ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN CUỐI KÌ

ĐỀ TÀI: PHÂN LOẠI ĐÁNH GIÁ CỦA KHÁCH HÀNG TRÊN SHOPEE

GIẢNG VIÊN: ThS. Phạm Nguyễn Trường An

LỚP : CS114.N21

NHÓM H2O:

STT	Họ và tên	MSSV
1	Nguyễn Quốc Huy Hoàng	20520051
2	Ngô Quang Vinh	19522523

Học kỳ 2 - Năm học 2022-2023

Tóm tắt đồ án

Nội dung mà nhóm đã làm:

- Đề tài của nhóm là phân loại đánh giá của khách hàng về một sản phẩm (tiêu cực/ không tiêu cực) trên shopee.
- Nhóm đã thực hiện thu thập dữ liệu bằng cách crawl các đánh giá của khách hàng trên shopee, sau đó tiến hành gán nhãn và kiểm tra chéo.
- Từ dữ liệu có được nhóm đã qua một số bước tiền xử lý sau đó tiếp tục dùng PhoBERT[1], ViDeBERTa[2] để chuyển đổi văn bản thành vector đặc trưng sau đó tiến hành sử dụng model SVM, LogisticRegression, RandomForest để huấn luyện, đánh giá, tinh chỉnh từ đó đưa ra được model tốt nhất mà nhóm đã thực nghiệm được.
- Trong nội dung nhóm đã nêu ra được các câu trả lời cho câu hỏi: WHY (Tại sao lại làm đề tài này) bằng cách đưa ra các trường hợp gặp phải trong thực tế; WHAT (Đề tài này làm cái gì) bằng cách nêu rõ input, output của bài toán; HOW (Làm đề tài này như thế nào) bằng cách đưa ra hướng giải quyết.

Các đường dẫn:

- Github của nhóm: https://github.com/hoangnqh/CS114.N21
- Link đồ án: https://github.com/hoangnqh/CS114.N21/tree/master/Final_Project
- Dataset: https://docs.google.com/spreadsheets/d/1bFf9ikhHDx9IRK6Y7wJE6jDpzME9rAAg/edit?usp=sharing&ouid=117264281666718723761&rtpof=true&sd=true
- Notebook: https://colab.research.google.com/drive/1xXNeTfU4TVi5g6odZ0cFacTT6m3BPUGV?usp=sharing

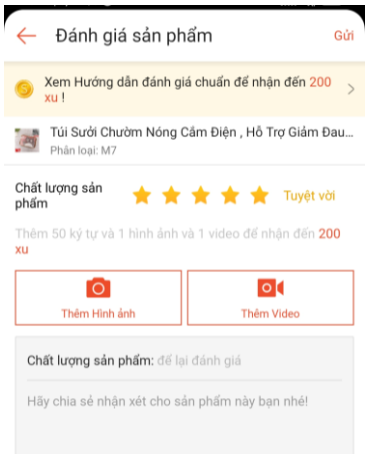
Table of Contents

I. Giới thiệu đề tài

1. Đặt vấn đề

Trong thời đại số hiện nay, việc thu thập, phân tích và hiểu được ý kiến của khách hàng đóng vai trò quan trọng đối với các cửa hàng/doanh nghiệp. Một trong những khía cạnh quan trọng trong việc nắm bắt ý kiến của khách hàng là khả năng phân loại đánh giá tiêu cực và không tiêu cực. Điều này giúp cửa hàng/doanh nghiệp định hướng chiến lược, cải thiện chất lượng sản phẩm/dịch vụ và tăng cường tương tác với khách hàng.

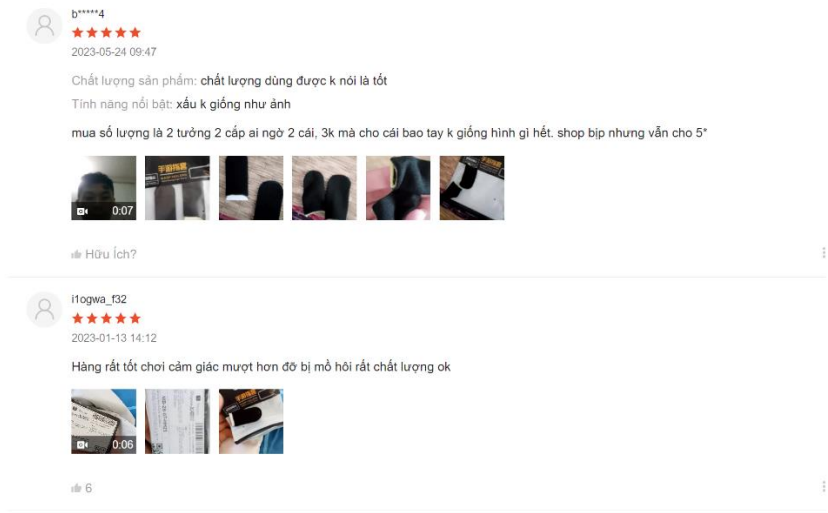
Tuy nhiên, phân loại đánh giá tiêu cực và không tiêu cực của khách hàng là một thách thức đáng kể. Đánh giá có thể đa dạng và mang tính chủ quan, và việc đọc, hiểu và phân tích hàng ngàn bài đánh giá một cách thủ công là một quá trình tốn thời gian và công sức. Đối với với các sàn thương mại điện tử, ví dụ như shopee thì đã có đánh giá theo số sao, nhưng khi đánh giá một sản phẩm shopee sẽ mặc định để số sao là 5/5, nếu người dùng chỉ đưa ra lời bình về sản phẩm mà quên sửa số sao đúng với lời bình của họ thì rất dễ bị sai.



Shopee để mặc định ban đầu là 5 sao

Như một số trường hợp sau:

Trường hợp vote 5 sao nhưng vẫn mang tính tiêu cực



Trường hợp dù tiêu cực nhưng vẫn vote 5 sao

Hiện nay các shop có nhiều đơn hàng lớn thì không thể lọc bằng tay những trường hợp thế này được, mà hầu như họ sẽ cài đặt tự động cảm ơn nếu khách hàng đánh giác ho họ 5 sao cho dù có tiêu cực hay là không. Và ngoài các sàn thương mại điện tử thì mua bán ở trên mạng xã hội như facebook cũng rất nhộn nhịp, nhưng mà ở đây không có tính năng đánh giá theo số sao nên lại càng khó hơn.

