
自然語言處理
商汤
sensetime

圖靈測試

圖靈測試——判斷機器是否具有智慧

圖靈在 1950 年提出過這樣一種試驗:

將多個測試者與被測試者隔開，被測試者可能是人也可能是機器，測試者通過一些裝置(如鍵盤)向被測試者隨意提問。

在多次問答後， 如果有超過 30% 的測試者不能確定出被測試者是人還是機器，那麼這台機器就通過了測試，並被認為具有人類智慧。

自然語言處理一直是人工智能的核心問題之一。

自然語言處理

自然語言是指本身存在於人類社會的語言，包括我們說的話和形成的文字。

如果機器對自然語言能夠有非常準確的處理，並且做出合適的回應，那麼很多人工智能的應用都可以由此實現。

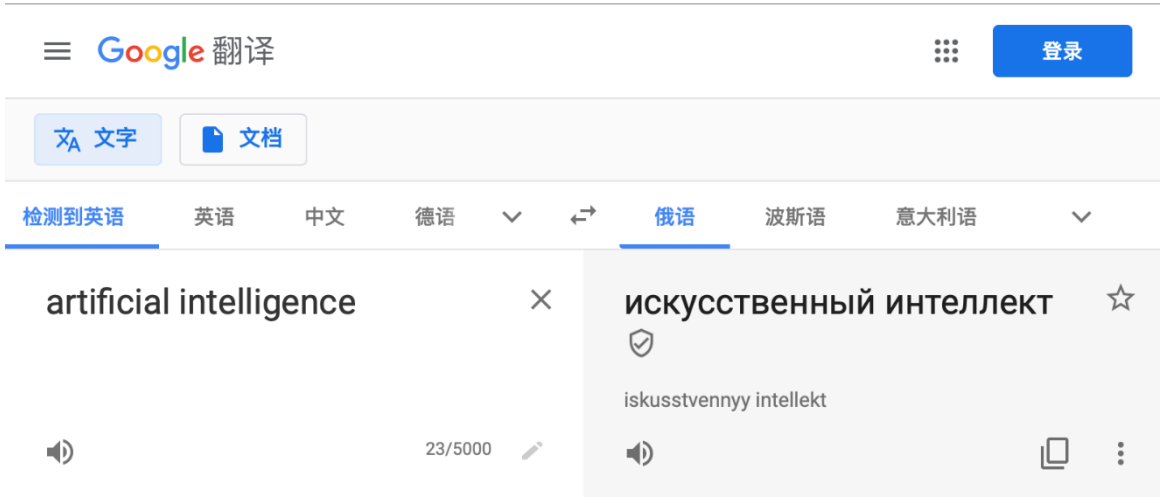


圖 翻譯軟體

翻譯軟體可以實現主流語言之間的轉換

圖 天貓精靈

智慧音箱可以跟我們聊天，按照我們的指令做一些簡單的事情。

智慧英文課文分析

小剛小朋友發現有些課外英文讀物的文章對於他來說太難了。於是，愛鑽研的小剛就想，可不可以讓電腦自動分析一篇英語文章的難度呢？

the little prince discovers a garden of roses

But it happened that after walking for a long time through sand, and rocks, and snow, the little prince at last came upon a road. And all roads lead to the abodes of men.

"Good morning," he said.

He was standing before a garden, all a-bloom with roses.

From his garden, Mr. Collins would have led them round his two meadows, but the ladies, not having shoes to encounter the remains of a white frost, turned back; and while Sir William accompanied him, Charlotte took her sister and friend over the house, extremely well pleased, probably, to have the opportunity of shewing it without her husband's help. It was rather small, but well built and convenient; and everything was fitted up and arranged with a neatness and consistency of which Elizabeth gave Charlotte all the credit. When Mr. Collins could be forgotten, there was really a great air of comfort throughout, and by Charlotte's evident enjoyment of it, Elizabeth supposed he must be often forgotten. She had already learnt that Lady Catherine was still in the country. It was spoken of again while they were at dinner, when Mr. Collins joining in, observed,

"Yes, Miss Elizabeth, you will have the honour of seeing Lady Catherine de Bourgh on the ensuing Sunday at church, and I need not say you will be delighted with her. She is all affability and condescension, and I doubt not but you will be honoured with some portion of her notice when service is over. I have scarcely any hesitation in saying that she will include you and my sister Maria in every invitation with which she honours us during your stay here. Her behaviour to my dear Charlotte is charming. We dine at Rosings twice every week, and are never allowed to walk home. Her ladyship's carriage is regularly ordered for us. I should say, one of her ladyship's carriages, for she has several."

