

## Sentimental Analysis 正反情緒分析

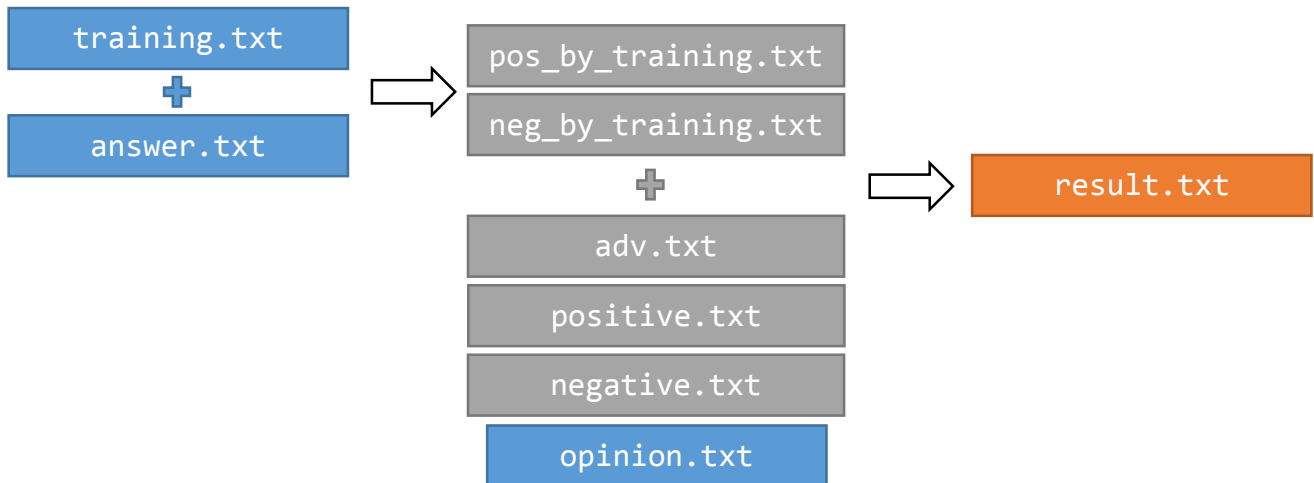
- Environment

Eclipse Java EE

JDK/JRE v1.6

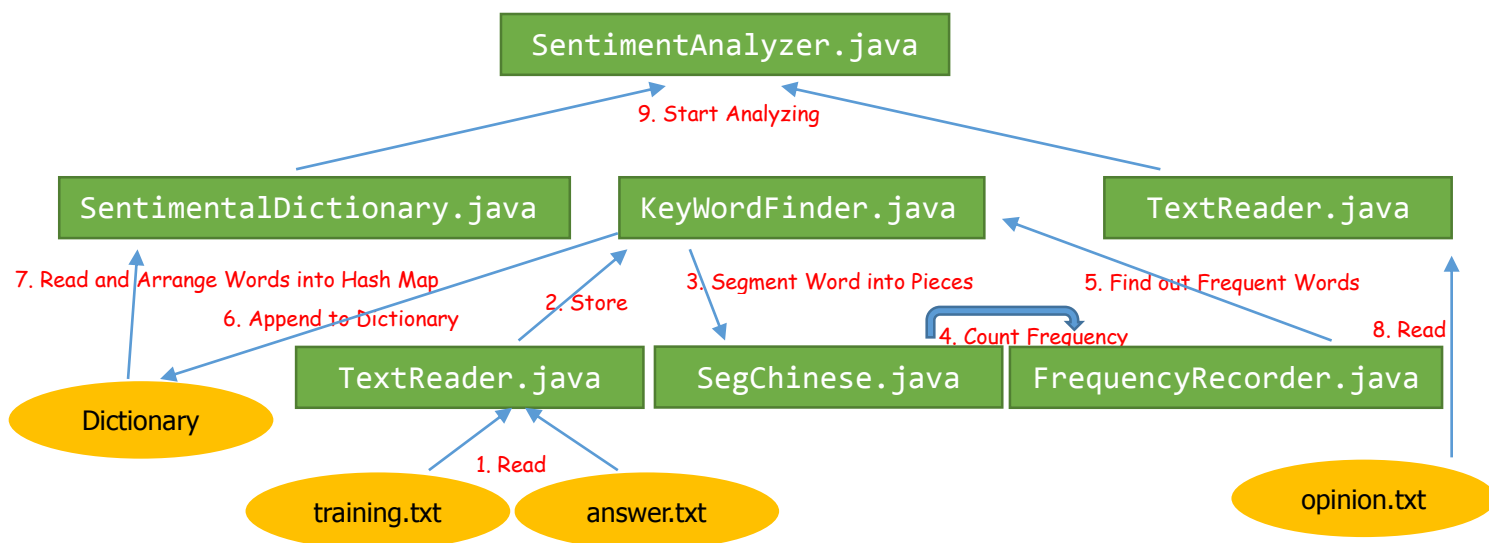
UTF-8 File Encoding

- Process



除了原有的外部情緒字典 ( 正、反、程度詞 )，利用 Training 找出其他特別的正反詞彙  
之後，根據句中「正反詞彙」出現的數量，搭配加重語氣的程度詞，給定一分數，作為評斷標準

