

CSDN新首页上线啦，邀请你来立即体验！(http://blog.csdn.net/)

立即体

验

**CSDN**

博客 (//blog.csdn.net?ref=toolbar) 学院 (//edu.csdn.net?ref=toolbar)

下载 (//download.csdn.net?ref=toolbar) GitChat (//gitbook.cn/?ref=csdn)

更多 

2



weixin\_3506... ▾

(//my.csdn.net?ref=toolbar)

(//write.blog.csdn.net/postedit?ref=toolbar)source=csdnblog

三

# 基于Attention Model的Aspect level文本情感分类---用Python+Keras实现



orlandowww (http://blog....)

+ 关注

(http://blog.csdn.net/orlandowww)

码云

未开通

(https://gite  
utm\_sourc

原创  
8

粉丝  
15

喜欢  
0

原创

2016年12月27日 17:49:02

标签：Attention (http://so.csdn.net/so/search/s.do?q=Attention&t=blog) /



情感分析 (http://so.csdn.net/so/search/s.do?q=情感分析&t=blog) /

python (http://so.csdn.net/so/search/s.do?q=python&t=blog) /

Keras (http://so.csdn.net/so/search/s.do?q=Keras&t=blog) /

深度学习 (http://so.csdn.net/so/search/s.do?q=深度学习&t=blog)

4961

## 他的最新文章

更多文章 (http://blog.csdn.net/orlandowww)

win10+64位 安装Theano并实现GPU加速 (http://blog.csdn.net/orlandowww/article/details/53313804)

文本分类的python实现-基于Xgboost算法 (http://blog.csdn.net/orlandowww/article/details/52967187)

文本分类的python实现-基于SVM算法 (http://blog.csdn.net/orlandowww/article/details/52966608)

## 1、关于aspect level的情感分析

给定一个句子和句子中出现的某个aspect，aspect-level 情感分析的目标是分析出这个句子在给定aspect上的情感倾向。

例如：great food but the service was dreadful! 在aspect “food”上，情感倾向为正，在aspect “service”上情感倾向为负。Aspect level的情感分析相对于document level来说粒度更细。

## 2、关于attention model

使用传统的神经网络模型能够捕捉背景信息，但是不能明确的区分对某个aspect更重要的上下文信息。为了解决这个问题，引入attention捕获对于判断不同aspect的情感倾向较重要的信息。

## 3、模型的示意图

Python读取csv的常用方法 (<http://blog.csdn.net/orlandowww/article/details/52870686>)

词性标注的python实现-基于平均感知机算法 (<http://blog.csdn.net/orlandowww/article/details/52744355>)

