D09 基礎語言模型：N-Gram

簡報閱讀 範例與作業 問題討論

重要知識點
N-Gram語言模型
建立詞頻字典
搭建N-Gram模型

重要知識點

- 了解如何實作與使用N-gram模型
- 透過實作更加了解N-gram模型
- 使用NLTK建構N-gram模型

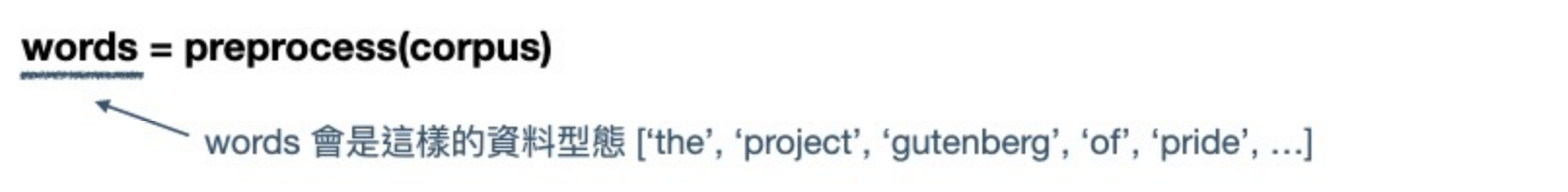
N-Gram語言模型

- N-Gram語言模型是基於統計的語言模型算法，基於馬可夫假設將文本中的內容取最靠近的N個字當作條件機率計算的先驗條件，形成長度是N的字詞片段序列。每個字詞片段及稱為gram。
- 在了解N-Gram語言模型的原理基礎後，現在來看該如何使用Python來實踐N-Gram模型。

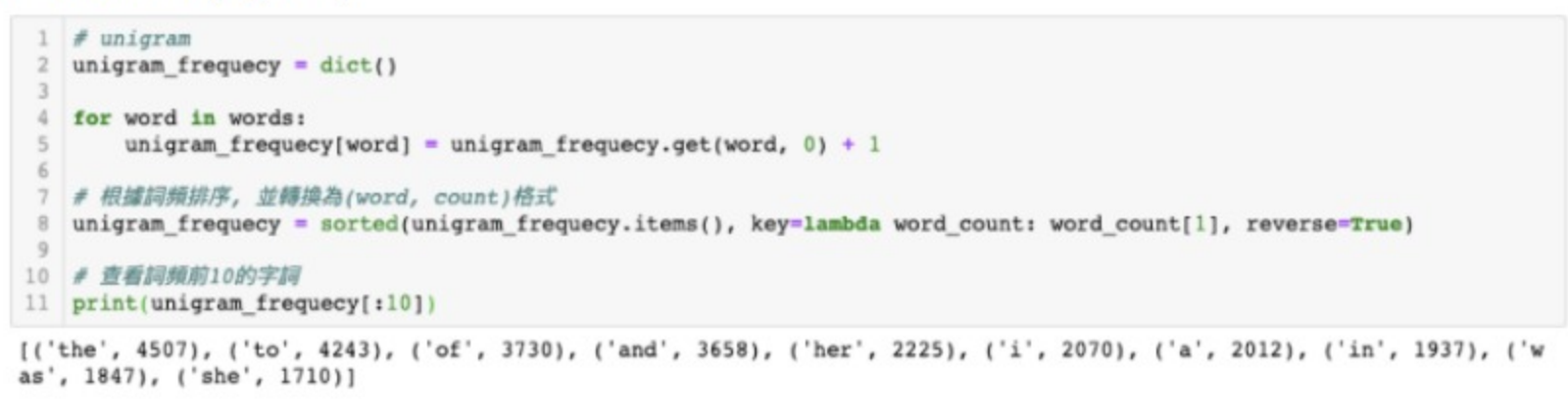
建立詞頻字典

N-Gram語言模型是基於詞頻來計算機率，因此須先將文本資料轉化為詞頻字典。

- 準備文本資料(這裡將文本資料儲存在words變數中)
- 這裡的preprocess為將文本資料做前處理(詳細可見附件.ipynb)



