TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA TOÁN - TIN HỌC

PYTHON CHO KHOA HỌC DỮ LIỆU

ĐỒ ÁN CUỐI KÌ
SUMMARIZING CONSERVATIONS

Giảng viên phụ trách
Thầy Hà Văn Thảo

Tp. Hồ Chí Minh, tháng 12, năm 2024

Mục lục

Chương 1

Giới thiệu

1.1 Thông tin nhóm

- 22110215 - Phạm Thị Anh Thư
- 22110177 - Phạm Đăng Quang
- 22110204 - Nguyễn Thiện Thanh
- 22110210 - Võ Xuân Thiện
- 22110208 - Nguyễn Ngọc Thiện

1.2 Mô hình ngôn ngữ lớn

Ngày 30 tháng 11 năm 2022 đánh dấu một chương quan trọng trong lịch sử của học máy. Đó là ngày OpenAI phát hành ChatGPT, thiết lập một tiêu chuẩn mới cho chatbot được hỗ trợ bởi các Mô hình Ngôn ngữ Lớn và mang đến cho công chúng một trải nghiệm trò chuyện chưa từng có.

Kể từ đó, các mô hình ngôn ngữ lớn - còn được gọi là LLM - đã được công chúng chú ý do số lượng lớn các tác vụ mà chúng có thể thực hiện. Ví dụ bao gồm:

- Tóm tắt văn bản: Các mô hình này có thể thực hiện việc tóm tắt các văn bản lớn, bao gồm văn bản pháp lý, đánh giá, hội thoại và nhiều văn bản khác.
- Phân tích cảm xúc: Chúng có thể đọc qua các đánh giá về sản phẩm và dịch vụ và phân loại chúng là tích cực, tiêu cực hoặc trung lập. Chúng cũng có thể được sử dụng trong Tài chính để xem liệu công

chúng nói chung cảm thấy lạc quan hay bi quan về một số chứng khoán nhất định.

- Dịch ngôn ngữ: Chúng có thể cung cấp bản dịch theo thời gian thực từ ngôn ngữ này sang ngôn ngữ khác.

- Hệ thống đề xuất dựa trên văn bản: Chúng cũng có thể đề xuất các sản phẩm mới cho khách hàng dựa trên đánh giá của họ về các sản phẩm đã mua trước đó.

1.3 The Transformer Architecture

Để hiểu được trạng thái hiện tại của các LLM(mô hình ngôn ngữ lớn), chúng ta phải quay trở lại bài báo "Attention is All You Need" của Google vào năm 2017. Trong bài báo này, kiến trúc Transformer đã được giới thiệu với thế giới, và nó đã thay đổi ngành công nghiệp mãi mãi.

Mặc dù mạng nơ-ron nhân tạo (neural networks) có thể được sử dụng để cho phép máy tính hiểu văn bản, nhưng các mô hình này cực kỳ hạn chế do thực tế là chúng chỉ cho phép máy xử lý từng từ một, dẫn đến việc mô hình không thể nắm bắt được toàn bộ ngữ cảnh của văn bản.

