TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO MÔN
KHAI THÁC DỮ LIỆU VÀ KHAI PHÁ TRI THỨC

TIỀN XỬ LÝ DỮ LIỆU:
"CÁC KỸ THUẬT TRÍCH XUẤT ĐỐI TƯỢNG VĂN BẢN"

Giảng viên giảng dạy: **TS. Lê Cung Tưởng**
Người thực hiện: **TÔ VĨNH KHANG - 51800408**
BÙI QUANG KHẢI - 51800785
Khoá : **22**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2021

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO MÔN
KHAI THÁC DỮ LIỆU VÀ KHAI PHÁ TRI THỨC

TIỀN XỬ LÝ DỮ LIỆU:
"CÁC KỸ THUẬT TRÍCH XUẤT ĐỐI TƯỢNG VĂN BẢN"

Giảng viên giảng dạy: **TS. Lê Cung Tưởng**
Người thực hiện: **TÔ VĨNH KHANG - 51800408**
BÙI QUANG KHẢI - 51800785
Khoá : **22**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2021

LỜI CẢM ƠN

Chúng em xin chân thành cảm ơn Khoa Công nghệ thông tin và Trường Đại học Tôn Đức Thắng đã tạo điều kiện cho chúng em được học tập trong suốt thời gian qua. Chân thành cảm ơn Thầy Lê Cung Tưởng đã giúp chúng em có thêm kiến thức về khai thác dữ liệu và khai phá tri thức. Tìm hiểu thêm được nhiều kỹ thuật, phương pháp phân tích dữ liệu ứng dụng thiết thực hiện nay.

Trong quá trình thực hiện bài báo cáo này nhóm vẫn khó tránh khỏi những sai sót không mong muốn, kính mong thầy có thể góp ý và giúp đỡ chúng em. Nhóm xin chân thành cảm ơn thầy.

BÁO CÁO NÀY ĐƯỢC HOÀN THÀNH
TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Chúng em xin cam đoan đây là sản phẩm của riêng chúng em được sự hướng dẫn của thầy Lê Cung Tưởng. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính chúng em thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong bài báo cáo này còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

Nếu phát hiện có bất kỳ sự gian lận nào chúng em xin hoàn toàn chịu trách nhiệm về nội dung bài báo cáo của mình. Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do chúng em gây ra trong quá trình thực hiện (nếu có).

TP. Hồ Chí Minh, ngày 01 tháng 02 năm 2021

Tác giả

(ký tên và ghi rõ họ tên)

Tô Vĩnh Khang

Bùi Quang Khải

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

Phần xác nhận của GV hướng dẫn

_____

_____

_____

_____

_____

_____

_____

Tp. Hồ Chí Minh, ngày tháng năm

(ký và ghi họ tên)

Phần đánh giá của GV chấm bài

_____

_____

_____

_____

_____

_____

_____

Tp. Hồ Chí Minh, ngày tháng năm

(ký và ghi họ tên)

TÓM TẮT

Ngày nay, công nghệ thông tin đã và đang đóng vai trò quan trọng trong xã hội. Nó giúp con người làm việc thoải mái hơn, dễ dàng hơn. Thông qua những bước phân tích dữ liệu, ta có thể đánh giá, dự đoán những vấn đề trong cuộc sống. Từ đó, quản lý và tổ chức công việc đạt hiệu quả cao. Việc ứng dụng khai thác dữ liệu sẽ giúp khai phá tri thức cho nhân loại, cho mọi lĩnh vực của đời sống xã hội.

Ở bài báo cáo này, chúng tôi sẽ giới thiệu khái quát về khai thác dữ liệu và một số chủ đề đã và đang được nghiên cứu hiện nay. Sau đó, tập trung phân tích chi tiết các phương pháp trong kỹ thuật trích xuất đối tượng văn bản và hướng dẫn hiện thực việc phân tích một tập dữ liệu bằng ngôn ngữ lập trình Python dùng thư viện scikit-learn.

