,4662,2052,3589,1871,4197,2452,13,2537,163,946
,1564,1563      5
1105,1964,1676,3725,1388,207,2451,2818,3038,514,3851,3772,2157,1660,1538,4343,219,303    9
2861,2950,1342,1391,1221,2085,1871,4197,4262,3190,78,1654,4419,4410,1195,932,21,2348,1273,1342,2620,1221,2045,3477,402,213
7,348,2285,4665,2883,3436,2052,4471,516,2620     5
1854,1854,1499,2917,3730,1044,107,358,52,3644,3495,3084,1402,2587,1221,4419      3
464,3507,1869,4216,1643,1828,3437,4208,3045,3690,271,3345,4585,946,1689,1594,3618,3112   3
2699,2861,788,703,2102,1493,1795,1216,2276,2829,3046,4419,3290,1292,2219,1643,119,411,8,1127,2085,412,3977,2348,3770,4344,
2064     8
```

知识讲解

关键代码

- 生成字典

知识讲解

```
# 生成数据字典：把每一个汉字编码成一个数字，并存入字典文件
def create_dict(data_file_path, dict_file_path):
    dict_set = set()
    with open(data_file_path, "r", encoding="utf-8") as f:  # 读取数据文件
        lines = f.readlines()
    # 把数据生成一个元组
    for line in lines:
        title = line.split("_!_")[-1].replace("\n", "")
        # print("title:", title)
        for w in title:
            dict_set.add(w)  # 将文字添加到集合中
    # 把文字转换编码为数字
    dict_list = []
    i = 0
    for s in dict_set:
        dict_list.append([s, i])  # 文字-数字 映射添加到dict_list
        i += 1
        print(s, ":", i)
    # 添加未知字符
    dict_txt = dict(dict_list)  # 将dict_list转换为字典
    end_dict = {"<unk>": i}
    dict_txt.update(end_dict)  # 将未知字符编码添加待编码字典中
    # 将字典保存到文件
    with open(dict_file_path, "w", encoding="utf-8") as f:
        f.write(str(dict_txt))
    print("生成数据字典完成!")
```

关键代码（续1）

- 创建测试/训练集

```
def create_data_list(data_root_path):
    test_file_path = os.path.join(data_root_path, test_file)
    with open(test_file_path, "w") as f: # 清空训练数据文件
        pass
    train_file_path = os.path.join(data_root_path, train_file)
    with open(train_file_path, "w") as f: # 清空测试数据文件
        pass
    dict_file_path = os.path.join(data_root_path, dict_file)  # 合并路径
    with open(dict_file_path, "r", encoding="utf-8") as f_dict:
        dict_txt = eval(f_dict.readlines()[0])  # 由文件生成字典
    news_file_path = os.path.join(data_root_path, data_file)
    with open(news_file_path, "r", encoding="utf-8") as f_data: # 读入原始数据
        lines = f_data.readlines()
    # 将文章中每个字转换为编码，存入新的文件
    i = 0
    for line in lines:
        words = line.replace("\n", "").split("_!_")
        lable = words[1]  # 分类
        title = words[3]  # 标题
        if i % 10 == 0:  # 每10笔写入测试文件
            with open(test_file_path, "a", encoding="utf-8") as f_test:
                new_line = line_encoding(title, dict_txt, lable)
                f_test.write(new_line)
        else:  # 其它写入训练文件
            with open(train_file_path, "a", encoding="utf-8") as f_train:
                new_line = line_encoding(title, dict_txt, lable)
                f_train.write(new_line)
        i += 1
    print("生成训练、测试数据文件结束!")
```

知识讲解

关键代码（续2）

- 读取器

```
100  # 获取字典长度
101  def get_dict_len(dict_path):
102      with open(dict_path, "r", encoding="utf-8") as f:
103          line = eval(f.readlines()[0])  # 读取文件
104
105      return len(line.keys())
106
107
108  # 创建数据读取器train_reader和test_reader
109  # 将传入数据由字符串转换为数字
110  def data_mapper(sample):
111      data, label = sample
112      val = [int(w) for w in data.split(",")]
113
114      return val, int(label)
```

知识讲解

关键代码（续3）

- 读取器2

知识讲解

```
117     # 创建读取器train_reader，每次读取一行编码后的数据
118     # 并拆分为标题(编码后的)、标记
119     def train_reader(train_file_path):
120         def reader():
121             with open(train_file_path, "r") as f:
122                 lines = f.readlines()
123                 np.random.shuffle(lines)  # 打乱数据
124 
125                 for line in lines:
126                     data, label = line.split("\t")
127                     yield data, label
128 
129         # 多线程下，使用自定义映射器 reader 返回样本到输出队列
130         # 将mapper生成的数据交给reader进行二次处理，并输出
131         return paddle.reader.xmap_readers(data_mapper,  # reader数据函数
132                                           reader,  # 产生数据的reader
133                                           cpu_count(),  # 处理样本的线程数(和CPU核数一样)
134                                           1024)  # 数据缓冲队列大小
```

关键代码（续4）

- 定义网络