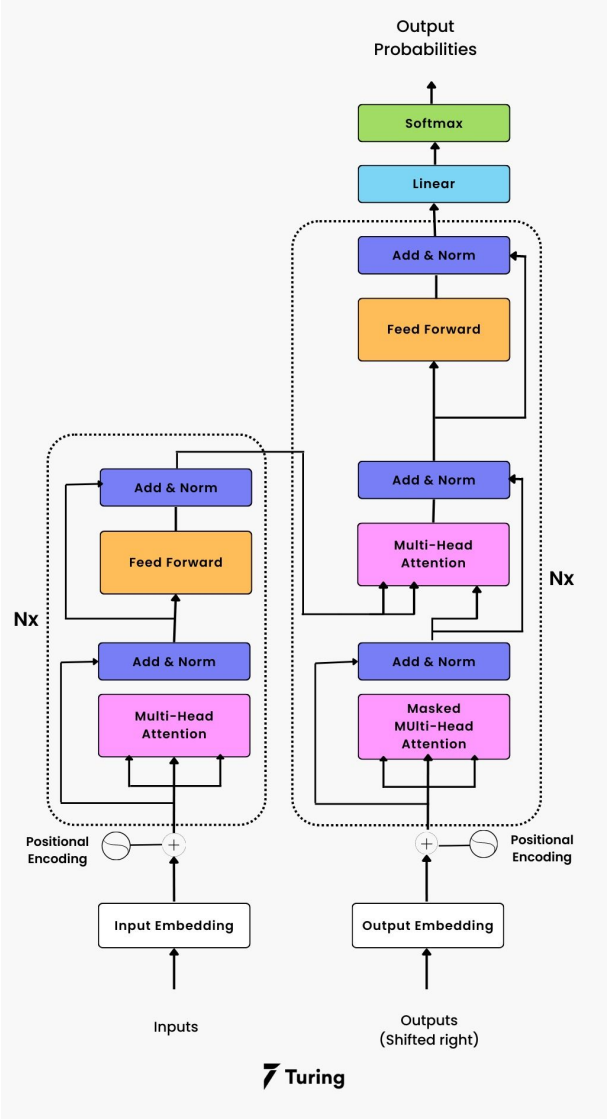
Tuy nhiên, kiến trúc Transformer dựa trên cơ chế chú ý (attention mechanism), cho phép mô hình xử lý toàn bộ câu hoặc đoạn văn cùng một lúc, thay vì từng từ một. Đây là bí mật chính đằng sau khả năng hiểu ngữ cảnh đầy đủ, mang lại sức mạnh lớn hơn cho tất cả các mô hình xử lý ngôn ngữ này.

Việc xử lý đầu vào văn bản với kiến trúc Transformer dựa trên tokenization, là quá trình chuyển đổi văn bản thành các thành phần nhỏ hơn được gọi là token. Đây có thể là từ, từ phụ, ký tự hoặc nhiều thành phần khác.

Các token sau đó được ánh xạ tới các ID số, là duy nhất cho mỗi từ hoặc từ phụ. Mỗi ID sau đó được chuyển đổi thành một embedding: một vectơ dày đặc, nhiều chiều chứa các giá trị số. Các giá trị này được thiết kế để nắm bắt ý nghĩa ban đầu của các token và đóng vai trò là đầu vào cho mô hình Transformer.

Điều quan trọng cần lưu ý là các embedding này có nhiều chiều, với mỗi chiều nắm bắt các khía cạnh nhất định về ý nghĩa của một token. Do bản chất nhiều chiều của chúng, embedding không dễ dàng được con người giải thích, nhưng các mô hình Transformer dễ dàng sử dụng chúng để xác định và nhóm các token có ý nghĩa tương tự trong không gian vectơ.

Bằng cách sử dụng vector này làm đầu vào, mô hình Transformer học cách tạo ra đầu ra dựa trên xác suất của các từ tiếp theo có thể xuất hiện một cách tự nhiên sau một từ đầu vào. Quá trình này được lặp lại cho đến khi mô hình tạo ra toàn bộ đoạn văn bắt đầu từ một câu lệnh ban đầu.



Hình 1.1: Kiến trúc Transformer

1.4 Mục tiêu

Mục tiêu của notebook này là để chứng minh cách các Mô hình Ngôn ngữ Lớn (Large Language Models - LLMs) có thể được sử dụng cho một số tác vụ liên quan đến xử lý ngôn ngữ (language processing). Trong trường hợp này, tôi sẽ tận dụng sức mạnh của học chuyển giao (transfer learning) để xây dựng một mô hình có khả năng tóm tắt các cuộc hội thoại (summarizing dialogues).

Nếu các bạn có thể chưa biết, transfer learning là một kỹ thuật học máy (machine learning technique) trong đó chúng ta sử dụng một mô hình được đào tạo trước (pre-trained model) - vốn đã có kiến thức trong một lĩnh vực rộng - và điều chỉnh chuyên môn của nó cho một tác vụ cụ thể (specific task) bằng cách huấn luyện nó trong một tập dữ liệu cụ thể (specific dataset) mà chúng ta có thể có. Quá trình này cũng có thể được gọi là tinh chỉnh (fine-tuning).