建立詞頻字典(bigram)



```
# unigram
unigram_frequecy = dict()

for word in words:
    unigram_frequecy[word] = unigram_frequecy.get(word, 0) + 1

# 根據詞頻排序, 並轉換為(word, count)格式
unigram_frequecy = sorted(unigram_frequecy.items(), key=lambda word_count: word_count[1], reverse=True)

# 查看詞頻前10的字詞
print(unigram_frequecy[:10])
```

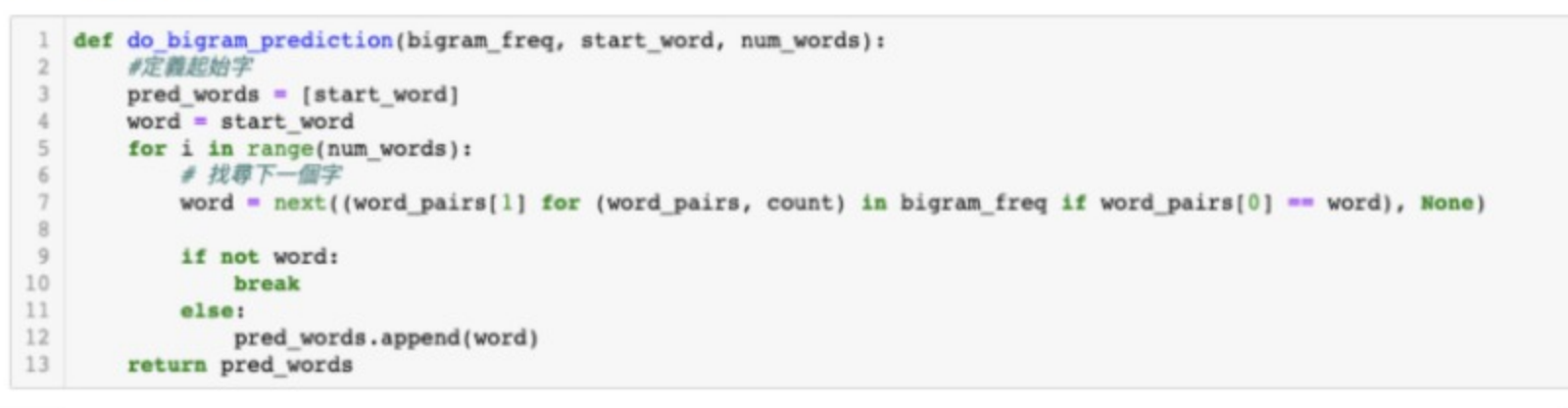
```
[('the', 4507), ('to', 4243), ('of', 3730), ('and', 3658), ('her', 2225), ('i', 2070), ('a', 2012), ('in', 1937), ('was', 1847), ('she', 1710)]
```



```
# unigram
unigram_frequecy = dict()

for word in words:
    unigram_frequecy[word] = unigram_frequecy.get(word, 0) + 1

# 根據詞頻排序, 並轉換為(word, count)格式
unigram_frequecy = sorted(unigram_frequecy.items(), key=lambda word_count: word_count[1], reverse=

# 查看詞頻前10的字詞
print(unigram_frequecy[:10])
```


