BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC CẦN THƠ
KHOA CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG

□ □ □

ĐỒ ÁN HỌC PHẦN
KHAI KHOÁNG DỮ LIỆU

ĐỀ TÀI

Phân lớp dữ liệu bình luận sản phẩm trên website thegioididong.com

Sinh viên thực hiện:
Dương Trung Hiền - Mã số: B1812267 - Khóa: 44
Lê Duy – Mã số: B1812256 – Khoá: 44

Cần Thơ, 12/2021

BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC CẦN THƠ

KHOA CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG

□ □ □

ĐỒ ÁN HỌC PHẦN
KHAI KHOÁNG DỮ LIỆU

ĐỀ TÀI

Phân lớp dữ liệu bình luận sản phẩm trên website thegioididong.com

Người hướng dẫn
TS. Lưu Tiến Đạo

Sinh viên thực hiện
Dương Trung Hiền - B1812267 - K44
Lê Duy - B1812256 - K44

Cần Thơ, 12/2021

NHẬN XÉT CỦA GIẢNG VIÊN

Cần Thơ, ngày 28 tháng 12 năm 2021

(GVHD ký và ghi rõ họ tên)

LỜI CẢM ƠN

Để có thể hoàn thành đồ án học phần Khai Khoáng Dữ Liệu một cách hoàn chỉnh, chúng em xin được bày tỏ lòng biết ơn chân thành và sâu sắc đến giảng viên TS. Lưu Tiến Đạo – người đã trực tiếp tận tình hướng dẫn, giúp đỡ chúng em. Trong suốt quá trình thực hiện đề tài, nhờ những sự chỉ bảo và hướng dẫn quý giá đó mà đồ án đã được hoàn thành một cách tốt nhất.

Tuy có nhiều cố gắng trong quá trình thực hiện đồ án học phần, nhưng không thể tránh khỏi những sai sót. Chúng em rất mong nhận được sự đóng góp ý kiến quý báu của Thầy và các bạn để đồ án hoàn thiện tốt hơn.

Cần Thơ, ngày 26 tháng 12năm 2021

Người viết

Dương Trung Hiền Lê Duy

MỤC LỤC

DANH MỤC HÌNH

TÓM TẮT

Đồ án đề cập quy trình xây dựng mô hình phân loại văn bản cụ thể là áp dụng vào việc phân loại bình luận của người dùng trên website thegioididong.com. Sử dụng các mô hình chuyên về sử lý văn bản như: Logistic regression, Naives Bayes Multinomial, SVM (Support Vector Machine) và mô hình deep learning PhoBert. Sau đó, đánh giá độ chính xác của các mô hình được áp dụng. Cuối cùng tạo ra website để người dùng có thể dự đoán được bình luận thuộc phân lớp nào dựa trên mô hình đã được huấn luyện sẵn.

PHẦN 1: GIỚI THIỆU

1. Bài toán phân lớp văn bản

1.1. Khái niệm

Bài toán phân lớp văn bản (Text Classification) là bài toán thuộc nhóm học có giám sát (Supervised learning) trong học máy. Bài toán yêu cầu dữ liệu phải có nhãn. Mô hình sẽ học từ tập dữ liệu có nhãn đó, sau đó được dùng để dữ đoán nhãn cho các tập dữ liệu chưa từng gặp qua trong tương lai [1].

1.2. Ứng dụng

Hiện nay bài toán phân lớp văn bản đã được áp dụng vào nhiều lĩnh vực như: phát hiện tin giả về dịch bệnh Covid, phân loại thư rác, phân loại bài báo khoa học,.. Chính vì điều đó, ta có thể áp dụng bài toán phân lớp văn bản để phân lớp bình luận của người dùng trên website thegiodidong.com để biết bình luận đó là tiêu cực hay tích cực.

2. Bố cục đồ án

Bố cục của đồ án khai khoáng dữ liệu gồm 3 phần chính như sau:

Phần giới thiệu

Giới thiệu tổng quát về đề tài.

Phần nội dung

Chương 1: Giới thiệu về bài toán

Chương 2: Nội dung bài toán và qui trình giải quyết bài toán

Chương 3: Đánh giá và kiểm thử

Phần kết luận

Trình bày kết quả đạt được và hướng phát triển hệ thống

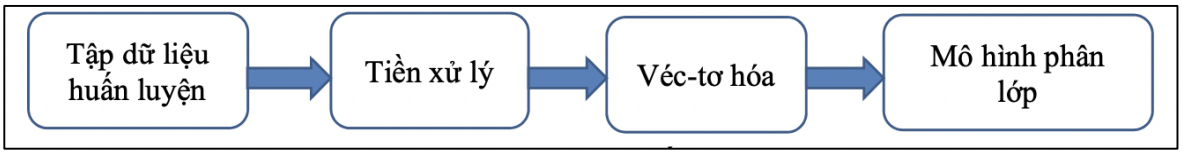
PHẦN 2: NỘI DUNG

CHƯƠNG 1: MÔ TẢ BÀI TOÁN

1. Mô tả quy trình xây dựng mô hình phân loại văn bản

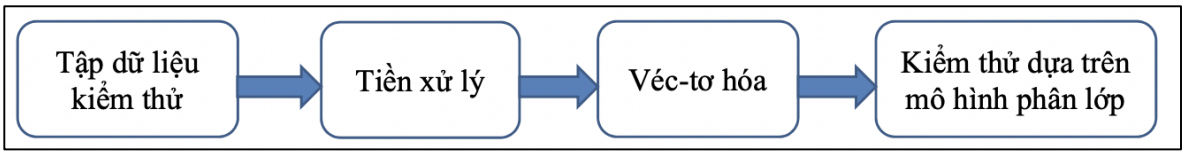
Quy trình xây dựng mô hình phân loại văn bản sử dụng các thuật toán học máy, gồm hai quá trình: huấn luyện và dự đoán.

Đầu vào của quá trình huấn luyện là các dữ liệu văn bản và các nhãn tương ứng với chủ đề cần phân loại. Quá trình này gồm 3 bước: tiền xử lý văn bản, trích xuất đặc trưng và huấn luyện sử dụng các thuật toán học máy [2].



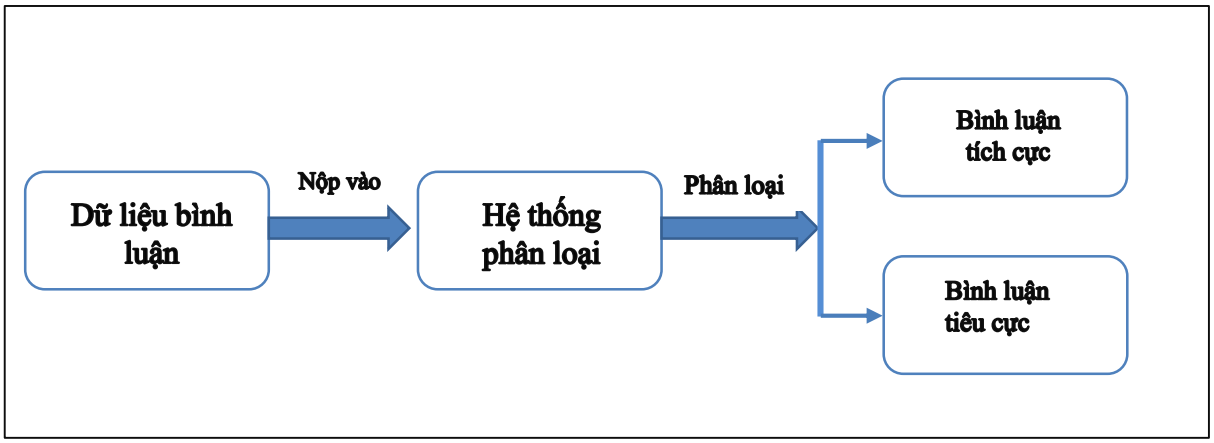
Hình 1. Giai đoạn huấn luyện

Đầu ra của quá trình huấn luyện là mô hình đã được xây dựng và các tham số tối ưu tương ứng cho mô hình [2].



