

TRƯỜNG ĐẠI HỌC CẦN THƠ
KHOA CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG
BỘ MÔN CÔNG NGHỆ PHẦN MỀM



KHAI KHOÁNG DỮ LIỆU

Đề tài

ỨNG DỤNG PHÂN LOẠI BÀI BÁO

Cán bộ hướng dẫn:

TS. Lưu Tiến Đạo

Sinh viên thực hiện:

Vương Cẩm Thanh B1805916

Cần Thơ, 05/2022

MỤC LỤC

Nội dung

A.	Phân lớp văn bản.....	3
1.	Tổng quan	3
2.	Ứng dụng	3
3.	Chuẩn bị dữ liệu.....	3
B.	Tiền xử lý dữ liệu	5
1.	Làm sạch dữ liệu	6
2.	Tách từ.....	7
3.	Phân tích dữ liệu	8
C.	Vector hóa văn bản	11
1.	Tổng quan	11
2.	Vector hóa văn bản với stop words sau khi phân tích:	11
D.	Train model	12
1.	Train model theo 3 thuật toán K-nearest neighbors, Naive Bayes và SVM	12
2.	Tạo hàm vẽ confusion matrix và tính F1	13
E.	Deploy Model	15
1.	Sử dụng Dockerfile	15
2.	Endpoints để dự đoán	16
3.	Thử api đã deploy lên heroku bằng Postman.....	17
4.	Giao diện cho người dùng	17

A. Phân lớp văn bản

1. Tổng quan

Phân lớp văn bản được coi là quá trình phân loại một văn bản bất kì vào một hay nhiều lớp cho trước. Quá trình này gồm hai bước. Ở bước thứ nhất, một mô hình phân lớp (classification model) được xây dựng dựa trên tri thức kinh nghiệm. Ở đây, tri thức kinh nghiệm chính là một tập dữ liệu huấn luyện (training dataset) được cung cấp bởi con người bao gồm một tập văn bản và phân lớp tương ứng của chúng. Bước này còn gọi là bước xây dựng huấn luyện (training process) hay ước lượng mô hình phân lớp. Ở bước thứ hai, mô hình phân lớp xây dựng ở bước đầu sẽ được sử dụng để phân lớp cho những văn bản (chưa được phân loại) trong tương lai. Bước đầu tiên được xem như là việc học có giám sát mà chúng ta có thể sử dụng rất nhiều các kỹ thuật học máy đã có như: Naïve Bayes, k láng giềng gần nhất (kNN), cây quyết định (Decision Tree), Support Vector Machine (SVM). Mục tiêu của bài toán phân lớp là nhằm xây dựng mô hình có khả năng gán nhãn cho một bài báo bất kì với độ chính xác cao nhất có thể.

2. Ứng dụng

Ứng dụng lớn nhất của bài toán phân lớp văn bản là áp dụng vào bài toán phân loại hay lọc nội dung. Trong bài toán lọc nội dung: một văn bản được phân loại vào nhóm: có ích hoặc không có ích, các ứng dụng cụ thể như: lọc email rác, Trong đề tài này sẽ là phân loại bài báo theo 10 chủ đề khác nhau. Với tất cả ý nghĩa thực tế trên, một lần nữa có thể khẳng định rằng trong thời đại Internet được coi là một phần không thể thiếu trong cuộc sống, phân lớp văn bản luôn là vấn đề đáng được quan tâm để có thể phát triển và xây dựng được những công cụ ngày càng hữu dụng hơn. Dựa trên nhu cầu cấp thiết đó, em chọn đề tài "Phân loại bài báo" để có thể nghiên cứu và phát triển ứng dụng này