Thư viện Transformers - một trong những thư viện phổ biến nhất để làm việc với các tác vụ học sâu (deep learning tasks) - cung cấp khả năng làm việc với các kiến trúc (architectures) sau:

Model Architectures
BART, BigBird-Pegasus, Blenderbot, BlenderbotSmall, Encoder decoder, FairSeq Machine-Translation, GPTSAN-japanese, LED, LongT5, M2M100, Marian, mBART, MT5, MVP, NLLB, NLLB-MOE, Pegasus, PEGASUS-X, PLBart, ProphetNet, SwitchTransformers, T5, UMT5, XLM-ProphetNet

Hình 1.2: Các kiến trúc mô hình mà thư viện Transformers hỗ trợ

Thư viện Transformers cho phép chúng ta dễ dàng tải xuống và tinh chỉnh các mô hình được đào tạo trước tiên tiến, đồng thời cho phép chúng ta dễ dàng làm việc với cả TensorFlow và PyTorch cho một số tác vụ liên quan đến Xử lý Ngôn ngữ Tự nhiên (Natural Language Preprocessing -

NLP), Thị giác Máy tính (Computer Vision), Âm thanh (Audio), v.v.

Chương 2

Cài đặt

2.1 Data source

Theo tài liệu của thư viện Transformers, tóm tắt có thể được mô tả là việc tạo ra một phiên bản ngắn hơn của một tài liệu hoặc một bài báo mà vẫn nắm bắt được tất cả các thông tin quan trọng.

Trong trường hợp này, chúng ta sẽ tóm tắt các cuộc hội thoại bằng cách sử dụng một tập dữ liệu chứa các đoạn chat.

Đối với nhiệm vụ này, chúng ta sẽ sử dụng Tập dữ liệu SamSum, bao gồm ba tệp csv cho huấn luyện, kiểm tra và xác thực. Tất cả các tệp này được cấu trúc thành một id cụ thể, một cuộc hội thoại(a dialogue) và một bản tóm tắt(a summary). Tập dữ liệu SamSum bao gồm các đoạn chat, lý tưởng cho việc tóm tắt các cuộc hội thoại.

```
# IMPORTANT: RUN THIS CELL IN ORDER TO IMPORT YOUR KAGGLE DATA SOURCES,
# THEN FEEL FREE TO DELETE THIS CELL.
# NOTE: THIS NOTEBOOK ENVIRONMENT DIFFERS FROM KAGGLE'S PYTHON
# ENVIRONMENT SO THERE MAY BE MISSING LIBRARIES USED BY YOUR
# NOTEBOOK.
import kagglehub

nileshmalode1_samsum_dataset_text_summarization_path = kagglehub.
    dataset_download('nileshmalode1/samsum-dataset-text-summarization')

print('Data source import complete.')
```

2.2 Thư viện

```
# Data Handling
import pandas as pd
import numpy as np
```

```
from datasets import Dataset
import evaluate
import shutil

# Data Visualization
import plotly.express as px
import plotly.graph_objects as go
import plotly.subplots as sp
from plotly.subplots import make_subplots
import plotly.figure_factory as ff
import plotly.io as pio
from IPython.display import display
from plotly.offline import init_notebook_mode
pio.renderers.default = "colab"

# Statistics & Mathematics
import scipy.stats as stats
import statsmodels.api as sm
from scipy.stats import shapiro, skew, anderson, kstest, gaussian_kde,
    spearmanr
import math

# Hiding warnings
import warnings
warnings.filterwarnings("ignore")

# Transformers
from transformers import BartTokenizer, BartForConditionalGeneration      #
    BERT Tokenizer and architecture
from transformers import Seq2SeqTrainer, Seq2SeqTrainingArguments         #
    These will help us to fine-tune our model
from transformers import pipeline                                         #
    Pipeline
from transformers import DataCollatorForSeq2Seq                           #
    DataCollator to batch the data
import torch                                                              #
    PyTorch
import evaluate                                                           #
    Hugging Face's library for model evaluation

# Other NLP libraries
from textblob import TextBlob                                             #
    This is going to help us fix spelling mistakes in texts
from sklearn.feature_extraction.text import TfidfVectorizer              #
    This is going to helps identify the most common terms in the corpus
import re                                                                 #
    This library allows us to clean text data
```

```
import nltk                                                    # Natural Language Toolkit
nltk.download('punkt')                                         # This divides a text into a list of sentences
```

