

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

<b>Mục lục</b>	<b>i</b>
<b>1 Giới thiệu</b>	<b>1</b>
1.1 Thông tin nhóm . . . . .	1
1.2 Mô hình ngôn ngữ lớn . . . . .	1
1.3 The Transformer Architecture . . . . .	2
1.4 Mục tiêu . . . . .	4
<b>2 Cài đặt</b>	<b>6</b>
2.1 Data source . . . . .	6
2.2 Thư viện . . . . .	6
2.3 Chưa rõ . . . . .	8
2.4 Tập dữ liệu . . . . .	12
<b>3 Xây dựng mô hình</b>	<b>13</b>
3.1 Tiền xử lý dữ liệu . . . . .	13
3.2 Mô hình hóa . . . . .	15
<b>Tài liệu tham khảo</b>	<b>17</b>

# 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.

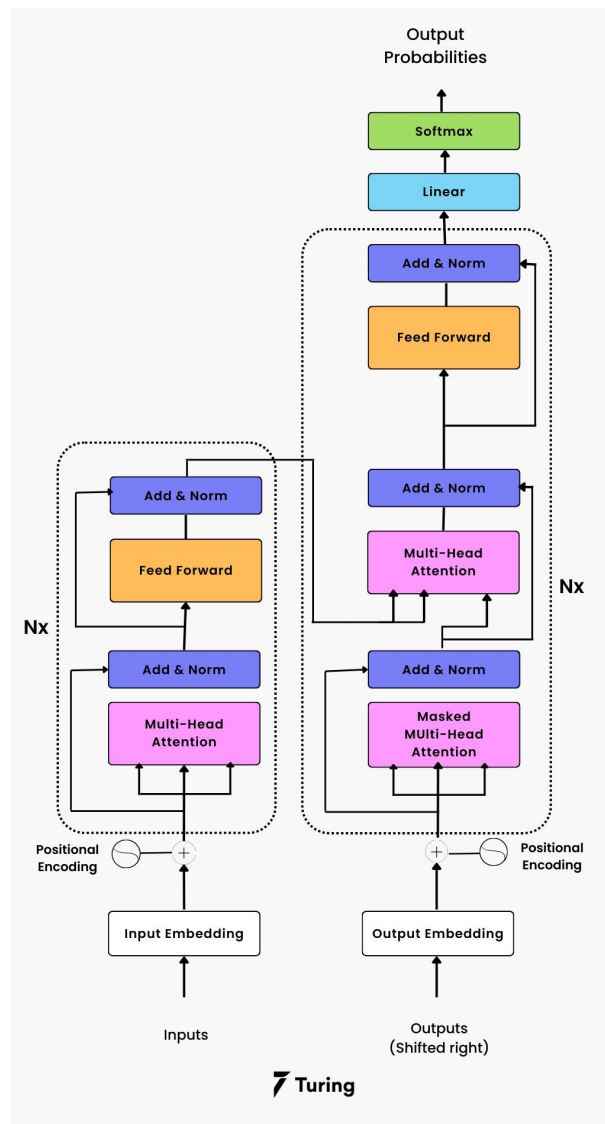
Tuy nhiên, kiến trúc Transformer dựa trên cơ chế chú ý (attention mechanism), cho phép mô hình xử lý toàn bộ câu hoặc đoạn văn cùng một lúc, thay vì từng từ một. Đây là bí mật chính đằng sau khả năng hiểu ngữ cảnh đầy đủ, mang lại sức mạnh lớn hơn cho tất cả các mô hình xử lý ngôn ngữ này.

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.

```
1 # IMPORTANT: RUN THIS CELL IN ORDER TO IMPORT YOUR KAGGLE DATA SOURCES,  
2 # THEN FEEL FREE TO DELETE THIS CELL.  
3 # NOTE: THIS NOTEBOOK ENVIRONMENT DIFFERS FROM KAGGLE'S PYTHON  
4 # ENVIRONMENT SO THERE MAY BE MISSING LIBRARIES USED BY YOUR  
5 # NOTEBOOK.  
6 import kagglehub  
7  
8 nileshmalode1_samsum_dataset_text_summarization_path = kagglehub.  
   dataset_download('nileshmalode1/samsum-dataset-text-summarization')  
9  
10 print('Data source import complete.')
```

