循环神经网络
——自然语言处理基础

- 文本数据的表示方法
- 词嵌入的生成和使用
- 循环神经网络的基本原理
- 基于Keras实现循环神经网络

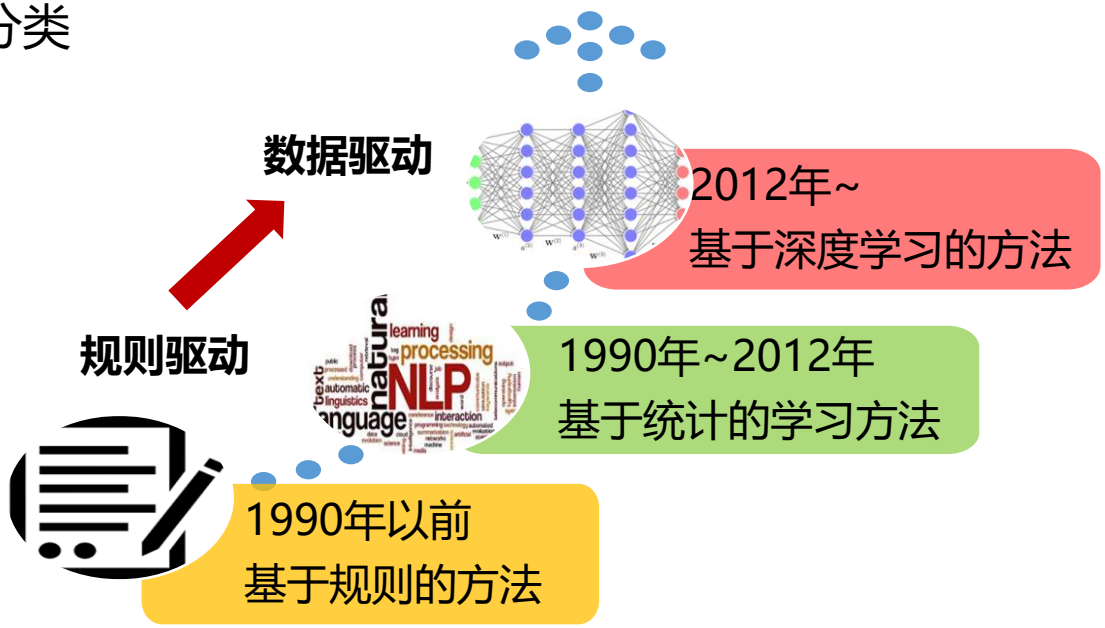
文本数据的表示方法

文本

文本是一种序列数据，是字符或单词构成的序列，文本处理也称为自然语言处理(Natural Language Processing)。

典型应用

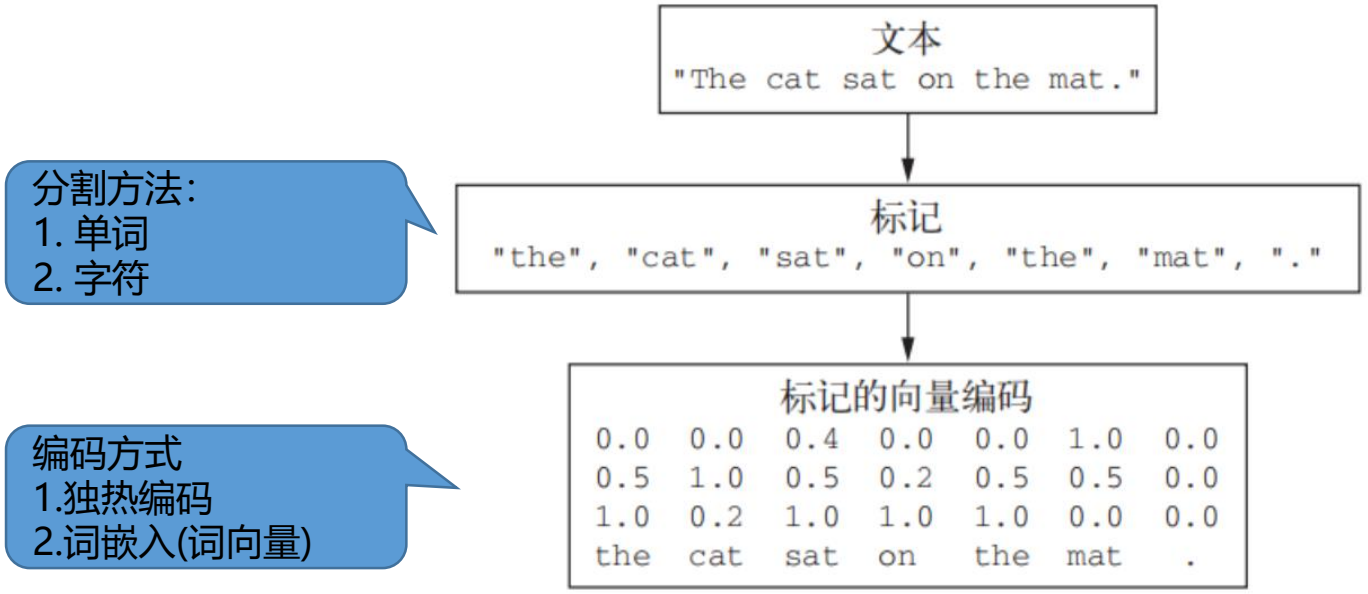
- 搜索引擎
- 文本主题与标签分类
- 文本创作与生成
- 机器翻译
- 情感分析
- 舆情监控
- 语音识别
- 对话机器人



文本向量化

将文本转换为数值张量的过程，称为文本向量化(Vectorize)。

首先将文本分割为单词或字符(称为标记token)，然后将每个标记转换为一个向量，这些向量组合为序列张量，就可以作为神经网络的输入。



分词

将连续的自然语言文本，切分成具有语义合理性和完整性的词汇序列。

- 英文：单词间以空格分割
- 汉语：以字为基本单位，词语之间无明确区分，而且存在切分歧义。

例如：致毕业和尚未毕业的同学

致	毕业	和	尚未	毕业	的	同学
致	毕业	和尚	未	毕业	的	同学

分词主要算法：

- 基于规则：按策略与词典匹配。
- 基于统计：依赖训练语料、考虑相邻字。如N-gram、隐马尔可夫模型、条件随机场CRF等。
- 基于理解：分词同时进行语法、语义分析。如专家系统、神经网络方法等

Python的中文分词工具包：jieba、SnowNLP、语言云、THULAC、NLPIR

独热编码(one-hot)

