多媒體程式設計
文字資料處理

Instructor: 馬豪尚

文本的前處理和正規化

- 架構分割(Structure Segment)
  - 可以將一篇文章根據不同需求切割成段落或句子。
- 記號化(Tokenize)
  - 可以將輸入的字符串分割為記號、進而將記號進行分類的過程。
- 文字正規化 (Normalize)
  - Lemmatization
  - Stemming

架構分割(Structure Segment)

- 簡單的文本可以根據符號來切分
  - 。！？...
  - str.split(delimiter, text)
- 複雜的文本可以透過學習的方法來切割

記號化(Tokenize)

- 資料清理
  - 去除標點符號
  - 去除stop words
- 斷詞
  - N-gram
  - 字典斷詞
  - 統計模型斷詞

正規表達式

- 用簡單字串來描述、符合文中全部符合指定格式的字串，現在很多文字編輯器都支援用正規表達式搜尋、取代符合指定格式的字串。
- 保留規則運算符
  - . ^ $ * + ? { } [ ] \ | ( )
- 用\來轉義
  - '\(abc\)', '\{apple\}'
- 用r來表示此字串為regular expression)
  - r'(abc)', r'{apple}'

維基百科: https://zh.wikipedia.org/zh-tw/%E6%AD%A3%E5%88%99%E8%A1%A8%E8%BE%BE%E5%BC%8F

正規表達式

- 用[起始字元-結束字元]來表示集合
  - [0-9], [a-z], [A-Z], [a-zA-Z0-9]
  - [x-z]
- 除了指定開始字元-結束字元也可以搭配個別字元使用
  - [6-9-z]　#or
  - [6-9]-z　#and
  - [A-Z]he
- 不要用模稜兩可的表示法
  - [a-Z]
  - [0-z]

正規表達式

- 用^表達not in
  - [^0-9]: 非數字
  - [^a-z]: 非小寫字母
- 用.來代表任意字符
  - .這個符號可以當作萬用字符，可以比對任何的字符，像是字、特殊符號、空格等等
- 用|來表達或的規律
  - 有兩個expression想要同時比對就可以使用 | 這個特殊運算符
  - [dogs|penguins]

正規表達式

- 常用的字符集
  - \d -> 比對數字 相當於[0-9]
  - \D -> 比對非數字 相當於[^0-9]
  - \s -> 比對空格
  - \S -> 比對非空格
  - \w -> 任何字母與數字
  - \W -> 任何非（字母與數字）

正規表達式

- 比對多次-量詞
  - ? → 出現一次或是沒出現
  - * → 沒出現或是出現無限次
  - \+ → 出現1次到無限次
  - {n, m} → 出現n~m次

re模組

- 編譯正規表達式
  - pattern = re.compile(r'正規表達式')
    - pattern.match(text)
    - pattern.search(text)
- 如果不想編譯正規表達式
  - re.match(r.'正規表達式'), text)
  - re.search(r.'正規表達式'), text)

re模組

- match(): 從頭開始比對找到符合正規表達式的整個字串
- search(): 搜尋符合正規式表達的字串

Example

```
text = 'Hello world. This is an apple.’
m = re.match(r'H.+', text)
print(m.group(0))  #'Hello world. This is an apple.
m = re.match(r’w.+', text)
print(m.group(0)) #Error, None type
m = re.search(r'w.+', text)
print(m.group(0))  #'world. This is an apple.
```

re模組

- findall():搜尋符合正規式表達的每一個部分字串
- sub('符合字串', '新字串', text): 將所有符合的字串取代成新字串

Example

```
text = 'Hello world. This is an apple.'
substrs = re.findall(r'\w+', text)
print(substrs) # ['Hello', 'world', 'This', 'is', 'an', 'apple']
new_text = re.sub('o', '-', text)
print(new_text) # 'Hell- w-rld. This is an apple'.
```

re模組

- subn(‘符合字串’, ‘新字串’, text, 替代次數): 將所有符合的字串取代成新字串
  - 不指定替代次數為全替代

Example

```
text = 'Hello world. This is an apple.'
new_text, sub_count = re.subn ('o', '-', text, 1)
print(new_text, sub_count ) #Hell- world. This is an apple. 1
new_text, sub_count = re.subn ('o', '-', text)
print(new_text, sub_count ) #Hell- w-rld. This is an apple. 2
```

練習1: 簡易斷詞 (Uni-gram)

- 讀取entext.txt
- 斷句
- 去除標點符號/特殊符號
  - 用正規表達式去除
- 全部改小寫
- 簡單斷詞
  - 將每個單詞切割出來
- 將結果寫入entext_out.txt

NLTK (Natural Language Tool Kit)

- import nltk
- nltk.download('punkt')
- 斷詞
  - nltk.word_tokenize(string)
- 斷句
  - nltk.sent_tokenize(paragragh)