Hình 2. Giai đoạn kiểm thử

Kiến trúc tổng thể về mô hình phân lớp dữ liệu bình luận sản phẩm trên website thegioididong.com [2].



Hình 3. Kiến trúc mô hình phân lớp dữ loại bình luận sản phẩm

CHƯƠNG 2: THIẾT KẾ VÀ CÀI ĐẶT

2.1. Chuẩn bị dữ liệu

Dữ liệu là điều tất yếu phải có trong bài toán này. Trong quá trình xây dựng bài toán phân lớp văn bản, ta cần phải quan tâm hơn cả là ở bước chuẩn bị và tiền xử lý dữ liệu vì sẽ có ảnh hưởng đến mô hình dự đoán có độ chính xác cao hay thấp ? [3]

Với bài toán "Phân lớp dữ liệu bình luận sản phẩm trên website thegioididong.com", tập dữ liệu sẽ gồm:

- Nội dung bình luận của khách hàng
- Số lượng sao tương ứng với bình luận

Việc thu nhập dữ liệu sẽ được thực hiện bằng đoạn code Python sau:

Lưu ý: đoạn code cào dữ liệu được viết để sử dụng trên colab của google nên có thể sẽ không chạy đối với các nền tảng khác.

- Import và setup để tiến hành cào

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from time import sleep
from random import randrange
import json
from bs4 import BeautifulSoup
from selenium.webdriver.common.keys import Keys
import re
import requests

#setting
headers = { 'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_1)' +
            'AppleWebKit/537.36 (KHTML, like Gecko) Chrome/39.0.2171.95 Safari/537.36'}
options = webdriver.ChromeOptions()
options.add_argument('--headless')
options.add_argument('--no-sandbox')
options.add_argument("--disable-infobars")
options.add_argument("start-maximized")
options.add_argument("--disable-extensions")
options.add_argument('--disable-dev-shm-usage')
options.add_experimental_option("prefs", {
    "profile.default_content_setting_values.notifications": 1
})
```

Hình 4. Thu nhập dữ liệu bước 1

- Ta truy cập vào đường dẫn www.thegioididong.com/dtdd để tiến hành cào dữ liệu từ toàn bộ điện thoại. Tuy nhiên khi truy cập vào đường dẫn trên thì website

không hiển thị toàn bộ số điện thoại hiện có mà ta cần ấn vào nút xem thêm để website có thể hiển thị toàn bộ điện thoại hiện có.

```
browser = webdriver.Chrome('chromedriver', options = options)
browser.get('https://www.thegioididong.com/dtdd')
button_show_more = browser.find_element(By.XPATH, '/html/body/section/div[3]/div[2]/a')
while True:
  try:
    button_show_more.click()
    sleep(3)
  except:
    break

html = browser.page_source
soup = BeautifulSoup(html, 'html.parser')
browser.close()
```

Hình 5. Thu nhập dữ liệu bước 2

- Ta tiến hành lấy toàn bộ đường dẫn đến từng chiếc điện thoại

```
list_products = soup.findAll('a', {'class':'main-contain'})
g_list_phones = []
list_links = [link.attrs['href'] for link in list_products]
for link in list_links:
  print(link[:len(link) - 8])
```

```
/dtdd/vivo-y53s
/dtdd/oppo-a74
/dtdd/xiaomi-redmi-note-10s
/dtdd/oppo-a93
/dtdd/vivo-y51-2020
/dtdd/realme-7i
/dtdd/realme-6
/dtdd/xiaomi-redmi-note-10s-6gb
```

Hình 6. Thu nhập dữ liệu bước 3

- Hàm cào các comment dựa trên đường dẫn tìm được

```
def get_reviews(link):
  #Truy cập vào trang đánh giá
  number_page = 1
  crawl_link = 'https://www.thegioididong.com' + link +'/danh-gia?p='
  get_request = requests.get(crawl_link + str(number_page), headers = headers)
  get_html = BeautifulSoup(get_request.content, 'html.parser')
  #Lấy tên sản phẩm và in ra tên sản phẩm
  product_name = get_html.find('div', class_='box-pdt__content').find('h3').text
  print('Begin collect data from', product_name)
  #Lấy và in ra tiêu đề của đường dẫn hiện tại
  print('title:', get_html.title.text)
  #Khởi tạo mảng chứa các comment sẽ cào được
  list_reviews = []
  while True:
    #Truy cập vào trang đánh giá dựa theo số trang
    get_request = requests.get(crawl_link + str(number_page), headers = headers)
    get_html = BeautifulSoup(get_request.content, 'html.parser')
    #Liệt kê toàn bộ các thẻ div chứa các comment
    reviews = get_html.findAll('div', class_ = 'comment__item par')
    number_page += 1
    #Dừng lại nếu không tìm thấy comment nào
    if len(reviews) == 0:
      break
    #Duyệt qua từng thẻ div chứa comment để lấy thông tin
    for review in reviews:
      #Lấy tên người dùng
      name = review.find('p', class_='txtname').text
      #Lấy nội dung comment
      content = review.find('p', class_='cmt-txt').text
      #Lấy đánh giá sao
      star = len(review.findAll('i', class_='icon-star'))
      #Đưa vào mảng chứa comment
      list_reviews.append({
          'name': name,
          'content': content,
          'star': star,
      })
    sleep(3)
  print('Collect data from', product_name, 'is done!, found', str(len(list_reviews)), 'comments')
  print('--------------------------------------------------')
  return list_reviews
```

Hình 7. Thu nhập dữ liệu bước 4

```
list_products = soup.findAll('a', {'class':'main-contain'})
list_reviews = []
list_links = [link.attrs['href'] for link in list_products]
for link in list_links:
  list_reviews += get_reviews(link[:len(link) - 8])

#Lưu lại thành file json
with open('data_TGDD.json', 'w', encoding='utf-8') as file:
 json.dump(list_reviews, file)
```

Hình 8. Thu nhập dữ liệu bước 5

2.2. Trực quan hoá dữ liệu

Data visualization là gì? Trực quan hóa dữ liệu (Data visualization) là một sự thể hiện dữ liệu hoặc là các thông tin thành biểu đồ, đồ thị hoặc về định dạng trực quan hoá dữ liệu khác [4].

Chúng ta cần trực quan hóa dữ liệu (Data visualization) vì một bản tóm tắt thông tin trực quan hoá dữ liệu sẽ giúp việc xác định đến những mô hình và với những xu hướng được dễ dàng hơn so với những việc xem qua hàng ngàn về bảng tính. Nó sẽ giúp bộ não của con người hoạt động. Vì với những mục đích của phân tích dữ liệu là sẽ tìm thấy về những insight ẩn sau dữ liệu, trực quan hoá dữ liệu sẽ có những giá trị hơn nhiều khi được trực quan hoá dữ liệu [4].

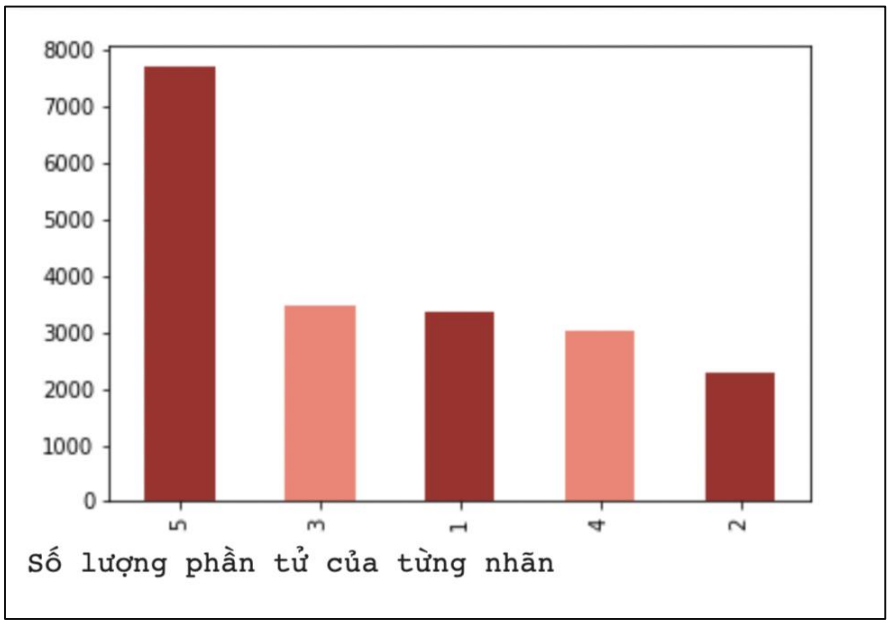
2.2.1. Trực quan hoá tập dữ liệu

Dưới đây là đoạn code viết bằng Python áp dụng lên tập dữ liệu để xem sự phân bố giữa các giá trị của nhãn.