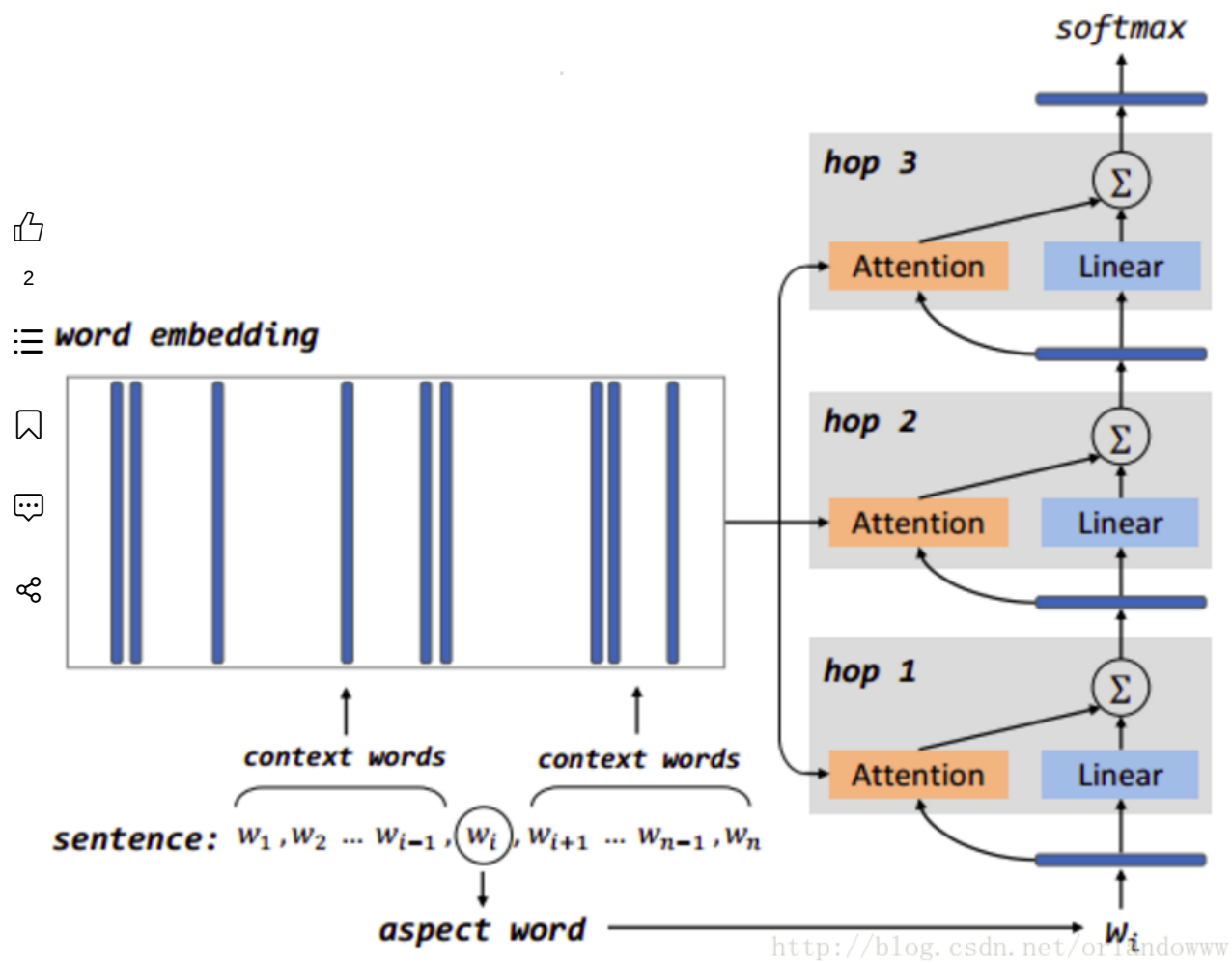
### 相关推荐

深度学习笔记——Attention Model (注意力模型) 学习总结 ([http://blog.csdn.net/mpk\\_no1/article/details/72862348](http://blog.csdn.net/mpk_no1/article/details/72862348))

attention 机制入门 (<http://blog.csdn.net/aliceyangxi1987/article/details/76284315>)

《Attention-based LSTM for Aspect-level Sentiment Classification》阅读笔记 ([http://blog.csdn.net/Together\\_CZ/article/details/73822295](http://blog.csdn.net/Together_CZ/article/details/73822295))

用于文本分类的RNN-Attention网络 ([http://blog.csdn.net/thriving\\_fcl/article/details/73381217](http://blog.csdn.net/thriving_fcl/article/details/73381217))



- 模型包括多个computational layers,每个computational layer包括一个attention layer和一个linear layer。
- 第一个computational layer, attention layer的输入是aspect vector, 输出memory中的比较重要的部分, linear layer的输入是aspect vector。第一个computational layer的attention layer和linear layer的输出结果求

腾讯云

新注册用户域名抢购1元起

.com首年28元 .cn首年9元

立即抢购

广告

70平米室内装修

他的热门文章

文本分类的python实现-基于SVM算法 (<http://blog.csdn.net/orlandowww/article/details/52966608>)

