Ứng dụng SVD giảm chiều dữ liệu trong phân loại văn bản

Giảng viên: T.S. Nguyễn Thị Hoài Thương

Nhóm: CHIU NHƯNG CHẤT

Thành viên:

Lê Thanh Tiến - 21110409

Phan Hồng Trâm - 21110414

Trần Anh Quân - 21110374

Trần Sĩ Tâm - 21110391

Lương Hồng Thái - 21110393

20/06/2024

Contents

Chapter 1. SVD là gì?

1.1. Phát biểu SVD

Phương pháp phân tích suy biến (Singular Value Decomposition) là phương pháp phân tích ma trận A kích thước $m \times n$ thành ba ma trận:

$$A_{m\times n} = U_{m\times m}\Sigma_{m\times n}(V_{n\times n})^T \quad (1)$$

Trong đó:

- U là ma trận trực giao có kích thước $m \times m$
- Σ là ma trận đường chéo có kích thước $m \times n$
- V là ma trận trực giao có kích thước $n \times n$

1.2. Tại sao cần phải giảm chiều dữ liệu?

Quá trình phân tích suy biến một ma trận có kích thước lớn sẽ:

- Trải qua nhiều bước phức tạp
- Tốn rất nhiều thời gian
- Chi phí lưu trữ lớn

Vì vậy, chúng ta cần sử dụng **"các dạng giảm chiều của phân tích suy biến"** để rút gọn quá trình tính toán và lưu trữ.

Compact SVD:

Định nghĩa: Compact SVD là một phương pháp tính toán ma trận SVD với kích thước nhỏ hơn so với ma trận gốc. Thay vì tính toán SVD đầy đủ, compact SVD chỉ tính toán các vector đặc trưng và giá trị đặc trưng tương ứng với k giá trị đặc trưng lớn nhất của ma trận.

1.3. Các dạng giảm chiều của phân tích suy biến

Compact SVD:

Giả sử trên đường chéo chính của ma trận Σ có r giá trị khác 0. Nghĩa là:

$$\lambda_1 \geq \lambda_2 \geq \lambda_3 ... \geq \lambda_r \geq 0.$$

Ta có thể biểu diễn phân tích suy biến dưới dạng tổng của các vector cột $u_i \in \mathbb{R}^m$ của U và các vector dòng $v_i \in \mathbb{R}^n$ của V^T như sau:

$$A = \sum_{i=1}^{r} u_i \lambda_i v_i.$$

Khi đó ta có một phân tích suy biến gọn hơn gọi là **Compact SVD**:

$$A = U_r \Sigma_r V_r^T.$$

Trong đó:

- U_r và V_r lần lượt là ma trận được tạo bởi r cột đầu tiên của U và V.
- Σ_r là ma trận được tạo ra bởi r cột và r hàng đầu tiên của Σ.

Truncated SVD:

Định nghĩa: Truncated SVD là phương pháp tính toán một ước lượng ma trận với kích thước nhỏ hơn, dựa trên các vector và giá trị đặc trưng lớn nhất của ma trận gốc.

1.3. Các dạng giảm chiều của phân tích suy biến

Truncated SVD:
Trong ma trận Σ, các giá trị trên đường chéo chính là không âm và giảm dần. Nghĩa là:

$$\lambda_1 \geq \lambda_2 \geq \lambda_3 ... \geq \lambda_r \geq 0.$$

Thông thường, chỉ có một số giá trị riêng λ_i mang giá trị lớn, các giá trị còn lại thường nhỏ và gần bằng 0. Khi đó với phương pháp **Truncated SVD**, ta có thể xấp xỉ ma trận A bằng tổng k<r ma trận có hạng 1:

$$A \approx A_k = U_k \Sigma_k V_k^T$$

Hoặc ta có thể biểu diễn dưới dạng tổng của tích vô hướng của các vectơ cột và dòng của U, V^T như sau:

$$A \approx A_k = \sum_{i=1}^{k} \lambda_i \Sigma_i V_i^T$$

Chapter 2. Ứng dụng SVD giảm chiều

2. Ứng dụng SVD giảm chiều

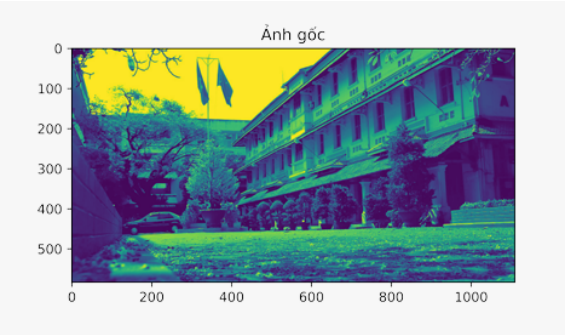
Giảm chiều trong Phân tích Giá trị Đặc biệt (SVD - Singular Value Decomposition) là một kỹ thuật mạnh mẽ được ứng dụng rộng rãi trong nhiều lĩnh vực khác nhau. Dưới đây là một số ứng dụng chính của giảm chiều trong SVD:

Nén dữ liệu (Data Compression):

- **Nén ảnh (Image Compression)**: SVD có thể được sử dụng để nén hình ảnh bằng cách giữ lại các thành phần giá trị đặc biệt lớn nhất và loại bỏ các thành phần nhỏ hơn, từ đó giảm kích thước tập tin ảnh mà vẫn giữ được chất lượng tương đối cao.
- **Nén văn bản (Text Compression)**: Tương tự, SVD có thể áp dụng để nén dữ liệu văn bản, giúp giảm dung lượng lưu trữ và tăng tốc độ truyền tải dữ liệu.

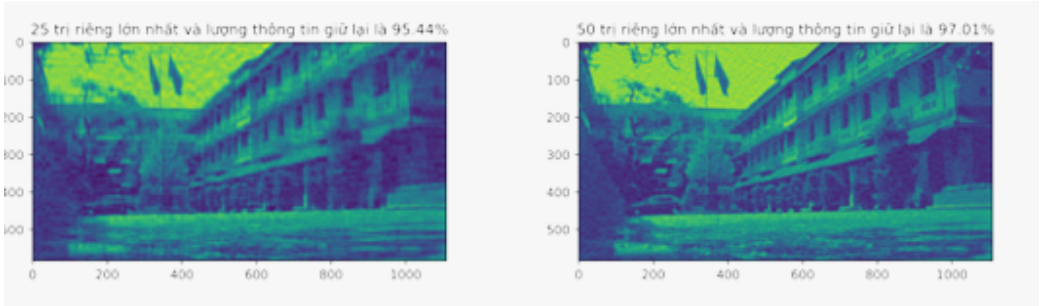
2. Ứng dụng SVD giảm chiều

Ví dụ:
Đây là ảnh gốc:



Hình: Ảnh gốc

Đây là kết quả nén ảnh với lần lượt 25 và 50 trị riêng lớn nhất:



Hình: Ảnh sau khi nén

2. Ứng dụng SVD giảm chiều

Truy hồi thông tin (Information Retrieval):

- **Tìm kiếm văn bản (Text Retrieval)**: Trong các hệ thống tìm kiếm, SVD có thể được sử dụng để giảm chiều dữ liệu văn bản, giúp tăng hiệu quả và tốc độ truy xuất thông tin.
- **Phân tích ngữ nghĩa tiềm ẩn (Latent Semantic Analysis - LSA)**: SVD được áp dụng để xác định mối quan hệ ngữ nghĩa giữa các từ trong văn bản, từ đó cải thiện chất lượng tìm kiếm và phân loại văn bản.

Học máy (Machine Learning):

- **Giảm chiều dữ liệu (Dimensionality Reduction)**: SVD được sử dụng để giảm chiều dữ liệu, giúp cải thiện hiệu suất và độ chính xác của các mô hình học máy bằng cách loại bỏ các thành phần không quan trọng và giảm nhiễu.
- **Phân cụm (Clustering)**: Giảm chiều bằng SVD giúp tăng cường độ chính xác của các thuật toán phân cụm bằng cách giảm số chiều của không gian dữ liệu.

2. Ứng dụng SVD giảm chiều

Xử lý tín hiệu (Signal Processing):

- **Lọc nhiễu (Noise Reduction)**: SVD có thể được sử dụng để lọc nhiễu từ tín hiệu, bằng cách giữ lại các giá trị đặc biệt chính và loại bỏ các giá trị đặc biệt nhỏ, thường đại diện cho nhiễu.
- **Phục hồi tín hiệu (Signal Reconstruction)**: SVD giúp trong việc phục hồi tín hiệu bị mất hoặc hư hỏng bằng cách tái tạo tín hiệu từ các giá trị đặc biệt lớn nhất.

