人工智慧文本分析
(Artificial Intelligence for Text Analytics)

處理解和理解文本
(Processing and Understanding Text)

1082AITA04
MBA, IMTKU (M2455) (8410) (Spring 2020)
Wed 8, 9 (15:10-17:00) (B605)

Min-Yuh Day
戴敏育
Associate Professor
副教授

Dept. of Information Management, Tamkang University
淡江大學 資訊管理學系

http://mail.tku.edu.tw/myday/
2020-03-25
課程大綱 (Syllabus)

週次 (Week)   日期 (Date)   內容 (Subject/Topics)
1   2020/03/04   人工智慧文本分析課程介紹
   (Course Orientation on Artificial Intelligence for Text Analytics)
2   2020/03/11   文本分析的基礎：自然語言處理
   (Foundations of Text Analytics: Natural Language Processing; NLP)
3   2020/03/18   Python自然語言處理
   (Python for Natural Language Processing)
4   2020/03/25   處理和理解文本
   (Processing and Understanding Text)
5   2020/04/01   文本表達特徵工程
   (Feature Engineering for Text Representation)
6   2020/04/08   人工智慧文本分析個案研究 I
   (Case Study on Artificial Intelligence for Text Analytics I)
<table>
<thead>
<tr>
<th>週次 (Week)</th>
<th>日期 (Date)</th>
<th>內容 (Subject/Topics)</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>2020/04/15</td>
<td>文本分類 (Text Classification)</td>
</tr>
<tr>
<td>8</td>
<td>2020/04/22</td>
<td>文本摘要和主題模型 (Text Summarization and Topic Models)</td>
</tr>
<tr>
<td>9</td>
<td>2020/04/29</td>
<td>期中報告 (Midterm Project Report)</td>
</tr>
<tr>
<td>10</td>
<td>2020/05/06</td>
<td>文本相似度和分群 (Text Similarity and Clustering)</td>
</tr>
<tr>
<td>11</td>
<td>2020/05/13</td>
<td>語意分析和命名實體識別 (Semantic Analysis and Named Entity Recognition; NER)</td>
</tr>
<tr>
<td>12</td>
<td>2020/05/20</td>
<td>情感分析 (Sentiment Analysis)</td>
</tr>
<tr>
<td>過次 (Week)</td>
<td>日期 (Date)</td>
<td>內容 (Subject/Topics)</td>
</tr>
<tr>
<td>------------</td>
<td>------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>13</td>
<td>2020/05/27</td>
<td>人工智慧文本分析個案研究 II  &lt;br&gt; (Case Study on Artificial Intelligence for Text Analytics II)</td>
</tr>
<tr>
<td>14</td>
<td>2020/06/03</td>
<td>深度學習和通用句子嵌入模型  &lt;br&gt; (Deep Learning and Universal Sentence-Embedding Models)</td>
</tr>
<tr>
<td>15</td>
<td>2020/06/10</td>
<td>問答系統與對話系統  &lt;br&gt; (Question Answering and Dialogue Systems)</td>
</tr>
<tr>
<td>16</td>
<td>2020/06/17</td>
<td>期末報告 I (Final Project Presentation I)</td>
</tr>
<tr>
<td>17</td>
<td>2020/06/24</td>
<td>期末報告 II (Final Project Presentation II)</td>
</tr>
<tr>
<td>18</td>
<td>2020/07/01</td>
<td>教師彈性補充教學</td>
</tr>
</tbody>
</table>
Outline

• Processing Text
• Understanding Text
Processing and Understanding Text
Free eBooks - Project Gutenberg

Some of the Latest eBooks

Welcome

New website available for testing. Visit https://dev.gutenberg.org (or http://dev.gutenberg.org) to test the site (it may have occasional outages, as improvements are made). There is a new website page that lists some known issues, and part of the motivation for the change. If you visit the new website, please consider providing your input and suggestions via an anonymous online survey afterwards.

Project Gutenberg is a library of over 60,000 free eBooks. Choose among free epub and Kindle eBooks, download them or read them online. You will find the world's great literature here, with focus on older works for which U.S. copyright has expired. Thousands of volunteers digitized and diligently proofread the eBooks, for enjoyment and education.

