FIT@HCMUS
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN, ĐHQG - HCM
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN CUỐI KỲ

BỘ MÔN: HỌC THỐNG KÊ

Sinh viên thực hiện:
21120396 - Đào Thị Ngọc Giàu
21120419 - Vũ Thành Công
21120446 - Kiên Đinh Mỹ Hạnh

TP. Hồ Chí Minh, tháng 6/2024

Mục lục

1. Thông tin chung

a) Thông tin thành viên

MSSV	*Họ và tên*
21120396	Đào Thị Ngọc Giàu
21120419	Vũ Thành Công
21120446	Kiên Đinh Mỹ Hạnh

b) Thông tin về đồ án

Bài toán trong lĩnh vực Xử lý Ngôn ngữ Tự nhiên: Text Classification

2. Mô tả về tập dữ liệu

- Có thể download dataset từ Kaggle.
- Dữ liệu này ở định dạng CSV và có hai cột: text và category. Nó chứa 2226 văn bản khác nhau, mỗi văn bản được gán nhãn dưới một trong 5 danh mục: entertainment, sport, tech, business, politics.

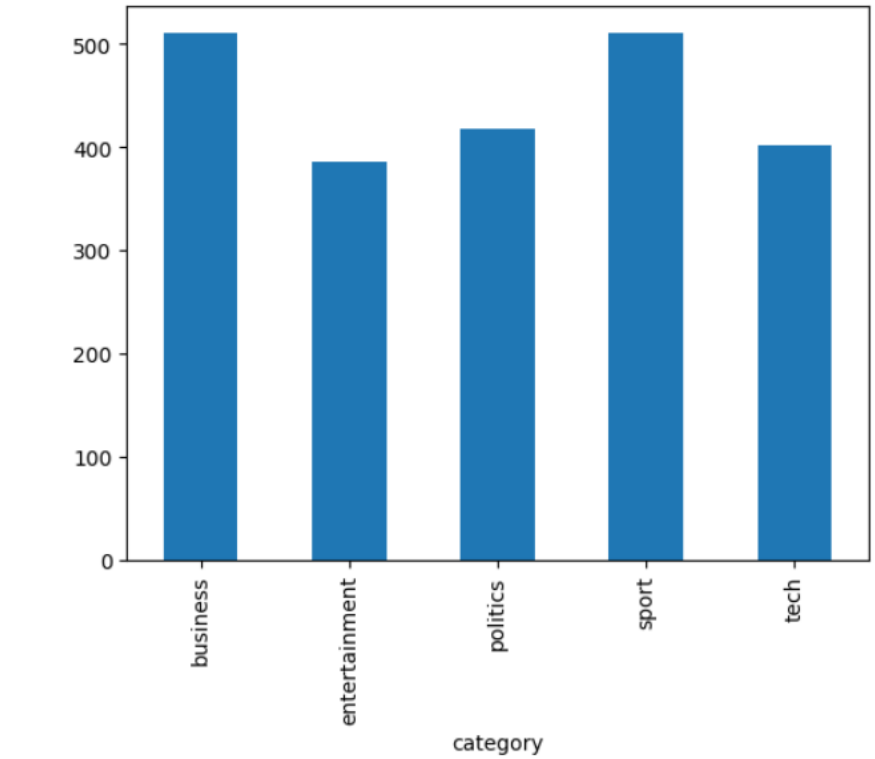
```
df.head()
```

	category	text
0	tech	tv future in the hands of viewers with home th...
1	business	worldcom boss left books alone former worldc...
2	sport	tigers wary of farrell gamble leicester say ...
3	sport	yeading face newcastle in fa cup premiership s...
4	entertainment	ocean s twelve raids box office ocean s twelve...

Next steps: Generate code with df View recommended plots

```
df.groupby("category").size().plot.bar()
```

```
<Axes: xlabel='category'>
```