搭建N-Gram模型

建立Bigram模型



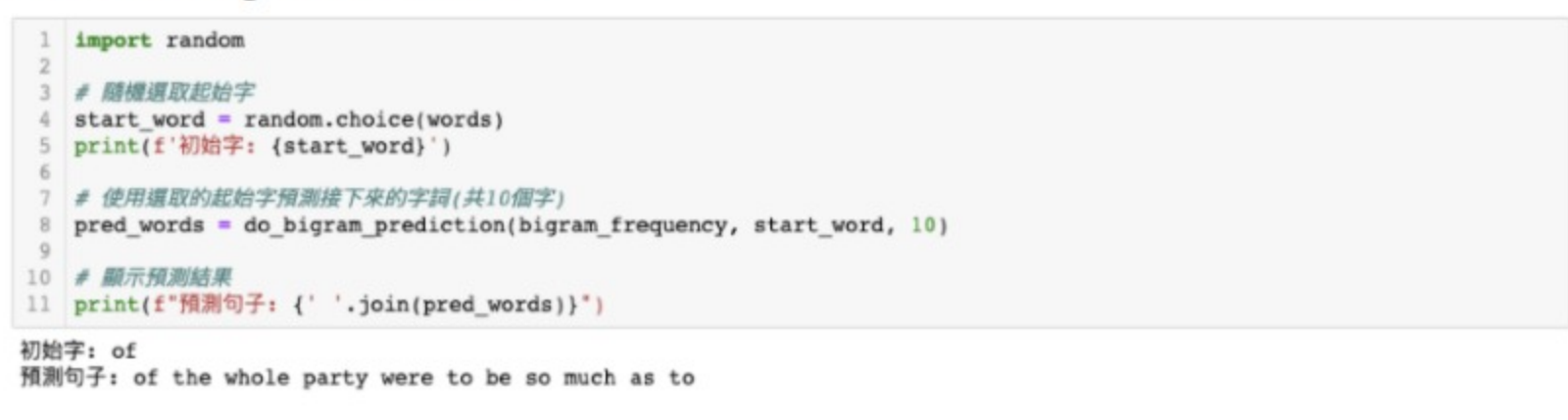
```
def do_bigram_prediction(bigram_freq, start_word, num_words):
    #定義起始字
    pred_words = [start_word]
    word = start_word
    for i in range(num_words):
        # 找尋下一個字
        word = next((word_pairs[1] for (word_pairs, count) in bigram_freq if word_pairs[0] == word), None)

        if not word:
            break
        else:
            pred_words.append(word)
    return pred_words
```



```
def do_bigram_prediction(bigram_freq, start_word, num_words):
    #定義起始字
    pred_words = [start_word]
    word = start_word
    for i in range(num_words):
        # 找尋下一個字
        word = next((word_pairs[1] for (word_pairs, count) in bigram_freq if word_pairs[0] == word

        if not word:
            break
        else:
            pred_words.append(word)
```


使用N-Gram模型進行預測

使用建立好的Bigram模型進行預測



```
import random

# 隨機選取起始字
start_word = random.choice(words)
print(f'初始字: {start_word}')

# 使用選取的起始字預測接下來的字詞(共10個字)
pred_words = do_bigram_prediction(bigram_frequency, start_word, 10)

# 顯示預測結果
print(f"預測句子: {' '.join(pred_words)}")
```

```
初始字: of
預測句子: of the whole party were to be so much as to
```



```
import random

# 隨機選取起始字
start_word = random.choice(words)
print(f'初始字: {start_word}')

# 使用選取的起始字預測接下來的字詞(共10個字)
pred_words = do_bigram_prediction(bigram_frequency, start_word, 10)

# 顯示預測結果
print(f"預測句子: {' '.join(pred_words)}")
```

