

Phân tích dữ liệu thông minh

LEARNING AGENCY LAB

AUTOMATED ESSAY SCORING 2.0

Nhóm 15

THÀNH VIÊN

01

21120501
Nguyễn Ngọc
Gia Minh

02

21120504
Nguyễn
Phương Nam

03

21120516
Võ Bá
Hoàng Nhất

04

21120529
Nguyễn
Gia Phúc

05

21120576
Trần Đình
Nhật Trí

06

21120501
Nguyễn
Thủy Uyên

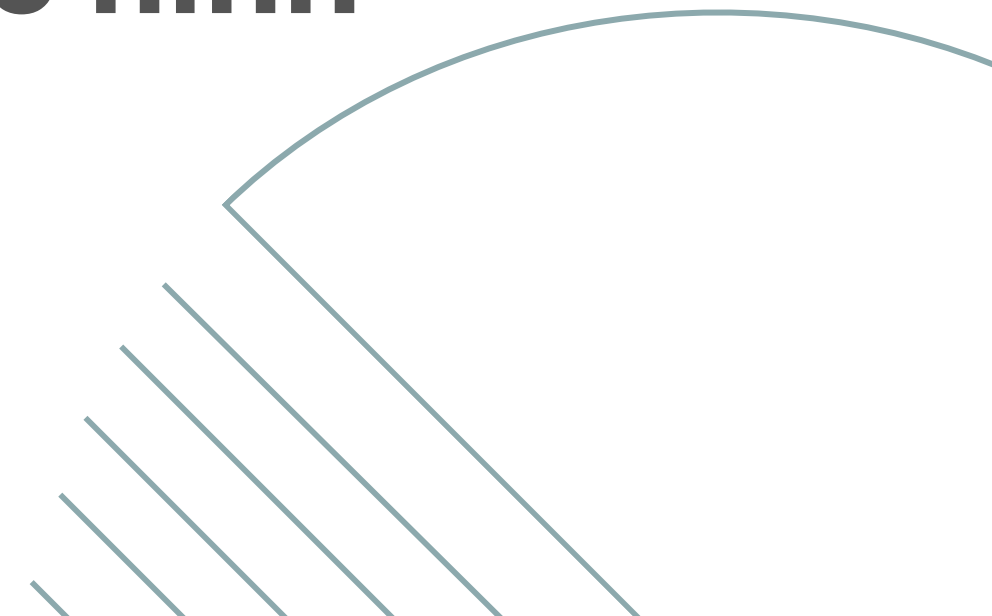
NỘI DUNG

01. Tổng quan dataset

02. EDA - Khám phá và phân tích

**03. Thử nghiệm trên mô hình
tự xây dựng**

**04. Thử nghiệm trên mô hình
ngôn ngữ lớn**





01. TỔNG QUAN DATASET

- Ngữ cảnh dataset
 - Nguồn dữ liệu
 - Mô tả dataset
- 



**IMPROVE UPON ESSAY
SCORING ALGORITHMS
TO IMPROVE STUDENT
LEARNING OUTCOMES**



NGŨ CẢNH

2012

=> Cần có một bộ dữ liệu rộng hơn

2024

- Cuộc thi Chấm điểm Bài luận Tự động đầu tiên đã chấm điểm các câu trả lời ngắn do học sinh viết.
- Bị hạn chế bởi các bộ dữ liệu nhỏ, không đa dạng trên toàn quốc hoặc không tập trung vào các định dạng bài luận phổ biến.

- Bộ dữ liệu viết truy cập mở lớn nhất, phù hợp với các tiêu chuẩn hiện hành để đánh giá phù hợp với học sinh.
- Tạo ra một thuật toán chấm điểm bài luận open-source để cải thiện cuộc thi Giải thưởng Đánh giá Sinh viên Tự động ASAP ban đầu được tổ chức vào năm 2012.

NGUỒN DỮ LIỆU

Learning Agency Lab - Automated Essay Scoring 2.0

Improve upon essay scoring algorithms to improve student learning outcomes

[Overview](#) [Data](#) [Code](#) [Models](#) [Discussion](#) [Leaderboard](#) [Rules](#) [Team](#) [Submissions](#)

Overview

The first automated essay scoring competition to tackle automated grading of student-written essays was twelve years ago. How far have we come from this initial competition? With an updated dataset and light years of new ideas we hope to see if we can get to the latest in automated grading to provide a real impact to overtaxed teachers who continue to have challenges with providing timely feedback, especially in underserved communities.

The goal of this competition is to train a model to score student essays. Your efforts are needed to reduce the high expense and time required to hand grade these essays. Reliable automated techniques could allow essays to be introduced in testing, a key indicator of student learning that is currently commonly avoided due to the challenges in grading.



