Step1: Text Preprocessing

IF.EDF

- Emoji and Emoticons Handling: convert to text

- Stemming and Lemmatization:

- Tokenization:

+ Split paragraph, document into sentences

+ Use RegEx or library: nltk, genism,… => nltk.word\_tokenize()

**- Exploratory Data Analysis (EDA):**

**- Bag of work (BoW):**

**Bag of Words (BoW)** là một kỹ thuật phổ biến dùng để biểu diễn văn bản trong các bài toán xử lý ngôn ngữ tự nhiên (NLP). Ý tưởng chính của BoW là đơn giản hóa văn bản thành một tập hợp các từ mà không quan tâm đến thứ tự từ trong câu hay ngữ pháp. Mục tiêu là chuyển một đoạn văn bản hoặc một tài liệu thành một vectơ số mà mô hình máy học có thể hiểu và sử dụng.

**Nguyên lý hoạt động của BoW:**

1. **Tạo từ điển (vocabulary)**: BoW tạo một từ điển gồm tất cả các từ xuất hiện trong toàn bộ tập dữ liệu. Từ điển này là một danh sách các từ duy nhất (unique words) có trong các tài liệu hoặc văn bản.
2. **Biểu diễn văn bản**: Mỗi tài liệu hoặc câu được biểu diễn dưới dạng một vectơ với chiều dài bằng kích thước của từ điển. Mỗi phần tử của vectơ tương ứng với một từ trong từ điển.
3. **Đếm số lần xuất hiện (frequency count)**: Vectơ của tài liệu chứa các số đếm tương ứng với số lần xuất hiện của mỗi từ trong tài liệu. Nếu một từ xuất hiện nhiều lần trong tài liệu, giá trị tương ứng của từ đó trong vectơ sẽ là số lần xuất hiện của từ.

**Ví dụ minh họa:**

Giả sử bạn có hai câu sau:

* "Tôi yêu học máy"
* "Học máy rất thú vị"

1. **Tạo từ điển**: Từ điển của hai câu trên sẽ bao gồm các từ duy nhất: ['tôi', 'yêu', 'học', 'máy', 'rất', 'thú vị'].
2. **Biểu diễn câu**: Chuyển các câu thành vectơ BoW:
   * "Tôi yêu học máy" → [1, 1, 1, 1, 0, 0] (có 1 lần "tôi", 1 lần "yêu", 1 lần "học", 1 lần "máy", và không có từ "rất", "thú vị").
   * "Học máy rất thú vị" → [0, 0, 1, 1, 1, 1] (có 1 lần "học", 1 lần "máy", 1 lần "rất", 1 lần "thú vị", và không có từ "tôi", "yêu").

**Đặc điểm của BoW:**

* **Đơn giản và dễ hiểu**: BoW chỉ quan tâm đến tần suất xuất hiện của từ và không quan tâm đến ngữ pháp, ngữ nghĩa hay vị trí của từ trong câu.
* **Không giữ được ngữ cảnh**: Một hạn chế lớn của BoW là nó không thể nhận biết được ngữ cảnh hay mối liên hệ giữa các từ. Ví dụ, hai câu có cùng các từ nhưng thứ tự khác nhau sẽ được biểu diễn giống hệt nhau.
* **Kích thước từ điển lớn**: Khi áp dụng BoW cho các tập dữ liệu lớn, từ điển có thể rất lớn, dẫn đến việc vectơ của văn bản trở nên rất dài và thưa (sparse matrix), gây khó khăn trong xử lý và tốn tài nguyên.

**Biến thể:**

* **TF-IDF (Term Frequency-Inverse Document Frequency)**: Đây là một biến thể của BoW, giúp tính trọng số của các từ không chỉ dựa trên tần suất xuất hiện trong một tài liệu mà còn dựa trên tần suất xuất hiện trong toàn bộ tập dữ liệu. Điều này giúp giảm trọng số của các từ phổ biến như "là", "và", "có", v.v. và tăng trọng số của các từ quan trọng trong ngữ cảnh cụ thể.

**Ứng dụng của BoW:**

* **Phân loại văn bản**: BoW được sử dụng rộng rãi trong các bài toán phân loại văn bản như phân loại email spam, phân loại tin tức, hay dự đoán cảm xúc của người dùng.
* **Trích xuất đặc trưng**: Nó được dùng để trích xuất các đặc trưng từ văn bản để sử dụng trong các thuật toán máy học như Naive Bayes, SVM, hay các mô hình hồi quy logistic.

