Term Frequency – Inverse Document Frequence (TF-IDF)

I. Giới thiệu

TF-IDF là một phương pháp thống kê được sử dụng rộng rãi ở các bước tiền xử lý trong các bài toán xử lý ngôn ngữ tự nhiên và là một trong những cách vector hóa dữ liệu từ dạng văn bản về dạng số sao cho máy có thể hiểu được. Phương pháp này được dùng để đánh giá mức độ quan trọng của một từ trong văn bản mà văn bản đó lại nằm trong một tập dữ liệu chứa các văn bản . Một giá trị TF-IDF của một từ được chia ra thành 2 thành phần chính để tính toán đó là TF (term frequency) và IDF (inverse document frequency) và giá trị $TFIDF = TF \times IDF$.

1 . TF – term frequency

TF là giá trị thống kê của một từ thể hiện tần xuất xuất hiện của từ đó trong đoạn văn bản chứa nó. Bởi vì tập dữ liệu có thể chứa các đoạn văn bản có độ dài khác nhau và để đánh giá được một cách tổng quát hơn trên toàn tập thì ta cần chuẩn hóa tiếp giá trị TF bằng cách chia cho độ dài văn bản . Cuối cùng ta có công thức sau :

$$tf(w \in d) = \frac{f(w,d)}{length(d)}$$

Với w là một từ bất kì trong văn bản d ,f(w,d) là số lần xuất hiện của w trong d và length(d) là độ dài của văn bản đó. Ngoài ra còn có những cách tính khác cho giá trị này được đề cập ở nhiều tài liệu khác ví dụ như chuẩn hóa bằng cách chia cho giá trị lớn nhất của số lần xuất hiện các từ trong văn bản :

$$tf(w \in d) = \frac{f(w,d)}{\max\{f(w,d) : w \in d\}}$$

2 . IDF – inverse document frequency

Tuy đại lượng TF cho phép ta đánh giá xem được sự có mặt của một từ nào đó trong một đoạn văn bản là nhiều hay ít nhưng nhiều không có nghĩa là từ đấy quan trọng trong đoạn văn bản đó và ngược lại. Ta có thể lấy ví dụ như các từ thuộc danh sách các từ dừng (stopwords) trong tiếng Anh : “the”, “on”, “in”, … thì việc đánh giá nội dung của một câu nào đó mà chỉ dựa trên số lần xuất hiện của các từ này sẽ không thu được gì cả . Mặt khác ,giống trong quá trình luyện đọc hiểu tiếng Anh thì câu trả lời lại nằm ở những từ khóa ít xuất hiện hơn và quan trọng hơn . Để tránh việc đưa ra những từ mang tính bổ sung về mặt ngữ pháp trong câu mà không có giá trị nhiều trong việc đánh giá ngữ nghĩa thì ta cần đến giá trị IDF (inverse document frequency) . Giá trị này phản ứng mức độ quan trọng thực tế của một từ trên toàn tập dữ liệu. Công thức tính toán như sau :

$$idf(w,D) = \log \frac{|D|}{|\{d \in D : w \in d\}|}$$

Trong đó |D| là tổng số văn bản trong tập dữ liệu D , $|\{d \in D : w \in d\}|$ là số văn bản d chứa từ w trong tập D .

Qua công thức trên ta có thể thấy một điều rằng đối với những từ ngữ chức năng xuất hiện nhiều mà không có tính ngữ nghĩa cao thì giá trị mẫu số trong biểu thức chứa trong hàm log sẽ có giá trị cao và tiến dần về giá trị độ dài của tập dữ liệu D :

$$\lim_{d->D} idf(w,D) = \log \frac{|D|}{|\{d \in D : w \in d\}|} = 0$$

Từ công thức tính TF thì $\max(tf(w \in d))$ =1 và $\min(tf(w \in d)) = 0$ và kết hợp với biểu thức lấy giới hạn ở trên ta suy ra được giá trị TFIDF của những từ này sẽ rất nhỏ thậm chí là bằng 0 cho dù tần xuất của từ đó trong các đoạn văn bản có thay đổi như thế nào. Ngược lại đối với những từ hiếm gặp thì ta lại thu được giá trị TFIDF của nó cao . Đặc biết nếu một từ không xuất hiện trong tập văn bản nào thì $|\{d \in D : w \in d\}| = 0$, lúc này nó sẽ được cộng thêm 1 để cho mẫu số khác 0.