Do đó, để giải quyết vấn đề này, cần phát triển một hệ thống tự động phân loại đánh giá tiêu cực và không tiêu cực. Một hệ thống như vậy có khả năng xử lý và phân loại tự động các đánh giá của khách hàng dựa trên nội dung và ngữ cảnh, từ đó đưa ra thông tin hữu ích và phân tích định hướng cho doanh nghiệp.

Việc xây dựng một hệ thống phân loại đánh giá hiệu quả đòi hỏi sự kết hợp giữa các phương pháp xử lý ngôn ngữ tự nhiên (NLP) và các thuật toán học máy. Hệ thống cần

được huấn luyện trên một tập dữ liệu đa dạng và có sự cung cấp đầy đủ các ví dụ về đánh giá tiêu cực và không tiêu cực.

Mục tiêu cuối cùng của việc phân loại đánh giá là giúp cửa hàng/doanh nghiệp hiểu rõ hơn về ý kiến của khách hàng và nhanh chóng đưa ra biện pháp cải thiện nếu cần thiết. Ngoài ra, việc phân loại đánh giá cũng có thể giúp doanh nghiệp định hình hình ảnh công ty và tăng cường sự tin tưởng của khách hàng.

Trong bối cảnh đòi hỏi sự nhanh chóng và hiệu quả, việc phát triển một hệ thống phân loại đánh giá tiêu cực, không tiêu cực của khách hàng là một bước quan trọng để tăng cường sự cạnh tranh và đáp ứng nhu cầu của thị trường ngày càng khắt khe.

2. Phát biểu bài toán

Với những thách thức đã đặt ra ở trên thì nhóm chúng em sẽ sử dụng máy học để giải quyết bài toán phân loại đánh giá của khách hàng. Nhưng phạm trù của vấn đề này quá lớn so với đồ án của một môn học, nên nhóm chúng em sẽ thu hẹp nó lại chỉ còn là giải quyết bài toán "phân loại đánh giá của khách hàng trên Shopee".

Input: Một đánh giá của một khách hàng về một sản phẩm trên Shopee.

Output: Tiêu cực/không tiêu cực (1/0)

Ví dụ:

Input: Sản phẩm tốt

Output: không tiêu cực (0)

Input: Tệ, không có lần thứ 2

Output: tiêu cực (1)

3. Tính ứng dụng

Chúng em hướng tới ứng dụng vào để giải quyết vấn đề nhiều người đánh giá 5 sao nhưng lời bình vẫn mang tính tiêu cực, điều này làm cho người bán hàng khó mà hỗ trợ tới những người đó được, lỡ may người mua hàng thấy lời bình tiêu cực mà shop không xử lý thì họ sẽ giảm tỉ lệ mua hàng. Và với người mua hàng thì có thể dễ dàng xem được

các đánh giá tiêu cực về sản phẩm của khách hàng khác thế nào mà không cần phải ngồi lọc xem các đánh giá.

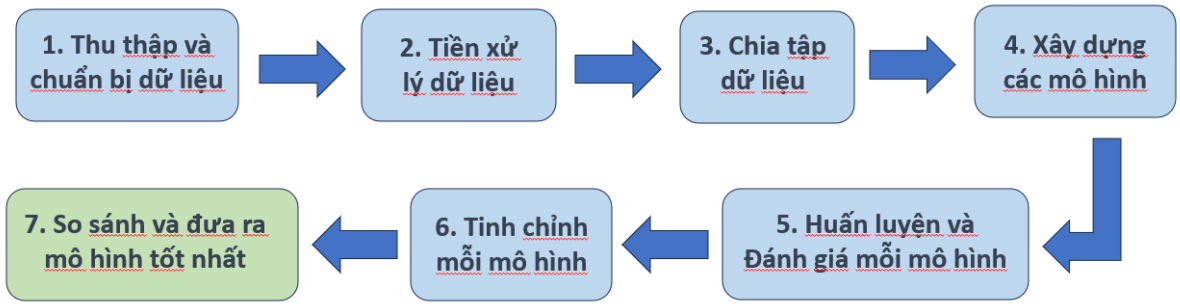
4. Tình hình nghiên cứu hiện nay

Hiện nay đã có các mô hình ngôn ngữ có kết quả tốt dành cho tiếng Việt như là PhoBERT[1] (2020), ViDeBERTa[2] (2023) nên việc chuyển đổi văn bản tiếng Việt sang vector đặc trưng sẽ có kết quả tốt hơn.

Các mô hình máy học phân lớp nổi tiếng như SVM, Logistic Regression, … đã có thư viện cài sẵn giúp cho việc cài đặt tiện lợi và nhanh chóng.

II. Hướng giải pháp

Nhóm chúng em đề xuất giải pháp theo pipeline sau đây:



Được chia thành 2 phần chính:

1. Dataset

   Là 3 bước đầu có chức năng để xây dựng tập dữ liệu bao gồm thu thập và chuẩn bị dữ liệu, tiền xử lý dữ liệu, chia tập dữ liệu.
2. Mô hình máy học

   Là 4 bước còn lại dùng để xây dựng, tinh chỉnh, đánh giá các mô hình từ đó đưa ra được mô hình tốt nhất mà nhóm thực nghiệm được để giải quyết vấn đề

Dưới đây là chi tiết của mỗi phần.