- Frame Work



- Details about Training

- Step1 斷詞

- 先由標點符號斷句，並使用 Open Source 的 Library ( MMSEG )

- 實現最大匹配、最大單詞長度的分詞

- Reference: <http://function1122.blogspot.tw/2010/10/mmseg4j-java-55.html>

- Step2 計算各單詞出現次數 ( 頻率 )

- 得到以「詞」為單位的資料後，計算整份 Training Data 中，各單詞出現的字數 ( 頻率 )

- Step3 選擇一些在該類文章中，具代表性的正負面詞彙，加入字典

- 選擇的標準：SO 值 > 4.5，加入 Positive 字典；SO 值 < -4.5，加入 Negative 字典

- ※  $SO\_PMI(word)$

- $= PMI(word, POSITIVE) - PMI(word, NEGATIVE)$

- $= \log_2 \frac{P(word \& POSITIVE)}{P(word)P(POSITIVE)} - \log_2 \frac{P(word \& NEGATIVE)}{P(word)P(NEGATIVE)}$

- $= \log_2 \frac{P(word \& POSITIVE)P(NEGATIVE)}{P(word \& NEGATIVE)P(POSITIVE)}$

- 其中

- $P(POSITIVE)$ 代表正詞 ( 正評 ) 出現的概率， $P(word)$ 代表 $word$ 這個單詞出現的概率

- 而 $P(word \& POSITIVE)$ 代表 $word$ 與正詞 ( 正評 ) 「同時」出現的機率

- Details about Analyzing

- Step1 斷句

- 以標點、各式符號斷句 ( 不以分行斷句，因為一行視為一則評論或回覆 )

- Step2 找程度詞

- 將一個句子切分成小部分，判斷截斷後的詞彙是否屬於 Dictionary 中的程度詞

- 如果是，則將該句子的分數倍率乘以 2

- ※ Example

- 「這家旅館的爛服務非常差勁」會切成

- 「家旅館的爛服務非常差勁」、「這家旅館的爛服務非常差」...「常差」、「非常」...「這」

- 由長到短、後往前的截字方式 ( 避免長詞關鍵字沒先抓到，反而抓到短詞 )

- 抓到程度詞關鍵字後，會將倍率乘 2，並把關鍵詞從句子中刪除

- Step3 找正反面情緒用詞

- 截字、刪字方式同上一步，只是把截斷後的詞彙拿去 Positive、Negative Dictionary 中比對

- 比對後，如果是正面詞彙，分數+1，負面則-1 ( 搭配程度詞的倍率，可能變為±2 )

- Step4 找出 Shifter ( 不、沒 )

