整理了25個Python文本處理案例，收藏！

原創 週蘿蔔 蘿蔔大雜燴 2022-02-07 18:30

收錄於話題
#python 32 #Python 9 #文本處理 1

Python 處理文本是一項非常常見的功能，本文整理了多種文本提取及NLP相關的案例，還是非常用心的

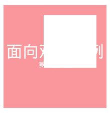
文章很長，高低要忍一下，如果忍不了，那就收藏吧，總會用到的

蘿蔔哥也貼心的做成了PDF，在文末獲取！

前情回顧：

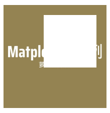
再再肝3天，整理了70個Python面向對象編程案例，怎能不收藏？

2021-11-19



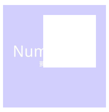
又再肝3天，整理了65個Matplotlib案例，這能不收藏？

2021-11-01



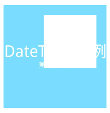
再肝3天，整理了90個NumPy案例，不能不收藏！

2021-10-27



又肝了3天，整理了80個Python DateTime 例子，必須收藏！

2021-10-20



肝了3天，整理了90個Pandas案例，強烈建議收藏！

2021-10-18

- 提取PDF 內容
- 提取Word 內容
- 提取Web 網頁內容
- 讀取Json 數據
- 讀取CSV 數據
- 刪除字符串中的標點符號
- 使用NLTK 刪除停用詞

- 使用TextBlob 更正拼寫
- 使用NLTK 和TextBlob 的詞標記化
- 使用NLTK 提取句子單詞或短語的詞幹列表
- 使用NLTK 進行句子或短語詞形還原
- 使用NLTK 從文本文件中查找每個單詞的頻率
- 從語料庫中創建詞云
- NLTK 詞法散佈圖
- 使用 countvectorizer 将文本转换为数字
- 使用 TF-IDF 创建文档术语矩阵
- 为给定句子生成 N-gram
- 使用带有二元组的 sklearn CountVectorize 词汇规范
- 使用 TextBlob 提取名词短语
- 如何计算词-词共现矩阵
- 使用 TextBlob 进行情感分析
- 使用 Goslate 进行语言翻译
- 使用 TextBlob 进行语言检测和翻译
- 使用 TextBlob 获取定义和同义词
- 使用 TextBlob 获取反义词列表

1 提取 PDF 内容

```
# pip install PyPDF2  安装 PyPDF2
import PyPDF2
from PyPDF2 import PdfFileReader

# Creating a pdf file object.
pdf = open("test.pdf", "rb")

# Creating pdf reader object.
pdf_reader = PyPDF2.PdfFileReader(pdf)

# Checking total number of pages in a pdf file.
print("Total number of Pages:", pdf_reader.numPages)

# Creating a page object.
page = pdf_reader.getPage(200)

# Extract data from a specific page number.
print(page.extractText())
```

```
# Closing the object.
pdf.close()
```

2 提取 Word 内容

```


# pip install python-docx  安装 python-docx



import docx



def main():
    try:
        doc = docx.Document('test.docx')  # Creating word reader object.
        data = ""
        fullText = []
        for para in doc.paragraphs:
            fullText.append(para.text)
            data = '\n'.join(fullText)

        print(data)

    except IOError:
        print('There was an error opening the file!')
        return



if __name__ == '__main__':
    main()
```

3 提取 Web 网页内容

```


# pip install bs4  安装 bs4


from urllib.request import Request, urlopen
from bs4 import BeautifulSoup


req = Request('http://www.cmegroup.com/trading/products/#sortField=oi&sortAsc=false&venues=3&page=1
              headers={'User-Agent': 'Mozilla/5.0'})
```

```
webpage = urlopen(req).read()

# Parsing
soup = BeautifulSoup(webpage, 'html.parser')

# Formating the parsed html file
strhtm = soup.prettify()

# Print first 500 lines
print(strhtm[:500])

# Extract meta tag value
print(soup.title.string)
print(soup.find('meta', attrs={'property':'og:description'}))

# Extract anchor tag value
for x in soup.find_all('a'):
    print(x.string)

# Extract Paragraph tag value
for x in soup.find_all('p'):
    print(x.text)
```

4 读取 Json 数据

```
import requests
import json

r = requests.get("https://support.oneskyapp.com/hc/en-us/article_attachments/202761727/example_2.js
res = r.json()

# Extract specific node content.
print(res['quiz']['sport'])

# Dump data as string
data = json.dumps(res)
print(data)
```

