**Bag of Words TF-IDF - Xử lý ngôn ngữ tự nhiên**

**Bag of Words (BoW) - Túi đựng từ** là một thuật toán hỗ trợ xử lý ngôn ngữ tự nhiên và mục đích của **BoW** là phân loại text hay văn bản. Ý tưởng của BoW là phân tích và phân nhóm dựa theo "Bag of Words". Với test data mới, tiến hành tìm ra số lần từng từ của test data xuất hiện trong "bag". Tuy nhiên BoW vẫn tồn tại khuyết điểm, nên TF-IDF là phương pháp khắc phục. Có thể ứng dụng BoW + TF-IDF vào việc tìm kiếm, phân loại tài liệu, lọc mail spam xác định ý định của người dùng…

**BoW hoạt động như thế nào ?**

Trong hầu hết các ngôn ngữ, có một số từ có xu hướng xuất hiện thường xuyên như trong tiếng anh có "is", "the"... tương tự tiếng việt có các từ như "là", "của", "cứ"... Chính vì vậy nếu chỉ xét theo tần số xuất hiện của từng từ thì việc phân loại văn bản rất có thể cho kết quả sai dẫn tỷ lệ chính xác sẽ thấp.

Giải pháp phổ biến là sử dụng một phương pháp thống kê có tên là **Term Frequency– Inverse Document Frequency (TF-IDF)**, giá trị **TF-IDF** của một từ là một con số thu được qua thống kê thể hiện mức độ quan trọng của từ này trong một văn bản, mà bản thân văn bản đang xét nằm trong một tập hợp các văn bản.

Đầu tiên, **TF**(Term Frequency) là tần số xuất hiện của 1 từ trong 1 văn bản có cách tính như sau:

*Trong đó:*

* ***f(t,d)*** *số lần xuất hiện từ t trong văn bản d*
* *Mẫu số là tổng số từ trong văn bản d*

Tiếp theo, là **IDF** (Inverse Document Frequency): Tần số nghịch của 1 từ trong tập văn bản.

Mục đích của việc tính IDF là giảm giá trị của các từ thường xuyên xuất hiện như "is", "the"... Do các từ này không mang nhiều ý nghĩa trong việc phân loại văn bản.

*Trong đó:*

* ***N:*** *là tổng các văn bản trong một tập văn bản* ***N =* |D|**
* *số lượng văn bản khi mà t xuất hiện.*

Cuối cùng, **TF-IDF** những từ có giá trị TF-IDF cao là những từ xuất hiện nhiều trong văn bản này, và xuất hiện ít trong các văn bản khác. Việc này giúp lọc ra những từ phổ biến và giữ lại những từ có giá trị cao (từ khoá của văn bản đó). Được tính bởi công thức:

**Áp dụng:**

Khởi tạo 2 văn bản, tính số lần xuất hiện của mỗi từ trong văn bản:

*# Khởi tạo 2 văn bản text1, text2*

text1 = **"Phúc thích xem phim , Trâm cũng thích xem phim"**text2 = **"Ngoài ra , Phúc còn thích bơi lội"**bowA = text1.split(**" "**) *# tách từ ở văn bản 1*bowB = text2.split(**" "**) *# tách từ ở văn bản 2*

*#Tạo một dictionary*word\_dict = set(bowA).union(set(bowB))  
wordDictA = dict.fromkeys(word\_dict, 0)  
wordDictB = dict.fromkeys(word\_dict, 0)  
  
*#Đếm số lượng từ***for** word **in** bowA:  
 wordDictA[word]+=1  
**for** word **in** bowB:  
 wordDictB[word]+=1  
  
print(**"Các từ trong 2 văn bản là:\n {}"**.format(word\_dict))  
print(**"Số từ xuất hiện trong văn bản 1 là:\n {}\n\nSố từ xuất hiện trong văn bản 2 là:\n {}"**.format(wordDictA,wordDictB))

*Kết quả:*

Các từ trong 2 văn bản là:

{'thích', 'phim', ',', 'lội', 'cũng', 'ra', 'Trâm', 'còn', 'Ngoài', 'Phúc', 'xem', 'bơi'}

Số từ xuất hiện trong văn bản 1 là:

{'thích': 2, 'phim': 2, ',': 1, 'lội': 0, 'cũng': 1, 'ra': 0, 'Trâm': 1, 'còn': 0, 'Ngoài': 0, 'Phúc': 1, 'xem': 2, 'bơi': 0}

Số từ xuất hiện trong văn bản 2 là:

{'thích': 1, 'phim': 0, ',': 1, 'lội': 1, 'cũng': 0, 'ra': 1, 'Trâm': 0, 'còn': 1, 'Ngoài': 1, 'Phúc': 1, 'xem': 0, 'bơi': 1}

**Tính TF:**

**def** compute\_TF(word\_dict, bow):  
 tf\_dict = {}  
 bow\_count = len(bow)  
 **for** word, count **in** word\_dict.items():  
 tf\_dict[word] = count / float(bow\_count)  
 **return** tf\_dict

print(**"\nKết quả TF:\n văn bản 1: {}\n văn bản 2 {}"** .format(compute\_TF(wordDictA,bowA),compute\_TF(wordDictB,bowB)))

*Kết quả:*

Kết quả TF:

văn bản 1: {'xem': 0.2, 'thích': 0.2, 'còn': 0.0, 'Ngoài': 0.0, 'ra': 0.0, 'Trâm': 0.1, ',': 0.1, 'bơi': 0.0, 'lội': 0.0, 'Phúc': 0.1, 'phim': 0.2, 'cũng': 0.1}

văn bản 2: {'xem': 0.0, 'thích': 0.125, 'còn': 0.125, 'Ngoài': 0.125, 'ra': 0.125, 'Trâm': 0.0, ',': 0.125, 'bơi': 0.125, 'lội': 0.125, 'Phúc': 0.125, 'phim': 0.0, 'cũng': 0.0}

**Tính IDF:**

**def** compute\_IDF(doc\_list):  
 **import** math *#import thư viện math* idf\_dict = {} *#tạo một dictionary rỗng* N = len(doc\_list) *#gán độ dài của list cho biến N* idf\_dict = dict.fromkeys(doc\_list[0].keys(), 0) *#tạo dictionary lưu các keys với value = 0  
 #lọc ra thành 1 list gồm các từ xuất hiện >=1 lần* **for** doc **in** doc\_list:  
 **for** word, count **in** doc.items():  
 **if** count > 0:  
 idf\_dict[word] += 1  
 **for** word, count **in** idf\_dict.items():  
 idf\_dict[word] = math.log(N / float(count))  
 **return** idf\_dict

print(**"Kết Quả IDF:\n {}"**.format(compute\_IDF([wordDictA, wordDictB])))

*Kết quả:*

Kết Quả IDF:

{'cũng': 0.6931471805599453, 'Phúc': 0.0, 'bơi': 0.6931471805599453, 'Trâm': 0.6931471805599453, 'ra': 0.6931471805599453, 'phim': 0.6931471805599453, 'lội': 0.6931471805599453, 'còn': 0.6931471805599453, 'xem': 0.6931471805599453, 'Ngoài': 0.6931471805599453, ',': 0.0, 'thích': 0.0}

Từ kết quả trên có thể nhìn thấy những từ có trọng số càng cao thì những từ đó càng có giá trị phân loại và ngược lại ví dụ như từ "the" xuất hiện nhiều nên sẽ không có giá trị phân loại các văn bản với nhau. Tuy nhiên bộ train data lần này ít nên hiệu quả không rõ rệt, nếu bạn thử trên lượng data lớn chắc chắn sẽ rất hiệu quả.