完全可以！這個問題可以通過自然語言處理技術來解決。

字串的創建和列印

上述這種字元到整數的轉換並不需要手動完成。

在實際編寫程式時， 可以利用 Python 中的字串類型方便地對文本進行表示和處理。

在 Python 中，將一段文本用**成對**的雙引號或者單引號包含起來，就可以創建字串。比如:

```
string = “Hello World!”
```

string 中保存的就是一個字串，內容為“Hello World!”。我們可以用 print 函數將它列印在螢幕上:

```
print(string)
```

文本的輸入

Q：之前我們介紹了創建字串的一種方式——用成對的引號將一段文本包含起來。這個方式在面對大量文本的時候還合適嗎？

對大量的文本進行分析的時候，將所有的文本內容都寫在程式裡很不方便。我們可以將文本存儲在文件中，每當需要時就從相應檔中讀取。

實驗平臺提供了**readtext**函數

該函數接受一個表示檔路徑的字串作為參數，返回一個字串，其中就是該文字檔的全部內容。

readtext 函數

readtext實驗1：假設文字檔的路徑為 /path/to/text.txt，利用 readtext 函數讀取該檔中的內容。

```
text = readtext("/path/to/text.txt") # 讀取檔中的內容
```

注意：此處的 readtext 表示我們調用的是 SenseStudy 提供的 nlp 包中的 readtext 函數。之後當我們需要調用 nlp 包中的函數時都可 以參考這種格式。

readtext實驗2：將變數 text 所保存的字串內容列印出來。

```
print(text) # 列印字串
# 輸出:路徑為 /path/to/text.txt 的文字檔中的內容
```

writetext 函數

writetext實驗1： 利用 writetext 函數將自己名字的全拼寫入路徑為 /path/to/text_1. txt 的文字檔中。

```
text = "XiaoMing" # 創建字串
writetext(text, "/path/to/text_1.txt") # 將字串存入檔
```

注意：當存在 /path/to/text_1.txt 文件時，writetext 會將其內容覆蓋; 當不存在 /path/to/text_1.txt 文件時，writetext 會創建相應文件。

writetext實驗2： 利用 readtext 函數讀取路徑為 /path/to/text_1.txt 的文字檔中 的內容，然後列印出來，看一看是不是你的名字呢。

```
text_1 = readtext ("/path/to/text_1.txt") # 讀取檔中的內容
print(text_1) # 列印字串
# 輸出:"XiaoMing"
```

分詞

單詞組成了句子，

句子組成了段落，

段落又組成了完整的文章。

單詞是表達語義的最小單位。

單詞如同一磚一瓦，是一篇文章的基石。

當我們看到一篇文章時，大腦會自動將文章分割成了一個個單詞——也就是所謂的分詞。

1. 下面是去除了單詞之間分隔的一句話，嘗試理解其中的含義。

everyoneisgoodatsomethingbutsomepeopleareтrulytalented

2. 把單詞之間的分隔都恢復，再試試呢。

everyone is good at something, but some people are truly talented.

分詞

為了讓電腦能夠自動分析英語文章的難度，第一步是教會它如何分詞。

Q：聯繫剛才的實驗，想一想，英語文章通過什麼來進行分詞?

就英文而言，可以直接通過**空格**、**標點符號**等劃分不同的單詞。

實現自動分詞功能的函數——splitwords

實驗平臺為大家提供了可以實現英文分詞功能的函數——splitwords。只要將字串傳入該函數，就可以實現自動分詞。

splitwords 函數

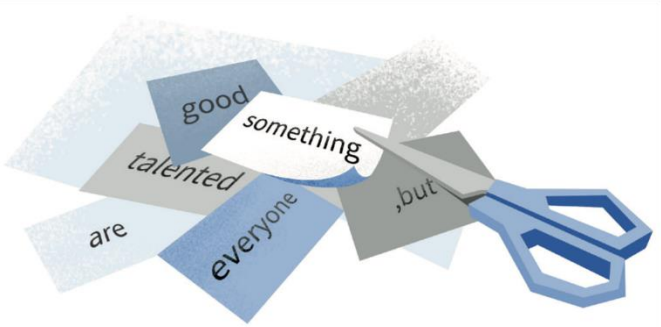
實驗： 利用 splitwords 函數對下面這句話進行分詞。

everyone is good at something, but some people are truly talented.

```
text_2 = "everyone is good at something, but some people are truly talented." # 創建字串
words = nlp.splitwords(text_2) # 分詞
print(words) # 列印分詞結果
# 輸出:["everyone", "is", "good", "at", "something", "but", "some", "people", "are", "truly", "talented"]
```