MỤC LỤC

DANH MỤC CÁC BẢNG, HÌNH

DANH MỤC HÌNH

DANH MỤC BẢNG

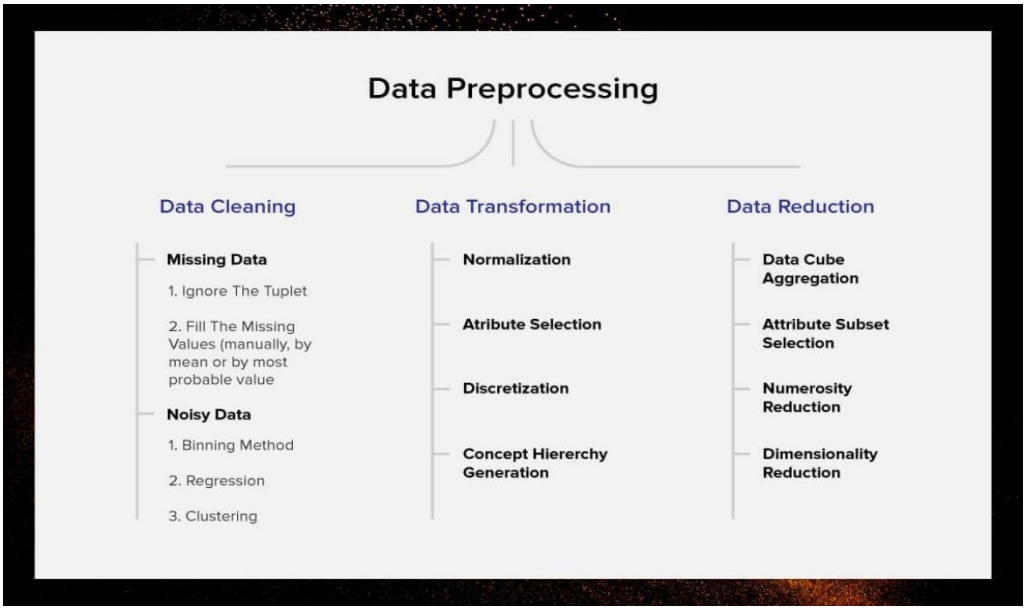
CHƯƠNG I: GIỚI THIỆU CHUNG

1.1 Khai thác dữ liệu (Data Mining)

Là quá trình tính toán để tìm ra các mẫu trong các bộ dữ liệu lớn. Trích xuất thông tin từ một bộ dữ liệu và chuyển nó thành một cấu trúc dễ hiểu để sử dụng tiếp. Ngoài bước phân tích thô, nó còn liên quan tới cơ sở dữ liệu và các khía cạnh quản lý dữ liệu, xử lý dữ liệu trước, suy xét mô hình và suy luận thống kê, các thước đo, các cân nhắc phức tạp, xuất kết quả.

Khai thác dữ liệu có rất nhiều ứng dụng thiết thực trong các lĩnh vực như tài chính, chăm sóc sức khỏe, viễn thông, marketing và sales, thương mại điện tử, giáo dục, kỹ thuật sản xuất, khoa học.

1.2 Tiền xử lý dữ liệu (Data Preprocessing)



Hình 1. Sơ đồ tổng quát về Tiền xử lý dữ liệu trong Khai thác dữ liệu

Là một bước quan trọng trong quá trình khai thác dữ liệu. Ý tưởng là tổng hợp thông tin hiện có và tìm kiếm trong nội dung. Cho phép loại bỏ dữ liệu không mong muốn bằng việc làm sạch dữ liệu, điều này cho phép người dùng có tập dữ liệu để chứa nhiều thông tin có giá trị hơn sau giai đoạn tiền xử lý để thao tác dữ liệu sau này trong quá trình khai thác dữ liệu. Do đó, tính đại diện và chất lượng của dữ liệu là đầu tiên và quan trọng nhất trước khi chạy bất kỳ phân tích nào. Khi có nhiều thông tin không liên quan và dư thừa hiện tại hoặc dữ liệu nhiễu và không đáng tin cậy thì việc khai phá tri thức sẽ khó khăn hơn. Các bước chuẩn bị và lọc dữ liệu có thể mất nhiều thời gian xử lý. Tiền xử lý dữ liệu bao gồm làm sạch, lựa chọn phiên bản, chuẩn hóa, chuyển đổi, trích xuất, lựa chọn tính năng,… Bên cạnh đó, tiền xử lý dữ liệu có thể ảnh hưởng đến cách diễn giải kết quả của quá trình xử lý dữ liệu cuối cùng.

Một số chủ đề đã và đang nghiên cứu hiện nay như: Biến đổi Wavelet, phân tích thành phần chính, lựa chọn tập hợp con thuộc tính, tính toán chi bình phương, phân tích biểu đồ, phương sai, hệ số, chuẩn hóa, kỹ thuật tiền xử lý dữ liệu văn bản, kỹ thuật trích xuất đối tượng văn bản, tính năng cho thị giác máy tính, thu thập thông tin web.

1.3 Thư viện sử dụng

Tên thư viện: ***scikit-learn 0.24.0***

Tên mô-đun con: ***sklearn.feature_extraction.text***

Công dụng: tập hợp các tiện ích để xây dựng các vectơ đặc trưng từ các tài liệu văn bản.

Hình 2. Hình ảnh thư viện scikit-learn của Python

CHƯƠNG II: CÁC KỸ THUẬT TRÍCH XUẤT ĐỐI TƯỢNG VĂN BẢN

2.1 Cơ sở lý thuyết

2.1.1 Mô hình túi từ (Bag of words Model)

Mô hình túi từ là một biểu diễn đơn giản hóa được sử dụng trong xử lý ngôn ngữ tự nhiên và truy xuất thông tin. Trong mô hình này, một văn bản được biểu thị như một túi các từ của nó, không tính đến ngữ pháp và thậm chí cả trật tự từ nhưng vẫn giữ tính đa nghĩa. Mô hình này cũng đã được sử dụng cho thị giác máy tính. Nó mã hóa các chuỗi và cung cấp một "id" số nguyên cho mỗi mã thông báo có thể có, chẳng hạn bằng cách sử dụng khoảng trắng và dấu chấm câu làm dấu phân cách mã thông báo, đếm số lần xuất hiện của mã thông báo trong mỗi tài liệu hay chuẩn hóa và trọng số với các mã thông báo quan trọng giảm dần xảy ra trong phần lớn các mẫu/tài liệu.

Giả sử có 3 người bạn bàn luận với nhau về bức ảnh họ vừa chụp như sau:

Người 1: "Ảnh này rất là đẹp và bóng"

Người 2: "Ành này là không đẹp và là thô"

Người 3: "Ảnh này là mộc và xinh"

Đầu tiên ta sẽ xây dựng vốn từ vựng từ tất cả các từ duy nhất trong 3 lời nhận xét. Từ vựng bao gồm 11 từ sau: 'Ảnh', 'này', 'rất', 'là', 'đẹp', 'và', 'bóng', 'không', 'thô','mộc', 'xinh'.