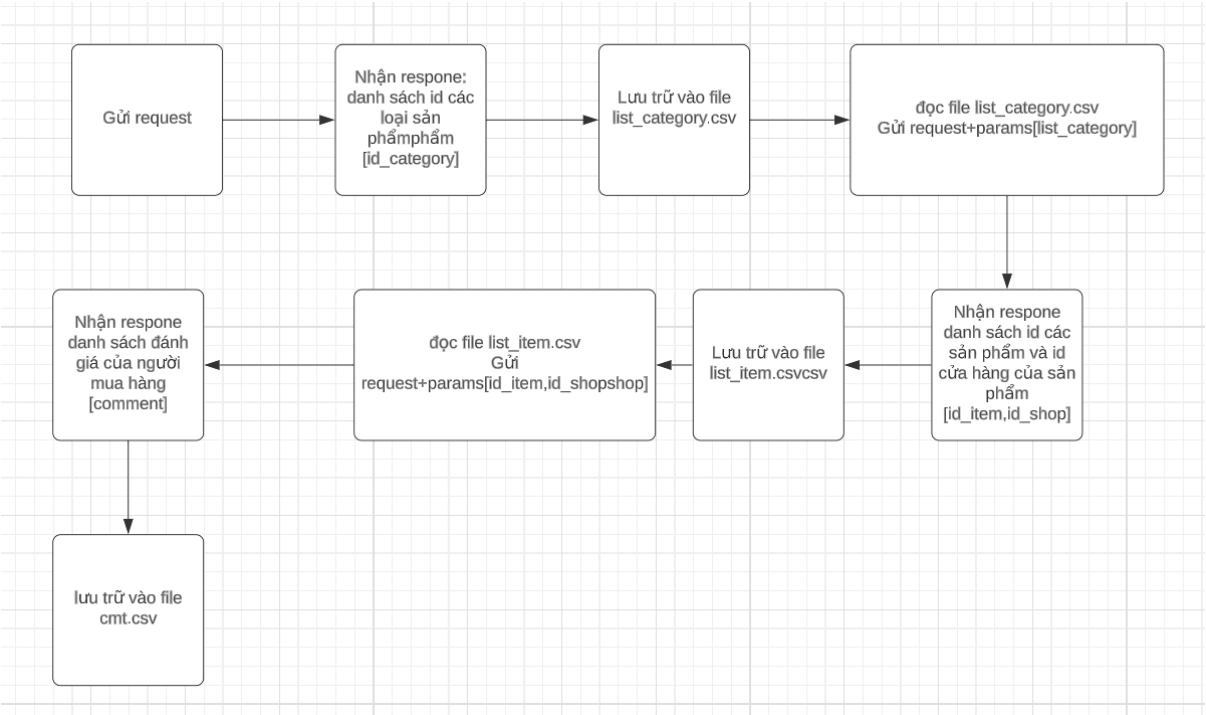
1. Dataset

1.1 Thu thập và chuẩn bị dữ liệu

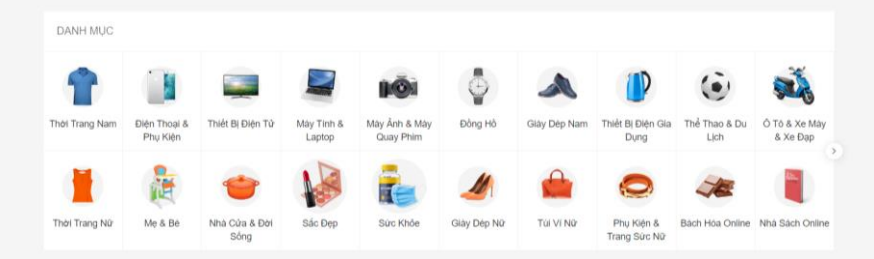
Chúng em sẽ tiến hành crawl dữ liệu (đánh giá của khách hàng về một sản phẩm trên Shopee) theo quy trình sau:

1. Thiết kế và kiến trúc
    - Cấu trúc tổng thể của quá trình crawl dựa trên thư viện "requests" trong Python
    - Một số thành phần chính bao gồm:
        - Mô-đun "requests": Được sử dụng để gửi yêu cầu HTTP đến trang Shopee và nhận phản hồi.
        - Xác định URL: quyết định các URL mục tiêu để crawl dữ liệu đánh giá sản phẩm.
        - Xác thực: nếu cần thiết, đảm báo rằng các yêu cầu được xác thực để truy cập vào các trang yêu cầu đăng nhập hoặc quyền truy cập.
        - Xử lý phân trang: Xác định và xử ký các phân trang để thu thập dữ liệu đánh giá từ nhiều trang.
        - Trích xuất dữ liệu: trích xuất dữ liệu dựa vào phản hồi từ trang shopee dưới dạng JSON.
2. Quá trình crawl đánh giá sản phẩm

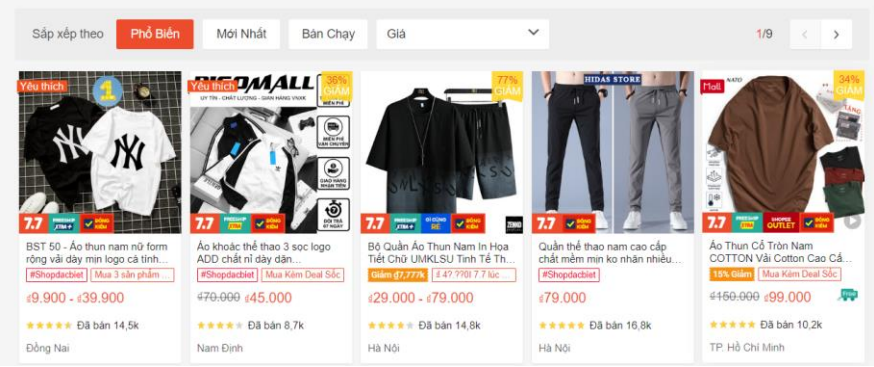
    Sơ đồ quá trình crawl sản phẩm:



- Xác định các URL đích:
  - Danh mục các loại sản phẩm (thời trang nam, thiết bị điện tử, đồ gia dụng,..) : https://shopee.vn/api/v4/pages/get_category_tree

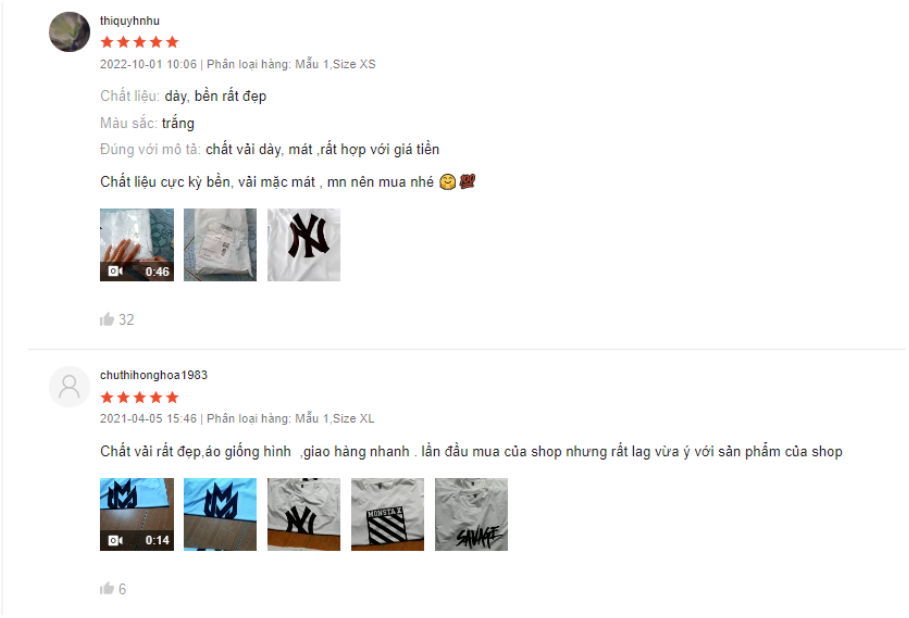


  - Danh sách các sản phẩm: https://shopee.vn/api/v4/recommend/recommend



- Danh sách đánh giá của người mua hàng

  https://shopee.vn/api/v2/item/get_ratings



- Tạo yêu cầu HTTP : sử dụng thư viện “requests” để gửi yêu cầu HTTP đế URL đích và nhận phản hồi.
  - Danh mục các loại sản phẩm:

```
3
4   import csv
5   import requests
6   url = "https://shopee.vn/api/v4/pages/get_category_tree"
7
8   res = requests.get(url)
9
```