```
154  # 定义网络
155  def CNN_net(data, dict_dim, class_dim=10, emb_dim=128, hid_dim=128, hid_dim2=98):
156      # embeding(词向量)层：将高度稀疏的离散输入嵌入到一个新的实向量空间
157      # 以使用更少的维度，表示更丰富的信息
158      emb = fluid.layers.embedding(input=data, size=[dict_dim, emb_dim])
159
160      # 第一个卷积、池化层
161      # sequence_conv_pool: 序列卷积、池化层构成
162      conv_1 = fluid.nets.sequence_conv_pool(input=emb,  # 输入
163                                             num_filters=hid_dim,  # 卷积核数目
164                                             filter_size=3,  # 卷积核大小
165                                             act="tanh",  # 激活函数
166                                             pool_type="sqrt")  # 池化类型
167
168      conv_2 = fluid.nets.sequence_conv_pool(input=emb,  # 输入
169                                             num_filters=hid_dim2,  # 卷积核数目
170                                             filter_size=4,  # 卷积核大小
171                                             act="tanh",  # 激活函数
172                                             pool_type="sqrt")  # 池化类型
173
174      output = fluid.layers.fc(input=[conv_1, conv_2], size=class_dim, act="softmax")
175      return output
```

知识讲解

关键代码（续5）

- 损失函数及优化器

```
182     # 定义输入数据
183     words = fluid.layers.data(name="words", shape=[1], dtype="int64",
184                               lod_level=1)  # 张量层级
185     label = fluid.layers.data(name="label", shape=[1], dtype="int64")
186
187     # 获取字典长度
188     dict_dim = get_dict_len(os.path.join(data_root_path, dict_file))
189     # 调用函数，生成卷积神经网络
190     model = CNN_net(words, dict_dim)
191     # 定义损失函数
192     cost = fluid.layers.cross_entropy(input=model, label=label)  # 交叉熵作为损失函数
193     avg_cost = fluid.layers.mean(cost)
194     # 计算准确率
195     acc = fluid.layers.accuracy(input=model,  # 输入，即预测网络
196                                 label=label)  # 数据集标签
197
198     # 复制program用于测试
199     test_program = fluid.default_main_program().clone(for_test=True)
200
201     # 定义优化器
202     # Adaptive Gradient(自适应梯度下降优化)，对于不同参数自动调整梯度大小
203     # 对于数据较为稀疏的特征梯度变大，数据较为稠密的特征梯度减小
204     optimizer = fluid.optimizer.AdagradOptimizer(learning_rate=0.002)
205     opt = optimizer.minimize(avg_cost)  # 求avg_cost最小值
```

知识讲解

关键代码（续6）

- 执行器及数feeder

知识讲解

```
207  # 创建执行器
208  place = fluid.CPUPlace()
209  # place = fluid.CUDAPlace(0)
210  exe = fluid.Executor(place)
211  exe.run(fluid.default_startup_program())  # 初始化系统参数
212
213  # 准备数据
214  ## 训练数据读取器
215  tr_reader = train_reader(os.path.join(data_root_path, train_file))
216  train_reader = paddle.batch(reader=tr_reader, batch_size=128)
217  ## 测试数据读取器
218  ts_reader = test_reader(os.path.join(data_root_path, test_file))
219  test_reader = paddle.batch(reader=ts_reader, batch_size=128)
220
221  feeder = fluid.DataFeeder(place=place, feed_list=[words, label])  # feeder
```

关键代码（续7）

- 执行训练

```
# 开始训练
for pass_id in range(EPOCH_NUM):
    for batch_id, data in enumerate(train_reader()):
        train_cost, train_acc = exe.run(program=fluid.default_main_program(),
                                        feed=feeder.feed(data),
                                        fetch_list=[avg_cost, acc])
        # 每100次打印一次cost, acc
        if batch_id % 100 == 0:
            print("pass_id:%d, batch_id:%d, cost:%0.5f, acc:%0.5f" %
                  (pass_id, batch_id, train_cost[0], train_acc[0]))

    # 使用测试数据集测试
    test_costs_list = []
    test_accs_list = []
    for batch_id, data in enumerate(test_reader()):
        test_cost, test_acc = exe.run(program=test_program,
                                      feed=feeder.feed(data),
                                      fetch_list=[avg_cost, acc])
        test_costs_list.append(test_cost[0])
        test_accs_list.append(test_acc[0])

    # 计算平均损失值和准确率
    avg_test_cost = (sum(test_costs_list) / len(test_costs_list))
    avg_test_acc = (sum(test_accs_list) / len(test_accs_list))

    print("pass_id:%d, test_cost:%0.5f, test_acc:%0.5f" %
          (pass_id, avg_test_cost, avg_test_acc))
```

知识讲解

关键代码（续8）

- 测试

知识讲解