Competition Host
The Learning Agency Lab



Prizes & Awards
\$50,000
Awards Points & Medals

Participation
7,762 Entrants
2,470 Participants
2,116 Teams
34,053 Submissions

- **Giấy phép:** Attribution-NonCommercial 4.0 International (CC BY-NC 4.0)
- **Tổ chức bởi:** The Learning Agency Lab

MÔ TẢ DỮ LIỆU

- Dữ liệu được chia ra 2 tập là tập dữ liệu train và dữ liệu test được tham khảo từ bộ dữ liệu cuộc thi bao gồm khoảng 24000 bài luận, tranh luận do học sinh viết. Mỗi bài luận được chấm theo thang điểm từ 1 đến 6.
- Mục tiêu sẽ là từ bộ dữ liệu train xây dựng mô hình phù hợp để dự đoán số điểm mà một bài luận nhận được từ văn bản của nó (tập test).

- train.csv: Các bài luận và điểm số được sử dụng làm dữ liệu train


Field	Description
essay_id	ID duy nhất của bài luận
full_text	Câu trả lời đầy đủ của bài luận
score	Điểm tổng thể của bài luận theo thang điểm 1-6

- test.csv: Các bài luận được sử dụng làm dữ liệu test.

Field	Description
essay_id	ID duy nhất của bài luận
full_text	Câu trả lời đầy đủ của bài luận

MÔ TẢ DỮ LIỆU

	train.csv	test.csv
kích thước tập dữ liệu	17307 dòng, 3 cột	3 dòng, 2 cột
missing value	không	không
dữ liệu trùng lặp	không	không
các cột numerical	score	không
các cột categorical	full_text, essay_id	full_text, essay_id



