TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN, ĐHQG-HCM

KHOA TOÁN - TIN HỌC

BÁO CÁO

MÔN: XỬ LÝ NGÔN NGỮ TỰ NHIÊN

Chủ đề:

Fake News Detection Using BERT & XLNet

Sinh viên thực hiện	**MSSV**
Hồ Minh Quân	22110170
Trần Châu Phú	22110158
Trần Võ Nhật Phong	22110156

Thành phố Hồ Chí Minh, ngày 19 tháng 6 năm 2025

Mục Lục

Giới thiệu tổng quát

Với sự phát triển nhanh chóng của mạng xã hội và các nền tảng truyền thông số, tin giả (fake news) đang trở thành mối đe dọa nghiêm trọng đối với xã hội. Nó ảnh hưởng đến lòng tin công chúng, gây chia rẽ chính trị và lan truyền thông tin sai lệch trong các tình huống khẩn cấp. Trong dự án này, nhóm chúng em đã áp dụng mô hình **BERT** và **XLNet** trong xử lý ngôn ngữ tự nhiên (NLP) để giải quyết bài toán phân loại tin tức thật và giả.

I. Cơ sở lý thuyết.

1. Khái quát về Transformer.

- Transformer gồm hai khối chính:
  + **Encoder (Bộ mã hóa)**: Nhận đầu vào là chuỗi văn bản (ví dụ: câu tiếng Anh), mã hóa chúng thành một biểu diễn ngữ nghĩa ẩn (latent representation).
  + **Decoder (Bộ giải mã)**: Từ biểu diễn này, tạo ra chuỗi đầu ra (ví dụ: dịch sang tiếng Pháp).
- Sơ đồ tổng quát:

$$\boldsymbol{InputSequence} \rightarrow [\boldsymbol{EncoderxN}] \rightarrow \boldsymbol{ContextVectors} \rightarrow [\boldsymbol{DecoderxN}] \rightarrow \boldsymbol{OutputSequence}$$

- Mỗi lớp **Encoder** gồm:
  + **Multi-Head Self-Attention**: Giúp mỗi từ "nhìn" được tất cả các từ còn lại trong chuỗi đầu vào.
  + **Feed Forward Network (FFN)**: Hai lớp tuyến tính có hàm phi tuyến (ReLU).
  + **Add & Norm**: Mỗi khối có kết nối tắt (residual connection) và chuẩn hóa lớp (Layer Normalization).
  → **Input** được cộng với **Positional Encoding** để truyền thông tin thứ tự (vì không có RNN).
- Mỗi lớp **Decoder** gồm:
  + **Masked Multi-Head Self-Attention**: Tự chú ý nhưng bị "mask" tương lai (để mô hình không nhìn thấy các từ chưa sinh).
  + **Encoder-Decoder Attention**: Chú ý đến đầu ra từ Encoder (Context).
  +**Feed Forward Network** và các lớp chuẩn hóa tương tự Encoder.
  → Cuối cùng, output sẽ được đưa qua một lớp softmax để sinh từ.
- ***Cơ chế Self-Attention, linh hồn của Transformer:*** Self-Attention là một phép toán ánh xạ một chuỗi các vector đầu vào thành một chuỗi các vector đầu ra bằng cách tính toán mức độ quan trọng của từng từ đối với các từ khác trong cùng một chuỗi.
- Công thức tính: Giả sử đầu vào là một chuỗi vector ($x_1, x_2, ..., x_n$), mỗi vector sẽ được biến đổi thành:

$$\mathbf{Q=XW^Q},\ \mathbf{K=XW^K},\ \mathbf{V=XW^V}$$

$$\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}\left(\frac{\boldsymbol{QK^T}}{\sqrt{\boldsymbol{d_k}}}\right)\boldsymbol{V}$$

Trong đó:

+ $\left(\frac{\boldsymbol{QK^T}}{\sqrt{\boldsymbol{d_k}}}\right)$: tính mức độ liên quan giữa các từ

+ Softmax đảm bảo tổng xác suất = 1.

→ **Multi-Head Attention**: Dùng nhiều đầu attention song song → học nhiều kiểu quan hệ giữa từ và ngữ cảnh.

- **Positional Encoding**: giúp mô hình phân biệt vị trí các từ trong câu.
- ***Vai trò của Transformer trong NLP***:

  + Tính song song cao: Không phụ thuộc vào thời gian (khác RNN), tăng tốc huấn luyện.

  + Hiệu quả với chuỗi dài: Nhờ self-attention, mô hình nhớ toàn bộ chuỗi đầu vào.

  + Tổng quát hóa tốt: Dễ mở rộng cho nhiều task như dịch máy, tóm tắt, sinh văn bản, v.v.

2. Mô hình BERT (Bidirectional Encoder Representations from Transformer).

- **BERT** là một mô hình ngôn ngữ được giới thiệu bởi Google vào năm 2018. Điểm đột phá của BERT là khả năng **hiểu ngữ cảnh hai chiều** (trái và phải) của từ trong câu, nhờ vào kiến trúc **Transformer Encoder**.

  → BERT không sinh văn bản mà tập trung **hiểu ý nghĩa và ngữ cảnh** của từ/ngữ/câu.
- BERT được huấn luyện trước (pre-train) trên tập dữ liệu lớn với hai mục tiêu chính:
  - Masked Language Model (MLM):

    + Một số từ trong câu đầu vào được thay thế bằng token [MASK] (~15%).

    + Mô hình học cách **dự đoán từ bị che** dựa trên **cả ngữ cảnh trước và sau**.

    →Nhờ đó, BERT hiểu sâu sắc mối quan hệ giữa các từ trong cả hai chiều.
  - Next Sentence Prediction (NSP):

    + Nhằm giúp BERT hiểu mối quan hệ giữa hai câu liên tiếp.

    + Mỗi lần huấn luyện, BERT nhận vào hai câu A và B: 50% là câu liên tiếp thật hay 50% là câu ngẫu nhiên.

    + Nhiệm vụ: Dự đoán xem B có phải là câu tiếp theo của A không?
- ***Cơ chế hiểu ngữ cảnh hai chiều của BERT:*** Khác với các mô hình ngôn ngữ truyền thống chỉ nhìn một chiều (trái → phải hoặc phải → trái), BERT sử dụng self-attention trong cả hai chiều cùng lúc.