5 读取 CSV 数据

```
import csv

with open('test.csv','r') as csv_file:
    reader =csv.reader(csv_file)
    next(reader) # Skip first row
    for row in reader:
        print(row)
```

6 删除字符串中的标点符号

```
import re
import string

data = "Stuning even for the non-gamer: This sound track was beautiful!\
It paints the senery in your mind so well I would recomend\
it even to people who hate vid. game music! I have played the game Chrono \
Cross but out of all of the games I have ever played it has the best music! \
It backs away from crude keyboarding and takes a fresher step with grate\
guitars and soulful orchestras.\
It would impress anyone who cares to listen!"

# Methood 1 : Regex
# Remove the special charaters from the read string.
no_specials_string = re.sub('[!#?,.:";]', '', data)
print(no_specials_string)


# Methood 2 : translate()
# Rake translator object
translator = str.maketrans('', '', string.punctuation)
data = data.translate(translator)
print(data)
```

7 使用 NLTK 删除停用词

```
from nltk.corpus import stopwords
```

```
data = ['Stuning even for the non-gamer: This sound track was beautiful!\
It paints the senery in your mind so well I would recomend\
it even to people who hate vid. game music! I have played the game Chrono \
Cross but out of all of the games I have ever played it has the best music! \
It backs away from crude keyboarding and takes a fresher step with grate\
guitars and soulful orchestras.\
It would impress anyone who cares to listen!']

# Remove stop words
stopwords = set(stopwords.words('english'))

output = []
for sentence in data:
    temp_list = []
    for word in sentence.split():
        if word.lower() not in stopwords:
            temp_list.append(word)
    output.append(' '.join(temp_list))


print(output)
```

8 使用 TextBlob 更正拼写

```
from textblob import TextBlob

data = "Natural language is a cantral part of our day to day life, and it's so antresting to work o

output = TextBlob(data).correct()
print(output)
```

9 使用 NLTK 和 TextBlob 的词标记化

```
import nltk
from textblob import TextBlob
```

```
data = "Natural language is a central part of our day to day life, and it's so interesting to work

nltk_output = nltk.word_tokenize(data)
textblob_output = TextBlob(data).words

print(nltk_output)
print(textblob_output)
```

Output:

```
['Natural', 'language', 'is', 'a', 'central', 'part', 'of', 'our', 'day', 'to', 'day', 'life', ',',
['Natural', 'language', 'is', 'a', 'central', 'part', 'of', 'our', 'day', 'to', 'day', 'life', 'and
```

10 使用 NLTK 提取句子单词或短语的词干列表

```
from nltk.stem import PorterStemmer

st = PorterStemmer()
text = ['Where did he learn to dance like that?',
        'His eyes were dancing with humor.',
        'She shook her head and danced away',
        'Alex was an excellent dancer.']

output = []
for sentence in text:
    output.append(" ".join([st.stem(i) for i in sentence.split()]))

for item in output:
    print(item)

print("-" * 50)
print(st.stem('jumping'), st.stem('jumps'), st.stem('jumped'))
```

Output:

```
where did he learn to danc like that?
hi eye were danc with humor.
she shook her head and danc away
alex wa an excel dancer.
```

```
--------------------------------------------------
jump jump jump
```

11 使用 NLTK 进行句子或短语词形还原

```
from nltk.stem import WordNetLemmatizer

wnl = WordNetLemmatizer()
text = ['She gripped the armrest as he passed two cars at a time.',
        'Her car was in full view.',
        'A number of cars carried out of state license plates.']

output = []
for sentence in text:
    output.append(" ".join([wnl.lemmatize(i) for i in sentence.split()]))

for item in output:
    print(item)

print("*" * 10)
print(wnl.lemmatize('jumps', 'n'))
print(wnl.lemmatize('jumping', 'v'))
print(wnl.lemmatize('jumped', 'v'))

print("*" * 10)
print(wnl.lemmatize('saddest', 'a'))
print(wnl.lemmatize('happiest', 'a'))
print(wnl.lemmatize('easiest', 'a'))
```

Output:

```
She gripped the armrest a he passed two car at a time.
Her car wa in full view.
A number of car carried out of state license plates.
**********
jump
jump
jump
**********
sad
happy
easy
```