2. Ứng dụng SVD giảm chiều

Phân tích ma trận (Matrix Analysis):

- **Giải hệ phương trình (Solving Linear Systems)**: SVD cung cấp một phương pháp hiệu quả để giải các hệ phương trình tuyến tính, đặc biệt là các hệ phương trình không xác định hoặc dư xác định.
- **Tính toán giá trị đặc biệt (Eigenvalue Computation)**: SVD giúp tính toán các giá trị đặc biệt của ma trận, phục vụ cho nhiều bài toán trong toán học và khoa học.

Đề xuất sản phẩm (Product Recommendation):

- **Hệ thống gợi ý (Recommendation Systems)**: SVD được sử dụng trong các hệ thống gợi ý để phân tích và dự đoán sở thích của người dùng dựa trên dữ liệu về các lựa chọn trước đó, như trong hệ thống gợi ý phim, âm nhạc, sách, v.v.

2. Ứng dụng SVD giảm chiều

Phân tích tài chính (Financial Analysis):

- **Phân tích rủi ro (Risk Analysis)**: SVD giúp phân tích các yếu tố rủi ro trong tài chính bằng cách giảm chiều dữ liệu tài chính và xác định các yếu tố quan trọng ảnh hưởng đến rủi ro.
- **Quản lý danh mục đầu tư (Portfolio Management)**: Giảm chiều bằng SVD giúp xác định các yếu tố chính ảnh hưởng đến hiệu suất của danh mục đầu tư, từ đó đưa ra các quyết định đầu tư hiệu quả hơn.

Chapter 3. Áp dụng SVD giảm chiều trong phân loại văn bản

3.1. Lý thuyết về SVM

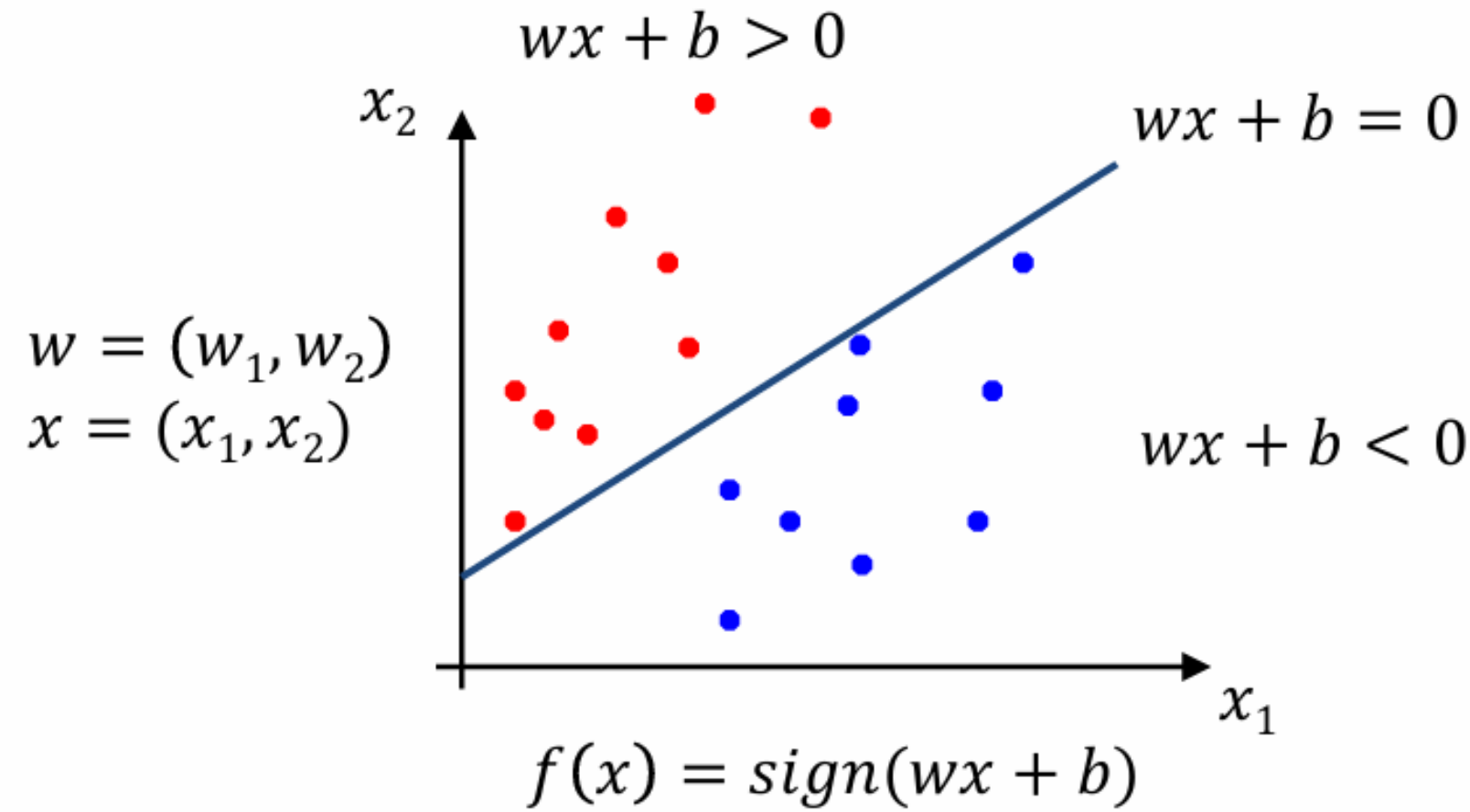
Máy Vector Hỗ Trợ (Support Vector Machine, SVM) là một thuật toán học máy được sử dụng cho các bài toán phân loại và hồi quy. SVM chủ yếu được biết đến với khả năng phân loại dữ liệu trong không gian nhiều chiều và tìm ra siêu phẳng tối ưu để phân tách các lớp dữ liệu khác nhau. Trong không gian nhiều chiều, siêu phẳng có dạng:

$$\mathrm{w} \cdot \mathrm{x} - b = 0$$

Trong đó:

- w là vector trọng số.
- x là vector đặc trưng của điểm dữ liệu.
- b là hằng số.

3.1. Lý thuyết về SVM



3.1. Lý thuyết về SVM

Mục tiêu của SVM là tìm w và b sao cho siêu phẳng phân tách tối ưu được xác định. Điều này được thực hiện bằng cách giải bài toán tối ưu hóa sau:

$$\min_{\mathrm{w},b} \frac{1}{2}\|\mathrm{w}\|^2$$

với ràng buộc:

$$y_i(\mathrm{w} \cdot \mathrm{x}_i - b) \geq 1, \quad \forall i$$

Trong đó $y_i \in \{-1, 1\}$ là nhãn lớp của điểm dữ liệu x_i.

Hàm Kernel:
Trong nhiều trường hợp, dữ liệu không thể phân tách tuyến tính trong không gian đầu vào. SVM sử dụng hàm kernel để ánh xạ dữ liệu vào không gian đặc trưng cao hơn, nơi mà dữ liệu có thể phân tách tuyến tính.
Một số hàm kernel phổ biến:

- **Kernel tuyến tính**: $K(\mathrm{x}_i, \mathrm{x}_j) = \mathrm{x}_i \cdot \mathrm{x}_j$
- **Kernel Gaussian (RBF)**: $K(\mathrm{x}_i, \mathrm{x}_j) = \exp(-\gamma \|\mathrm{x}_i - \mathrm{x}_j\|^2)$
- **Kernel bậc đa thức**: $K(\mathrm{x}_i, \mathrm{x}_j) = (\mathrm{x}_i \cdot \mathrm{x}_j + c)^d$

Ưu Điểm:

- **Hiệu Quả Cao**: SVM hiệu quả trong không gian nhiều chiều và có khả năng làm việc tốt với các dữ liệu có số lượng mẫu lớn.
- **Tránh Overfitting**: Bằng cách tối đa hóa biên độ phân cách, SVM giảm thiểu nguy cơ overfitting, đặc biệt hữu ích khi số lượng đặc trưng lớn hơn số lượng mẫu.
- **Linh Hoạt với Các Hàm Kernel**: Khả năng sử dụng các hàm kernel cho phép SVM xử lý các bài toán phân loại phi tuyến tính.

Nhược Điểm:

- **Chi Phí Tính Toán Cao**: Đối với các bộ dữ liệu rất lớn, SVM có thể tiêu tốn nhiều tài nguyên tính toán và thời gian để tìm siêu phẳng tối ưu.
- **Khó Hiểu và Điều Chỉnh Tham Số**: Việc lựa chọn và điều chỉnh các tham số như loại hàm kernel, giá trị γ, và hệ số phạt C có thể phức tạp và cần thử nghiệm nhiều.