  → Do đó BERT có thể hiểu được nghĩa mơ hồ của từ trong từng ngữ cảnh và mối quan hệ logic giữa các thành phần trong câu.

  →BERT hiểu được đúng nghĩa tùy theo ngữ cảnh xung quanh.
- ***Fine-tuning BERT cho các tác vụ phân loại văn bản:*** Sau khi pre-train xong, BERT có thể được tinh chỉnh (fine-tuned) cho các bài toán cụ thể bằng cách thêm một tầng output phía trên và huấn luyện lại.
- Các bước tinh chỉnh:

  Bước 1: **Thêm một lớp phân loại (dense + softmax)** trên token [CLS] đầu tiên – token đặc biệt đại diện cho toàn bộ câu.

  Bước 2: **Huấn luyện mô hình** với dữ liệu nhãn thật (label) cho từng tác vụ:

  + Phân loại cảm xúc (sentiment classification).

  + Phân loại chủ đề văn bản.

  + Phát hiện spam, v.v.

→BERT là nền tảng cho rất nhiều mô hình NLP hiện đại như RoBERTa, DistilBERT, ALBERT... và được áp dụng rộng rãi trong các hệ thống tìm kiếm, chatbot, phân tích cảm xúc, trích xuất thực thể, v.v. Sự hiểu ngôn ngữ hai chiều giúp BERT nổi bật so với các mô hình trước đây.

3. Mô hình XLNet.

- **XLNet** là một mô hình ngôn ngữ được giới thiệu năm 2019 bởi nhóm nghiên cứu Google Brain và Carnegie Mellon University.
- Mục tiêu của XLNet là kết hợp ưu điểm của **BERT** (hiểu ngữ cảnh hai chiều) và **Transformer-XL** (xử lý chuỗi dài hiệu quả), đồng thời khắc phục một số hạn chế của BERT trong việc giả định độc lập giữa các token bị che (masked).
- **Lấy cảm hứng từ Transformer-XL :** Transformer-XL là một phiên bản cải tiến của Transformer nhằm xử lý phụ thuộc dài hạn (long-term dependency).
  + Điểm nổi bật của Transformer-XL:
    • **Cơ chế ghi nhớ (memory)**: Giữ lại trạng thái của các đoạn trước đó → giúp mô hình học được các quan hệ vượt ra ngoài giới hạn của một đoạn ngắn.
    • **Segment-level Recurrence**: Cho phép truyền thông tin giữa các đoạn liên tiếp.

→ XLNet kế thừa cơ chế này để tăng khả năng nắm bắt ngữ cảnh xa trong văn bản.
- ***Permutation Language Modeling (PLM):*** Thay vì dùng **Masked Language Model** như BERT (ẩn đi các từ), XLNet sử dụng cơ chế **PLM** – mô hình hóa tất cả các khả năng sắp xếp (hoán vị) của chuỗi đầu vào.
- ***So sánh với BERT:***

Khía Cạnh	BERT	XLNet
Cách huấn luyện	Masked Language Modeling	Permutation Language Modeling
Tính hai chiều	Có (nhưng giả định độc lập từ bị che)	Có (với mối phụ thuộc chặt chẽ giữa các từ)
Xử lý quan hệ dài	Giới hạn trong 512 token	Kế thừa từ Transformer-XL → ghi nhớ được dài hơn
Dự đoán ngữ cảnh	Dựa trên các từ xung quanh	Dựa trên mọi thứ tự hoán vị → học được nhiều cấu trúc

- ***Ưu điểm của XLNet:***
  + Khắc phục hạn chế của BERT:
    • Không cần dùng token [MASK] → không có sự khác biệt giữa giai đoạn huấn luyện và dự đoán.
    • Tránh giả định độc lập giữa các từ bị che.
  + Hiểu được ngữ cảnh linh hoạt:
    • Nhờ học từ mọi thứ tự có thể có → XLNet hiểu rõ mối quan hệ giữa các từ trong các cấu trúc ngữ pháp khác nhau.
  + Ghi nhớ ngữ cảnh dài:
    • Nhờ sử dụng cơ chế memory từ Transformer-XL, XLNet có thể xử lý các mối quan hệ giữa từ trong các đoạn dài hơn (ví dụ: đoạn văn, chương sách...).

→ **XLNet** là bước tiến tiếp theo sau **BERT**, kết hợp giữa ý tưởng "hiểu ngữ cảnh hai chiều" và khả năng xử lý chuỗi dài từ Transformer-XL. Nhờ đó, XLNet vượt trội trong nhiều tác vụ NLP như phân loại văn bản, trả lời câu hỏi, suy luận ngôn ngữ và các bài toán cần hiểu mối liên hệ giữa các phần xa trong văn bản.

4. Phương pháp đánh giá mô hình phân loại.

- ***Ma trận nhầm lẫn (Confusion Matrix):***

	Predicted: Negative	Predicted: Positive
Actual: Negative	True Negative(TN)	False Positive(FP)
Actual: Positive	False Negative(FN)	True Positive(TP)

Trong đó:
- TN: Dự đoán đúng trường hợp âm tính.
- TP: Dự đoán đúng trường hợp dương tính.
- FP: Dự đoán nhầm âm tính thành dương (dương giả).
- FN: Dự đoán nhầm dương tính thành âm (âm giả).

Cách hiểu: + **Hàng**: Giá trị thực tế (Actual).
+ **Cột**: Dự đoán của mô hình (Predicted).

- Dựa vào ma trận để tìm ra các chỉ số đánh giá:

+ **Accuracy** – Độ chính xác: Tỷ lệ tổng số dự đoán đúng trên toàn bộ dữ liệu.

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

→ Được dùng khi **dữ liệu cân bằng**.

+ **Precision** – Độ chính xác theo lớp dương: Tỷ lệ các dự đoán dương là **đúng**.

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

→ Quan trọng khi giảm nhầm lẫn **dương giả (**FP thấp**).**

+ Recall – Độ bao phủ (hay Sensitivity): Tỷ lệ các mẫu dương thực sự được phát hiện **đúng**.

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

→ Quan trọng khi cần phát hiện hết các trường hợp dương.