  - Danh sách các sản phẩm:

```
url = "https://shopee.vn/api/v4/recommend/recommend"
params = {
    "bundle": "category_landing_page",
    "cat_level": "1",
    "catid": "",
    "limit": "50",
    "offset": "60"
}
df_category=pd.read_csv('category_list.csv')
# print(df_category)
for i in df_category.values:
    params['catid']=str(i[0])


    response = requests.get(url, params=params)
```

- Danh sách đánh giá của người tiêu dùng:

```
url = "https://shopee.vn/api/v2/item/get_ratings"
params = {
    "exclude_filter": "0",
    "filter": "0",
    "filter_size": "0",
    "flag": "1",
    "fold_filter": "0",
    # "itemid": "11504674171",
    "limit": "50",
    "offset": "0",
    "relevant_reviews": "false",
    "request_source": "1",
    # "shopid": "",
    "tag_filter": "",
    #
    "type": "0",
    "variation_filters": "",


}
df_id = pd.read_csv('id_san_pham_7_4_lan1.csv')
print(df_id)
for i in df_id.values :
    print(i)
    params['itemid']= str(i[0])
    params['shopid']= str(i[1])
    response = requests.get(url, params=params)
```

- Trích xuất dữ liệu:
  - Danh mục các loại sản phẩm:

```
res = requests.get(url)


data= res.json()

if res.status_code==200:
    print("gui request thanh cong")
    with open('category_list.csv','w',encoding='utf-8',newline='') as file:
        writer= csv.writer(file)
        for i in data['data']['category_list']:
            category =str(i['catid'])
            writer.writerow([category])
    print("xuat file category list thanh cong")
else:
    print('gui requset that bai')
```

- Danh sách các sản phẩm:

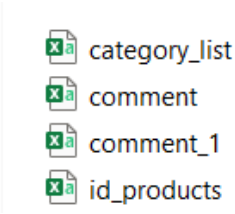
```
response = requests.get(url, params=params)
data = response.json()

if response.status_code == 200:
    # Kiểm tra phản hồi có chứa dữ liệu sản phẩm không
    print('request  thanh cong')
    product_ids=data['data']['sections'][0]['data']['item']
    if product_ids is None:
        print(str(i[0]))
    else:
        with open('id_san_pham.csv', 'a', encoding='utf-8', newline="") as file:
            writer = csv.writer(file)
            # writer.writerow(['id_product'])

            for id in product_ids:
                id_product=id['itemid']
                id_shop=id['shopid']
                writer.writerow([id_product,id_shop])
                # file.write(str(id['itemid']) +'\n')
        print('xuat file id san pham thanh cong')

else:
    print("Yêu cầu không thành công.")
```

- Danh sách đánh giá của người tiêu dùng:

```
response = requests.get(url, params=params)
data = response.json()

if response.status_code == 200:
    cmts= data['data']['ratings']
    if cmts is None:
        print(data)
    else:
        with open('5sao.csv', 'a', encoding='utf-8', newline="") as file:
            writer = csv.writer(file)
            for cmt in cmts:
                comment=str(cmt['comment'])
                writer.writerow([comment])
                # file.write(str('-'+cmt['comment']) +'\n')
        print('xuat file cmts  thanh cong')

else:
    print("Yêu cầu không thành công.")
```

3. Lưu trữ dữ liệu đã crawl được

   Lưu trữ dữ liệu đánh giá sản phẩm trong định dạng file CSV



4. Gán nhãn dữ liệu

   Sau khi đã có được dữ liệu thì chúng em tiến hành gán nhãn dữ liệu, sau đó tiến hành kiểm tra chéo. Do nhóm có ít thành viên (2 thành viên) nên chúng em gán nhãn đến khi nhóm cảm thấy đủ nhiều thì dừng. Kết quả là gán nhắn được 2435 đánh giá, trong đó có 1381 đánh giá tiêu cực, 1054 là không tiêu cực.

1.2 Tiền xử lý dữ liệu

1.2.1 Chuẩn hóa dữ liệu

- Nếu có nhiều dòng thì rút gọn hết thành 1 dòng
- Thay thế các icon biểu tượng cảm xúc thành văn bản.

  + Hàng phía trên là các icon biểu hiện tính tiêu cực, ta có thể thay thế chúng thành “sản phẩm tệ kém chất lượng”.
  + Hàng phía dưới là các icon biểu hiện tính tích cực, ta có thể thay thế chúng thành “sản phẩm tốt chất lượng tốt”.
- Xóa hết những ký tự không phải là chữ và số
- Xóa hết các khoảng trắng thừa
- Đưa hết về chữ thường
- Với 2 ký tự cạnh nhau mà giống nhau thì xóa bớt 1 ký tự

1.2.2 Tách từ và loại bỏ stop word nếu cần

1. Tách từ

- Underthesea: Underthesea là một thư viện xử lý ngôn ngữ tự nhiên (NLP) mã nguồn mở cho tiếng Việt. Nó cung cấp các chức năng tiền xử lý và xử lý ngôn ngữ tự nhiên cho tiếng Việt, bao gồm tách từ, gán nhãn từ loại, phân tích cú pháp, và các tính năng khác.Underthesea được phát triển với mục tiêu hỗ trợ xử lý ngôn ngữ tự nhiên cho tiếng Việt. Thư viện này cung cấp các chức năng tiền xử lý và xử lý ngôn ngữ tự nhiên như tách từ, gán nhãn từ loại, phân tích cú pháp, trích xuất thông tin, và nhiều tính năng khác để giúp phân tích và xử lý văn bản tiếng Việt.
- ViTokenizer: ViTokenizer là một công cụ phân tách từ tiếng Việt được phát triển bởi học viện Khoa học và Công nghệ Việt Nam. Nó sử dụng một số quy tắc ngữ pháp và từ điển để tách từ trong câu tiếng Việt. Công cụ này tập trung vào việc phân tách từ và có khả năng xử lý với hiệu suất cao. ViTokenizer cung cấp API dễ sử dụng và có sẵn như một thư viện Python.