12 使用 NLTK 从文本文件中查找每个单词的频率

```
import nltk

from nltk.corpus import webtext

from nltk.probability import FreqDist


nltk.download('webtext')

wt_words = webtext.words('testing.txt')

data_analysis = nltk.FreqDist(wt_words)


# Let's take the specific words only if their frequency is greater than 3.

filter_words = dict([(m, n) for m, n in data_analysis.items() if len(m) > 3])


for key in sorted(filter_words):

    print("%s: %s" % (key, filter_words[key]))


data_analysis = nltk.FreqDist(filter_words)


data_analysis.plot(25, cumulative=False)
```

Output:

```
[nltk_data] Downloading package webtext to
[nltk_data]     C:\Users\amit\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\webtext.zip.
1989: 1
Accessing: 1
Analysis: 1
Anyone: 1
Chapter: 1
Coding: 1
Data: 1
...
```

13 从语料库中创建词云

```
import nltk

from nltk.corpus import webtext
from nltk.probability import FreqDist
from wordcloud import WordCloud
import matplotlib.pyplot as plt

nltk.download('webtext')
wt_words = webtext.words('testing.txt')  # Sample data
data_analysis = nltk.FreqDist(wt_words)

filter_words = dict([(m, n) for m, n in data_analysis.items() if len(m) > 3])

wcloud = WordCloud().generate_from_frequencies(filter_words)

# Plotting the wordcloud
plt.imshow(wcloud, interpolation="bilinear")

plt.axis("off")
(-0.5, 399.5, 199.5, -0.5)
plt.show()
```

14 NLTK 词法散布图

```
import nltk
from nltk.corpus import webtext
from nltk.probability import FreqDist
from wordcloud import WordCloud
import matplotlib.pyplot as plt

words = ['data', 'science', 'dataset']

nltk.download('webtext')
wt_words = webtext.words('testing.txt')  # Sample data

points = [(x, y) for x in range(len(wt_words))
          for y in range(len(words)) if wt_words[x] == words[y]]

if points:
    x, y = zip(*points)
else:
    x = y = ()

plt.plot(x, y, "rx", scalex=.1)
plt.yticks(range(len(words)), words, color="b")
plt.ylim(-1, len(words))
plt.title("Lexical Dispersion Plot")
plt.xlabel("Word Offset")
plt.show()
```

15 使用 countvectorizer 将文本转换为数字

```
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer

# Sample data for analysis
data1 = "Java is a language for programming that develops a software for several platforms. A compi
data2 = "Python supports multiple programming paradigms and comes up with a large standard library,
data3 = "Go is typed statically compiled language. It was created by Robert Griesemer, Ken Thompsor

df1 = pd.DataFrame({'Java': [data1], 'Python': [data2], 'Go': [data2]})

# Initialize
vectorizer = CountVectorizer()
doc_vec = vectorizer.fit_transform(df1.iloc[0])

# Create dataFrame
df2 = pd.DataFrame(doc_vec.toarray().transpose(),
                   index=vectorizer.get_feature_names())

# Change column headers
df2.columns = df1.columns
print(df2)
```

Output:

```
             Go  Java  Python
and           2     2       2
application   0     1       0
are           1     0       1
bytecode      0     1       0
can           0     1       0
code          0     1       0
comes         1     0       1
compiled      0     1       0
derived       0     1       0
```

```
develops     0    1    0
for          0    2    0
from         0    1    0
functional   1    0    1
imperative   1    0    1
...
```

16 使用 TF-IDF 创建文档术语矩阵

```python
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer

# Sample data for analysis
data1 = "Java is a language for programming that develops a software for several platforms. A compi
data2 = "Python supports multiple programming paradigms and comes up with a large standard library,
data3 = "Go is typed statically compiled language. It was created by Robert Griesemer, Ken Thompsor

df1 = pd.DataFrame({'Java': [data1], 'Python': [data2], 'Go': [data2]})

# Initialize
vectorizer = TfidfVectorizer()
doc_vec = vectorizer.fit_transform(df1.iloc[0])

# Create dataFrame
df2 = pd.DataFrame(doc_vec.toarray().transpose(),
                   index=vectorizer.get_feature_names())

# Change column headers
df2.columns = df1.columns
print(df2)
```

Output:

```
                   Go      Java    Python
and          0.323751  0.137553  0.323751
application  0.000000  0.116449  0.000000
are          0.208444  0.000000  0.208444
bytecode     0.000000  0.116449  0.000000
can          0.000000  0.116449  0.000000
code         0.000000  0.116449  0.000000
comes        0.208444  0.000000  0.208444
compiled     0.000000  0.116449  0.000000
```

```
derived      0.000000  0.116449  0.000000
develops     0.000000  0.116449  0.000000
for          0.000000  0.232898  0.000000
...
```