+ **F1-score** – Trung bình điều hòa giữa Precision và Recall.

$$\text{F1-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

→ Cân bằng giữa Precision và Recall, đặc biệt hữu ích khi dữ liệu không cân bằng.

+ **Support** – Số lượng mẫu thuộc mỗi lớp.
→ Đánh giá mức độ đóng góp của từng lớp.

II. Chuẩn bị và tiền xử lý dữ liệu.

1. Chuẩn bị dữ liệu.

- Dữ liệu được lấy từ hai tập tin MisinfoSuperset_TRUE.csv và MisinfoSuperset_FAKE.csv, sau đó được tải và giải nén thông qua thư viện gdown. Tin thật đến từ các báo uy tín, còn tin giả từ các trang cực đoan và tập dữ liệu công khai.
- Tải file .zip từ Google Drive → giải nén → đọc file CSV thật & giả vào true_df, fake_df.

```
[ ]  true_df = pd.read_csv("data2/DataSet_Misinfo_TRUE.csv")
     fake_df = pd.read_csv("data1/DataSet_Misinfo_FAKE.csv")
```

```
     true_df.head()
```

	Unnamed: 0	text
0	0	The head of a conservative Republican faction ...
1	1	Transgender people will be allowed for the fir...
2	2	The special counsel investigation of links bet...
3	3	Trump campaign adviser George Papadopoulos tol...
4	4	President Donald Trump called on the U.S. Post...

```
[ ]  fake_df.head()
```

	Unnamed: 0	text
0	0	Donald Trump just couldn t wish all Americans ...
1	1	House Intelligence Committee Chairman Devin Nu...
2	2	On Friday, it was revealed that former Milwauk...
3	3	On Christmas day, Donald Trump announced that ...
4	4	Pope Francis used his annual Christmas Day mes...

2. Tiền xử lý dữ liệu.

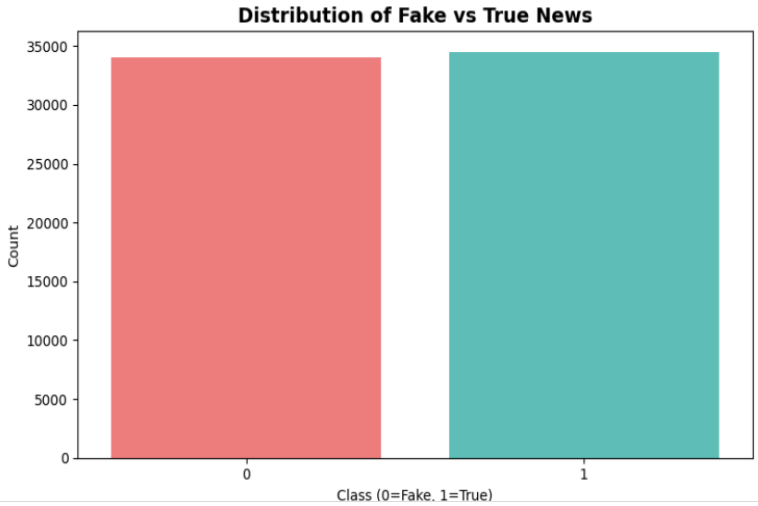
- Gán nhãn class = 1 cho tin thật, class = 0 cho tin giả, loại bỏ giá trị thiếu và trùng lặp.
- Dùng ***hàm def clean_text_for_transformers(text)*** để chuẩn hóa văn bản (loại bỏ URL, HTML, ký tự đặc biệt, chuyển chữ thường, mở rộng từ viết tắt).

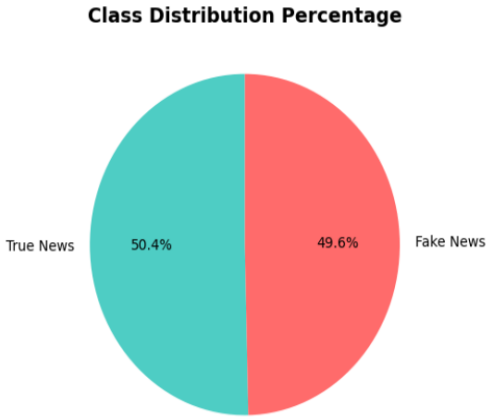
	text	class	text_clean	text_length	word_count
0	Teens walk free after gang-rape conviction Jud...	0	teens walk free after gang rape conviction jud...	879	154
1	Republican presidential candidate Donald Trump...	1	republican presidential candidate donald trump...	2491	417
2	By Gordon Duff, Senior Editor on October 30, 2...	0	by gordon duff senior editor on october consti...	14935	2493
3	President Barack Obama on Friday signed into l...	1	president barack obama on friday signed into l...	4086	664
4	United Nations Secretary-General Antonio Guter...	1	united nations secretary general antonio guter...	395	64

- Tính toán độ dài văn bản, số lượng từ để phục vụ phân tích.

3. Exploratory Data Analysis (EDA).

a. Phân bô nhãn (class distribution) bằng cách vẽ biểu đồ cột và tròn.





- Nhận xét:
  + **Fake News (class 0):** chiếm khoảng **49.6%**
  + **True News (class 1):** chiếm khoảng **50.4%**
  → **Tập dữ liệu gần như cân bằng**, chênh lệch chỉ khoảng **0.8%**, rất nhỏ.

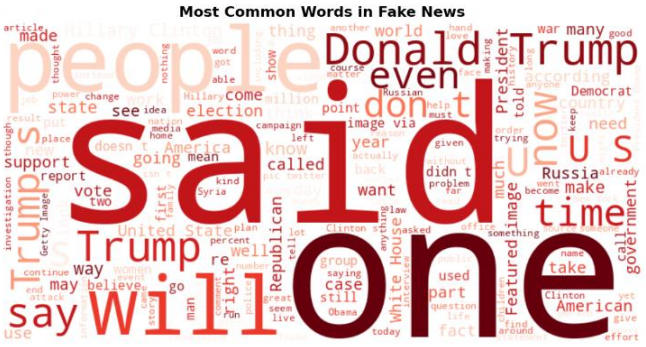
b. Text Length Analysis: số từ, số ký tự, phân tích theo lớp…

Text length statistics:

class	count	mean	std	min	25%	50%	75%	max
0	34034.0	426.777957	625.650201	1.0	97.0	331.0	509.00	24234.0
1	34526.0	537.157302	567.222748	1.0	181.0	391.0	724.75	14641.0

- Nhận xét:
  + **Tin thật có xu hướng dài hơn, nội dung phong phú hơn** → đây là đặc trưng có thể khai thác trong mô hình.
  + Cần **xem xét cắt ngưỡng độ dài tối đa** để loại bỏ các văn bản quá dài (outlier).

