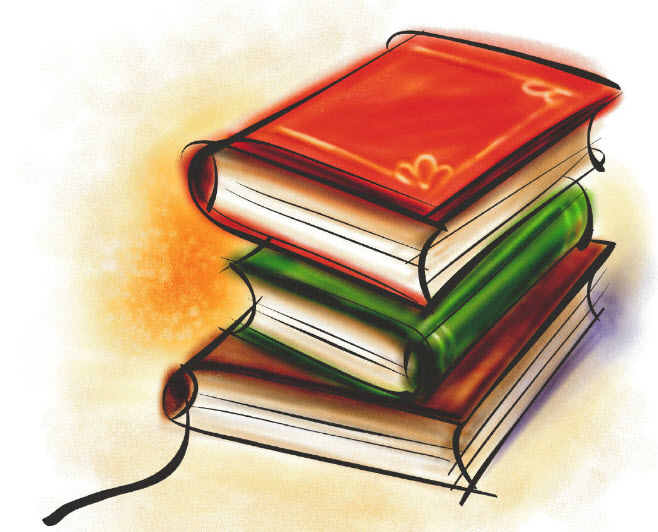
TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP TP HCM

🕮--⸙--🕮



**Bài báo cáo**

*Đề tài*: Phân loại chủ đề các bài báo



**Thực hiện:**

Đặng Lê Huy

Trần Nhất Thống

Lê Sỹ Hồ

**Giáo viên hướng dẫn:** Đặng Thị Phúc

Mục lục

[**I. Tổng quan về đề tài 3**](#_Toc57876567)

[**II. Chuẩn bị dữ liệu 3**](#_Toc57876568)

[**III. Tiền xử lý dữ liệu 3**](#_Toc57876569)

[1.Chuẩn hóa kiểu gõ dấu 4](#_Toc57876570)

[2. Tách từ 6](#_Toc57876571)

[3.Đưa về chữ viết thường 7](#_Toc57876572)

[4. Xóa các ký tự không cần thiết 9](#_Toc57876573)

[**IV.**  **Loại bỏ stopword** 10](#_Toc57876574)

[**V. Xây dựng mô hình phân loại văn bản 13**](#_Toc57876579)

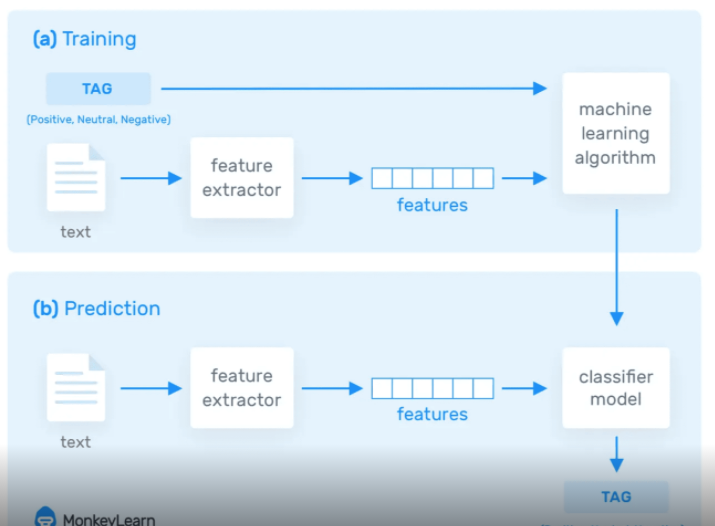
[**VI. Phân loại với logistic regression và navie bayes 13**](#_Toc57876580)

[**VII. Đánh giá mô hình phân loại văn bản 25**](#_Toc57876581)

# Tổng quan về đề tài

Phân loại văn bản (text classification) là bài toán thuộc nhóm học có giám sát (Supervised learning) trong học máy. Bài toán yêu cầu dữ liệu cần có nhãn (label). Mô hình sẽ học từ dữ liệu có nhãn đó, sau đó được dùng để dự đoán nhãn cho các dữ liệu mới chưa gặp

Lấy ví dụ, cần xây dựng một mô hình học máy để dự đoán chủ để (Kinh tế, giáo dục, thể thao…) của một bài báo bất kỳ. Khi đó, bạn cần rất nhiều dữ liệu có gán nhãn, tức là bạn cần rất nhiều bài báo và mỗi bài báo đó chúng ta phải biết trước nó nằm trong chủ đề nào rồi.