17 为给定句子生成 N-gram

NLTK

```
import nltk
from nltk.util import ngrams

# Function to generate n-grams from sentences.
def extract_ngrams(data, num):
    n_grams = ngrams(nltk.word_tokenize(data), num)
    return [ ' '.join(grams) for grams in n_grams]

data = 'A class is a blueprint for the object.'

print("1-gram: ", extract_ngrams(data, 1))
print("2-gram: ", extract_ngrams(data, 2))
print("3-gram: ", extract_ngrams(data, 3))
print("4-gram: ", extract_ngrams(data, 4))
```

TextBlob

```
from textblob import TextBlob

# Function to generate n-grams from sentences.
def extract_ngrams(data, num):
    n_grams = TextBlob(data).ngrams(num)
    return [ ' '.join(grams) for grams in n_grams]

data = 'A class is a blueprint for the object.'

print("1-gram: ", extract_ngrams(data, 1))
print("2-gram: ", extract_ngrams(data, 2))
print("3-gram: ", extract_ngrams(data, 3))
print("4-gram: ", extract_ngrams(data, 4))
```

Output:

```
1-gram:  ['A', 'class', 'is', 'a', 'blueprint', 'for', 'the', 'object']
2-gram:  ['A class', 'class is', 'is a', 'a blueprint', 'blueprint for', 'for the', 'the object']
3-gram:  ['A class is', 'class is a', 'is a blueprint', 'a blueprint for', 'blueprint for the', 'fo
4-gram:  ['A class is a', 'class is a blueprint', 'is a blueprint for', 'a blueprint for the', 'blu
```

18 使用带有二元组的 sklearn CountVectorize 词汇规范

```python
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer

# Sample data for analysis
data1 = "Machine language is a low-level programming language. It is easily understood by computers
data2 = "Assembly language is a representation of machine language. In other words, each assembly l

df1 = pd.DataFrame({'Machine': [data1], 'Assembly': [data2]})

# Initialize
vectorizer = CountVectorizer(ngram_range=(2, 2))
doc_vec = vectorizer.fit_transform(df1.iloc[0])

# Create dataFrame
df2 = pd.DataFrame(doc_vec.toarray().transpose(),
                   index=vectorizer.get_feature_names())

# Change column headers
df2.columns = df1.columns
print(df2)
```

Output:

```
                        Assembly  Machine
also either                    0        1
and or                         0        1
are also                       0        1
are readable                   1        0
are still                      1        0
```

```
assembly language                5         0
because each                     1         0
but difficult                    0         1
by computers                     0         1
by people                        0         1
can execute                      0         1
...
```

19 使用 TextBlob 提取名词短语

```
from textblob import TextBlob

#Extract noun
blob = TextBlob("Canada is a country in the northern part of North America.")

for nouns in blob.noun_phrases:
    print(nouns)
```

Output:

```
canada
northern part
america
```

20 如何计算词-词共现矩阵

```
import numpy as np
import nltk
from nltk import bigrams
import itertools
import pandas as pd


def generate_co_occurrence_matrix(corpus):
    vocab = set(corpus)
    vocab = list(vocab)
    vocab_index = {word: i for i, word in enumerate(vocab)}

    # Create bigrams from all words in corpus
    bi_grams = list(bigrams(corpus))

    # Frequency distribution of bigrams ((word1, word2), num_occurrences)
    bigram_freq = nltk.FreqDist(bi_grams).most_common(len(bi_grams))

    # Initialise co-occurrence matrix
    # co_occurrence_matrix[current][previous]
    co_occurrence_matrix = np.zeros((len(vocab), len(vocab)))

    # Loop through the bigrams taking the current and previous word,
    # and the number of occurrences of the bigram.
    for bigram in bigram_freq:
        current = bigram[0][1]
        previous = bigram[0][0]
        count = bigram[1]
        pos_current = vocab_index[current]
        pos_previous = vocab_index[previous]
        co_occurrence_matrix[pos_current][pos_previous] = count
    co_occurrence_matrix = np.matrix(co_occurrence_matrix)

    # return the matrix and the index
    return co_occurrence_matrix, vocab_index


text_data = [['Where', 'Python', 'is', 'used'],
             ['What', 'is', 'Python' 'used', 'in'],
             ['Why', 'Python', 'is', 'best'],
             ['What', 'companies', 'use', 'Python']]

# Create one list using many lists
data = list(itertools.chain.from_iterable(text_data))
matrix, vocab_index = generate_co_occurrence_matrix(data)


data_matrix = pd.DataFrame(matrix, index=vocab_index,
                             columns=vocab_index)
print(data_matrix)
```