Vậy công thức để tính toán đại lượng TFIDF của một từ w bất kì trong đoạn văn bản d nằm trong tập dữ liệu D là :

$$TFIDF(w,d,D) = TF(w,d) \times IDF(w,D)$$

Một cách tương tự ta có thể xây dựng công thức cho mô hình bigrams hay ngrams .

II. Thực hành

Trong mục này ta sẽ thực hành tính toán TFIDF bằng 2 cách . Cách thứ nhất không sử dụng đến thư viện có sẵn , cách thứ hai sẽ sử dụng đến sự hỗ trợ của thư viện scikit-learn.

1. Không sử dụng thư viện scikit-learn

Để đơn giản thì mục này ta sẽ chỉ thực hành với mô hình unigram.

- Giả sử tập dữ liệu trông như thế này :

```
dataset=[
        "This is the first sentence. Hope you enjoy it !",
       "I love reading this book  .",
       "That is the second sentence.",
       "I hate my dog and my sister love her. Tommorow I will kill my dog and buy a new cat."
]
```

- Trước hết ta cần phải loại bỏ các loại dấu câu (punctuations)

```
#remove punctuations
punc list=string.punctuation
for i in range(len(dataset)):
  dataset[i]=dataset[i].lower()
  tokens=word tokenize(dataset[i])
  for token in tokens:
    token=token.lower()
    if token in punc_list:
      dataset[i]=dataset[i].replace(token,"")
```

- Tiếp theo ta định nghĩa một số hàm để tính TF : hàm TF_at_rows phục vụ việc TF tại mỗi hàng còn hàm TF dùng để tính toán TF trên toàn tập dữ liệu

```
#calculate TF
def TF at rows(dataset,index):
  tf dict at t={}
  sent_length=len(dataset[index].split())
  for word in dataset[index].split():
    if word in tf dict at t:
      tf dict at t[word]+=1
    else:
      tf dict at t[word]=1

  for word in tf dict at t:
    tf_dict_at_t[word]/=sent_length

  return tf dict at t

def TF(dataset):
  tf map=[]
  for i in range(len(dataset)):
    tf map.append(TF at rows(dataset,i))
  return tf_map

tf_map=TF(dataset)
```

- Theo cách đặt vấn đề phần lý thuyết thì cũng xây dựng được các hàm tính toán IDF, trong đó hàm count_on_dataset chính là việc tính toán các giá trị ở dưới mẫu số của phân số chứa trong hàm log

```
#calculate frequency for IDF
def count_on_dataset(tf_map,dataset):
  count on ds={}
  for sample in tf map:
    for word in sample:
      if word in count on ds:
        count on ds[word]+=1
      else:
        count_on_ds[word]=1
  return count on ds
#calculate IDF
def IDF(dataset,freq on dataset):
  idf_map={}
  for word in freq on dataset:
    if(freq on dataset[word]==0):
      idf map[word]=math.log(len(dataset)/(1))
    else:
      idf_map[word]=math.log(len(dataset)/(freq_on_dataset[word]))
```

```
  return idf_map
```

- Cuối cùng chính là hàm để tính toán TFIDF

```
#calculate TFIDF
def TFIDF(tf map,idf map):
  tfidf map=[]
  for sample in tf map:
    temp_tfidf={}
    for word in sample:
      temp tfidf[word]=sample[word]*idf map[word]
    tfidf_map.append(temp_tfidf)
  return tfidf_map

print("TFIDF at first sample : \n",TFIDF(tf_map,idf_map)[0])
```

Như vậy ta đã thực hiện xong việc mô tả phương pháp TFIDF mà không cần sử dụng đến thư viện . Nhưng trên thực tế , để tiết kiệm thời gian thì ta sẽ học cách sử dụng những thư viện có sẵn .