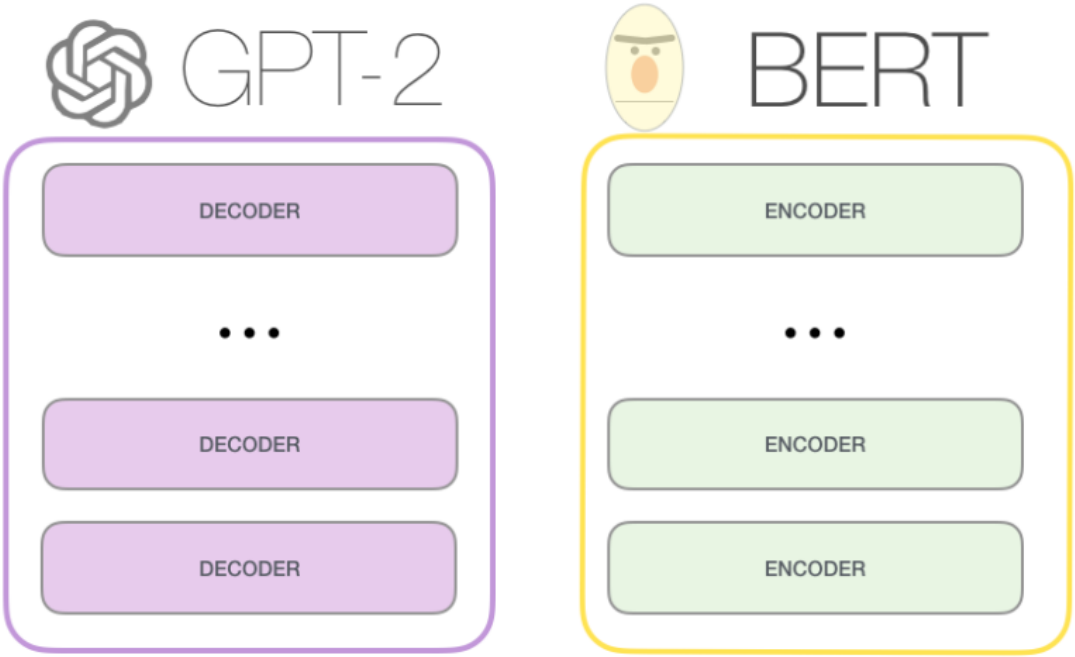
3. Mô hình đã sử dụng

Phân loại văn bản (text classification) là một nhiệm vụ rất phổ biến trong xử lý ngôn ngữ tự nhiên (NLP). Nó có thể được sử dụng trong nhiều ứng dụng, từ lọc thư rác, phân tích cảm xúc đến tự động hóa hỗ trợ khách hàng và phân loại tin tức. Việc sử dụng các mô hình ngôn ngữ học sâu (deep learning) cho các tác vụ phân loại văn bản quy mô lớn đã trở nên khá phổ biến trong ngành công nghiệp gần đây, đặc biệt là với sự xuất hiện của các mô hình Transformer trong những năm gần đây. Do kích thước của mô hình Transformer này thường quá lớn để huấn luyện trên các máy tính cục bộ, các nền tảng điện toán đám mây (như GCP, AWS, Azure, Alibabacloud) thường được sử dụng. Do đó, trong bài viết này, tôi muốn demonstrating cách huấn luyện triển khai mô hình GPT-2 được tinh chỉnh để thực hiện các tác vụ phân loại văn bản