### 2.2 Thư viện

```
1 # Data Handling  
2 import pandas as pd  
3 import numpy as np
```



```

4 from datasets import Dataset
5 import evaluate
6 import shutil
7
8 # Data Visualization
9 import plotly.express as px
10 import plotly.graph_objects as go
11 import plotly.subplots as sp
12 from plotly.subplots import make_subplots
13 import plotly.figure_factory as ff
14 import plotly.io as pio
15 from IPython.display import display
16 from plotly.offline import init_notebook_mode
17 pio.renderers.default = "colab"
18
19 # Statistics & Mathematics
20 import scipy.stats as stats
21 import statsmodels.api as sm
22 from scipy.stats import shapiro, skew, anderson, kstest, gaussian_kde,
    spearmanr
23 import math
24
25 # Hiding warnings
26 import warnings
27 warnings.filterwarnings("ignore")
28
29 # Transformers
30 from transformers import BartTokenizer, BartForConditionalGeneration #
    BERT Tokenizer and architecture
31 from transformers import Seq2SeqTrainer, Seq2SeqTrainingArguments #
    These will help us to fine-tune our model
32 from transformers import pipeline #
    Pipeline
33 from transformers import DataCollatorForSeq2Seq #
    DataCollator to batch the data
34 import torch #
    PyTorch
35 import evaluate #
    Hugging Face's library for model evaluation
36
37 # Other NLP libraries
38 from textblob import TextBlob #
    This is going to help us fix spelling mistakes in texts
39 from sklearn.feature_extraction.text import TfidfVectorizer #
    This is going to help identify the most common terms in the corpus
40 import re #
    This library allows us to clean text data

```

```

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

```

## 2.3 Chưa rõ

```

1 # Configuring Pandas to exhibit larger columns
2 '''
3 This is going to allow us to fully read the dialogues and their summary
4 '''
5 pd.set_option('display.max_colwidth', 1000)
6
7 # Configuring notebook
8 seed = 42
9 #paper_color =
10 #bg_color =
11 colormap = 'cividis'
12 template = 'plotly_dark'
13
14 # Checking if GPU is available
15 if torch.cuda.is_available():
16     print("GPU is available. \nUsing GPU")
17     device = torch.device('cuda')
18 else:
19     print("GPU is not available. \nUsing CPU")
20     device = torch.device('cpu')
21
22 def display_feature_list(features, feature_type):
23
24     '''
25     This function displays the features within each list for each type of
26     data
27     '''
28
29     print(f"\n{feature_type} Features: ")
30     print(', '.join(features) if features else 'None')
31
32 def describe_df(df):
33     """
34     This function prints some basic info on the dataset and
35     sets global variables for feature lists.
36     """
37
38     global categorical_features, continuous_features, binary_features
39     categorical_features = [col for col in df.columns if df[col].dtype == '
    object']

```

```

39     binary_features = [col for col in df.columns if df[col].nunique() <= 2
and df[col].dtype != 'object']
40     continuous_features = [col for col in df.columns if df[col].dtype != '
object' and col not in binary_features]
41
42     print(f"\n{type(df).__name__} shape: {df.shape}")
43     print(f"\n{df.shape[0]:,.0f} samples")
44     print(f"\n{df.shape[1]:,.0f} attributes")
45     print(f'\nMissing Data: \n{df.isnull().sum()}')
46     print(f'\nDuplicates: {df.duplicated().sum()}')
47     print(f'\nData Types: \n{df.dtypes}')
48
49     #negative_valued_features = [col for col in df.columns if (df[col] < 0)
.any()]
50     #print(f'\nFeatures with Negative Values: {" ".join(
negative_valued_features) if negative_valued_features else "None"}')
51
52     display_feature_list(categorical_features, 'Categorical')
53     display_feature_list(continuous_features, 'Continuous')
54     display_feature_list(binary_features, 'Binary')
55
56     print(f'\n{type(df).__name__} Head: \n')
57     display(df.head(5))
58     print(f'\n{type(df).__name__} Tail: \n')
59     display(df.tail(5))
60
61 def histogram_boxplot(df, hist_color, box_color, height, width, legend, name
):
62     '''
63     This function plots a Histogram and a Box Plot side by side
64
65     Parameters:
66     hist_color = The color of the histogram
67     box_color = The color of the boxplots
68     heigh and width = Image size
69     legend = Either to display legend or not
70     '''
71
72     features = df.select_dtypes(include = [np.number]).columns.tolist()
73
74     for feat in features:
75         try:
76             fig = make_subplots(
77                 rows=1,
78                 cols=2,
79                 subplot_titles=["Box Plot", "Histogram"],
80                 horizontal_spacing=0.2