5272

基于Attention Model的Aspect level文本情感分类---用Python+Keras实现 (<http://blog.csdn.net/orlandowww/article/details/53897634>)

和作为下一个computational layer的输入。

- 其它computational layer执行同样的操作，上一层的输出作为输入，通过attention机制获取memory中较重要的信息，与线性层得到的结果求和作为下一层的输入。
- 最后一层的输出作为结合aspect信息的sentence representation，作为aspect-level情感分类的特征，送到softmax。

## 4、深度学习框架Keras

Keras是一个高层神经网络库，Keras由纯Python编写而成并基于Tensorflow或Theano。Keras 为支持快速实验而生，能够把idea迅速转换为结果。

具体可查看官方的中文文档 (<http://keras-cn.readthedocs.io/en/latest/>)写的很详细。

## 5、大概流程

读取数据集 -> 分词 -> 将所有词排序并标号 -> 词嵌入 -> Attention Model -> 编译 -> 训练+测试

## 6、参数设置

- 句子最大长度：80

[g.csdn.net/orlandowww/article/details/53897634](http://g.csdn.net/orlandowww/article/details/53897634)

4960

中文分词的python实现-基于HMM算法 (<http://blog.csdn.net/orlandowww/article/details/52706135>)

3420

文本分类的python实现-基于Xgboost算法 (<http://blog.csdn.net/orlandowww/article/details/52967187>)

2344

词性标注的python实现-基于平均感知机算法 (<http://blog.csdn.net/orlandowww/article/details/52744355>)