```
df.star.value_counts().plot(kind="bar", color=["brown", "salmon"])
plt.show()
porc = (len(df[df.star==1]) / len(df.star)) * 100
print('Số lượng phần tử của từng nhãn')
```

Hình 9. Trực quan hoá dữ liệu

Sau khi trực quan hoá, có thể thấy: tập dữ liệu không cân bằng. Vì có sự chênh lệch lớn giữa nhãn có giá trị là 5 so với các nhãn còn lại qua hình



Hình 10. Sự phân bố giữa tỉ lệ các giá trị nhãn

Kết luận: gộp nhãn có giá trị 5 và 4 lại là nhãn đánh giá tích cực (nhãn 1). Nhãn có giá trị 1,2 và 3 thành nhãn đánh giá tiêu cực (nhãn 0).

2.2.2. Chuyển đổi nhãn

Để không xuất hiện tập dữ liệu mất cân bằng sẽ ảnh hưởng tới kết quả dự đoán. Vì thế ta sẽ tiến hành gộp chuyển đổi từ 5 giá trị nhãn (1, 2, 3, 4, 5) thành hai giá trị nhãn là 0 (gộp nhãn 1, 2, 3) – nhãn đánh giá tiêu cực và nhãn 1 (gộp nhãn 4, 5) – nhãn đánh giá tích cực.

Dưới đây là đoạn code viết bằng Python áp dụng lên tập dữ liệu để chuyển đổi giá trị nhãn tham khảo. Đoạn code được tham khảo từ [5].

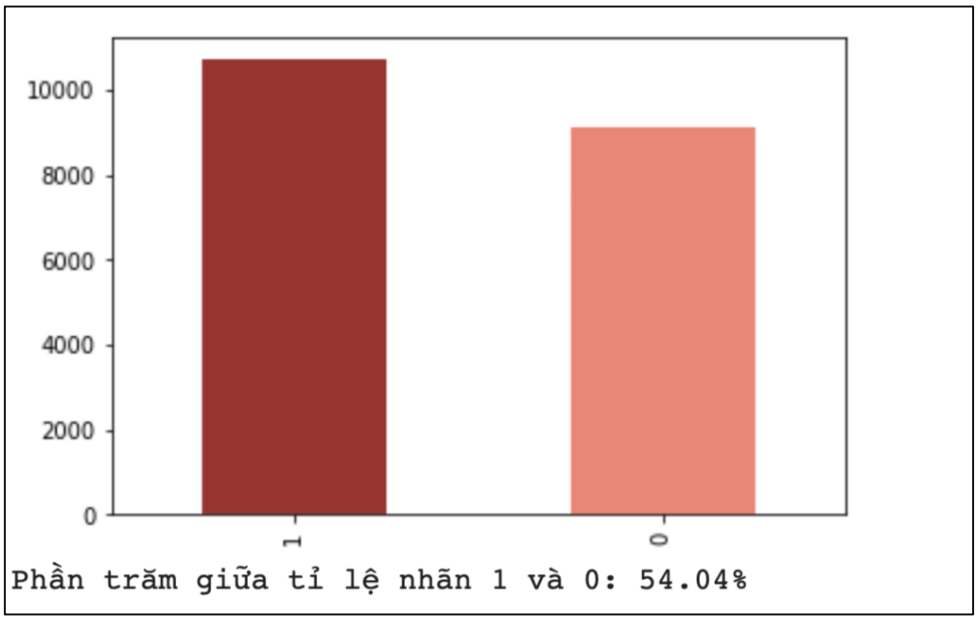
```
def change_label(star):
    if star <= 3:
        return 0
    return 1

#Chuyển đổi nhãn thành 0: tiêu cực và 1: tích cực
df['label'] = df['star'].apply(change_label)

df.label.value_counts().plot(kind="bar", color=["brown", "salmon"])
plt.show()
porc = (len(df[df.label==1]) / len(df.label)) * 100
print('Phần trăm giữa tỉ lệ nhãn 1 và 0: {:.2f}%'.format(porc))
```

Hình 11. Chuyển đổi nhãn

Sau khi chuyển đổi nhãn, tập dữ liệu phân bố theo: (54%:46%) hay (6:4). Tập dữ không quá mất cân bằng, không ảnh hưởng đáng kể tới kết quả dự đoán.



Hình 12. Kết quả chuyển đổi nhãn thành giá trị 0 và 1

2.3. Tiền xử lý dữ liệu

Bước tiền sử lý dữ liệu là quá trình chuẩn hóa dữ liệu và loại bỏ các thành phần không có ý nghĩa cho việc phân loại văn bản. Chúng ta sẽ tiến hành tiền xử lý dữ liệu trước khi đưa vào huấn luyện mô hình phân loại văn bản. Việc tiền xử lý dữ liệu là hết sức quan trọng để đảm bảo mô hình đạt được kết quả tốt [1].

Tiền xử lý dữ liệu tiếng Việt cho bài toán phân loại văn bản thường gồm các việc sau:

- Đưa về viết thường (lowercase)
- Xoá kí tự xuống dòng
- Xử lý dấu câu
- Xoá bớt khoảng trắng thừa
- Thay thế các từ viết tắt
- Xoá các kí tự lặp
- Thực hiện tách từ tiếng Việt (sử dụng thư viện tách từ như pyvi)
- Chuẩn hóa bảng mã Unicode (đưa về Unicode tổ hợp dựng sẵn)

2.3.1. Đưa về 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 (vì máy tính hiểu hoa thường là 2 từ khác nhau) và tăng độ chính xác hơn cho mô hình.

Đoạn code viết bằng Python sau đây, sẽ mô tả việc đưa dữ liệu về viết thường:

```
def preProcessing(txt):
  txt = txt.lower()
  return txt
```

Hình 13. Đưa dữ liệu về viết thường

2.3.2. Làm sạch dữ liệu

Tiền xử lý bao gồm việc loại bỏ các dữ liệu không có tác dụng cho việc phân loại văn bản. Việc này giúp:

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

Các dấu ngắt câu và các ký tự đặc biệt khác không giúp bạn phân loại một văn bản thuộc chuyên mục nào. Do đó, chúng ta nên loại bỏ nó đi [1].

2.3.2.1. Xoá kí tự xuống dòng

