人工智慧概論 HW2 報告

110550088 李杰穎

April 9, 2022

1 Preprocessing

在本次作業中，我主要實作了以下幾種 preprocessing 的方法，條列如下：

1. 將英文大寫轉成小寫: 使用 Python 內建的 `lower()` 函式，將字串文字全部轉成小寫
2. 移除 stopwords: 利用 nltk 提供的 stopwords 列表，將 stopwords 從字串中移除。
3. 移除 `<br />` HTML tag: 使用 Python 內建的 `replace()` 函式，將 `<br />` 用一個空白取代
4. 移除標點符號: 利用 for 迴圈檢查每一個 char 是否為標點符號，如果非標點符號則將其加進一個 list，最後再使用 `“”.join()` 來將 list 內元素轉為字串。檢查標點符號的部分則使用內建之 `string.punctuation` 來檢查。
5. Stemming (使用 nltk 內建之 SnowballStemmer): Stemming (詞幹提取) 是一種將詞彙去除後綴的方式。將單字進行 stemming 會讓模型不用處理額外的訊息，以下是一些經過 SnowballStemmer 處理後的單字。
   cared → care, university → univers, fairly → fair, easily → easili, singing → sing, sings → sing, sung → sung, singer → singer, sportingly → sport

在進行 preprocessing 時，會依照上面排列的順序進行這五個步驟。

下方為一英文句子通過以上 preprocessing 後的句子。

"It is a truth universally acknowledged that
 a single man in possession of a good fortune must be in want of a wife."
→ "truth univers acknowledg singl man possess good fortun must want wife"

2 Implement the bi-gram language model

本部分介紹 bi-gram model 的實作，因為大部分實作細節都可以在繳交的程式碼中看到，故在此我只大致說明實作內容。

1. 計算各 unigram 及 bigram 的出現頻率: 由於在計算 $P(w_i|w_{i-1})$ 時，需要同時知道 bigram 及 unigram 的出現頻率，故先 iterate 所有 document 並利用 dict 來統計出現的頻率。
2. 利用上一個步驟的頻率求出 $P(w_i|w_{i-1})$: 左述之條件機率可由下式得到:

$$P(w_i|w_{i-1}) = \frac{\text{count}(w_{i-1}, w_i)}{\text{count}(w_{i-1})} \tag{1}$$

3. 將所有算出的機率利用 Python 內建的 dict 資料結構，存為 `model[`w_{i-1}`][`w_i`]` 的形式，而 feature 則儲存各 bigram 的頻率 (即為 `feature[(`w_{i-1}`, `w_i`)]`)