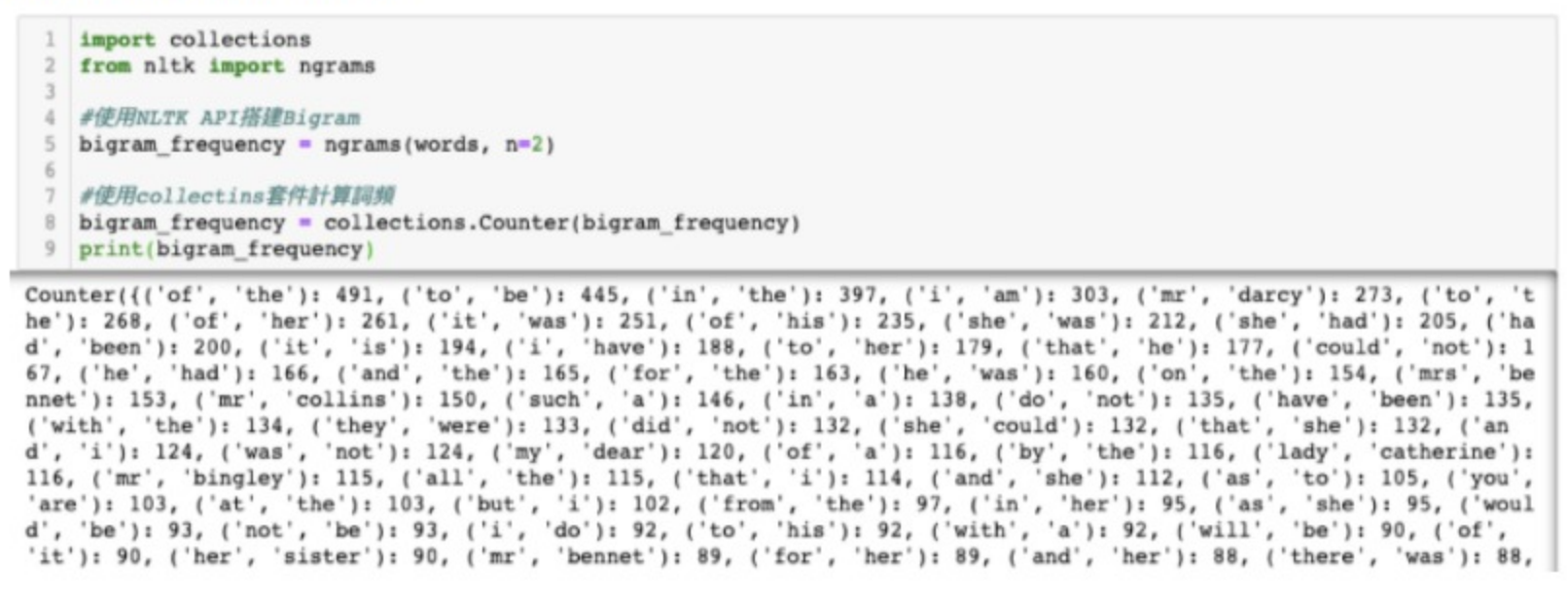

使用NLTK建構N-Gram模型

NLTK(Natural Language Toolkit)使一個python的自然語言處理套件，內含許多應用，
包括了N-Gram模型

安裝NLTK套件

- pip install nltk

使用NLTK API搭建N-Gram



```
import collections
from nltk import ngrams

#使用NLTK API搭建Bigram
bigram_frequency = ngrams(words, n=2)

#使用collectins套件計算詞頻
bigram_frequency = collections.Counter(bigram_frequency)
print(bigram_frequency)
```

```
Counter({('of', 'the'): 491, ('to', 'be'): 445, ('in', 'the'): 397, ('i', 'am'): 303, ('mr', 'darcy'): 273, ('to', 'the'): 268, ('of', 'her'): 261, ('it', 'was'): 251, ('of', 'his'): 235, ('she', 'was'): 212, ('she', 'had'): 205, ('had', 'been'): 200, ('it', 'is'): 194, ('i', 'have'): 188, ('to', 'her'): 179, ('that', 'he'): 177, ('could', 'not'): 167, ('he', 'had'): 166, ('and', 'the'): 165, ('for', 'the'): 163, ('he', 'was'): 160, ('on', 'the'): 154, ('mrs', 'bennet'): 153, ('mr', 'collins'): 150, ('such', 'a'): 146, ('in', 'a'): 138, ('do', 'not'): 