自然語言處理
分詞與文句分析

Instructor: 馬豪尚

文本資料前處理

- Data preprocessing and cleaning
  - 預處理資料可以減少雜訊並處理缺失值
  - 斷字、斷詞
- Relevance analysis(feature selection)
  - 刪除不相關或多餘的屬性
  - 移除stopwords,擷取有用資訊(TF-IDF)
- Data transformation
  - Generalize and/or normalize data
  - 轉成向量(Vector representation)

中文斷詞

- 基於詞典的斷詞方式，只要詞典中沒有收錄句子中的詞，那可能效果會非常差
- 大部份比較好的斷詞系統都是使用全切分方法，切分出與詞庫匹配的所有可能，然後再運用統計模型決定最好的切分結果

中文停用詞

- 在資訊檢索中，停用詞是從計算的角度來討論，會造成計算上的負擔或是降低搜尋準確度的詞，都可以當作停用詞。
- 在中文裡「停用詞」跟代名詞、助動詞、介系詞、連接詞等和文法、句法有關的「功能詞」相似。
- 是否去除停用詞要取決於處理文本的目的或應用
  - 希望可以最大化地凸顯手邊文本集的特徵時
  - 訓練文本的數量較小時
- 在中文的自然語言處理不一定需要去除停用詞

Jieba斷詞套件

- Jieba其實是簡體中文版本的中文斷詞系統
- 演算法大概可分成三個部份
  - 第一個部分是建立 Trie DAG 資料結構，快速算出全切分法所有合法的切分組合。
  - 採用了動態規劃查找最大概率路徑, 找出基於詞頻的最大切分組合。
  - 最後一步再使用 HMM 模型計算來辨識新詞。

Jieba斷詞套件

- 全模式 ：把句子中所有的可以成詞的詞語都掃描出來
- 精確模式 ：將句子最精確地切開，適合文本分析
- 搜索引擎模式 ：在精確模式的基礎上，對長詞再次切分，適合用於搜索引擎分詞
- paddle模式 :利用PaddlePaddle深度學習框架，訓練序列標註（雙向GRU）網絡模型實現分詞。同時提供詞性標註功能。目前paddle模式支持jieba v0.40及以上版本

安裝套件 → pip install jieba

paddle模式需安裝

pip install paddlepaddle-tiny==1.6.1

Jieba官方github https://github.com/fxsjy/jieba

Jieba斷詞套件

- 斷詞函數jieba.cut(text, cut_all=True, HMM=False, use_paddle=True)
  - 方法接受四個輸入參數:
    - 需要分詞的字符串
    - cut_all 布林參數用來控制是否採用全模式
    - HMM 布林參數用來控制是否使用 HMM 模型
    - use_paddle 布林參數用來控制是否使用paddle模式下的分詞模式，paddle模式採用延遲加載方式

Jieba斷詞套件-全模式

- seg_list = jieba.cut(text, cut_all=True)
  - 返回的結構都是一個可疊代的 generator，可以使用 for迴圈來獲得分詞後得到的每一個詞語
- seg_list = jieba.lcut(text, cut_all=True)
  - 返回的結構為一個list，可以使用 for 迴圈來獲得分詞後得到的每一個詞語
- Example output: ['我', '們', '在野', '野生', '動', '物', '園', '玩']

Jieba斷詞套件-精準模式

- seg_list = jieba.cut(text, cut_all=False)
  - 返回的結構都是一個可疊代的 generator，可以使用 for迴圈來獲得分詞後得到的每一個詞語
- seg_list = jieba.lcut(text, cut_all=False)
  - 返回的結構為一個list，可以使用 for迴圈來獲得分詞後得到的每一個詞語
- Example output: ['我們', '在', '野生', '動物園', '玩']