```
#Xóa kí tự xuống dòng
txt = " ".join(re.sub("\n", " ", txt).split())
```

Hình 14. Xoá kí tự xuống dòng

2.3.2.2. Xử lý dấu câu

```
#Xử lý dấu câu
def split_punctuations(txt):
    punctuations = '@#!?+&*[]-%:/();$=><|{}^_' + "'`"
    for p in punctuations:
      txt = txt.replace(p, f' {p} ')
    return txt
```

Hình 15. Xử lý dấu câu

2.3.2.3. Xoá khoảng trắng thừa

```
def preProcessing(txt):
  return " ".join(txt.split())
```

Hình 16. Xoá khoảng trắng thừa

2.3.3. Tách từ

Một công đoạn quan trọng bậc nhất trong xử lý ngôn ngữ tự nhiên, do Tiếng Việt có độ phức tạp hơn ngôn ngữ khác (bởi có các từ ghép). Việc tách từ theo nhiều cách khác nhau có thể gây ra sự hiểu nhầm về mặt ngữ nghĩa [6].

Đối với tập dữ liệu này sẽ sử dụng thư viện nguồn mở pyvi để thực hiện tách từ:

```
#Tách từ
txt = ViTokenizer.tokenize(txt)
```

Hình 17. Tách từ

2.3.4. Chuẩn hoá

Đưa văn bản từ các dạng không đồng nhất về cùng một dạng nhằm tối ưu bộ nhớ lưu trữ và tính xác khi dự đoán.

Trong tiếng Việt có 2 loại mã Unicode là Unicode tổ hợp và Unicode dựng sẵn, làm cho mô hình máy học khó phân biệt chính xác từ vựng. Ngoài ra, cách bỏ dấu tiếng Việt và cách viết tắt không giống nhau ở mỗi khu vực cũng làm ảnh hưởng đến tỷ lệ nhận dạng từ ngữ của máy [3].

2.3.4.1. Chuẩn hóa unicode sang chuẩn unicode dựng sẵn

Để chuẩn hóa dữ liệu đưa về cùng 1 chuẩn Unicode dựng sẵn như câu lệnh bên dưới:

```
#Chuẩn hóa unicode sang chuẩn unicode dựng sẵn
def covert_unicode(txt):
    return re.sub(
    r'à|á|ả|ã|ạ|ầ|ấ|ẩ|ẫ|ậ|ằ|ắ|ẳ|ẵ|ặ|è|é|ẻ|ẽ|ẹ|ề|ế|ể|ễ|ệ|ì|í|ỉ
    |ĩ|ị|ò|ó|ỏ|õ|ọ|ồ|ố|ổ|ỗ|ộ|ờ|ớ|ở|ỡ|ợ|ù|ú|ủ|ũ|ụ|ừ|ứ|ử|ữ|ự|ỳ|ý|ỷ|ỹ
    |ỵ|À|Á|Ả|Ã|Ạ|Ầ|Ấ|Ẩ|Ẫ|Ậ|Ằ|Ắ|Ẳ|Ẵ|Ặ|È|É|Ẻ|Ẽ|Ẹ|Ề|Ế|Ể|Ễ|Ệ|Ì|Í|Ỉ|Ĩ|Ị
    |Ò|Ó|Ỏ|Õ|Ọ|Ồ|Ố|Ổ|Ỗ|Ộ|Ờ|Ớ|Ở|Ỡ|Ợ|Ù|Ú|Ủ|Ũ|Ụ|Ừ|Ứ|Ử|Ữ|Ự|Ỳ|Ý|Ỷ|Ỹ|Ỵ',
    lambda x: dicchar[x.group()], txt)
```

Hình 18. Chuẩn hoá unicode sang chuẩn unicode dựng sẵn

2.3.4.2. Thay thế các từ viết tắt

Sử dụng bộ từ điển thay thế các kiểu gõ dấu và viết tắt.

```
#Xử lý từ viết tắt
def replace_acronyms(txt):
  pat = re.compile(r"\b(%s)\b" % "|".join(acronyms))
  txt = pat.sub(lambda m: acronyms.get(m.group()), txt)
  return txt
```

Hình 19. Thay thế các từ viết tắt

Với danh sách từ viết tắt được thống kê thủ công trong tập tin acronyms_vi được tham khảo từ [3] và đã được nhóm bổ sung các từ viết tắt thông dụng:

acronym_vi.txt

ship	vận chuyển
shop	cửa hàng
m	mình
mik	mình
ko	không
k	không
kh	không
khong	không
kg	không
khg	không
tl	trả lời
rep	trả lời
r	rồi
fb	facebook
face	faceook
thanks	cảm ơn
thank	cảm ơn
tks	cảm ơn
tk	cảm ơn
ok	tốt
oki	tốt
okie	tốt
sp	sản phẩm
dc	được
vs	với
đt	điện thoại
thjk	thích
thik	thích
qá	quá

Hình 20. Danh sách các từ thay thế các từ viết tắt

2.3.4.3. Xoá các từ lặp

Ngoài ra, các từ có ý nghĩa nhưng bị viết trùng lắp như: hoaa, caii...cũng làm ảnh hưởng đến tỷ lệ chính xác của mô hình, vì vậy cần phải chuẩn hóa bằng đoạn code Python sau:

```
#Xử lý các từ lặp
def remove_loop_char(txt):
  txt = re.sub(r'([A-Z])\1+', lambda m: m.group(1).upper(), str(txt), flags=re.IGNORECASE)
  txt = re.sub(r'[^\s\wáàảãạăắằẳẵặâấầẩẫậéèẻẽẹêếềểễệóòỏõọôốồổỗộơớờởỡợíìỉĩịúùủũụưứừửữựýỳỷỹỵđ_]',' ',txt)
  return txt
```

Hình 21. Xử lý từ lặp

2.3.5. Các bước tiền xử lý đã áp dụng lên tập dữ liệu

Dưới đây là đoạn code bằng Python, áp dụng các bước tiền xử lý dữ liệu:

```
#Tiền xử lý
def preProcessing(txt):
  txt = txt.lower()
  #Xóa kí tự xuống dòng
  txt = " ".join(re.sub("\n", " ", txt).split())
  #Xử lý dấu câu
  txt = split_punctuations(txt)
  #Thay thế các từ viết tắt
  txt = replace_acronyms(txt)
  #Xóa các kí tự lặp
  txt = remove_loop_char(txt)
  #Tách từ
  txt = ViTokenizer.tokenize(txt)
  #Xóa từ dừng
  # txt = remove_stopwords(txt)
  #Chuẩn hóa sang 1 kiểu unicode
  txt = covert_unicode(txt)
  txt = txt.lower()
  # Xóa bớt các khoảng trắng thừa
  return " ".join(txt.split())

df['content_clean'] = df['content'].apply(preProcessing)
```

Hình 22. Cấu trúc các bước tiền xử lý dữ liệu đã áp dụng

Đối với bài toán này, sẽ không có bước loại bỏ stopword vì đã thử áp dụng nhưng làm giảm độ chính xác khi dự đoán.

2.4. Xây dựng mô hình phân loại văn bản

Trước khi huấn luyện mô hình phân loại văn bản, ta cần xây dựng tập huấn luyện và tập kiểm thử. Việc này là cần thiết để đánh giá kết quả huấn luyện, lựa chọn mô hình cũng như tinh chỉnh để mô hình cho tốt hơn.

2.4.1. Xây dựng tập train/test

Đối với tập dữ liệu này, ta sẽ áp dụng nghi thức hold-out: lấy ngẫu nhiên 2/3 tập dữ liệu để học và 1/3 tập dữ liệu còn lại dùng cho kiểm tra.

Dưới đây là đoạn code viết bằng Python dùng để phân chia tập dữ liệu:

```
#Phân chia tập dữ liệu huấn luyện và test bằng nghi thức Hold-out tỉ lệ 7/3
X, y = df['content_clean'], df['label']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state = 42)
```

Hình 23. Phân chia tập dữ liệu

Tập dữ liệu X_train và y_train sau khi được phân chia:

```
X_train

8839     nếu ai chơi trò_chơi không nên mua con này quá...
7301     có ai bị loa nó dính mấy cái lỗ nhỏ trên loa n...
16465    vsmart live 4 dùng rất ồn tuy_nhiên loa vẫn cò...
4369     khi mở loa to phần lưng điện_thoại rất rung kh...
3021     mua về là bị lỗi zalo mỗi lần vào là bấn cả ch...
                              ...
11284    máy mới mua về xài rất tốt không có điều gì để...
11964    mong quản_trị viên phản_hồi giúp em_em mới mua...
5390     nếu tôi làm mất cục sạc chính hãng thì khi tôi...
860      còn thiếu cái cuối_cùng là giá rẻ thôi còn nhi...
15795    quá ngon trong tầm giá máy đẹp màn_hình sắc né...
Name: content_clean, Length: 13870, dtype: object
```

Hình 24. Tập dữ liệu X_train

```
y_train

8839     0
7301     1
16465    1
4369     0
3021     0
        ..
11284    0
11964    0
5390     1
860      1
15795    1
Name: label, Length: 13870, dtype: int64
```

Hình 25. Tập dữ liệu y_train

2.4.2 Vectơ hoá văn bản

Ở bước này, chúng ta sẽ đưa dữ liệu dạng văn bản cụ thể là tập dữ liệu X_train và X_test đã được xử lý về dạng vector thuộc tính có dạng số học [7]. Có nhiều các khác nhau để đưa dữ liệu văn bản dạng *text* về dữ liệu dạng số mà chúng ta có thể thực hiện như:

- Count Vectors
- TF-IDF Vectors
    - Word level
    - N-Gram level
    - Character level
- Word Embeddings
- Text / NLP
- Topic Models

Nhưng trong bài toán này chỉ áp dụng 2 phương pháp điển hình là: Count Vectors as features và TF-IDF Vectors (N-Gram level).

2.4.2.1. Vectơ hoá văn bản bằng Count Vectors

Khi sử dụng phương pháp này, chúng ta sẽ thu được một ma trận mà trong đó, mỗi hàng sẽ đại diện cho một văn bản, mỗi cột đại diện cho một từ có trong từ điển, và mỗi ô

(cell) sẽ chứa tần suất xuất hiện của từ trong văn bản tương ứng. Bằng cách sử dụng thư viện *sklearn*, chúng ta sẽ làm như sau [7]:

```
#Hàm chuyển dữ liệu train và test thành các vector dùng CountVectorizer
def count_vector(data_train, data_test):
    count_vectorizer = CountVectorizer(max_features=5000)
    vector_train = count_vectorizer.fit_transform(data_train)
    vector_test = count_vectorizer.transform(data_test)
    return count_vectorizer, vector_train, vector_test
#Tạo ra vector cho tập train và test với CountVectorizer
count_vector, X_train_Count, X_test_Count = count_vector(X_train, X_test)
```

Hình 26. Vector hóa văn bảng bằng CountVectorizer

2.4.2.1. Vectơ hoá văn bản bằng TF-IDF Vectors (N-Gram level)

Phương pháp TF-IDF (Term Frequency - Inverse Document Frequency), đây là một phương pháp cực kì phổ biến trong xử lý văn bản [7]. Nó được tính theo công thức dưới đây:

- TF(t) = (Number of times term t appears in a document) / (Total number of terms in the document)
- IDF(t) = log_e(Total number of documents / Number of documents with term t in it)

Chúng ta có thể thực hiện TF-IDF cho các cấp độ khác nhau của văn bản như sau:

- Word Level TF-IDF : Thực hiện tính toán dựa trên mỗi thành phần là một từ riêng lẻ.
- N-gram Level TF-IDF : Kết hợp *n* thành phần (từ) liên tiếp nhau. Ví dụ: "thủ_tướng đức nhận_lời tham_dự lễ kỷ_niệm". Khi đó, 2-gram cho ta kết quả: {thủ_tướng đức, đức nhận_lời, nhận_lời tham_dự, tham_dự lễ , lễ kỷ_niệm}. Mỗi phần từ là cặp 2 từ liên tiếp nhau [PhanLoaiVanBanViBLO].

Bằng cách sử dụng thư viện *sklearn*, chúng ta sẽ làm như sau [7]:

```
#Hàm chuyển dữ liệu train và test thành các vector dùng TfidfVectorizer
def tfidf_vector(data_train, data_test):
    tfidf_vectorizer = TfidfVectorizer(ngram_range=(1, 2), min_df=1, use_idf=True)
    vector_train = tfidf_vectorizer.fit_transform(data_train)
    vector_test = tfidf_vectorizer.transform(data_test)
    return tfidf_vectorizer, vector_train, vector_test
#Tạo ra vector cho tập train và test với TfidfVectorizer
tf_idf, X_train_tfidf, X_test_tfidf = tfidf_vector(X_train, X_test)
```

Hình 27. Vector hóa văn bảng bằng TF-IDF Vectors (N-Gram level)

2.4.3 Lưu lại model vector hoá

Để tiện sử dụng cho việc sử dụng lại dữ liệu đã được xử lý, ta sẽ lưu dữ liệu vào file *.pkl*:

```
#Lưu lại model vector hóa
import pickle

with open("vectorizes.pkl", "wb") as f:
  pickle.dump((count_vector, tf_idf), f)

f.close()
```

Hình 28. Lưu lại model vector hoá

2.4.4 Phân loại văn bản với Naives Bayes Multinomial

Naïve Bayes Classifier (NBC) là mô hình phân lớp văn bản dựa trên định lý Bayes. Một số mô hình Naïve Bayes hay sử dụng là Multinomial Naive Bayes, Bernoulli Naive Bayes và Gaussian Naive Bayes. Mô hình này thường được sử dụng để phân lớp văn bản [8].

Đây là đoạn code áp dụng mô hình **Naïve Bayes Classifier** bằng cách sử dụng thư viện *sklearn:*

```
# Với Naives Bayes Multinomial
bayes_model_cv = MultinomialNB()
bayes_model_cv.fit(X_train_Count, y_train)
y_bayes_pred = bayes_model_cv.predict(X_test_Count)

print('Độ chính xác với CountVectorizer: %.3f' % accuracy_score(y_test, y_bayes_pred))

bayes_model_tf = MultinomialNB()
bayes_model_tf.fit(X_train_tfidf, y_train)
y_bayes_pred = bayes_model_tf.predict(X_test_tfidf)

print('Độ chính xác với TfidfVectorizer: %.3f' % accuracy_score(y_test, y_bayes_pred))

#Lưu model bayes lại
with open("bayes_model.pkl", "wb") as f:
  pickle.dump((bayes_model_cv, bayes_model_tf), f)

f.close()
```

```
Độ chính xác với CountVectorizer: 0.834
Độ chính xác với TfidfVectorizer: 0.847
```

Hình 29. Mô hình Naïve Bayes Classifier

2.4.5 Phân loại văn bản với SVM

SVM (Support Vector Machine) là 1 thuật toán học máy thuộc nhóm Supervised Learning (học có giám sát) được sử dụng trong các bài toán phân lớp dữ liệu (classification) hay hồi qui (Regression). Tuy nhiên nó được sử dụng chủ yếu cho việc phân loại [9].

Đây là đoạn code áp dụng mô hình **SVM** bằng cách sử dụng thư viện *sklearn:*