135, ('have', 'been'): 135, ('with', 'the'): 134, ('they', 'were'): 133, ('did', 'not'): 132, ('she', 'could'): 132, ('that', 'she'): 132, ('and', 'i'): 124, ('was', 'not'): 124, ('my', 'dear'): 120, ('of', 'a'): 116, ('by', 'the'): 116, ('lady', 'catherine'): 116, ('mr', 'bingley'): 115, ('all', 'the'): 115, ('that', 'i'): 114, ('and', 'she'): 112, ('as', 'to'): 105, ('you', 'are'): 103, ('at', 'the'): 103, ('but', 'i'): 102, ('from', 'the'): 97, ('in', 'her'): 95, ('as', 'she'): 95, ('would', 'be'): 93, ('not', 'be'): 93, ('i', 'do'): 92, ('to', 'his'): 92, ('with', 'a'): 92, ('will', 'be'): 90, ('of', 'it'): 90, ('her', 'sister'): 90, ('mr', 'bennet'): 89, ('for', 'her'): 89, ('and', 'her'): 88, ('there', 'was'): 88,
```



```
import collections
from nltk import ngrams

#使用NLTK API搭建Bigram
bigram_frequency = ngrams(words, n=2)

#使用collectins套件計算詞頻
bigram_frequency = collections.Counter(bigram_frequency)
print(bigram_frequency)
```