Ứng Dụng Thực Tiễn:
SVM được sử dụng rộng rãi trong nhiều lĩnh vực, bao gồm:

- **Phân loại văn bản và thư rác**
- **Nhận dạng chữ viết tay**
- **Phân loại hình ảnh và nhận dạng đối tượng**
- **Sinh học tính toán và phân tích gene**

$\Rightarrow$ Sử dụng SVM kết hợp với SVD giảm chiều dữ liệu để phân loại văn bản.

3.2. Phần code

3.2. Khai báo thư viện và đường dẫn

- Các thư viện được sử dụng trong bài code:

```
import os
import glob
import shutil
from sklearn.feature_extraction.text import
    TfidfVectorizer
from sklearn.decomposition import TruncatedSVD
from sklearn.svm import SVC
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import LabelEncoder
```

- Khai báo đường dẫn tới thư mục chứa data cần huấn luyện:

```
BASE_DIR = '/content/drive/MyDrive/PPSKHDL_NHOM_/
    DATA_TRAIN'
TEST_DIR = '/content/drive/MyDrive/PPSKHDL_NHOM_/
    DATA_TEST'
RESULT_DIR = '/content/drive/MyDrive/PPSKHDL_NHOM_/
    RESULT'
```

3.3. Các hàm được sử dụng

- Hàm `read_stopwords`
- Hàm `segmentation`
- Hàm `split_words`
- Hàm `remove_special_chars`
- Hàm `get_clean_text`
- Hàm `read_data_from_directory`
- Hàm `classify_and_copy_files`

3.3.1 Hàm read_stop_word

Hàm `read_stopwords(path)`:

```
def read_stopwords(path):
    with open(path, 'r', encoding='utf-8') as file:
        return set(file.read().splitlines())
```

- Đọc danh sách stopword từ tệp tin `stopwords_VIE.txt`.
- Trả về một tập hợp (set) các stopword.

Tại sao phải loại bỏ stop words?

Stop words là các từ thường xuất hiện rất nhiều trong ngôn ngữ nhưng không có nhiều giá trị ngữ nghĩa, chẳng hạn như "và", "của", "là", "một" trong tiếng Việt, hoặc "and", "the", "is" trong tiếng Anh. Các từ này thường không đóng góp nhiều vào việc hiểu nội dung chính của văn bản.

Lý do loại bỏ stop words:

- **Giảm Số Lượng Từ**: Giảm số lượng từ cần xử lý giúp các thuật toán hoạt động nhanh hơn và hiệu quả hơn.
- **Tập Trung Vào Từ Quan Trọng**: Loại bỏ các từ không mang nhiều thông tin giúp tập trung vào các từ quan trọng hơn trong văn bản.
- **Giảm Nhiễu**: Loại bỏ các từ không cần thiết giúp giảm nhiễu trong dữ liệu.

3.3.2. Hàm segmentation

Hàm `segmentation(text)`:

```
def segmentation(text):
    from pyvi import ViTokenizer
    return ViTokenizer.tokenize(text)
```

- Sử dụng thư viện `pyvi` để tách từ trong văn bản tiếng Việt.
- Trả về văn bản đã được phân đoạn.

3.3.3. Hàm split_words

Hàm `split_words(text)`:

```
def split_words(text):
    text = segmentation(text)
    return text.split()
```

- Chia văn bản thành các từ bằng cách sử dụng hàm `segmentation()` phía trên.

3.3.4. Hàm remove_special_chars

Hàm `remove_special_chars(word, special_chars)`:

```
def remove_special_chars(word, special_chars):
    return ''.join(char for char in word if char not in
        special_chars)
```

- Loại bỏ các ký tự đặc biệt khỏi một từ.
- Trả về từ mới đã được loại bỏ ký tự đặc biệt.

3.3.5. Hàm get_clean_text

Hàm `get_clean_text(text, stopwords, special_chars)`:

```
def get_clean_text(text, stopwords, special_chars):
    words = split_words(text)
    return ' '.join([remove_special_chars(word.lower(),
        special_chars) for word in words if word.lower()
        not in stopwords])
```

- Thực hiện các bước xử lý tiền xử lý văn bản:
  - Phân đoạn văn bản.
  - Tách từ.
  - Loại bỏ các stop words.
  - Loại bỏ các ký tự đặc biệt.
- Trả về văn bản đã được làm sạch.