02. EDA

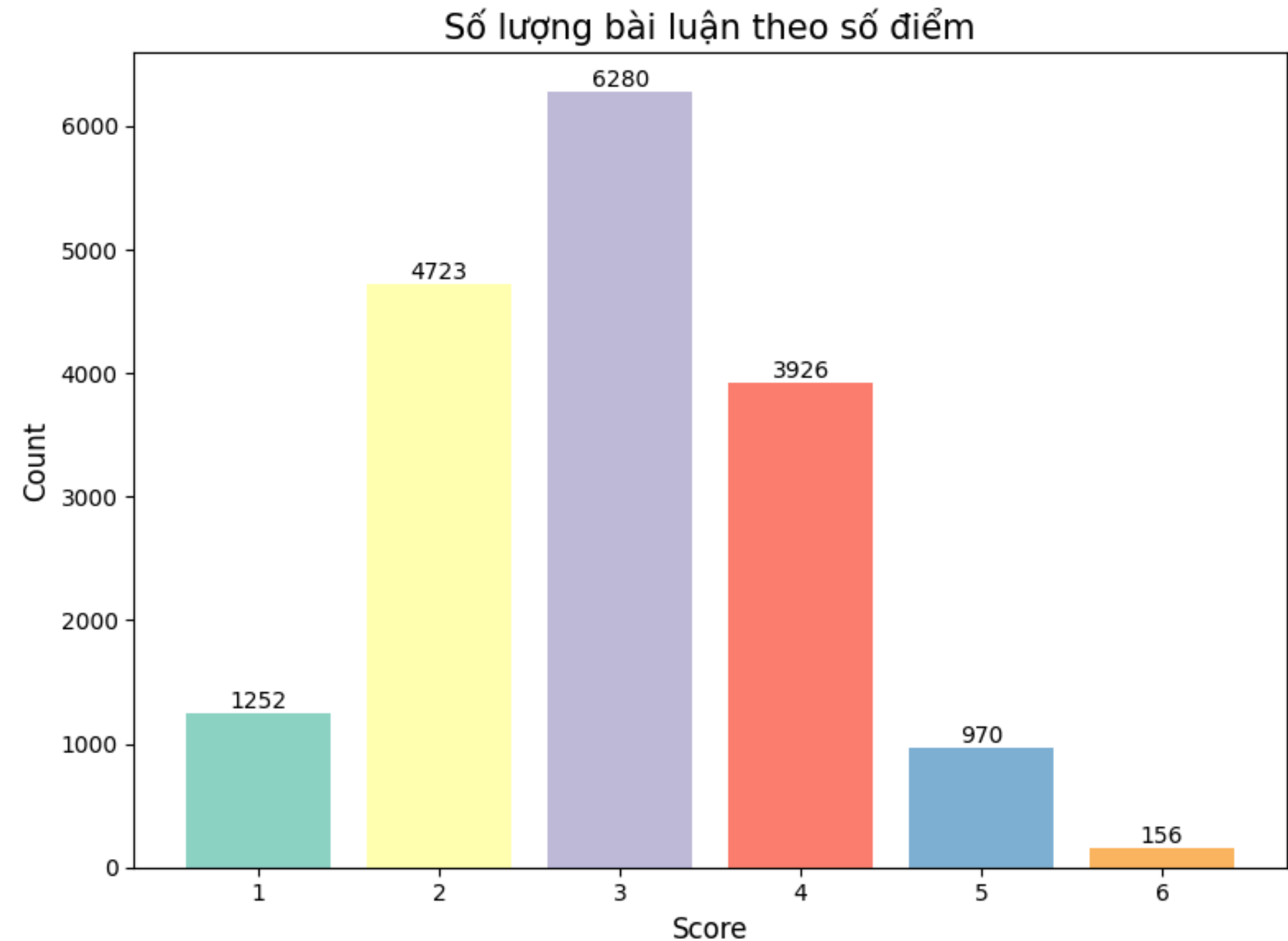
KHÁM PHÁ VÀ PHÂN TÍCH

- Một số câu hỏi phân tích và thống kê dataset
- 

PHÂN BỐ CỦA CÁC BÀI LUẬN THEO ĐIỂM SỐ

NHẬN XÉT

- Phần lớn bài luận nhận được điểm số 3 => mức điểm phổ biến nhất. Điểm số 2 và 4 cũng có số lượng bài luận khá cao.
- Điểm số 1 và 5 có số lượng bài luận ít hơn đáng kể. Điểm số 6 là mức điểm hiếm gặp nhất => rất ít bài đạt được điểm số này.
- Có xu hướng giảm dần số lượng bài luận từ điểm số 3 xuống điểm số 6.



PHÂN BỐ ĐIỂM SỐ CÁC BÀI LUẬN THEO ĐỘ DÀI

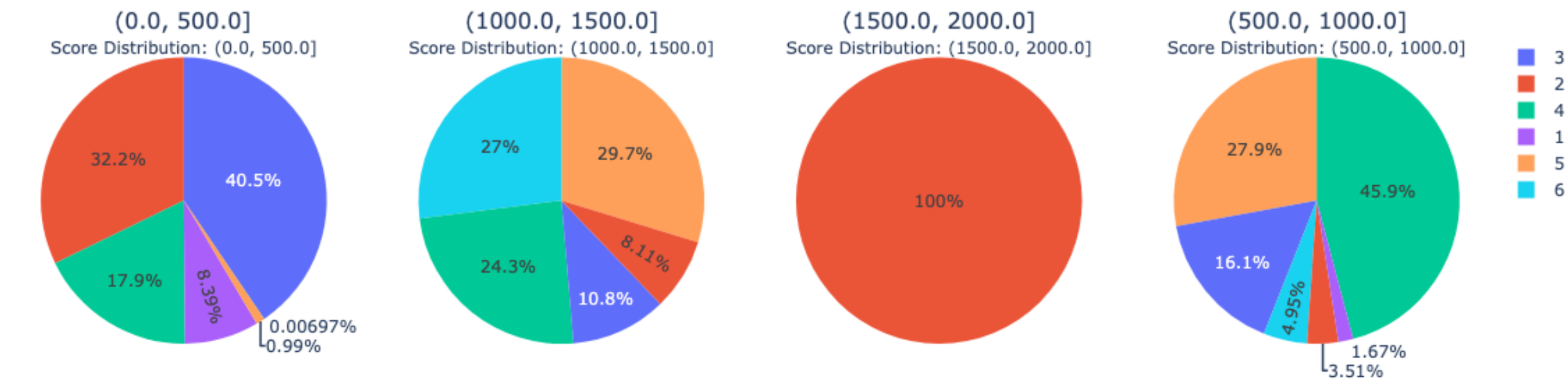
Cột counts đếm số lượng từ trong khoảng.

Chia thành các khoảng

	length_text_range object	score int64	counts int64
	(0.0, 500.0] 33.3%	1 - 6	1 - 5804
	(500.0, 1000.0] 33.3%		
	2 others 33.3%		
0	(0.0, 500.0]	1	1203
1	(0.0, 500.0]	2	4616
2	(0.0, 500.0]	3	5804
3	(0.0, 500.0]	4	2572
4	(0.0, 500.0]	5	142