NLTK正規表達式客製斷詞

- from nltk.tokenize import RegexpTokenizer
- RegexpTokenizer() 的參數跟用法：
  - 第一個參數會是你希望它留下來的東西。也就是說，你要告訴它每次遇到"非什麼條件的東西"就要停下來分割字串
  - 第二個參數gaps告訴它我們需不需要保留第一個參數指涉的東西，也就是遇到的分割符號(false: 不需要保留)

NLTK正規表達式客製斷詞

- 自定義規則去除標點符號
  - tokenizer = RegexpTokenizer(r'\w+', gaps = False)
  - clean_sent = tokenizer.tokenize(string)
- 保留不是標點符號的符號
  - tokenizer2 = RegexpTokenizer(r'\w+|\'\w+', gaps = False)
  - clean_sent2 = tokenizer2.tokenize(string)
- 保留切割用的字符串
  - tokenizer3 = RegexpTokenizer(r'\w+|\'\w+', gaps = True)
  - clean_sent3 = tokenizer3.tokenize(string)

NLTK正規表達式客製斷詞

- 去除stopword
  - from nltk.corpus import stopwords
  - nltk.download('stopwords')
  - stopword = stopwords.words('english')

文字正規化

- 詞幹提取(Stemming)
  - 詞幹提取是去除詞綴得到詞根的過程
  - 較偏向基於規則(rule-based)的方式去拆解單詞
  - 後綴去除法
    - 如果詞的結尾是「ed」,則去掉「ed」
    - 如果詞的結尾是「ing」,則去掉「ing」
    - 如果詞的結尾是「ly」,則去掉「ly」

文字正規化

- 詞形還原(Lemmatization)
  - 首先確定詞彙的發音部分，然後根據發音的部分確定詞彙的根，停頓詞規則隨著單詞的發聲部分的改變而改變
  - 動詞形式、形容詞形式、名詞形式、名詞複數、過去分詞...將不同形式的字還原成同一個字
  - 降低文本詞彙的複雜度
- Example
  - play, played, playing, plays
  - baby, babies
  - feet, foot

文字正規化

- Stemming
  - 計算速度快
  - 容易over stemming
  - 在處理特殊的詞上效果較差(run/ran)
- Lemmatization
  - 相較於 Stemming 會更精準
  - 處理的時間較長

NLTK stemming

- 載入模組
  - from nltk.stem.porter import PorterStemmer
- 創建 PorterStemmer 物件
  - ps=PorterStemmer()
- 使用已創的物件去做詞幹提取
  - ps.stem(words)

NLTK lemmatizer

- 載入模組
  - from nltk.stem import WordNetLemmatizer
  - nltk.download('omw-1.4')
- 創建WordNetLemmatizer物件
  - wnl = WordNetLemmatizer()
- 使用已創的物件去做詞形還原
  - wnl.lemmatize(words)

NLTK 詞性標註

- 載入模組
  - from nltk.stem import WordNetLemmatizer
  - nltk.download('averaged_perceptron_tagger')
- nltk詞性標註函式物件
  - nltk.pos_tag(token list)
- 詞性表
  - https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html

NLTK詞頻統計

- 載入模組
  - from nltk.probability import FreqDist
- 建立字典
  - fdist = FreqDist(result)
- 詞頻統計函式
  - fdist.most_common()
  - fdist.most_common(number)

NLTK語料庫

- Gutenberg
  - 是第一個提供免費的網路電子書平台，根據官方網站說明，project gutenberg 已經有超過 57,000本免費的電子書，NLTK 的 package 僅納入部分語料
- Brown
  - brown 語料庫是第一個百萬等級的電子語料庫(英文)， 1961 年由 Brown University 所整理，這個語料庫包含的字詞來自 500 個資料源，並參考資料源的種類做分類，例如：adventure 、news、reviews...等。
- Reuters
  - reuters 是路透社語料庫，涵蓋 10,788 個新聞文本，共有 90 個分類，例如：housing、income、tea...等。
- Inaugural
  - inaugural 是歷屆美國總統就職演說的語料庫，文本的命名方式是『年份+人名』，共有 56 個文本，最新一筆收錄的是 2009 年 Obama 的演說稿。

NLTK語料庫

- 載入語料庫(以gutenberg為例)
  - from nltk.corpus import gutenberg
- 查找語料庫當中的文本 id
  - corpus.fileids()
- 原始內容、單詞列表、句子列表
  - corpus.raw(fileids)
  - corpus.words(fileids)
  - corpus.sents(fileids)
- 語料庫內文本的分類屬性
  - corpus.categories(fileids)

練習2

- 基於NLTK套件建立一個英文文本詞頻統計程式
  - 下載NLTK語料庫內的任意一篇文章當作要處理的文件
  - 資料清理
    - 全部改成小寫
    - 去除非標點符號(但是要保留一般符號)
    - 去除stop words
  - 斷詞
  - 文字正規化
  - 詞頻統計並輸出