3. Chuẩn bị dữ liệu




Dữ liệu là yếu tố quan trọng nhất và cũng là vấn đề mà chúng ta cần quan tâm nhất. Trong quá trình xây dựng một hệ thống phân lớp văn bản, bước chuẩn bị và tiền xử lý dữ liệu quyết định tới thành bại của hệ thống hơn cả. Tất cả dữ liệu đã có sẵn tại đây:

https://github.com/ThanhB1805916/DataMining_CT312

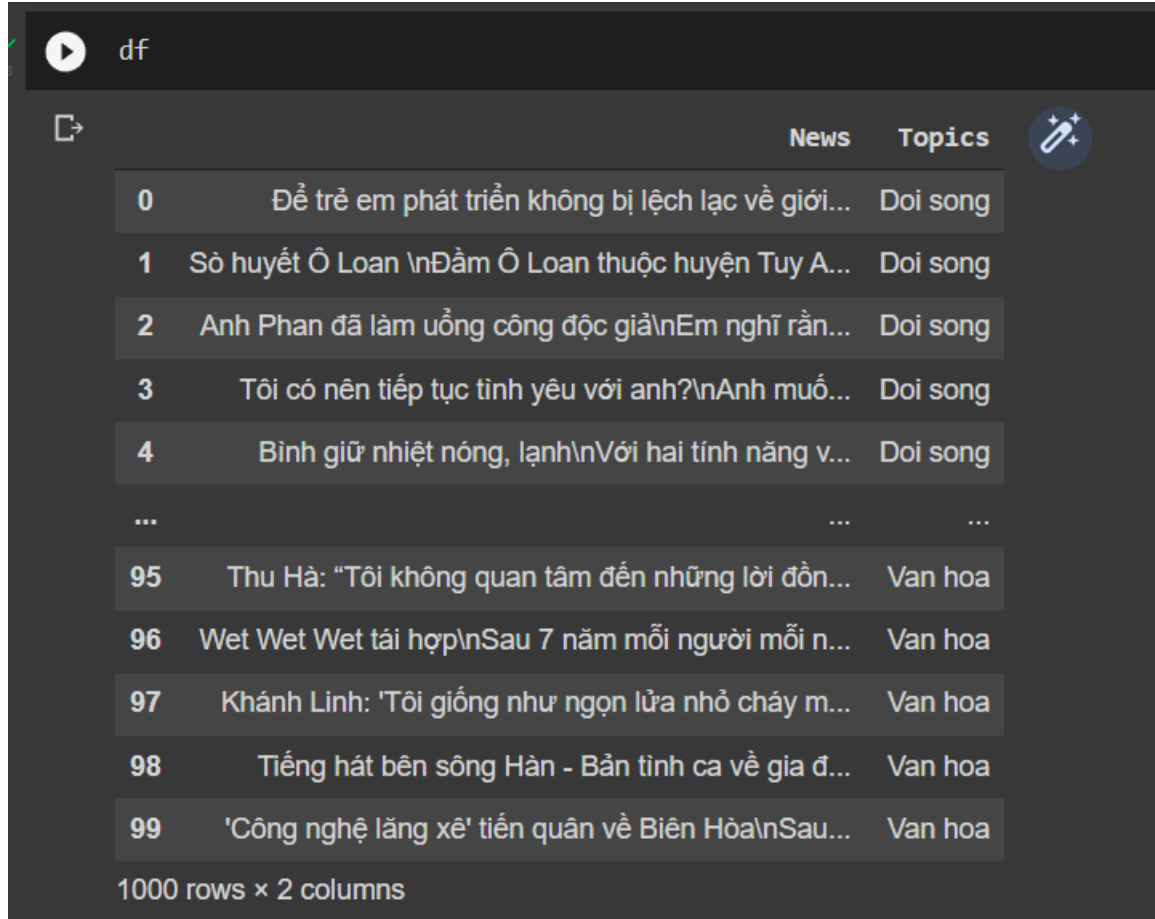
Các file này nằm trong thư mục: project/google_colab/

