Best Match Model

資管碩一 M10909112 石家安

運用工具 & 資料前處理說明

1. 透過 nltk 的 word.tokenize function，將 Documents 與 Queries 進行斷詞。
2. 運用 snowball_stemmer function 將詞性還原。
3. 使用 nlt.corpus 中 stopwords 篩選掉不帶有資訊的字詞，ex: the, a, and…。
4. 使用 numpy 做陣列操作。而 math、operator 做資料運算 ex:math.log。

TF 參數設定-Documents、Quries

```
for wc in doc_dict:
    # IDF
    self.docidf[wc] = self.docidf.get(wc, 0.0)+1.0
    # TF
    doc_dict[wc] = 1+math.log(doc_dict.get(wc, 0.0),2)
```

1. Documents TF 計算：
   計算該文章每個字出現的次數，並運用 Log Normalization (1+log(tf))概念平滑 TF 參數，以完成該 doc 各文字的 TF 計算。
2. Query TF 計算：
   計算 query 各文字出現的次數則是運用最原始的 TF 計算方式。

相似度計算-BM25

```
for w in self.queidf:
    if w in docTFtemp:
        ctd = 0.765*docTFtemp[w]/((1-b)+b*len/avglen)
        first = ((K1 + 1) * (ctd+delta)) / (K1+ctd*0.7)
    else:
        first = 0.0
    if w in self.queries[queryName]:
        second = ((K3+1) *self.queries[queryName][w]) / (K3 + self.queries[queryName][w]
    else:
        second = 0.0

    score += first*second*math.log10((4191-self.docidf[w]+0.5)/(self.docidf[w]+0.5))

bm11 = K2*qalllen*abs(avglen-len)/(avglen+len)
score += bm11

simscore_dic[doc] = score
```

1. 相似度計算：
   本作業主要使用方法為 BM25L 結合 BM11 中 Correction Factor 的概念進行實作，經由實驗參數設定結果如下：
   K1 = 3、b = 0.685、K2 = 0.01751、K3 = 0.2、delta = 0.5，
   由於，各大論文得出 BM11 的 K2 值=0 通常會得到較好的準確度，但經由實驗 K2>0 在我的模型中的準確率會提升，此外，「平均長度-長度」有可能為負值導致所求分數減少，因此我試套上絕對值後準確度的確會提升。
   再者，在實作 $tf_{i'j}$ 時前面有多 X0.765，該參數亦是經由實驗而進行設定，而我認為該參數設定是要將 Doc Term Frequency 的值更平滑。

Best Match Model

資管碩一 M10909112 石家安

心得
由於第一次的作業結束後，老師有請作業 1 表現前幾名的同學上台分享他們的做法，聽完他們的做法後，了解到在製作 tf 的參數設定時，盡量讓 tf 的向量平滑，因此，再做第二份作業的時候，就沒有那麼耗費心力就可超過 baseline。但想要達到前 20 名還是相當困難的，所以這次我還是很好奇前面的同學是怎麼實做的，希望這次亦可聽到前面的高手回答。 當然，在實作的過程中，促使我對整個 model 的運算過程更加了解，雖然排名沒有達到前 20，其實還是蠻難過的....。但從無到有，並且嘗試了各式各樣的參數以及異想天開的想法後，當看到準確度有越來越提升，其實還是相當開心的。