2030






- 词向量维度：300
- 梯度下降batch：32
- 总迭代次数：5

---

## 实验代码：

2





```
1  #-*- coding: utf-8 -*-
2  import csv
3  import jieba
4  jieba.load_userdict('wordDict.txt')
5  import pandas as pd
6  from keras.preprocessing import sequence
7  from keras.utils import np_utils
8  from keras.models import *
9  from keras.optimizers import *
10 from keras.layers.core import *
11 from keras.layers import Input,merge, TimeDistributed
12 from keras.layers.core import Dense, Dropout, Activation, Flatten
13 from keras.layers.embeddings import Embedding
14 from keras.regularizers import l2
15 from keras import backend as K
16
17 np.random.seed(1337)
18
19
20 # 读取训练集
21 def readtrain():
22     with open('allTrain_includeView.csv', 'rb') as csvfile:
23         reader = csv.reader(csvfile)
24         column1 = [row for row in reader]
25         content_train = [i[1] for i in column1[1:]]
26         view_train = [i[2] for i in column1[1:]]
27         opinion_train = [i[3] for i in column1[1:]]
28         print '训练集有 %s 条句子' % len(content_train)
29         train = [content_train, view_train, opinion_train]
30         return train
31
32
```

```
33 # 将utf8的列表转换成unicode
34 def changeListCode(b):
35     a = []
36     for i in b:
37         a.append(i.decode('utf8'))
38     return a
39
40 # 对列表进行分词用逗号连接
41 def segmentWord2(cont):
42     c = []
43     for i in cont:
44         a = list(jieba.cut(i))
45         c.append(a)
46     return c
47
48
49 def transLabel(labels):
50     for i in range(len(labels)):
51         if labels[i] == 'pos':
52             labels[i] = 2
53         elif labels[i] == 'neu':
54             labels[i] = 1
55         elif labels[i] == 'neg':
56             labels[i] = 0
57         else: print "label无效 :", labels[i]
58     return labels
59
60
61
62 # content = ["我 来到 北京 清华大学", "他 来到 了 网易 杭研 大厦"] sklearn输入格式
63 # content = [['我', '来到', '北京'], ['他', '来到', '了']] keras输入格式
64 train = readtrain()
```

```
65
66 content = segmentWord2(train[0]) #所有的词一起分词，包括训练集和预测集
67 view = changeListCode(train[1])
68 opinion = transLabel(train[2])
69
70
71
👍 72 w = [] # 将所有词语整合在一起
2 73 for i in content:
74     w.extend(i)
☰ 75 for i in view: # 把view的词也加入进去
76     w.append(i)
🔖 77
78
💬 79
80 def get_aspect(X):
🔗 81     ans = X[:, 0, :]
82     return ans
83
84 def get_context(X):
85     ans = X[:, 1:, :]
86     return ans
87
88 def get_R(X):
89     Y, alpha = X[0], X[1]
90     ans = K.T.batched_dot(Y, alpha)
91     return ans
92
93
94
95 # 参数设置
96 maxlen = 81
```



```
97 epochs = 5
98 batch = 32
99 emb = 300
100
101
102
103 print('Preprocessing...')
👍104 dict = pd.DataFrame(pd.Series(w).value_counts()) # 统计词的出现次数
2 105 del w
106 dict['id'] = list(range(1, len(dict) + 1))
⋮107 get_sent = lambda x: list(dict['id'][x])
108 sent = pd.Series(content).apply(get_sent)
📌109
110 for i in range(len(content)): # 在第一个位置插入view的值，每个句子的第一个词为视角
💬111     a = dict['id'][view[i]]
112     sent[i].insert(0,a)
🔗113
114 sent = list(sequence.pad_sequences(sent, maxlen=maxlen))
115
116
117 train_content = np.array(sent)
118 train_opinion = np.array(opinion)
119 train_opinion1 = np_utils.to_categorical(train_opinion, 3)
120
121
122
123 print('Build model...')
124
125 main_input = Input(shape=(maxlen,), dtype='int32', name='main_input')
126 x = Embedding(output_dim=emb, input_dim=len(dict)+1, input_length=maxlen, name='x')(main_input)
127 drop_out = Dropout(0.1, name='dropout')(x)
128 w_aspect = Lambda(get_aspect, output_shape=(emb,), name="w_aspect")(drop_out)
```

```

129 w_context = Lambda(get_context, output_shape=(maxlen-1,emb), name="w_context")(drop_out)
130
131 w_aspect = Dense(emb, W_regularizer=l2(0.01), name="w_aspect_1")(w_aspect)
132
133 # hop 1
134 w_aspects = RepeatVector(maxlen-1, name="w_aspects1")(w_aspect)
135 merged = merge([w_context, w_aspects], name='merged1', mode='concat')
136 distributed = TimeDistributed(Dense(1, W_regularizer=l2(0.01), activation='tanh'), name="distributed1")(merged)
137 flat_alpha = Flatten(name="flat_alpha1")(distributed)
138 alpha = Dense(maxlen-1, activation='softmax', name="alpha1")(flat_alpha)
139 w_context_trans = Permute((2, 1), name="w_context_trans1")(w_context)
140 r_ = merge([w_context_trans, alpha], output_shape=(emb, 1), name="r_1", mode=get_R)
141 r = Reshape((emb,), name="r1")(r_)
142 w_aspect_linear = Dense(emb, W_regularizer=l2(0.01), activation='linear')(w_aspect)
143 merged = merge([r, w_aspect_linear], mode='sum')
144
145 w_aspect = Dense(emb, W_regularizer=l2(0.01), name="w_aspect_2")(merged)
146
147 # hop 2
148 w_aspects = RepeatVector(maxlen-1, name="w_aspects2")(w_aspect)
149 merged = merge([w_context, w_aspects], name='merged2', mode='concat')
150 distributed = TimeDistributed(Dense(1, W_regularizer=l2(0.01), activation='tanh'), name="distributed2")(merged)
151 flat_alpha = Flatten(name="flat_alpha2")(distributed)
152 alpha = Dense(maxlen-1, activation='softmax', name="alpha2")(flat_alpha)
153 w_context_trans = Permute((2, 1), name="w_context_trans2")(w_context)
154 r_ = merge([w_context_trans, alpha], output_shape=(emb, 1), name="r_2", mode=get_R)
155 r = Reshape((emb,), name="r2")(r_)
156 w_aspect_linear = Dense(emb, W_regularizer=l2(0.01), activation='linear')(w_aspect)
157 merged = merge([r, w_aspect_linear], mode='sum')
158
159 w_aspect = Dense(emb, W_regularizer=l2(0.01), name="w_aspect_3")(merged)
160