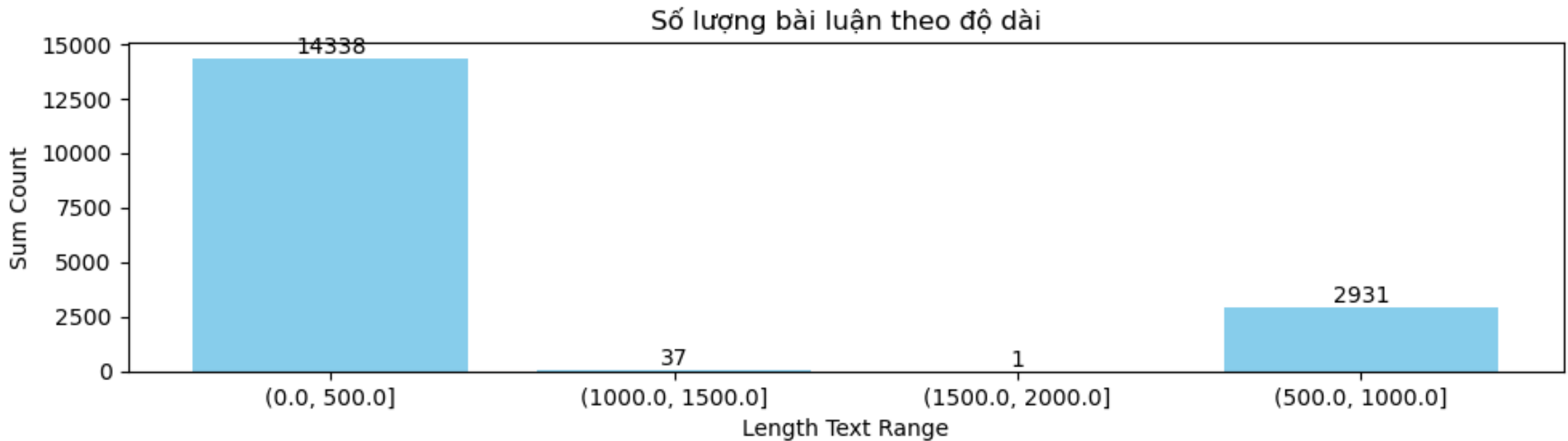
PHÂN BỐ ĐIỂM SỐ CÁC BÀI LUẬN THEO ĐỘ DÀI

Phân bố Score theo Độ dài văn bản



NHẬN XÉT

- Các bài luận (500-1000] rất đa dạng điểm, số lượng bài điểm 4, 5 chiếm đa số.
- Có thể thấy phần lớn bài luận có số lượng từ vựng nhiều sẽ có điểm số cao hơn.



PHÂN BỐ ĐIỂM SỐ VỚI SỐ TỪ VỰNG SAI CHÍNH TẢ TRONG MỘT BÀI LUẬN

```
def removeStopwords(sentence):  
    word_tokens = word_tokenize(sentence)  
    filtered_sentence = [w for w in word_tokens if not w.lower() in stop_words]  
    return ' '.join(filtered_sentence)
```

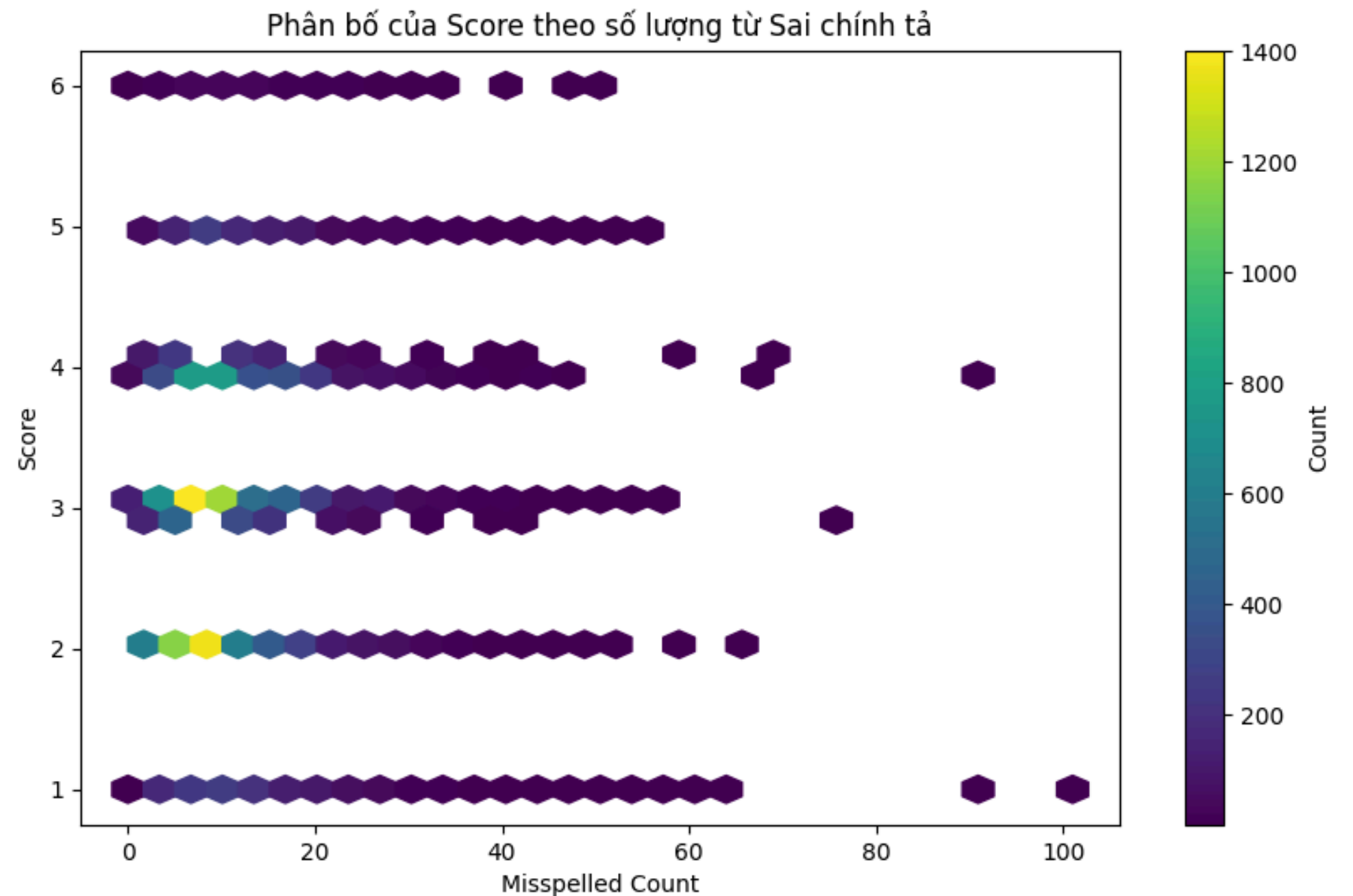
Many people have car where they live. The thing they don't
Many people car live thing know use car alot thing happen

```
def removePunctuations(sentence):  
    cleaned_text = re.sub('[^a-zA-Z]', ' ', sentence)  
    return (cleaned_text)
```