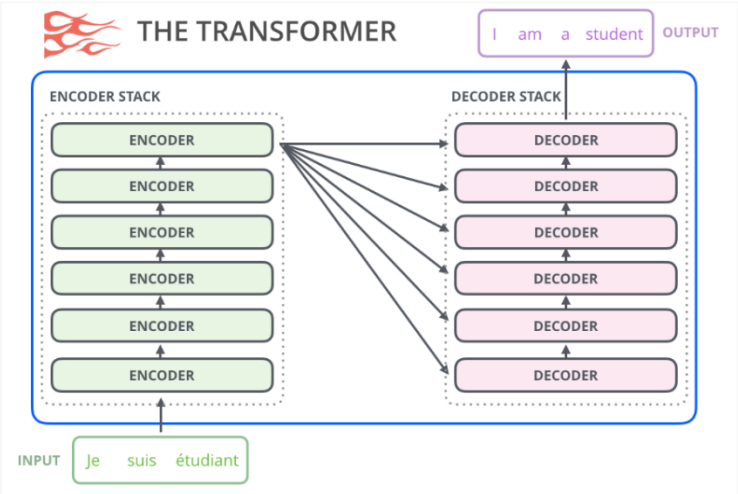
3.1. Introduction

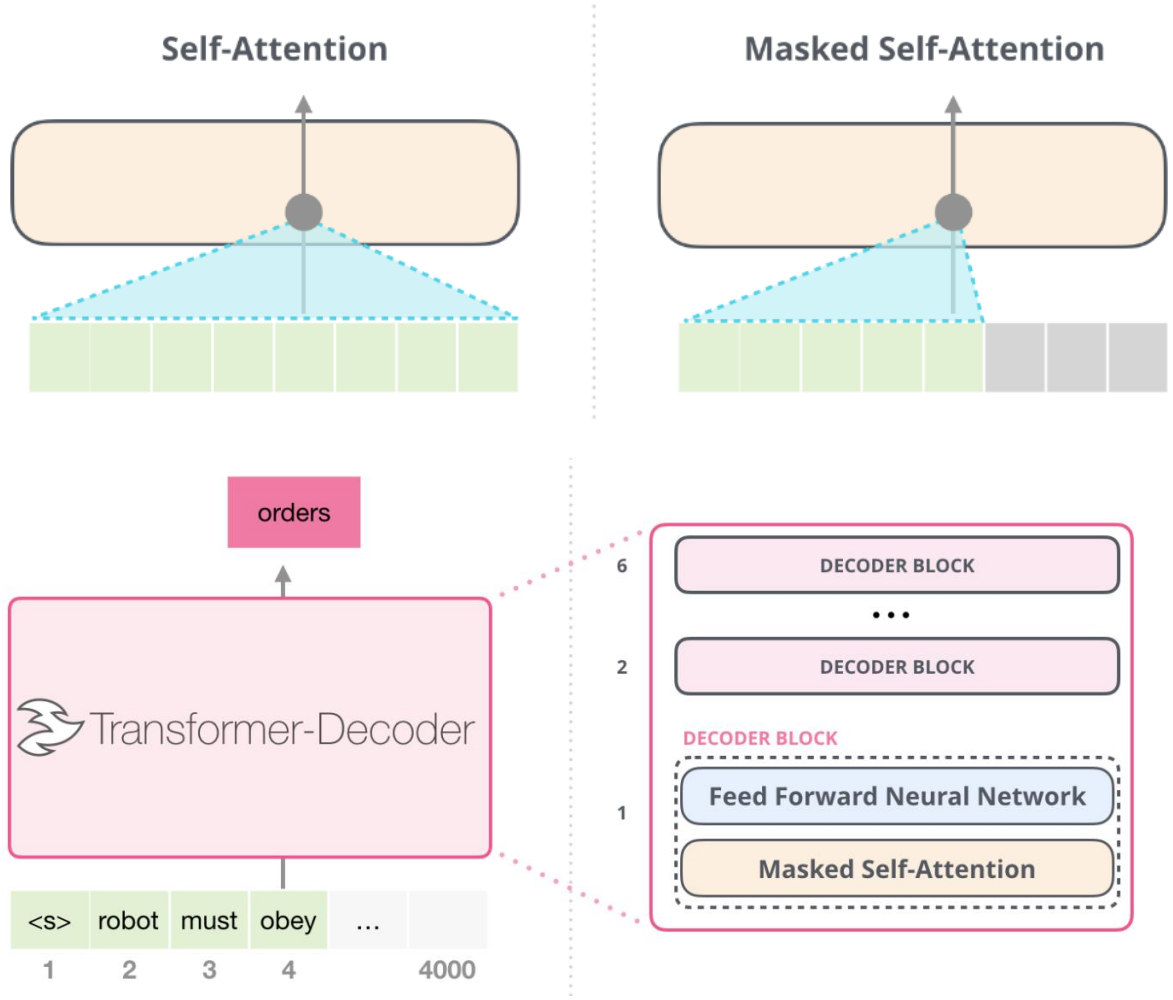
a) Transformers: GPT-2 và BERT

- Các mô hình Transformer bao gồm GPT-2, là một họ các mô hình học sâu tiên tiến hiện đang được sử dụng rộng rãi trong lĩnh vực xử lý ngôn ngữ tự nhiên (NLP). Transformer sử dụng cơ chế "seft-attention" mối quan hệ giữa các từ ở các vị trí khác nhau trong một câu.
- Cấu trúc điển hình của một mô hinh Transformer bao gồm hai thành phần chính: encoder và decoder. Cả hai đều hoạt động như các biểu diễn dạng vector của mối quan hệ giữa các từ.
- Không thể giải thích chi tiết cách hoạt động mô hình Transformer trong đoạn ngắn, nhưng có thể nói rằng đây là nền tảng kiến thức NLP tiên tiến hiện nay, tạo nền tảng cho nhiều ứng dụng NLP tiên tiến khác
- Chi tiết về các mô hình Transformer như GPT-2 và BERT. Điểm khác biệt của hai mô hình này
    - GPT-2 là một mô hình tạo ra văn bản (generative model), trong khi BERT không phải là mô hình như vậy
    - Do đó, BERT thường được sử dụng nhiều hơn cho các tác vụ văn bản, trong khi GPT-2 thường được dùng nhiều hơn cho các tác vụ tạo ra văn bản



- Dù có những mô hình mới hơn được phát triển sau đó như RoBert và GPT-3, GPT-2 và BERT vẫn còn phổ biến và được ứng dụng rộng rãi trong thực tế, một phần là nhờ vào khả năng và hiệu suất tốt của chúng
- Sử dụng GPT-2 ở đây là một lựa chọn thú vị, đặc biệt là khi nhấn mạnh rằng việc sử dụng GPT-2 cho tác vụ phân loại văn bản





b) SageMaker

- Ưu điểm của SageMaker đây là công cụ tuyệt vời cho các nhà khoa học dữ liệu:
    - Cung cấp một nền tảng quản lý hoàn toàn, giúp giảm thiểu công việc cài đặt môi trường
    - Tích hợp sẵn nhiều môi trường Conda và DocKer rất tiện lợi
    - Cho phép bạn xây dựng, huấn luyện và phát triển các mô hình học sâu nhanh chóng
- SageMaker cũng có thể được sử dụng để phát triển ứng dụng Streamlit. Điều này rất hữu ích đặc biệt trong quá trình nghiên cứu

sản phẩm, khi bạn có thể ngay lập tức xây dưng một ứng dụng trên cùng một môi trường sau khi huấn luyện mô hình.

3.2. Yêu cầu

- Tài khoản AWS: Đây là yêu cầu cơ bản vì SageMaker là dịch vụ của AWS.
- Tài khoản Google Drive: Nếu bạn muốn tận dụng môi trường Colab Notebook để huấn luyện mô hình, có một tài khoán GG drive có thể hữu ích, vì bạn có thể dễ dàng kết nối các notebook huấn luyện của mình với Google Drive
- Terminal Bash/Zsh cục bộ: Mặc dù không bắt buộc, có một terminal cục bộ có thể hữu ích, đặc biệt là đối với các tác vụ như triển khai các ứng dụng Streamlit thuận tiện để tương tác với cơ sở hạ tầng đám mây từ máy tính cục bộ

4. Cách phân chia dữ liệu (train, validation, test)

Thực hiện chia tập dữ liệu thành 3 phần:

- df_train: tập huấn luyện, chứa 80% dữ liệu
- df_val: tập validation, chứa 10% dữ liệu tiếp theo
- df_test: tập kiểm tra, chứa 10% dữ liệu còn lại

```
[ ] np.random.seed(112)
    df_train, df_val, df_test = np.split(df.sample(frac=1, random_state=42), [int(.8*len(df)), int(.9*len(df))])
    print(len(df_train),len(df_val), len(df_test))
```

```
1780 222 223
```

5. Huấn luyện và đánh giá mô hình

a) Huấn luyện

- Tận dụng sức mạnh của mô hình GPT-2 đã được tiền huấn luyện: Mô hình GPT-2 đã được huấn luyện trên một lượng dữ liệu lớn, cho phép nó hiểu và mô hình hóa ngôn ngữ một cách hiệu quả. Bằng cách sử dụng GPT-2 làm nền tảng, chúng ta có thể tận dụng được những kiến thức tiền huấn luyện này.
- Đơn giản hóa quá trình huấn luyện: Thay vì phải xây dựng một mô hình từ đầu, chúng ta chỉ cần thêm một tầng linear lên trên 12 tầng

decoder của GPT-2. Điều này đơn giản hóa đáng kể quá trình huấn luyện và giảm thiểu tài nguyên tính toán so với xây dựng một mô hình hoàn toàn mới.

- Tận dụng các tính năng của GPT-2: Các tầng decoder của GPT-2 có thể học được các đặc trưng tinh vi của ngôn ngữ, như ngữ cảnh, cấu trúc ngữ pháp, v.v. Bằng cách kết nối một tầng linear lên trên, chúng ta có thể tận dụng những đặc trưng này để thực hiện nhiệm vụ phân loại.

```python
class SimpleGPT2SequenceClassifier(nn.Module):
    def __init__(self, hidden_size: int, num_classes:int ,max_seq_len:int, gpt_model_name:str):
        super(SimpleGPT2SequenceClassifier,self).__init__()
        self.gpt2model = GPT2Model.from_pretrained(gpt_model_name)
        self.fc1 = nn.Linear(hidden_size*max_seq_len, num_classes)


    def forward(self, input_id, mask):
        """
        Args:
                input_id: encoded inputs ids of sent.
        """
        gpt_out, _ = self.gpt2model(input_ids=input_id, attention_mask=mask, return_dict=False)
        batch_size = gpt_out.shape[0]
        linear_output = self.fc1(gpt_out.view(batch_size,-1))
        return linear_output
```