Giai đoạn (a): Huấn luyện (trainning) là giai đoạn học tập của mô hình phân loại văn bản. Mô hình sẽ học từ dữ liệu có nhãn. Dữ liệu văn bản sẽ được số hóa thông qua bộ trích xuất đặc trưng để mỗi mẫu dự liệu trong tập huấn trở thành 1 vector nhiều chiều. Thuật toán máy học sẽ học và tối ưu các tham số để đạt được kết quả tốt trên tập dữ liệu này. Nhãn của dữ liệu được dùng để đánh giá việc mô hình học tốt.

Giai đoạn (b): Dự đoán, là giai đoạn sử dụng mô hình học máy sau khi nó đã học xong. Ở giai đoạn này, dữ liệu cần dự đoán cũng vẫn thực hiện các bước trích xuất đặc trưng. Mô hình đã học sau đó nhận vào là đặc trưng đó va đưa ra kết quả dự đoán.

# Chuẩn bị dữ liệu

Với bài toán phân loại văn bản bài báo, dữ liệu nhóm mình chuẩn bị là dữ liệu các bài báo được thu thập trên trang vnexpress kèm theo chủ đề của bài báo đó.

Ở bước này nhóm mình thu thập dữ liệu bằng cách: cào tất cả các link bài báo, sau đó dùng link để cào tất cả nội dung bài báo đính kèm theo chủ đề của nó.

Nhóm mình thu thập được trên 15000 document là nội dung các bài báo với nhiều chủ đề khác nhau.

Dữ liệu thu thập được lưu trữ trên cơ sở dữ liệu mongoDB

from selenium import webdriver  
import xlsxwriter  
from datetime import date  
import json  
driver = webdriver.Chrome('../chromedriver.exe')  
driver.get("https://vnexpress.net/kinh-doanh/doanh-nghiep-p36")  
dataLink = {}  
with open('../file/link\_kinh\_te.txt.txt', 'r') as json\_file:  
 dataLink= json.loads(json\_file.read())  
stt= len(dataLink['link'])  
def addLink(link,stt):  
 a= date.today().strftime("%m/%d/%Y")  
 dataLink['link'].append({  
 "id": ++stt,  
 "topic":"Giao duc",  
 "link": link,  
 "state": False,  
 "date": a  
 })  
 # with open('../file/LINK2.txt', 'a') as outfile:  
 # json.dump(dataLink, outfile)  
 # outfile.write("\n")  
def add\_Link\_file():  
 with open('../file/link\_kinh\_te.txt', 'w') as outfile:  
 json.dump(dataLink, outfile)  
def crawl\_web(stt):  
 contain = driver.find\_elements\_by\_css\_selector("h2.title-news")  
 for p in contain:  
 link=driver.find\_element\_by\_link\_text(p.text).get\_attribute('href')  
 stt+=1  
 addLink(link,stt)  
  
 step = driver.find\_element\_by\_xpath('//a[@class="btn-page next-page "]')  
 step.click()  
 add\_Link\_file()  
 crawl\_web(stt)  
def main():  
 crawl\_web(stt)  
 driver.close()  
main()

*Thực thi cào link các bài báo*

from selenium import webdriver  
import time  
from pymongo import MongoClient  
import json  
client= MongoClient("mongodb+srv://huylu:npf.huy8@cluster0.ua0vg.mongodb.net/<dbname>?retryWrites=true&w=majority")  
db\_content= client.NLP  
driver = webdriver.Chrome('../chromedriver.exe')  
  
def addContent(data):  
 dataContent = {  
 "topic": "Cong nghe",  
 "data": data,  
 "state": False  
 }  
 db\_content.data.insert\_one(dataContent)  
def main():  
 try:  
 with open('../file/link\_kinh\_te.txt', 'r') as json\_file:  
 dataLink= json.loads(json\_file.read())  
 for i in dataLink['link']:  
 # driver.get(i['link'])  
 # data = driver.find\_element\_by\_xpath('//div[@class="container"]')  
 # time.sleep(1)  
 # addContent(data.text)  
 print(i)  
 except:  
 driver.close()  
  
main()

*Thực thi cào dữ liệu nội dung bài báo*

# Tiền xử lý dữ liệu

* Xóa html code nếu có
* Chuẩn hóa kiểu gõ dấu tiếng Việt
* Thực hiện tách từ tiếng việt (sử dụng thư viện của nhóm tác giả underthesea…)
* Đưa văn bản về chữ viết thường
* Xóa các ký tự đặc biệt

