

# Báo Cáo Đồ Án

## Khai Khoáng Dữ Liệu

### Đề tài 3: Xây dựng ứng dụng phân loại văn bản của 10 lĩnh vực

#### A. Train Model

##### 1. Tổng quan

Cho 10 lĩnh vực của các bài báo, hãy viết 1 web service để phân loại văn bản đầu vào. Tạo thêm một website để tương tác với web service.




##### 2. Chuẩn bị dữ liệu

Link github: [https://github.com/ThanhB1805916/DataMining\\_CT312](https://github.com/ThanhB1805916/DataMining_CT312)

Các file này nằm trong thư mục: project/google\_colab/

Chạy các file này trên google colab và có yêu cầu quyền truy cập drive.

Dữ liệu có sẵn là các file pickle dưới đây nằm trong thư mục Src:

| Name   | Date modified      | Type     | Size     |
|--|--------------------|----------|----------|
|  news_df.pkl    | 11/05/2022 8:18 CH | PKL File | 2.711 KB |
|  stop_words.pkl | 11/05/2022 8:18 CH | PKL File | 41 KB    |
|  topics.pkl     | 11/05/2022 8:18 CH | PKL File | 1 KB     |

1. news\_df.pkl: là data frame chứa các bài báo vào topics

df

|     | News  | Topics   |
|-----|---|----------|
| 0   | Để trẻ em phát triển không bị lệch lạc về giới... | Doi song |
| 1   | Sò huyết Ô Loan \nĐầm Ô Loan thuộc huyện Tuy A... | Doi song |
| 2   | Anh Phan đã làm uống công độc giả\nEm nghĩ rằ...  | Doi song |
| 3   | Tôi có nên tiếp tục tình yêu với anh?\nAnh muố... | Doi song |
| 4   | Bình giữ nhiệt nóng, lạnh\nVới hai tính năng v... | Doi song |
| ... | ...   | ...      |
| 95  | Thu Hà: "Tôi không quan tâm đến những lời đồn...  | Van hoa  |
| 96  | Wet Wet Wet tái hợp\nSau 7 năm mỗi người mỗi n... | Van hoa  |
| 97  | Khánh Linh: 'Tôi giống như ngọn lửa nhỏ cháy m... | Van hoa  |
| 98  | Tiếng hát bên sông Hàn - Bản tình ca về gia đ...  | Van hoa  |
| 99  | 'Công nghệ lặn xê' tiến quân về Biên Hòa\nSau...  | Van hoa  |

1000 rows x 2 columns

2. topics.pkl: là list 10 topics

```
import pickle

with open(os.path.join(path, "Src/topics.pkl"), "rb") as rb:
    topics = pickle.load(rb)
topics
```

```
['Doi song',
 'Chinh tri Xa hoi',
 'Khoa hoc',
 'Suc khoe',
 'Phap luat',
 'Vi tinh',
 'Kinh doanh',
 'The thao',
 'The gioi',
 'Van hoa']
```

3. stop\_words.pkl là danh sách các từ dừng

```
[24] import pickle
      with open(os.path.join(path, "Src/stop_words.pkl"), 'rb') as rb:
          vn_sw = pickle.load(rb)
```

```
vn_sw
'cũng vậy',
'cũng vậy thôi',
'cũng được',
'cơ',
'cơ chỉ',
'cơ chừng',
'cơ cùng',
'cơ dẫn',
'cơ hồ',
'cơ hội',
'cơ mà',
'cơ',
```

### 3. Làm sạch dữ liệu

1. Chuyển các topics sang số để chạy các thuật toán máy học. Số là vị trí của tiêu đề trong topics

```
[10] # chuyển topics sang số
      df["Topics"] = df["Topics"].apply(lambda x: topics.index(x))
```

```
[11] df
```

|   | News   | Topics |
|---|--|--------|
| 0 | Đề trẻ em phát triển không bị lệch lạc về giới...  | 0      |
| 1 | Sò huyết Ô Loan \nĐầm Ô Loan thuộc huyện Tuy A...  | 0      |
| 2 | Anh Phan đã làm uống công độc giả\nEm nghĩ rằng... | 0      |
| 3 | Tôi có nên tiếp tục tình yêu với anh?\nAnh muố...  | 0      |