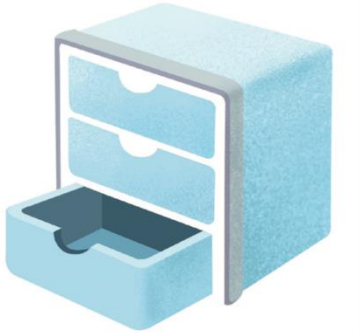
分詞

如果一段文本是寫在紙上的，
那麼分詞的過程就是將單詞從紙上一個一個裁剪下來，
得到一張張紙片，每張紙片上有且只有一個單詞。



目的：把單詞和單詞區分開

Q：區分開的單詞像上圖那樣存放合適嗎?



收納盒的思想

- 每張紙片單獨存放在一個抽屜中，不會出現混淆;
- 當需要取紙片時，只需要知道它在第幾個抽屜，直接打開相應抽屜就可以。

列表

Python 中類似“收納盒”的資料類型——列表

列表

列表是一個有順序的容器，裡面有一系列位置，每一個位置都能容納一個值，我們稱之為元素。

可以用中括弧將一些用逗號分開的元素括起來，得到一個列表:

變量 = [元素0， 元素1， .....]

```
# 輸出:["everyone", "is", "good", "at", "something", "but", "some", "people", "are", "truly", "talented"]
```

我們前面介紹過 的 splitwords 函數返回的就是一個列表，裡面的每個元素就是對一段文本分詞得到的各個單詞。

列表元素的個數

收納盒看的元素得見摸得著，我們很容易數出它有幾個抽屜。列表存放在電腦中，我們怎麼能知道它有多少個元素呢?用 len 函數就可以獲取一個列表的元素個數:

len(列表對象)

練習：創建列表，列印出其中的元素個數。

```
list_3 = [1, 3, 5] # 創建列表
print(len(list_3)) # 列印列表元素個數
# 輸出:3
```

列表元素的個數

使用收納盒時，我們可以依次打開每個抽屜，使用裡面的物品。同樣， 我們也可以依次訪問列表中的每個元素。方法很簡單，使用 Python 的 for 循環語句就可以。形式如下:

for item in 列表對象 :

循環體

上述代碼在每次循環中依次取出列表中的元素，賦給變量 item(變量名可以自行定義，並不一定是 item)，循環體中就可以使用 item 了。

```
list_4 = [1, 3, 5] # 創建列表
for item in list_4: # 使用 for 循環對列表元素逐個訪問
print(item) # 列印該元素 # 輸出:1 3 5
```

SenseStudy實驗

SenseStudy課程平臺“人工智能入門（下）”： 實驗6 – 2 分詞和列表

```
text = readtext("./nlp/data/textbooks/grade0/text0.txt")
words = splitwords(text)
print(words)

print(len(words))

for item in words[0:5]:
    print(item)
```

詞頻

經過分詞，我們將一篇文章分解成了一個個最小的語義單位——單詞。不同單詞按照一定的順序組合在一起，就可以表達出某種含義。

從這個角度思考，當我們要判斷一篇**文章的主題或者其它屬性**的話，我們是否可以從組成它的單詞出發來做出判斷呢?

Q：下面是分別從兩篇文章中截取的段落，同學們試著判斷一下哪個更難。

段落 1:

Today's story is about Zhu Hui, a student from Shenzhen. He's now studying in the United States. He's living with an American family in New York. Today is the Dragon Boat Festival. It's 9:00 a.m. and Zhu Hui's family are at home. His mom and aunt are making zongzi. His dad and uncle are watching the boat races on TV.

段落二:

Good morning to all our visitors from 2008. First we're going to examine one of the latest forms of communication among our space citizens. No more typists working on a typewriter or computer! No more postage or postcodes! Messages can now be sent using a "thoughtpad". You place the metal band over your head, clear your mind, press the sending button, think your message and the next instant it's sent.