Tóm lại, **Bag of Words (BoW)** là một kỹ thuật đơn giản nhưng hiệu quả để biểu diễn văn bản, nhưng nó có những hạn chế về ngữ cảnh và khả năng mở rộng khi làm việc với các bộ dữ liệu lớn

**- TF-IDF:**

**TF-IDF (Term Frequency - Inverse Document Frequency)** là một kỹ thuật phổ biến trong xử lý ngôn ngữ tự nhiên (NLP) dùng để biểu diễn văn bản dưới dạng các đặc trưng số. Kỹ thuật này được dùng để đánh giá tầm quan trọng của một từ trong một tài liệu tương ứng với toàn bộ tập tài liệu (corpus). Nó cải thiện kỹ thuật **Bag of Words (BoW)** bằng cách không chỉ đếm số lần xuất hiện của từ (tần suất), mà còn cân nhắc đến mức độ phổ biến của từ trong toàn bộ tập dữ liệu.

### Thành phần của TF-IDF

TF-IDF gồm hai thành phần chính:

1. **TF (Term Frequency - Tần suất từ)**: Đo tần suất xuất hiện của từ trong một tài liệu.
2. **IDF (Inverse Document Frequency - Tần suất nghịch đảo của tài liệu)**: Đo mức độ phổ biến của từ trên toàn bộ tập tài liệu.

#### 1. **TF (Tần suất từ)**

Tần suất của từ (TF) trong một tài liệu được tính bằng tỷ lệ giữa số lần xuất hiện của từ đó với tổng số từ trong tài liệu. Nó thể hiện mức độ quan trọng của từ trong tài liệu cụ thể.

TF(t,d)=soˆˊ laˆˋn từ t xuaˆˊt hiện trong taˋi liệu dtổng soˆˊ từ trong taˋi liệu dTF(t, d) = \frac{\text{số lần từ } t \text{ xuất hiện trong tài liệu } d}{\text{tổng số từ trong tài liệu } d}TF(t,d)=tổng soˆˊ từ trong taˋi liệu dsoˆˊ laˆˋn từ t xuaˆˊt hiện trong taˋi liệu d​

Ví dụ: Nếu từ "học" xuất hiện 3 lần trong tài liệu có tổng cộng 100 từ, thì TF của từ "học" là 3100=0.03\frac{3}{100} = 0.031003​=0.03.

#### 2. **IDF (Tần suất nghịch đảo của tài liệu)**

IDF đo mức độ ít phổ biến của một từ trên toàn bộ tập tài liệu. Từ nào xuất hiện trong nhiều tài liệu sẽ có IDF thấp hơn, vì nó được coi là ít quan trọng. IDF được tính bằng cách lấy logarit của tỷ lệ giữa tổng số tài liệu và số tài liệu chứa từ đó, và cộng thêm 1 vào mẫu số để tránh phép chia cho 0.

IDF(t,D)=log⁡(tổng soˆˊ taˋi liệusoˆˊ taˋi liệu chứa từ t+1)IDF(t, D) = \log \left( \frac{\text{tổng số tài liệu}}{\text{số tài liệu chứa từ } t + 1} \right)IDF(t,D)=log(soˆˊ taˋi liệu chứa từ t+1tổng soˆˊ taˋi liệu​)

Ví dụ: Nếu từ "học" xuất hiện trong 10 tài liệu trên tổng số 100 tài liệu, thì IDF của từ "học" sẽ là:

IDF(học)=log⁡(10010+1)=log⁡(10011)≈1.00IDF(học) = \log \left( \frac{100}{10 + 1} \right) = \log \left( \frac{100}{11} \right) \approx 1.00IDF(học)=log(10+1100​)=log(11100​)≈1.00

#### 3. **TF-IDF**

TF-IDF của một từ trong một tài liệu là tích của TF và IDF của từ đó:

TF-IDF(t,d,D)=TF(t,d)×IDF(t,D)TF\text{-}IDF(t, d, D) = TF(t, d) \times IDF(t, D)TF-IDF(t,d,D)=TF(t,d)×IDF(t,D)

Từ nào xuất hiện nhiều trong một tài liệu cụ thể (TF cao) nhưng ít xuất hiện trong các tài liệu khác (IDF cao) sẽ có giá trị TF-IDF cao, thể hiện sự quan trọng của từ đó trong ngữ cảnh tài liệu.