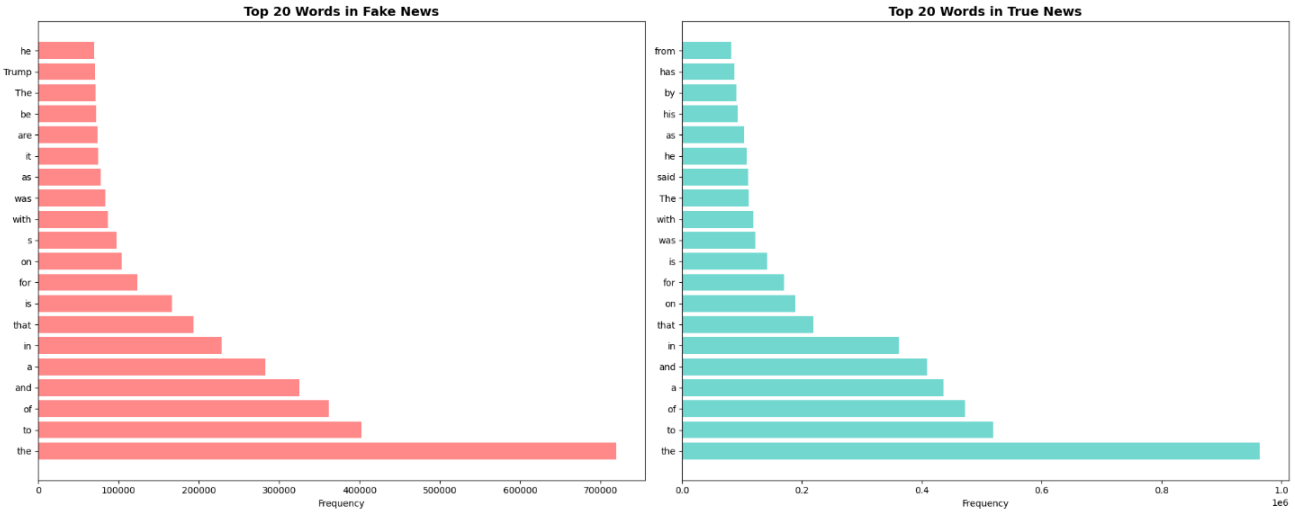
c. Word Cloud: để tạo từ khóa nổi bật trong từng lớp.



- Nhận xét:
  + **Fake news** thường dùng từ ngữ mang tính cảm xúc mạnh, tập trung vào **nhân vật và tranh luận chính trị**.

+ **True news** thì mang tính **trung lập, khách quan, thông tin cụ thể rõ ràng**, phù hợp với tính chất báo chí chính thống.
→ Phân tích từ vựng giúp mô hình học máy phân biệt tốt hơn giữa hai loại văn bản.

d. Most Common Word Analysis: từ phổ biến hay xuất hiện.



- Nhận xét:
+ **Fake News** có xu hướng tập trung vào cá nhân (như Trump) và các nội dung liên quan đến phát ngôn, gây tranh cãi.
+ **True News** thường được viết theo lối báo chí chính thống, thông tin cụ thể, trích dẫn phát ngôn chính xác.

→Thông qua EDA ta phần nào thấy được sự khác biệt rõ rệt về đặc trưng ngôn ngữ giữa tin thật và tin giả.

III. Xây dựng và đánh giá mô hình.

1. Xây dựng mô hình.

- Dùng tokenizer: bert-base-cased và xlnet-base-cased.
- Tạo lớp ***class FakeNewsDataset(Dataset)*** để chuẩn hóa dữ liệu cho mô hình Transformer giúp cho:
  + Biến văn bản thành định dạng đầu vào chuẩn cho mô hình NLP.
  + Dễ dàng sử dụng với `DataLoader` của PyTorch để huấn luyện theo mini-batch.
  + Tùy chỉnh tokenizer, max_len, nhãn, số lượng mẫu...

```python
class FakeNewsDataset(Dataset):
    def __init__(self, texts, labels, tokenizer, max_len):
        self.texts = texts
        self.labels = labels
        self.tokenizer = tokenizer
        self.max_len = max_len

    def __len__(self):
        return len(self.texts)

    def __getitem__(self, idx):
        text = str(self.texts[idx])
        label = self.labels[idx]

        encoding = self.tokenizer(
            text,
            add_special_tokens=True,
            max_length=self.max_len,
            padding='max_length',
            truncation=True,
            return_token_type_ids=True,
            return_attention_mask=True,
            return_tensors='pt'
        )

        return {
            'input_ids': encoding['input_ids'].squeeze(0),
            'attention_mask': encoding['attention_mask'].squeeze(0),
            'token_type_ids': encoding.get('token_type_ids', torch.zeros(self.max_len, dtype=torch.long)).squeeze(0),
            'label': torch.tensor(label, dtype=torch.long)
        }
```

- Hàm ***create_data_loaders(...) :*** tạo DataLoader cho train/val/test từ tokenizer tương ứng (BERT hoặc XLNet).

```python
train_dataset = FakeNewsDataset(
    texts=train_df['text'].values,
    labels=train_df['encoded_label'].values,
    tokenizer=tokenizer,
    max_len=MAX_LEN
)

val_dataset = FakeNewsDataset(
    texts=val_df['text'].values,
    labels=val_df['encoded_label'].values,
    tokenizer=tokenizer,
    max_len=MAX_LEN
)

test_dataset = FakeNewsDataset(
    texts=test_df['text'].values,
    labels=test_df['encoded_label'].values,
    tokenizer=tokenizer,
    max_len=MAX_LEN
)
```

- Sử dụng ***RandomSampler*** cho tập train giúp tăng tính ngẫu nhiên trong quá trình học.

```python
# Training sampler and DataLoader - shuffles data at each epoch
train_sampler = RandomSampler(train_dataset)
train_dataloader = DataLoader(
```