	Ảnh	**này**	**rất**	**là**	**đẹp**	**và**	**bóng**	**không**	**thô**	**mộc**	**xinh**	**Tổng số**
Người 1	1	1	1	1	1	1	1	0	0	0	0	7
Người 2	1	1	0	2	1	1	0	1	1	0	0	8
Người 3	1	1	0	1	0	1	0	0	0	1	1	6

Bảng 1. Bảng xây dựng ma trận vốn từ vựng của mô hình túi từ

Kết quả mô hình túi từ này là:

Người 1 (N1): [1 1 1 1 1 1 1 0 0 0 0]

Người 2 (N2): [1 1 0 2 1 1 0 1 1 0 0]

Người 3 (N3): [1 1 0 1 0 1 0 0 0 1 1]

2.1.2 Tính thưa thớt của ma trận (Sparsity of Matrix)

Ma trận thưa là ma trận trong đó đa số phần tử có giá trị là 0. Tính thưa thớt (mật độ thưa) của ma trận thể hiện qua việc số phần tử có giá trị bằng 0 chia cho tổng số phần tử. Về bản chất, nó sẽ được nén lại dễ dàng hơn và do đó giảm thiểu yêu cầu lưu trữ trên máy tính.

Giả sử ta có ma trận như sau:

1 2 0 0 0 0 0 0
0 0 3 5 0 0 0 0
0 0 0 0 1 4 2 0
0 0 0 0 0 0 0 3

Ta có thể thấy, với ma trận này có 8 phần tử khác 0 và 24 phần tử bằng 0. Ta dễ dàng tính được mật độ thưa của ma trận này: 24/32 = 0.75 = 75% và mật độ dày sẽ là 1-0.75 = 0.25 = 25%.

2.1.3 Vectơ hóa (Vectorizer)

2.1.3.1 Vectơ hóa thông thường (Common Vectorizer)

Vectơ hóa thông thường là chuyển đổi tập dữ liệu văn bản thành ma trận số lượng mã thông báo. Tần suất xuất hiện của mỗi mã thông báo riêng lẻ được coi là một tính năng. Vectơ của tất cả các tần số mã thông báo cho một tài liệu nhất định được coi là một mẫu đa biến.

Do đó, có thể được biểu diễn bằng một ma trận với một hàng trên mỗi tài liệu và một cột trên mỗi mã thông báo. Đây là phương pháp phân loại tài liệu theo tần suất xuất hiện của từ được sử dụng như một đặc điểm.

2.1.3.2 Vectơ hóa băm (Hashing Vectorizer)

Vectơ hóa băm là chuyển đổi tập dữ liệu văn bản thành một ma trận các lần xuất hiện mã thông báo. Nó sử dụng thủ thuật băm để tìm tên chuỗi mã thông báo để làm nổi bật chỉ mục số nguyên ánh xạ. Khả năng mở rộng bộ nhớ lớn.

Tuy nhiên, Không thể tính toán biến đổi nghịch đảo. Trong một số trường hợp có thể xảy ra xung đột bởi các mã thông báo riêng biệt có thể được ánh xạ tới cùng một chỉ mục tính năng.

2.1.4 Từ dừng (Stop Words)

Từ dừng là những từ ví dụ như "và" được cho là không có ý nghĩa trong việc thể hiện nội dung của một văn bản và có thể bị xóa để tránh chúng được hiểu là tín hiệu để dự đoán. Tuy nhiên, đôi khi, những từ tương tự lại hữu ích cho việc dự đoán, chẳng hạn như phân loại phong cách viết hoặc tính cách.

Nếu một tập dữ liệu được giả định chứa các từ dừng, tất cả các từ này sẽ bị xóa khỏi các mã thông báo kết quả.

2.1.5 Chuẩn hóa Tf-Idf (Tf-Idf Representation)

Tf-Idf là một thống kê số học nhằm phản ánh tầm quan trọng của một từ đối với một văn bản trong một tập hợp hay một ngữ liệu văn bản. Giá trị Tf-Idf được tính bằng cách lấy **Tf(t,d)** tần số xuất hiện của một từ trong văn bản nhân **Idf(t)** tần số nghịch của một từ trong tập văn bản (corpus).

Công thức:

- **Tf(t,d) = n(t,d)/S(d)**

  Với: n(t,d) là Số lần thuật ngữ t xuất hiện trong tài liệu d;

  S(d) là Tổng số thuật ngữ trong tài liệu d
- **Idf(t) = log(S/m(t))**

  Với: S là Tổng số tài liệu

  m(t) là Số lượng tài liệu có chứa thuật ngữ t
- **Tf-Idf = Tf(t,d)*Idf(t)**

Giả sử ta sử dụng lại ví dụ về 3 người ban đầu nhận xét bức ảnh, ta được các kết quả:

- **Tf:** Tf('Ảnh',N1) = 1/7; Tf('này',N1) = 1/7; Tf('rất',N1) = 1/7; Tf('là',N1) = 1/7; Tf('đẹp',N1) = 1/7; Tf('và',N1) = 1/7; Tf('bóng',N1) = 1/7; Tương tự tính toán Tf của 2 người còn lại là N2 và N3

Kết quả tính toán Tf khi hiển thị dưới dạng bảng là:

t	**Người (N)**			**Tf(t,N)**		
	N1	**N2**	**N3**	**Tf(t,N1)**	**Tf(t,N2)**	**Tf(t,N3)**
Ảnh	1	1	1	1/7	1/8	1/6
này	1	1	1	1/7	1/8	1/6
rất	1	0	0	1/7	0	0
là	1	2	1	1/7	1/4	1/6
đẹp	1	1	0	1/7	1/8	0
và	1	1	1	1/7	1/8	1/6

bóng	1	0	0	1/7	0	0
không	0	1	0	0	1/8	0
thô	0	1	0	0	1/8	0
mộc	0	0	1	0	0	1/6
xinh	0	0	1	0	0	1/6