No fee or registration! Everything from Project Gutenberg is gratis, libre, and completely without cost to readers. If you find Project Gutenberg useful, please consider a small donation, to help Project Gutenberg digitize more books, maintain our online presence, and improve Project Gutenberg programs and offerings. Other ways to help include digitizing, proofreading and formatting, recording audio books, or reporting errors.

https://www.gutenberg.org/
The Project Gutenberg Ebook of Alice's Adventures in Wonderland, by Lewis Carroll

This ebook is for the use of anyone anywhere at no cost and with almost no restrictions whatsoever. You may copy it, give it away or re-use it under the terms of the Project Gutenberg License included with this ebook or online at www.gutenberg.org

Title: Alice's Adventures in Wonderland
Author: Lewis Carroll
Release Date: June 25, 2008 [EBook #11]
Last Updated: February 22, 2020
Language: English
Character set encoding: UTF-8

*** START OF THIS PROJECT GUTENBERG EBOOK ALICE'S ADVENTURES IN WONDERLAND ***

Produced by Arthur DiBianca and David Widger

https://www.gutenberg.org/files/11/11-h/11-h.htm
Alice Top 50 Tokens

50 most common tokens (no stopwords or punctuation)

https://tinyurl.com/imtkupyter101
Python in Google Colab (Python101)

https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT

```
nltk.download('gutenberg')
alice = Text(nltk.corpus.gutenberg.words('carroll-alice.txt'))
```

Text Processing and Understanding

- NLTK (Natural Language Processing with Python – Analyzing Text with the Natural Language Toolkit) Book: https://www.nltk.org/book/

```
[ ] 1 !pip install nltk
2 import nltk
3 nltk.download('gutenberg')
```

```
[ ] 1 from nltk.text import Text
2 alice = Text(nltk.corpus.gutenberg.words('carroll-alice.txt'))
3 alice
```

```
<Text: Alice ' s Adventures in Wonderland by Lewis Carroll 1865>
```

```
[ ] 1 print(nltk.corpus.gutenberg.fileids())
```

```
['austen-emma.txt', 'austen-persuasion.txt', 'austen-sense.txt', 'bible-kjv.txt', 'blake-poems.txt', 'bryant-stories.txt', 'burgess-buster:
Displaying 25 of 398 matches:

Alice's Adventures in Wonderland by Lewis Carroll

J CHAPTER I. Down the Rabbit-Hole Alice was beginning to get very tired of sitting about on nothing, and there was nothing - except a tiny golden key, and Alice's first thought was that it might be of very little use what things had happened lately, that Alice had begun to think that very few things were likely to be of any use to her. she was not here before,' said Alice, 'and round the neck of the bottle lay 'Drink me,' and the wise little Alice was not going to do THAT in a hurry; the bottle was NOT marked 'poison,' so Alice ventured to taste it, and finding it * * * 'What a curious feeling!' said Alice; 'I must be shutting up like a telescope for it might end, you know,' said Alice to herself, 'in my going out altogether at once; but, alas for poor Alice! when she got to the door, she fou
`alice.dispersion_plot(["Alice", "Rabbit", "Hatter", "Queen"])`

```python
import matplotlib.pyplot as plt
plt.figure(figsize=(10, 6))
alice.dispersion_plot(["Alice", "Rabbit", "Hatter", "Queen"])
```
```python
# import matplotlib.pyplot as plt
plt.figure(figsize=(10, 6))
fdist = nltk.FreqDist(alice)
fdist.plot(50)
```

https://tinyurl.com/imtkupython101
```python
for word, freq in fdist.items():
    if word.isalpha():
        # Code block
```

https://tinyurl.com/imtkupython101
```python
import nltk
nltk.download('stopwords')
stopwords = nltk.corpus.stopwords.words('english')
```

`stopwords` contains:

- 'same',
- 'so',
- 'than',
- 'too',
- 'very',
- 's',
- 't',
- 'can',
- 'will',
- 'just',
- 'don',
- "don't",
- 'should',
- 'should've',
- 'now',

[1] https://tinyurl.com/imtkupython101
for word, freq in fdist.items()
if word not in stopwords and word.isalpha()
Alice Top 50 Tokens

50 most common tokens (no stopwords or punctuation)

https://tinyurl.com/imtkupython101
BeautifulSoup