- Riêng val và test phải dùng ***SequentialSampler*** do không cần shuffle vì không cần học nữa.
- Mô hình được khởi tạo bằng ***AutoModelForSequenceClassification*** (BERT) và ***XLNetForSequenceClassification*** với num_labels = 2.

```
bert_model = AutoModelForSequenceClassification.from_pretrained(
    bert_model_name,
    num_labels=num_classes,
    output_attentions=False,
    output_hidden_states=False
)
bert_model = bert_model.to(device)
xlnet_model = XLNetForSequenceClassification.from_pretrained(
    xlnet_model_name,
    num_labels=num_classes,
    output_attentions=False,
    output_hidden_states=False
)
xlnet_model = xlnet_model.to(device)
```

- Ta tạo ***def train_model(model, train_dataloader, val_dataloader, ...)*** để huấn luyện 2 mô hình với quy trình:
  + Đặt mô hình ở chế độ huấn luyện (***model.train()***), duyệt qua các batch.
  + Với mỗi batch: tính loss, thực hiện ***loss.backward()***, cắt gradient, cập nhật trọng số bằng ***optimizer.step()*** và điều chỉnh bằng scheduler.
  + Kết thúc mỗi epoch, tính loss, cập nhật trọng số, đánh giá trên validation.
  + **Early stopping** nếu loss không cải thiện.
  + Lưu model tốt nhất (best_model.pt).
- Ta setup siêu tham số:
  • Dùng ***AdamW*** optimizer + ***get_linear_schedule_with_warmup()*** để setup, tạo optimizer & scheduler riêng cho BERT & XLNet.

```
bert_optimizer = AdamW(bert_model.parameters(), lr=learning_rate, eps=epsilon)
bert_total_steps = len(bert_train_dataloader) * epochs
bert_scheduler = get_linear_schedule_with_warmup(


xlnet_optimizer = AdamW(xlnet_model.parameters(), lr=learning_rate, eps=epsilon)
xlnet_total_steps = len(xlnet_train_dataloader) * epochs
xlnet_scheduler = get_linear_schedule_with_warmup(
```

```
# Set hyperparameters
epochs = 3
learning_rate = 2e-5
epsilon = 1e-8
warmup_steps = 0
```

- Sau khi tiến hành train 2 model ta thu được:

```
Training XLNet model...

============================== Epoch 1/3 ==============================
Training...
100%|██████████| 1500/1500 [24:22<00:00,  1.03it/s]
Average training loss: 0.0948
Validating...
100%|██████████| 322/322 [02:02<00:00,  2.63it/s]
Validation Loss: 0.1292
Validation Accuracy: 0.9552
Validation loss improved, saving model to best_model.pt

============================== Epoch 2/3 ==============================
Training...
100%|██████████| 1500/1500 [24:21<00:00,  1.03it/s]
Average training loss: 0.0961
Validating...
100%|██████████| 322/322 [02:01<00:00,  2.64it/s]
Validation Loss: 0.1292
Validation Accuracy: 0.9552
No improvement in validation loss for 1 epoch(s)

============================== Epoch 3/3 ==============================
Training...
100%|██████████| 1500/1500 [24:22<00:00,  1.03it/s]
Average training loss: 0.0948
Validating...
100%|██████████| 322/322 [02:01<00:00,  2.64it/s]
Validation Loss: 0.1292
Validation Accuracy: 0.9552
No improvement in validation loss for 2 epoch(s)
Early stopping triggered after 3 epochs
```

```
Training BERT model...

============================== Epoch 1/3 ==============================
Training...
100%|████████| 1500/1500 [17:28<00:00,  1.43it/s]
Average training loss: 0.1989
Validating...
100%|████████| 322/322 [01:12<00:00,  4.47it/s]
Validation Loss: 0.1325
Validation Accuracy: 0.9516
Validation loss improved, saving model to best_model.pt

============================== Epoch 2/3 ==============================
Training...
100%|████████| 1500/1500 [17:26<00:00,  1.43it/s]
Average training loss: 0.0838
Validating...
100%|████████| 322/322 [01:11<00:00,  4.48it/s]
Validation Loss: 0.1241
Validation Accuracy: 0.9613
Validation loss improved, saving model to best_model.pt

============================== Epoch 3/3 ==============================
Training...
100%|████████| 1500/1500 [17:28<00:00,  1.43it/s]
Average training loss: 0.0384
Validating...
100%|████████| 322/322 [01:12<00:00,  4.46it/s]
Validation Loss: 0.1663
Validation Accuracy: 0.9633
No improvement in validation loss for 1 epoch(s)
```

2. Đánh giá mô hình.

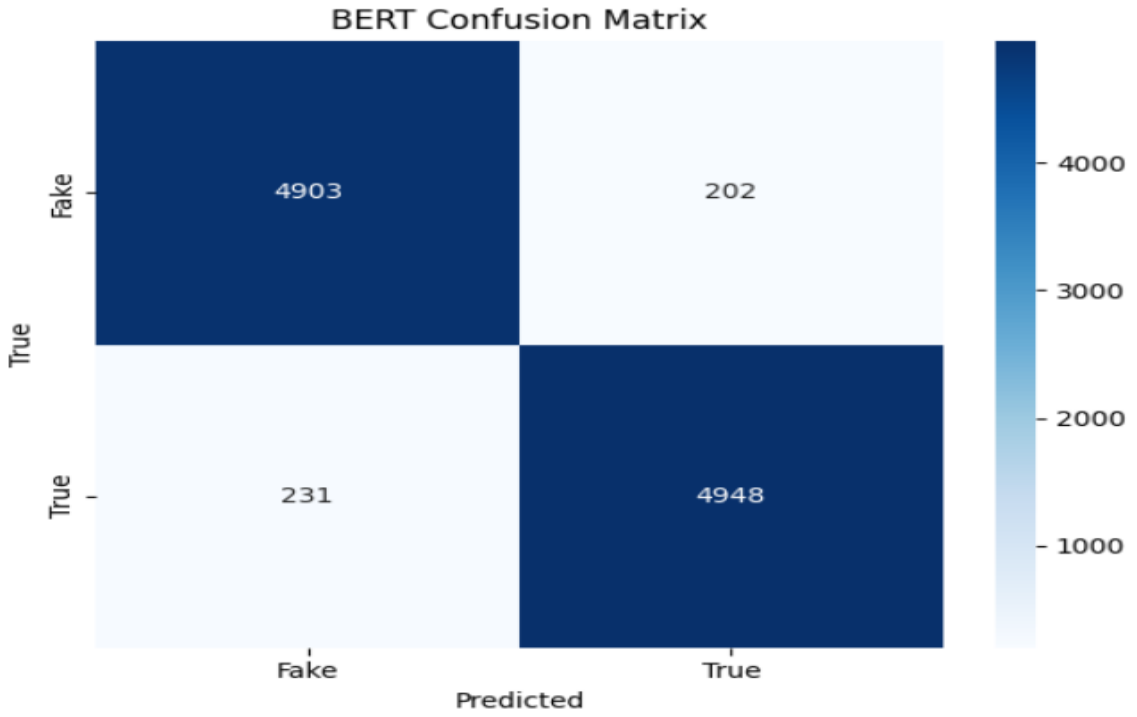
- Sau khi huấn luyện, mô hình được đánh giá bằng hàm ***evaluate_model()***:

+ Hàm này tính accuracy, in ra báo cáo phân loại (classification_report gồm precision, recall, f1-score).

```
BERT Classification Report:
              precision    recall  f1-score   support

        Fake       0.96      0.96      0.96      5105
        True       0.96      0.96      0.96      5179

    accuracy                           0.96     10284
   macro avg       0.96      0.96      0.96     10284
weighted avg       0.96      0.96      0.96     10284


XLNet Classification Report:
              precision    recall  f1-score   support

        Fake       0.94      0.98      0.96      5105
        True       0.97      0.94      0.96      5179

    accuracy                           0.96     10284
   macro avg       0.96      0.96      0.96     10284
weighted avg       0.96      0.96      0.96     10284
```

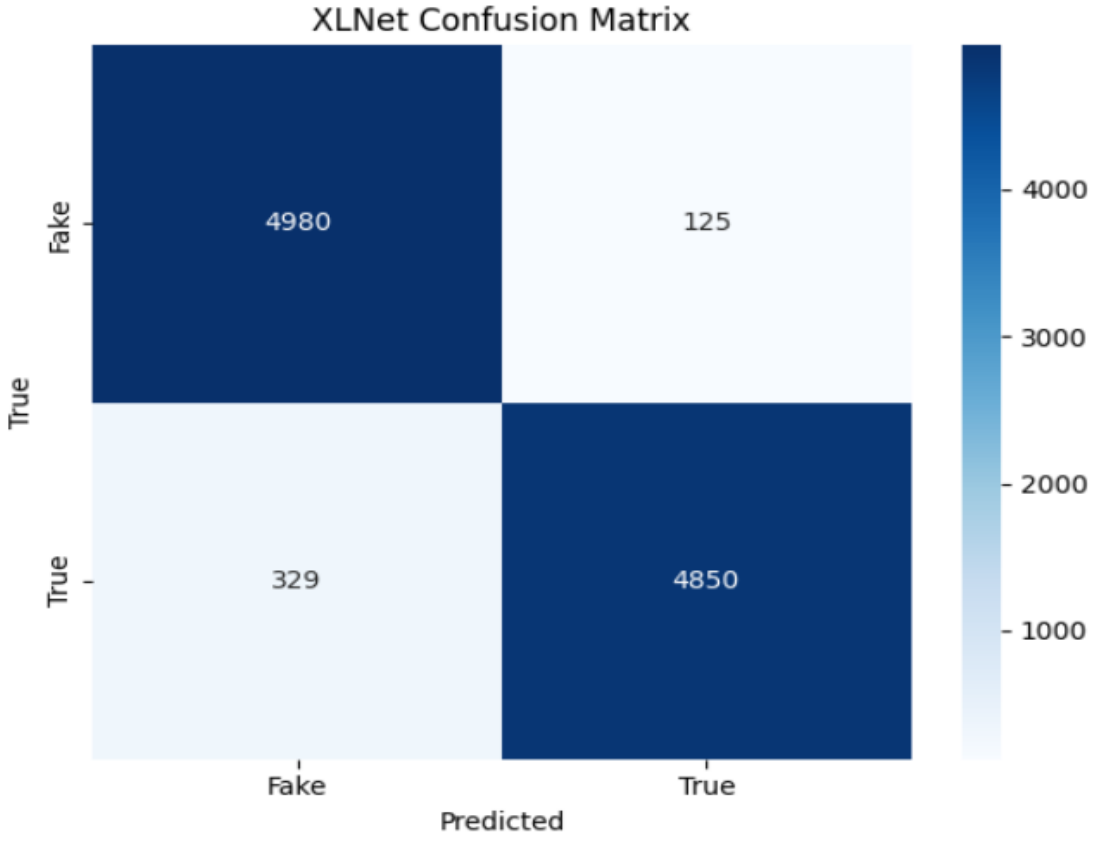
→ Cả **BERT và XLNet** đều đạt **accuracy = 0.96 (96%)** → rất cao, cho thấy cả hai mô hình đều phân loại tốt.
→ **BERT**: Cân bằng giữa precision và recall.
→ **XLNet** có xu hướng **ưu tiên recall cho "Fake"** và **precision cho "True"**, phù hợp nếu ta muốn **phát hiện càng nhiều tin giả càng tốt**, chấp nhận một ít lỗi.

+ Tính **confusion matrix** cho cả BERT và XLNet.



- Nhận xét:
  • **True Positives (TP) – 4948**: Số lượng tin thật được mô hình dự đoán đúng là **rất cao**.
  • **True Negatives (TN) – 4903**: Mô hình cũng phân loại chính xác rất nhiều tin giả.
  • **False Positives (FP) – 202**: Có một **số ít tin giả** bị mô hình **nhầm là tin thật**.
  • **False Negatives (FN) – 231**: Cũng có một số **tin thật** bị dự đoán **sai thành tin giả**, nhưng tỉ lệ không lớn.