2. Sử dụng thư viện scikit-learn

Trong mục này có 2 cách làm :

Cách 1 : Sử dụng TfidfVectorizer , tham khảo api tại đây :

https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

```
from sklearn.feature extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(smooth idf=True,ngram range=(1,1),stop words='english')
dataset=[
         "This is the first sentence. Hope you enjoy it !",
        "I love reading this book  .",
        "That is the second sentence.",
        "I hate my dog and my sister love her. Tommorow I will kill my dog and buy a new cat."
]

tfidf = vectorizer.fit transform(corpus).toarray()
feature names=vectorizer.get feature names()
print("Vocabs : ",feature names)
print("Vocab size : ",len(feature names))
print("TFIDF matrix has shape : ",tfidf.shape)
print(tfidf)
```

Theo các thông tin chú thích từ đường dẫn trên ,ta thấy rằng khi sử dụng lớp TfidfVectorizer thì nó sẽ cung cấp them cho những lựa chọn hữu ích như có muốn chuyển tất cả về dạng chữ không viết hoa hay không (lowercase) hay có muốn loại ra những stopwords trong tiếng Anh hay không, thậm chí ta có thể điều chỉnh linh hoạt để đầu ra là unigram hay bigrams đều được ,…

Đầu ra của dòng code vectorizer.get_feature_names() sẽ trả về bộ từ vựng của tập dữ liệu và tfidf sẽ là một numpy array có cỡ là (length_dataset x vocab_size). Như ví dụ trên thì tfidf có cớ là 4 x 15 (chú ý rằng cỡ của bộ từ vựng sẽ tang lên nếu stopwords không được lọc ra).

```
Vocabs :  ['book', 'buy', 'cat', 'dog', 'enjoy', 'hate', 'hope', 'kill', 'love', 'new', 'reading', 'second', 'sentence', 'sister', 'tommorow']
Vocab size :  15
TFIDF matrix has shape :  (4, 15)
[[0.         0.         0.         0.         0.61761437 0.
  0.61761437 0.         0.         0.         0.         0.
  0.48693426 0.         0.        ]
 [0.61761437 0.         0.         0.         0.         0.
  0.         0.         0.48693426 0.         0.61761437 0.
  0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.78528828
  0.6191303  0.         0.        ]
 [0.         0.29333722 0.29333722 0.58667444 0.         0.29333722
  0.         0.29333722 0.23127044 0.29333722 0.         0.
  0.         0.29333722 0.29333722]]
```

Xét trường hợp của đoạn văn bản thứ nhất : “ This is the first sentence . Hope you enjoy it !”, so sánh với tfidf[0] ta có nhận xét sau :

```
[0.         0.         0.         0.         0.61761437 0.
 0.61761437 0.         0.         0.         0.         0.
 0.48693426 0.         0.        ]
```

Khi đối chiếu những chỉ số mà tại đó tfidf[0] khác 0 với giá trị có cùng chỉ số tại bộ từ vựng thì có thể thấy rằng những giá trị khác 0 đó chính là giá trị tfidf của những từ xuất hiện trong đoạn văn bản thứ nhất, còn những giá trị bằng 0 thể hiện những từ theo thứ tự có trong bộ từ vựng nhưng lại không có mặt trong tfidf[0]. Tương tự cho các đoạn văn bản khác.