2.3 Chưa rõ

```
# Configuring Pandas to exhibit larger columns
'''
This is going to allow us to fully read the dialogues and their summary
'''
pd.set_option('display.max_colwidth', 1000)

# Configuring notebook
seed = 42
#paper_color =
#bg_color =
colormap = 'cividis'
template = 'plotly_dark'

# Checking if GPU is available
if torch.cuda.is_available():
    print("GPU is available. \nUsing GPU")
    device = torch.device('cuda')
else:
    print("GPU is not available. \nUsing CPU")
    device = torch.device('cpu')

def display_feature_list(features, feature_type):

    '''
    This function displays the features within each list for each type of data
    '''

    print(f"\n{feature_type} Features: ")
    print(', '.join(features) if features else 'None')

def describe_df(df):
    """
    This function prints some basic info on the dataset and
    sets global variables for feature lists.
    """

    global categorical_features, continuous_features, binary_features
    categorical_features = [col for col in df.columns if df[col].dtype == 'object']
```

```
    binary_features = [col for col in df.columns if df[col].nunique() <= 2 and df[col].dtype != 'object']
    continuous_features = [col for col in df.columns if df[col].dtype != 'object' and col not in binary_features]

    print(f"\n{type(df).__name__} shape: {df.shape}")
    print(f"\n{df.shape[0]:,.0f} samples")
    print(f"\n{df.shape[1]:,.0f} attributes")
    print(f'\nMissing Data: \n{df.isnull().sum()}')
    print(f'\nDuplicates: {df.duplicated().sum()}')
    print(f'\nData Types: \n{df.dtypes}')

    #negative_valued_features = [col for col in df.columns if (df[col] < 0).any()]
    #print(f'\nFeatures with Negative Values: {", ".join(negative_valued_features) if negative_valued_features else "None"}')

    display_feature_list(categorical_features, 'Categorical')
    display_feature_list(continuous_features, 'Continuous')
    display_feature_list(binary_features, 'Binary')

    print(f'\n{type(df).__name__} Head: \n')
    display(df.head(5))
    print(f'\n{type(df).__name__} Tail: \n')
    display(df.tail(5))

def histogram_boxplot(df,hist_color, box_color, height, width, legend, name):
    '''
    This function plots a Histogram and a Box Plot side by side

    Parameters:
    hist_color = The color of the histogram
    box_color = The color of the boxplots
    heigh and width = Image size
    legend = Either to display legend or not
    '''

    features = df.select_dtypes(include = [np.number]).columns.tolist()

    for feat in features:
        try:
            fig = make_subplots(
                rows=1,
                cols=2,
                subplot_titles=["Box Plot", "Histogram"],
                horizontal_spacing=0.2
```

```
            )

            density = gaussian_kde(df[feat])
            x_vals = np.linspace(min(df[feat]), max(df[feat]), 200)
            density_vals = density(x_vals)

            fig.add_trace(go.Scatter(x=x_vals, y = density_vals, mode = 'lines',
                                     fill = 'tozeroy', name="Density", line_color=hist_color), row=1, col=2)
            fig.add_trace(go.Box(y=df[feat], name="Box Plot", boxmean=True, line_color=box_color), row=1, col=1)

            fig.update_layout(title={'text': f'<b>{name} Word Count<br><sup><i>    {feat}</i></sup></b>',
                                     'x': .025, 'xanchor': 'left'},
                              margin=dict(t=100),
                              showlegend=legend,
                              template = template,
                              #plot_bgcolor=bg_color,paper_bgcolor=paper_color,
                              height=height, width=width
                             )

            fig.update_yaxes(title_text=f"<b>Words</b>", row=1, col=1, showgrid=False)
            fig.update_xaxes(title_text="", row=1, col=1, showgrid=False)

            fig.update_yaxes(title_text="<b>Frequency</b>", row=1, col=2, showgrid=False)
            fig.update_xaxes(title_text=f"<b>Words</b>", row=1, col=2, showgrid=False)

            fig.show()
            print('\n')
        except Exception as e:
            print(f"An error occurred: {e}")

def plot_correlation(df, title, subtitle, height, width, font_size):
    '''
    This function is resposible to plot a correlation map among features in the dataset.

    Parameters:
    height = Define height
    width = Define width
    font_size = Define the font size for the annotations
```

```
    '''
    corr = np.round(df.corr(numeric_only = True), 2)
    mask = np.triu(np.ones_like(corr, dtype = bool))
    c_mask = np.where(~mask, corr, 100)

    c = []
    for i in c_mask.tolist()[1:]:
        c.append([x for x in i if x != 100])



    fig = ff.create_annotated_heatmap(z=c[::-1],
                                      x=corr.index.tolist()[:-1],
                                      y=corr.columns.tolist()[1:][::-1],
                                      colorscale = colormap)

    fig.update_layout(title = {'text': f"<b>{title} Heatmap<br><sup>    <i>{subtitle}</i></sup></b>",
                               'x': .025, 'xanchor': 'left', 'y': .95},
                      margin = dict(t=210, l = 110),
                      yaxis = dict(autorange = 'reversed', showgrid = False),
                      xaxis = dict(showgrid = False),
                      template = template,
                      #plot_bgcolor=bg_color,paper_bgcolor=paper_color,
                      height = height, width = width)


    fig.add_trace(go.Heatmap(z = c[::-1],
                             colorscale = colormap,
                             showscale = True,
                             visible = False))
    fig.data[1].visible = True

    for i in range(len(fig.layout.annotations)):
        fig.layout.annotations[i].font.size = font_size

    fig.show()

def compute_tfidf(df_column, ngram_range=(1,1), max_features=15):
    vectorizer = TfidfVectorizer(max_features=max_features, stop_words='english', ngram_range=ngram_range)
    x = vectorizer.fit_transform(df_column.fillna(''))
    df_tfidfvect = pd.DataFrame(x.toarray(), columns=vectorizer.get_feature_names_out())
    return df_tfidfvect
```