## **1.Chuẩn hóa kiểu gõ dấu**

def chuan\_hoa\_dau\_tu\_tieng\_viet(word):  
 if not is\_valid\_vietnam\_word(word):  
 return word  
  
 chars = list(word)  
 dau\_cau = 0  
 nguyen\_am\_index = []  
 qu\_or\_gi = False  
 for index, char in enumerate(chars):  
 x, y = nguyen\_am\_to\_ids.get(char, (-1, -1))  
 if x == -1:  
 continue  
 elif x == 9: # check qu  
 if index != 0 and chars[index - 1] == 'q':  
 chars[index] = 'u'  
 qu\_or\_gi = True  
 elif x == 5: # check gi  
 if index != 0 and chars[index - 1] == 'g':  
 chars[index] = 'i'  
 qu\_or\_gi = True  
 if y != 0:  
 dau\_cau = y  
 chars[index] = bang\_nguyen\_am[x][0]  
 if not qu\_or\_gi or index != 1:  
 nguyen\_am\_index.append(index)  
 if len(nguyen\_am\_index) < 2:  
 if qu\_or\_gi:  
 if len(chars) == 2:  
 x, y = nguyen\_am\_to\_ids.get(chars[1])  
 chars[1] = bang\_nguyen\_am[x][dau\_cau]  
 else:  
 x, y = nguyen\_am\_to\_ids.get(chars[2], (-1, -1))  
 if x != -1:  
 chars[2] = bang\_nguyen\_am[x][dau\_cau]  
 else:  
 chars[1] = bang\_nguyen\_am[5][dau\_cau] if chars[1] == 'i' else bang\_nguyen\_am[9][dau\_cau]  
 return ''.join(chars)  
 return word  
  
 for index in nguyen\_am\_index:  
 x, y = nguyen\_am\_to\_ids[chars[index]]  
 if x == 4 or x == 8: # ê, ơ  
 chars[index] = bang\_nguyen\_am[x][dau\_cau]  
 return ''.join(chars)  
  
 if len(nguyen\_am\_index) == 2:  
 if nguyen\_am\_index[-1] == len(chars) - 1:  
 x, y = nguyen\_am\_to\_ids[chars[nguyen\_am\_index[0]]]  
 chars[nguyen\_am\_index[0]] = bang\_nguyen\_am[x][dau\_cau]  
 else:  
  
 x, y = nguyen\_am\_to\_ids[chars[nguyen\_am\_index[1]]]  
 chars[nguyen\_am\_index[1]] = bang\_nguyen\_am[x][dau\_cau]  
 else:  
 x, y = nguyen\_am\_to\_ids[chars[nguyen\_am\_index[1]]]  
 chars[nguyen\_am\_index[1]] = bang\_nguyen\_am[x][dau\_cau]  
 return ''.join(chars)  
  
  
def is\_valid\_vietnam\_word(word):  
 chars = list(word)  
 nguyen\_am\_index = -1  
 for index, char in enumerate(chars):  
 x, y = nguyen\_am\_to\_ids.get(char, (-1, -1))  
 if x != -1:  
 if nguyen\_am\_index == -1:  
 nguyen\_am\_index = index  
 else:  
 if index - nguyen\_am\_index != 1:  
 return False  
 nguyen\_am\_index = index  
 return True  
  
  
def chuan\_hoa\_dau\_cau\_tieng\_viet(sentence):  
 *"""  
 Chuyển câu tiếng việt về chuẩn gõ dấu kiểu cũ.* ***:param*** *sentence:* ***:return****:  
 """* sentence = sentence.lower()  
 words = sentence.split()  
 for index, word in enumerate(words):  
 cw = re.sub(r'(^\p{P}\*)([p{L}.]\*\p{L}+)(\p{P}\*$)', r'\1/\2/\3', word).split('/')  
 # print(cw)  
 if len(cw) == 3:  
 cw[1] = chuan\_hoa\_dau\_tu\_tieng\_viet(cw[1])  
 words[index] = ''.join(cw)  
 return ' '.join(words)

## **2. Tách từ**

Bài toán tách từ, nhóm sử dụng mã nguồn mở có sẳn của underthesea. Do đó chỉ cần cài đặt và sử dụng.