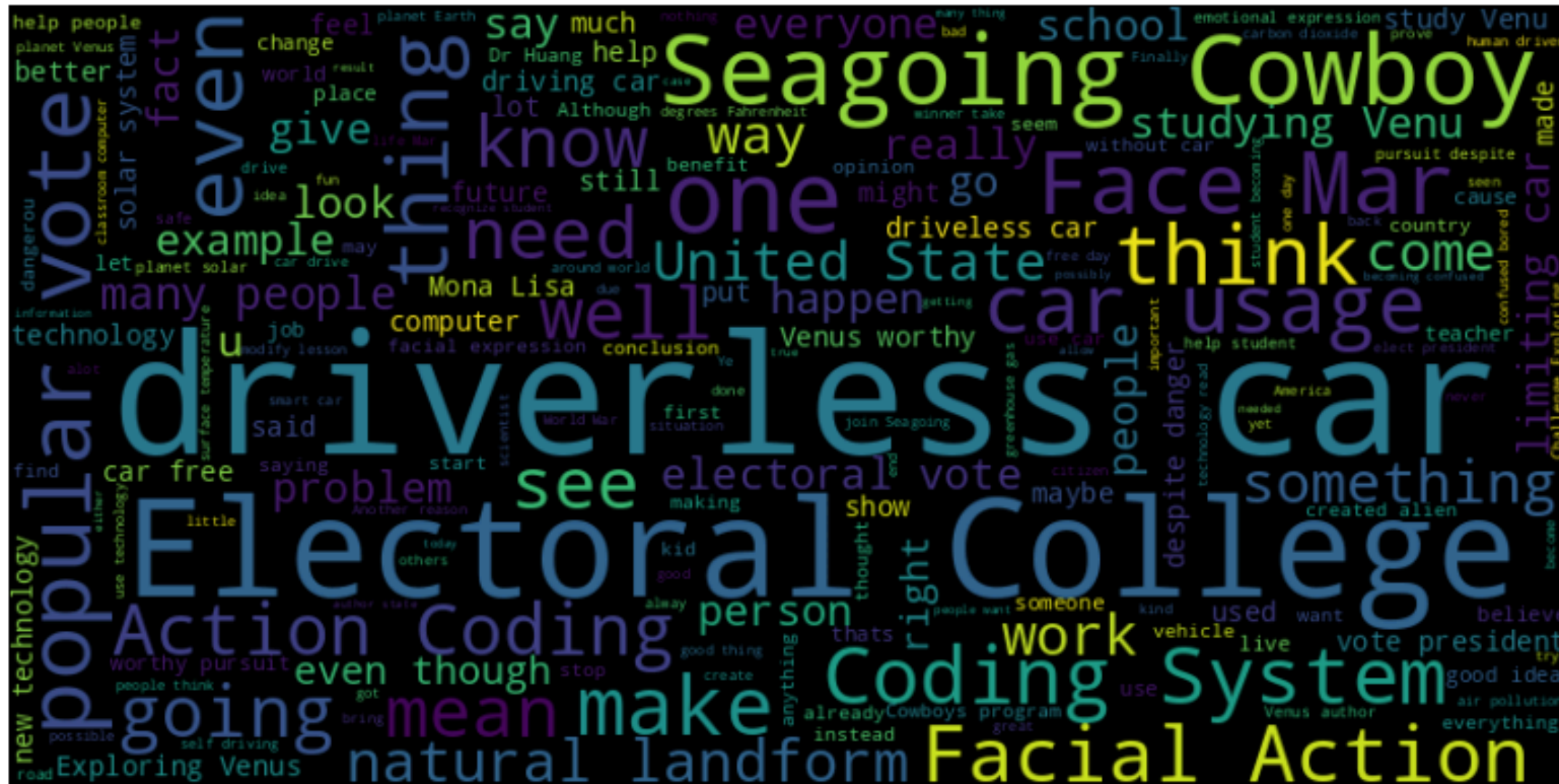
PHÂN BỐ ĐIỂM SỐ VỚI SỐ TỪ VƯNG SAI CHÍNH TẢ TRONG MỘT BÀI LUẬN

NHẬN XÉT

- Số lượng bài luận mắc lỗi từ 0 - 20 từ là nhiều nhất và tập trung ở mức điểm 2 - 4.
- Hầu như các bài luận được điểm càng cao thì càng mắc ít lỗi chính tả.