3.3.6. Hàm read_data_from_directory

```
def read_data_from_directory(base_dir):
    texts = []
    labels = []
    for category in os.listdir(base_dir):
        category_path = os.path.join(base_dir, category)
        if os.path.isdir(category_path):
            for file_path in glob.glob(os.path.join(
                category_path, '*.txt')):
                with open(file_path, 'r', encoding='utf
                    -8') as file:
                    text = file.read()
                    texts.append(get_clean_text(text,
                        stopwords, special_chars))
                    labels.append(category)
    return texts, labels
```

- Đọc các file văn bản từ các thư mục con trong `base_dir`.
- Áp dụng hàm `get_clean_text()` để tiền xử lý văn bản.
- Trả về danh sách các văn bản đã làm sạch và nhãn tương ứng.

3.3.7. Hàm classify_and_copy_files

Hàm `classify_and_copy_files`:

```
def classify_and_copy_files(test_dir, result_dir,
    pipeline, label_encoder):
    for file_path in glob.glob(os.path.join(test_dir, '
        *.txt')):
        with open(file_path, 'r', encoding='utf-8') as
            file:
            text = file.read()
            if text.strip():
                test_text_clean = get_clean_text(text,
                    stopwords, special_chars)
                prediction = pipeline.predict([
                    test_text_clean])
                predicted_label = label_encoder.
                    inverse_transform(prediction)[0]
```

3.3.7. Hàm classify_and_copy_files

```
                destination_dir = os.path.join(
                    result_dir, predicted_label)
                os.makedirs(destination_dir, exist_ok=
                    True)  # Tao thu muc neu chua ton tai
                shutil.copy(file_path, destination_dir)
                     # Copy file vao thu muc tuong ung
                print(f"Copied {file_path} to {
                    destination_dir}")
            else:
                print(f"Error: Empty test file {
                    file_path}.")
```

- Đọc các file văn bản từ các thư mục con trong `base_dir`.
- Áp dụng hàm `get_clean_text()` để làm sạch từng văn bản.
- Trả về danh sách các văn bản đã làm sạch và nhãn tương ứng.

3.4. Phần xử lý chính

- Đọc danh sách từ dừng từ tệp `stopwords_VIE.txt`:

```
stopwords = read_stopwords('/content/drive/MyDrive/
    PPSKHDL_NHOM_/stopwords_VIE.txt')
```

- Tệp `stopwords_VIE.txt` có dạng:

```
a lô
a ha
ai
ai ai
ai nấy
ai đó
alô
amen
anh
anh ấy
ba
ba ba
ba bản
ba cùng
ba họ
ba ngày
ba ngôi
```

3.4. Phần xử lý chính

- Định nghĩa tập hợp các ký tự đặc biệt:

```
special_chars = set(".,!?\"':;()")
```

- Đọc dữ liệu huấn luyện từ thư mục `BASE_DIR` sử dụng hàm `read_data_from_directory()`:

```
train_texts, train_labels = read_data_from_directory(
    BASE_DIR)
```

3.4. Phần xử lý chính

- In ra các nhãn trong tập train và số lượng nhãn:

```
print("Unique labels:", set(train_labels))
print("Number of unique labels:", len(set(train_labels))
    )
```

```
Unique labels: {'GIAO DUC', 'XA HOI', 'VAN HOA', 'THE THAO', 'PHAP LUAT', 'KINH TE', 'Y TE'}
Number of unique labels: 7
```

3.4. Phần xử lý chính

- Nếu số lượng nhãn duy nhất nhỏ hơn hoặc bằng 1, in ra thông báo lỗi.

```
if len(set(train_labels)) <= 1:
    print("Error: The number of classes has to be
        greater than one.")
```

3.4. Phần xử lý chính

- Nếu số lượng nhãn duy nhất lớn hơn 1, ta bắt đầu thực hiện phân loại dữ liệu bằng các bước sau:

```
else:
    # Ma hoa thanh cac so nguyen
    label_encoder = LabelEncoder()
    train_labels_encoded = label_encoder.fit_transform(
        train_labels)
    # Tao pipeline voi TfidfVectorizer, TruncatedSVD va
        SVM
    pipeline = make_pipeline(
        TfidfVectorizer(),
        TruncatedSVD(n_components=200),
        SVC()
    )
```

3.4. Phần xử lý chính

```
# Huan luyen mo hinh voi du lieu huan luyen
pipeline.fit(train_texts, train_labels_encoded)
# Phan loai va sao chep cac file test vao thu muc
    tuong ung
classify_and_copy_files(TEST_DIR, RESULT_DIR,
    pipeline, label_encoder)
```

3.4. Phần xử lý chính

Cụ thể:

- Mã hóa nhãn thành số nguyên: Nếu số lượng lớp lớn hơn 1, đoạn code sẽ sử dụng `LabelEncoder` để mã hóa các nhãn thành số nguyên, lưu trữ kết quả vào biến `train_labels_encoded`.
- Tạo một pipeline gồm:
  - `TfidfVectorizer()`: Chuyển đổi văn bản thành ma trận tần suất từ.
  - `TruncatedSVD(n_components=200)`: Thực hiện giảm chiều dữ liệu xuống 200 chiều.
  - `SVC()`: Sử dụng SVM để phân loại.
- Huấn luyện mô hình: `pipeline` được huấn luyện trên tập dữ liệu huấn luyện `train_text` và `train_labels_encoded`.
- Phân loại và sao chép các file trong thư mục `TEST_DIR` vào thư mục `RESULT_DIR` tương ứng với nhãn dự đoán.

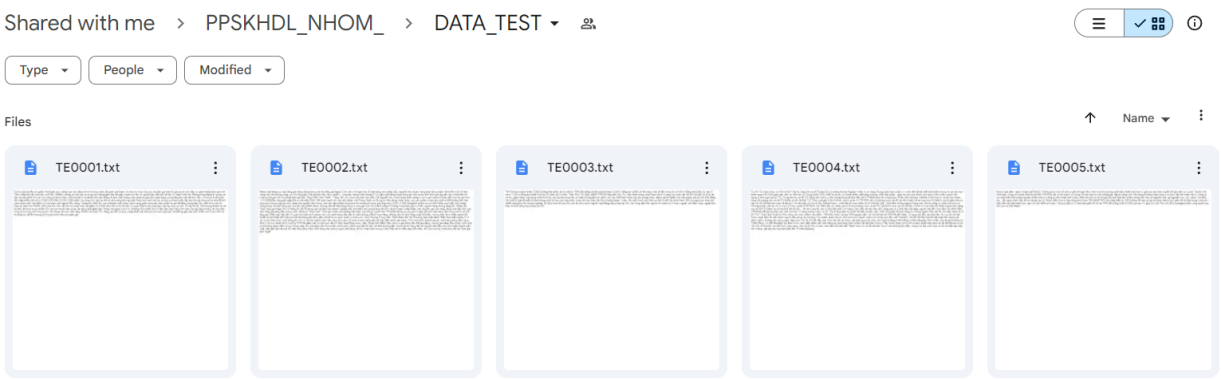
3.4. Phần xử lý chính

Ví dụ:

- Ta có các mục data trong tập train như sau:

Hình: Các thư mục trong tập train

- Ta lại có các file văn bản trong tập test như sau (gồm 861 files):