Jieba斷詞套件-paddle模式

- seg_list = jieba.cut(text, use_paddle=True)
  - 返回的結構都是一個可疊代的 generator，可以使用 for迴圈來獲得分詞後得到的每一個詞語
- seg_list = jieba.lcut(text, use_paddle=True)
  - 返回的結構為一個list，可以使用 for迴圈來獲得分詞後得到的每一個詞語

Jieba斷詞套件-搜尋引擎模式

- seg_list = jieba.cut_for_search (text)
  - 返回的結構都是一個可疊代的 generator，可以使用 for 循環來獲得分詞後得到的每一個詞語
- seg_list = jieba.lcut_for_search (text)
  - 返回的結構為一個list，可以使用 for 循環來獲得分詞後得到的每一個詞語
- 該方法適合用於搜索引擎構建倒排索引的分詞，會將有可能可以成詞的詞都分出來，粒度比較細

Jieba斷詞套件-詞典

- 載入自定義的詞典
  - jieba.load_userdict(file_name)
- 詞典格式
  - 一個詞佔一行
  - 每一行分三部分：詞語、詞頻（可省略）、詞性（可省略），用空格隔開，順序不可顛倒。
  - file_name 若為路徑或二進制方式打開的文件，則文件必須為 UTF-8 編碼。
  - Example: T恤 28 N

Jieba斷詞套件-詞典操作

- 直接加詞入現有詞典
  - Jieba.add_word(word, freq=None, tag=None)
- 刪除詞典內的詞
  - Jieba.del_word(word)
- 修改詞典內詞的頻率
  - Jieba.suggest_freq(segment, tune=True)

Jieba Tokenize

- 返回詞語在原文的起止和結束位置
- 函式
  - jieba.tokenize(u'字串')
  - 字串前面要加一個u (因為只能用unicode模式)
  - 開檔編碼一定要是utf-8

Jieba Tokenize 搜尋引擎模式

- 返回搜尋引擎模式詞語在原文的起止和結束位置
- 函式
  - jieba.tokenize(u'字串', mode='search')
  - 字串前面要加一個u (因為只能用unicode模式)
  - 開檔編碼一定要是utf-8

CKIP Transformer套件

- 此套件為一具有新詞辨識能力並附加詞類標記的選擇性功能之中文分詞系統。
- 分詞依據為此一詞彙庫及定量詞、重疊詞等構詞規律及線上辨識的新詞，並解決分詞歧義問題。
- 含有詞類標記，可附加文本中切分詞的詞類解決詞類歧義並猜測新詞之詞類。
- https://github.com/ckiplab/ckip-transformers

CKIP Transformer套件

- 載入模組
  - from ckip_transformers.nlp import CkipWordSegmenter, CkipPosTagger, CkipNerChunker
- 載入預訓練模型
  - ws_driver = CkipWordSegmenter(model="bert-base")
  - pos_driver = CkipPosTagger(model="bert-base")
  - ner_driver = CkipNerChunker(model="bert-base")
- 載入自定義模型
  - ws_driver = CkipWordSegmenter(model_name="path_to_your_model")
  - pos_driver = CkipPosTagger(model_name="path_to_your_model")
  - ner_driver = CkipNerChunker(model_name="path_to_your_model")

CKIP Transformer套件

- 斷詞
  - ws = ws_driver(text)
- 詞性標註
  - pos = pos_driver(ws)
- 命名實體識別
  - ner = ner_driver(text)

基礎文本字詞分析

Pointwise Mutual Information

- Pointwise Mutual Information是一種用於衡量兩個事件之間相互關聯性的統計變量
- 可以衡量兩個詞w_i和w_j之間的關聯程度

$$PMI(w_i, w_j) = \log\left(\frac{P(w_i \cap w_j)}{P(w_i)P(w_j)}\right)$$

NLTK 計算 PMI (1/2)

- 從nltk 套件中載入BigramCollocationFinder
  - from nltk collocations import BigramCollocationFinder
- 先進行斷詞
  - tokenized_corpus =[nltk.word_tokenize(doc) for doc in corpus]
  - doc為一個文本
  - corpus為整個語料庫
- 建立BigramCollocationFinder 物件
  - finder=BigramCollocationFinder.from_documents(tokenized_corpus)