- Các điểm ngoại lai (các bài mắc rất nhiều lỗi - khoảng trên 60 lỗi) có điểm từ 1 - 4.
- Bài mắc nhiều lỗi nhất (trên 100 lỗi) thường có điểm là 1.



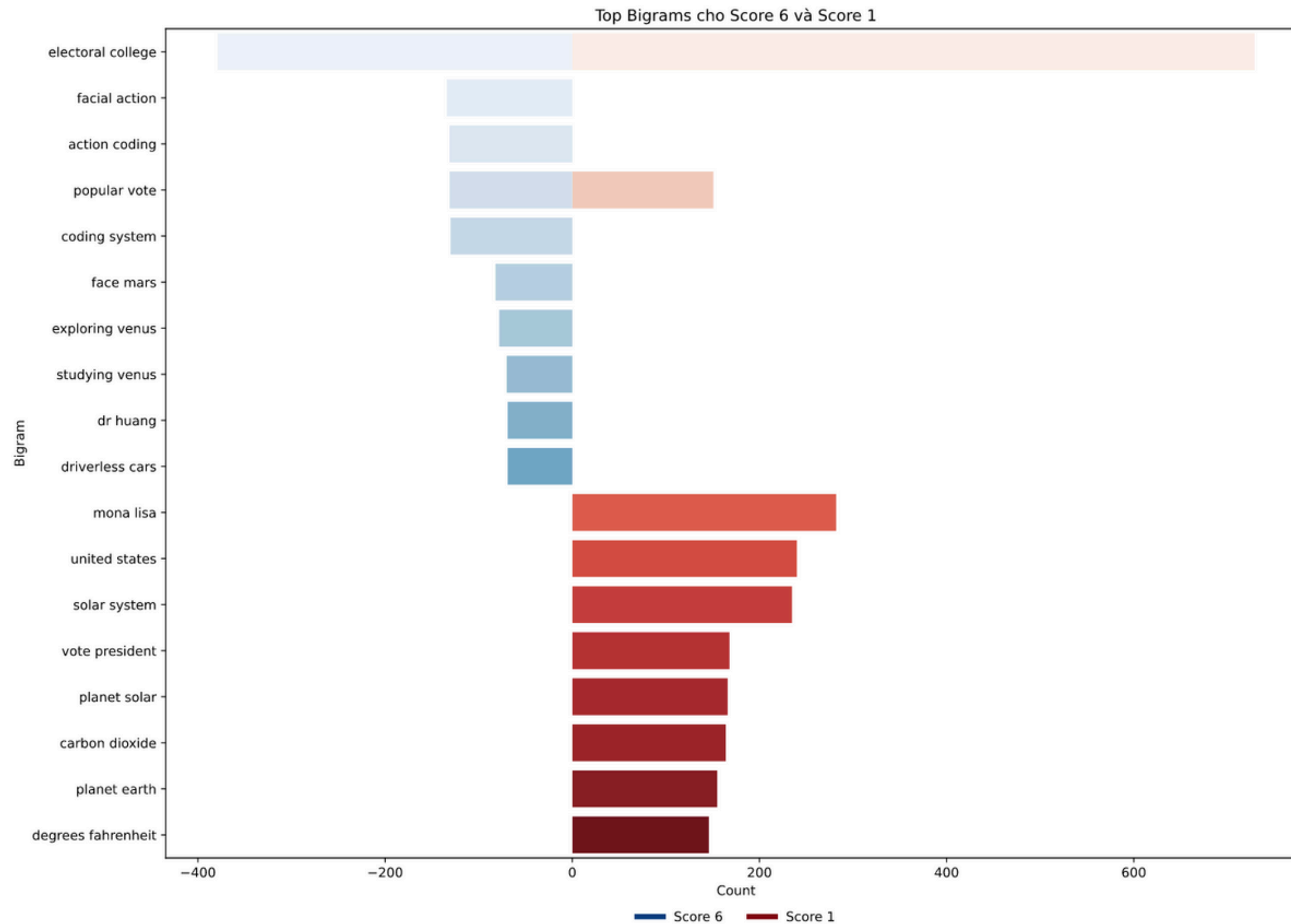
BIỂU ĐỒ WORDCLOUD



NHẬN XÉT

- Chủ đề của các bài luận xoay quanh **driverless car, Electoral College, Seagoing Cowboy, Coding System, Face Mar,...**