将每个单词与一个唯一索引的整数 i 相关联，并将该整数表示为长度为N(N是词表大小)的二进制向量，其中只有第 i 位为1。

杭州 [0,0,0,0,0,0,0,1,0,……，0,0,0,0,0,0,0]
上海 [0,0,0,0,1,0,0,0,0,……，0,0,0,0,0,0,0]
宁波 [0,0,0,1,0,0,0,0,0,……，0,0,0,0,0,0,0]
北京 [0,0,0,0,0,0,0,0,0,……，1,0,0,0,0,0,0]

1. 二进制、稀疏、维度高。
2. 维度取决于语料库中词数，通常高于20000，导致维数灾难
3. 是一种词袋模型，向量之间相互独立，忽略文本在词序、语法和句法的关联关系。

one-hot词向量：
- 稀疏
- 高维
- 硬编码

独热编码实现：可在文档集载入时计算完成。

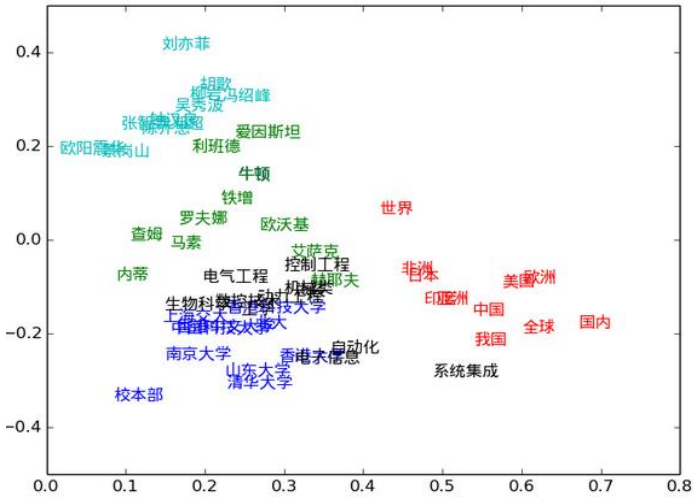
词嵌入的生成和使用

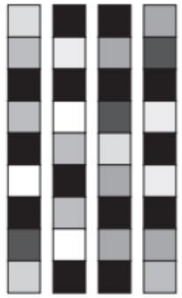
词嵌入(词向量)

将one-hot编码转化为低维度的连续值，即稠密向量。

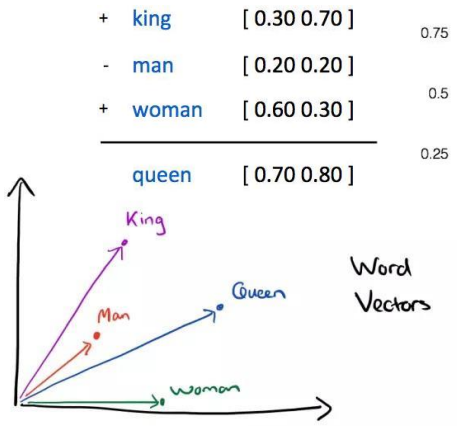
刘亦菲 [0.15, 0.41]
胡歌 [0.20, 0.32]
中国 [0.55. -0.15]
美国 [0.58, -0.10]

1. 浮点数、密集、低维
2. 维度一般选 256、512 或 1024
3. 语法上、语义上相近的词距离接近





词嵌入：
- 密集
- 低维
- 从数据中学习得到



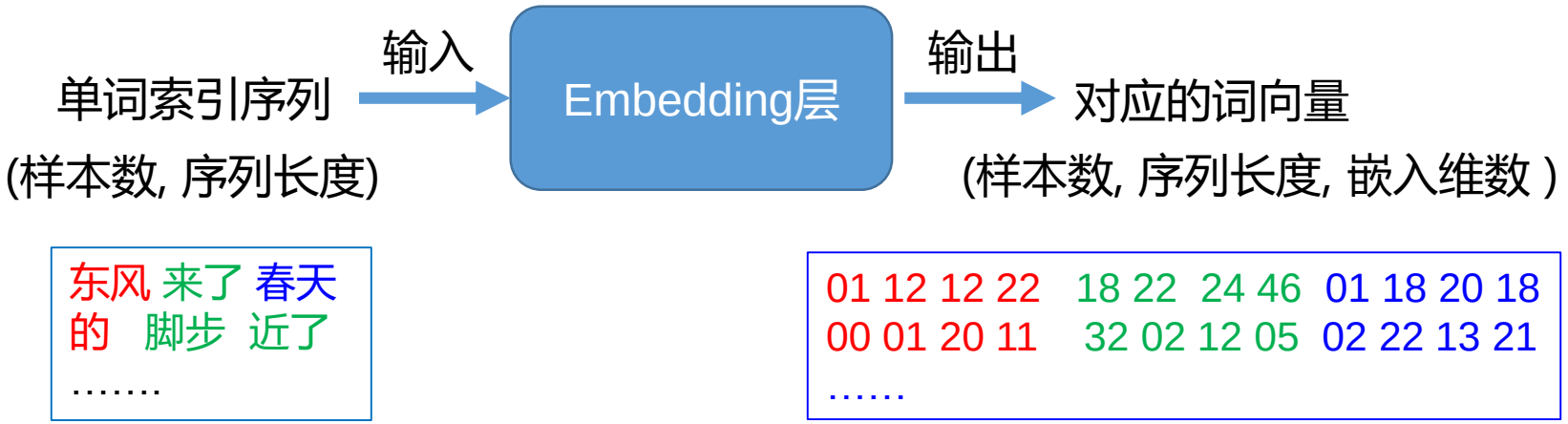
词嵌入实现：从大量语料集学习获得。
1）完成文本分类等主任务的同时学习获得，学习方法与神经网络权重类似。
2）加载其他任务上获得的预训练词嵌入直接使用。

学习获得一个词嵌入

在神经网络中添加Embedding层可以从训练集学习获得一个词嵌入空间。即将单词的整数索引转换为低维的词向量表示。

```
from keras.layers import Embedding
embedding_layer = Embedding(vocab_size,vector_dimension,input_length)
```

vocab_size: 总单词个数
vector_dimension：词向量维度
input_length：输入序列的长度



案例1：使用词嵌入进行电影评论分类

1. 读取数据并进行预处理

```
#数据准备
from keras.datasets import imdb
from keras.preprocessing import sequence
max_features = 10000  #最大特征数，即该训练集最常见的前10000个单词
maxlen = 20      #每条评论最大长度20，超过将被截断
#加载数据集，获得各评论文本对应的整数列表
(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=max_features)
print(x_train[:1,])
#整数列表填充处理：即超长截断、不足则前补0，默认从尾部截取，得到等长二维整数张量(samples,maxlen)
x_train =sequence.pad_sequences(x_train, maxlen=maxlen)
x_test = sequence.pad_sequences(x_test, maxlen=maxlen)
print(x_train[:1,])
```

第一条评论的索引数据序列, 有218个元素

[list([1, 14, 22, 16, 43, 530, 973, 1622, 1385, 65, 458, 4468, 66, 3941, 4, 173, 36, 256, 5, 25, 100, 43,

…….