詳細使用操作

請參照使用基礎語言模型_Ngram.ipynb檔進行更詳細的使用操作

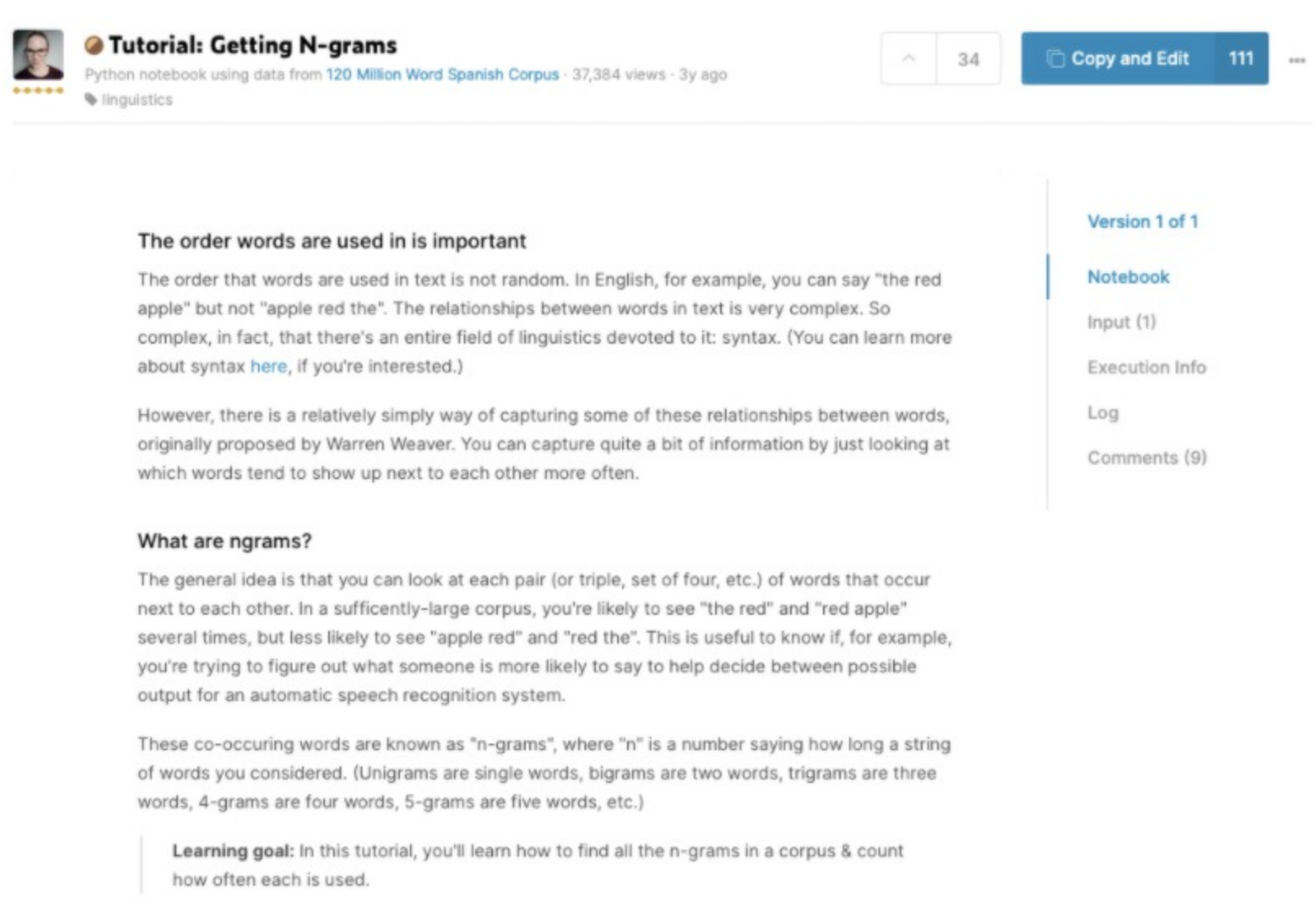
知識點回顧

在這章節我們學習到了

- 實作與使用N-gram模型
- 使用NLTK建構N-gram模型

延伸閱讀

網站：[取的N-Gram]
使用NLTK建構N-Gram



Tutorial: Getting N-grams
Python notebook using data from 120 Million Word Spanish Corpus · 37,384 views · 3y ago
linguistics

34 Copy and Edit 111

The order words are used in is important

The order that words are used in text is not random. In English, for example, you can say "the red apple" but not "apple red the". The relationships between words in text is very complex. So complex, in fact, that there's an entire field of linguistics devoted to it: syntax. (You can learn more about syntax here, if you're interested.)

However, there is a relatively simply way of capturing some of these relationships between words, originally proposed by Warren Weaver. You can capture quite a bit of information by just looking at which words tend to show up next to each other more often.

What are ngrams?

The general idea is that you can look at each pair (or triple, set of four, etc.) of words that occur next to each other. In a sufficently-large corpus, you're likely to see "the red" and "red apple" several times, but less likely to see "apple red" and "red the". This is useful to know if, for example, you're trying to figure out what someone is more likely to say to help decide between possible output for an automatic speech recognition system.

These co-occuring words are known as "n-grams", where "n" is a number saying how long a string of words you considered. (Unigrams are single words, bigrams are two words, trigrams are three words, 4-grams are four words, 5-grams are five words, etc.)

> **Learning goal:** In this tutorial, you'll learn how to find all the n-grams in a corpus & count how often each is used.

Version 1 of 1
Notebook
Input (1)
Execution Info
Log
Comments (9)