Bảng 2. Bảng kết quả tính toán Tf của một ví dụ

- **Idf:** Idf('Ảnh',) = log(3/3) = 0; Idf('này',) = log(3/3) = 0; Idf('rất') = log(3/1) = 0.48; Idf('là') = log(3/3) = 0; Idf('đẹp') = log(3/2) = 0.18; Idf('và') = log(3/3) = 0; Idf('bóng') = log(3/1) = 0.48; Idf('không') = log(3/1) = log(3) = 0.48; Idf('thô') = log(3/1) = 0.48; Idf('mộc') = log(3/1) = log(3) = 0.48; Idf('xinh') = log(3/1) = log(3) = 0.48;

Kết quả tính toán Idf khi hiển thị dưới dạng bảng là:

t	**Người (N)**			**Idf(t)**
	N1	**N2**	**N3**	
Ảnh	1	1	1	0
này	1	1	1	0
rất	1	0	0	0.48
là	1	2	1	0

đẹp	1	1	0	0.18
và	1	1	1	0
bóng	1	0	0	0.48
không	0	1	0	0.48
thô	0	1	0	0.48
mộc	0	0	1	0.48
xinh	0	0	1	0.48

Bảng 3. Bảng kết quả tính toán Idf của một ví dụ

- **Tf-Idf:** Áp dụng công thức tính tích Tf và Idf

Kết quả tính toán Tf-Idf khi hiển thị dưới dạng bảng là:

t	**Tf-Idf(N)**		
	Tf-Idf(N1)	**Tf-Idf(N2)**	**Tf-Idf(N3)**
Ảnh	0.000	0.000	0.000
này	0.000	0.000	0.000
rất	0.068	0.000	0.000
là	0.000	0.000	0.000

đẹp	0.025	0.022	0.000
và	0.000	0.000	0.000
bóng	0.068	0.000	0.000
không	0.000	0.060	0.000
thô	0.000	0.060	0.000
mộc	0.000	0.000	0.080
xinh	0.000	0.000	0.080

Bảng 4. Bảng kết quả tính toán Tf-Idf của một ví dụ

2.1.6 Giải mã tệp văn bản (Decoding text files)

Văn bản được tạo từ các ký tự, nhưng tệp được tạo bằng byte. Các byte này đại diện cho các ký tự theo một số bảng mã. Để làm việc với các tệp văn bản bằng Python, các byte của chúng phải được giải mã thành một bộ ký tự gọi là Unicode. Các bảng mã phổ biến là ASCII, UTF-8,..

2.2 Sử dụng các lớp có sẵn trong mô-đun con sklearn.feature_extraction.text

- *Yêu cầu: Python(≥ 3.6), NumPy(≥ 1.13.3), SciPy(≥ 0,19,1), Joblib(≥ 0.11), Threadpoolctl (≥2.0.0)*
- *Cài đặt: Dùng lệnh pip:* ***pip install -U scikit-learn*** *Hoặc lệnh conda:* ***conda install -c conda-forge scikit-learn***

2.2.1 CountVectorizer

Các tham số tùy chỉnh của lớp *CountVectorizer:*

- input: *string {'filename', 'file', 'content'}, default='content'*
- encoding: *string, default='utf-8'*
- decode_error: *{'strict', 'ignore', 'replace'}, default='strict'*
- *strip_accents: {'ascii', 'unicode'}, default=None*
- *lowercase: bool, default=True*
- *preprocessor: callable, default=None*
- *tokenizer: callable, default=None*
- *stop_words: string {'english'}, list, default=None*
- *token_pattern: str, default=r"(?u)\b\w\w+\b"*
- *ngram_range: tuple (min_n, max_n), default=(1, 1)*
- *analyzer: {'word', 'char', 'char_wb'} or callable, default='word'*
- *max_df: float in range [0.0, 1.0] or int, default=1.0*
- *min_df: float in range [0.0, 1.0] or int, default=1*
- *max_features: int, default=None*
- *vocabulary: Mapping or iterable, default=None*
- *binary: bool, default=False*
- *dtype: type, default=np.int64*

Các thuộc tính tùy chỉnh của lớp *CountVectorizer:*

- vocabulary_: *dict*
- *fixed_vocabulary_: boolean*
- *stop_words_: set*

Lớp *CountVectorizer* gồm các phương thức:

Phương thức	**Ứng dụng**
build_analyzer()	Trả về một hàm có khả năng quản lý quá trình tiền xử lý, mã hóa và tạo n-grams.