詞頻

Q：下面是分別從兩篇文章中截取的段落，並打亂了單詞順序，同學們試著判斷一下哪個更難。

段落一:
help needs school Mrs.Miller the not to or you It piano have play Please teach Do at call Can is the time on violin 555-3721 difficult The weekend music you the
段落二:
that stars directions according a threw widely Bang the create bodies After all Big began However matter to and in other a atoms accepted to theory and universe to began with form combine that

Q：說說你判斷段落難度的依據是什麼？單詞順序對於對文章難度的判斷，影響大嗎?

對文章難度的判斷主要是單詞種類——如果有很多生詞（之前沒有學習過的單詞種類），就會覺得這篇文章很難；如果大部分都是熟悉的詞（之前學習過的單詞種類），就會覺得比較簡單。

單詞出現的先後關係並不太影響我們對一篇文章難度的判斷。

詞頻

在忽略了順序後，對一篇文章分詞得到的結果就可以看成一些無序的單詞。

我們關心的是，每種單詞一共出現了多少次，

以及單詞出現的頻次，也就是**詞頻**（單詞出現次數除以總單詞數）。

試一試：同學們嘗試對下麵的段落，統計表格中單詞的出現次數與段落的單詞總數:

Do you want a friend whom you could tell everything to, like your deepest feelings and thoughts? Or are you afraid that your friend would laugh at you, or would not understand what you are going through? Anne Frank wanted the first kind, so she made her diary her best friend.

單詞	you	the	friend	her	feelings	thoughts
出現次數						
詞頻						

詞頻統計的存儲問題

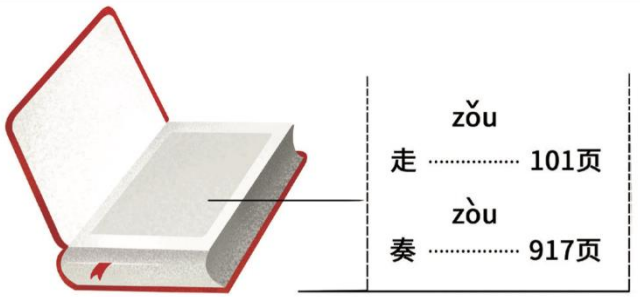
面對大量的文本，手動統計詞頻既麻煩又低效，可以編寫程式讓電腦自動統計。

思考一下詞頻統計的特點，需要將**單詞**和**對應的詞頻**成對存儲。

一個辦法是用一個長度為 2 的列表表示一對資料，然後將這些列表嵌套地存放在一個更大的列表中。效果如下:

[['you', 0.1], ['friend', 0.06], ..., ['thoughts', 0.02]]

這樣的結構可以很好地存儲詞頻，但是當我們需要查詢某個單詞對應的詞頻時，就需要依次去比對，很麻煩。為此，需要 引入一個新的資料類型 — **字典**。



字典

我們常用的字典記錄了每一個字的音韻、 解釋、例句、用法等。 字典的索引中已經把字按照一定的規則(拼音、筆劃等)排列， 很容易根據規則定位到某個字在索引中的位置，進而查詢到該字所在的頁碼。

類似地，Python 中的字典也存放著一個個關鍵字及其對應的內容， 我們稱之為**鍵值對**(如下表)。

關鍵字	值
“you”	0.1
“friend”	0.06
“thoughts”	0.02
……	……

字典的創建

(1)創建字典

形式：變數 = { 關鍵字 1：值 1，關鍵字 2：值 2，……}

創建字典時，可以用一對大括弧 {} 將一系列鍵值對包含起來。

其中，每一個鍵值對都用 關鍵字：值 的形式表示。（需要注意的是，關鍵字不能重複。）

練習：在之前的練習中，我們已經統計出了一些單詞的詞頻。請同學 們創建一個字典來存儲‘you’, ‘the’, ‘friend’的詞頻信息。

```
word_freq_dict = {'you': 0.1, 'the': 0.2, 'friend': 0.05} # 創建字典
```

字典中關鍵字的查詢

(2)查詢關鍵字

形式：字典 [關鍵字]

當我們需要查詢字典中某個關鍵字的內容時，只需要用類似於清單的索引方法，這裡用一對方括弧 [ ] 將關鍵字括起來，放在字典類型資料的後面，表示按照這個關鍵字去字典中取出對應的內容。

練習：從前面創建的字典 word_freq_dict 中獲取‘you’的詞頻。

```
freq = word_freq_dict['you'] # 查詢並獲取字典中某個關鍵字對應的值
print(freq) #列印詞頻的值
# 輸出:0.1
```