```
# Với SVM
svm_model_cv = svm.SVC(kernel='linear', C=1.3)
svm_model_cv.fit(X_train_Count, y_train)
y_svm_pred = svm_model_cv.predict(X_test_Count)
print('Độ chính xác với CountVectorizer: %.3f' % accuracy_score(y_test, y_svm_pred))

svm_model_tf = svm.SVC(kernel='linear', C=1.3)
svm_model_tf.fit(X_train_tfidf, y_train)
y_svm_pred = svm_model_tf.predict(X_test_tfidf)
print('Độ chính xác với TfidfVectorizer: %.3f' % accuracy_score(y_test, y_svm_pred))

#Lưu model SVM lại
with open("svm_model.pkl", "wb") as f:
  pickle.dump((svm_model_cv, svm_model_tf), f)
f.close()
```

```
Độ chính xác với CountVectorizer: 0.828
Độ chính xác với TfidfVectorizer: 0.863
```

Hình 30. Mô hình SVM

2.4.6 Phân loại văn bản với Logistic Regression

Logistic regression là thuật toán đơn giản nhưng lại rất hiệu quả trong bài toán phân loại (Classification).

Logistic regression được áp dụng trong bài toán phân loại nhị phân (Binary classification) tức ta sẽ có hai output, hoặc có thể gọi là hai nhãn (ví dụ như 0 và 1) [10].

Đây là đoạn code áp dụng mô hình **Logistic regression** bằng cách sử dụng thư viện *sklearn:*

```
# Với Logistic Regression
linear_model_cv = LogisticRegression(solver='lbfgs', multi_class='auto', max_iter=10000)
linear_model_cv.fit(X_train_Count, y_train)
y_linear_pred = linear_model_cv.predict(X_test_Count)
print('Độ chính xác với CountVectorizer: %.3f' % accuracy_score(y_test, y_linear_pred))

linear_model_tf = LogisticRegression(solver='lbfgs', multi_class='auto', max_iter=10000)
linear_model_tf.fit(X_train_tfidf, y_train)
y_linear_pred = linear_model_tf.predict(X_test_tfidf)
print('Độ chính xác với TfidfVectorizer: %.3f' % accuracy_score(y_test, y_linear_pred))

#Lưu model logistic lại
with open("linear_model.pkl", "wb") as f:
  pickle.dump((linear_model_cv, linear_model_tf), f)

f.close()
```

```
Độ chính xác với CountVectorizer: 0.842
Độ chính xác với TfidfVectorizer: 0.864
```

Hình 31. Mô hình Logistic regression

2.4.7. Phân loại văn bản với DeepLearing – Phobert

Phobert là mô hình deep learing được xây dựng dựa trên mô hình Bert của Google. Phobert được phát triển bởi VinAI dành cho xử lý ngôn ngữ tiếng Việt [11].

Phobert sử dụng phần encoder của mô hình Transformer để làm đầu vào huấn luyện.

Cài đặt các thư viện cần thiết để sử dụng Phobert.

Lưu ý: đoạn code cào dữ liệu được viết để sử dụng trên colab của google nên có thể sẽ không chạy đối với các nền tảng khác.

```
!pip install pyvi
!pip install transformers
# !pip install emoji
!wget https://public.vinai.io/PhoBERT_base_transformers.tar.gz
!tar -xzvf PhoBERT_base_transformers.tar.gz
!pip install fastBPE
!pip install fairseq
```

Hình 32. Cài đặt thư viện PhoBert

Tiếp theo ta tiến hành load từ điển cũng như bpe giúp encode 1 câu đầu vào thành 1 mảng các subword. Mỗi 1 subword có 1 id riêng trong từ điển của Phobert.

```
from fairseq.data.encoders.fastbpe import fastBPE
from fairseq.data import Dictionary
import argparse

parser = argparse.ArgumentParser()
parser.add_argument('--bpe-codes',
    default="/content/PhoBERT_base_transformers/bpe.codes",
    required=False,
    type=str,
    help='path to fastBPE BPE'
)

args, unknown = parser.parse_known_args()
bpe = fastBPE(args)

# Load the dictionary
vocab = Dictionary()
vocab.add_from_file("/content/PhoBERT_base_transformers/dict.txt")
```

Hình 33. Load từ điển cũng như bpe

Tiền xử lý dữ liệu đầu vào cho mô hình

```
def remove_emoji(txt):
    emoji_pattern = re.compile("["
        u"\U0001F600-\U0001F64F"  # emoticons
        u"\U0001F300-\U0001F5FF"  # symbols & pictographs
        u"\U0001F680-\U0001F6FF"  # transport & map symbols
        u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
                           "]+", flags=re.UNICODE)
    return emoji_pattern.sub(r" ", txt)

def fix_number(txt):
    txt = re.sub(r'(?<=\d)(?=[^\d\s])|(?<=[^\d\s])(?=\d)', ' ', txt)
    return txt

def preProcessing(txt):
    txt = txt.lower()
    txt = txt.replace('\n',' ')
    txt = txt.replace(':)',' ')
    txt = txt.replace(':))',' ')
    txt = txt.replace(':)))',' ')
    txt = txt.replace('=)',' ')
    txt = txt.replace('=))',' ')
    txt = txt.replace('=)))',' ')
    txt = txt.replace(':(',' ')
    txt = txt.replace(':((',' ')
    txt = txt.replace(':(((',' ')
    txt = txt.replace('...',' . ')
    txt = split_punctuations(txt)
    txt = fix_number(txt)
    #Thay thế các từ viết tắt
    txt = replace_acronyms(txt)
    txt = remove_emoji(txt)
    txt = ViTokenizer.tokenize(txt)
    txt = txt.lower()
    return txt
```

Hình 34. Tiền xử lý cho mô hình PhoBert

Ta bắt đầu encode dữ liệu đầu vào danh sách các mảng subword.

```
MAX_LEN = 84

X = df['clean_content']
y = df['label'].to_list()

train_sents, val_sents, train_labels, val_labels = train_test_split(X, y, test_size=0.2, random_state = 42)

#Tiến hành encode
train_ids = []
val_ids = []

for sent in train_sents:
    subwords = '<s> ' + bpe.encode(sent) + ' </s>'
    encoded_sent = vocab.encode_line(subwords, append_eos=True, add_if_not_exist=False).long().tolist()
    train_ids.append(encoded_sent)

for sent in val_sents:
    subwords = '<s> ' + bpe.encode(sent) + ' </s>'
    encoded_sent = vocab.encode_line(subwords, append_eos=True, add_if_not_exist=False).long().tolist()
    val_ids.append(encoded_sent)

#List các id
train_ids = pad_sequences(train_ids, maxlen=MAX_LEN, dtype="long", value=0, truncating="post", padding="post")
val_ids = pad_sequences(val_ids, maxlen=MAX_LEN, dtype="long", value=0, truncating="post", padding="post")
```

Hình 35. Encode dữ liệu đầu vào sách các mảng subword

Tạo thêm các mask là mảng gồm các số 0 và 1 nhằm kiểm tra các giá trị trong chuỗi đã được padding hay chưa.