25, 104, 4, 226, 65, 16, 38, 1334, 88, 12, 16, 283, 5, 16, 4472, 113, 103, 32, 15, 16, 5345, 19, 178, 32])]

处理后的第一条评论的索引整数序列，从后截取20个整数

[[65   16   38 1334   88   12   16 283   5   16 4472 113 103   32   15   16 5345   19 178   32]]

2. 建立和训练神经网络，在最底层添加Embedding层

```
#建立和训练神经网络
from keras.models import Sequential
from keras.layers import Flatten, Dense, Embedding
model = Sequential()
#添加embedding层，将10000个单词嵌入到维度8的向量中，输入序列长度20
model.add(Embedding(10000, 8, input_length=20))
#将三维的嵌入张量展平成形状为(samples, maxlen * 8) 的二维张量
model.add(Flatten())
model.add(Dense(1, activation='sigmoid'))
model.summary()

model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['acc'])
history = model.fit(x_train, y_train,
epochs=10,
batch_size=32,
validation_split=0.2)
```

Model.summary()

Layer (type)	Output Shape	Param #
embedding_1(Embedding)	(None, 20, 8)	80000
flatten_1(Flatten)	(None, 160)	0
dense_1(Dense)	(None, 1)	161

Total params: 80,161
Trainable params: 80,161
Non-trainable params: 0

最后一轮：
样本长度20、嵌入维度8
val_loss: 0.5303 - val_acc: 0.7474
样本长度50、嵌入维度8
val_loss: 0.4267 - val_acc: 0.8108

使用预训练的词嵌入

如果训练数据少，可以直接使用大规模文本训练得到的预训练模型，能得到比较好的性能。

常用的预训练模型：

word2Vec： 是一个词向量训练工具，可以在百万数量级的词典和上亿的数据集上进行高效地训练；具得到的训练结果——词向量。

GloVe：词嵌入数据库(英文)，从维基百科和Common Crawl数据学习得到。

其他中文词嵌入下载：
从百度百科、维基百科、人民日报、知乎、四库全书等学习
https://github.com/Embedding/Chinese-Word-Vectors
https://baijiahao.baidu.com/s?id=1600509930259553151&wfr=spider&for=pc

word2Vec预训练词嵌入

Gensim是一款开源的第三方Python工具包，用于从原始的非结构化的文本中，无监督地学习到文本隐层的主题向量表达。 支持word2Vec算法。

安装命令：

pip install gensim

预训练词嵌入有两种保存方式：

1. *.model，该类模型可以直接加载，并追加训练

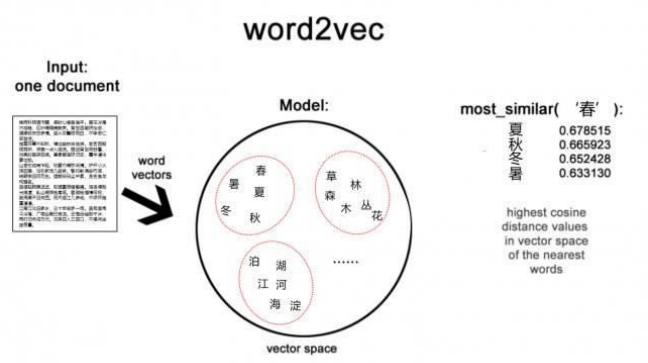
```
model = gensim.models.Word2Vec.load('/tmp/mymodel.model')
model.train(more_sentences)
```

2. 以txt文件或二进制文件保存，可直接查看，但不能追加训练

```
model = gensim.models.KeyedVectors.load_word2vec_format('/tmp/mymodel.txt',binary = False)
model = gensim.models.KeyedVectors.load_word2vec_format('/tmp/mymodel.bin',binary = True)
```

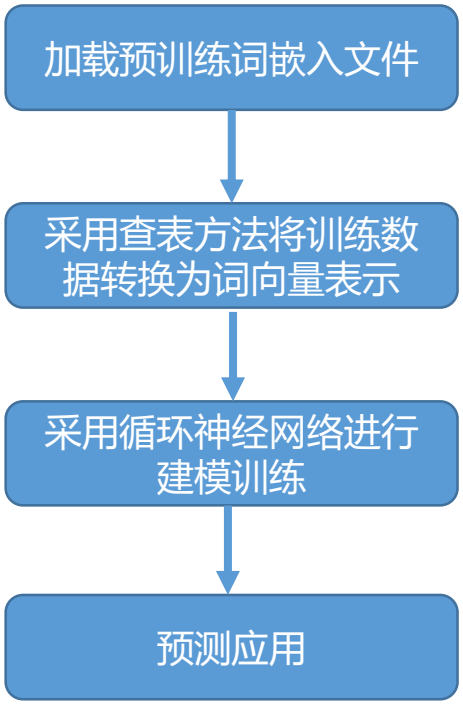
word2vec词向量txt格式：第一行说明数量和维度

```
4 3
太阳 0.134 0.254 0.354
脸    0.245 0.335 0.377
红    0.345 0.488 0.553
起来 0.564 0.234 0.564
```