字典中鍵值對的插入和修改

(3)鍵值對的插入和修改

前面創建字典時，我們直接給了三個鍵值對。實際上，我們很難在一開始就知道要在字典中存放的全部內容。因此，需要鍵值對的插入和修改功能。

形式：字典 [關鍵字] = 值

和查找關鍵字的方式類似，我們用一對方括弧 [ ] 將要插入的關鍵字 括起來，放在等號左邊，等號右邊是對應的值。

- 如果該關鍵字已經存在， 那麼就會把對應的值修改為新給出的值；
- 如果該關鍵字之前不存在，就新建一個鍵值對。

字典中鍵值對的插入和修改

練習1： 把‘her’的詞頻資訊加入到前面創建的詞頻字典 word_freq_ dict 中。

```
word_freq_dict['her'] = 0.06 # 向字典中添加鍵值對
```

練習2： 把詞頻字典 word_freq_dict 中‘you’的詞頻改為 0.3 。

```
word_freq_dict['you'] = 0.3 # 修改字典中某個鍵對應的值
```

判斷某個關鍵字是否在字典中

(4)判斷某個關鍵字是否在字典中

形式：關鍵字 in 字典

如果關鍵字在字典中，那麼上面的語句就為真；否則，就為假。

練習：分別判斷‘you’和‘he’在不在詞頻字典 word_freq_dict 中。

```
print(“you”in word_freq_dict) # 判斷關鍵字“you”是 否在字典中
# 輸出:True
print(“he”in word_freq_dict) # 判斷關鍵字“he”是 否在字典中是否在字典中
# 輸出:True
```

字典的遍歷

(4)字典的遍歷

當我們已經得到了一個字典，想要對裡面所有鍵值對全部訪問一遍時，可以使用 for 迴圈結構以及字典的 **items 方法**。

item 的意思是一項， 調用該方法可以將字典中的每一個鍵值對作為一項返回。

形式：

```
for 變數 1，變數 2 in 字典.items( ):
    循環體
```

結合 for 迴圈， 就可以每次得到一組鍵值對，用變數 1 存放**關鍵字**，變數 2 存放**值**。在循環體中，我們可以使用變數 1 和變數 2。

字典的遍歷

練習：使用字典的遍歷方法，將詞頻字典 word_freq_dict 中的資訊列印出來。

```
for key, value in word_freq_dict.items( ): # 使用 for 迴圈遍歷字典的鍵值對
print(key, ':', value) # 列印出關鍵字和值
# 輸出：  'you': 0.3
          'the': 0.2
          'friend': 0.05
          'her': 0.06
```

自動統計詞頻

學習完字典類型，我們不難想到，可以利用字典實現自動統計詞頻的功能。文章中的每種單詞都是一個關鍵字，對應的值就是單詞的詞頻。

案例：假設一篇文章分詞後的單字清單為 words，請計算出該文章中所有單詞的詞頻。

```
freq_dict = { } # 創建一個字典用於存儲詞頻資訊
for word in words: # 對單字清單進行遍歷
    if word in freq_dict: # 判斷單詞是否已經在詞頻字典中
        freq_dict[word] += 1 # 把詞頻字典中該單詞對應的值加 1
    else:
        freq_dict[word] = 1 # 將該單詞作為關鍵字存入字典，值 為1
num_words = len(words) # 獲取單字清單的總詞數
for word, count in freq_dict.items( ): # 對詞頻字典進行遍歷
    freq_dict[word] = count/num_words # 用當前單詞的個數除以單詞總數
```

SenseStudy實驗

SenseStudy課程平臺“人工智能入門（下）”： 實驗6 – 3 字典和詞頻統計

```
word_freq_dict = {'you': 0.098, 'the': 0.020, 'friend': 0.059}
print(word_freq_dict)
print(word_freq_dict['you'])
word_freq_dict['you'] = 0.088
print(word_freq_dict)
print('you' in word_freq_dict)
print('he' in word_freq_dict)

for key, value in word_freq_dict.items():
    print(key + ':' + str(value))

def word_freq(words):
    freq_dict = {}
    for word in words:
        if word in freq_dict:
            freq_dict[word] += 1
        else:
            freq_dict[word] = 1
    for word, freq in freq_dict.items():
        freq_dict[word] = freq / len(words)
    return freq_dict

text = readtext("./nlp/data/textbooks/grade0/text0.txt")
words = splitwords(text)
freq_dict = word_freq(words)
print(freq_dict)
```

文章難度分級

本章的項目是要實現文章難度的自動分級。同學們思考一下，你在閱讀過程中是如何判斷文章難度的呢?以下面兩篇文章為例，同學們能夠分辨出哪一個難度更大嗎?