GPT-2 là một mô hình tạo văn bản, trong đó nó sử dụng embedding của token cuối cùng để dự đoán các token tiếp theo. Do đó, khác với BERT sử dụng embedding của token đầu tiên, trong bước tokenization của văn bản đầu vào ở đây, chúng ta nên sử dụng token cuối cùng như sau.

```
tokenizer = GPT2Tokenizer.from_pretrained('gpt2')
tokenizer.padding_side ="left"
tokenizer.pad_token = tokenizer.eos_token
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:89: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as secret in your Google Colab and restart your session.
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
tokenizer_config.json: 100% 26.0/26.0 [00:00<00:00, 1.54kB/s]
vocab.json: 100% 1.04M/1.04M [00:00<00:00, 9.09MB/s]
merges.txt: 100% 456k/456k [00:00<00:00, 8.81MB/s]
tokenizer.json: 100% 1.36M/1.36M [00:00<00:00, 15.9MB/s]
/usr/local/lib/python3.10/dist-packages/huggingface_hub/file_download.py:1132: FutureWarning: `resume_download` is deprecated and will be removed in version 1.0.0. Downloads always resume when possi
  warnings.warn(
config.json: 100% 665/665 [00:00<00:00, 35.1kB/s]
```

Xây dựng vòng huấn luyện của bộ phân loại dựa vào dữ liệu đầu vào

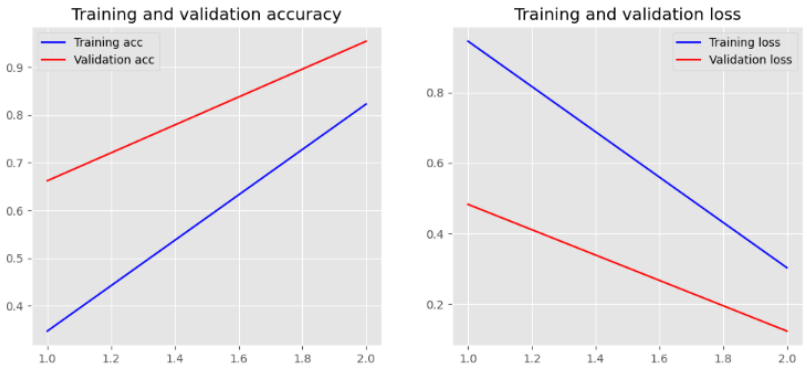
```python
def trainer(model, train_data, val_data, learning_rate, epochs):

    train, val = Dataset(train_data), Dataset(val_data)

    train_dataloader = torch.utils.data.DataLoader(train, batch_size=2, shuffle=True)
    val_dataloader = torch.utils.data.DataLoader(val, batch_size=2)

    use_cuda = torch.cuda.is_available()
    device = torch.device("cuda" if use_cuda else "cpu")

    criterion = nn.CrossEntropyLoss()
    optimizer = Adam(model.parameters(), lr= learning_rate)

    if use_cuda:

            model = model.cuda()
            criterion = criterion.cuda()

    for epoch_num in range(epochs):

            total_acc_train = 0
            total_loss_train = 0

            for train_input, train_label in tqdm(train_dataloader):

                train_label = train_label.to(device)
                mask = train_input['attention_mask'].to(device)
                input_id = train_input['input_ids'].to(device)

                output = model(input_id, mask)

                batch_loss = criterion(output, train_label)
                total_loss_train += batch_loss.item()

                acc = (output.argmax(dim=1) == train_label).sum().item()
                total_acc_train += acc

                model.zero_grad()
                batch_loss.backward()
                optimizer.step()

            total_acc_val = 0
            total_loss_val = 0
```

```python
            optimizer.step()

        total_acc_val = 0
        total_loss_val = 0

        with torch.no_grad():

            for val_input, val_label in val_dataloader:

                val_label = val_label.to(device)
                mask = val_input['attention_mask'].to(device)
                input_id = val_input['input_ids'].to(device)

                output = model(input_id, mask)

                batch_loss = criterion(output, val_label)
                total_loss_val += batch_loss.item()

                acc = (output.argmax(dim=1) == val_label).sum().item()
                total_acc_val += acc

        print(
            f'Epochs: {epoch_num + 1} | Train Loss: {total_loss_train / len(train_data): .3f} \
            | Train Accuracy: {total_acc_train / len(train_data): .3f} \
            | Val Loss: {total_loss_val / len(val_data): .3f} \
            | Val Accuracy: {total_acc_val / len(val_data): .3f}')
        train_accuracies.append(total_acc_train / len(train_data))
        val_accuracies.append(total_acc_val / len(val_data))
        train_losses.append(total_loss_train / len(train_data))
        val_losses.append(total_loss_val / len(val_data))
          # Plot metrics after trainin
EPOCHS = 2
model = SimpleGPT2SequenceClassifier(hidden_size=768, num_classes=5, max_seq_len=128, gpt_model_name="gpt2")
LR = 1e-6

trainer(model, df_train, df_val, LR, EPOCHS)
```