```python
import requests
from bs4 import BeautifulSoup

url = 'https://www.gutenberg.org/files/11/11-h/11-h.htm'
reqs = requests.get(url)
html_doc = reqs.text

soup = BeautifulSoup(html_doc, 'html.parser')
text = soup.get_text()
```

https://tinyurl.com/imtkupython101
from tensorflow.keras.preprocessing.text import Tokenizer

sentences = ['i love my dog', 'I, love my cat', 'You love my dog!']

tokenizer = Tokenizer(num_words=100)
tokenizer.fit_on_texts(sentences)
word_index = tokenizer.word_index
print('sentences:', sentences)
print('word index:', word_index)

sentences: ['i love my dog', 'I, love my cat', 'You love my dog!']
word index: {'love': 1, 'my': 2, 'i': 3, 'dog': 4, 'cat': 5, 'you': 6}
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

sentences = 
['I love my dog',
'I love my cat',
'You love my dog!',
'Do you think my dog is amazing?'
]

tokenizer = Tokenizer(num_words = 100, oov_token="<OOV>")
tokenizer.fit_on_texts(sentences)
word_index = tokenizer.word_index
sequences = tokenizer.texts_to_sequences(sentences)
padded = pad_sequences(sequences, maxlen=5)
print("sentences = ", sentences)
print("Word Index = ", word_index)
print("Sequences = ", sequences)
print("Padded Sequences:")
print(padded)
import pad_sequences

sentences = ['I love my dog', 'I love my cat', 'You love my dog!', 'Do you think my dog is amazing?']

Word Index = {'<OOV>': 1, 'my': 2, 'love': 3, 'dog': 4, 'i': 5, 'you': 6, 'cat': 7, 'do': 8, 'think': 9, 'is': 10, 'amazing': 11}

Sequences = [[5, 3, 2, 4], [5, 3, 2, 7], [6, 3, 2, 4], [8, 6, 9, 2, 4, 10, 11]]

Padded Sequences: [[0 5 3 2 4] [0 5 3 2 7] [0 6 3 2 4] [9 2 4 10 11]]

https://tinyurl.com/imtkupython101
Keras preprocessing text

```python
# keras.preprocessing.text Tokenizer
from keras.preprocessing.text import Tokenizer

# define 5 documents
docs = ['Well done!', 'Good work', 'Great effort', 'nice work', 'Excellent!']

# create the tokenizer
t = Tokenizer()

t.fit_on_texts(docs)

print('docs:', docs)

print('word_counts:', t.word_counts)

print('document_count:', t.document_count)

print('word_index:', t.word_index)

print('word_docs:', t.word_docs)

# integer encode documents
texts_to_matrix = t.texts_to_matrix(docs, mode='count')

print('texts_to_matrix:')

print(texts_to_matrix)
```

Using TensorFlow backend.

```text
docs: ['Well done!', 'Good work', 'Great effort', 'nice work', 'Excellent!']
word_counts:OrderedDict([('well', 1), ('done', 1), ('good', 1), ('work', 2), ('great', 1), ('effort', 1), ('nice', 1), ('excellent', 1)]
document_count: 5
word_index: {'work': 1, 'well': 2, 'done': 3, 'good': 4, 'great': 5, 'effort': 6, 'nice': 7, 'excellent': 8}
word_docs: {'done': 1, 'well': 1, 'work': 2, 'good': 1, 'great': 1, 'effort': 1, 'nice': 1, 'excellent': 1}
texts_to_matrix:
[[0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 1. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0.]]```
One-hot encoding

'The mouse ran up the clock' =

\[
\text{The} 1 \quad [0, 1, 0, 0, 0, 0, 0, 0], \\
\text{mouse} 2 \quad [0, 0, 1, 0, 0, 0, 0, 0], \\
\text{ran} 3 \quad [0, 0, 0, 1, 0, 0, 0, 0], \\
\text{up} 4 \quad [0, 0, 0, 0, 1, 0, 0, 0], \\
\text{the} 1 \quad [0, 1, 0, 0, 0, 0, 0, 0], \\
\text{clock} 5 \quad [0, 0, 0, 0, 0, 0, 1, 0] \\n\]

\[0, 1, 2, 3, 4, 5, 6\]
Word embeddings

Male-Female

Verb Tense

Country-Capital

Source: https://developers.google.com/machine-learning/guides/text-classification/step-3
Word embeddings