STUDENTS WHO VOLUNTEER

Mario Green and Mary Brown from Riverside High School give up several hours each week to help others.

Mario loves animals and wants to be animal doctor. He voluneers at an animal hospital every Saturday morning. Mario believes it can help him to get his future dream job. "It' s hard work," he says, "but I want to learn more about how to care for animals. I get such a strong feeling of satisfaction when I see the animals get better and the book of joy on their owners' faces."

Mary is a book lover. She could read by herself at the age of four.Last year, she decided to try out for a volunteer after-school reading program. She still works there once a week to help kids learn to read." The kids are sitting in the library,but you can see in their eyes that they' re going on a differernt journey with each new book. Volunteering here is a dream come ture for me. I can do what I love to do and help others at the same time.

COMMUNICATION: NO PROBLEM?

Yesterday. another student and I, representing our university' s student association, went to the Capital International Airport to meet this year' s international students. They were coming to student at Beijing University. We would take them first to their dormitories and then to the student canteem. After half an hour of waiting for their fight to arrive, I saw several young people enter the waiting area looking around curiously. I stood for a minute watching them and then went to greet them.

The first person to arrive was Tony Garcia form Colombia, closely followed by Julia Smith from Britain. After I met them and then introduced them to each other, I was very surprised. Tony approached Julia touched her shoulder and kissed her on the cheek!She stepped back appearing surprised and put up her hands,sa if in defence. I guessed that there was probably a major misunderstanding. Then Akira Nagata from Japan came in smiling, together with George Cook from Canada. As they were introduced. George reached his hand out to the Japanese student. Just at that moment, however, Akira bowed so his nose touched George' s moving hand. They both apologized-another cultural mistake!

根據**"難詞"的多少**來判斷**文章的難易**

單詞難度的量化

Q：怎樣給單詞的難度進行量化呢?

英文課本背後的生詞表能夠給你啟發嗎?

隨著年級的增加，生詞表的難度是遞增的嗎?

- 一個單詞屬於哪個年級的生詞表和它的難度有著緊密的聯繫。

一年級生詞表中的單詞難度可以近似認為是 **1 級**，

二年級生詞表中的單詞難度可以近似認為是 **2 級**，

.

.

.

八年級生詞表中的單詞難度可以近似認為是 **8 級**，

高三生詞表中的單詞難度可以近似認為是 **12 級**。

對單詞的難度進行**量化**

文章難度的判斷

Q：電腦如何判斷文章的難度呢?

總結判斷一篇文章難度的步驟：

1.將文章每個單詞依次取出；

Q：如何得到這 12 份生詞表呢?

2.對照 12 個年級的生詞表，找到單詞對應的難度等級；

3.計算各個難度等級單詞所占的比例；

4.根據難詞的比例判斷出整篇文章難度等級。

生詞表的構建

可以把電腦當成一個沒有學習過英語的人，然後按年級從低到高讓它依次學習英文課文。

一年級的課文中，所有單詞對他來說都是生詞。 → 所有單詞 納入一年級的生詞表中。

二年級的課文中，很多單詞在一年級時見過，只有一部分單詞是陌生的。 → 除去在一年級生詞表中出現的單詞納入二年級的生詞表中。

⋮

以此類推，電腦可以很快的把從小學到高中的全部課文學習一遍，並計算出每個年級的生詞表。

生詞表的構建

生詞表可以用前面所學的字典類型來表示(如下表)。

- 關鍵字是單詞，
- 對應的值表示該單詞屬於哪個年級的生詞表(也可以近似地認為是它的難度等級) 。

關鍵字	值
“you”	1
“friend”	3
“thoughts”	10
……	……

每個單詞對應的數值就等於該單詞首次出現的年級數。

生詞表的構建

練習： 假設我們有 3 個文字檔，每個檔裡面存儲了一篇英文課文， 這些檔的路徑為:

["/path/to/text1.txt","/path/to/text2.txt","/path/to/text3.txt"]

這些文章對應的年級為: [1,2,3]