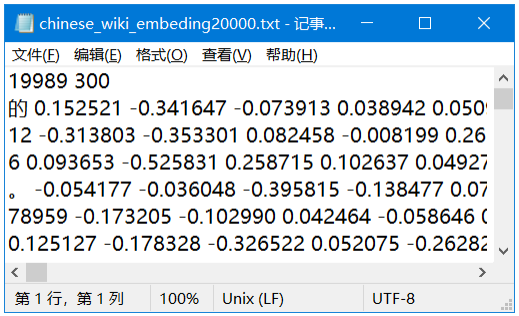
案例2：使用预训练词嵌入

使用采用Wiki百科训练得到的预训练词嵌入文件，对电影评论数据进行情感分类建模，并进行预测应用。

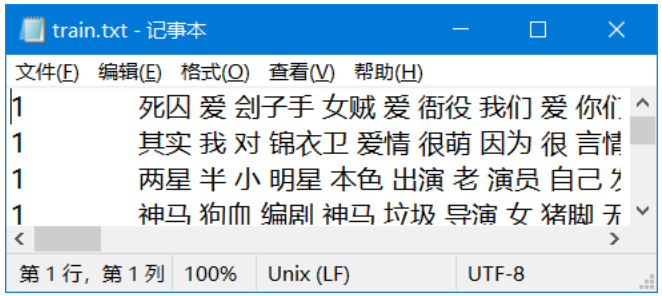


有关文件

（1）预训练词嵌入文件：
chinese_wiki_embeding20000.txt



（2）训练数据为已分词的文本，训练数据19998条，测试数据5998条：
data/train.txt
data/test.txt



程序代码: WordEmbeddingWord2Vector.py

文本数据表示 小结

文本是一种由字符或单词构成的序列数据。文本向量化将文本转换为数值张量，才能作为神经网络的输入。

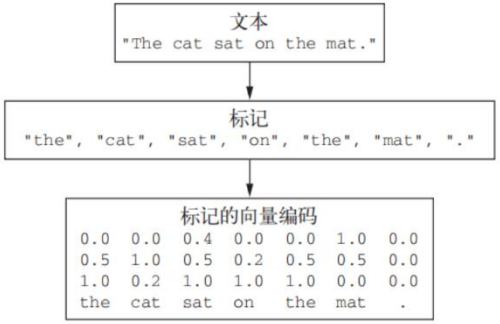
文本首先进行分词，然后将单词转换为词向量。

文本向量化两种表示方法：

- 独热编码(one-hot) ：二进制、高维、稀疏。忽略词序和语义。
- 词嵌入(词向量)：浮点数、低维、密集。体现语义关系

文本向量化实现方法：

- 在神经网络中添加Embedding层，在完成主任务的同时学习得到。
- 使用预训练的词嵌入，如word2vector、Glove等

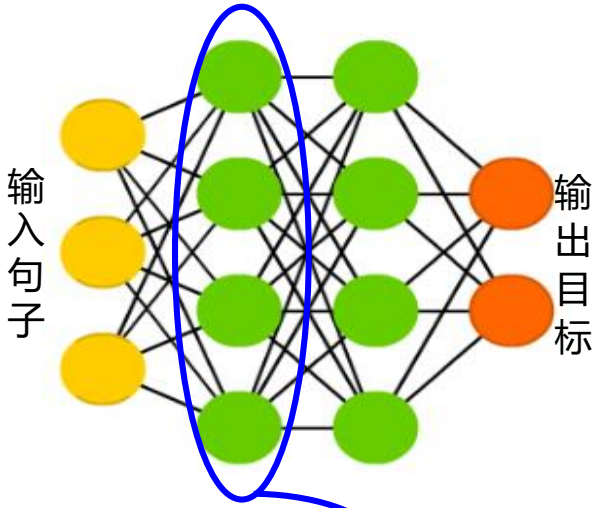


循环神经网络基本原理

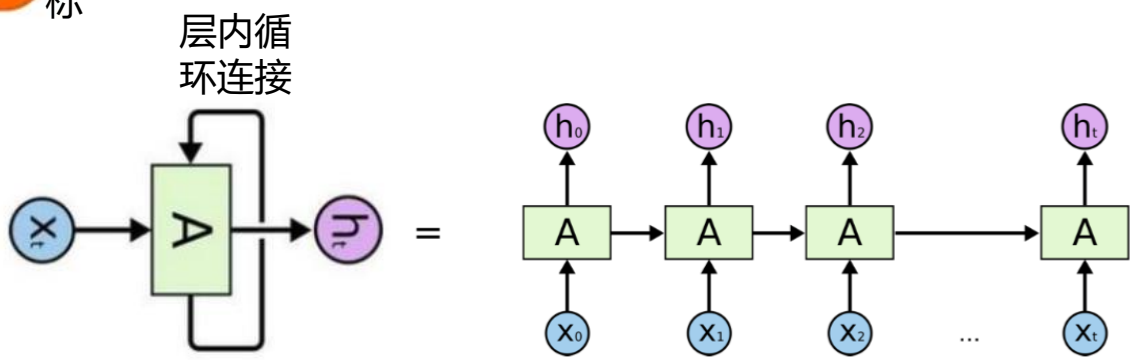
前馈神经网络处理序列数据

有以下弱点：

- 层与层之间连接，每层内节点无连接，无法体现词序关系
- 输入和输出的长度都是固定的，无法处理变长的序列数据

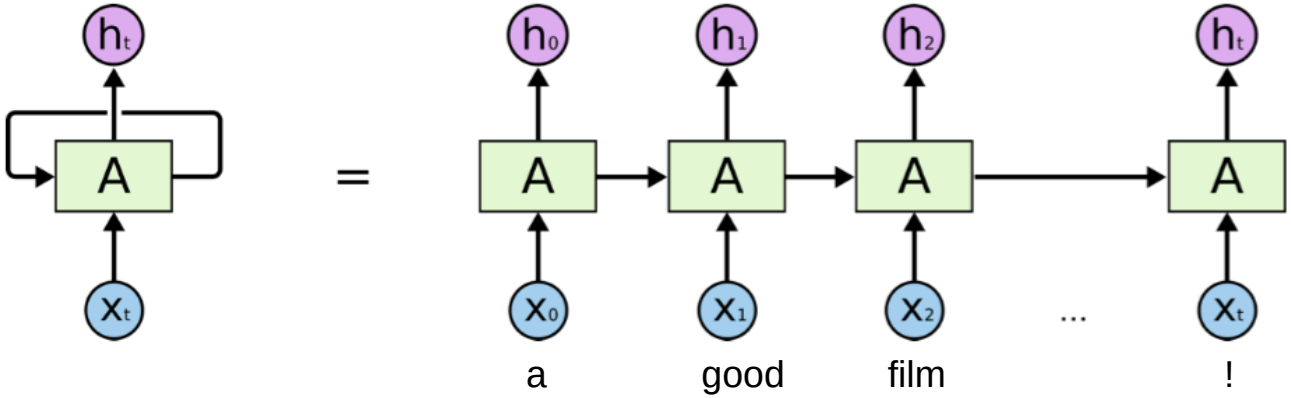


例如下面两句对网络来说是一样的：
1. 特朗普的儿子是谁？
2. 谁的儿子是特朗普？



循环神经网络(Recurrent Neural Network，RNN)