2. Tạo hàm dọn dữ liệu đầu vào:

**Xóa các ký tự đặc biệt**

✓  
0s

```
[13] import re
import string

# xóa các ký tự đặc biệt chỉ chừa lại chữ
def clean_text(text):

    # viết thường
    text = text.lower()

    # xóa dấu ngoặc
    text = re.sub('[\(\[\].*?[\]\)]', '', text)

    # dấu nháy
    text = re.sub("['\"“”...]", '', text)

    # bỏ dấu câu ( , . : ; )
    text = re.sub('[%s]' % re.escape(string.punctuation), '', text)

    # xóa từ có số
    text = re.sub('\w*\d\w*', '', text)

    # xóa dấu new line
    text = re.sub('\n', ' ', text)

    return text
```

### 3. Dọn dữ liệu trong News:

```
✓ [14] clean_df = pd.DataFrame(df.News.apply(lambda x: clean_text(x)))
0s

✓ [15] clean_df["Topics"] = df["Topics"]
0s

✓ [16] clean_df.head(10)
0s
```

|   | News  | Topics |
|---|---|--------|
| 0 | để trẻ em phát triển không bị lệch lạc về giới... | 0      |
| 1 | sò huyết ô loan đầm ô loan thuộc huyện tuy an...  | 0      |
| 2 | anh phan đã làm uổng công độc giả em nghĩ rằng... | 0      |
| 3 | tôi có nên tiếp tục tình yêu với anh anh muốn ... | 0      |
| 4 | bình giữ nhiệt nóng lạnh với hai tính năng vừa... | 0      |
| 5 | miến xào chay với các nguyên liệu rau mát khôn... | 0      |

#### 4. Tách từ

Sửa dụng thư viện pyvi để tìm các từ tiếng việt gom các từ ghép thành dạng trẻ em -> trẻ\_em để tạo 1 từ gọi là token, vì tiếng anh các từ cách nhau bởi khoảng trắng.

a. Đầu tiên sẽ token hóa các News:

```
[20] from pyvi import ViTokenizer # thư viện NLP tiếng Việt

vi_df = pd.DataFrame(clean_df.News.apply(lambda w: ViTokenizer.tokenize(w)))
vi_df["Topics"] = clean_df["Topics"]
vi_df
```

|     | News  | Topics |
|-----|---|--------|
| 0   | dễ trẻ_em phát_triển không bị lệch_lạc về giới... | 0      |
| 1   | sò_huyết ô loan đâm ô loan thuộc huyện tuy an ... | 0      |
| 2   | anh phan đã làm uổng công độc_giả em nghĩ rằng... | 0      |
| 3   | tôi có nên tiếp_tục tình_yêu với anh anh muốn ... | 0      |
| 4   | bình giữ nhiệt nóng lạnh với hai tính_năng vừa... | 0      |
| ... | ...   | ...    |
| 95  | thu hà tôi không quan_tâm đến những lời đồn_th... | 9      |
| 96  | wet wet wet tái_hợp sau năm mỗi người mỗi ngà ... | 9      |

b. Sau đó sẽ token các stop\_words tiếng việt và thêm stop\_words của tiếng anh:

```
# gom stop word sang từ việt
vn_stop_words = []

for w in vn_sw:
    vn_stop_words.append(ViTokenizer.tokenize(w))

[26] # chắc có tiếng anh nên gom thêm stopwords
from sklearn.feature_extraction import text




stop_words = text.ENGLISH_STOP_WORDS.union(vn_stop_words)
```

c. Lưu files

Các file sẽ được lưu lại vào thư mục Clean để chuẩn bị cho bước tiếp theo:

```
✓ [28] with open(os.path.join(path, "Clean/stop_words.pkl"), 'wb') as wb:
      pickle.dump(stop_words, wb)
```