Kết quả:

```
model.safetensors: 100%|██████████| 548M/548M [00:04<00:00, 121MB/s]
100%|██████████| 890/890 [59:28<00:00,  4.01s/it]
Epochs: 1 | Train Loss:  0.946              | Train Accuracy:  0.347              | Val Loss:  0.483              | Val Accuracy:  0.662
100%|██████████| 890/890 [59:49<00:00,  4.03s/it]
Epochs: 2 | Train Loss:  0.303              | Train Accuracy:  0.823              | Val Loss:  0.124              | Val Accuracy:  0.955
```

```
epochs=2
x = range(1, epochs + 1)
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.plot(x, train_accuracies, 'b', label='Training acc')
plt.plot(x, val_accuracies, 'r', label='Validation acc')
plt.title('Training and validation accuracy')
plt.legend()
plt.subplot(1, 2, 2)
plt.plot(x, train_losses, 'b', label='Training loss')
plt.plot(x, val_losses, 'r', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()
plt.show()  # Display the plot
```




```
SimpleGPT2SequenceClassifier(
  (gpt2model): GPT2Model(
    (wte): Embedding(50257, 768)
    (wpe): Embedding(1024, 768)
    (drop): Dropout(p=0.1, inplace=False)
    (h): ModuleList(
      (0-11): 12 x GPT2Block(
        (ln_1): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
        (attn): GPT2Attention(
          (c_attn): Conv1D()
          (c_proj): Conv1D()
          (attn_dropout): Dropout(p=0.1, inplace=False)
          (resid_dropout): Dropout(p=0.1, inplace=False)
        )
        (ln_2): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
        (mlp): GPT2MLP(
          (c_fc): Conv1D()
          (c_proj): Conv1D()
          (act): NewGELUActivation()
          (dropout): Dropout(p=0.1, inplace=False)
        )
      )
    )
    (ln_f): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
  )
  (fc1): Linear(in_features=98304, out_features=5, bias=True)
)
```

b) Đánh giá mô hình

- Từ tập dữ liệu kiểm tra (test) và một bộ nạp dữ liệu(test_dataloader) từ dữ liệu kiểm tra. Kiểm tra sự khả dụng của GPU (CUDA) và đặt thiết bị tương ứng (device) là "cuda" hoặc "cpu". Nếu CUDA khả dụng, chuyển mô hình sang sử dụng GPU. Khởi tạo các biến để lưu trữ các nhãn dự đoán và nhãn thực tế. Lặp qua từng mẫu dữ liệu

kiểm tra, đưa vào mô hình để dự đoán nhãn và lưu trữ kết quả. Tính toán độ chính xác tổng thể của mô hình trên tập dữ liệu kiểm tra. In ra độ chính xác và trả về các nhãn thực tế và dự đoán.

```python
def evaluate (model, test_data):
  test = Dataset(test_data)
  test_dataloader = torch.utils.data.DataLoader(test, batch_size=2)
  use_cuda = torch.cuda.is_available() # Call is_available as a function
  device = torch.device("cuda" if use_cuda else "cpu")
  if use_cuda:
    model = model.cuda()
  prediction_labels =[]
  true_labels=[]
  total_acc_test = 0
  with torch.no_grad():
    for test_input, test_label in test_dataloader:
      test_label = test_label.to(device)
      mask = test_input['attention_mask'].to(device)
      input_id = test_input['input_ids'].to(device)
      output = model(input_id, mask)
      acc = (output.argmax(dim=1) == test_label).sum().item()
      total_acc_test += acc
      true_labels += test_label.cpu().numpy().flatten().tolist()
      prediction_labels += output.argmax(dim=1).cpu().numpy().flatten().tolist()
  print(f'Test Accuracy:{total_acc_test/len(test_data):.3f}')
  return true_labels, prediction_labels
```

```python
true_labels, pred_labels = evaluate(model, df_test)
```

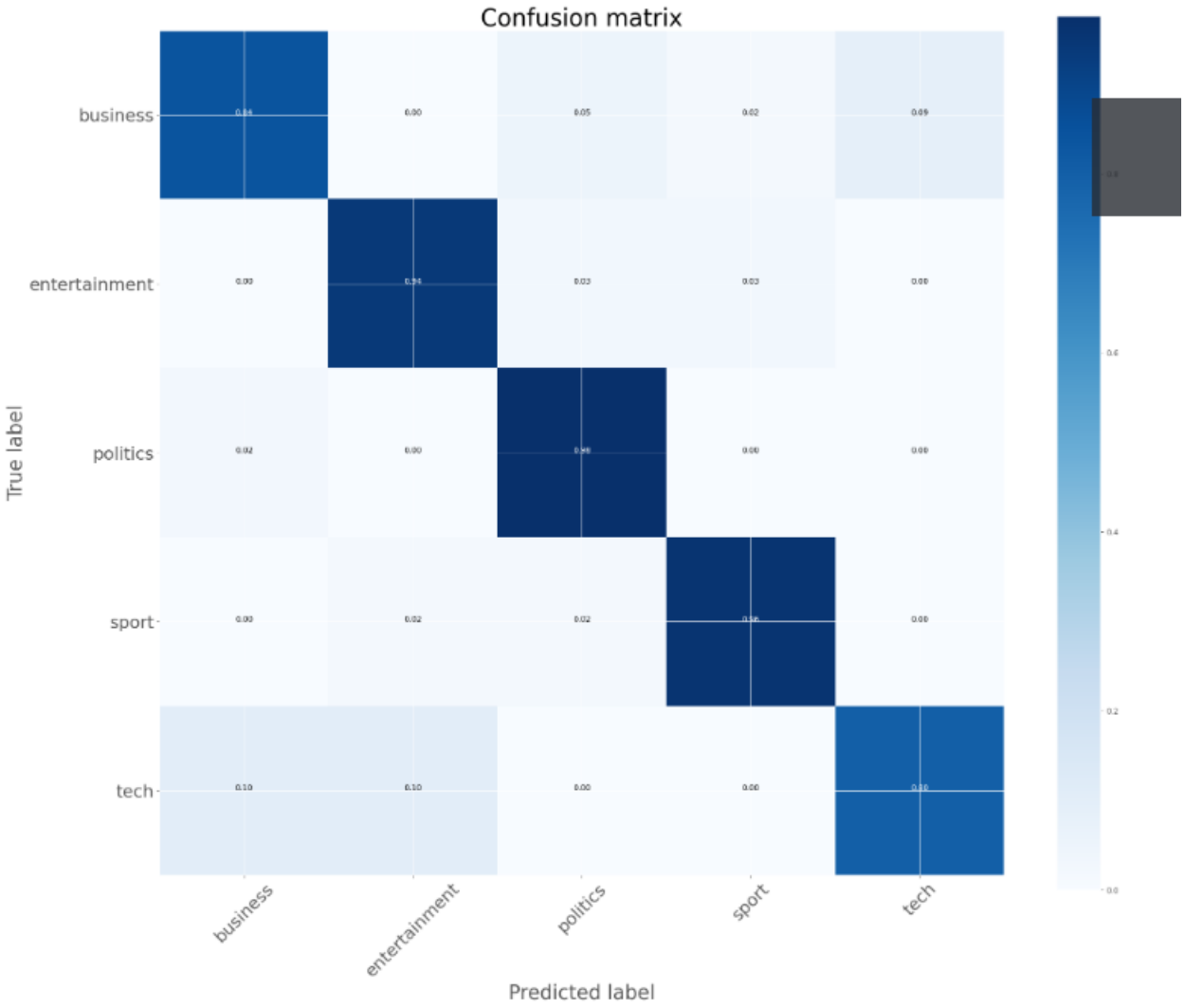
Kết quả đánh giá mô hình với khả năng chính xác lên tới 91%

```
[ ]  true_labels, pred_labels = evaluate(model, df_test)

     Test Accuracy:0.910
```

