**Bài 12: NLTK Package phần 2 - Lập trình AI bằng Python**

**6. Bag of Word (BoW) Model :**

Bag of Word (BoW) một mô hình trong xử lý ngôn ngữ tự nhiên, về cơ bản được sử dụng để trích xuất các tính năng từ văn bản để văn bản có thể được sử dụng trong mô hình hóa như trong các thuật toán học máy.

Bây giờ câu hỏi đặt ra là tại sao chúng ta cần trích xuất các tính năng từ văn bản. Đó là bởi vì các thuật toán học máy không thể hoạt động với dữ liệu thô và chúng cần dữ liệu số để có thể trích xuất thông tin có ý nghĩa từ đó. Việc chuyển đổi dữ liệu văn bản thành dữ liệu số được gọi là trích xuất đối tượng hoặc mã hóa đối tượng.

**a. Hoạt động :**

Đây là cách tiếp cận rất đơn giản để trích xuất các tính năng từ văn bản. Giả sử chúng ta có một tài liệu văn bản và chúng ta muốn chuyển nó thành dữ liệu số hoặc nói muốn trích xuất các tính năng của nó thì trước hết mô hình này trích xuất một từ vựng từ tất cả các từ trong tài liệu. Sau đó, bằng cách sử dụng ma trận thuật ngữ tài liệu, nó sẽ xây dựng một mô hình. Theo cách này, BoW chỉ đại diện cho tài liệu như bag of words. Mọi thông tin về thứ tự hoặc cấu trúc của các từ trong tài liệu sẽ bị loại bỏ.

**b. Khái niệm về ma trận thuật ngữ document :**

Thuật toán BoW xây dựng mô hình bằng cách sử dụng ma trận thuật ngữ tài liệu. Như tên cho thấy, ma trận thuật ngữ tài liệu là ma trận của các số lượng từ khác nhau xuất hiện trong tài liệu. Với sự trợ giúp của ma trận này, tài liệu văn bản có thể được biểu diễn dưới dạng sự kết hợp có trọng số của các từ khác nhau. Bằng cách đặt ngưỡng và chọn các từ có nghĩa hơn, ta có thể xây dựng biểu đồ của tất cả các từ trong tài liệu có thể được sử dụng làm vectơ đặc trưng. Sau đây là một ví dụ để hiểu khái niệm về ma trận thuật ngữ tài liệu:

Giả sử ta có. 2 câu :

* **Câu 1** : We are using the Bag of Words model.
* **Câu 2** : Bag of Words model is used for extracting the features.

Bây giờ, bằng cách xem xét hai câu này, chúng ta có 13 từ khác biệt sau đây :

* we
* are
* using
* the
* bag
* of
* words
* model
* is
* used
* for
* extracting
* features

chúng ta cần xây dựng biểu đồ cho mỗi câu bằng cách sử dụng số từ trong mỗi câu -

* **Câu 1** − [1,1,1,1,1,1,1,1,0,0,0,0,0]
* **Câu 2** − [0,0,0,1,1,1,1,1,1,1,1,1,1]

141/5000Bằng cách này, ta có các vectơ đặc trưng đã được trích xuất. Mỗi vectơ đặc trưng là 13 chiều vì chúng ta có 13 từ riêng biệt.

**7. Khái niệm về thống kê :**

Khái niệm về số liệu thống kê được gọi là TermFrequency-Inverse Document Frequency (tf-idf). Mỗi từ đều quan trọng trong tài liệu. Các số liệu thống kê giúp chúng tôi hiểu tầm quan trọng của mỗi từ.

**Term Frequency(tf) :**

Nó là thước đo tần suất mỗi từ xuất hiện trong tài liệu. Nó có thể nhận được bằng cách chia số lượng của mỗi từ cho tổng số từ trong một tài liệu nhất định.