Lưu thêm các file của các bước Src

| Name   | Date modified      | Type     | Size     |
|--|--------------------|----------|----------|
|  clean_df.pkl   | 11/05/2022 8:18 CH | PKL File | 2.577 KB |
|  stop_words.pkl | 11/05/2022 8:18 CH | PKL File | 46 KB    |
|  vi_df.pkl      | 11/05/2022 8:18 CH | PKL File | 2.559 KB |

#### 4. Phân tích dữ liệu

Load các dữ liệu bên clean

##### 1. Dùng word cloud để phân tích

```
[ ] from wordcloud import WordCloud
    import matplotlib.pyplot as plt

    def showWordCloud(news, stop_words):
        wc = WordCloud(stopwords=stop_words, background_color="white", colormap="Dark2",
                        max_font_size=150, random_state=42)

        plt.rcParams['figure.figsize'] = [16, 6]

        topics = news.keys()
        for index, key in enumerate(topics):
            wc.generate(news[key])

            plt.subplot(3, 4, index+1)
            plt.imshow(wc, interpolation="bilinear")
            plt.axis("off")
            plt.title(key)

        plt.show()
```

##### 2. Top các từ xuất hiện nhiều trong các tiêu đề

```
[ ] showWordCloud(corpus_dict, stop_words)
```



```
Doi song
chồng, gia đình, hai, gái, phụ nữ, vợ, sống, hạnh phúc, cuộc sống, món, tình yêu, đũa, đàn ông, trai
---
Chinh tri Xa hoi
dân, tp, du lịch, công ty, xây dựng, hai, tổ chức, tỉnh, khu vực, trường, khu, tphcm, xe, nguyên
---
Khoa hoc
nghiên cứu, khoa học, đại học, khả năng, sống, phát hiện, người ta, loài, biển, con người, hai, đá, giúp, chó
---
Suc khoe
bệnh, thuốc, trẻ, bác sĩ, điều trị, máu, uống, bệnh viện, bệnh nhân, đau, trường hợp, viêm, bé, giúp
---
Phap luat
vụ, nguyên, tiền, công an, bị cáo, án, điều tra, cơ quan, oanh, đối tượng, thu, hai, kim, công ty
---
Vi tinh
máy, dịch vụ, máy tính, công nghệ, phần mềm, thông tin, cung cấp, hãng, internet, sản phẩm, công ty, tính năng, thiết bị, virus
---
Kinh doanh
doanh nghiệp, thị trường, usd, đầu tư, công ty, xe, thương mại, mỹ, vn, tiền, kinh tế, thuế, ngân hàng, sản xuất
---
The thao
đội, trận, hlv, bóng, hai, cầu thủ, bàn, phút, trận đấu, giải, sân, vn, tiền đạo, giành
---
The gioi
mỹ, tổng thống, iraq, bầu cử, vụ, bush, thủ tướng, tuyên bố, hai, chính phủ, đảng, tổ chức, hôm qua, quốc
---
Van hoa
phim, nhạc, ca sĩ, khán giả, hát, diễn, hai, vai, đạo diễn, âm nhạc, album, diễn viên, nghề, đẹp
---
```

### 3. Tìm từ xuất hiện nhiều giữa các topics

Vì các từ này đều có trong các topics nên ta sẽ bỏ nó vào stop\_words



```

▶ # Thêm các từ trùng với nhau trong các topic vào stop word
from collections import Counter

words = []
for topic in r_data_dtm.columns:
    top = [word for (word, count) in top_dict[topic]]
    for t in top:
        words.append(t)

[ ] # Đếm số lần xuất hiện
Counter(words).most_common()

[('hai', 8),
 ('hàng', 5),
 ('tiền', 4),
 ('đường', 4),
 ('giúp', 4),
 ('công_ty', 4),
 ('nam', 4),
 ('mỹ', 4),
 ('máy', 3),
 ('phát_triển', 3),
 ('vn', 3),
 ('triệu', 3),
 ('quốc', 3),
 ('phụ_nữ', 2),
 ('sống', 2),