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 rằn...	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 × 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)
```

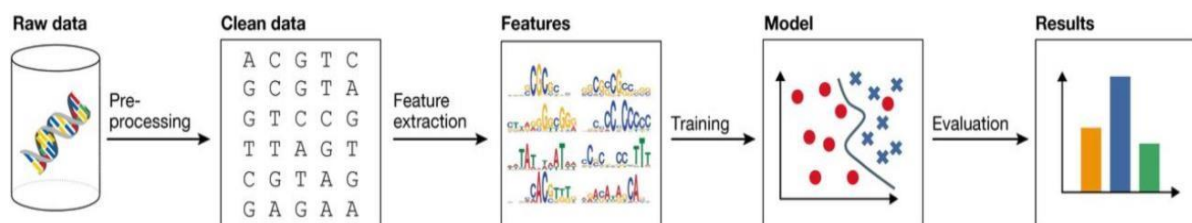
vn_sw

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

B. Tiền xử lý dữ liệu

Xử lý ngôn ngữ tự nhiên (natural language processing - NLP) là một nhánh của trí tuệ nhân tạo tập trung vào các ứng dụng trên ngôn ngữ của con người. Trong trí tuệ nhân tạo thì xử lý ngôn ngữ tự nhiên là một trong những phần khó nhất vì nó liên quan đến việc phải hiểu ý nghĩa ngôn ngữ-công cụ hoàn hảo nhất của tư duy và giao tiếp¹.

Bước đầu tiên trong xử lý ngôn ngữ tự nhiên là tiền xử lý dữ liệu. Vì văn bản trong ngôn ngữ tự nhiên không có cấu trúc, làm cho chương trình không thể hiểu được đâu là dữ liệu có giá trị. Quy trình tiền xử lý dữ liệu để có dữ liệu phù hợp được tổng kết như hình sau:



Các bước tiền xử lý được sử dụng cho máy học thường gồm các bước: làm sạch dữ liệu, tách từ, chuẩn hóa từ, loại bỏ stopword... và được thể hiện như hình sau:



1. Làm sạch dữ liệu

Các chuỗi văn bản sẽ được chuyển sang viết thường, bỏ các ký tự đặc biệt, các dấu câu và số.

Xóa các ký tự đặc biệt

```
[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 cổ số
    text = re.sub('\w*\d\w*', '', text)

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

    return text
```

Dữ liệu News sau khi được làm sạch:

```
✓ [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 uống 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	miễn xào chay với các nguyên liệu rau mát khôn...	0

2. Tách từ

Đơn vị từ trong tiếng Việt bao gồm từ đơn và từ ghép. Nên chúng ta cần phải nói cho mô hình học máy biết đâu là từ đơn, đâu là từ ghép. Nếu không thì từ nào cũng sẽ là từ đơn hết.

Bởi vì mô hình của chúng ta sẽ coi các từ là đặc trưng, tách nhau theo dấu cách. Do đó, chúng ta phải nối các từ ghép lại thành một từ để không bị tách sai.

Sử 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	đề 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)
```

3. 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)
```



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 nửa 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)
```

C. Vector hóa văn bản

1. Tổng quan

Thông thường, máy tính không thể hiểu được ý nghĩa các từ. Như vậy, để xử lý được ngôn ngữ tự nhiên, ta cần có một phương pháp để biểu diễn văn bản dưới dạng mà máy tính có thể hiểu được. Phương pháp tiêu chuẩn để biểu diễn văn bản đó là biểu diễn các văn bản theo vector. Trong đó, các từ/cụm từ thuộc kho tài liệu ngôn ngữ được ánh xạ thành những vector trên hệ không gian số thực. Ở đây em sẽ sử dụng thuật toán TF-IDF

TF- IDF (term frequency–inverse document frequency) – tần suất- tần suất đảo nghịch từ. Đây là một phương pháp thống kê, nhằm phản ánh độ quan trọng của mỗi từ hoặc n-gram đối với văn bản trên toàn bộ tài liệu đầu vào. TF-IDF thể hiện trọng số của mỗi từ theo ngữ cảnh văn bản. TF-IDF sẽ có giá trị tăng tỷ lệ thuận với số lần xuất hiện của từ trong văn bản và số văn bản có chứa từ đó trên toàn bộ tập tài liệu. Phương pháp này giúp cho TF-IDF có tính phân loại cao hơn.

$$tf_i = \frac{n_i}{N_i}$$

Trong đó:

i : 1 .. D

n_i : tần số xuất hiện của từ trong văn bản i .

N_i : tổng số từ trong văn bản i .

$$idf_i = \log_2 \frac{D}{d}$$

Trong đó:

D : tổng số documents trong tập dữ liệu.

d : số lượng documents có sự xuất hiện của từ.

$$tfidf_i = tf_i \times idf_i$$

2. Vector hóa văn bản với stop words sau khi phân tích:

```
[ ] 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)
```

D. Train model

1. Train model theo 3 thuật toán K-nearest 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
```

2. 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-nearest 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