BIỂU ĐỒ WORDCLOUD



NHẬN XÉT

- Những bigrams được sử dụng ở cả hai mức điểm là electoral college, popular vote
- Các bài luận có score 1 không có tính thống nhất với chủ đề

PHÂN TÍCH CẢM XÚC (SENTIMENT ANALYSIS)

	essay_id	object	sentiment	float64
	000d118	0%	-0.3651892551892...	
	000fe60	0%		
	17305 others	100%		
0	000d118		0.1790203882	
1	000fe60		0.1017857143	
2	001ab80		0.1684351923	
3	001bdc0		0.0942913001	
4	002ba53		0.170952381	

Tính điểm sentiment
cho các bài luận

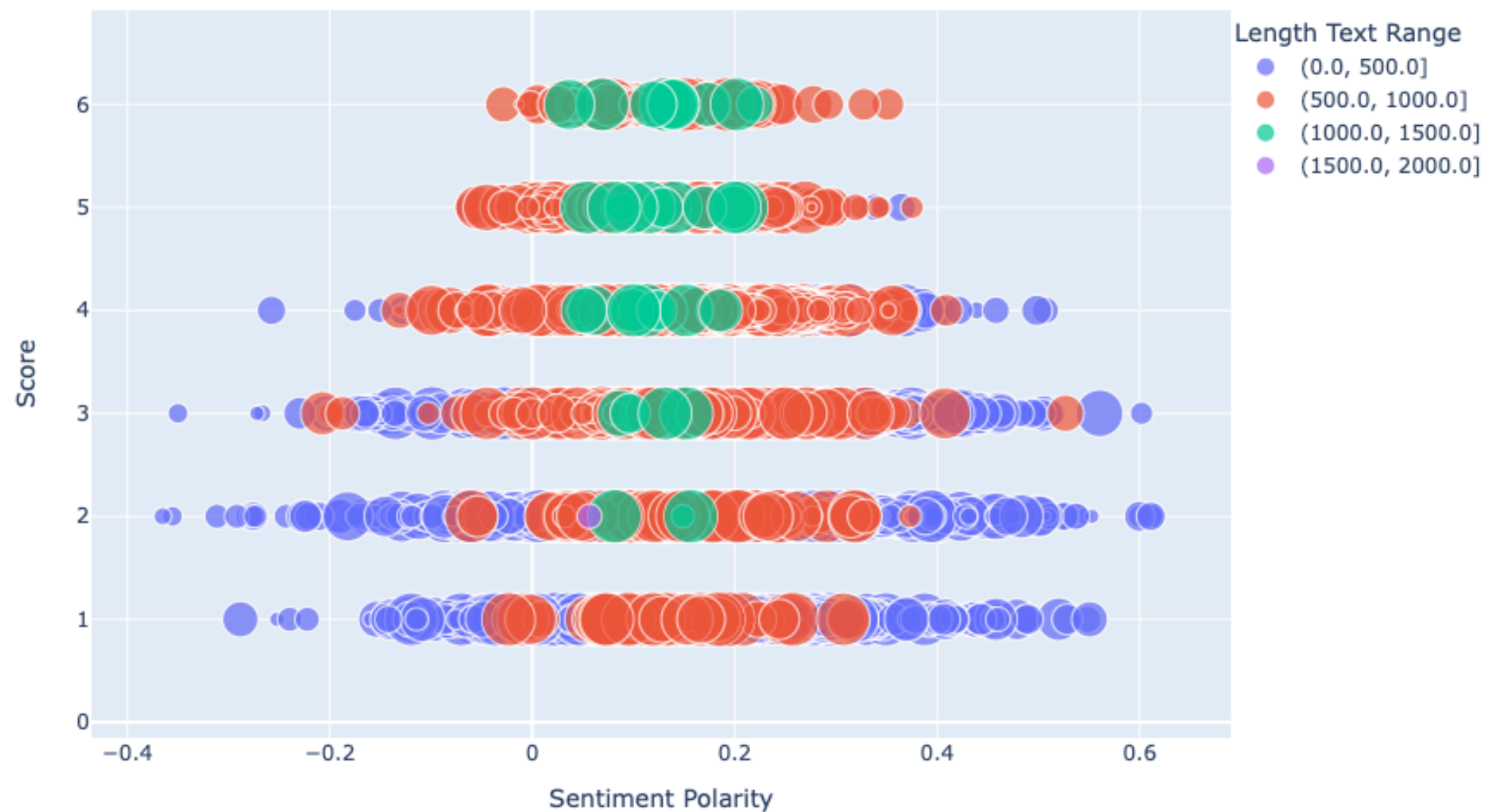
```
def get_sentiment_score(text):  
    blob = TextBlob(text)  
    return blob.sentiment.polarity
```