build_preprocessor()	Trả về một hàm để thực hiện tiền xử lý văn bản trước khi mã hóa.
build_tokenizer()	Trả về một hàm có tác dụng cắt chuỗi thành các đoạn mã hóa.
decode(doc)	Có tác dụng giải mã dữ liệu đầu vào dạng chuỗi thành một chuỗi các ký tự unicode.
fit(raw_documents, y=None)	Nhận vào tham số đầu vào là loại tài liệu thô (một vòng lặp có kiểu dữ liệu là chuỗi, ký tự unicode hoặc tập tin). Phương thức này sẽ học từ vựng từ điển của các đoạn mã hóa trong tài liệu thô.
fit_transform(raw_documents, y=None)	Nhận vào tham số đầu vào là loại tài liệu thô (một vòng lặp có kiểu dữ liệu là chuỗi, ký tự unicode hoặc tập tin). Phương thức này sẽ học từ vựng từ điển và trả về ma trận dữ liệu của tài liệu. Phương thức này cũng tương tự như “fit” nhưng có hiệu quả thực thi tốt hơn.
get_feature_names()	Tạo một mảng mới để chuyển đổi dữ liệu từ các số nguyên đặc trưng thành một danh sách chứa các tên đặc trưng.
get_params(deep=True)	Tham số đầu vào có dạng bool, mặc định là True. Nếu là True thì nó sẽ trả về các tham số cho cái mà nó ước lượng và trả về dữ liệu dạng từ điển bao gồm tên tham số ứng với giá trị của nó.
get_stop_words()	Xây dựng hoặc lấy danh sách dừng của các từ ngữ sau

	đó trả về dữ liệu dạng danh sách.
inverse_transform(X)	Nhận vào ma trận dữ liệu và trả về danh sách các tập dữ liệu văn bản.
set_params(**params)	Gán các tham số cho cái mà nó ước lượng, phương thức này nhận vào từ điển của các tham số ước lượng sau đó trả về một từ điển mới đã được gán.
transform(raw_documents)	Nhận vào tham số đầu vào là loại tài liệu thô (một vòng lặp có kiểu dữ liệu là chuỗi, ký tự unicode hoặc tập tin). Sau đó chuyển các tài liệu văn bản thành tập dữ liệu dạng ma trận, phân giải các số đếm mã hóa ra khỏi các đoạn văn bản.

Bảng 5. Bảng các phương thức sử dụng trong lớp CountVectorizer

2.2.2 HashingVectorizer

Các tham số tùy chỉnh của Lớp *HashingVectorizer:*

- input: *string {'filename', 'file', 'content'}, default='content'*
- encoding: *string, default='utf-8'*
- decode_error: *{'strict', 'ignore', 'replace'}, default='strict'*
- *strip_accents: {'ascii', 'unicode'}, default=None*
- *lowercase: bool, default=True*
- *preprocessor: callable, default=None*
- *tokenizer: callable, default=None*
- *stop_words: string {'english'}, list, default=None*
- *token_pattern: str, default=r"(?u)\b\w\w+\b"*

- *ngram_range: tuple (min_n, max_n), default=(1, 1)*
- *analyzer: {'word', 'char', 'char_wb'} or callable, default='word'*
- *n_features: int, default=(2 ** 20)*
- *binary: bool, default=False*
- *norm: {'l1', 'l2'}, default='l2'*
- *alternate_sign: bool, default=True*
- *dtype: type, default=np.float64*

Lớp *HashingVectorizer* gồm các phương thức:

Phương thức	**Ứng dụng**
build_analyzer()	Trả về một hàm có khả năng quản lý quá trình tiền xử lý, mã hóa và tạo n-grams.
build_preprocessor()	Trả về một hàm để thực hiện tiền xử lý văn bản trước khi mã hóa.
build_tokenizer()	Trả về một hàm có tác dụng cắt chuỗi thành các đoạn mã hóa.
decode(doc)	Có tác dụng giải mã dữ liệu đầu vào dạng chuỗi thành một chuỗi các ký tự unicode.
fit(X, y=None)	Nhận vào tham số đầu vào là mảng dữ liệu. Phương thức chuyển đổi này là dạng không trạng thái.
fit_transform(X, y=None)	Nhận vào tham số đầu vào là loại tài liệu thô (một vòng lặp có kiểu dữ liệu là chuỗi, ký tự unicode hoặc tập tin) và nó sẽ được băm và mã hóa. Phương thức này sẽ chuyển đổi một chuỗi các tài liệu thành dạng ma trận.

get_params(deep=True)	Tham số đầu vào có dạng bool, mặc định là True. Nếu là True thì nó sẽ trả về các tham số cho cái mà nó ước lượng và trả về dữ liệu dạng từ điển bao gồm tên tham số ứng với giá trị của nó.
get_stop_words()	Xây dựng hoặc lấy danh sách dừng của các từ ngữ sau đó trả về dữ liệu dạng danh sách.
partial_fit(X, y=None)	Nhận vào tham số đầu vào là mảng dữ liệu. Phương thức chuyển đổi này là dạng không trạng thái, nó sẽ chỉ đánh dấu sự chuyển đổi có thể xảy ra trong quá trình xử lý luồng dữ liệu.
set_params(**params)	Gán các tham số cho cái mà nó ước lượng, phương thức này nhận vào từ điển của các tham số ước lượng sau đó trả về một từ điển mới đã được gán.
transform(X)	Nhận vào tham số đầu vào là loại tài liệu thô (một vòng lặp có kiểu dữ liệu là chuỗi, ký tự unicode hoặc tập tin) và nó sẽ được băm và mã hóa. Sau đó phương thức này sẽ chuyển đổi một chuỗi các tài liệu thành dạng ma trận.