網站：[使用Python搭建N-Gram模型應用]
利用Python搭建得N-Gram模型進行新聞文本預測應用

用Python 實現n-gram 語言模型進行新聞文本內容預測

Jed · 2019-12-01 · 3評論 · 1593閱讀

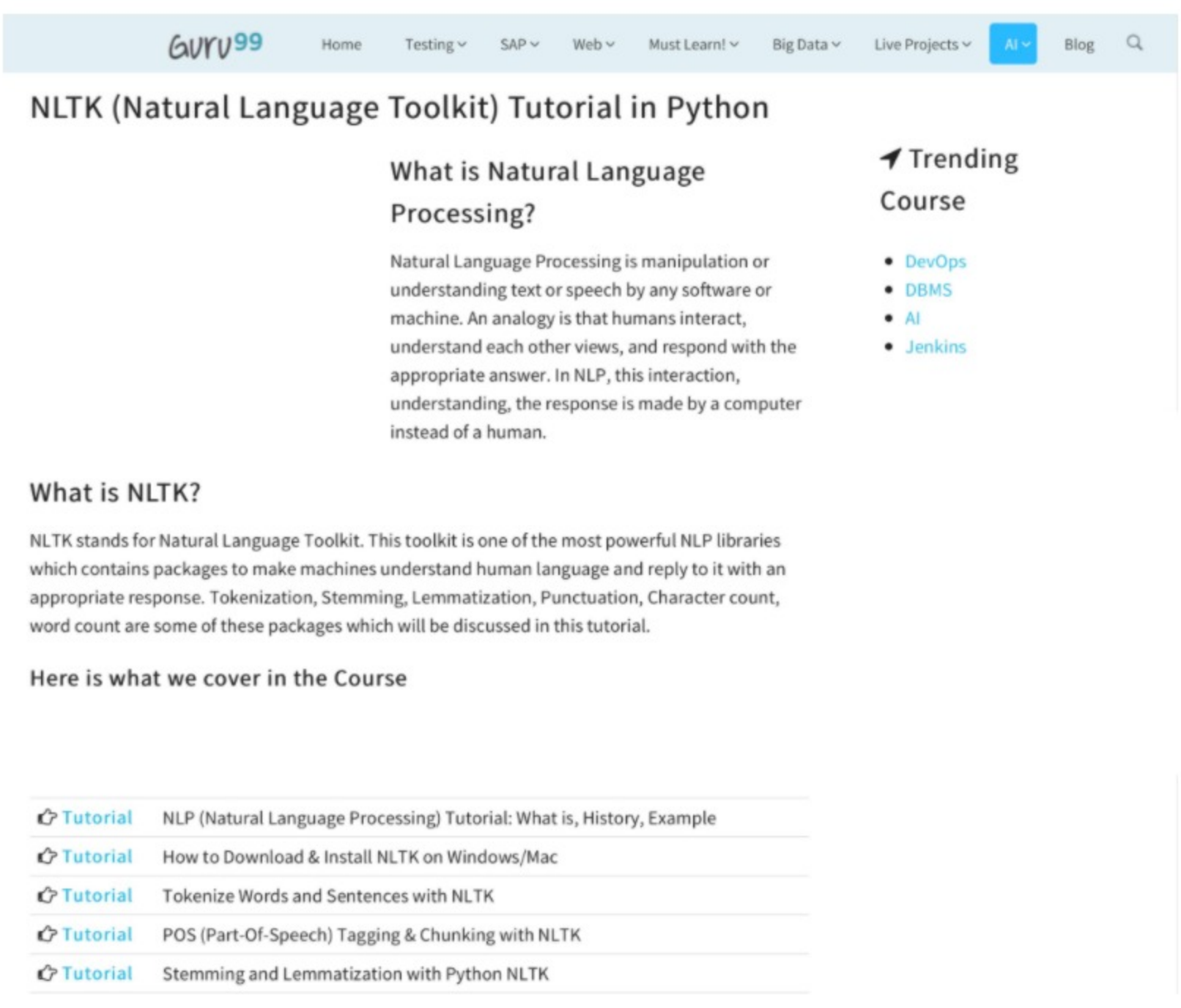
概述

語言模型與新聞的內容預測，分為4 個階段：

1. 從主流新聞網站科技頻道抓取新聞內容；
2. 數據預處理；
3. 實現n-gram 語言模型；
4. 在測試集上檢驗模型。

完整代碼：https://github.com/xJed/ngram-text-prediction

網站：[NLTK使用教學手冊]
NLTK各項應用教學



Guru99 Home Testing SAP Web Must Learn! Big Data Live Projects AI Blog

NLTK (Natural Language Toolkit) Tutorial in Python

What is Natural Language Processing?

Natural Language Processing is manipulation or understanding text or speech by any software or machine. An analogy is that humans interact, understand each other views, and respond with the appropriate answer. In NLP, this interaction, understanding, the response is made by a computer instead of a human.

Trending Course

- DevOps
- DBMS
- AI
- Jenkins

What is NLTK?

NLTK stands for Natural Language Toolkit. This toolkit is one of the most powerful NLP libraries which contains packages to make machines understand human language and reply to it with an appropriate response. Tokenization, Stemming, Lemmatization, Punctuation, Character count, word count are some of these packages which will be discussed in this tutorial.

Here is what we cover in the Course

Tutorial	NLP (Natural Language Processing) Tutorial: What is, History, Example
Tutorial	How to Download & Install NLTK on Windows/Mac
Tutorial	Tokenize Words and Sentences with NLTK
Tutorial	POS (Part-Of-Speech) Tagging & Chunking with NLTK
Tutorial	Stemming and Lemmatization with Python NLTK

下一步：閱讀範例與完成作業