## 3**. Đưa về chữ viết thường**

Việc đưa dữ liệu về chữ viết thường là rất cần thiết. Bởi vì đặc trưng này không có tác dụng ở bài toán phân loại văn bản. Đưa về chữ viết thường giúp giảm số lượng đặc trưng và tăng độ chính xác hơn

## **4. Xóa các ký tự không cần thiết**

Việc xóa các ký tự không cần thiết giúp:

* Giảm số chiều đặc trưng, tăng tốc độ học và xử lý
* Tránh làm ảnh hưởng xấu tới kết quả mô hình

Các dấu ngắt câu, các số đếm và các ký tự đặc biệt khác không chỉ giúp bạn phân loại một văn bản thuộc chuyên mục nào.

def text\_preprocess(document):  
 # xóa html code  
 document = remove\_html(document)  
 # chuẩn hóa unicode  
 document = convert\_unicode(document)  
 # chuẩn hóa cách gõ dấu tiếng Việt  
 document = chuan\_hoa\_dau\_cau\_tieng\_viet(document)  
 # tách từ  
 document = word\_tokenize(document, format="text")  
 # đưa về lower  
 document = document.lower()  
 # xóa các ký tự không cần thiết  
 document = re.sub(r'[^\s\wáàảãạăắằẳẵặâấầẩẫậéèẻẽẹêếềểễệóòỏõọôốồổỗộơớờởỡợíìỉĩịúùủũụưứừửữựýỳỷỹỵđ\_]', ' ', document)  
 # xóa khoảng trắng thừa  
 document = re.sub(r'\s+', ' ', document).strip()  
 return document

*Thực thi các hàm tiền xử lý*

# Loại bỏ stopword

Danh sách stopword được xây dựng tư bộ dữ liệu, Stopword là các từ xuất hiện nhiều trong tất cả các chuyên mục

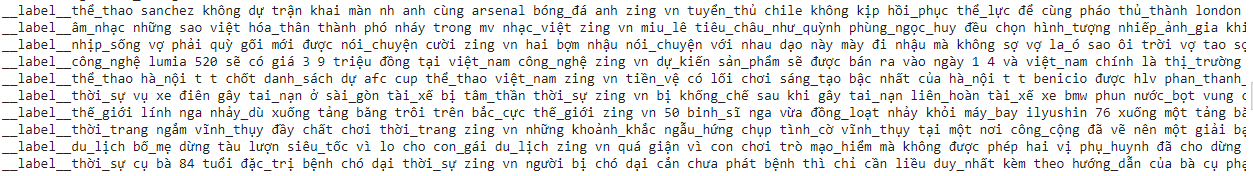
# Thống kê các word xuất hiện ở tất cả các nhãn  
total\_label = 18  
vocab = {}  
label\_vocab = {}  
for line in open('news\_categories.txt'):  
 words = line.split()  
 # lưu ý từ đầu tiên là nhãn  
 label = words[0]  
 if label not in label\_vocab:  
 label\_vocab[label] = {}  
 for word in words[1:]:  
 label\_vocab[label][word] = label\_vocab[label].get(word, 0) + 1  
 if word not in vocab:  
 vocab[word] = set()  
 vocab[word].add(label)  
  
count = {}  
for word in vocab:  
 if len(vocab[word]) == total\_label:  
 count[word] = min([label\_vocab[x][word] for x in label\_vocab])  
  
sorted\_count = sorted(count, key=count.get, reverse=True)  
for word in sorted\_count[:100]:  
 print(word, count[word])

*Thực thi loại bỏ stopword*

# Xây dựng mô hình phân loại văn bản

Xây dựng tập train/test:

Dữ liệu dùng cho bài toán phân loại văn bản của nhóm mình sau khi tiền xử lý được lưu thành 1 file duy nhất. Mỗi dòng là một bài báo kèm thông tin danh mục của nó.