Nhóm chúng em sẽ sử dụng Underthesea và ViTokenizer để tách từ xem cái nào tốt hơn thì chọn, sau đó hợp lại ví dụ như sau:

> *Đoạn văn ban đầu*: "Thủ tướng Đức nhận lời tham dự lễ kỷ niệm D-Day Thủ tướng Gerhard Schroeder sẽ trở thành nguyên thủ Đức đầu tiên tham dự lễ kỷ niệm ngày quân đồng minh đổ bộ lên bãi biển Normandy trong Thế chiến II (mang mật danh D-Day) vào tháng 6 tới."
>
> *Đoạn văn sau khi xử lý*: "thủ_tướng đức nhận_lời tham_dự lễ kỷ_niệm day thủ_tướng gerhard schroeder sẽ trở_thành nguyên_thủ đức đầu_tiên tham_dự lễ kỷ_niệm ngày quân đồng_minh đổ_bộ lên bãi biển normandy trong thế_chiến ii mang mật_danh day vào tháng tới"

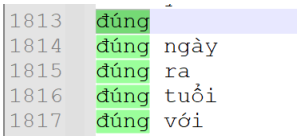
Lúc này những từ mà được ghép từ 1 hoặc nhiều chữ sẽ được ghép lại với nhau và dễ nhận biết được nghĩa. Ví dụ như từ "mật danh", nếu tách ra thành "mật" và "danh" thì 2 từ này nghĩa nó khác hoàn toàn với nghĩa gốc.

2. Loại bỏ stop word

Sử dụng bộ dữ liệu vietnamese-stopwords của:

https://github.com/stopwords/vietnamese-stopwords

Bộ dữ liệu này có 1942 từ/cụm từ. Khi sử dụng bộ stop word này cần tinh chỉnh lại và xóa bớt những từ/cụm từ có thể ảnh hưởng tới kết quả. Ví dụ:



Với từ đúng thì nếu ta cho nó vào stop word thì sẽ bỏ qua thông tin cực kỳ quan trọng. Vì khách thường khi đánh giá thường sẽ đánh giá như là "Sản phẩm đúng", "đúng hàng",…

1.2.3 Vector hóa dữ liệu văn bản

Sau khi đã làm các bước chuẩn hóa, tách từ, … trước đó thì giờ ta đã có dữ liệu "sạch" hơn, nhưng mà chưa thể tiến thành huấn luyện mô hình máy học bởi vì dữ liệu của ta hiện tại là dạng văn bản, còn các mô hình máy học thì yêu cầu input là dạng số. Vì thế ta cần chuyển văn bản thành vector đặc trưng, từ đó mới có thể sử dụng các mô hình máy học được.

Dưới đây là 2 mô hình xử lý ngôn ngữ tự nhiên được nhóm em tìm hiểu và sử dụng để chuyển văn bản thành vector đặc trưng:

1. PhoBERT[1]

- PhoBERT là một mô hình ngôn ngữ dựa trên kiến trúc BERT (Bidirectional Encoder Representations from Transformers) được huấn luyện trên dữ liệu tiếng Việt. Tên "PhoBERT" kết hợp từ "Phở" (một món ăn truyền thống của Việt Nam) và "BERT". Mô hình này được huấn luyện để hiểu và đại diện cho ngôn ngữ tiếng Việt trong các tác vụ xử lý ngôn ngữ tự nhiên, như phân loại văn bản, phân tích cảm xúc, dịch máy và nhiều ứng dụng khác.
- PhoBERT được công bố tại *Findings of the Association for Computational Linguistics: EMNLP 2020*
- PhoBERT sử dụng kiến trúc transformer để xử lý văn bản và học các biểu diễn ngữ nghĩa của các từ và câu. Nó sử dụng các phép biến đổi đa tầng và tự học thông qua quá trình huấn luyện mạng thần kinh sâu trên dữ liệu lớn. Việc sử dụng PhoBERT cho xử lý ngôn ngữ tiếng Việt đã giúp nâng cao chất lượng và

hiệu suất của nhiều tác vụ liên quan đến ngôn ngữ trong lĩnh vực NLP (Natural Language Processing).

- Nhóm tác giả đã giới PhoBERT với hai phiên bản, PhoBERT-base và PhoBERT-large. Ngoài ra còn có PhoBERT-base-v2. Nhóm em sẽ thực nghiệm cả 3 phiên bản này

Pre-trained models

Model	#params	Arch.	Max length	Pre-training data
`vinai/phobert-base`	135M	base	256	20GB of Wikipedia and News texts
`vinai/phobert-large`	370M	large	256	20GB of Wikipedia and News texts
`vinai/phobert-base-v2`	135M	base	256	20GB of Wikipedia and News texts + 120GB of texts from OSCAR-2301

- Cài đặt:

```
model_trans = AutoModel.from_pretrained("vinai/phobert-base")
tokenizer = AutoTokenizer.from_pretrained("vinai/phobert-base")
```

2. ViDeBERTa[2]

- ViDeBERTa là một mô hình ngôn ngữ đơn ngữ pre-trained mới dành cho tiếng Việt, với ba phiên bản - ViDeBERTaxsmall, ViDeBERTabase và ViDeBERTalarge, được đào tạo trước trên một kho ngữ liệu quy mô lớn gồm các văn bản tiếng Việt đa dạng và chất lượng cao sử dụng kiến trúc DeBERTa.

Model	#layers	#heads	hidden size
ViDeBERTa$_{xsmall}$	6	12	768
ViDeBERTa$_{base}$	12	12	768
ViDeBERTa$_{large}$	24	12	1024

- Mặc dù nhiều mô hình ngôn ngữ tiền huấn luyện thành công dựa trên Transformer đã được đề xuất rộng rãi cho ngôn ngữ tiếng Anh, nhưng vẫn còn rất ít mô hình tiền huấn luyện cho tiếng Việt, một ngôn ngữ ít tài nguyên, thực hiện tốt các nhiệm vụ tiếp theo, đặc biệt là trả lời câu hỏi. Nhóm tác giả đã tinh chỉnh và đánh giá mô hình của họ trên ba tác vụ quan trọng ở downstream ngôn

ngữ tự nhiên, Part-ofspeech tagging, Named-entity recognition, and Question answering.