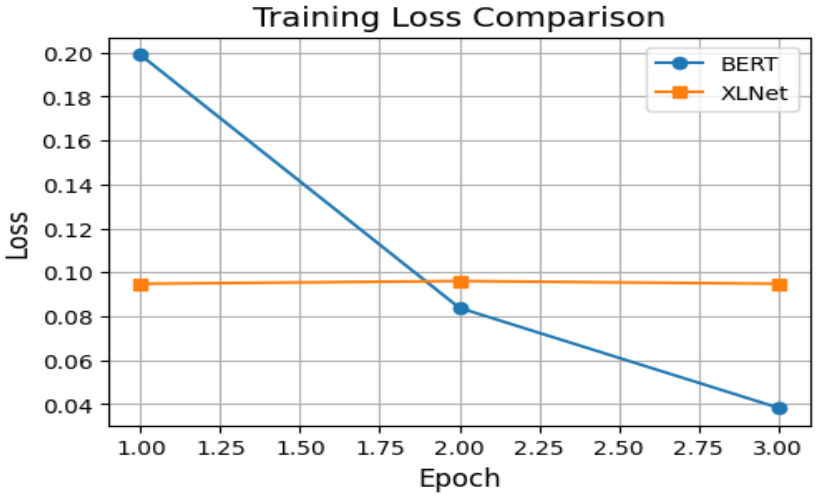
→ Mô hình BERT đạt hiệu năng **rất tốt** trong bài toán phân loại văn bản thật và giả.



- Nhận xét:
  • **True Positives (TP) – 4850**: Số lượng tin thật được mô hình dự đoán đúng là **rất cao**.
  • **True Negatives (TN) – 4980**: Mô hình cũng phân loại chính xác rất nhiều tin giả.
  • **False Positives (FP) – 125**: Có một **số ít tin giả** bị mô hình **nhầm là tin thật**.
  • **False Negatives (FN) – 329**: Cũng có một số **tin thật** bị dự đoán **sai thành tin giả**, nhưng tỉ lệ không lớn.

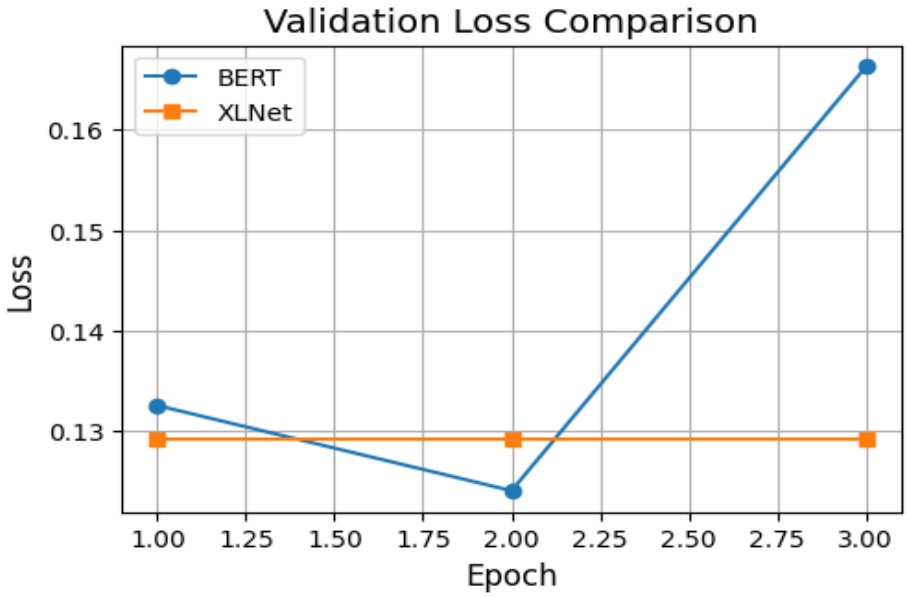
→Mô hình **XLNet hoạt động rất hiệu quả** trong bài toán phân loại tin tức giả.

- Vẽ biểu đồ để minh họa:
  + Training loss qua epoch.



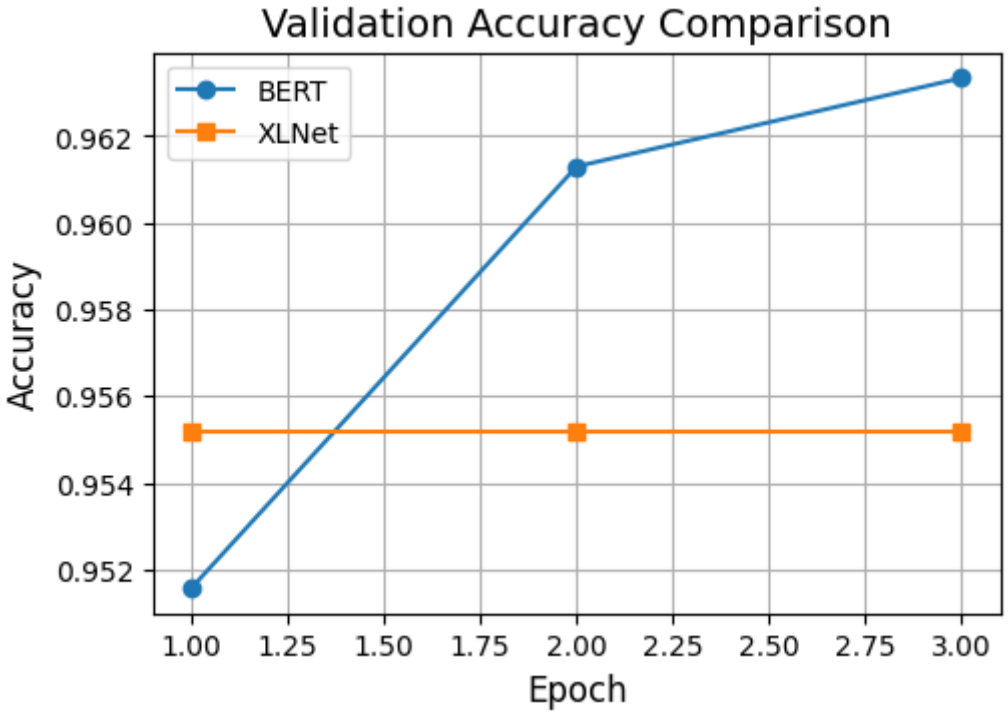
→ **BERT có hiệu quả huấn luyện rõ rệt hơn XLNet** trên biểu đồ này.

+ Validation loss qua epoch.