Bảng 6. Bảng các phương thức sử dụng trong lớp HashingVectorizer

2.2.3 TfidfTransformer

Các tham số tùy chỉnh của lớp *TfidfTransformer:*

- *norm: {'l1', 'l2'}, default='l2'*
- *use_idf: bool, default=True*
- *smooth_idf: bool, default=True*

- *sublinear_tf: bool, default=False*

Các thuộc tính tùy chỉnh của lớp *TfidfTransformer:*

- idf_*array of shape (n_features)*

Lớp *TfidfTransformer* gồm các phương thức:

Phương thức	**Ứng dụng**
fit(X, y=None)	Nhận vào tham số đầu vào là một ma trận. Phương thức này sẽ học vecto idf.
fit_transform(X, y=None, **fit_params)	Nhận vào X là một mảng và các tham số chuẩn. Phương thức này sẽ trả về một mảng đã chuyển đổi và chuẩn hóa dữ liệu.
get_params(deep=True)	Tham số đầu vào có dạng bool, mặc định là True. Nếu là True thì nó sẽ trả về các tham số cho cái mà nó ước lượng và trả về dữ liệu dạng từ điển bao gồm tên tham số ứng với giá trị của nó.
set_params(**params)	Gán các tham số cho cái mà nó ước lượng, phương thức này nhận vào từ điển của các tham số ước lượng sau đó trả về một từ điển mới đã được gán.
transform(X, copy=True)	Nhận vào X là một ma trận, copy là một giá trị bool để có thể sao chép X và khởi tạo bản sao hoặc thực hiện các hoạt động trong giới hạn. Phương thức này sẽ chuyển đổi ma trận số đếm thành dạng tf hoặc tf-idf.

Bảng 7. Bảng các phương thức sử dụng trong lớp TfidfTransformer

2.2.4 TfidfVectorizer

Các tham số tùy chỉnh của lớp *TfidfVectorizer*:

- input: *string {'filename', 'file', 'content'}, default='content'*
- encoding: *string, default='utf-8'*
- decode_error: *{'strict', 'ignore', 'replace'}, default='strict'*
- *strip_accents: {'ascii', 'unicode'}, default=None*
- *lowercase: bool, default=True*
- *preprocessor: callable, default=None*
- *tokenizer: callable, default=None*
- *stop_words: string {'english'}, list, default=None*
- *token_pattern: str, default=r"(?u)\b\w\w+\b"*
- *ngram_range: tuple (min_n, max_n), default=(1, 1)*
- *analyzer: {'word', 'char', 'char_wb'} or callable, default='word'*
- *max_df: float in range [0.0, 1.0] or int, default=1.0*
- *min_df: float in range [0.0, 1.0] or int, default=1*
- *max_features: int, default=None*
- *vocabulary: Mapping or iterable, default=None*
- *binary: bool, default=False*
- *dtype: type, default=np.int64*
- *norm: {'l1', 'l2'}, default='l2'*
- *use_idf: bool, default=True*
- *smooth_idf: bool, default=True*
- *sublinear_tf: bool, default=False*

Các thuộc tính tùy chỉnh của lớp *TfidfVectorizer*:

- vocabulary_: *dict*
- *fixed_vocabulary_: bool*
- *idf_: array of shape (n_features,)*

- *stop_words_: set*

Lớp *TfidfVectorizer* gồm các phương thức:

Phương thức	**Ứng dụng**
build_analyzer()	Trả về một hàm có khả năng quản lý quá trình tiền xử lý, mã hóa và tạo n-grams.
build_preprocessor()	Trả về một hàm để thực hiện tiền xử lý văn bản trước khi mã hóa.
build_tokenizer()	Trả về một hàm có tác dụng cắt chuỗi thành các đoạn mã hóa.
decode(doc)	Có tác dụng giải mã dữ liệu đầu vào dạng chuỗi thành một chuỗi các ký tự unicode.
fit(raw_documents, y=None)	Nhận vào tham số đầu vào là loại tài liệu thô (một vòng lặp có kiểu dữ liệu là chuỗi, ký tự unicode hoặc tập tin). Phương thức này sẽ học từ vựng và idf từ chuỗi dữ liệu và vecto hóa chúng.
fit_transform(raw_documents, y=None)	Nhận vào tham số đầu vào là loại tài liệu thô (một vòng lặp có kiểu dữ liệu là chuỗi, ký tự unicode hoặc tập tin). Phương thức này sẽ học từ vựng và idf sau đó trả về ma trận dữ liệu của tài liệu có trọng số tf-idf. Phương thức này cũng tương tự như "fit" nhưng có hiệu quả thực thi tốt hơn.
get_feature_names()	Tạo một mảng mới để chuyển đổi dữ liệu từ các số nguyên đặc trưng thành một danh sách chứa các tên đặc

	trưng.
get_params(deep=True)	Tham số đầu vào có dạng bool, mặc định là True. Nếu là True thì nó sẽ trả về các tham số cho cái mà nó ước lượng và trả về dữ liệu dạng từ điển bao gồm tên tham số ứng với giá trị của nó.
get_stop_words()	Xây dựng hoặc lấy danh sách dừng của các từ ngữ sau đó trả về dữ liệu dạng danh sách.
inverse_transform(X)	Nhận vào ma trận dữ liệu và trả về danh sách các tập dữ liệu văn bản.
set_params(**params)	Gán các tham số cho cái mà nó ước lượng, phương thức này nhận vào từ điển của các tham số ước lượng sau đó trả về một từ điển mới đã được gán.
transform(raw_documents)	Nhận vào tham số đầu vào là loại tài liệu thô (một vòng lặp có kiểu dữ liệu là chuỗi, ký tự unicode hoặc tập tin). Sau đó chuyển các tài liệu văn bản thành tập dữ liệu dạng ma trận có trọng số tf-idf, sử dụng từ vựng và tần số tài liệu (df) được học từ chuẩn hóa (hoặc fit_transform).