- Các kết quả thực nghiệm chứng minh rằng ViDeBERTa với ít tham số hơn nhiều vượt qua các mô hình hiện đại trước đây về nhiều nhiệm vụ hiểu ngôn ngữ tự nhiên dành riêng cho tiếng Việt. Đáng chú ý, ViDeBERTabase với 86M tham số, chỉ bằng khoảng 23% của PhoBERTlarge với 370M tham số, vẫn cho kết quả tương đương hoặc tốt hơn so với mô hình hiện đại trước đó.
- ViDeBERTa được công bố tại *Findings of the Association for Computational Linguistics: EACL 2023*.
- Cài đặt:

```
model_trans = AutoModel.from_pretrained("Fsoft-AIC/videberta-base")
tokenizer = AutoTokenizer.from_pretrained("Fsoft-AIC/videberta-base")
```

1.3 Chia tập dữ liệu

Với dữ liệu ban đầu chúng em sử dụng train test split để chia dữ liệu. Tỷ lệ là 80% dùng để train, 20% dùng để test. Sau khi chia dữ liệu ra thành 2 tập train và test thì chúng em mới tiến hành xử lý dữ liệu của từng tập riêng biệt, điều này để tránh tối đa khả năng bị lộ tập test dẫn tới kết quả phản ánh không chính xác khi đánh giá.

2. Mô hình máy học

2.1 Xây dựng các mô hình máy học

2.1.1 Support vector machine (SVM)

- Support Vector Machine (SVM) là một thuật toán học máy được sử dụng trong bài toán phân loại và hồi quy. Nó được phát triển bởi Vapnik và đồng nghiệp của ông vào những năm 1990 và đã trở thành một trong những phương pháp phân loại phổ biến và mạnh mẽ.
- Ý tưởng cơ bản của SVM là xây dựng một siêu mặt phẳng trong không gian dữ liệu, tối đa hóa khoảng cách từ siêu mặt phẳng đến các điểm dữ liệu gần nhất của các lớp khác nhau. Các điểm dữ liệu gần nhất này được gọi là các vector hỗ trợ

(support vectors). SVM cố gắng tìm một siêu mặt phẳng tối ưu để phân chia các điểm dữ liệu của các lớp khác nhau sao cho khoảng cách giữa các lớp là lớn nhất.

- Cách thức hoạt động của SVM là tìm kiếm một siêu mặt phẳng phân chia tốt nhất các điểm dữ liệu của các lớp khác nhau bằng cách giải một bài toán tối ưu hóa. Cụ thể, SVM cố gắng tìm một đường biên phân chia sao cho tổng các lỗi phân loại (khoảng cách từ các điểm dữ liệu đến đường biên) là nhỏ nhất. Đường biên này được chọn sao cho nằm giữa hai vector hỗ trợ gần nhất và cách xa nhất các điểm dữ liệu.
- SVM có thể sử dụng các hàm nhân (kernel function) để biến đổi dữ liệu vào một không gian cao chiều hơn, từ đó tạo ra các đường biên phân chia phi tuyến tính. Các hàm nhân phổ biến trong SVM bao gồm hàm tuyến tính, hàm đa thức, hàm Radial Basis Function (RBF) và hàm Sigmoid. Các hàm nhân cho phép SVM xử lý hiệu quả các bài toán phân loại phi tuyến tính.
- Ưu điểm của SVM bao gồm:
  + Hiệu suất tốt: SVM thường mang lại hiệu suất tốt trong việc phân loại các tập dữ liệu lớn và phức tạp.
  + Tính tổng quát cao: SVM giúp giảm nguy cơ overfitting và có khả năng tổng quát hóa tốt trên dữ liệu mới.
  + Hỗ trợ vector: SVM chỉ sử dụng một số lượng nhỏ các vector hỗ trợ để xác định đường biên phân chia, do đó, nó tiêu thụ ít bộ nhớ hơn so với các phương pháp khác.
  + Khả năng xử lý dữ liệu phi tuyến: SVM có thể sử dụng các hàm nhân để xử lý dữ liệu phi tuyến, mở rộng khả năng phân loại của nó.
- Tuy nhiên, SVM cũng có một số hạn chế, bao gồm khả năng xử lý tốn kém với các tập dữ liệu rất lớn, yêu cầu quá trình chuẩn bị dữ liệu kỹ càng và độ phức tạp tính toán cao đối với các bài toán đa lớp.
- Cài đặt:

```
from sklearn import svm

# Tạo mô hình SVM
model_svm = svm.SVC()
model_svm.fit(X_train, y_train)

# Dự đoán trên tập test
y_pred_svm = model_svm.predict(X_test)

# Đánh giá mô hình
f1 = f1_score(y_test, y_pred_svm)
```

2.1.2 Logistic Regression

- Logistic Regression (Hồi quy Logistic) là một thuật toán học máy được sử dụng phổ biến trong bài toán phân loại. Mặc dù có tên là "regression", nhưng logistic regression thực chất là một thuật toán phân loại và không phải là một thuật toán hồi quy truyền thống.
- Ý tưởng cơ bản của logistic regression là sử dụng hàm logistic (hay còn gọi là sigmoid) để tạo ra một đường cong S dựa trên các biến đầu vào. Hàm sigmoid có dạng S-shaped và có giá trị trong khoảng từ 0 đến 1. Đường cong sigmoid được sử dụng để ước lượng xác suất rơi vào một lớp nhất định.
- Trong quá trình huấn luyện, logistic regression tìm cách tối ưu hóa các tham số sao cho đường cong sigmoid phù hợp nhất với dữ liệu huấn luyện. Để làm điều này, thuật toán sử dụng phương pháp tối ưu hóa như Gradient Descent hoặc Newton's Method để điều chỉnh các tham số. Mục tiêu cuối cùng là tìm ra một mô hình logistic regression có khả năng phân loại tốt và dự đoán xác suất rơi vào mỗi lớp.
- Logistic regression có một số ưu điểm quan trọng:
  + Đơn giản và dễ hiểu: Logistic regression là một thuật toán đơn giản và dễ hiểu, không đòi hỏi nhiều thông số cần điều chỉnh.
  + Tính tường minh của kết quả: Logistic regression cung cấp thông tin về xác suất và đường cong sigmoid, giúp hiểu rõ hơn về quan hệ giữa biến đầu vào và đầu ra.