```

#### 4. Thêm các từ xuất hiện trên 5 topics vào stop\_words vào lưu vào Analyze

```

▶ # Nếu xuất hiện hơn 1 nữa topic thì vào stop words
common_stop_words = [word for word, count in Counter(words).most_common() if count > 5]

[ ] common_stop_words

['hai']

[ ] stop_words_v2 = stop_words.union(common_stop_words)

[ ] with open(os.path.join(path, "Analyze/stop_words.pkl"), 'wb') as wb:
    pickle.dump(stop_words_v2, wb)

```

#### 5. Train model

1. Load các dữ liệu topics và vi\_df bên Clean và stop\_words vừa phân tích

```

import os

path = "/content/drive/MyDrive/Colab Notebooks/google_colab"

[ ] import pickle

with open(os.path.join(path, "Clean/topics.pkl"), 'rb') as rb:
    topics = pickle.load(rb)
topics

['Doi song',
 'Chinh tri Xa hoi',
 'Khoa hoc',
 'Suc khoe',
 'Phap luat',
 'Vi tinh',
 'Kinh doanh',
 'The thao',
 'The gioi',
 'Van hoa']

[ ] import pandas as pd
df = pd.read_pickle(os.path.join(path, "Clean/vi_df.pkl"))
df

```

```

[ ] with open(os.path.join(path, "Analyze/stop_words.pkl"), 'rb') as rb:
    stop_words = pickle.load(rb)

```

## 2. Chia test và train

```

[ ] from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(df.News, df.Topics, stratify=df.Topics)

```

### 3. Train model theo 3 thuật toán K-neighbors, Naive Bayes và SVM

```
from sklearn.neighbors import KNeighborsClassifier

def knn(X):
    model = KNeighborsClassifier(n_neighbors=15)
    model.fit(X.toarray(), y_train)

    with open(os.path.join(path, "Model/knn.pkl"), "wb") as file:
        pickle.dump(model, file)

    return model
```

```
from sklearn.naive_bayes import GaussianNB

def bayes_model(X):
    model = GaussianNB()
    model.fit(X.toarray(), y_train)

    with open(os.path.join(path, "Model/bayes.pkl"), "wb") as file:
        pickle.dump(model, file)

    return model
```

```
[ ] from sklearn.svm import SVC

def svc_model(X):
    model = SVC() # mặc định ok
    model.fit(X.toarray(), y_train)

    with open(os.path.join(path, "Model/svm.pkl"), "wb") as file:
        pickle.dump(model, file)

    return model
```

4. Chuyển các news sang số sử dụng tf-idf và train các model

```
[ ] from sklearn.feature_extraction.text import TfidfVectorizer
tv = TfidfVectorizer(stop_words = stop_words)
X = tv.fit_transform(X_train)
```

```
[ ] with open(os.path.join(path, "Model/tv.pkl"), "wb") as file:
    pickle.dump(tv, file)
```

```
[ ] tv_bayes = bayes_model(X)
```

```
[ ] tv_knn = knn(X)
```

```
[ ] tv_scv = svc_model(X)
```