Bảng 8. Bảng các phương thức sử dụng trong lớp TfidfVectorizer

CHƯƠNG III: HIỆN THỰC BẰNG CODE VÀ KẾT QUẢ

3.1 Hiện thực bằng code

3.1.1 Sử dụng CountVectorizer

3.1.1.1 Code Ví dụ 1

Đầu tiên, chúng ta sẽ import thư viện của sklearn để sử dụng lớp CountVectorizer:

```
from sklearn.feature_extraction.text import CountVectorizer
corpus = [
        'Welcome to my presentation.',
        'This is my presentation demo.',
        'And this is the first demo.',
        'Is this the class CountVectorizer?',
]
```

Khởi tạo lớp CountVectorizer:

```
vectorizer = CountVectorizer()
```

Sử dụng phương thức fit_transform() của lớp CountVectorizer để phân tích nội dung của tập tài liệu đầu vào thành ma trận thưa:

```
X = vectorizer.fit_transform(corpus)
```

Xuất ra danh sách các tên đặc trưng bằng phương thức get_feature_names(), phương thức này lấy ra tất cả các từ hoặc cụm từ có trong tài liệu nhưng sẽ loại bỏ những từ hoặc cụm từ lặp lại để xuất ra danh sách không bị trùng lặp:

```
print(vectorizer.get_feature_names())
```

Xuất ra ma trận thưa:

```
print(X.toarray())
```

3.1.1.2 Code Ví dụ 2

Chúng ta sẽ import thư viện của sklearn để sử dụng lớp CountVectorizer và dữ liệu đầu vào tương tự như ví dụ 1.

Khởi tạo lớp CountVectorizer với các tham số analyzer, ngram_range và stop_words:

```
vectorizer2 = CountVectorizer(analyzer='word', ngram_range=(2, 2),
stop_words=['and', 'this'])
```

Sử dụng phương thức fit_transform() của lớp CountVectorizer để phân tích nội dung của tập tài liệu đầu vào thành ma trận thưa:

```
X2 = vectorizer2.fit_transform(corpus)
```

Xuất ra danh sách các tên đặc trưng bằng phương thức get_feature_names(), phương thức này lấy ra tất cả các từ hoặc cụm từ có trong tài liệu nhưng sẽ loại bỏ những từ hoặc cụm từ lặp lại để xuất ra danh sách không bị trùng lặp:

```
print(vectorizer2.get_feature_names())
```

Xuất ra ma trận thưa:

```
print(X2.toarray())
```

3.1.2 Sử dụng HashingVectorizer

3.1.2.1 Code Ví dụ 1

Đầu tiên, chúng ta sẽ import thư viện của sklearn để sử dụng lớp HashingVectorizer:

```
from sklearn.feature_extraction.text import HashingVectorizer
corpus = [
        'Welcome to my presentation.',
        'This is my presentation demo.',
        'And this is the first demo.',
        'Is this the class HashingVectorizer?',
]
```

Khởi tạo lớp HashingVectorizer với tham số n_feature để tạo ra 2^4 cột:

*vectorizer = HashingVectorizer(n_features=2**4)*

Sử dụng phương thức fit_transform() của lớp HashingVectorizer để phân tích nội dung của tập tài liệu đầu vào thành ma trận thưa:

X = vectorizer.fit_transform(corpus)

Xuất ra ma trận thưa:

print(X.toarray())

3.1.2.2 Code Ví dụ 2

Ta cũng import thư viên và dữ liệu đầu vào tương tự ví dụ 1.

Khởi tạo lớp HashingVectorizer với tham số n_feature để tạo ra 2^4 cột và binary là True để chuyển ma trận dữ liệu thành mô hình nhị phân:

*vectorizer2 = HashingVectorizer(n_features=2**4, binary=True)*

Sử dụng phương thức fit_transform() của lớp HashingVectorizer để phân tích nội dung của tập tài liệu đầu vào thành ma trận thưa:

X2 = vectorizer2.fit_transform(corpus)

Xuất ra ma trận thưa:

print(X2.toarray())

3.1.3 Sử dụng TfidfTransformer và TfidfVectorizer

3.1.3.1 Code Ví dụ 1

Import thư viện và dữ liệu đầu vào:

```
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.pipeline import Pipeline
import numpy as np
```

```
corpus = [
        'Welcome to my presentation.',
        'This is my presentation demo.',
        'And this is the first demo.',
        'Is this the class TfidfTransformer?',
]
vocabulary = ['welcome', 'demo', 'this', 'is', 'first', 'class', 'presentation', 'my']
```

Phân tích nội dung của tập tài liệu thành ma trận thưa:

```
pipe = Pipeline([('count', CountVectorizer(vocabulary=vocabulary)), ('tfid',
TfidfTransformer())]).fit(corpus)
```

Xuất dữ liệu ma trận thưa:

```
pipe['count'].transform(corpus).toarray()
```

Xuất dữ liệu TfidfTransformer:

```
pipe['tfid'].idf_
```

3.1.3.2 Code Ví dụ 2

Import thư viện và dữ liệu đầu vào:

```
from sklearn.feature_extraction.text import TfidfVectorizer
corpus = [
        'Welcome to my presentation.',
        'This is my presentation demo.',
        'And this is the first demo.',
        'Is this the class TfidfTransformer?',
]
```

Khởi tạo lớp TfidfVectorizer:

```
vectorizer = TfidfVectorizer()
```

Sử dụng phương thức fit_transform() của lớp TfidfVectorizer để phân tích nội dung của tập tài liệu đầu vào thành ma trận thưa:

X = vectorizer.fit_transform(corpus)

Xuất ra danh sách các tên đặc trưng bằng phương thức get_feature_names(), phương thức này lấy ra tất cả các từ hoặc cụm từ có trong tài liệu nhưng sẽ loại bỏ những từ hoặc cụm từ lặp lại để xuất ra danh sách không bị trùng lặp:

print(vectorizer.get_feature_names())