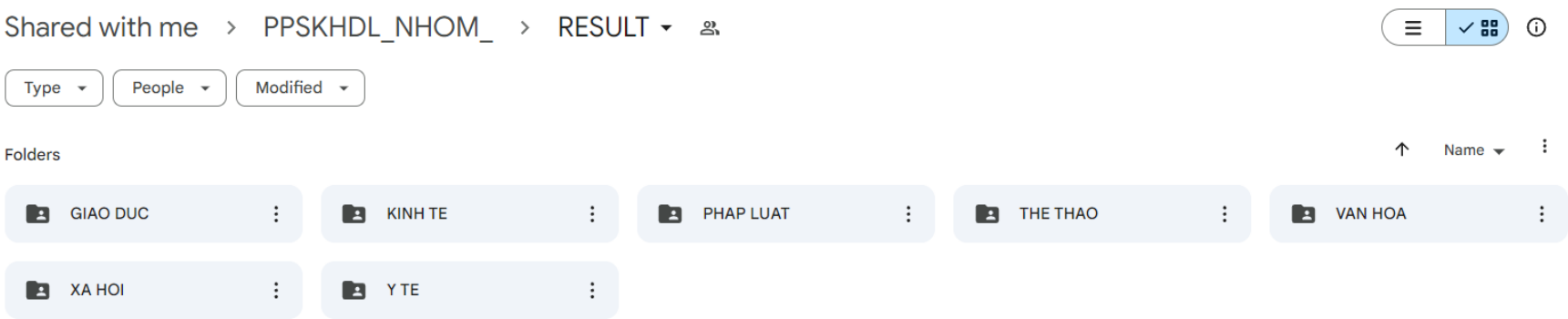
Hình: Các file văn bản trong tập test

3.4. Phần xử lý chính

- Sau khi chạy chương trình, ta in ra số nhãn và phân loại các file dữ liệu vào thư mục tương ứng:

```
Unique labels: {'GIAO DUC', 'XA HOI', 'VAN HOA', 'THE THAO', 'PHAP LUAT', 'KINH TE', 'Y TE'}
Number of unique labels: 7
Copied /content/drive/MyDrive/PPSKHDL_NHOM_/DATA_TEST/TE0001.txt to /content/drive/MyDrive/PPSKHDL_NHOM_/RESULT/XA HOI
Copied /content/drive/MyDrive/PPSKHDL_NHOM_/DATA_TEST/TE0002.txt to /content/drive/MyDrive/PPSKHDL_NHOM_/RESULT/XA HOI
Copied /content/drive/MyDrive/PPSKHDL_NHOM_/DATA_TEST/TE0003.txt to /content/drive/MyDrive/PPSKHDL_NHOM_/RESULT/XA HOI
Copied /content/drive/MyDrive/PPSKHDL_NHOM_/DATA_TEST/TE0004.txt to /content/drive/MyDrive/PPSKHDL_NHOM_/RESULT/XA HOI
Copied /content/drive/MyDrive/PPSKHDL_NHOM_/DATA_TEST/TE0005.txt to /content/drive/MyDrive/PPSKHDL_NHOM_/RESULT/XA HOI
Copied /content/drive/MyDrive/PPSKHDL_NHOM_/DATA_TEST/TE0006.txt to /content/drive/MyDrive/PPSKHDL_NHOM_/RESULT/KINH TE
```

Hình: Kết quả sau khi chạy chương trình



Hình: Các file dữ liệu trong tập test đã được phân loại vào thư mục tương ứng

3.4. Phần xử lý chính

$\Rightarrow$ Tóm lại, đoạn code này thực hiện các bước xử lý tiền xử lý văn bản, huấn luyện một mô hình phân loại SVM, và sử dụng mô hình đó để dự đoán nhãn của một văn bản mới.

Đường dẫn đến các folder:

- Dữ liệu huấn luyện: `DATA_TRAIN`
- Dữ liệu kiểm tra: `DATA_TEST`
- Kết quả phân loại văn bản: `RESULT`

Chapter 4. Nhận xét kết quả đạt được

4. Nhận xét kết quả đạt được

Những gì đã làm được:

- **Cải thiện hiệu suất phân loại**: Điều này cho thấy việc giảm chiều dữ liệu bằng SVD đã giúp tránh hiện tượng quá khớp, tăng khả năng khái quát hóa của mô hình.
- **Tăng tốc độ và hiệu quả tính toán**: Thời gian huấn luyện và suy luận mô hình phân loại, cũng như bộ nhớ sử dụng, đều giảm đáng kể khi sử dụng dữ liệu chiều thấp. Điều này cho phép xử lý các bộ dữ liệu văn bản lớn hơn trên cùng một phần cứng.
- **Khám phá các đặc trưng quan trọng**: Phân tích các vector đặc trưng chính (principal components) từ SVD đã giúp khám phá các đặc trưng ngữ nghĩa quan trọng trong dữ liệu văn bản, như các chủ đề chính.

4. Nhận xét kết quả đạt được

Những gì chưa làm được:

- **Phân tích sâu hơn về các vector đặc trưng chính:** Mặc dù đã gợi ý rằng các vector đặc trưng chính đầu tiên có thể tương ứng với các chủ đề chính, nhưng cần có thêm phân tích chi tiết hơn để xác định mối liên hệ cụ thể giữa các vector này và các chủ đề trong dữ liệu.
- **So sánh với các phương pháp giảm chiều khác:** Việc so sánh hiệu suất của SVD với các phương pháp giảm chiều khác, như PCA hoặc LDA, trên cùng một tập dữ liệu sẽ giúp đánh giá hiệu quả tương đối của SVD.
- **Ứng dụng trên các tập dữ liệu văn bản khác:** Cần thử nghiệm việc áp dụng SVD trên các tập dữ liệu văn bản khác, không chỉ giới hạn ở tập dữ liệu được sử dụng trong nghiên cứu này, để đánh giá tính phổ dụng của kết quả.

Chapter 5. Tài liệu tham khảo

5. Tài liệu tham khảo

- Singular Value Decomposition
- Support Vector Machine
- Method of Lagrange Multipliers: The Theory Behind Support Vector Machines
- [Handbook] Singular Values Decomposition và một số ứng dụng
- Singular Value Decomposition (Kaggle)

Thank you