循环神经网络通过使用带自反馈的神经元，能够处理任意长度的序列。

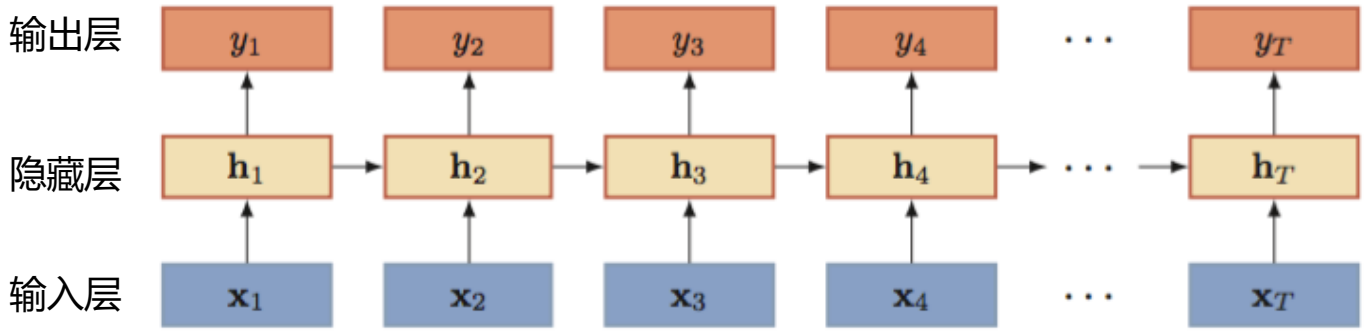


- 展开的各个单元是同一个A，使用同一套参数
- 循环的次数等于输入序列的长度，因此可处理任意长度的数据
- 后面数据的处理使用前面数据的相关状态，因此能利用上下文信息

简单循环神经网络SimpleRNN

- 假设输入是$\mathbf{x_1}$、$\mathbf{x_2}$……，$\mathbf{x_t}$，分别是序列数据的一个元素，依次输入
- t时刻隐层状态为 $\mathbf{h_t}$，$\mathbf{h_t}$ 不仅和当前输入相关，也和上一个时刻的隐层状态相关。
- 一般我们使用如下函数：

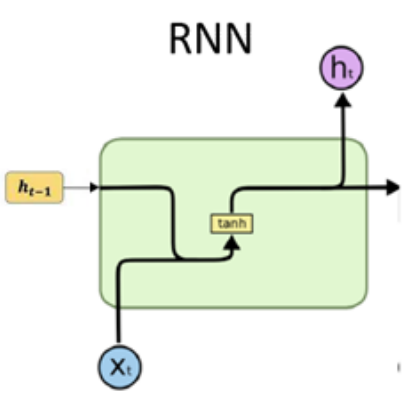
$$\boldsymbol{h}_t = f(\boldsymbol{U}\boldsymbol{h}_{\boldsymbol{t-1}} + \boldsymbol{W}\boldsymbol{x}_{\boldsymbol{t}} + b)$$ *f* 是非线性函数，通常为 *sigmod*函数或 *tanh* 函数

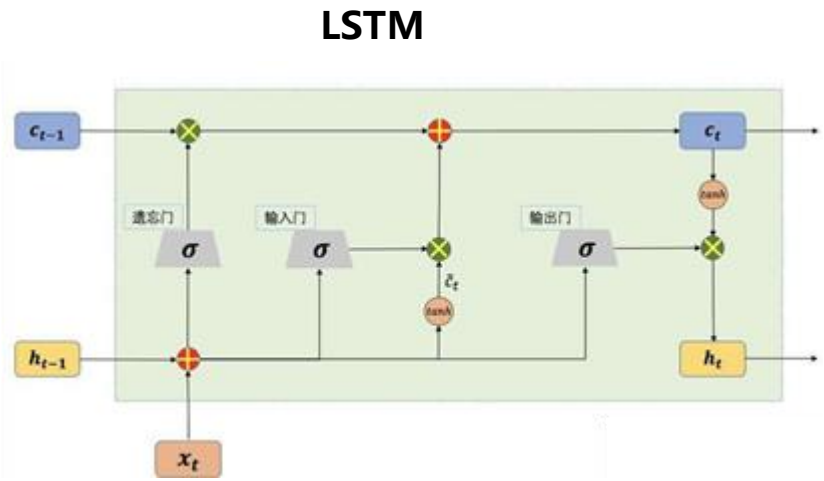


SimpleRNN虽然能处理时序数据，但由于过于简单，无法记住很多时刻之前的信息，存在梯度消失问题。

长短时记忆网络(Long short-term menory，LSTM)

在SimpleRNN基础上，增加一个携带轨道，可以携带信息跨越多个时间步，解决早期信号无法传递和梯度消失问题。





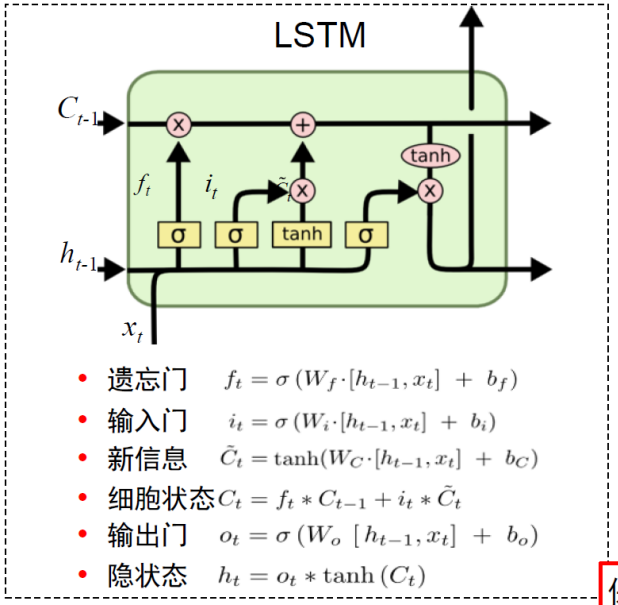
LSTM 模型引入了一组**记忆单元**(Memory Units)，允许网络可以学习何时遗忘历史信息，何时用新信息更新记忆单元。在时刻 t 时，记忆单元 $\mathbf{c}_t$ 记录了到当前时刻为止的所有历史信息，并受**三个“门”控制**：输入门 $\mathbf{i}_t$，遗忘门$\mathbf{f}_t$和输出门$\mathbf{o}_t$ 。三个门的输出值都在 [0, 1] 之间。

门限循环单元 (Gated Recurrent Unit，GRU)

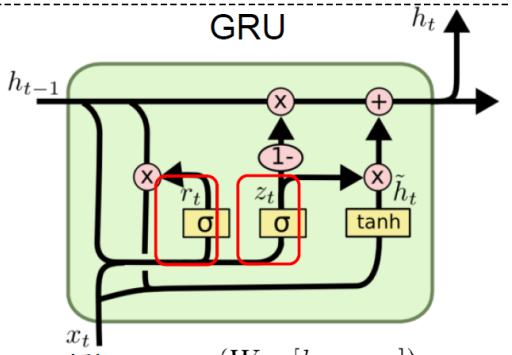
一种比 LSTM 简化的网络。GRU 将输入门与和遗忘门合并成一个门：更新门（Update Gate），同时还合并了记忆单元和隐藏神经元。