Cách 2: sử dụng CountVectorizer kết hợp với TfidfTranformer

Api để tham khảo cho CountVectorizer tại đây :

https://scikitlearn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html

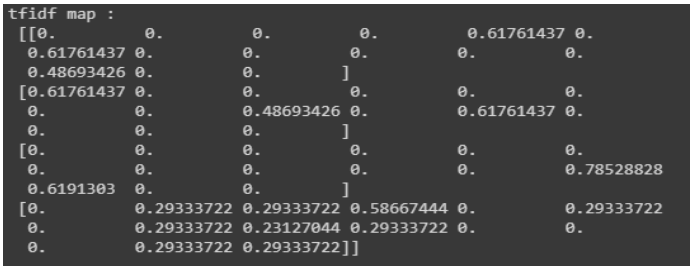
```
from sklearn.feature extraction.text import CountVectorizer, TfidfTransformer
countvectorizer=CountVectorizer(lowercase=True,stop words='english')
word count vector=countvectorizer.fit transform(dataset)
print("Vocab : ",countvectorizer.get_feature_names())
print(word_count_vector.toarray())
```

Đoạn mã trên cho ra kết quả là bộ từ vựng thu được cùng với một numpy array thể hiện tần xuất xuất hiện của mỗi từ trong bộ từ vựng tại mỗi đoạn văn bản.

```
Vocab :  ['book', 'buy', 'cat', 'dog', 'enjoy', 'hate', 'hope', 'kill', 'love', 'new', 'reading', 'second', 'sentence', 'sister', 'tommorow']
[[0 0 0 0 1 0 1 0 0 0 0 0 1 0 0]
 [1 0 0 0 0 0 0 0 1 0 1 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 1 1 0 0]
 [0 1 1 2 0 1 0 1 1 1 0 0 0 1 1]]
```

Để thu được các giá trị tfidf thì cần thực hiện thêm một bước nữa :

```
tf_idf_transformer=TfidfTransformer()
tfidf=tf idf transformer.fit transform(word count vector).toarray()
print("tfidf map : ",tfidf)
```



Có thể thấy rằng kết quả cho ra giống cách 1 . Vậy câu hỏi đặt ra khi nào dùng cách 1 , khi nào dùng cách 2 . Về cơ bản 2 cách là như nhau nhưng khi bài toán của ta cần sử dụng thêm kết quả liên quan đến tần xuất xuất hiện của các thành phần trong bộ từ vựng tại những điểm dữ liệu thì ta sử dụng cách 2 với phương thức CountVectorizer. Còn chỉ cần đầu ra là các giá trị TF-IDF thì cách 1 là ngắn gọn nhất.

Ngoài ra kết quả của việc thực hiện thuần theo công thức có thể khác khi thực hiện với thư viện bởi vì tác giả thư viện có thể định nghĩa một số hàm tính toán khác nhưng về mặt ý tưởng thì là một.

Tài liệu tham khảo :

https://codetudau.com/bag-of-words-tf-idf-xu-ly-ngon-ngu-tu-nhien/index.html

https://nguyenvanhieu.vn/tf-idf-la-gi/

https://viblo.asia/p/tf-idf-term-frequency-inverse-document-frequency-JQVkVZgKkyd

https://viblo.asia/p/trich-chon-thuoc-tinh-trong-doan-van-ban-voi-tf-idf-Az45bAOqlxY

https://monkeylearn.com/blog/what-is-tf-idf/#:~:text=TF%2DIDF%20is%20a%20statistical,across%20a%20set%20of%20documents.

https://towardsdatascience.com/introduction-to-nlp-part-3-tf-idf-explained-cedb1fc1f7dc

https://medium.com/analytics-vidhya/tf-idf-term-frequency-technique-easiest-explanation-for-text-classification-in-nlp-with-code-8ca3912e58c3

https://towardsdatascience.com/tf-idf-for-document-ranking-from-scratch-in-python-on-real-world-dataset-796d339a4089

https://www.tutorialspoint.com/gensim/gensim_creating_tf_idf_matrix.htm

https://medium.com/@cmukesh8688/tf-idf-vectorizer-scikit-learn-dbc0244a911a

https://www.huongnghiepaau.com/tf-idf

https://streamsql.io/blog/tf-idf-from-scratch