6. Dự đoán kết quả (predict)

```
#s utility function is from the sklearn docs: http://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html
def plot_confusion_matrix(cm, classes,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """

    cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title, fontsize=30)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45, fontsize=22)
    plt.yticks(tick_marks, classes, fontsize=22)

    fmt = '.2f'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.ylabel('True label', fontsize=25)
    plt.xlabel('Predicted label', fontsize=25)
```

```
import itertools
cnf_matrix = confusion_matrix(y_true=true_labels, y_pred=pred_labels)
plt.figure(figsize=(24,20))
plot_confusion_matrix(cnf_matrix, classes=list(labels.keys()), title="Confusion matrix")
plt.show()
```


Confusion matrix
business
entertainment
politics
sport
tech
True label
business
entertainment
politics
sport
tech
Predicted label

```
[32] test = "house prices show slight increase prices of homes in the uk rose a seasonally adjusted 0.5%";
     fixed_text = " ".join(test.lower().split())
     print(fixed_text)
     tokenizer = GPT2Tokenizer.from_pretrained('gpt2')
     tokenizer.padding_side = "left"
     tokenizer.pad_token = tokenizer.eos_token
     model_input = tokenizer(fixed_text, padding='max_length', max_length=128, truncation=True, return_tensors="pt")

     house prices show slight increase prices of homes in the uk rose a seasonally adjusted 0.5%
```

```
[33] mask = model_input['attention_mask'].cpu()
     input_id = model_input["input_ids"].cpu()
     output = model_test(input_id, mask)
```

```
[34] print(output)

     tensor([[-1.5847,  1.7195, -2.0726, -0.6836,  0.1081]],
            grad_fn=<AddmmBackward0>)
```

```
     prob = torch.nn.functional.softmax(output, dim=1)[0]
     print(prob)

     tensor([0.0272, 0.7411, 0.0167, 0.0670, 0.1479], grad_fn=<SelectBackward0>)
```

Kết quả dự đoán nhãn số 2 là cao nhất 0.7411 tương ứng là entertainment

- *Nhận xét:*

SimpleGPT2SequenceClassifier để dự đoán chủ đề của một đoạn văn bản. Mô hình này được xây dựng dựa trên kiến trúc GPT-2 và được huấn luyện để phân loại văn bản với 5 chủ đề: kinh doanh, chính trị, thể thao, giải trí và công nghệ

```
def predict(text):
  model_new = SimpleGPT2SequenceClassifier(hidden_size=768, num_classes=5, max_seq_len=128, gpt_model_name="gpt2")
  model_new.load_state_dict(torch.load("simple_gpt2_.pt"))
  model_new.eval()
  #test ="house prices show slight increase prices of homes in the uk rose a seasonally adjusted 0.5%"
  fixed_text = "".join(text.lower().split())
  tokenizer = GPT2Tokenizer.from_pretrained('gpt2')
  tokenizer.padding_side = "left"
  tokenizer.pad_token = tokenizer.eos_token
  model_input = tokenizer(fixed_text, padding='max_length', max_length=128, truncation=True, return_tensors="pt")
  mask = model_input['attention_mask'].cpu()
  input_id = model_input["input_ids"].cpu()

  output = model_new(input_id, mask)
  labels_map = {
    0: "business",
    1: "entertainment",
    2: "politics",
    3: "sport",
    4: "tech",
    }
  pred_label = labels_map[output.argmax(dim=1).item()]
  return pred_label
```

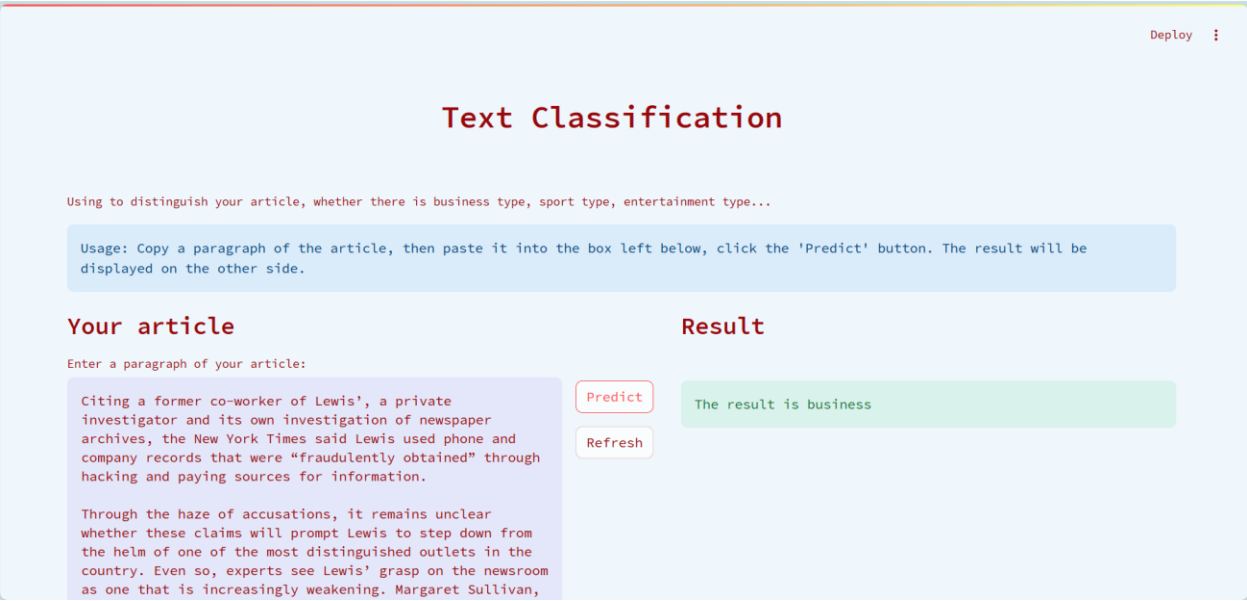
```
[36] test1 = "house prices show slight increase prices of homes in the uk rose a seasonally adjusted 0.5%"
     print(predict(test1))

     entertainment
```