```

```

81         )
82
83         density = gaussian_kde(df[feat])
84         x_vals = np.linspace(min(df[feat]), max(df[feat]), 200)
85         density_vals = density(x_vals)
86
87         fig.add_trace(go.Scatter(x=x_vals, y=density_vals, mode='
lines',
88                               fill='tozeroy', name="Density",
line_color=hist_color), row=1, col=2)
89         fig.add_trace(go.Box(y=df[feat], name="Box Plot", boxmean=True,
line_color=box_color), row=1, col=1)
90
91         fig.update_layout(title={'text': f'<b>{name}</b> Word Count<br><sup>
<i>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&~i>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&~i>&nbsp;&nbsp;&nbsp;&nbsp;&~i>&nbsp;&nbsp;&nbsp;&~i></sup></b>',
92                              'x': .025, 'xanchor': 'left'},
margin=dict(t=100),
93          showlegend=legend,
94          template = template,
95          #plot_bgcolor=bg_color, paper_bgcolor=
paper_color,
96
height=height, width=width
97      )
98
99
100     fig.update_yaxes(title_text=f"<b>Words</b>", row=1, col=1,
showgrid=False)
101     fig.update_xaxes(title_text="", row=1, col=1, showgrid=False)
102
103     fig.update_yaxes(title_text=f"<b>Frequency</b>", row=1, col=2,
showgrid=False)
104     fig.update_xaxes(title_text=f"<b>Words</b>", row=1, col=2,
showgrid=False)
105
106     fig.show()
107     print('\n')
108     except Exception as e:
109         print(f"An error occurred: {e}")
110
111 def plot_correlation(df, title, subtitle, height, width, font_size):
112     '''
113     This function is responsible to plot a correlation map among features in
the dataset.
114
115     Parameters:
116     height = Define height
117     width = Define width
118     font size = Define the font size for the annotations

```

```

119     '''
120     corr = np.round(df.corr(numeric_only = True), 2)
121     mask = np.triu(np.ones_like(corr, dtype = bool))
122     c_mask = np.where(~mask, corr, 100)
123
124     c = []
125     for i in c_mask.tolist()[1:]:
126         c.append([x for x in i if x != 100])
127
128
129
130     fig = ff.create_annotated_heatmap(z=c[:, :-1],
131                                     x=corr.index.tolist()[:-1],
132                                     y=corr.columns.tolist()[1:][::-1],
133                                     colorscale = colormap)
134
135     fig.update_layout(title = {'text': f"<b>{title}</b> Heatmap<br><sup>&nbsp;&nbsp;&nbsp;</sup>&nbsp;&nbsp;&nbsp;</sup></b></i></sup></b>",
136                             'x': .025, 'xanchor': 'left', 'y': .95},
137                       margin = dict(t=210, l = 110),
138                       yaxis = dict(autorange = 'reversed', showgrid = False),
139                       xaxis = dict(showgrid = False),
140                       template = template,
141                       #plot_bgcolor=bg_color, paper_bgcolor=paper_color,
142                       height = height, width = width)
143
144
145     fig.add_trace(go.Heatmap(z = c[:, :-1],
146                             colorscale = colormap,
147                             showscale = True,
148                             visible = False))
149     fig.data[1].visible = True
150
151     for i in range(len(fig.layout.annotations)):
152         fig.layout.annotations[i].font.size = font_size
153
154     fig.show()
155
156 def compute_tfidf(df_column, ngram_range=(1,1), max_features=15):
157     vectorizer = TfidfVectorizer(max_features=max_features, stop_words='
158     english', ngram_range=ngram_range)
159     x = vectorizer.fit_transform(df_column.fillna(''))
160     df_tfidfvect = pd.DataFrame(x.toarray(), columns=vectorizer.
get_feature_names_out())
161     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).