2.4 Tập dữ liệu

Chúng ta có thể bắt đầu phân tích tập dữ liệu bằng cách tải tất cả ba tập dữ liệu có sẵn, bao gồm tập huấn luyện (train), tập kiểm tra (test) và tập xác thực (val).

```
train = pd.read_csv(nileshmalode1_samsum_dataset_text_summarization_path + '/samsum-train.csv')
test = pd.read_csv(nileshmalode1_samsum_dataset_text_summarization_path + '/samsum-test.csv')
val = pd.read_csv(nileshmalode1_samsum_dataset_text_summarization_path + '/samsum-validation.csv')
```

	id	dialogue	summary
0	13862856	Hannah: Hey, do you have Betty's number?\nAmanda: Lemme check\nHannah: <file_gif>\nAmanda: Sorry, can't find it.\nAmanda: Ask Larry\nAmanda: He called her last time we were at the park together\nHannah: I don't know him well\nHannah: <file_gif>\nAmanda: Don't be shy, he's very nice\nHannah: If you say so..\nHannah: I'd rather you texted him\nAmanda: Just text him 😊\nHannah: Urgh.. Alright\nHannah: Bye\nAmanda: Bye bye	Hannah needs Betty's number but Amanda doesn't have it. She needs to contact Larry.
1	13729565	Eric: MACHINE!\r\nRob: That's so gr8!\r\nEric: I know! And shows how Americans see Russian ;)\r\nRob: And it's really funny!\r\nEric: I know! I especially like the train part!\r\nRob: Hahaha! No one talks to the machine like that!\r\nEric: Is this his only stand-up?\r\nRob: Idk. I'll check.\r\nEric: Sure.\r\nRob: Turns out no! There are some of his stand-ups on youtube.\r\nEric: Gr8! I'll watch them now!\r\nRob: Me too!\r\nEric: MACHINE!\r\nRob: MACHINE!\r\nEric: TTYL?\r\nRob: Sure :)	Eric and Rob are going to watch a stand-up on youtube.
2	13680171	Lenny: Babe, can you help me with something?\r\nBob: Sure, what's up?\r\nLenny: Which one should I pick?\r\nBob: Send me photos\r\nLenny: <file_photo>\r\nLenny: <file_photo>\r\nLenny: <file_photo>\r\nBob: I like the first ones best\r\nLenny: But I already have purple trousers. Does it make sense to have two pairs?\r\nBob: I have four black pairs :D :D\r\nLenny: yeah, but shouldn't I pick a different color?\r\nBob: what matters is what you'll give you the most outfit options\r\nLenny: So I guess I'll buy the first or the third pair then\r\nBob: Pick the best quality then\r\nLenny: ur right, thx\r\nBob: no prob :)	Lenny can't decide which trousers to buy. Bob advised Lenny on that topic. Lenny goes with Bob's advice to pick the trousers that are of best quality.
3	13729438	Will: hey babe, what do you want for dinner tonight?\r\nEmma: gah, don't even worry about it tonight\r\nWill: what do you mean? everything ok? \r\nEmma: not really, but it's ok, don't worry about cooking though, I'm not hungry\r\nWill: Well what time will you be home?\r\nEmma: soon, hopefully\r\nWill: you sure? Maybe you want me to pick you up?\r\nEmma: no no it's alright. I'll be home soon, i'll tell you when I get home. \r\nWill: Alright, love you. \r\nEmma: love you too.	Emma will be home soon and she will let Will know.