5. Tạo hàm vẽ confusion matrix và tính F1

```
[ ] from sklearn.metrics import confusion_matrix, classification_report
    from matplotlib import pyplot as plt
    import seaborn as sn

    def print_report(model):
        print(model)
        y_predicted = model.predict(tv.transform(X_test).toarray())

        cm = confusion_matrix(y_test, y_predicted)
        print(cm)

        sn.heatmap(cm, annot=True, fmt="d")
        plt.xlabel("Predicted")
        plt.ylabel("Truth")
        plt.show()

    print(classification_report(y_test, y_predicted))
```

a. K-neighbors

|              | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0            | 0.85      | 0.92   | 0.88     | 25      |
| 1            | 0.82      | 0.72   | 0.77     | 25      |
| 2            | 0.67      | 0.40   | 0.50     | 25      |
| 3            | 0.71      | 0.88   | 0.79     | 25      |
| 4            | 0.89      | 0.96   | 0.92     | 25      |
| 5            | 0.81      | 0.88   | 0.85     | 25      |
| 6            | 0.85      | 0.92   | 0.88     | 25      |
| 7            | 1.00      | 1.00   | 1.00     | 25      |
| 8            | 0.83      | 0.76   | 0.79     | 25      |
| 9            | 0.81      | 0.84   | 0.82     | 25      |
| accuracy     |           |        | 0.83     | 250     |
| macro avg    | 0.82      | 0.83   | 0.82     | 250     |
| weighted avg | 0.82      | 0.83   | 0.82     | 250     |

b. Naive Bayes

|              | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0            | 0.65      | 0.52   | 0.58     | 25      |
| 1            | 0.59      | 0.68   | 0.63     | 25      |
| 2            | 0.81      | 0.52   | 0.63     | 25      |
| 3            | 0.65      | 0.60   | 0.63     | 25      |
| 4            | 0.71      | 0.60   | 0.65     | 25      |
| 5            | 0.84      | 0.84   | 0.84     | 25      |
| 6            | 0.75      | 0.72   | 0.73     | 25      |
| 7            | 0.93      | 1.00   | 0.96     | 25      |
| 8            | 0.72      | 0.84   | 0.78     | 25      |
| 9            | 0.61      | 0.88   | 0.72     | 25      |
| accuracy     |           |        | 0.72     | 250     |
| macro avg    | 0.73      | 0.72   | 0.72     | 250     |
| weighted avg | 0.73      | 0.72   | 0.72     | 250     |

c. SVM

|              | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0            | 0.85      | 0.88   | 0.86     | 25      |
| 1            | 0.79      | 0.76   | 0.78     | 25      |
| 2            | 0.66      | 0.84   | 0.74     | 25      |
| 3            | 0.86      | 0.76   | 0.81     | 25      |
| 4            | 1.00      | 0.84   | 0.91     | 25      |
| 5            | 0.92      | 0.92   | 0.92     | 25      |
| 6            | 0.85      | 0.88   | 0.86     | 25      |
| 7            | 1.00      | 1.00   | 1.00     | 25      |
| 8            | 0.88      | 0.88   | 0.88     | 25      |
| 9            | 0.92      | 0.88   | 0.90     | 25      |
| accuracy     |           |        | 0.86     | 250     |
| macro avg    | 0.87      | 0.86   | 0.87     | 250     |
| weighted avg | 0.87      | 0.86   | 0.87     | 250     |

6. Bảng so sánh chỉ số F1

| Model | Naive Bayes | K-neighbors | SVM  |
|-------|-------------|-------------|------|
| F1    | 0.72        | 0.83        | 0.86 |

Vì SVM có F1 cao nhất nên sẽ sử dụng model này để deploy

## B. Deploy Model

### 1. Sử dụng docker file

Dockerfile > FROM

```
1 FROM python
2 WORKDIR /work
3 RUN pip3 install flask
4 RUN pip3 install pyvi
5 COPY . .
6
7 ENTRYPOINT [ "python3", "src/app.py" ]
```

