

实验四：自然语言理解

一、实验目的

- 1、编写 Python 代码实现原始数据加载的功能。
- 2、编写 Python 代码实现 Tokenize 功能。
- 3、编写 Python 代码实现 padding 功能。
- 4、编写 Python 代码搭建一个双向 RNN 神经网络
- 5、编写 Python 代码，实现机器翻译功能。

二、实验内容

- 1、编写 Python 代码实现原始数据加载的功能。
- 2、编写 Python 代码实现 Tokenize 功能。
- 3、编写 Python 代码实现 padding 功能。
- 4、编写 Python 代码搭建一个双向 RNN 神经网络
- 5、编写 Python 代码，实现机器翻译功能。

三、实验步骤

- 1、编写 Python 代码实现原始数据加载的功能。

原始数据非常简单，分为两个文件，一个是存放英语文本的 small_vocab_en.txt, 另一个是存放对应的法语翻译的文本文件 small_vocab_fr.txt。

每个文件中的每一行都是对应的翻译文本。如下图所示(第一张图是英文语料的部分截图，第二张图是法语语料的部分截图)：

```
new jersey is sometimes quiet during autumn , and it is snowy in april .
the united states is usually chilly during july , and it is usually freezing in november .
california is usually quiet during march , and it is usually hot in june .
the united states is sometimes mild during june , and it is cold in september .
your least liked fruit is the grape , but my least liked is the apple .
his favorite fruit is the orange , but my favorite is the grape .
paris is relaxing during december , but it is usually chilly in july .
new jersey is busy during spring , and it is never hot in march .
our least liked fruit is the lemon , but my least liked is the grape .
the united states is sometimes busy during january , and it is sometimes warm in november .
the lime is her least liked fruit , but the banana is my least liked .
he saw a old yellow truck .
india is rainy during june , and it is sometimes warm in november .
that cat was my most loved animal .
he dislikes grapefruit , limes , and lemons .
her least liked fruit is the lemon , but his least liked is the grapefruit .
california is never cold during february , but it is sometimes freezing in june .
china is usually pleasant during autumn , and it is usually quiet in october .
```

```
new jersey est parfois calme pendant l' automne , et il est neigeux en avril .  
les états-unis est généralement froid en juillet , et il gèle habituellement en novembre .  
california est généralement calme en mars , et il est généralement chaud en juin .  
les états-unis est parfois légère en juin , et il fait froid en septembre .  
votre moins aimé fruit est le raisin , mais mon moins aimé est la pomme .  
son fruit préféré est l'orange , mais mon préféré est le raisin .  
paris est relaxant en décembre , mais il est généralement froid en juillet .  
new jersey est occupé au printemps , et il est jamais chaude en mars .  
notre fruit est moins aimé le citron , mais mon moins aimé est le raisin .  
les états-unis est parfois occupé en janvier , et il est parfois chaud en novembre .  
la chaux est son moins aimé des fruits , mais la banane est mon moins aimé.  
il a vu un vieux camion jaune .  
inde est pluvieux en juin , et il est parfois chaud en novembre .  
ce chat était mon animal le plus aimé .  
il n'aime pamplemousse , citrons verts et les citrons .  
son fruit est moins aimé le citron , mais son moins aimé est le pamplemousse .  
californie ne fait jamais froid en février , mais il est parfois le gel en juin .  
chine est généralement agréable en automne , et il est généralement calme en octobre .  
paris est jamais le gel en novembre , mais il est merveilleux en octobre .  
les états-unis est jamais pluvieux en janvier , mais il est parfois doux en octobre .  
chine est généralement agréable en novembre , et il est jamais tranquille en octobre .
```

加载原始数据

既然想要实现将英语翻译成法语的功能，那么第一步肯定是加载原始数据。把硬盘中的数据加载到内存当中。使用 Python 读取文件中的数据非常简单，只需使用如下示例代码：

```
1. with open('data.txt', 'r') as f:  
2.     raw_data = f.read()
```

编程要求

实现 load_data 函数，该函数需要加载 path 所代表的文件中的数据，并将文件中所有的内容按\n 分割，转换成一个列表后返回。

测试说明

平台会对你编写的代码进行测试：

测试输入：

无

预期输出：

加载原始语料成功

```
1. #coding:utf8  
2. import os  
  
3.  
  
4.  
  
5. def load_data(path):  
6.     ...
```

```

7. 读取原始语料数据
8. :param path: 文件路径
9. :return: 句子列表, 如['he is a boy.', 'she is a girl']
10. ...
11. #*****Begin*****#
12.
13. #*****End*****#