The mouse ran up the clock

1. the (1) → [1, 2, 3, 4, 1, 5] → [[0.236, -0.141, 0.000, 0.045], [0.006, 0.652, 0.270, -0.556], [0.305, 0.569, -0.028, 0.496], [0.421, 0.195, -0.058, 0.477], [0.236, -0.141, 0.000, 0.045], [0.844, -0.001, 0.763, 0.201]]

2. mouse (2) → [1, 2, 3, 4, 1, 5] → [[0.236, -0.141, 0.000, 0.045], [0.006, 0.652, 0.270, -0.556], [0.305, 0.569, -0.028, 0.496], [0.421, 0.195, -0.058, 0.477], [0.236, -0.141, 0.000, 0.045], [0.844, -0.001, 0.763, 0.201]]

3. ran (3) → [1, 2, 3, 4, 1, 5] → [[0.236, -0.141, 0.000, 0.045], [0.006, 0.652, 0.270, -0.556], [0.305, 0.569, -0.028, 0.496], [0.421, 0.195, -0.058, 0.477], [0.236, -0.141, 0.000, 0.045], [0.844, -0.001, 0.763, 0.201]]

4. up (4) → [1, 2, 3, 4, 1, 5] → [[0.236, -0.141, 0.000, 0.045], [0.006, 0.652, 0.270, -0.556], [0.305, 0.569, -0.028, 0.496], [0.421, 0.195, -0.058, 0.477], [0.236, -0.141, 0.000, 0.045], [0.844, -0.001, 0.763, 0.201]]

5. clock (5) → [1, 2, 3, 4, 1, 5] → [[0.236, -0.141, 0.000, 0.045], [0.006, 0.652, 0.270, -0.556], [0.305, 0.569, -0.028, 0.496], [0.421, 0.195, -0.058, 0.477], [0.236, -0.141, 0.000, 0.045], [0.844, -0.001, 0.763, 0.201]]

The mouse ran down

6. down (6) → [1, 2, 3, 6] → [[0.236, -0.141, 0.000, 0.045], [0.006, 0.652, 0.270, -0.556], [0.305, 0.569, -0.028, 0.496], [0.466, -0.326, 0.884, 0.007]]
t1 = 'The mouse ran up the clock'
t2 = 'The mouse ran down'
s1 = t1.lower().split(' ')
s2 = t2.lower().split(' ')
terms = s1 + s2
sortedset = sorted(set(terms))
print('terms =', terms)
print('sortedset =', sortedset)

terms = ['the', 'mouse', 'ran', 'up', 'the', 'clock', 'the', 'mouse', 'ran', 'down']
sortedset = ['clock', 'down', 'mouse', 'ran', 'the', 'up']

https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT
t1 = 'The mouse ran up the clock'
t2 = 'The mouse ran down'
s1 = t1.lower().split(' ')
s2 = t2.lower().split(' ')
terms = s1 + s2
print(terms)

tfdict = {}
for term in terms:
    if term not in tfdict:
        tfdict[term] = 1
    else:
        tfdict[term] += 1

a = []
for k,v in tfdict.items():
    a.append('{}: {}'.format(k,v))
print(a)

['the', 'mouse', 'ran', 'up', 'the', 'clock', 'the', 'mouse', 'ran', 'down']
['the', 3, 'mouse', 2, 'ran', 2, 'up', 1, 'clock', 1, 'down', 1]
sorted_by_value_reverse = sorted(tfdict.items(),
key=lambda kv: kv[1], reverse=True)

sorted_by_value_reverse_dict =
dict(sorted_by_value_reverse)

id2word = {id: word for id, word in 
enumerate(sorted_by_value_reverse_dict)}

word2id = dict([(v, k) for (k, v) in 
id2word.items()])

https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT
sorted_by_value = sorted(tfdict.items(), key=lambda kv: kv[1])
print('sorted_by_value: ', sorted_by_value)

sorted_by_value2 = sorted(tfdict, key=tfdict.get, reverse=True)
print('sorted_by_value2: ', sorted_by_value2)

sorted_by_value_reverse = sorted(tfdict.items(), key=lambda kv: kv[1], reverse=True)
print('sorted_by_value_reverse: ', sorted_by_value_reverse)