- 比對句子中剩餘的字彙，是否包含「不」或「沒」

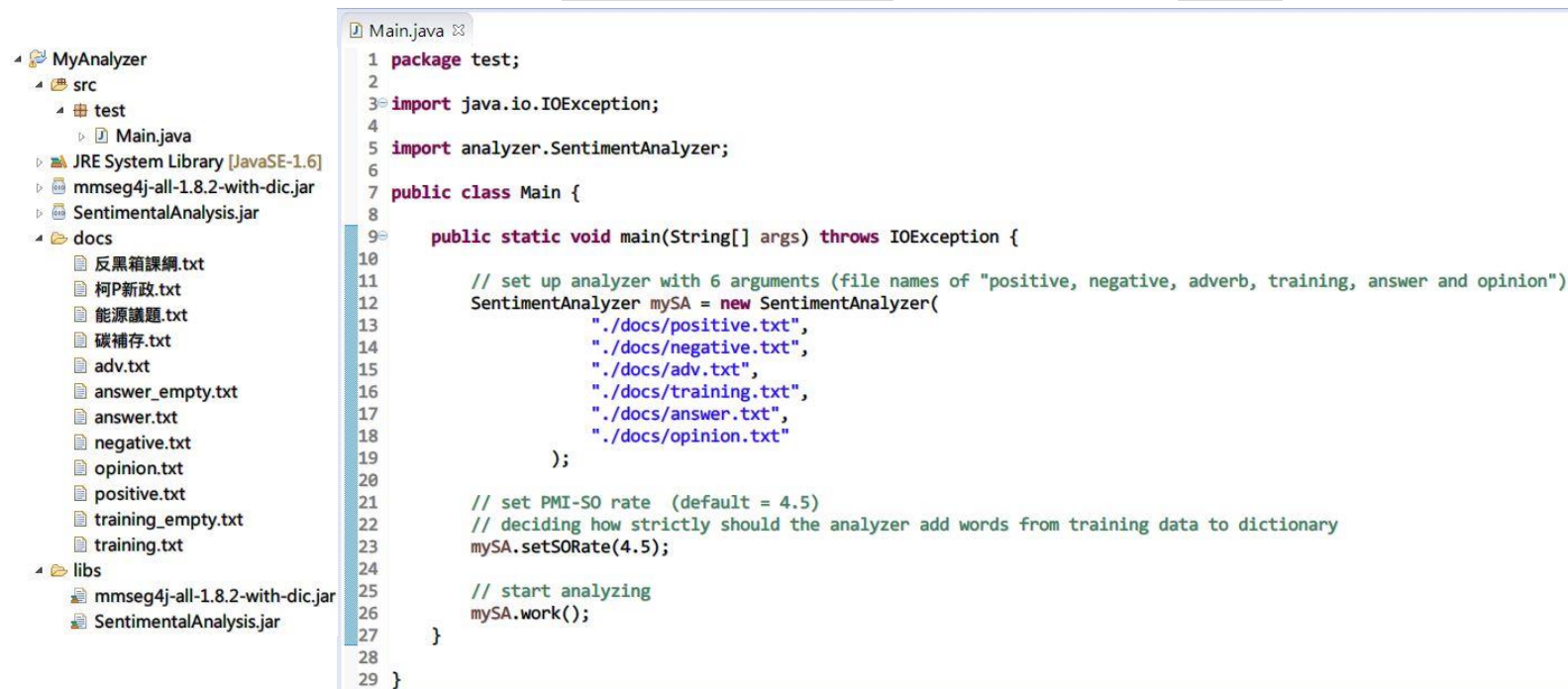
- 如果有，則將該句子的分數乘上-1

- Step5 判斷整則評論的正反傾向

- 整則評論的分數=各句子的分數加總，若評論分數  $\geq 0$ ，判為正面傾向，反之負面

## ● API

Source Code 已打包成 `SentimentAnalysis.jar`，外加 `mmseg4j-all-1.8.2-with-dic.jar`  
Library Setting 好之後，使用 `SentimentAnalyzer()` 建構子和 method - `work()` 來 run



The screenshot shows an IDE with a project named 'MyAnalyzer'. The project structure on the left includes a 'src' folder with a 'test' folder containing 'Main.java', and a 'docs' folder with various text files. The 'libs' folder contains 'mmseg4j-all-1.8.2-with-dic.jar' and 'SentimentAnalysis.jar'. The 'Main.java' file is open in the editor, showing the following code:

```
1 package test;
2
3 import java.io.IOException;
4
5 import analyzer.SentimentAnalyzer;
6
7 public class Main {
8
9     public static void main(String[] args) throws IOException {
10
11         // set up analyzer with 6 arguments (file names of "positive, negative, adverb, training, answer and opinion")
12         SentimentAnalyzer mySA = new SentimentAnalyzer(
13             "./docs/positive.txt",
14             "./docs/negative.txt",
15             "./docs/adv.txt",
16             "./docs/training.txt",
17             "./docs/answer.txt",
18             "./docs/opinion.txt"
19         );
20
21         // set PMI-SO rate (default = 4.5)
22         // deciding how strictly should the analyzer add words from training data to dictionary
23         mySA.setSORate(4.5);
24
25         // start analyzing
26         mySA.work();
27     }
28 }
29 }
```

## ● Input File Format

- Training 用的文字檔 預設為 `docs/training.txt`  
一則評論占一行，不加編號（若無，須建立空檔案）
- Training 的答案 預設為 `docs/answer.txt`  
行數與 `training.txt` 相同，一行一字，以半形大寫 P/N 來表示（若無，須建立空檔案）
- 正、反、程度字典 預設為 `docs/positive.txt`, `docs/negative.txt`, `docs/adv.txt`  
一個單詞（單字）占一行，不加編號
- 欲分析的評論 預設為 `docs/opinion.txt`  
格式與 Training 的檔案相同，一則評論占一行，不加編號

## ● Output File Format

分析後會於當前目錄產生 `result.txt`

`result.txt` 中每則評論的分析占 4 行：

- Line1 「NO.%d rate = %d (Positive)」 或 「NO.%d rate = %d (Negative)」
- Line2 原評論的斷詞結果
- Line3 「Keywords Found: 」+數個「%s(+1、-1 或 adv)」，為找到的關鍵字和其意義
- Line4 空行

`result.txt` 的檔尾會另列此次分析的資訊