Xuất ra ma trận thưa:

print(X.toarray())

3.2 Kết quả thu được

3.2.1 Sử dụng CountVectorizer

3.2.1.1 Kết quả Ví dụ 1

→ ['and', 'class', 'countvectorizer', 'demo', 'first', 'is', 'my', 'presentation', 'the', 'this', 'to', 'welcome']

→ [[0 0 0 0 0 0 1 1 0 0 1 1]
[0 0 0 1 0 1 1 1 0 1 0 0]
[1 0 0 1 1 1 0 0 1 1 0 0]
[0 1 1 0 0 1 0 0 1 1 0 0]]

3.2.1.2 Kết quả Ví dụ 2

→ ['class countvectorizer', 'first demo', 'is my', 'is the', 'my presentation', 'presentation demo', 'the class', 'the first', 'to my', 'welcome to']

→ [[0 1 1 0 1 0 0 0]
[1 0 1 1 0 1 0 0]

[0 0 1 0 0 0 1 1]

[0 1 1 0 1 0 0 0]]

3.2.2 Sử dụng HashingVectorizer

3.2.2.1 Kết quả Ví dụ 1

→ [[0. 0. 0. 0. 0. -0.5

0. 0. 0. 0. 0.5 0.5

0. 0. 0.5 0.]

[0. 0. 0. 0. 0. 0.

-0.37796447 0. 0. 0. 0.37796447 0.

0. 0.37796447 0.75592895 0.]

[0. 0. 0. 0. 0. 0.

-0.70710678 0. -0.70710678 0. 0. 0.

0. 0. 0. 0.]

[0. 0. -0.57735027 0. 0. 0.

0. 0. 0. 0. 0. 0.57735027

0. 0.57735027 0. 0.]]

3.2.2.2 Kết quả Ví dụ 2

→ [[0. 0. 0. 0. 0. 0.5 0. 0. 0. 0. 0.5 0.5 0. 0. 0.5 0.]

[0. 0. 0. 0. 0. 0. 0.5 0. 0. 0. 0.5 0. 0. 0.5 0.5 0.]

[0. 0. 0. 0. 0. 0. 0.5 0. 0.5 0. 0. 0. 0. 0.5 0.5 0.]

[0. 0. 0.5 0. 0. 0. 0. 0. 0. 0. 0. 0.5 0. 0.5 0.5 0.]]

3.2.3 Sử dụng TfidfTransformer và TfidfVectorizer

3.2.3.1 Kết quả Ví dụ 1

→ [[1, 0, 0, 0, 0, 0, 1, 1],

[0, 1, 1, 1, 0, 0, 1, 1],

[0, 1, 1, 1, 1, 0, 0, 0],

[0, 0, 1, 1, 0, 1, 0, 0]]

→ [1.91629073, 1.51082562, 1.22314355, 1.22314355, 1.91629073,

1.91629073, 1.51082562, 1.51082562]

3.2.3.2 Kết quả Ví dụ 2

→ ['and', 'class', 'demo', 'first', 'is', 'my', 'presentation', 'tfidftransformer', 'the', 'this', 'to', 'welcome']

→ [[0. 0. 0. 0. 0. 0.43779123

0.43779123 0. 0. 0. 0.55528266 0.55528266]

[0. 0. 0.48163503 0. 0.38992506 0.48163503

0.48163503 0. 0. 0.38992506 0. 0.]

[0.49641358 0. 0.39137817 0.49641358 0.31685436 0.

0. 0. 0.39137817 0.31685436 0. 0.]

[0. 0.53944516 0. 0. 0.34432086 0.

0. 0.53944516 0.42530476 0.34432086 0. 0.]]

CHƯƠNG IV: TỔNG KẾT

Bài báo cáo này đã giới thiệu về một số kỹ thuật thông dụng trong việc trích xuất và xử lý dữ liệu của văn bản, liệt kê một số tính năng nổi trội của thư viện scikit-learn. Các kết quả thu được từ những ví dụ đã giúp cho việc tìm hiểu về cách thức khai phá dữ liệu văn bản dễ hiểu hơn. Có thể phân tích, dự đoán được các ý nghĩa của dữ liệu đầu vào, rút gọn tập dữ liệu,...

Qua đó, nhóm đã hiểu khái quát hơn về công cụ, công nghệ mới hiện nay cũng như những ứng dụng thực tiễn của việc trích xuất đối tượng văn bản. Trong tương lai, nhóm có thể sẽ tìm hiểu thêm một số phương pháp, kỹ thuật khác trong việc phân tích dữ liệu người dùng, tổng hợp dữ liệu và dự đoán tính cách người dùng.

TÀI LIỆU THAM KHẢO

Tài liệu Tiếng Anh:

[1]: J.Han, M.Kamber, J.Pei, [2011], Data Mining Concepts and Techniques 3rd Edition, Illinois University, Urbana-Champaign, 83-123.

[2]: S.A.Alasadi, W.S.Bhaya , [2017], Review of data preprocessing techniques in data mining, College of Information Technology, Iraq, 4102–4107.

[3]: Scikit-learn Developers, [2020], Scikit-learn User guide - Release 0.23.2, 1930-1949.

Tài liệu Internet:

[4]: https://www.geeksforgeeks.org/feature-extraction-techniques-nlp/

[5]: https://www.geeksforgeeks.org/data-preprocessing-in-data-mining/

[5]: https://scikit-learn.org/stable/modules/feature_extraction.html