```

```

161 # hop 3
162 w_aspects = RepeatVector(maxlen-1, name="w_aspects3")(w_aspect)
163 merged = merge([w_context, w_aspects], name='merged3', mode='concat')
164 distributed = TimeDistributed(Dense(1, W_regularizer=l2(0.01), activation='tanh'), name="distributed3")(merged)
165 flat_alpha = Flatten(name="flat_alpha3")(distributed)
166 alpha = Dense(maxlen-1, activation='softmax', name="alpha3")(flat_alpha)
167 w_context_trans = Permute((2, 1), name="w_context_trans3")(w_context)
168 r_ = merge([w_context_trans, alpha], output_shape=(emb, 1), name="r_3", mode=get_R)
169 r = Reshape((emb,), name="r3")(r_)
170 w_aspect_linear = Dense(emb, W_regularizer=l2(0.01), activation='linear')(w_aspect)
171 merged = merge([r, w_aspect_linear], mode='sum')
172
173 h_ = Activation('tanh')(merged)
174 out = Dense(3, activation='softmax')(h_)
175 output = out
176 model = Model(input=[main_input], output=output)
177
178
179
180 model.compile(loss='categorical_crossentropy',
181               optimizer=adam(), # 或者SGD(lr=0.03, momentum=0.9, nesterov=True)
182               metrics=['accuracy'])
183
184
185
186 print('Train...')
187 model.fit(train_content, train_opinion1,
188         batch_size=batch, nb_epoch=epochs,
189         validation_split=0.1)

```

## 实验输出：

```
Building prefix dict from the default dictionary ...
Loading model from cache c:\users\www\appdata\local\temp\jieba.cache
Loading model cost 0.476 seconds.
Prefix dict has been built succesfully.
Using Theano backend.
Using gpu device 0: GeForce GTX 860M (CNMeM is disabled, cuDNN not available)
训练集有 20595 条句子
Preprocessing...
Build model...
Train... http://blog.csdn.net/orlandowww
```

```
Train on 18535 samples, validate on 2060 samples
Epoch 1/5
18535/18535 [=====] - 164s - loss: 5.5290 - acc: 0.7271 - val_loss: 0.6031 - val_acc: 0.7699
Epoch 2/5
18535/18535 [=====] - 163s - loss: 0.4471 - acc: 0.8379 - val_loss: 0.6053 - val_acc: 0.7752
Epoch 3/5
18535/18535 [=====] - 161s - loss: 0.3171 - acc: 0.8883 - val_loss: 0.7354 - val_acc: 0.7621
Epoch 4/5
18535/18535 [=====] - 159s - loss: 0.2430 - acc: 0.9139 - val_loss: 0.7015 - val_acc: 0.7612
Epoch 5/5
18535/18535 [=====] - 160s - loss: 0.1979 - acc: 0.9280 - val_loss: 0.7578 - val_acc: 0.7597

Process finished with exit code 0 http://blog.csdn.net/orlandowww
```

注：

通过实验，发现单独使用CNN、LSTM、fasttext做文本情感分析的效果不如引入Attention的Model效果好，可见如今Attention如此火爆也是有原因的。