**Inverse Document Frequency(idf) :**

Nó là thước đo mức độ duy nhất của một từ đối với tài liệu này trong bộ tài liệu nhất định. Để tính toán idf và xây dựng một vectơ đặc trưng riêng biệt, ta cần giảm trọng số của các từ thường xuất hiện như và và tăng trọng số của các từ hiếm.

**8. Xây dựng Bag of Words Model bằng NLTK :**

Trong bài này chúngta sẽ định nghĩa collection của chuỗi bằng các sử dụng CountVectorizer để tạo vectors từ câu

Import package

from sklearn.feature\_extraction.text **import** CountVectorizer

Định nghĩa set của câu :

Sentences = ['We are using the Bag of Word model', 'Bag of Word model is

used for extracting the features.']

vectorizer\_count = CountVectorizer()

features\_text = vectorizer.fit\_transform(Sentences).todense()

print(vectorizer.vocabulary\_)

Đoạn code trên tạo ra kết quả như hình dưới đây. Cho thấy rằng chúng ta có 13 từ khác biệt trong hai câu trên

{'we': 11, 'are': 0, 'using': 10, 'the': 8, 'bag': 1, 'of': 7,

'word': 12, 'model': 6, 'is': 5, 'used': 9, 'for': 4, 'extracting': 2, 'features': 3}

Đây là các vectơ đặc trưng (dạng văn bản thành số) có thể được sử dụng cho học máy.

**9. Xử lý vấn đề :**

**a. Category Prediction**

Trong một bộ văn bản, không chỉ từ ngữ mà thể loại của từ ngữ cũng rất quan trọng; trong loại văn bản mà một từ cụ thể rơi vào. Ví dụ: tôi muốn dự đoán liệu một câu đã cho có thuộc danh mục email, tin tức, thể thao, máy tính, v.v. Trong ví dụ sau, tôi sẽ sử dụng tf-idf để tạo một vectơ đặc trưng để tìm danh mục tài liệu. Tôi sẽ sử dụng dữ liệu từ 20 tập dữ liệu nhóm tin của sklearn.

from sklearn.datasets **import** fetch\_20newsgroups

from sklearn.naive\_bayes **import** MultinomialNB

from sklearn.feature\_extraction.text **import** TfidfTransformer

from sklearn.feature\_extraction.text **import** CountVectorizer

Xác địnhcategory map. Tôi đang sử dụng năm categories khác nhau có tên là Religion, Autos, Sports, Electronics và Space.

category\_map = {'talk.religion.misc':'Religion','rec.autos''Autos',

'rec.sport.hockey':'Hockey','sci.electronics':'Electronics', 'sci.space': 'Space'}

Tạo bộ training set:

training\_data = fetch\_20newsgroups(subset = 'train',

categories = category\_map.keys(), shuffle = True, random\_state = 5)

Xây dựng một vectơ đếm và trích xuất số lượng từ :

vectorizer\_count = CountVectorizer()

train\_tc = vectorizer\_count.fit\_transform(training\_data.data)

print("\nDimensions of training data:", train\_tc.shape)

tf-idf transformer được tạo như sau :

tfidf = TfidfTransformer()

train\_tfidf = tfidf.fit\_transform(train\_tc)

Tạo bộ dữ liệu test :

input\_data = [

'Discovery was a space shuttle',

'Hindu, Christian, Sikh all are religions',

'We must have to drive safely',

'Puck is a disk made of rubber',

'Television, Microwave, Refrigrated all uses electricity'

]

Dữ liệu trên sẽ giúp mình đào tạo bộ phân loại Naive Bayes đa thức -

classifier = MultinomialNB().fit(train\_tfidf, training\_data.target)

Biến đổi dữ liệu đầu vào bằng cách sử dụng vector đếm :

input\_tc = vectorizer\_count.transform(input\_data)

Bây giờ, chúng ta sẽ chuyển đổi dữ liệu được vector hóa bằng cách sử dụng tfidf transformer :

input\_tfidf = tfidf.transform(input\_tc)