```
16  # 将句子转换为编码
17  def get_data(sentence):
18      dict_file_path = os.path.join(data_root_path, dict_file)
19      with open(dict_file_path, "r", encoding="utf-8") as f:
20          dict_txt = eval(f.readlines()[0])
21
22      # dict_txt = dict(dict_txt)
23      keys = dict_txt.keys()
24      ret = []
25      for s in sentence:
26          if not s in keys:
27              s = "<unk>"
28          ret.append(int(dict_txt[s]))
29      return ret
30
31
32  # 读取模型
33  model_save_dir = "model/news_classify/"
34  place = fluid.CPUPlace()
35  exe = fluid.Executor(place)
36  exe.run(fluid.default_startup_program())
```

关键代码（续9）

- 测试2

```
print("加载模型:", model_save_dir)
infer_program, feeded_var_names, target_var = \
        fluid.io.load_inference_model(dirname=model_save_dir, executor=exe)
# 生成测试数据
texts = []
data1 = get_data("在获得诺贝尔文学奖7年之后，莫言15日晚间在山西汾阳贾家庄如是说")
data2 = get_data("综合'今日美国'、《世界日报》等当地媒体报道，芝加哥河滨警察局表示")
data3 = get_data("中国队无缘2020年世界杯")
data4 = get_data("中国人民银行今日发布通知，提高准备金率，预计释放4000亿流动性")
data5 = get_data("10月20日,第六届世界互联网大会正式开幕")

texts.append(data1)
texts.append(data2)
texts.append(data3)
texts.append(data4)
texts.append(data5)
```

知识讲解

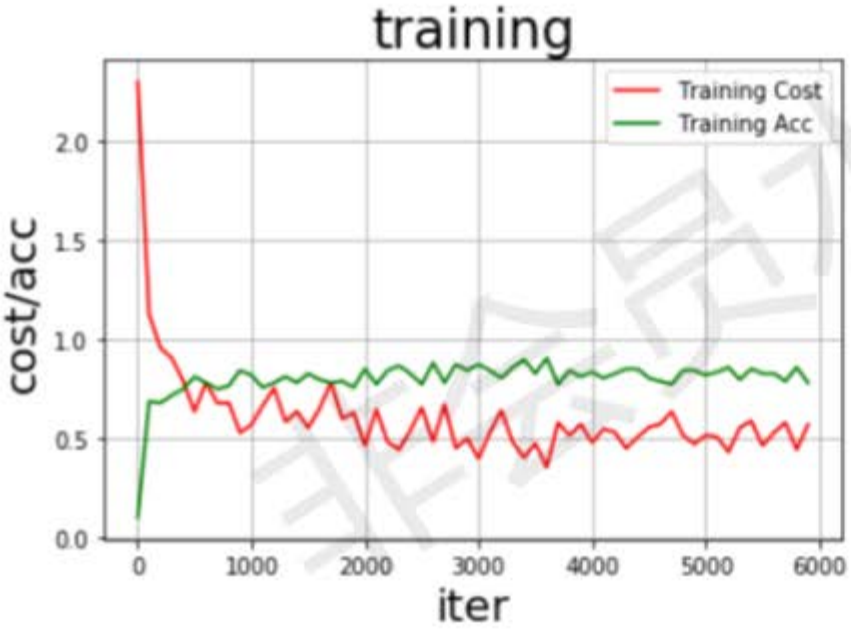
关键代码（续10）

- 测试3

```
55    # 获取每个句子词数量
56    base_shape = [[len(c) for c in texts]]
57    # 生成预测数据
58    tensor_words = fluid.create_lod_tensor(texts, base_shape, place)
59    # 执行预测
60    # tvar = np.array([target_var], dtype="int64")
61    result = exe.run(program=infer_program,
62                     feed={feeded_var_names[0]: tensor_words},
63                     fetch_list=target_var)
64                     # fetch_list=tvar)
65
66    names = ["文化", "娱乐", "体育", "财经", "房产", "汽车", "教育", "科技", "国际", "证券"]
67
68    # 获取结果概率最大的label
69    for i in range(len(texts)):
70        lab = np.argsort(result)[0][i][-1]
71        print("预测结果: %d, 名称:%s, 概率:%f" %
72              (lab, names[lab], result[0][i][lab]))
```

知识讲解

训练过程


training
Training Cost
Training Acc
cost/acc
iter
0.0
0.5
1.0
1.5
2.0
0
1000
2000
3000
4000
5000
6000

知识讲解

测试结果

1. 在获得诺贝尔文学奖7年之后，莫言15日晚间在山西汾阳贾家庄如是说
2. 综合'今日美国'、《世界日报》等当地媒体报道，芝加哥河滨警察局表示
3. 中国队无缘2020年世界杯
4. 中国人民银行今日发布通知，提高准备金率，预计释放4000亿流动性
5. 10月20日,第六届世界互联网大会正式开幕

知识讲解

```
预测结果: 0, 名称:文化, 概率:0.943867
预测结果: 8, 名称:国际, 概率:0.626188
预测结果: 2, 名称:体育, 概率:0.599507
预测结果: 3, 名称:财经, 概率:0.910002
预测结果: 7, 名称:科技, 概率:0.678429
```

案例1：利用TextCNN实现文本分类

- 代码详见：news_classify.py

课堂练习

今日总结

- 文本分类概述
- 常用自然语言处理模型
- 案例：新闻分类