2



发表你的评论

[http://my.csdn.net/weixin\\_35068028](http://my.csdn.net/weixin_35068028)

xiatian19932011 (/xiatian19932011) 2017-12-11 20:41

3楼

(/xiatian19932011) 同问，博主可以分享一下您的数据集格式吗，想要复现一下实验~~博主啥时候能看到留言啊？



回复



SentiAnalysis (/SentiAnalysis) 2017-10-11 18:48

2楼

(/SentiAnalysis) 您好，看了您的文章，对attention有了一定的了解，您能分享一下您的数据集格式吗？不要全部的数据，您的数据可能保密的，我只是想知道一下数据的格式，这样可以用您的方法跑一下实验？

回复



qinhui99 (/qinhui99) 2017-07-11 07:55

1楼


(/qinhui99) 这篇文章写的不错，对我挺有帮助的。博主能共享一下数据吗？我想用CNN+双向LSTM训练一下做个对比测试，看看那种模型测试准确率更高。

回复

## 相关文章推荐


## 深度学习笔记——Attention Model（注意力模型）学习总结 ([http://blog.csdn.net/mpk\\_no1/a...](http://blog.csdn.net/mpk_no1/a...))

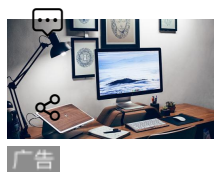
Attention Model（注意力模型）学习总结，包括soft Attention Model，Global Attention Model和Local Attention Model，静态AM，...

 mpk\_no1 ([http://blog.csdn.net/mpk\\_no1](http://blog.csdn.net/mpk_no1)) 2017年08月06日 21:49 7632

## attention 机制入门 (<http://blog.csdn.net/aliceyangxi1987/article/details/76284315>)

在下面这两篇文章中都有提到 attention 机制：使聊天机器人的对话更有营养 如何自动生成文章摘要今天来看看 attention 是什么。下面这篇论文算是在NLP中第一个使用attenti...

 aliceyangxi1987 (<http://blog.csdn.net/aliceyangxi1987>) 2017年07月28日 22:53 1655



### 票选结果：Python再上天，微软要求全员学Python？

宇宙语言Python荣登年度排行榜，吴恩达，微软纷纷为它站台，Python这么牛逼的原因是....

([http://www.baidu.com/cb.php?c=lgF\\_pyfqHmknjnvPjc0IZ0qnfK9ujYzP1nYPH0k0Aw-5Hc3rHnYnHb0TAq15HfLPWRznjb0T1dBmhF9ryfYuhckmvP-uAc40AwY5HDdnHfznjRkPWc0lgF\\_5y9YIZ0lQzq-uZR8mLPbUB48ugfEIAqspynETZ-YpAq8nWqdlAdxTvqdThP-5yF\\_UvTkn0KzujYk0AFV5H00TZcqn0KdpyfqHRLPjnvnfKEpyfqHc4rj6kP0KWpyfqP1cvrHnz0AqLUWYs0ZK45HcsP6KWThnqPW6vn6](http://www.baidu.com/cb.php?c=lgF_pyfqHmknjnvPjc0IZ0qnfK9ujYzP1nYPH0k0Aw-5Hc3rHnYnHb0TAq15HfLPWRznjb0T1dBmhF9ryfYuhckmvP-uAc40AwY5HDdnHfznjRkPWc0lgF_5y9YIZ0lQzq-uZR8mLPbUB48ugfEIAqspynETZ-YpAq8nWqdlAdxTvqdThP-5yF_UvTkn0KzujYk0AFV5H00TZcqn0KdpyfqHRLPjnvnfKEpyfqHc4rj6kP0KWpyfqP1cvrHnz0AqLUWYs0ZK45HcsP6KWThnqPW6vn6))

## 《Attention-based LSTM for Aspect-level Sentiment Classification》阅读笔记 (<http://blog...>)

《Attention-based LSTM for Aspect-level Sentiment Classification》阅读笔记 simple 7 个月前 转载请注明...

 Together\_CZ ([http://blog.csdn.net/Together\\_CZ](http://blog.csdn.net/Together_CZ)) 2017年06月27日 19:19 755

## 用于文本分类的RNN-Attention网络 ([http://blog.csdn.net/thriving\\_fcl/article/details/733812...](http://blog.csdn.net/thriving_fcl/article/details/733812...))