- 有单独的细胞状态
- 用输入门和遗忘门决定保留或放弃
- 新信息 $C_t^{\sim}$ 来源于 h_{t-1} 和 x_t
- 输出门控制细胞状态的输出

- 没有单独的细胞状态
- 用更新门决定保留或放弃
- $h_t^{\sim}$ 由重置门决定来自 h_{t-1} 的信息
- 直接输出隐状态



- 遗忘门 $f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right)$
- 输入门 $i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right)$
- 新信息 $\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$
- 细胞状态 $C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$
- 输出门 $o_t = \sigma\left(W_o\ [h_{t-1}, x_t] + b_o\right)$
- 隐状态 $h_t = o_t * \tanh\left(C_t\right)$



- 更新门 $z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$
- 重置门 $r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$
- 新信息 $\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$
- 隐状态 $h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$

保留哪些旧状态　　接收哪些新状态

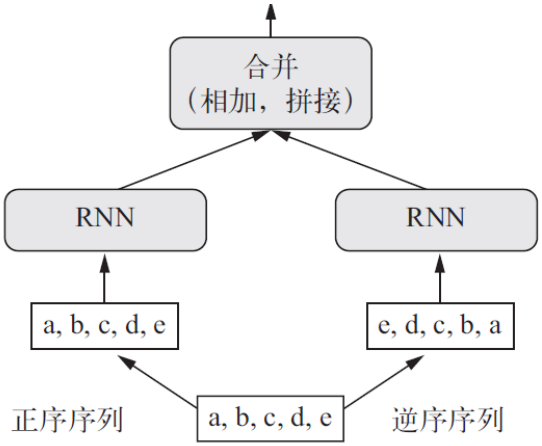
双向循环网络(Bidirectional RNN)

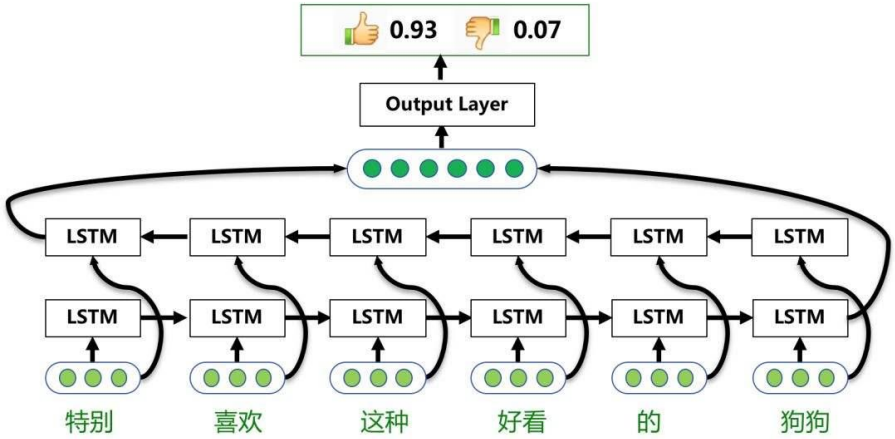
自然语言单词在句子中位置很重要，不仅体现在正序上，逆序也很重要。

例如：我今天不舒服，我打算_____一天。

去医院?　　睡觉?　　请假?