+ Tính tương thích với các biến đầu vào liên tục và rời rạc: Logistic regression có thể xử lý cả các biến đầu vào liên tục và rời rạc, và có thể mở rộng để xử lý nhiều biến đầu vào.
+ Hiệu suất tốt trong các vấn đề phân loại nhị phân: Logistic regression thường mang lại hiệu suất tốt trong các bài toán phân loại nhị phân, đặc biệt là khi dữ liệu tương đối tuyến tính và có ít biến đầu vào.

- Tuy nhiên, logistic regression cũng có một số hạn chế:
  + Khả năng mô hình hóa phụ thuộc tuyến tính: Logistic regression chỉ phù hợp với các mô hình có mối quan hệ tuyến tính giữa biến đầu vào và xác suất phân loại. Nếu mối quan hệ này không tuyến tính, logistic regression có thể không thể mô hình hóa tốt.
  + Nhạy cảm với nhiễu và giá trị ngoại lai: Logistic regression có thể bị ảnh hưởng bởi nhiễu và giá trị ngoại lai trong dữ liệu, gây ra các ảnh hưởng không mong muốn đến mô hình.
  + Giới hạn trong bài toán phân loại đa lớp: Logistic regression gốc chỉ hỗ trợ phân loại nhị phân, nhưng có thể được mở rộng để xử lý bài toán phân loại đa lớp bằng các kỹ thuật như One-vs-Rest hoặc Softmax regression.
- Cài đặt:

```
from sklearn.linear_model import LogisticRegression

# Tạo mô hình Logistic Regression
model_lr = LogisticRegression()
model_lr.fit(X_train, y_train)

# Dự đoán trên tập test
y_pred_lr = model_lr.predict(X_test)

# Đánh giá mô hình
f1 = f1_score(y_test, y_pred_lr)
```

2.1.3 Random Forest

- Random Forest (Rừng ngẫu nhiên) là một thuật toán học máy được sử dụng phổ biến trong bài toán phân loại, hồi quy và nhận dạng mẫu. Nó được xây dựng trên ý tưởng của Ensemble Learning, tức là kết hợp dự đoán của nhiều cây quyết định (Decision Trees) để tạo ra một dự đoán cuối cùng.

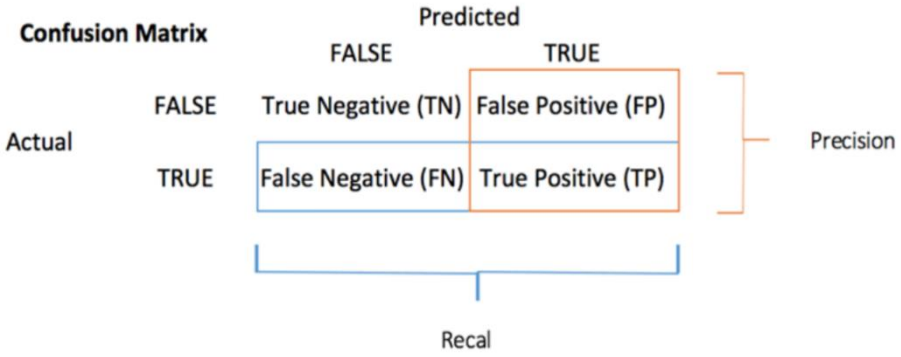
- Ý tưởng cơ bản của Random Forest là xây dựng một tập hợp các cây quyết định độc lập và đa dạng. Mỗi cây quyết định được huấn luyện trên một tập dữ liệu con được lấy ngẫu nhiên từ tập dữ liệu huấn luyện ban đầu thông qua quá trình Bootstrap Aggregating (hoặc Bagging). Quá trình này cho phép mỗi cây quyết định có một tập dữ liệu con riêng, đồng thời giảm hiện tượng overfitting.
- Khi thực hiện dự đoán, Random Forest sử dụng phương pháp đa số phiếu bầu (voting) hoặc trung bình các dự đoán từ các cây quyết định thành viên. Trong trường hợp phân loại, lớp được chọn là lớp xuất hiện nhiều nhất trong các dự đoán của các cây thành viên. Trong trường hợp hồi quy, kết quả cuối cùng được tính bằng trung bình của các dự đoán từ các cây thành viên.
- Random Forest có một số ưu điểm quan trọng:
  + Độ chính xác cao: Random Forest thường mang lại độ chính xác cao trong việc phân loại và hồi quy. Việc kết hợp nhiều cây quyết định giúp giảm hiện tượng overfitting và tạo ra dự đoán ổn định.
  + Khả năng xử lý các biến đầu vào rời rạc và liên tục: Random Forest có thể xử lý cả các biến đầu vào rời rạc và liên tục mà không yêu cầu quá trình chuẩn bị dữ liệu phức tạp.
  + Khả năng xử lý dữ liệu thiếu: Random Forest có khả năng xử lý tốt dữ liệu có giá trị thiếu mà không cần phải điền giá trị thay thế.
  + Độ tin cậy và giải thích: Random Forest cung cấp độ tin cậy của dự đoán và có khả năng xếp hạng quan trọng của các biến đầu vào, giúp hiểu rõ hơn về quan hệ giữa biến đầu vào và đầu ra.
- Tuy nhiên, Random Forest cũng có một số hạn chế:
  + Phức tạp tính toán: Random Forest có thể tốn nhiều thời gian và tài nguyên tính toán khi có một số lượng lớn cây quyết định.
  + Khó để giải thích các quyết định riêng lẻ: Với một Random Forest lớn, việc giải thích một quyết định cụ thể của cây quyết định trở nên khó khăn.
  + Thông số tùy chỉnh: Random Forest có một số thông số tùy chỉnh như số lượng cây, độ sâu cây, số lượng biến đầu vào được chọn ngẫu nhiên, v.v. Điều này yêu cầu tinh chỉnh thích hợp để đạt hiệu suất tốt nhất.
- Cài đặt:

```python
from sklearn.ensemble import RandomForestClassifier

# Tạo mô hình Random Forest
model_rf = RandomForestClassifier()
model_rf.fit(X_train, y_train)

# Dự đoán trên tập test
y_pred_rf = model_rf.predict(X_test)

# Đánh giá mô hình
f1 = f1_score(y_test, y_pred_rf)
```

2.2 Huấn luyện và đánh giá mô hình

- Sau khi đã xây dựng được mô hình thì ta tiến hành huấn luyện dựa trên tập train.
- Huấn luyện mô hình xong ta tiến hành đánh giá: Để có thể đánh giá được thì chúng ta cần phải có độ đo. Ta có thể dung thang đo Accuracy, Recall, Precision, F1 score để đánh giá với bài toán này.