請同學們構建生詞表。

```
paths = ["/path/to/text1.txt", "/path/to/text2.txt", "/path/to/text3.txt"]# 創建列表，存放課文對應的文字檔路徑
grades = [1,2,3]      # 創建列表，存放上述課文對應的年級序號
diff_level = {}       # 創建字典，存放單詞難度等級
# 將paths和grades中的元素同時進行遍歷
for path, grade in zip(paths, grades):
    text = nlp.readtext(path)    # 讀取路徑path對應的文字檔內容
    words = nlp.splitwords(text) # 對text存儲的文本進行分詞
for word in words:          # 對單字列表words進行遍歷
    # 如果當前單詞在難度等級字典中且它在字典中對應的值小於或等於當前的年級序號，就繼續循環
    if (word in diff_level) and (diff_level[word] <= grade):
        continue
      # 否則，就將當前單詞在字典中對應的值修改為當前年級序號
    else:
        diff_level[word] = grade
```

文本特徵

監督學習所學習的是從**一種數據 A**，到**另一種資料 B** 的**映射**（或者說函數）。

課文難度分級別就是從 **課文** ，到 **難度等級** 的**映射**。

很大程度上決定於不同難度單詞所占的比例。

用不同難度詞彙的出現頻率作為**文本的特徵**，用來進行文章難度分級是一個可行的方案。

文章難度分級

練習： 假設一篇文章分詞後的單字列表為 words，生詞表為 diff_level，請統計出文章中不同難度詞彙的出現頻率。

```
def get_freq_per_diff(words, diff_level):# 創建清單，保存每種難度等級的單詞在單字清單中出現的頻率
    freq_per_diff = [0,0,0,0,0,0,0,0,0,0,0,0]
    for word in words:        # 對單字清單words進行遍歷
        if word in diff_level: # 如果當前單詞word在diff_level中
            level = diff_level[word] # 就找出該單詞對應的難度等級
            freq_per_diff[level - 1] += 1 # 並將freq_per_diff對應位置處的值加1，注意：清單位置索引從0開始，
因此level減1才是位置索引
        else:
            continue   # 否則，就繼續迴圈
    num_words = sum(freq_per_diff) # 求出freq_per_diff所有元素的和，也就是所有難度等級單詞總數
    i = 0  # 創建整數i，作為計數器
    for ferq in freq_per_diff:        # 對freq_per_diff進行遍歷
        freq_per_diff[i] = freq / num_words # 用當前難度單詞的個數除以所有難度等級單詞總數
        i += 1                        # 計數器加1
return freq_per_diff          # 返回每種難度等級的單詞在單字清單中出現的頻率
```

SenseStudy實驗

SenseStudy課程平臺“人工智能入門（下）”： 實驗6 – 4 智能課文分析1

```
textbooks_data = load_textbooks_data()
print(textbooks_data)
print(len(textbooks_data))

def get_diff_level(path_grade):
    diff_level = {}
    for path, grade in path_grade:
        text = readtext(path)
        words = splitwords(text)
        #grade = int(grade)
        for word in words:
            if word in diff_level and diff_level[word] <= grade:
                continue
            else:
                diff_level[word] = grade
    return diff_level

diff_level = get_diff_level(textbooks_data)
print(diff_level)
save_private(diff_level, "./data/tmp/diff_level")
diff_level = load_private("./data/tmp/diff_level")
print(diff_level)
```