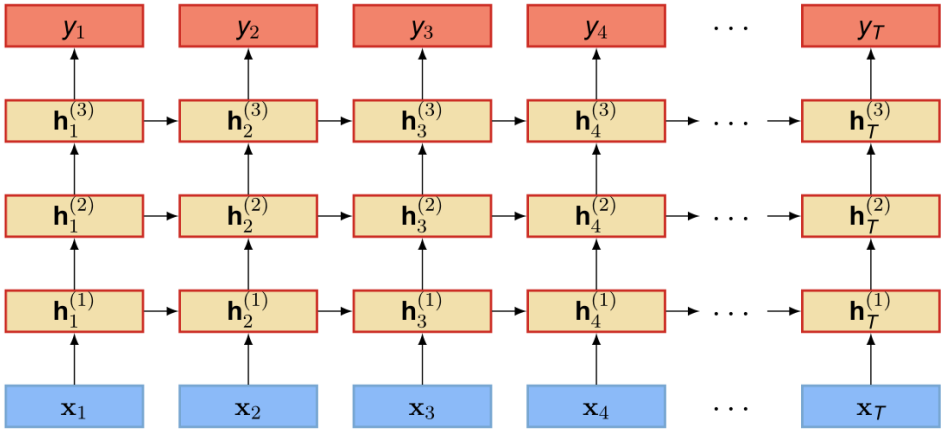
双向循环网络对输入数据进行正向和方向两次RNN处理，合并两次结果。





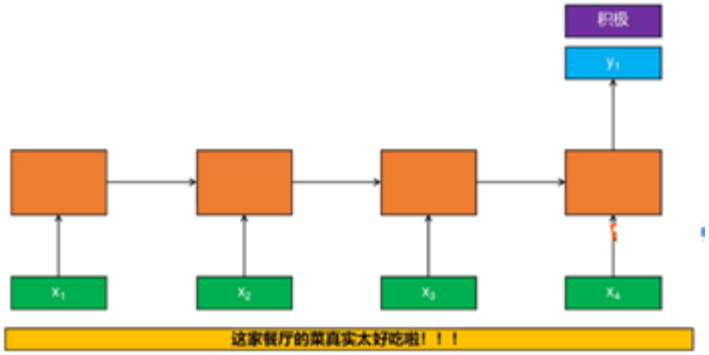
堆叠循环网络

通过添加多个RNN层形成深度循环神经网络。

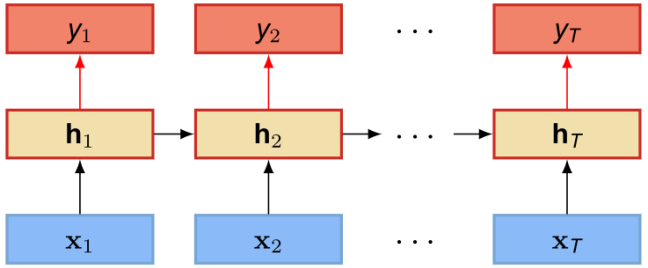


这里的RNN可以是SimpleRNN、LSTM、GRU、BiLSTM等各种类型的层，增加层数可以增加网络容量，提高模型性能。

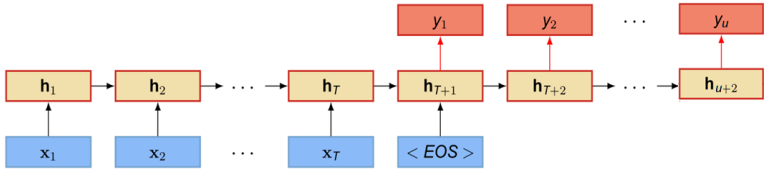
RNN用于文本处理



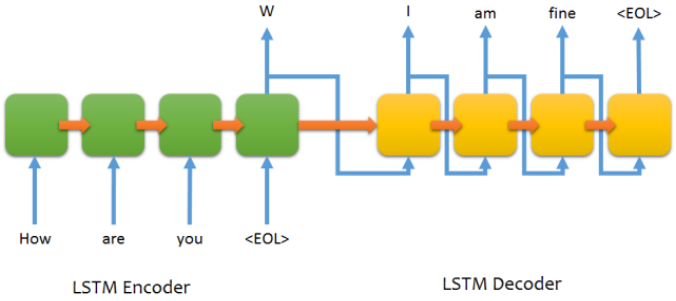
文本分类
输入：单词序列
输出：类别标签



序列标注
输入：单词序列
输出：标注序列



机器翻译
输入：源语言文本序列
输出：目标语言文本序列

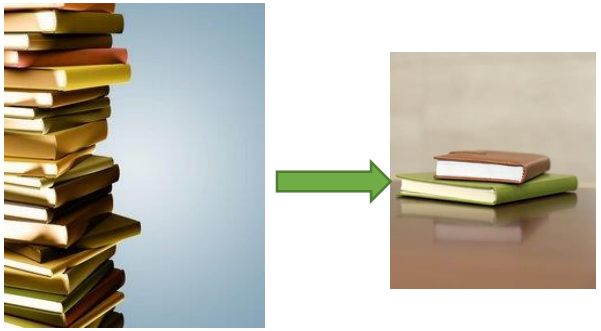


对话系统
输入：问句文本序列
输出：回答文本序列

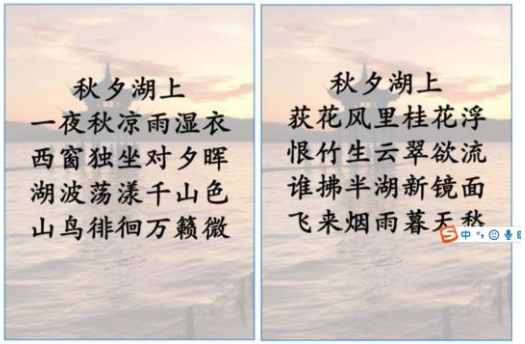
RNN用于文本处理

"Two people are walking down at river in a wooded area"

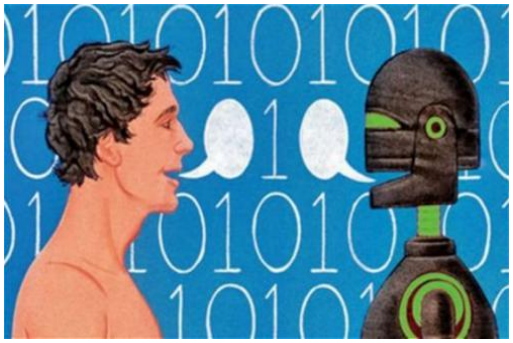
看图说话
输入：图片
输出：文本序列



自动摘要
输入：长文本序列
输出：短文本序列



自动写诗
输入：第一句、关键词或标题等文本序列
输出：完整的一首诗



对话系统
输入：问句或上句
输出：答句或下局

循环神经网络 小结

循环神经网络在序列的演进方向进行递归且所有节点按链式连接，具有短期记忆能力。适合处理序列数据，常用于自然语言处理。

- SimpleRNN过于简单，不擅长处理长序列。
- LSTM增加一个携带轨道，能实现对早期信号的传递。对机器翻译、问答等困难问题都很有效。
- 双向RNN从两个方向处理同一个序列，对自然语言处理非常有效。但如果序列相邻数据比两头包含更多信息，效果不明显。
- 堆叠RNN可以提高网络表示能力，但计算代价高。在机器翻译等复杂应用效果好，较小和简单问题不一定有用。

基于Keras实现循环神经网络

Python支持的RNN实现

Keras.layers中提供了多种RNN层，可堆叠到神经网络中。

```
from keras.layers import SimpleRNN,LSTM,GRU,Bidirectional
```

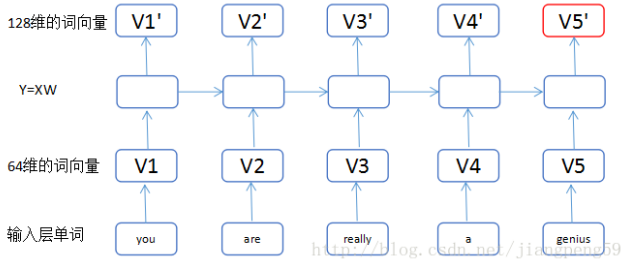
层实例初始化：

LSTM(units, input_dim, return_sequences,dropout,recurrent_dropout)

units: 正整数，输出空间的维度
input_dim: 输入序列的维度，如果是网络的第一层需要说明input_shape
return_sequences: 布尔值，默认False输出最后一个输出，True表示输出完整序列
dropout: 输入的失活比例
recurrent_dropout: 循环状态的失活比例

例如：

LSTM(128, input_dim=64,return_sequences=True)



Bidirectional(双向层实例)

实例化的各种RNN层。例：Bidirectional(LSTM(32))

案例3：1-使用LSTM实现电影评论分类

1. 读取数据并进行预处理

```
#数据准备
from keras.datasets import imdb
from keras.preprocessing import sequence
max_features = 10000  #最大特征数，即该训练集最常见的前10000个单词
maxlen = 20      #每条评论最大长度20，超过将被截断
#加载数据集，获得各评论文本对应的整数列表
(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=max_features)
print(x_train[:1,])
#整数列表填充处理：即超长截断、不足则补0，默认从尾部截取，得到等长二维整数张量(samples,maxlen)
x_train =sequence.pad_sequences(x_train, maxlen=maxlen)
x_test = sequence.pad_sequences(x_test, maxlen=maxlen)
print(x_train[:1,])
```

第一条评论的索引数据序列, 有218个元素

[list([1, 14, 22, 16, 43, 530, 973, 1622, 1385, 65, 458, 4468, 66, 3941, 4, 173, 36, 256, 5, 25, 100, 43,

…….