Dùng flask để tạo web service và pyvi để token hóa các dữ liệu đầu vào

## 2. Endpoints để dự đoán

```
@app.route("/svm", methods=["POST"])
def svm_predict():
    ... try:
    ...     news = request.json["news"][:max_length]

    ...     start = time.time_ns()

    ...     result = pre.svm_predict(news)

    ...     print(result)

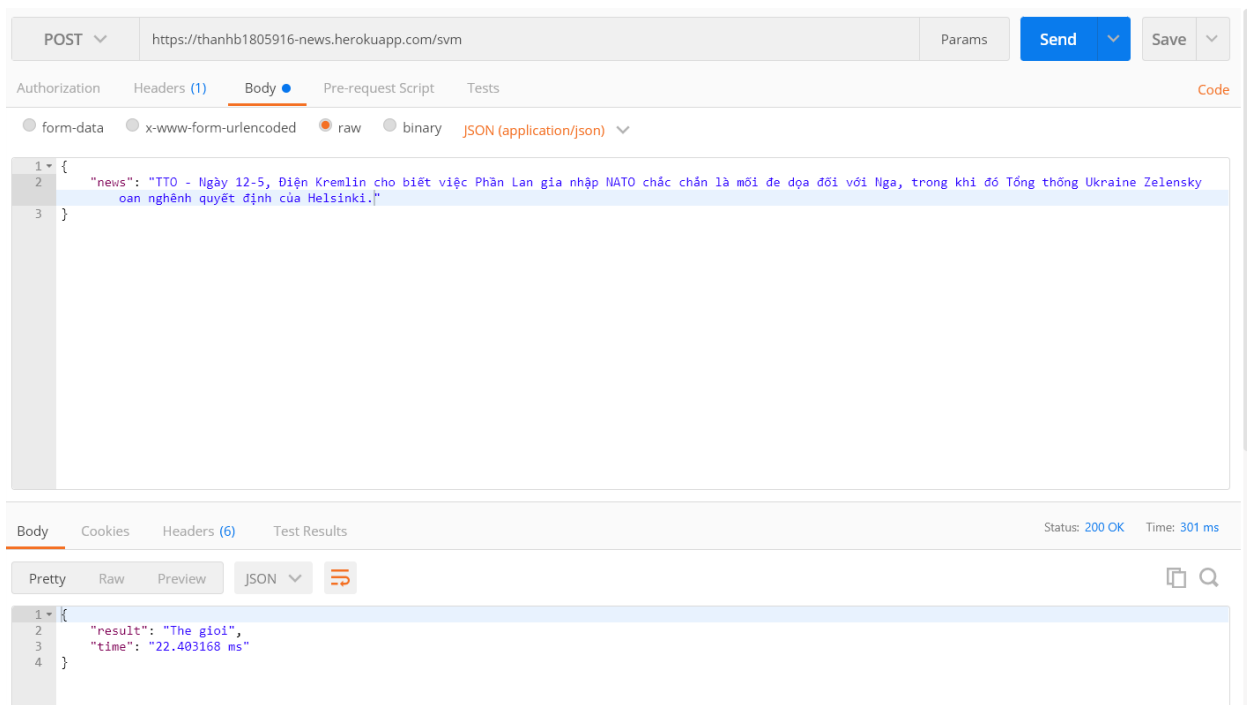
    ...     end = time.time_ns()

    ...     # to ms
    ...     exec_time = f"{str((end-start)/pow(10, 6))} ms"

    ...     return {
    ...         "time": str(exec_time),
    ...         "result": result
    ...     }
    ... except Exception as e:
    ...     print(e)

    ...     return "Error happen", 500
```

Thử api đã deploy lên heroku bằng post man





## Giao diện cho người dùng



### Tải file lên

Choose File No file chosen

TTO - Ngày 12-5, Điện Kremlin cho biết việc Phần Lan gia nhập NATO chắc chắn là một đe dọa đối với Nga, trong khi đó Tổng thống Ukraine Zelensky hoan nghênh quyết định của Helsinki.  
ĐỌC NHANH 8-5: Đại sứ Nga cảnh báo Thụy Điển, Phần Lan việc gia nhập NATO

Dự đoán

## Kết Quả

The gioi