```
train_masks = []
for sent in train_ids:
    mask = [int(token_id > 0) for token_id in sent]
    train_masks.append(mask)

val_masks = []
for sent in val_ids:
    mask = [int(token_id > 0) for token_id in sent]

    val_masks.append(mask)
```

Hình 36. Tạo mask để tra padding

Chuyển đổi dữ liệu đầu vào thành DataLoader

```
from torch.utils.data import TensorDataset, DataLoader, RandomSampler, SequentialSampler
import torch
BATCH_SIZE=12

train_inputs = torch.tensor(train_ids)
val_inputs = torch.tensor(val_ids)
train_labels = torch.tensor(train_labels)
val_labels = torch.tensor(val_labels)
train_masks = torch.tensor(train_masks)
val_masks = torch.tensor(val_masks)

train_data = TensorDataset(train_inputs, train_masks, train_labels)
train_sampler = SequentialSampler(train_data)
train_dataloader = DataLoader(train_data, sampler=train_sampler, batch_size=BATCH_SIZE)

val_data = TensorDataset(val_inputs, val_masks, val_labels)
val_sampler = SequentialSampler(val_data)
val_dataloader = DataLoader(val_data, sampler=val_sampler, batch_size=BATCH_SIZE)
```

Hình 37. Chuyển đổi dữ liệu đầu vào thành DataLoader

Tiến hành load mô hình Phobert sử dụng GPU

```
from transformers import RobertaForSequenceClassification, RobertaConfig, AdamW, BertForSequenceClassification

config = RobertaConfig.from_pretrained(
    "/content/PhoBERT_base_transformers/config.json", from_tf=False, num_labels = 2, output_hidden_states=False,
)
BERT_SA = BertForSequenceClassification.from_pretrained(
    "/content/PhoBERT_base_transformers/model.bin",
    config=config
)

BERT_SA.cuda()
```

Hình 38. Tiến hành load mô hình Phobert sử dụng GPU

Hàm huấn luyện

```python
def flat_accuracy(logits, labels):
    preds = []
    for line in logits:
        preds.append(torch.argmax(line).cpu().numpy())
    return accuracy_score(preds, labels.cpu().numpy())

def train_model(model):
    model.train()
    train_loss = 0
    acc = []
    for batch in train_dataloader:
        b_input_ids = batch[0].to(device)
        b_input_mask = batch[1].to(device)
        b_labels = batch[2].to(device)
        BERT_SA.zero_grad()
        outputs = model(b_input_ids,
            token_type_ids=None,
            attention_mask=b_input_mask,
            labels=b_labels)
        loss = outputs[0]
        logits = outputs[1]
        acc.append(flat_accuracy(logits, b_labels))
        loss.backward()
        torch.nn.utils.clip_grad_norm_(model.parameters(), 1.0)
        optimizer.step()
        train_loss += loss.item()

    return train_loss, round(np.mean(acc), 3)
```

Hình 39. Hàm huấn luyện mô hình PhoBert

Tiến hành huấn luyện

```
from tqdm.notebook import tqdm
import time
device = 'cuda'
epochs = 4

param_optimizer = list(BERT_SA.named_parameters())
no_decay = ['bias', 'LayerNorm.bias', 'LayerNorm.weight']
optimizer_grouped_parameters = [
    {'params': [p for n, p in param_optimizer if not any(nd in n for nd in no_decay)], 'weight_decay': 0.01},
    {'params': [p for n, p in param_optimizer if any(nd in n for nd in no_decay)], 'weight_decay': 0.0}
]

optimizer = AdamW(BERT_SA.parameters(), lr=1e-5, correct_bias=False)
train_loss_ = []
test_loss_ = []

for epoch in range(epochs):
    begin_time = time.time()
    loss, acc = train_model(BERT_SA)
    done_time = time.time() - begin_time
    print(f"Epoch {epoch + 1}: {round(done_time)} seconds - loss {round(loss, 3)} - accuracy {acc}")
    train_loss_.append([loss, acc])
```

Hình 40. Tiến hành huấn luyện mô hình PhoBert

Mô hình đã huấn luyện xong tiến hành dự đoán để kiểm tra độ chính xác của mô hình PhoBert.

```
device = 'cuda'
preds = []
for batch in val_dataloader:
  b_input_ids = batch[0].to(device)
  b_input_mask = batch[1].to(device)
  with torch.no_grad():
    pred=BERT_SA(b_input_ids,token_type_ids=None,attention_mask=b_input_mask)
    preds.append(pred[0])
```

```
lab = []
for line in preds:
  for i in line:
    predicts = i.to('cpu').detach().numpy().copy()
    lab.append(np.argmax(predicts))
```

Hình 41. Tiến hành dự đoán mô hình để dự đoán độ chính xác của mô hình PhoBert

2.5. Đánh giá mô hình phân loại văn bản

2.5.1 So sánh các vectơ hoá

Ta có thể thấy được độ chính xác khi sử dụng TfidfVectorizer luôn cao hơn so với khi sử dụng CountVectorizer.

2.5.2 So sánh các mô hình

Mô hình SVM và Logistic Regresstion sử dụng TfidfVectorizer có độ chính xác cao vượt trội hơn so với mô hình Bayes sử dụng TfidfVectorizer.

Mô hình Logistic Regresstion sử dụng CountVectorizer có độ chính xác cao hơn so với SVM.

Mô hình Logistic Regresstion luôn cao hơn so với 2 mô hình còn lại kể cả sử dụng CountVectorizer hay là sử dụng TfidfVectorizer.

2.6. Xây dựng Restfu API phân loại

Sử dụng ngôn ngữ python với Flask để xây dựng RESTful API để phân loại bình luận.

Ở url / ta sẽ render template index.html, thực hiện truy vấn cơ sở dữ liệu Firebase bên phía Client.

```python
@app.route('/')
@cross_origin(origin='*')
def index_process():
    return render_template("index.html")
```

Hình 42. API của url /

```javascript
$(document).ready(function() {
    firebase.initializeApp({
        apiKey: "AIzaSyDhpFRiDvSe5u7cehOMJG9RNSMd5dvWHKg",
        authDomain: 'sentiment-prediction-tgdd.firebaseapp.com',
        projectId: 'sentiment-prediction-tgdd'
    });
    var db = firebase.firestore();
```

Hình 43. Đọc kết nối API của client

Ở url /predict ta sẽ lấy dữ liệu từ request gồm có nội dung bình luận và tên người dùng, sau đó tiến hành load model và thực hiện dự đoán rồi đưa dự đoán lên cơ sở dữ liệu Firebase.

```python
@app.route('/predict', methods=["POST", "GET"])
@cross_origin(origin='*')
def predict_star():
    comment = request.form['input_comment']
    username = request.form['input_name']
    if len(username) == 0:
        username = "Anonymous"
```

Hình 44. Url /predict

```
df = pd.DataFrame()
df['comment'] = [comment]
df['comment'] = df['comment'].apply(PREPROCESS.preProcessing)
cmt = df['comment']
#load vectorizes
count_vector, tf_idf_vector = MODEL.load_vectorizes()
input_cv = count_vector.transform(cmt)
input_tf = tf_idf_vector.transform(cmt)
output = []
#Ta sẽ dùng mô hình logistic với tf-idf để dự đoán chính. Các
# để phục vụ cho chức năng xem chi tiết nếu người dùng muốn x
#load bayes
bayes_cv, bayes_tf = MODEL.load_bayes_model()
#Lưu lại dự đoán
output.append(bayes_cv.predict(input_cv)[0])
output.append(bayes_tf.predict(input_tf)[0])

#load svm
svm_cv, svm_tf = MODEL.load_svm_model()
#Lưu lại dự đoán
output.append(svm_cv.predict(input_cv)[0])
output.append(svm_tf.predict(input_tf)[0])