这篇博客主要介绍Attention机制在文本分类任务上的作用，原理以及附带的代码实现。...



thriving\_fcl ([http://blog.csdn.net/thriving\\_fcl](http://blog.csdn.net/thriving_fcl)) 2017年06月17日 15:51 2411

## 基于Theano的深度学习(Deep Learning)框架Keras学习随笔-17-Embedding Layers (<http://b...>)

基于Theano的深度学习(Deep Learning)框架Keras学习随笔-17-Embedding Layers -- 本篇介绍的内容主要用于NLP(Nature Language Processing)...



niuwei22007 (<http://blog.csdn.net/niuwei22007>) 2015年10月25日 20:29 15990



## 使用CNN进行文本分类 (<http://blog.csdn.net/xiewenbo/article/details/77874080>)

1. 卷积神经网络简介 卷积神经网络 (Convolutional Neural Network, CNN) 是一种前馈神经网络，它的人工神经元可以响应一部分覆盖范围内的周围单元，对于大型图像处理有出色表现...



xiewenbo (<http://blog.csdn.net/xiewenbo>) 2017年09月06日 23:08 855

## CNN字符级中文文本分类-基于TensorFlow实现 (<http://blog.csdn.net/u011439796/article/det...>)


本章旨在使用TensorFlow API实现卷积神经网络文本分类。代码地址：Github转载请注明出处：Gaussic - 写干净的代码基于CNN的文本分类问题已经有了一定的研究成果，CNN做句子分...



u011439796 (<http://blog.csdn.net/u011439796>) 2017年08月30日 01:29 2033

## 深度学习笔记——情感分析 ([http://blog.csdn.net/mpk\\_no1/article/details/71698725](http://blog.csdn.net/mpk_no1/article/details/71698725))

本文使用三种深度学习方法对IMDB评论进行情感分析。这三种方法为：MLP、BiRNN ( LSTM、GRU )、BiGRU+Attention。 ...

 mpk\_no1 ([http://blog.csdn.net/mpk\\_no1](http://blog.csdn.net/mpk_no1)) 2017年05月11日 22:14 1085


## python 文本情感分类 (<http://blog.csdn.net/momaojia/article/details/73555107>)

对于一个简单的文本情感分类来说，其实就是一个二分类，这篇博客主要讲述的是使用scikit-learn来做文本情感分类。分类主要分为两步：1) 训练，主要根据训练集来学习分类模型的规则。2) 分类，先用已知...

 momaojia (<http://blog.csdn.net/momaojia>) 2017年06月21日 19:54 740


## attention 机制 ([http://blog.csdn.net/qq\\_26609915/article/details/52086772](http://blog.csdn.net/qq_26609915/article/details/52086772))

attention 机制什么是attentionattention机制是（非常）松散地基于人类的视觉注意机制。就是按照“高分辨率”聚焦在图片的某个特定区域并以“低分辨率”感知图像的周边区域的模式，然后...

 qq\_26609915 ([http://blog.csdn.net/qq\\_26609915](http://blog.csdn.net/qq_26609915)) 2016年08月01日 16:24 5281

## [Elasticsearch] 全文搜索 ([http://blog.csdn.net/wang\\_zhenwei/article/details/50377955](http://blog.csdn.net/wang_zhenwei/article/details/50377955))


[Elasticsearch] 全文搜索 (一) - 基础概念和match查询 现在我们已经讨论了搜索结构化数据的一些简单用例，是时候开始探索全文搜索了 - 如何在全文字段中搜索来找到最相关的文档...

 wang\_zhenwei ([http://blog.csdn.net/wang\\_zhenwei](http://blog.csdn.net/wang_zhenwei)) 2015年12月22日 10:00 739