4	13828600	Ollie: Hi , are you in Warsaw\r\nJane: yes, just back! Btw are you free for diner the 19th?\r\nOllie: nope!\r\nJane: and the 18th?\r\nOllie: nope, we have this party and you must be there, remember?\r\nJane: oh right! i lost my calendar.. thanks for reminding me\r\nOllie: we have lunch this week?\r\nJane: with pleasure!\r\nOllie: friday?\r\nJane: ok\r\nJane: what do you mean " we don't have any more whisky!" lol..\r\nOllie: what!!!\r\nJane: you just call me and the all thing i heard was that sentence about whisky... what's wrong with you?\r\nOllie: oh oh... very strange! i have to be carefull may be there is some spy in my mobile! lol\r\nJane: dont' worry, we'll check on friday.\r\nOllie: don't forget to bring some sun with you\r\nJane: I can't wait to be in Morocco..\r\nOllie: enjoy and see you friday\r\nJane: sorry Ollie, i'm very busy, i won't have time for lunch tomorrow, but may be at 6pm after my courses?this trip to Morocco was so nice, but time consuming!\r\nOllie: ok fo...	Jane is in Warsaw. Ollie and Jane has a party. Jane lost her calendar. They will get a lunch this week on Friday. Ollie accidentally called Jane and talked about whisky. Jane cancels lunch. They'll meet for a tea at 6 pm.

Hình 2.1: Minh họa Dataframe

Chương 3

Xây dựng mô hình

3.1 Tiền xử lý dữ liệu

Chúng ta nhận thấy rằng trong một số đoạn hội thoại sẽ có những thẻ như file photo, hãy xem qua một ví dụ:

```
print(train['dialogue'].iloc[14727])
```

```
Theresa: <file_photo>
Theresa: <file_photo>
Theresa: Hey Louise, how are u?
Theresa: This is my workplace, they always give us so much food here 😊
Theresa: Luckily they also offer us yoga classes, so all the food isn't much of a problem 🤩
Louise: Hey!! 😀
Louise: Wow, that's awesome, seems great 😎 Haha
Louise: I'm good! Are you coming to visit Stockholm this summer? 😀
Theresa: I don't think so :/ I need to prepare for Uni.. I will probably attend a few lessons this winter
Louise: Nice! Do you already know which classes you will attend?
Theresa: Yes, it will be psychology :) I want to complete a few modules that I missed :)
Louise: Very good! Is it at the Uni in Prague?
Theresa: No, it will be in my home town :)
Louise: I have so much work right now, but I will continue to work until the end of summer, then I'm also back to Uni, on the 26th September
Theresa: You must send me some pictures, so I can see where you live :)
Louise: I will, and of my cat and dog too 🤗
Theresa: Yeeeesss pls :)))
Louise: 👌👌
Theresa: 🐱💕
```