### Ví dụ minh họa:

Giả sử chúng ta có 3 tài liệu:

1. "Tôi yêu học máy"
2. "Học máy rất thú vị"
3. "Tôi yêu thú vị"

Từ điển (vocabulary) sẽ là: ['tôi', 'yêu', 'học', 'máy', 'rất', 'thú', 'vị'].

* **TF của từ "học" trong tài liệu 1**: Từ "học" xuất hiện 1 lần trong tài liệu có 4 từ, nên TF = 14\frac{1}{4}41​.
* **IDF của từ "học"**: Từ "học" xuất hiện trong 2 tài liệu (tài liệu 1 và tài liệu 2). Tổng số tài liệu là 3, do đó IDF = log⁡(32+1)=log⁡(1)=0\log \left( \frac{3}{2 + 1} \right) = \log (1) = 0log(2+13​)=log(1)=0.

Kết quả: Nếu từ nào xuất hiện trong tất cả các tài liệu (như từ rất phổ biến "học"), giá trị TF-IDF của nó sẽ nhỏ, vì nó không giúp phân biệt các tài liệu với nhau. Trong khi đó, những từ xuất hiện nhiều trong một tài liệu nhưng ít xuất hiện trong các tài liệu khác sẽ có TF-IDF cao, thể hiện tầm quan trọng của từ đó trong tài liệu cụ thể.

### Ưu điểm của TF-IDF

* **Giảm trọng số của các từ phổ biến**: TF-IDF làm giảm trọng số của các từ xuất hiện thường xuyên (như "là", "và", "có"), giúp mô hình tập trung vào các từ mang tính phân biệt cao.
* **Đơn giản và hiệu quả**: TF-IDF là một kỹ thuật đơn giản nhưng rất hiệu quả cho việc biểu diễn văn bản khi áp dụng cho các mô hình máy học.

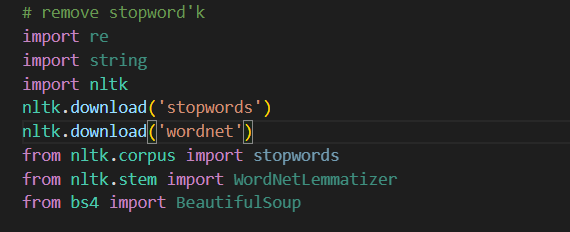
### Nhược điểm của TF-IDF

* **Không nắm bắt được ngữ cảnh**: TF-IDF không quan tâm đến vị trí và ngữ nghĩa của từ trong câu. Nó chỉ tập trung vào tần suất xuất hiện.
* **Không xử lý được mối quan hệ giữa các từ**: TF-IDF không xem xét mối quan hệ giữa các từ, ví dụ như các cụm từ cố định (collocations) hoặc sự thay đổi ngữ nghĩa theo ngữ cảnh.

### Ứng dụng của TF-IDF

* **Phân loại văn bản**: TF-IDF thường được sử dụng để trích xuất đặc trưng cho các mô hình phân loại văn bản (email spam detection, sentiment analysis).
* **Tìm kiếm thông tin (Information Retrieval)**: TF-IDF được sử dụng trong các hệ thống tìm kiếm để đánh giá mức độ liên quan của các tài liệu với truy vấn của người dùng.

Tóm lại, **TF-IDF** là một phương pháp mạnh mẽ và phổ biến để đánh giá tầm quan trọng của từ trong các bài toán liên quan đến văn bản, giúp cải thiện chất lượng biểu diễn văn bản so với các phương pháp đơn giản như Bag of Words.