```
1 train = pd.read_csv(nileshmalode1_samsum_dataset_text_summarization_path +  
    '/samsum-train.csv')  
2 test = pd.read_csv(nileshmalode1_samsum_dataset_text_summarization_path + '  
    /samsum-test.csv')  
3 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? Amanda: Lemme check Hannah: <file_gif> Amanda: Sorry, can't find it. Amanda: Ask Larry Amanda: He called her last time we were at the park together Hannah: I don't know him well Hannah: <file_gif> Amanda: Don't be shy, he's very nice Hannah: If you say so.. Hannah: I'd rather you texted him Amanda: Just text him 🙄 Hannah: Urgh.. Alright Hannah: Bye Amanda: Bye bye	Hannah needs Betty's number but Amanda doesn't have it. She needs to contact Larry.
1	13729565	Eric: MACHINE! Rob: That's so gr8! Eric: I know! And shows how Americans see Russian :) Rob: And it's really funny! Eric: I know! I especially like the train part! Rob: Hahaha! No one talks to the machine like that! Eric: Is this his only stand-up? Rob: Idk, I'll check. Eric: Sure. Rob: Turns out no! There are some of his stand-ups on youtube. Eric: Gr8! I'll watch them now! Rob: Me too! Eric: MACHINE! Rob: MACHINE! Eric: TTYL? Rob: Sure :)	Eric and Rob are going to watch a stand-up on youtube.
2	13680171	Lenny: Babe, can you help me with something? Bob: Sure, what's up? Lenny: Which one should I pick? Bob: Send me photos Lenny: <file_photo> Lenny: <file_photo> Lenny: <file_photo> Bob: I like the first ones best Lenny: But I already have purple trousers. Does it make sense to have two pairs? Bob: I have four black pairs :D Lenny: yeah, but shouldn't I pick a different color? Bob: what matters is what you'll give you the most outfit options Lenny: So I guess I'll buy the first or the third pair then Bob: Pick the best quality then Lenny: ur right, thx Bob: 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? Emma: gah, don't even worry about it tonight Will: what do you mean? everything ok? Emma: not really, but it's ok, don't worry about cooking though, I'm not hungry Will: Well what time will you be home? Emma: soon, hopefully Will: you sure? Maybe you want me to pick you up? Emma: no no it's alright. I'll be home soon, I'll tell you when I get home. Will: Alright, love you. Emma: love you too.	Emma will be home soon and she will let Will know.
4	13828600	Ollie: Hi , are you in Warsaw? Jane: yes, just back! Btw are you free for diner the 19th? Ollie: nope! Jane: and the 18th? Ollie: nope, we have this party and you must be there, remember? Jane: oh right! I lost my calendar.. thanks for reminding me Ollie: we have lunch this week? Jane: with pleasure! Ollie: friday? Jane: ok Jane: what do you mean " we don't have any more whisky!" lol.. Ollie: what!!! Jane: you just call me and the all thing i heard was that sentence about whisky... what's wrong with you? Ollie: oh oh... very strange! i have to be carefull may be there is some spy in my mobile! lol Jane: dont' worry, we'll check on friday Ollie: dont forget to bring some sun with you Jane: I can't wait to be in Morocco.. Ollie: enjoy and see you friday Jane: sorry Ollie, i'm very busy, i wont have time for lunch tomorrow, but may be at 6pm after my courses? this trip to Morocco was so nice, but time consuming! Ollie: 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: Yeeesss 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:

```
1 def clean_tags(text):
2     clean = re.compile('<.*?>') # Compiling tags
3     clean = re.sub(clean, '', text) # Replacing tags text by an empty
4     string
5
6     # Removing empty dialogues
7     clean = '\n'.join([line for line in clean.split('\n') if not re.match('
8     .*:\s*$', line)])
9
10    return clean
11
12 test1 = clean_tags(train['dialogue'].iloc[14727]) # Applying function to
13    example text
14 test2 = clean_tags(test['dialogue'].iloc[0]) # Applying function to example
15    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: Yeeeeesss 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.

```
1 # Transforming dataframes into datasets
2 train_ds = Dataset.from_pandas(train)
3 test_ds = Dataset.from_pandas(test)
4 val_ds = Dataset.from_pandas(val)
5
6 # Visualizing results
7 print(train_ds)
8 print('\n' * 2)
9 print(test_ds)
10 print('\n' * 2)
11 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.

```
1 # Loading summarization pipeline with the bart-large-cnn model
2 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.

```
[ ] news = '''Bobi, the world's oldest dog ever, has died after reaching the almost inconceivable age of 31 years and 165 days. 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 there were many secrets to Bobi's extraordinary old age, his owner Leonel Costa told GWR in February. He always roamed freely. He spent his whole life in Conqueiros, a small Portuguese village about 150 kilometers (93 miles) north of the capital Lisbon. Bobi was a purebred Rafeiro do Alentejo - a breed of livestock guardian dog - according to his owner. Rafeiro do Alentejos But Bobi lived more than twice as long as that life expectancy, surpassing an almost century-old record to become the oldest. However, Bobi's story almost had a different ending. When he and his three siblings were born in the family's woodshed, Costa's father decided they already had too many animals. Costa and his brothers thought their parents had taken all the puppies away to be destroyed. However, a few sad days later The children hid the puppy from their parents and, by the time Bobi's existence became known, he was too old to be put down. 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. "Bobi is special because looking at him is like remembering the people who were part of our family and unfortunately are no more."'''

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.'}]
```

Hình 3.1: Tóm tắt bài báo

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