```

2、编写 Python 代码实现 Tokenize 功能。

对于深度学习和机器学习程序来说，除了数字之外，什么都不认识。也就是说假设现在有一个机器翻译的程序，能将英语翻译成法语，然后我们把英语句子作为输入，输入到程序中。此时程序其实会将句子中的单词、符号等信息转换成数字，再丢给模型去计算。所以将单词转换成数字是实现机器翻译的第一步，而 Tokenize 就是最为常用的一种将单词转换成数字的方式。

Tokenize 看起来高大上，其实就是将每个单词都映射成一个数字而已。例如现在语料库中的句子如下：

```

1. ['The quick brown fox jumps over the lazy dog .', 'By Jove , my quick
2. study of lexicography won a prize .', 'This is a short sentence .']

```

那么 Tokenize 就是统计所有句子中出现的不同的词，然后将每个单词与一个数字对应起来。如：

```

1. {'the': 1, 'quick': 2, 'a': 3, 'brown': 4, 'fox': 5, 'jumps': 6,
2. 'over': 7, 'lazy': 8, 'dog': 9, 'by': 10, 'jove': 11, 'my': 12,
3. 'study': 13, 'of': 14, 'lexicography': 15, 'won': 16, 'prize': 17,
4. 'this': 18, 'is': 19, 'short': 20, 'sentence': 21}

```

所以经过 Tokenize 后，语料库中的句子变成了只有数字的列表，如下所示：

```

1. [[1, 2, 4, 5, 6, 7, 1, 8, 9], [10, 11, 12, 2, 13, 14, 15, 16, 3, 17],
2. [18, 19, 3, 20, 21]]

```

怎样实现 Tokenize

keras 为我们已经实现好了 Tokenize 功能，我们只需调用接口即可实现 Tokenize 。示例代码如下：

```

1. from keras.preprocessing.text import Tokenizer

```

```

2.     # 语料
3.     x = ['The quick brown fox jumps over the lazy dog .', 'By Jove , my quick
study of lexicography won a prize .', 'This is a short sentence .']
4.     # 实例化 Tokenizer 对象，使用单词级别的 Tokenize
5.     x_tk = Tokenizer(char_level=False)
6.     # 对语料进行 Tokenize
7.     x_tk.fit_on_texts(x)
8.     # 打印 Tokenize 的语料
9.     print(x_tk.texts_to_sequences(x))

```

编程要求

实现 tokenize 函数。

测试说明

平台会对你编写的代码进行测试：

测试输入：

```

1.     ['The quick brown fox jumps over the lazy dog .', 'By Jove , my quick
2.     study of lexicography won a prize .', 'This is a short sentence .']

```

预期输出：

```

1. [[1, 2, 4, 5, 6, 7, 1, 8, 9], [10, 11, 12, 2, 13, 14, 15, 16, 3, 17],
2. [18, 19, 3, 20, 21]]

```

```

1.     #coding:utf8
2.     from keras.preprocessing.text import Tokenizer
3.
4.
5.     def tokenize(data):
6.         '''
7.         tokenize
8.         :param data: 语料，类型为list
9.         :return: tokenize 后的语料，类型为list
10.        '''
11.        #*****Begin*****#
12.
13.        #*****End*****#

```

3、编写 Python 代码实现 padding 功能。

通常情况下语料库中的每个句子中的单词数量不可能会是完全一致的。就比如上一关中用来举例的语料中 3 条句子的单次数量是不同的。但是在搭建神经网络

络时，神经网络中每层的神经元的个数其实就已经确定下来了。特别是输入层的神经元数量，每一个神经元代表着一个词。一边是单次数量不确定，另一边是需要确定好单词数量，怎么办呢？这个时候可以使用 padding 方法。

padding 其实就是设定一个最大的单词数量，当 tokenize 后的句子中单词数量小于最大单词数量时，就在后面补 0。

假设 tokenize 后的句子为:[18 19 3 20 21]，最大单词数量为 10，那么 padding 后的句子为:[18 19 3 20 21 0 0 0 0 0]。

keras 同样为我们实现好了 padding 功能，省得我们重新造轮子。示例代码如下：

```
1. from keras.preprocessing.sequence import pad_sequences
2. # tokenize 后的语料
3. x = [[18 19 3 20 21]]
4. # 打印 padding 的结果，最大单词数量为 10
5. print(pad_sequences(x, maxlen=10, padding='post'))
```