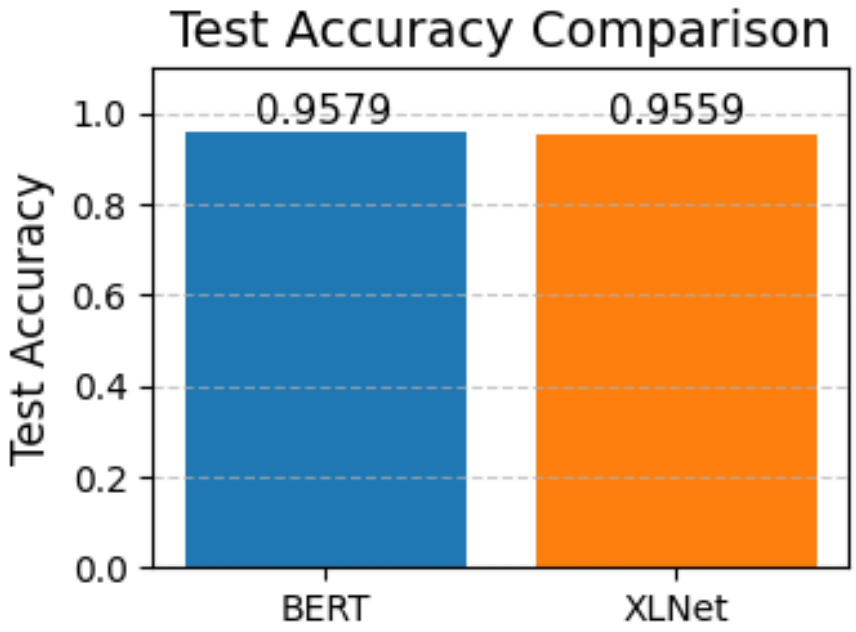
→ **XLNet** cho thấy khả năng **giữ ổn định và tránh overfitting tốt hơn** so với BERT

+ Validation accuracy qua epoch.



→ **BERT**: Có xu hướng học tốt hơn khi được huấn luyện thêm → phù hợp khi có thể sử dụng nhiều epoch.
→ **XLNet**: Ổn định, nhưng **ít cải thiện theo thời gian**, có thể tốt nếu muốn mô hình đơn giản, nhanh, ít bị overfit.

+ Biểu đồ so sánh test accuracy giữa 2 mô hình.



→Cả hai mô hình đều đạt độ chính xác rất cao trên tập test.

→Sau khi đánh giá xong mô hình, ta tiến hành dự đoán, kiểm tra mô hình có tốt không.

- Hàm ***def predict_fake_news(text, model, tokenizer, ...)*** được thực hiện theo quy trình:
  + Nhận 1 văn bản đầu vào → làm sạch → token hóa → model dự đoán.
  + In kết quả dự đoán & độ tin cậy (confidence).
  + Hàm ***infer_and_print_random(...)*** chọn ngẫu nhiên 5 dòng từ DataFrame → in ra text + dự đoán + nhãn thực.

```
BERT Model Predictions:
Text: President Trump signed a new economic relief bill yesterday to support small businesses.
Prediction: Real (Confidence: 0.9788)
--------------------------------------------------------------------------------
Text: Fake news claims that Trump has secret plans to leave the country after the election.
Prediction: Real (Confidence: 0.9993)
--------------------------------------------------------------------------------
Text: Trump held a rally attended by thousands of supporters despite the ongoing pandemic.
Prediction: Real (Confidence: 0.9934)
--------------------------------------------------------------------------------
Text: A viral post falsely states that Trump endorsed a fake cure for COVID-19.
Prediction: Real (Confidence: 0.9762)
--------------------------------------------------------------------------------
Text: The latest polls show Trump's approval rating increasing steadily over the past month.
Prediction: Fake (Confidence: 0.8553)
--------------------------------------------------------------------------------
Text: Rumors spreading on social media that Trump has been impeached twice this year — this is not true.
Prediction: Real (Confidence: 0.9929)
--------------------------------------------------------------------------------
```

```
XLNet Model Predictions:
Text: President Trump signed a new economic relief bill yesterday to support small businesses.
Sentiment: Fake (Confidence: 0.8822)
--------------------------------------------------------------------------------
Text: Fake news claims that Trump has secret plans to leave the country after the election.
Sentiment: Real (Confidence: 0.9990)
--------------------------------------------------------------------------------
Text: Trump held a rally attended by thousands of supporters despite the ongoing pandemic.
Sentiment: Real (Confidence: 0.9960)
--------------------------------------------------------------------------------
Text: A viral post falsely states that Trump endorsed a fake cure for COVID-19.
Sentiment: Real (Confidence: 0.9993)
--------------------------------------------------------------------------------
Text: The latest polls show Trump's approval rating increasing steadily over the past month.
Sentiment: Real (Confidence: 0.9669)
--------------------------------------------------------------------------------
Text: Rumors spreading on social media that Trump has been impeached twice this year — this is not true.
Sentiment: Real (Confidence: 0.9930)
--------------------------------------------------------------------------------
```

→Sau khi dự đoán ta đều thấy ở cả 2 mô hình đều đưa ra là "Real" khi có Confidence cao(0.96-0.97), còn khi dự đoán là "Fake" thì nằm trong (0.85-0.88)→Cả 2 mô hình dự đoán tương đối tốt.

IV. Kết luận.

- Dự án này nhằm giải quyết vấn đề nhức nhối của xã hội hiện đại: sự lan truyền của tin giả (fake news), bằng cách ứng dụng các mô hình xử lý ngôn ngữ tự nhiên tiên tiến. Với tập dữ liệu bao gồm các bài báo thật và giả từ nhiều nguồn, nhóm em đã

tiến hành đầy đủ quy trình học máy gồm: tiền xử lý, phân tích dữ liệu khám phá, xây dựng và huấn luyện mô hình, hiệu chỉnh siêu tham số và đánh giá hiệu suất.

- Hai mô hình chính được triển khai là **BERT** và **XLNet**. Cả hai đều thể hiện hiệu năng tốt trong việc phân loại tin tức, trong đó **BERT vượt trội hơn nhẹ**, đạt **độ chính xác khoảng 95.79%** trên tập kiểm tra, trong khi **XLNet đạt khoảng 95.59%**. BERT cũng thể hiện độ ổn định cao hơn trong quá trình huấn luyện và đánh giá.