SenseStudy實驗

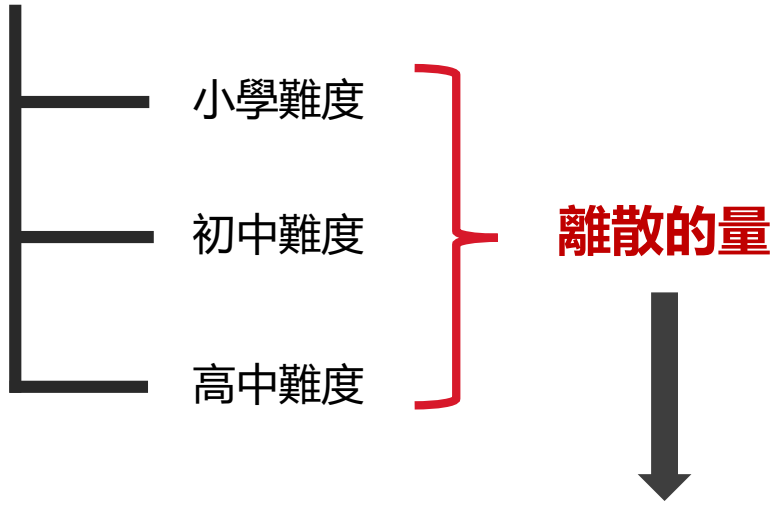
SenseStudy課程平臺“人工智能入門（下）”： 實驗6 – 5 智能課文分析2

```
diff_level = load_private("./data/tmp/diff_level")

def extract_features(path, diff_level):
    text = readtext(path)
    words = splitwords(text)
    grade_freq = [0]*12
    for word in words:
        if word in diff_level:
            grade = diff_level[word]
            grade_freq[grade] += 1
        else:
            continue
    num = sum(grade_freq)
    for i in range(12):
        grade_freq[i] /= num
    return grade_freq

grade_freq = extract_features("./nlp/data/reading/train/text0.txt", diff_level)
print(grade_freq)

grade_freq = extract_features("./nlp/data/reading/train/text1.txt", diff_level)
print(grade_freq)
```

分類

有了不同難度詞彙的出現頻率作為**文本特徵**。

利用機器學習的演算法，建立起 **文本特徵** 到 **文章難度等級** 的**映射**。



用“分類”的方法來解決。

Q：這個分類問題和我們之前學習的鑽石高低淨度分類問題有什麼不同?

文章難度等級分類是一個**多分類**的問題，而鑽石高低淨度分類是個**二分類**問題。

分類

先將訓練資料中高中的文章全部拋開，僅設計一個區分一篇文章是小學難度還是初中難度的分類器。

1.將文章難度進行數位化。

這裡，我們可以令小學難度為 0， 初中難度為 1。

2.確立一個帶參數的預測函數。

$$f(x_1, \ldots, x_{12}) = a_0 + a_1 x_1 + a_2 x_2 + \ldots + a_{11} x_{11} + a_{12} x_{12}$$

其中，a_0到 a_{12}是該函數的參數。這就是我們所熟悉的線性分類器。

用線性分類器對課文難度進行分類

例題： 假設我們有 4 篇文章，並且已經分別計算出了不同難度詞彙的出現頻率這個特徵，存放在 xs 中。這些文章對應的難度等級為:

[0,0,1,1]

利用線性分類器對課文難度進行分類。

```
model = LinearClassifier()# 創建線性分類器
xs = []                   # 數據 4篇課文形成一個4行12列的二維陣列
ys = [0, 0, 1, 1]         # 標籤
model.train(xs, ys)       # 用資料和標籤訓練分類器
```

SenseStudy實驗

SenseStudy課程平臺“人工智能入門（下）”： 實驗6 – 6 智能課文分析3

```
diff_level = load_private("./data/tmp/diff_level")
train_data = load_train_data()
print(len(train_data))
print(train_data)
test_data = load_test_data()
print(len(test_data))
print(test_data)

features = []
labels = []

def get_feats_labels(data, diff_level):
    features = []
    labels = []
    for path, label in data:
        features.append(extract_features(path, diff_level))
        labels.append(int(label))
    return features, labels

train_feats, train_labels = get_feats_labels(train_data, diff_level)
print("train_feats:", train_feats)
print(train_labels)
save_private([train_feats, train_labels], "./data/tmp/train_features")

test_feats, test_labels = get_feats_labels(test_data, diff_level)
print(test_feats)
print(test_labels)
save_private([test_feats, test_labels], "./data/tmp/tests_features")
```

SenseStudy實驗

SenseStudy課程平臺“人工智能入門（下）”： 實驗6 – 7 智能課文分析4

```
diff_level = load_private("./data/tmp/diff_level")
train_data = load_binary_train_data("primary", "junior")
print(len(train_data))
print(train_data)
train_feats, train_labels = get_feats_labels(train_data, diff_level)
print(train_labels)
test_data = load_binary_test_data("primary", "junior")
test_feats, test_labels = get_feats_labels(test_data, diff_level)
print(test_labels)

save_private([train_feats, train_labels], "./data/tmp/pri_jun_train_features")
save_private([test_feats, test_labels], "./data/tmp/pri_jun_test_features")

model = linear_classifier()
model.train(train_feats, train_labels)
pred_y = model.pred(test_feats)
print(test_labels)
print(pred_y)
acc = accuracy(pred_y, test_labels)
print(acc)
```