#load svm
logistic_cv, logistic_tf = MODEL.load_svm_model()
#Lưu lại dự đoán
output.append(logistic_cv.predict(input_cv)[0])
output.append(logistic_tf.predict(input_tf)[0])
```

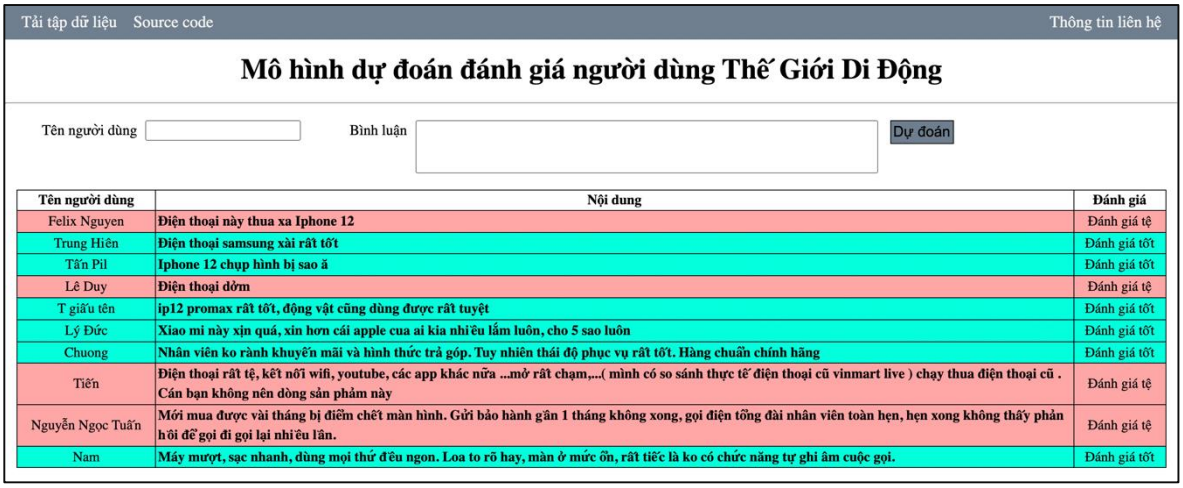
Hình 45. Load các mô hình và dự đoán

```
doc_ref = db.collection(u'data').document()
doc_ref.set({
    u'comment': comment,
    u'username': username,
    u'bayes_cv': int(output[0]),
    u'bayes_tf': int(output[1]),
    u'svm_cv': int(output[2]),
    u'svm_tf': int(output[3]),
    u'logis_cv': int(output[4]),
    u'logis_tf': int(output[5]),
})
predict = 'Đánh giá tệ'
if output[5] == 1:
    predict = 'Đánh giá tốt'
return "<script> alert('"+ predict +"'); window.location.replace('/') </script>"
```

Hình 46. Lưu kết quả dự đoán vào database

2.7. Xây dựng website dự đoán đánh giá

Đường dẫn tới website phân lớp dữ liệu bình luận sản phẩm trên website thegioididong.com : https://sentiment-analysis-tgdd.herokuapp.com/



Hình 47. Giao diện website

CHƯƠNG 3: ĐÁNH GIÁ KIỂM THỬ PHẦN KẾT LUẬN

❖ Kết luận

Các bước đã thực hiện tốt trong đồ án:

- Cào dữ liệu từ website thegioididong.com. Dữ liệu cần thu thập bao gồm: nội dung bình luận và số lượng sao tương ứng với bình luận. ✓
- Làm sạch dữ liệu, tách từ, chuẩn hóa từ, loại bỏ stopword. ✓
- Xây dựng mô hình (sử dụng ít nhất 3 giải thuật trong đó có PhoBERT). ✓
- Đánh giá mô hình. ✓
- Xây dựng Restful API phân loại. ✓
- Xây dựng website dự đoán đánh giá. ✓

❖ Hướng phát triển của đề tài

- Thu thập thêm dữ liệu nhằm từ bài toán phân loại 2 nhãn thành bài toán phân loại 5 nhãn (tương ứng 5 lớp: 1 sao, 2 sao, 3 sao, 4 sao, 5sao).
- Cải thiện giao diện website thân thiện với người dùng.
- Đưa mô hình PhoBert lên website.

TÀI LIỆU THAM KHẢO

[1] Nguyễn Văn Hiếu, "Phân loại văn bản tiếng Việt sử dụng machine learning". Địa chỉ: https://nguyenvanhieu.vn/phan-loai-van-ban-tieng-viet/#chuan-bi-du-lieu [Truy cập 01/12/2021].

[2] Trần Thanh Điện, Thái Nhựt Thanh và Nguyễn Thái Nghe, "Giải pháp phân loại bài báo khoa học bằng kỹ thuật máy học" , Tạp chí *Khoa học Trường Đại Học Cần Thơ*, tập 55, số 4A, trang số 33, (2019): 29-37 .

[3] Hứa Quốc Thi, *"TheGioiDiDong"*, Đồ án Khai khoáng dữ liệu, khoa Công Nghệ Thông Tin Và Truyền Thông, trường Đại Học Cần Thơ.

[4] Hiền Mai, "Trực quan hoá dữ liệu là gì? Tại sao nói Data Visualization vô cùng quan trọng", 2021-08-09 09:55:30. Địa chỉ: https://123job.vn/bai-viet/truc-quan-hoa-du-lieu-la-gi-tai-sao-noi-data-visualization-vo-cung-quan-trong-2944.html [Truy cập 03/12/2021].

[5] VITALII MOKIN, "NLP - EDA, Bag of Words, TF IDF, GloVe, BERT". Địa chỉ: https://www.kaggle.com/vbmokin/nlp-eda-bag-of-words-tf-idf-glove-bert [Truy cập 04/12/2021].

[6] "Ứng dụng công nghệ trí tuệ nhân tạo trong bài toán phân loại văn bản", 11:00 | 22/03/2019 |. Địa chỉ: http://antoanthongtin.gov.vn/giai-phap-khac/ung-dung-cong-nghe-tri-tue-nhan-tao-trong-bai-toan-phan-loai-van-ban-105149 [Truy cập 04/12/2021].

[7] Nguyễn Thành Hậu, "Phân loại văn bản tự động bằng Machine Learning như thế nào?", 10/11/2018 1:50 PM. Địa chỉ: https://viblo.asia/p/phan-loai-van-ban-tu-dong-bang-machine-learning-nhu-the-nao-4P856Pa1ZY3 [Truy cập 05/12/2021].

[8] Tran Duc Tan, "Mô hình phân lớp Naive Bayes", 22/06/2019 03:16 PM. Địa chỉ: https://viblo.asia/p/mo-hinh-phan-lop-naive-bayes-vyDZO0A7lwj [Truy cập 12/12/2021].

[9] Trương Xuân Đạt, "Giới Thiệu về Mô Hình SVM", 04/11/2020. Địa chỉ: https://www.stdio.vn/computer-vision/gioi-thieu-ve-mo-hinh-svm-D15jcg [Truy cập ngày 14/12/2021].

[10] Mai Trong Nghia, "Logistic regression", 03/04/2020. Địa chỉ: http://maitrongnghia.com/2020/04/logistic-regression/ [Truy cập ngày 16/12/2021].

[11] Phạm Hữu Quang, "BERT, RoBERTa, PhoBERT, BERTweet: Ứng dụng state-of-the-art pre-trained model cho bài toán phân loại văn bản", 19/07/2020 4:34 PM. Địa chỉ: https://viblo.asia/p/bert-roberta-phobert-bertweet-ung-dung-state-of-the-art-pre-trained-model-cho-bai-toan-phan-loai-van-ban-4P856PEWZY3 [Truy cập ngày 20/12/2021].

[12] Brandon Walsh, "Creating Web APIs with Python and Flask", 05/12/2020. Địa chỉ: https://programminghistorian.org/en/lessons/creating-apis-with-python-and-flask [Truy cập ngày 25/12/2021].