7. Ứng dụng có giao diện web từ mô hình

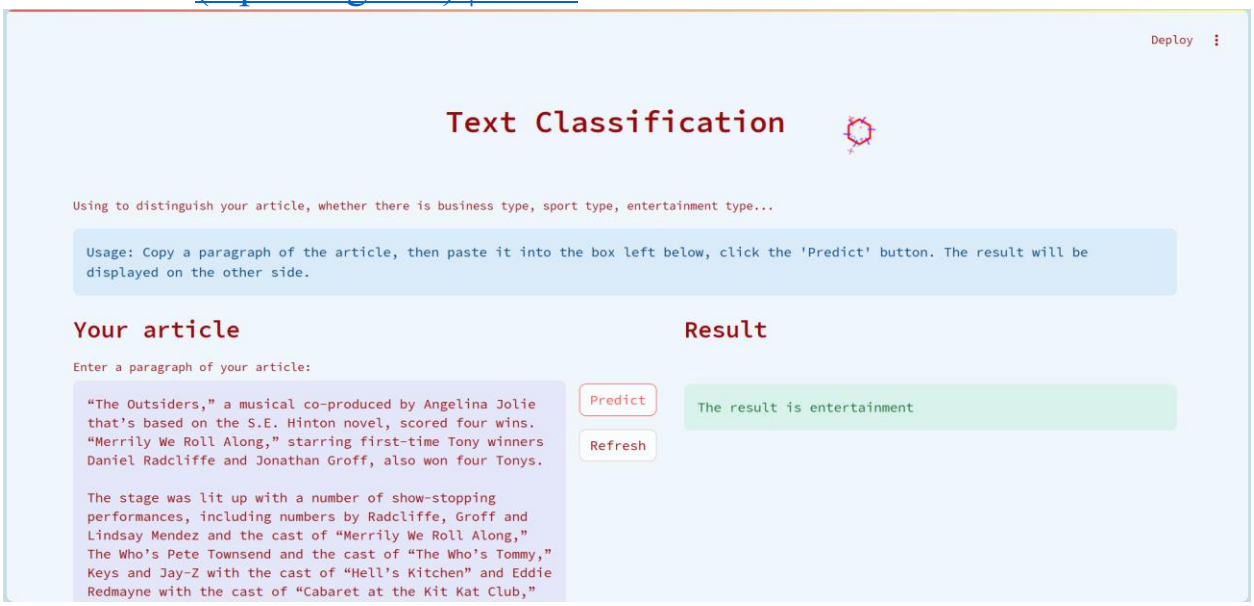
Mô hình có độ chính xác là 91%, nên vì vậy vẫn có những trường hợp đúng, và có trường hợp xảy ra sự sai sót.

- Đối với trường hợp đúng:
  - Sport: Euro 2024: Wout Weghorst breaks Poland's hearts with late winning goal for the Netherlands | CNN

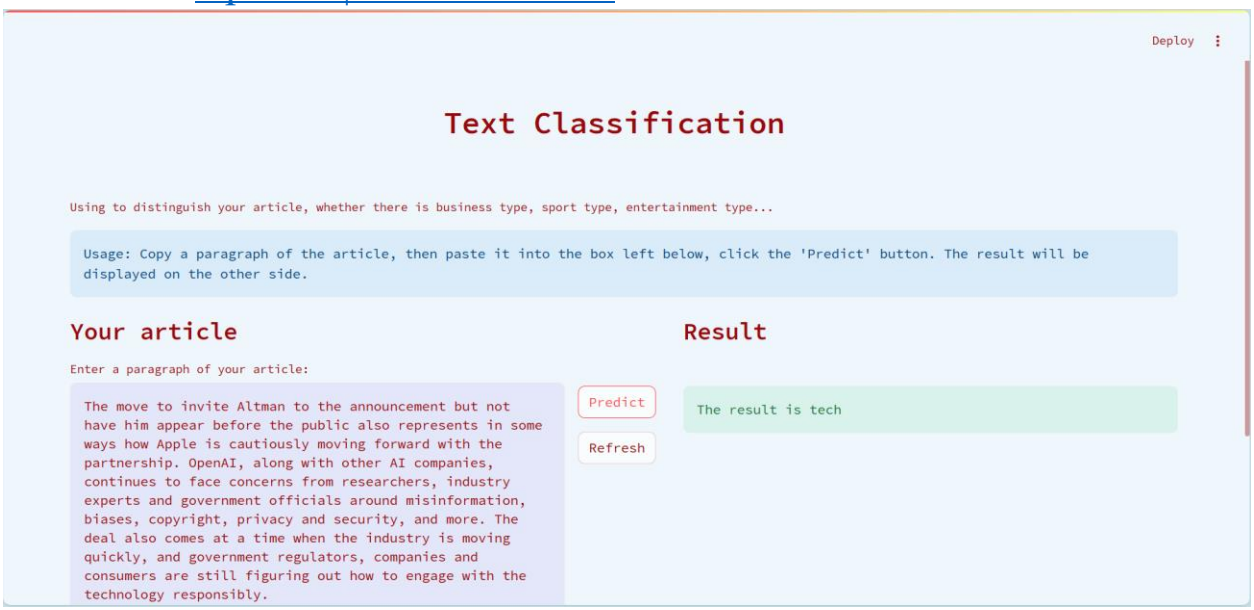
  - Business: Washington Post CEO Will Lewis faced with new allegations of phone hacking | CNN Business