NLTK 計算 PMI (2/2)

- 計算bigram 的PMI 值
  - pmi_scores = finder.score_ngrams(bigram_measures.pmi)
- 印出前N個PMI值最高的詞組

```
For bigram , pmi in pmi_scores[:N]
    print(f "PMI({bigram}): {pmi:.2f}")
```

TF-IDF （Term Frequency - Inverse Document Frequency）

- TF-IDF 是一種用於文字資訊檢索與探勘的常用加權技術，為一種統計方法，用來評估單詞對於文件的集合或詞庫中一份文件的重要程度
- TF（Term Frequency）
  - 這個單詞出現在該文件的次數/該文件的總字數

$$TF(t,d) = \frac{n_{t,d}}{\sum_{w \in d} n_{w,d}}$$

- IDF(Inverse Document Frequency)
  - 總文件數/這個單詞有出現的文件數

$$IDF(t,D) = \log(\frac{N}{|\{d \in D: t \in d\}|})$$

TF-IDF

TF=5/24=0.208，IDF=log(3/1)=0.477

TF=7/24=0.292，IDF=log(3/1)=0.477

TF=19/24=0.792，IDF=log(3/3)=0

TF=7/24=0.292，IDF=log(3/1)=0.477

Jieba-TFIDF關鍵詞

- 載入
  - import jieba.analyse
- 函式
  - jieba.analyse.extract_tags(sentence, topK=20, withWeight=False, allowPOS=())
    - sentence: 待提取的文本
    - topK: 返回幾個TF / IDF權重最大的關鍵詞，默認值為20
    - withWeight: 是否一併返回關鍵詞權重值，默認值為False
    - allowPOS: 僅包括指定詞性的詞，默認值為空，即不篩選

sklearn套件- TF-IDF關鍵詞(1/2)

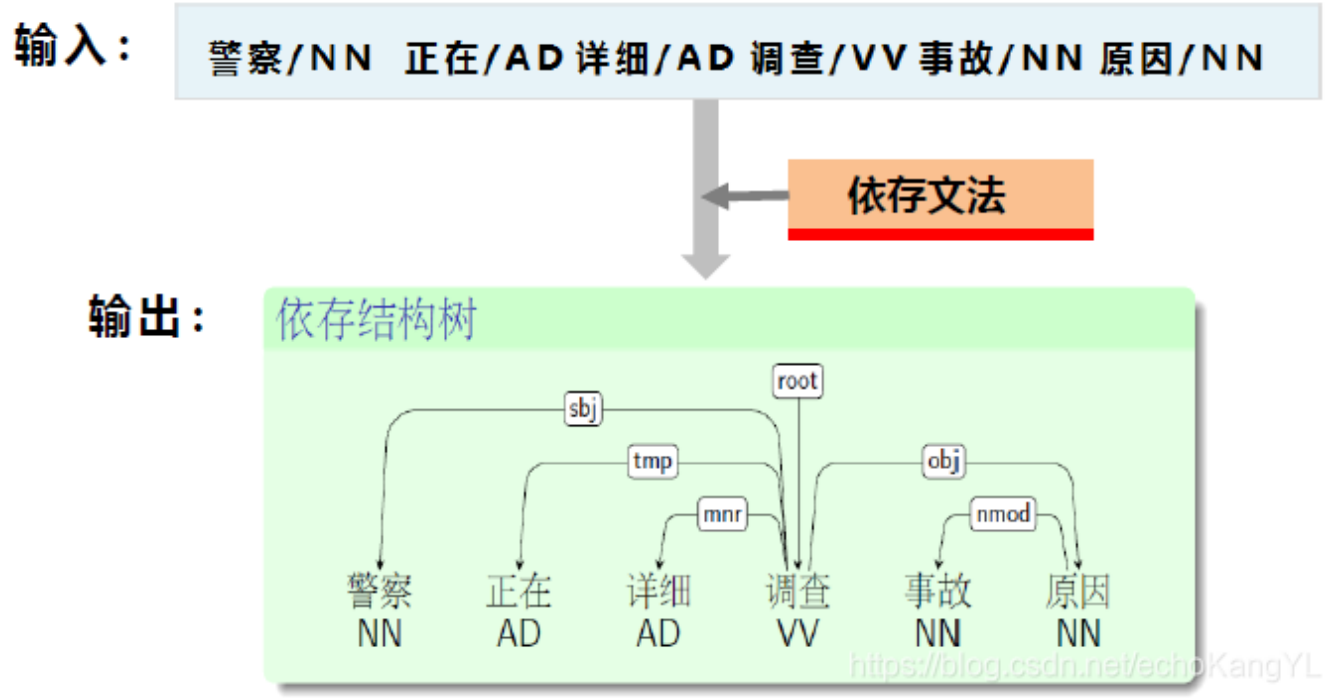
- 載入套件
  - from sklearn.feature_extraction text import TfidfVectorizer
- 建立計算TF-IDF物件
  - vectorizer = TfidfVectorizer()
- 對語料庫進行TF-IDF 計算，得到特徵矩陣X
  - X = vectorizer.fit_transform(corpus)
  - corpus為傳入的語料庫