  => Để có thể đánh giá tổng thể mô hình và đảm bảo tính cân bằng giữa các lớp thì nhóm em quyết định **chọn F1 score làm độ đo.** Công thức:



$$\text{Accuracy} = \frac{TN+TP}{TN+FP+FN+TP}$$

$$\text{Recall} = \frac{TP}{TP+FN}$$

$$\text{Precision} = \frac{TP}{TP+FP}$$

$$\text{F1} = \frac{2 * Recall * Precision}{Recall+Precision}$$

2.3 Tinh chỉnh mô hình

Chúng em sử dụng Grid Search để tinh chỉnh mô hình.

- Với SVC chúng em sử dụng bộ params sau:

```
parameters = {
                'kernel': ('linear', 'rbf'),
                'C': [0.5, 1, 2, 4],
                'gamma': ['scale', 0.125, 0.25, 0.5, 1, 2, 4]
            }
```

- Với Logistic Regression chúng em sử dụng bộ params sau:

```
param_grid = {
    'C': [0.1, 1, 10],
    'penalty': ['l1', 'l2'],
    'solver': ['lbfgs', 'liblinear', 'saga']
}
```

- Với RandomForest chúng em sử dụng bộ params sau:

```
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 5, 10],
    'min_samples_split': [2, 5, 10]
}
```

Chúng em chọn Grid Search bởi vì nhận thấy thời gian chạy mô hình khá nhanh, điều này làm khuyết điểm về mặt tài nguyên của Grid Search không quá quan trọng. Và có các lợi ích như sau:

+ Tìm kiếm quét không gian siêu tham số lớn: Grid Search cho phép ta tìm kiếm qua một tập hợp các giá trị tham số được định trước cho mô hình. Bằng cách xác định các giá trị tham số và phạm vi của chúng, ta có thể khám phá một không gian siêu tham số lớn hoặc toàn bộ của mô hình. Điều này giúp cho chúng ta khó có khả năng bỏ sót bất kỳ cấu hình tham số tốt nào.
+ Giảm thời gian và công sức: Grid Search tự động thực hiện việc lặp lại huấn luyện và đánh giá mô hình với các giá trị tham số khác nhau. Điều này giúp giảm thời gian và công sức so với việc thủ công tinh chỉnh các tham số mô hình một cách tuần tự. Bạn không cần phải thử nghiệm từng cấu hình tham số một cách độc lập, mà chỉ cần định nghĩa các giá trị tham số và Grid Search sẽ tự động thực hiện việc tìm kiếm và tinh chỉnh.

2.4 So sánh và đưa ra mô hình tốt nhất

Với mỗi mô hình sau khi đã tinh chỉnh với bộ siêu tham số tốt nhất tìm được bởi Grid Search thì tiến hành đánh giá với bộ dữ liệu test. Từ đó có thể so sánh dựa trên F1 score và đưa ra được mô hình tốt nhất. Sau đó ta model xuống để sử dụng (ví dụ cho demo):

```
# Save model
dump(svm_model, 'save_model.joblib')
```

III. Thực nghiệm và đánh giá

Nhóm chúng em đã thực nghiệm trên các phiên bản của PhoBert và videberta để chuyển văn bản thành dạng vector đặc trưng, sau đó dùng 3 model là SVM, LogisticRegression, RandomForest để tiến hành phân lớp và thu được kết quả trên thang đo F1 score như sau:

	SVM	LogisticRegression	RandomForestClassifier
PhoBert-base	0.9257	0.9263	0.9100
PhoBert-base-v2	0.9437	0.9476	0.9131
PhoBert-large	0.9136	0.9171	0.8703
videberta-base	0.7334	0.7334	0.7334
videberta-xsmall	0.7334	0.7334	0.7334

Nhận xét:

- Sử dụng PhoBert-base-v2 và LogisticRegression cho kết quả tốt nhất 94.76%. Với kết quả này mô hình cho ra kết quả tốt.
- Sử dụng mô hình ngôn ngữ PhoBert cho kết quả tốt hơn videberta, gần như mọi trường hợp tốt hơn khoảng 20%. Videberta tệ hơn có thể là do số tham số ít hơn so với PhoBert (ViDeBERTabase với 86M tham số, chỉ bằng khoảng 23% của PhoBERTlarge với 370M tham số[2])

IV. Kết luận và hướng phát triển

1. Đánh giá, nhận xét, khó khăn gặp phải

- Kết quả đạt được với model tốt nhất có F1 score là 94.76% có kết quả tốt. Đủ cao để có thể áp dụng vào một số các ứng dụng, như là "Phát hiện đánh giá 5* nhưng vẫn mang tính tiêu cực". Từ đó giúp cho nhà bán hàng có thể chăm sóc

khách hàng tốt hơn, người mua hàng có thể có cái nhìn chính xác hơn về các đánh giá của sản phẩm.

- Các khó khăn khi thực hiện đề tài:
  + Quá trình gán nhãn dữ liệu đòi hỏi phải kiểm tra chéo nhưng nhóm chỉ có 2 thành viên nên mỗi người phải kiểm tra lại nhiều lần tốn nhiều thời gian.
  + Quá trình tiền xử lý tốn nhiều thời gian, nguyên nhân là do phải thử thử nghiệm rất nhiều lần thì mới đưa ra được cách tiền xử lý tối ưu.
  + Chưa dùng thư viện transformer nhiều nên gặp khó khăn trong việc sử dụng videberta.
  + Nhóm ít thành viên nên phải làm khối lượng công việc khá nhiều.

2. Hướng phản triển từ bài toán trong tương lai

Ngoài áp dụng vào Shopee thì có thể phát triển sang phân loại bình luận tiêu cực hoặc không tiêu cực trên các nền tảng khác như Lazada, Tiki hoặc trên các trang mạng xã hội như Facebook.

Bảng phân công công việc

Thành viên	MSSV	Nội dung tìm hiểu	Đánh giá
Nguyễn Quốc Huy Hoàng	20520051	Model	100%
Ngô Quang Vinh	19522523	Data	100%

Tài liệu tham khảo

[1] Dat Quoc Nguyen and Anh Tuan Nguyen. 2020. PhoBERT: Pre-trained language models for Vietnamese. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1037–1042, Online. Association for Computational Linguistics.

[2] Cong Dao Tran, Nhut Huy Pham, Anh Tuan Nguyen, Truong Son Hy, and Tu Vu. 2023. ViDeBERTa: A powerful pre-trained language model for Vietnamese. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 1071–1078, Dubrovnik, Croatia. Association for Computational Linguistics.