编程要求

实现 padding 函数。注意：最大单次数量为 data 中所有句子的单次数量的最大值。

测试说明

平台会对你编写的代码进行测试：

测试输入：

```
1. [[1, 2, 4, 5, 6, 7, 1, 8, 9], [10, 11, 12, 2, 13, 14, 15, 16, 3, 17],
2. [18, 19, 3, 20, 21]]
```

预期输出：

```
1. Using TensorFlow backend.
2. [[ 1  2  4  5  6  7  1  8  9  0]
3. [10 11 12  2 13 14 15 16  3 17]
4. [18 19  3 20 21  0  0  0  0  0]]
```

```
1. #coding:utf8
2. from keras.preprocessing.sequence import pad_sequences
3.
4.
```

```

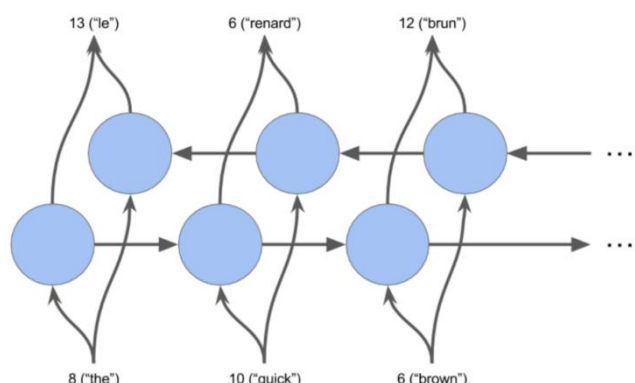
5.  def padding(data):
6.  ...
7.  padding, 最大单词数量为 data 中所有句子的单词数量的最大值
8.  :param data: 语料, 类型为 list
9.  :return: padding 后的语料, 类型为 list
10. ...
11. #####Begin#####
12.
13. #####End#####

```

4、编写 Python 代码搭建一个双向 RNN 神经网络

双向 RNN

为什么需要既能从前往后又能从后往前的 RNN 呢，因为如果仅仅是从前往后的方式来理解句子的语义的话，会有可能产生歧义。例如：“He said, Teddy bears are on sale” 和 “He said, Teddy Roosevelt was a great President。在上面的两句话中，当我们看到 “Teddy” 和前两个词 “He said” 的时候，我们有可能无法理解这个句子是指 President 还是 Teddy bears。因此，为了解决这种歧义性，我们需要从后往前看。这就是双向 RNN 所能实现的。



怎样搭建双向 RNN 神经网络

以将英语翻译成法语的功能为例。当我们把英语和法语的语料进行 `tokenize` 和 `padding` 处理后，就可以将经过处理后的英语语料作为伸进网络的输入，将经过处理后的法语语料作为输出。所以这个网络应该是一个处理分类问题的神经网络。

因此，网络的输出层其实就是一个全连接层，神经元的数量就是法语 `tokenize` 后词的数量，激活函数是 `softmax`。

确定了输出层后，就要确定处理输入层数据的层了。很明显，就是用双向 RNN ！ `keras` 中有相应的接口来帮助我们实现双向 RNN 。示例代码如下：

```

1. # SimpleRNN 表示 RNN, Bidirectional 表示双向的意思。128 表示 RNN 有 128 个神经
   元, 由于我们需要 RNN 处理后的序列, 所以 return_sequences 为 True, dropout 是随机丢弃的
   概率, 能够防止过拟合
2. Bidirectional(SimpleRNN(128, return_sequences = True, dropout=0.1),
   input_shape=input_shape)

```

知道怎样构建双向 RNN 层之后, 相信你能够很轻松的构建出整个神经网络了。

编程要求

实现 build_model 函数, 该函数的功能是构建一个含有 128 个神经元的双向 RNN 层和含有 french_vocab_size 个神经元的全连接层的神经网络。(怎样将层与层之间连接起来, 请查阅 keras 官方文档。)

测试说明

平台会对你编写的代码进行测试:

测试输入:

无

预期输出:

```

1. Using TensorFlow backend.
2.
3. Layer (type)                Output Shape                Param #
4. =====
5. bidirectional_1 (Bidirection (None, 52, 256)            35072
6.
7. dense_1 (Dense)              (None, 52, 22)              5654
8. =====
9. Total params: 40,726
10. Trainable params: 40,726
11. Non-trainable params: 0
12.