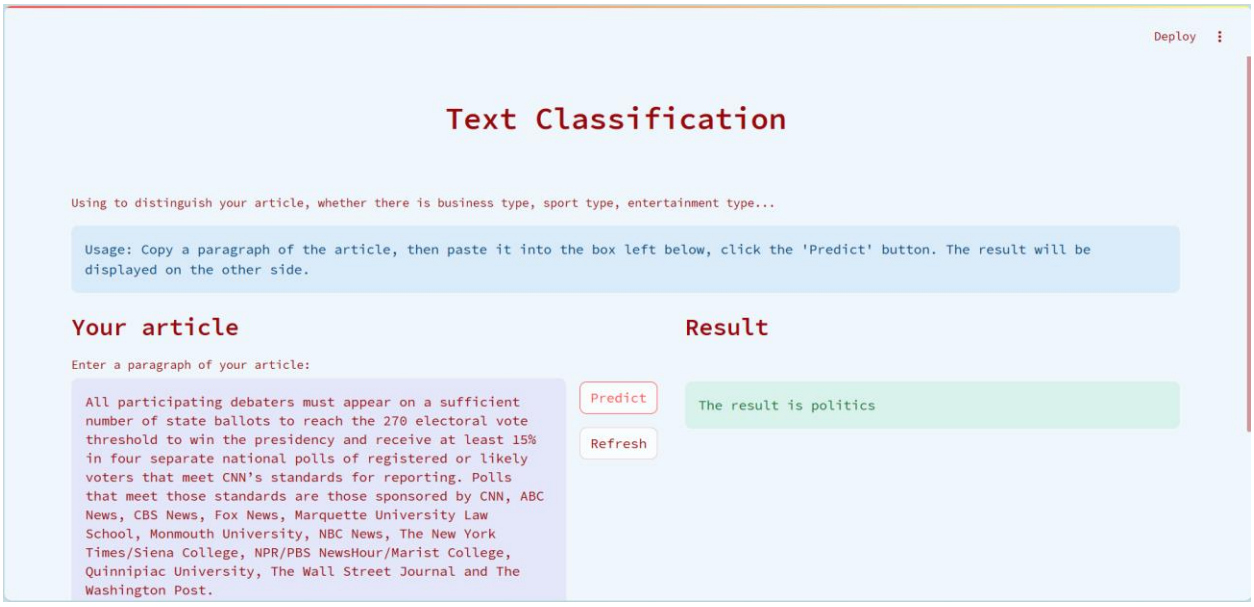
- Entertainment: Tony Awards 2024: See the full list of winners (Updating live) | CNN



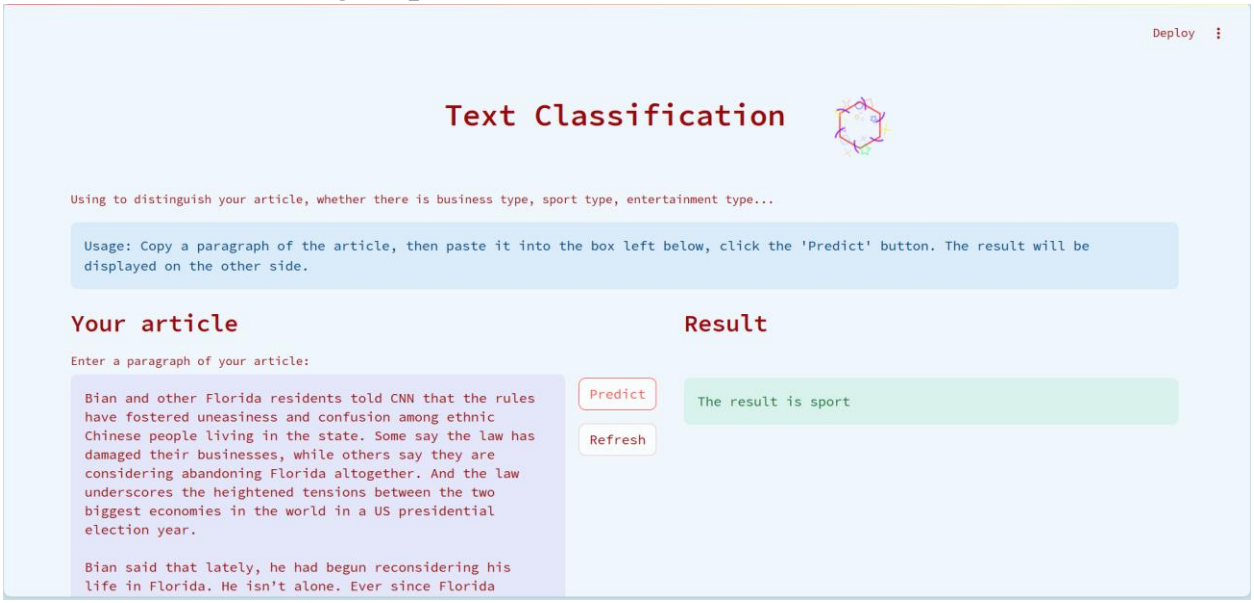
- Tech: The complicated partnership between Apple and OpenAI | CNN Business



- Politics: Biden and Trump campaigns agreed to mic muting, podiums among rules for upcoming CNN debate | CNN Politics



- Đối với trường hợp sai:



Bài báo thuộc thể loại là business: [Chinese citizens grapple with Florida law that bars them from buying property | CNN Business]