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

Bag of Words(BoW) 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 Word"(corpus). 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 + IF-IDF vào việc tìm kiếm , phân loại tài liệu , lọc mail spam xác định ý của người dùng

# BoW hoạt động như thế nào ?

* Hình thành vector
* Đánh lại trọng số với TF-IDF
  + 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.
  + Phương pháp phổ biến là sử dụng một phương pháp thống kê có tên là 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.

**Áp dụng**

* Đầu tiên là bước tạo vector

**import** **pandas** **as** **pd**

docA = "the quick brown fox jumps over the lazy dog and"

docB = "never jump over the lazy dog quickly"

bowA = docA.split(" ")

bowB = docB.split(" ")

*#Create dictionary*

word\_dict = set(bowA).union(set(bowB))

wordDictA = dict.fromkeys(word\_dict, 0)

wordDictB = dict.fromkeys(word\_dict, 0)

*#count the word in bads*

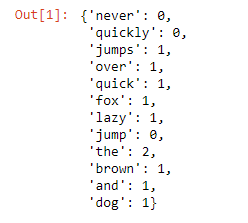
**for** word **in** bowA:

wordDictA[word]+=1

**for** word **in** bowB:

wordDictB[word]+=1

wordDictA



* Bước 2: Tính TF

**def** compute\_TF(word\_dict, bow):

tf\_dict = {}

bow\_count = len(bow)

**for** word, count **in** word\_dict.items():

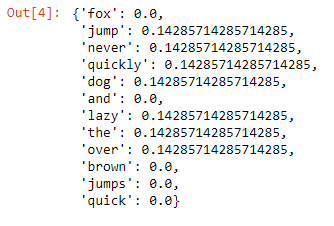
tf\_dict[word] = count/float(bow\_count)

**return** tf\_dict

tf\_bowA = compute\_TF(wordDictA, bowA)

tf\_bowB = compute\_TF(wordDictB, bowB)

tf\_bowB



* Bước 3: Tính IDF

**def** compute\_IDF(doc\_list):

**import** **math**

idf\_dict = {}

N = len(doc\_list)

*#count number of documents that contain this word*

idf\_dict = dict.fromkeys(doc\_list[0].keys(), 0)

**for** doc **in** doc\_list:

**for** word, count **in** doc.items():

**if** count > 0:

idf\_dict[word] += 1

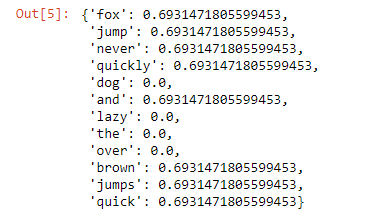
**for** word, count **in** idf\_dict.items():

idf\_dict[word] = math.log(N/float(count))

**return** idf\_dict

idfs = compute\_IDF([wordDictA, wordDictB])

idfs



* Cuối cùng: Tính TF-IDF

**def** compute\_TFIDF(tf\_bow, idfs):

tfidf = {}

**for** word, val **in** tf\_bow.items():

tfidf[word] = val\*idfs[word]

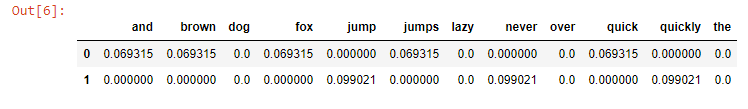
**return** tfidf

tfidf\_bowA = compute\_TFIDF(tf\_bowA, idfs)

tfidf\_bowB = compute\_TFIDF(tf\_bowB, idfs)

df = pd.DataFrame([tfidf\_bowA, tfidf\_bowB])

df



* 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.