Nhóm mình sử dụng sklearn trong python giúp mình tách dữ liệu làm 2 tập train/test riêng biệt:

# Chia tập train/test  
from sklearn.model\_selection import train\_test\_split  
from sklearn.preprocessing import LabelEncoder  
  
# tỉ lệ tập test - train là 8 : 2  
test\_percent = 0.2  
  
text = []  
label = []  
  
for line in open('news\_categories.prep'):  
 words = line.strip().split()  
 label.append(words[0])  
 text.append(' '.join(words[1:]))  
  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(text, label, test\_size=test\_percent, random\_state=42)  
  
# Lưu train/test data  
# Giữ nguyên train/test để về sau so sánh các mô hình cho công bằng  
with open('train.txt', 'w') as fp:  
 for x, y in zip(X\_train, y\_train):  
 fp.write('{} {}\n'.format(y, x))  
  
with open('test.txt', 'w') as fp:  
 for x, y in zip(X\_test, y\_test):  
 fp.write('{} {}\n'.format(y, x))  
  
# encode label  
label\_encoder = LabelEncoder()  
label\_encoder.fit(y\_train)  
print(list(label\_encoder.classes\_), '\n')  
y\_train = label\_encoder.transform(y\_train)  
y\_test = label\_encoder.transform(y\_test)

*Thực thi chia tập dữ liêu thành train và test*

* Đọc dữ liệu từ file và tách làm 2 list text(dữ liệu) và label(nhãn). Dữ liệu text[i] sẽ có nhãn là label[i]
* Chia làm 2 tập train(X\_train, y\_train) và test(X\_test, y\_test) theo tỉ lệ 80:20
* Lưu train/test data ra file để sử dụng cho việc train với thư viện logistic regression
* Đưa label về dạng vector để tiện cho tính toán sử dụng LabelEncode

# Phân loại văn bản với Logistic regression

from sklearn.linear\_model import LogisticRegression  
  
start\_time = time.time()  
text\_clf = Pipeline([('vect', CountVectorizer(ngram\_range=(1, 1),  
 max\_df=0.8,  
 max\_features=None)),  
 ('tfidf', TfidfTransformer()),  
 ('clf', LogisticRegression(solver='lbfgs',  
 multi\_class='auto',  
 max\_iter=10000))  
 ])  
text\_clf = text\_clf.fit(X\_train, y\_train)  
  
train\_time = time.time() - start\_time  
print('Done training Linear Classifier in', train\_time, 'seconds.')  
  
# Save model  
pickle.dump(text\_clf, open(os.path.join(MODEL\_PATH, "linear\_classifier.pkl"), 'wb')

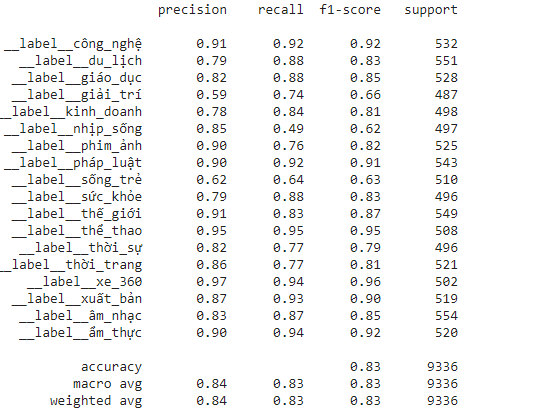
*Logistic regression*

import pickle  
import time  
from sklearn.feature\_extraction.text import CountVectorizer  
from sklearn.feature\_extraction.text import TfidfTransformer  
from sklearn.naive\_bayes import MultinomialNB  
from sklearn.pipeline import Pipeline  
  
start\_time = time.time()  
text\_clf = Pipeline([('vect', CountVectorizer(ngram\_range=(1, 1),  
 max\_df=0.8,  
 max\_features=None)),  
 ('tfidf', TfidfTransformer()),  
 ('clf', MultinomialNB())  
 ])  
text\_clf = text\_clf.fit(X\_train, y\_train)  
  
train\_time = time.time() - start\_time  
print('Done training Naive Bayes in', train\_time, 'seconds.')  
  
# Save model  
pickle.dump(text\_clf, open(os.path.join(MODEL\_PATH, "naive\_bayes.pkl"), 'wb'))

*Phân loại với navie bayes*

# Đánh giá mô hình phân loại văn bản

# Xem kết quả trên từng nhãn  
from sklearn.metrics import classification\_report  
  
nb\_model = pickle.load(open(os.path.join(MODEL\_PATH, "naive\_bayes.pkl"), 'rb'))  
y\_pred = nb\_model.predict(X\_test)  
print(classification\_report(y\_test, y\_pred, target\_names=list(label\_encoder.classes\_)))



# Deploy lên web local

Sử dụng thư viện Flask để render code html và xử lý sự kiện javascript

from flask import Flask, render\_template, request  
  
app = Flask(\_\_name\_\_)  
  
@app.route('/')  
def man():  
 return render\_template('index.html')  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 app.run(debug=True)