```

阅读代码中注释部分要求, 补充实现 begin-end 间代码, 并运行测试结果。

```

1. #coding:utf8
2. from keras.models import Sequential, Model
3. from keras.layers import Input, Dense, SimpleRNN, Bidirectional
4.
5.
6. def build_model(input_shape, french_vocab_size):

```

```

7.     '''
8.     tokenize
9.     :param input_shape: 英语语料的 shape, 类型为 list
10.    :param french_vocab_size: 法语语料词语的数量, 类型为 int
11.    :return: 构建好的模型
12.    '''
13.    #*****Begin*****#
14.    #*****End*****#

```

5、编写 Python 代码，实现机器翻译功能。

编程要求

实现机器翻译功能。由于平台无法训练太久，所以这里只训练 10 个 epoch，若能达到比较好的翻译效果，可以在自己电脑上训练久一点。

```

1. #coding:utf8
2. import os
3. import warnings
4. warnings.filterwarnings('ignore')
5. os.environ["TF_CPP_MIN_LOG_LEVEL"] = "2"
6. from keras.models import Sequential, Model
7. from keras.layers import Input, Dense, SimpleRNN, Bidirectional,
TimeDistributed
8. from keras.preprocessing.sequence import pad_sequences
9. from keras.preprocessing.text import Tokenizer
10.    from keras.losses import sparse_categorical_crossentropy
11.    from keras.optimizers import Adam
12.    import numpy as np
13.
14.    def load_data(path):
15.        '''
16.        读取原始语料数据，只读取前 3000 条文本
17.        :param path: 文件路径
18.        :return: 句子列表，如['he is a boy.', 'she is a girl']
19.        '''
20.        with open(path, 'r') as f:
21.            return f.readlines()[:3000]
22.
23.
24.    def tokenize(data):
25.        '''
26.        tokenize
27.        :param data: 语料，类型为 list
28.        :return: (tokenize 后的语料,tokenizer 对象)

```

```

29.     '''
30.     x_tk = Tokenizer(char_level = False)
31.     x_tk.fit_on_texts(data)
32.     return x_tk.texts_to_sequences(data), x_tk
33.
34.
35.     def padding(data, length=None):
36.         '''
37.         padding, 最大单词数量为 dlength
38.         :param data: 语料, 类型为 list
39.         :param data: 词数量, 类型为 int
40.         :return: padding 后的语料, 类型为 list
41.         '''
42.         if length is None:
43.             length = max([len(sentence) for sentence in data])
44.         return pad_sequences(data, maxlen = length, padding = 'post')
45.
46.
47.
48.     def build_model(input_shape, french_vocab_size):
49.         '''
50.         tokenize
51.         :param input_shape: 英语语料的 shape, 类型为 list
52.         :param french_vocab_size: 法语语料词语的数量, 类型为 int
53.         :return: 构建好的模型
54.         '''
55.         model = Sequential()
56.         model.add(Bidirectional(SimpleRNN(128, return_sequences=True,
57. dropout=0.1), input_shape=input_shape[1:]))
58.         model.add(Dense(french_vocab_size, activation='softmax'))
59.         return model
60.
61.     def logits_to_text(logits, tokenizer):
62.         """
63.         将神经网络的输出转换成句子
64.         :param logits: 神经网络的输出
65.         :param tokenizer: 语料的 tokenizer
66.         :return: 神经网络的输出所代表的字符串
67.         """
68.         index_to_words = {id: word for word, id in
69. tokenizer.word_index.items()}
70.         index_to_words[0] = '<PAD>'

```

```

70.         return ' '.join([index_to_words[prediction] for prediction in
np.argmax(logits, 1)])
71.
72.
73.
74.     #*****Begin*****#
75.     #*****End*****#
76.
77.     # 编译神经网络
78.     model.compile(loss=sparse_categorical_crossentropy, optimizer =
Adam(1e-3))
79.     # 训练（由于平台原因，无法训练太久，这里只训练 10 个 epoch）
80.     model.fit(tmp_x, preproc_french_sentences, batch_size=512, epochs=10,
validation_split=0.2, verbose=0)
81.
82.     # 保存翻译结果
83.     with open('result.txt', 'w') as f:
84.         for i in range(10):
85.             result = english_sentences[i]+' -> ' +
logits_to_text(model.predict(np.expand_dims(tmp_x[i], 0))[0], french_tokenizer)
86.             f.write(result)

```

四、实验报告要求

参见实验报告模板