## Aspect-level Sentiment Classification using attention mechanism (<http://blog.csdn.net/g...>)


这篇论文其实是无意间发现的，其中运用到了attention的思想。其实当我发现机器翻译里可以引入attention机制后，我就一直在琢磨，文本的情感分类是不是也可以引入这种思想，但是不知道该如何用，该...



 guoyuhaoaaa (<http://blog.csdn.net/guoyuhaoaaa>) 2017年02月12日 21:42 1007

## 论文笔记：Two-level attention model for fine-grained Image classification (<http://blog.cs...>)


The Application of Two-level Attention Models in Deep Convolutional Neural Network for Fine-grained ...

 baidu\_17806763 ([http://blog.csdn.net/baidu\\_17806763](http://blog.csdn.net/baidu_17806763)) 2017年04月17日 15:59 1608

2

## 用python实现简单的文本情感分析 ([http://blog.csdn.net/github\\_37287822/article/details/74...](http://blog.csdn.net/github_37287822/article/details/74...))

情感分析+python

 github\_37287822 ([http://blog.csdn.net/github\\_37287822](http://blog.csdn.net/github_37287822)) 2017年07月06日 15:57 551




## 2.keras实现MNIST手写数字分类问题初次尝试（Python）(<http://blog.csdn.net/fuwenyan/ar...>)

根据我上一篇文章下载完MNIST数据集后，下一步就是看看keras是如何对它进行分类的。参考博客：<http://blog.csdn.net/vs412237401/article/detail...>

 fuwenyan (<http://blog.csdn.net/fuwenyan>) 2016年12月03日 21:31 3054

## 用深度学习（CNN RNN Attention）解决大规模文本分类问题 - 综述和实践（转载）(<http://blo...>)

近来在同时做一个应用深度学习解决淘宝商品的类目预测问题的项目，恰好硕士毕业时论文题目便是文本分类问题，趁此机会总结下文本分类领域特别是应用深度学习解决文本分类的相关的思路、做法和部分实践的经验。业...

 u013818406 (<http://blog.csdn.net/u013818406>) 2017年04月05日 21:42 2569

## 用深度学习（CNN RNN Attention）解决大规模文本分类问题 - 综述和实践（转载）(<http://blo...>)

本文转载自:<http://blog.csdn.net/u013818406/article/details/69359816> 近来在同时做一个应用深度学习解决淘宝商品的类目预测...



chivalrousli (<http://blog.csdn.net/chivalrousli>) 2017年08月07日 11:29 4226

## 用深度学习（CNN RNN Attention）解决大规模文本分类问题 - 综述和实践 (<http://blog.csdn.net/ch1209498273>)



转处: <https://zhuanlan.zhihu.com/p/25928551> 用深度学习（CNN RNN Attention）解决大规模文本分类问题 - 综述和实践 清淞 ?



xiewenbo (<http://blog.csdn.net/xiewenbo>) 2017年05月31日 20:29 1564



## 用深度学习（CNN RNN Attention）解决大规模文本分类问题 (<http://blog.csdn.net/ch1209498273>)



一、传统文本分类方法 文本分类问题算是自然语言处理领域中一个非常经典的问题了，相关研究最早可以追溯到上世纪50年代，当时是通过专家规则（Pattern）进行分类，甚至在80年代初一度发展到利用知识工...



ch1209498273 (<http://blog.csdn.net/ch1209498273>) 2017年11月05日 21:09 156

## 用深度学习（CNN RNN Attention）解决大规模文本分类问题 - 综述和实践 (<http://blog.csdn.net/hlang8160>)

转载自<https://zhuanlan.zhihu.com/p/25928551> 近来在同时做一个应用深度学习解决淘宝商品的类目预测问题的项目，恰好硕士毕业时论文题目便是文本分类问题，趁此机会总结...



hlang8160 (<http://blog.csdn.net/hlang8160>) 2017年11月20日 22:54 202