****

1. **import re**: Thư viện re cung cấp các công cụ làm việc với biểu thức chính quy (regular expressions) trong Python. Nó hữu ích trong việc tìm kiếm và thao tác với các mẫu ký tự trong văn bản.
2. **import string**: Thư viện string cung cấp các hằng số và hàm thao tác với chuỗi ký tự, như chữ cái, ký tự trắng, ký tự đặc biệt, v.v.
3. **import nltk**: nltk là thư viện Python mạnh mẽ cho xử lý ngôn ngữ tự nhiên (Natural Language Toolkit). Nó cung cấp các công cụ để phân tích và thao tác với văn bản.
4. **nltk.download('stopwords')**: Dòng lệnh này tải xuống bộ dữ liệu "stopwords" từ nltk. Stopwords là các từ thường không có ý nghĩa lớn trong phân tích ngôn ngữ, như "và", "hoặc", "nhưng", v.v., thường bị loại bỏ trong xử lý văn bản.
5. **nltk.download('wordnet')**: Tải xuống WordNet từ nltk. WordNet là một cơ sở dữ liệu từ vựng tiếng Anh được tổ chức thành các nhóm từ đồng nghĩa, giúp việc xử lý từ vựng và phân tích ngôn ngữ.
6. **from nltk.corpus import stopwords**: Import danh sách stopwords từ nltk.corpus. Danh sách này sẽ được sử dụng để loại bỏ các từ không cần thiết khỏi văn bản.
7. **from nltk.stem import WordNetLemmatizer**: Import công cụ "WordNetLemmatizer" từ thư viện nltk. Lemmatization là quá trình chuyển đổi các từ về dạng cơ bản của chúng, ví dụ như chuyển từ "running" thành "run".
8. **from bs4 import BeautifulSoup**: Import BeautifulSoup từ thư viện bs4. BeautifulSoup là công cụ dùng để phân tích cú pháp HTML và XML, rất hữu ích để trích xuất dữ liệu từ các trang web.
9. **import contractions**: Thư viện contractions giúp mở rộng các dạng rút gọn trong tiếng Anh, ví dụ như chuyển "don't" thành "do not".

### 1. ****Dòng đầu tiên****:

python

Sao chép mã

text = "".join([word.lower() for word in text if word not in string.punctuation])

* **Chức năng**:
  + Loại bỏ dấu câu và chuyển tất cả chữ cái trong văn bản thành chữ thường.
* **Cách thức hoạt động**:
  + word.lower(): Chuyển từng từ trong văn bản thành chữ thường.
  + for word in text: Duyệt qua từng ký tự trong chuỗi text.
  + if word not in string.punctuation: Điều kiện kiểm tra ký tự có phải dấu câu hay không (thư viện string.punctuation chứa danh sách các dấu câu như .,!?).
  + ''.join([...]): Kết hợp tất cả các ký tự lại thành một chuỗi mới mà không có dấu câu.

### 2. ****Dòng thứ hai****:

python

Sao chép mã

text = " ".join([wl.lemmatize(word) for word in text.split() if word not in stop and word.isalpha()])

* **Chức năng**:
  + Lemmatization (đưa các từ về dạng gốc của chúng) và loại bỏ stopwords cùng các từ không phải là chữ cái (như số).
* **Cách thức hoạt động**:
  + wl.lemmatize(word): Dùng hàm lemmatize (từ WordNetLemmatizer đã được import trước đó) để đưa mỗi từ về dạng gốc. Ví dụ, từ "running" sẽ được chuyển về "run".
  + for word in text.split(): Duyệt qua từng từ trong chuỗi text, sau khi chuỗi đã được chia thành các từ riêng lẻ bằng cách sử dụng split().
  + if word not in stop: Điều kiện để loại bỏ stopwords (các từ không mang nhiều ý nghĩa như "and", "the", "is", v.v.). stop là danh sách stopwords được lấy từ NLTK.
  + word.isalpha(): Kiểm tra xem từ có phải là một chuỗi ký tự (không chứa số hoặc ký tự đặc biệt) không.
  + " ".join([...]): Kết hợp các từ lại thành chuỗi, cách nhau bằng dấu cách " ", để tạo ra văn bản sau khi đã lemmatize và lọc từ.

### Tóm tắt:

* **Dòng đầu tiên**: Loại bỏ tất cả dấu câu và chuyển văn bản thành chữ thường.
* **Dòng thứ hai**: Thực hiện lemmatization để đưa từ về dạng gốc, loại bỏ các stopwords và từ không phải là chữ cái (như số hoặc ký tự đặc biệt).

Đoạn code này là một bước quan trọng trong việc tiền xử lý văn bản trước khi thực hiện phân tích ngôn ngữ tự nhiên (NLP), giúp văn bản được làm sạch và chuẩn hóa.