25, 104, 4, 226, 65, 16, 38, 1334, 88, 12, 16, 283, 5, 16, 4472, 113, 103, 32, 15, 16, 5345, 19, 178, 32])]

处理后的第一条评论的索引整数序列，从后截取500个整数, 不足补0

[[65 16 38 1334 88 12 16 283 5 16 4472 113 103 32 15 16 5345 19 178 32]]

2. 建立和训练神经网络，在分类器前添加一个LSTM层

```
#建立和训练神经网络
from keras.models import Sequential
from keras.layers import Dense,LSTM
model = Sequential()
model.add(Embedding(max_features, 32))
model.add(LSTM(32))   #输出数据维度32
model.add(Dense(1, activation='sigmoid'))
model.summary()

model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['acc'])
history = model.fit(x_train, y_train,
                    epochs=10,
                    batch_size=128,
                    validation_split=0.2)
```

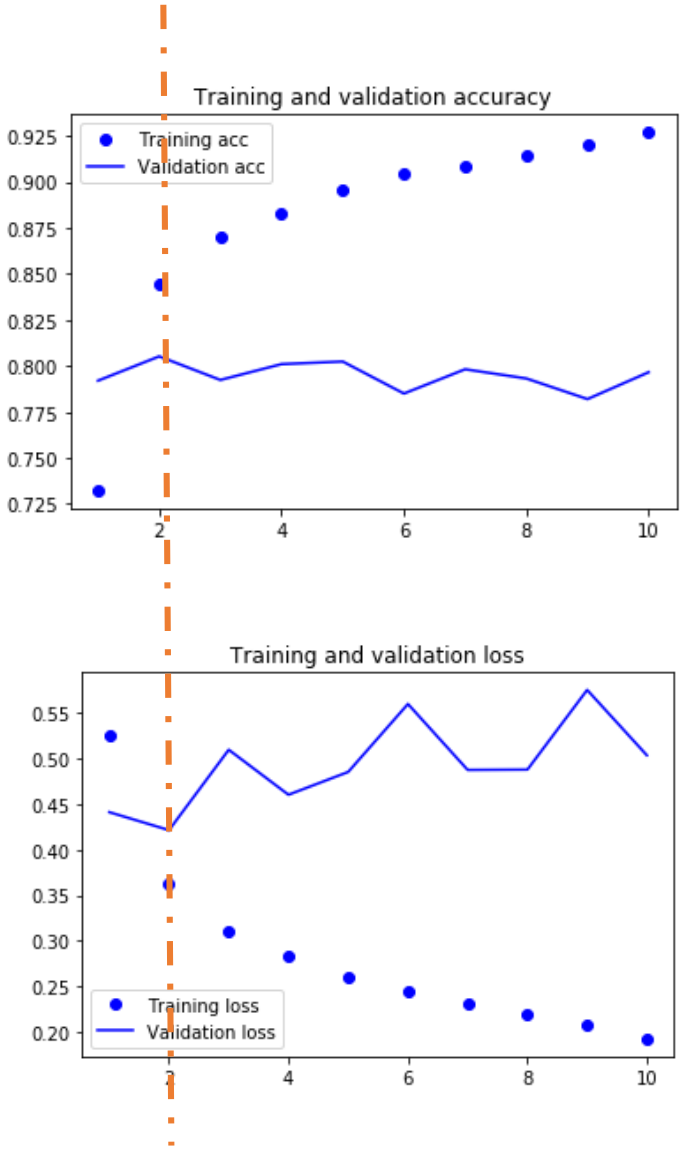
```
Model.summary()
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1(Embedding)       (None, None, 32)          320000
_________________________________________________________________
flatten_1(LSTM)              (None, 32)                8320
_________________________________________________________________
dense_1(Dense)               (None, 1)                 33
=================================================================
Total params: 328,353
Trainable params: 328,353
Non-trainable params: 0
```

最后一轮：
val_loss: 0.3551 - val_acc: 0.8104

3. 绘图观察训练过程

```
#绘图观察训练过程
import matplotlib.pyplot as plt
acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(acc) + 1)

plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.legend()
plt.figure()
plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()
plt.show()
```



案例3：2-使用堆叠LSTM实现电影评论分类

```
#建立和训练神经网络
from keras.models import Sequential
from keras.layers import Dense,LSTM
model = Sequential()
model.add(Embedding(max_features, 32))
model.add(LSTM(32,dropout=0.1,return_sequences=True))  #输出全部序列
model.add(LSTM(32,dropout=0.1,return_sequences=True))  #输出全部序列
model.add(LSTM(32,dropout=0.1))     #注意最后一个LSTM只输出一个输出
model.add(Dense(1, activation='sigmoid'))
model.summary()

model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['acc'])
history = model.fit(x_train, y_train,
                    epochs=10,
                    batch_size=128,
                    validation_split=0.2)
```

最后一轮：
val_loss: 0.3508 - val_acc: 0.8822

案例3：3-使用BiLSTM实现电影评论分类

```
#建立和训练神经网络,在分类器前堆叠一个BiLSTM层
from keras.models import Sequential
from keras.layers import Dense,LSTM,Bidirectional
model = Sequential()
model.add(Embedding(max_features, 32))
model.add(Bidirectional(LSTM(32)))
model.add(Dense(1, activation='sigmoid'))
model.summary()

model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['acc'])
history = model.fit(x_train, y_train,
                    epochs=10,
                    batch_size=128,
                    validation_split=0.2)
```

最后一轮：
val_loss: 0.3551 - val_acc: 0.89

实现循环神经网络 小结

适合处理序列数据，常用于自然语言处理。

Keras.layers中提供了SimpleRNN、LSTM、GRU、Bidirectional等多种RNN层，可以堆叠到神经网络中。

可以先尝试建立一个基准模型，然后逐步提高。可以尝试：

- 调节优化器的学习率
- 在循环层上用更大的Dense层或堆叠Dense层
- 使用Dropout降低过拟合
- 堆叠循环层及每层单元数

计算代价也是需要考虑的一个因素。

实验

作业

- 在超星平台完成单元测验题

实验

1. 模仿案例1使用词嵌入进行电影评论分类。对比采用不同的样本长度、嵌入维度的模型性能。
2. 模仿案例3-3实现使用BiLSTM实现电影评论分类。

注意：本次作业需在Jupyter Notebook或Spyder下完成，在超星作业中提交。
1）程序命名“学号姓名_RNN_n.py”，n为题目号。
2）代码对应你最高模型性能。另外，注意为关键语句增加注释。
3）代码最后用注释说明实验过程、模型性能分析、以及对模型实际意义的解释。

THANK YOU!