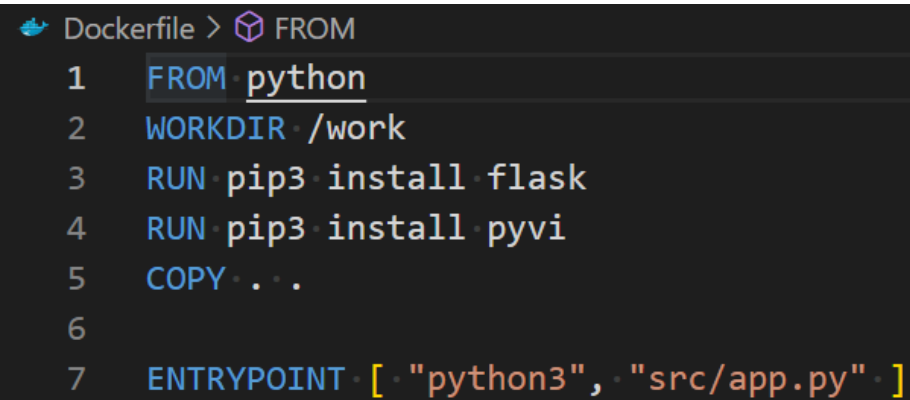
2. Bảng so sánh chỉ số F1

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

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

E. Deploy Model

1. Sử dụng Dockerfile



```
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)

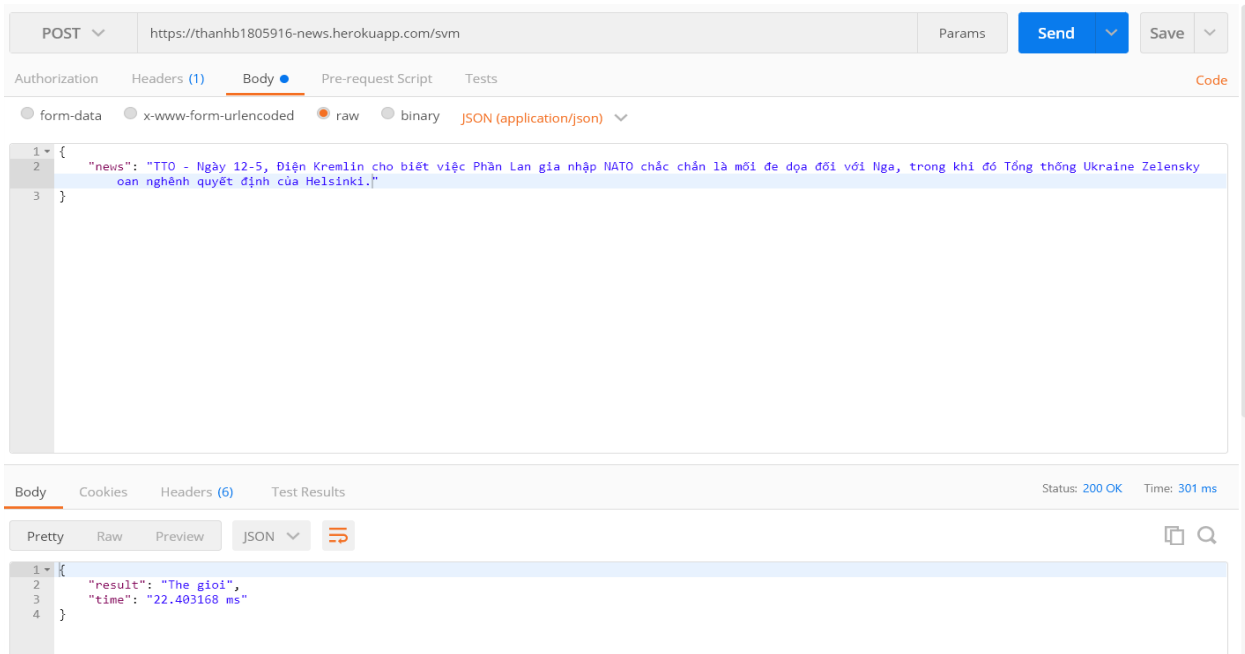
    ....     end = time.time_ns()

    ....     # 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
```


3. Thử api đã deploy lên heroku bằng Postman



4. Giao diện cho người dùng