Output:

```
            best  use  What  Where  ...   in   is  Python  used
best         0.0  0.0   0.0    0.0  ...  0.0  0.0     0.0   1.0
use          0.0  0.0   0.0    0.0  ...  0.0  1.0     0.0   0.0
What         1.0  0.0   0.0    0.0  ...  0.0  0.0     0.0   0.0
Where        0.0  0.0   0.0    0.0  ...  0.0  0.0     0.0   0.0
```

```
Pythonused   0.0  0.0   0.0   0.0  ...   0.0  0.0    0.0   1.0
Why          0.0  0.0   0.0   0.0  ...   0.0  0.0    0.0   1.0
companies    0.0  1.0   0.0   1.0  ...   1.0  0.0    0.0   0.0
in           0.0  0.0   0.0   0.0  ...   0.0  0.0    1.0   0.0
is           0.0  0.0   1.0   0.0  ...   0.0  0.0    0.0   0.0
Python       0.0  0.0   0.0   0.0  ...   0.0  0.0    0.0   0.0
used         0.0  0.0   1.0   0.0  ...   0.0  0.0    0.0   0.0

[11 rows x 11 columns]
```

21 使用 TextBlob 进行情感分析

```
from textblob import TextBlob


def sentiment(polarity):
    if blob.sentiment.polarity < 0:
        print("Negative")
    elif blob.sentiment.polarity > 0:
        print("Positive")
    else:
        print("Neutral")


blob = TextBlob("The movie was excellent!")
print(blob.sentiment)
sentiment(blob.sentiment.polarity)

blob = TextBlob("The movie was not bad.")
print(blob.sentiment)
sentiment(blob.sentiment.polarity)

blob = TextBlob("The movie was ridiculous.")
print(blob.sentiment)
sentiment(blob.sentiment.polarity)
```

Output:

```
Sentiment(polarity=1.0, subjectivity=1.0)
Positive
Sentiment(polarity=0.3499999999999999, subjectivity=0.6666666666666666)
Positive
```

```
Sentiment(polarity=-0.3333333333333333, subjectivity=1.0)
Negative
```

22 使用 Goslate 进行语言翻译

```python
import goslate

text = "Comment vas-tu?"

gs = goslate.Goslate()

translatedText = gs.translate(text, 'en')
print(translatedText)

translatedText = gs.translate(text, 'zh')
print(translatedText)

translatedText = gs.translate(text, 'de')
print(translatedText)
```

23 使用 TextBlob 进行语言检测和翻译

```python
from textblob import TextBlob

blob = TextBlob("Comment vas-tu?")

print(blob.detect_language())

print(blob.translate(to='es'))

print(blob.translate(to='en'))

print(blob.translate(to='zh'))
```

Output:

```
fr
¿Como estas tu?
How are you?
你好吗？
```

24 使用 TextBlob 获取定义和同义词

```
from textblob import TextBlob

from textblob import Word

text_word = Word('safe')

print(text_word.definitions)

synonyms = set()
for synset in text_word.synsets:
    for lemma in synset.lemmas():
        synonyms.add(lemma.name())

print(synonyms)
```

Output:

```
['strongbox where valuables can be safely kept', 'a ventilated or refrigerated cupboard for securin
{'secure', 'rubber', 'good', 'safety', 'safe', 'dependable', 'condom', 'prophylactic'}
```

25 使用 TextBlob 获取反义词列表

```
from textblob import TextBlob

from textblob import Word

text_word = Word('safe')

antonyms = set()
for synset in text_word.synsets:
    for lemma in synset.lemmas():
        if lemma.antonyms():
            antonyms.add(lemma.antonyms()[0].name())

print(antonyms)
```

Output:

```
{'dangerous', 'out'}
```

点下"**在看**"，微信私聊获取完整**PDF**文件

往期推荐

使用 Pandas, Jinja 和 WeasyPrint，轻松创建一个 PDF 报表

分享几段祖传的Python代码，拿来直接使用！

介绍一个打怪升级练习Python的网站，寓教于乐了属于是！

喜欢此内容的人还喜欢

Python中的生成器

我不爱机器学习

【人生苦短，我學Python】基礎篇—列表（Day8）

AI 修煉之路

Pendulum：可能是最好的Python DateTime 庫!

大鄧和他的Python