sorted_by_value_reverse_dict = dict(sorted_by_value_reverse)
print('sorted_by_value_reverse_dict: ', sorted_by_value_reverse_dict)

id2word = {id: word for id, word in enumerate(sorted_by_value_reverse_dict)}
print('id2word: ', id2word)

word2id = dict([(v, k) for (k, v) in id2word.items()])
print('word2id: ', word2id)

print('len_words: ', len(word2id))

sorted_by_key = sorted(tfdict.items(), key=lambda kv: kv[0])
print('sorted_by_key: ', sorted_by_key)


tfstring = '\n'.join(a)
print(tfstring)

tf = tfdict.get('mouse')
print(tf)
from keras.preprocessing.text import Tokenizer

docs = ['Well done!', 'Good work', 'Great effort', 'nice work', 'Excellent!']

t = Tokenizer()
t.fit_on_texts(docs)

docs: ['Well done!', 'Good work', 'Great effort', 'nice work', 'Excellent!']

word_counts:OrderedDict([('well', 1), ('done', 1), ('good', 1), ('work', 2), ('great', 1), ('effort', 1), ('nice', 1), ('excellent', 1)]

word_index: {'work': 1, 'well': 2, 'done': 3, 'good': 4, 'great': 5, 'effort': 6, 'nice': 7, 'excellent': 8}

texts_to_matrix:

[[0. 0. 1. 1. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 1. 1. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 1.]]
from keras.preprocessing.text import Tokenizer

# define 5 documents
docs = ['Well done!', 'Good work', 'Great effort', 'nice work', 'Excellent!']
# create the tokenizer
t = Tokenizer()
# fit the tokenizer on the documents
t.fit_on_texts(docs)
print('docs:', docs)
print('word_counts:', t.word_counts)
print('document_count:', t.document_count)
print('word_index:', t.word_index)
print('word_docs:', t.word_docs)
# integer encode documents
texts_to_matrix = t.texts_to_matrix(docs, mode='count')
print('texts_to_matrix:
', texts_to_matrix)
print(texts_to_matrix)
texts_to_matrix =
t.texts_to_matrix(docs, mode='count')

docs: ['Well done!', 'Good work', 'Great effort',
'nice work', 'Excellent!']
word_counts: OrderedDict([('well', 1), ('done', 1),
('good', 1), ('work', 2), ('great', 1), ('effort', 1),
('nice', 1), ('excellent', 1)])
document_count: 5
word_index: {'work': 1, 'well': 2, 'done': 3, 'good': 4,
'great': 5, 'effort': 6, 'nice': 7, 'excellent': 8}
word_docs: {'done': 1, 'well': 1, 'work': 2, 'good': 1,
'great': 1, 'effort': 1, 'nice': 1, 'excellent': 1}
texts_to_matrix:

[[0. 0. 1. 1. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 1. 1. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 1.]]
from keras.preprocessing.text import Tokenizer
# define 5 documents
docs = ['Well done!', 'Good work', 'Great effort', 'nice work', 'Excellent!']
# create the tokenizer
t = Tokenizer()
# fit the tokenizer on the documents
t.fit_on_texts(docs)
print('docs:', docs)
print('word_counts:', t.word_counts)
print('document_count:', t.document_count)
print('word_index:', t.word_index)
print('word_docs:', t.word_docs)
# integer encode documents
texts_to_matrix = t.texts_to_matrix(docs, mode='tfidf')
print('texts_to_matrix:')
print(texts_to_matrix)

Source: https://machinelearningmastery.com/prepare-text-data-deep-learning-keras/
Summary

• Processing Text
• Understanding Text
References


• NLTK (Natural Language Processing with Python – Analyzing Text with the Natural Language Toolkit) Book: https://www.nltk.org/book/


• Laurence Moroney (2020), Natural Language Processing - Tokenization (NLP Zero to Hero, part 1), https://www.youtube.com/watch?v=fNxaJsNG3-s

• Laurence Moroney (2020), Natural Language Processing - Sequencing - Turning sentence into data (NLP Zero to Hero, part 2), https://www.youtube.com/watch?v=r9QjkdSJZ2g

• Laurence Moroney (2020), Natural Language Processing - Training a model to recognize sentiment in text (NLP Zero to Hero, part 3), https://www.youtube.com/watch?v=Y_hzMnRXjhl