Dự đoán kết quả phân loại

predictions = classifier.predict(input\_tfidf)

Output tạo ra như sau :

**for** **sent**, **category** **in** **zip**(input\_data, predictions):

**print**('\nInput Data:', sent, '\n Category:', \

category\_map[training\_data.target\_names[category]])

Output :

Dimensions of training data: (2755, 39297)

Input Data: Discovery was a space shuttle

Category: Space

Input Data: Hindu, Christian, Sikh all are religions

Category: Religion

Input Data: We must have to drive safely

Category: Autos

Input Data: Puck is a disk made of rubber

Category: Hockey

Input Data: Television, Microwave, Refrigrated all uses electricity

Category: Electronics

**b. Gender Finder :**

Trong câu lệnh bài toán này, một bộ phân loại sẽ được đào tạo để tìm giới tính (nam hoặc nữ) bằng cách cung cấp tên. Ta cần sử dụng heuristic để xây dựng một vector đặc trưng và đào tạo bộ phân loại. Ta sẽ sử dụng dữ liệu được gắn nhãn từ gói scikit-learning. Đoạn code sau để xây dựng công cụ tìm giới tính:

**import** random

from nltk **import** NaiveBayesClassifier

from nltk.classify **import** accuracy as nltk\_accuracy

from nltk.corpus **import** names

Bây giờ chúng ta cần trích xuất N ký tự cuối cùng từ đầu vào

**def** **extract\_features**(word, N = 2):

last\_n\_letters = word[-N:]

**return** {'feature': last\_n\_letters.lower()}

**if** \_\_name\_\_=='\_\_main\_\_':

Tạo dữ liệu training sử dụng tên label (male hoặc female) bằng NLTK

male\_list = [(name, 'male') **for** name **in** names.words('male.txt')]

female\_list = [(name, 'female') **for** name **in** names.words('female.txt')]

data = (male\_list + female\_list)

random.seed(5)

random.shuffle(data)

Dữ liệu test được tạo như sau :

namesInput = ['Rajesh', 'Gaurav', 'Swati', 'Shubha']

Xác định số lượng mẫu được sử dụng để train và test như sau

train\_sample = int(0.8 \* len(data))

chúng ta cần lặp qua các độ dài khác nhau để có thể so sánh độ chính xác -

**for** i in **range**(1, 6):

print('\nNumber of end letters:', i)

features = [(extract\_features(n, i), gender) **for** (n, gender) in data]

train\_data, test\_data = features[:train\_sample],

features[train\_sample:]

classifier = NaiveBayesClassifier.train(train\_data)

Độ chính xác của bộ phân loại có thể được tính như sau:

accuracy\_classifier = round(100 \* nltk\_accuracy(classifier, test\_data), 2)

print('Accuracy = ' + str(accuracy\_classifier) + '%')

Ta có thể dự đoán output :

**for** name **in** namesInput:

print(name, '==>', classifier.classify(extract\_features(name, i)))

Chương trình trên sẽ tạo ra output như sau :

Number of end letters: 1

Accuracy = 74.7%

Rajesh -> female

Gaurav -> male

Swati -> female

Shubha -> female

Number of end letters: 2

Accuracy = 78.79%

Rajesh -> male

Gaurav -> male

Swati -> female

Shubha -> female

Number of end letters: 3

Accuracy = 77.22%

Rajesh -> male

Gaurav -> female

Swati -> female

Shubha -> female

Number of end letters: 4

Accuracy = 69.98%

Rajesh -> female

Gaurav -> female

Swati -> female

Shubha -> female

Number of end letters: 5

Accuracy = 64.63%

Rajesh -> female

Gaurav -> female

Swati -> female

Shubha -> female

Trong đầu ra ở trên,ta có thể thấy rằng độ chính xác về số lượng chữ cái cuối tối đa là hai và nó đang giảm khi số lượng chữ cái cuối ngày càng tăng.