sklearn套件- TF-IDF關鍵詞(2/2)

- 將特徵矩陣轉換成DataFrame
  - data ={'word': vectorizer.get_feature_names_out(), 'tfidf': X.toarray().sum(axis=0).tolist()}
    - # 取得特徵詞列表, #計算每個詞的TF-IDF 值
  - df= pd.DataFrame(data)
- 根據TF-IDF值降序排序
  - df_sorted= df.sort_values(by='tfidf', ascending=False)

句法分析-語法樹 Syntax Tree

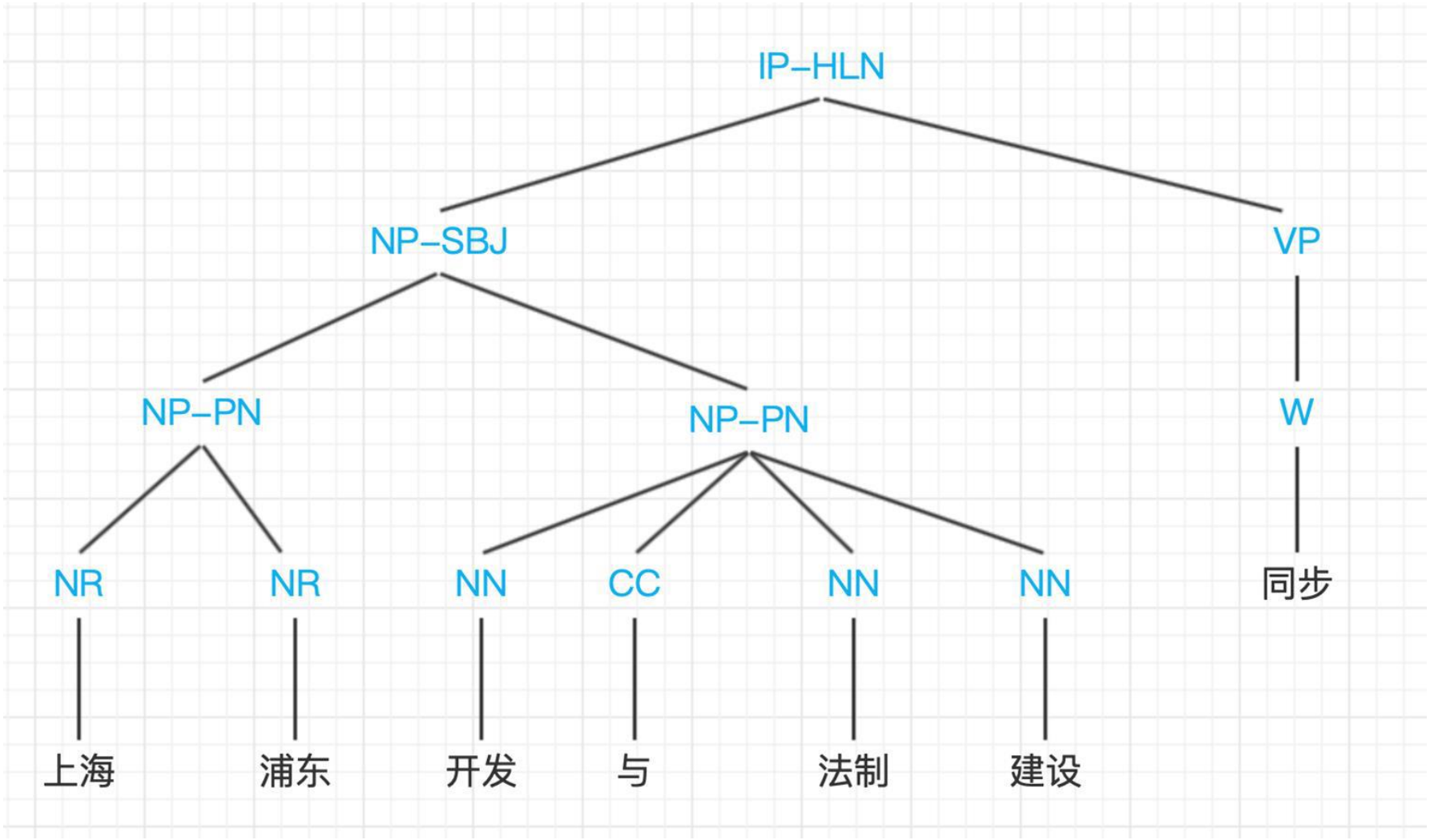
- 語法樹(Syntax Tree)也稱為句法樹(Parse Tree)或結構樹(Parse Tree)
- 是一種用於表示句子結構的樹狀結構，用於分析句子的語法結構和詞彙之間的關係



語法樹結構和組成成分

- 節點(Node)
  - 每個節點代表了句子中的一個詞或一個組合的詞。這些節點包括詞彙節點和非詞彙節點。
- 邊(Edge)
  - 邊表示節點之間的關係，通常用於連接不同節點 。
- 根節點(Root Node)
  - 整棵樹的 最頂層節點 ，它代表了整個句子 。
- 子節點(Child Node)
  - 一個節點下面連接的較低層級的節點稱為子節點 。
- 父節點(Parent Node)
  - 一個節點連接到上面的節點稱為父節點 。

語法樹



NTLK語法樹實例 (1/2)

- nltk 中載入pos_tag 函數，用於詞性標註
  - fom nltk import pos_tag
- 從nltk 中載入RegexpParserRegexpParser，用於正則表達式句法分析
  - from nltk import RegexpParser
- 斷詞和詞性標註
  - tokens = nltk.word_tokenize(text)
  - tagged_tokens= pos_tag(tokens)

NTLK語法樹實例 (2/2)

- 定義一個名為grammar 的文法，用於句法分析
  - grammar ="NP: {<DT>?<JJ>?< NN>}“
- 使用定義的文法初始化句法分析器
  - chunk_parser = RegexpParser (grammar)
- 進行句法分析
  - parse_tree = chunk_parser.parse(tagged_tokens)
- 繪製句法樹
  - parse_tree.draw ()

練習

- 用CKIP套件建立一個中文文本前處理程式
  - 資料清理
    - 斷詞
    - 詞性標註
    - 去除停用詞(某些詞性)
  - 詞頻統計
  - 分析PMI最高的前10個詞組
  - 分析TF-IDF分數前10高的詞