Để loại bỏ các thẻ này khỏi văn bản, giúp văn bản trở nên sạch hơn, chúng ta sẽ sử dụng hàm clean-tags được định nghĩa dưới đây:

```
def clean_tags(text):
    clean = re.compile('<.*?>') # Compiling tags
    clean = re.sub(clean, '', text) # Replacing tags text by an empty
    string

    # Removing empty dialogues
    clean = '\n'.join([line for line in clean.split('\n') if not re.match('
    .*:\s*$', line)])

    return clean

test1 = clean_tags(train['dialogue'].iloc[14727]) # Applying function to
    example text
test2 = clean_tags(test['dialogue'].iloc[0]) # Applying function to example
     text
```

```
12
13 # Printing results
14 print(test1)
15 print('\n' *3)
16 print(test2)
```

```
Theresa: Hey Louise, how are u?
Theresa: This is my workplace, they always give us so much food here 😊
Theresa: Luckily they also offer us yoga classes, so all the food isn't much of a problem 😂
Louise: Hey!! 😀
Louise: Wow, that's awesome, seems great 😎 Haha
Louise: I'm good! Are you coming to visit Stockholm this summer? 😀
Theresa: I don't think so :/ I need to prepare for Uni.. I will probably attend a few lessons this winter
Louise: Nice! Do you already know which classes you will attend?
Theresa: Yes, it will be psychology :) I want to complete a few modules that I missed :)
Louise: Very good! Is it at the Uni in Prague?
Theresa: No, it will be in my home town :)
Louise: I have so much work right now, but I will continue to work until the end of summer, then I'm also back to Uni, on the 26th September!
Theresa: You must send me some pictures, so I can see where you live :)
Louise: I will, and of my cat and dog too 😻
Theresa: Yeeeesss pls :)))
Louise: 👌👌
Theresa: 🙀💕
```

Có thể thấy rằng chúng ta đã loại bỏ thành công các thẻ khỏi văn bản. Bây giờ, chúng ta sẽ định nghĩa hàm clean-df, trong đó sẽ áp dụng hàm clean-tags cho toàn bộ các bộ dữ liệu.

```
1 # Defining function to clean every text in the dataset.
2 def clean_df(df, cols):
3     for col in cols:
4         df[col] = df[col].fillna('').apply(clean_tags)
5     return df
6
7 # Cleaning texts in all datasets
8 train = clean_df(train,['dialogue', 'summary'])
9 test = clean_df(test,['dialogue', 'summary'])
10 val = clean_df(val,['dialogue', 'summary'])
```

Các thẻ đã được loại bỏ khỏi văn bản. Việc thực hiện quá trình dọn dẹp dữ liệu như vậy là rất có ích để loại bỏ nhiễu - những thông tin không đóng góp đáng kể vào ngữ cảnh tổng thể và có thể làm giảm hiệu suất.

Bây giờ, chúng ta sẽ thực hiện một số bước tiền xử lý cần thiết để chuẩn bị dữ liệu của chúng ta làm đầu vào cho mô hình đã được huấn luyện sẵn và để tinh chỉnh mô hình. Phần lớn những gì chúng ta đang làm ở đây là một phần trong hướng dẫn về Tóm tắt Văn bản được mô tả trong tài liệu của Transformers.

Đầu tiên, chúng ta sẽ sử dụng thư viện Datasets để chuyển các Pandas DataFrame của chúng ta thành Datasets. Điều này sẽ giúp dữ liệu của

chúng ta sẵn sàng để xử lý trong toàn bộ hệ sinh thái của Hugging Face.

```
# Transforming dataframes into datasets
train_ds = Dataset.from_pandas(train)
test_ds = Dataset.from_pandas(test)
val_ds = Dataset.from_pandas(val)

# Visualizing results
print(train_ds)
print('\n' * 2)
print(test_ds)
print('\n' * 2)
print(val_ds)
```

Sau khi chuyển đổi thành công các Pandas DataFrames thành Datasets, chúng ta có thể tiếp tục với quá trình mô hình hóa.

3.2 Mô hình hóa

Như chúng ta đã đề cập trước đó, chúng ta sẽ tinh chỉnh một phiên bản của BART đã được huấn luyện trên nhiều bài báo tin tức cho nhiệm vụ tóm tắt văn bản, đó là facebook/bart-large-xsum. Chúng ta sẽ trình bày ngắn gọn mô hình này bằng cách tải một pipeline tóm tắt với mô hình này để cho bạn thấy cách nó hoạt động trên dữ liệu tin tức.

```
# Loading summarization pipeline with the bart-large-cnn model
summarizer = pipeline('summarization', model = 'facebook/bart-large-xsum')
```

Ví dụ, chúng ta sẽ sử dụng bài báo tin tức sau, được xuất bản trên CNN vào ngày 24 tháng 10 năm 2023, với tiêu đề "Bobi, the world's oldest dog ever, died aged 31". Lưu ý rằng đây là một bài báo tin tức hoàn toàn mới mà chúng ta đưa vào mô hình, để có thể xem nó hoạt động như thế nào.

```

new="""Bobi, the worlds oldest dog ever, has died after reaching the
    almost inconceivable age of 31 years and 165 days, said Guinness World
    Records (GWR) on Monday.His death at an animal hospital on Friday was
    initially announced by veterinarian Dr. Karen Becker.She wrote on
    Facebook that  despite  outliving every dog in history, his 11,478 days
```

```
    on earth would never be enough, for those who loved him. ÍThere were
    many secrets to Bobis extraordinary old age, his owner Leonel Costa
    told GWR in February. He always roamed freely, without a leash or chain,
    lived in a calm, peacefulÍ environment and ate human food soaked
    in water to remove seasonings, Costa said.He spent his whole life in
    Conqueiros, a small Portuguese village about 150 kilometers (93 miles)
    north of the capital Lisbon, often wandering around with cats.Bobi was a
    purebred Rafeiro do Alentejo a breed of livestock guardian dog
    according to his owner. Rafeiro do Alentejos have a life expectancy of
    about 12–14 years, according to the American Kennel Club.But Bobi lived
    more than twice as long as that life expectancy, surpassing an almost
    century–old record to become the oldest living dog and the oldest dog
    ever a title which had previously been held by Australian cattle–dog
    Bluey, who was born in 1910 and lived to be 29 years and five months
    old.However, Bobis story almost had a different ending.When he and
    his three siblings were born in the familys woodshed, Costas
    father decided they already had too many animals at home.Costa and his
    brothers thought their parents had taken all the puppies away to be
    destroyed. However, a few sad days later, they found Bobi alive, safely
    hidden in a pile of logs.The children hid the puppy from their parents
    and, by the time Bobis existence became known, he was too old to be
    put down and went on to live his record–breaking life.His 31st birthday
    party in May was attended by more than 100 people and a performing dance
    troupe, GWR said.His eyesight deteriorated and walking became harder as
    Bobi grew older but he still spent time in the backyard with the cats,
    rested more and napped by the fire. Bobi is special because looking at
    him is like remembering the people who were part of our family and
    unfortunately are no longer here, like my father, my brother, or my
    grandparents who have already left this world, Í Costa told GWR in May.
    Bobi represents those generations"""
3
4
5 summarizer(news) # Using the pipeline to generate a summary of the text
    above
```

```
[{'summary_text': 'The world's oldest dog has died, Guinness World Records has confirmed.'}]
```

Bạn có thể quan sát thấy rằng mô hình có thể tạo ra một đoạn văn bản ngắn gọn hơn rất nhiều, chứa đựng thông tin quan trọng nhất từ văn bản đầu vào. Đây là một ví dụ tóm tắt thành công.

Tuy nhiên, mô hình này đã được huấn luyện chủ yếu trên các tập dữ liệu bao gồm nhiều bài báo tin tức từ CNN và Daily Mail, chứ không phải trên nhiều dữ liệu hội thoại. Vì vậy, chúng ta sẽ tiến hành tinh chỉnh nó với SamSum dataset.

Bây giờ, chúng ta sẽ tải BartTokenizer và BartForConditionalGeneration sử dụng checkpoint facebook/bart-large-xsum.

Tài liệu tham khảo