PHÂN TÍCH CẢM XÚC (SENTIMENT ANALYSIS)

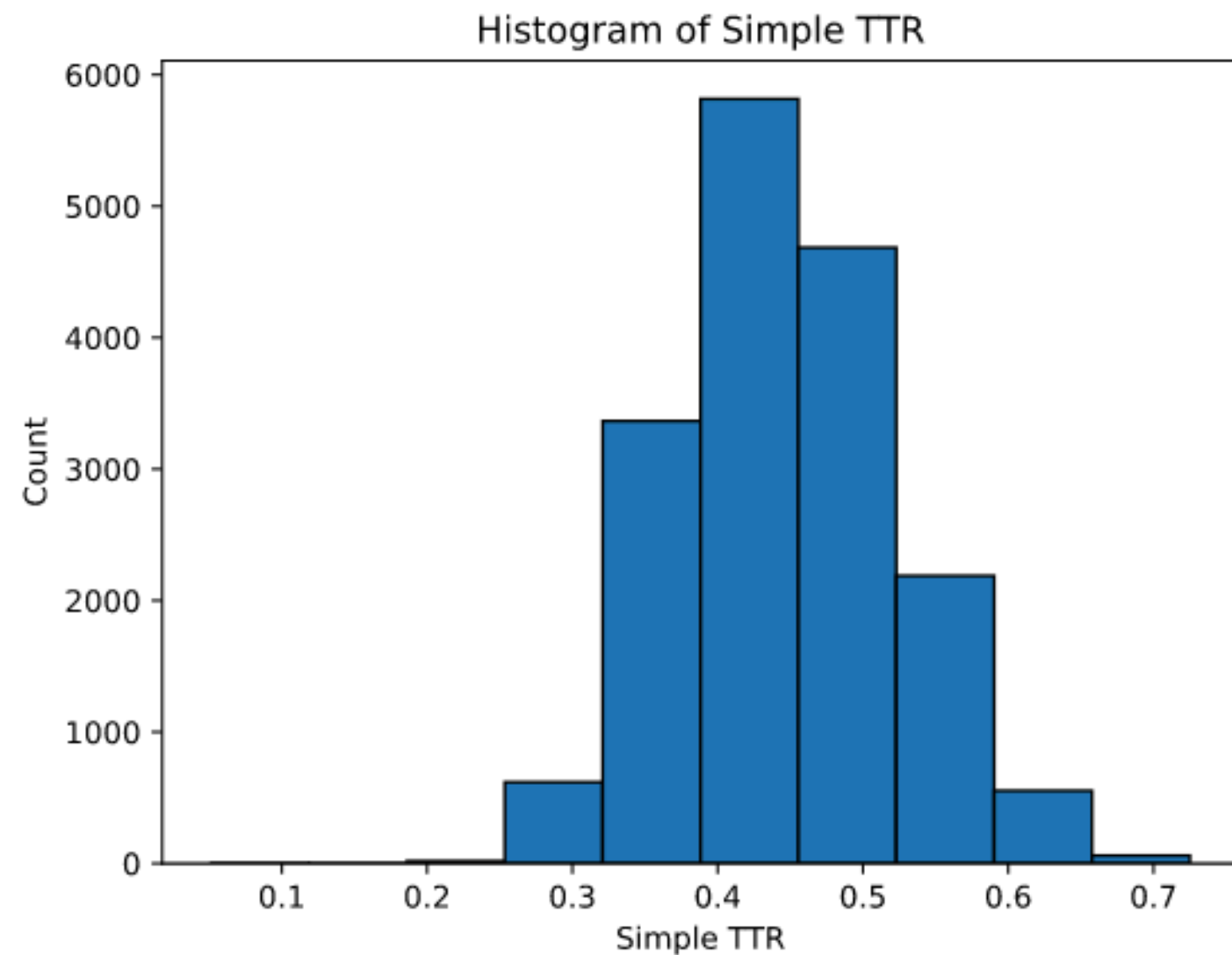
NHẬN XÉT

- sentiment $< 0 \Rightarrow$ các bài essay điểm càng cao thì có sentiment càng cao.
- Ngược lại sentiment $> 0 \Rightarrow$ Hầu như các bài essay điểm càng thấp thì có sentiment càng cao. \Rightarrow Càng bài essay điểm càng cao có miền sentiment càng thấp.

Scatter Plot of Score vs Sentiment Polarity



PHÂN TÍCH ĐỘ ĐA DẠNG TỪ VỰNG



```
def get_Simple_TTR(text):  
    flt = ld.flemmatize(text)  
    return ld.ttr(flt)
```

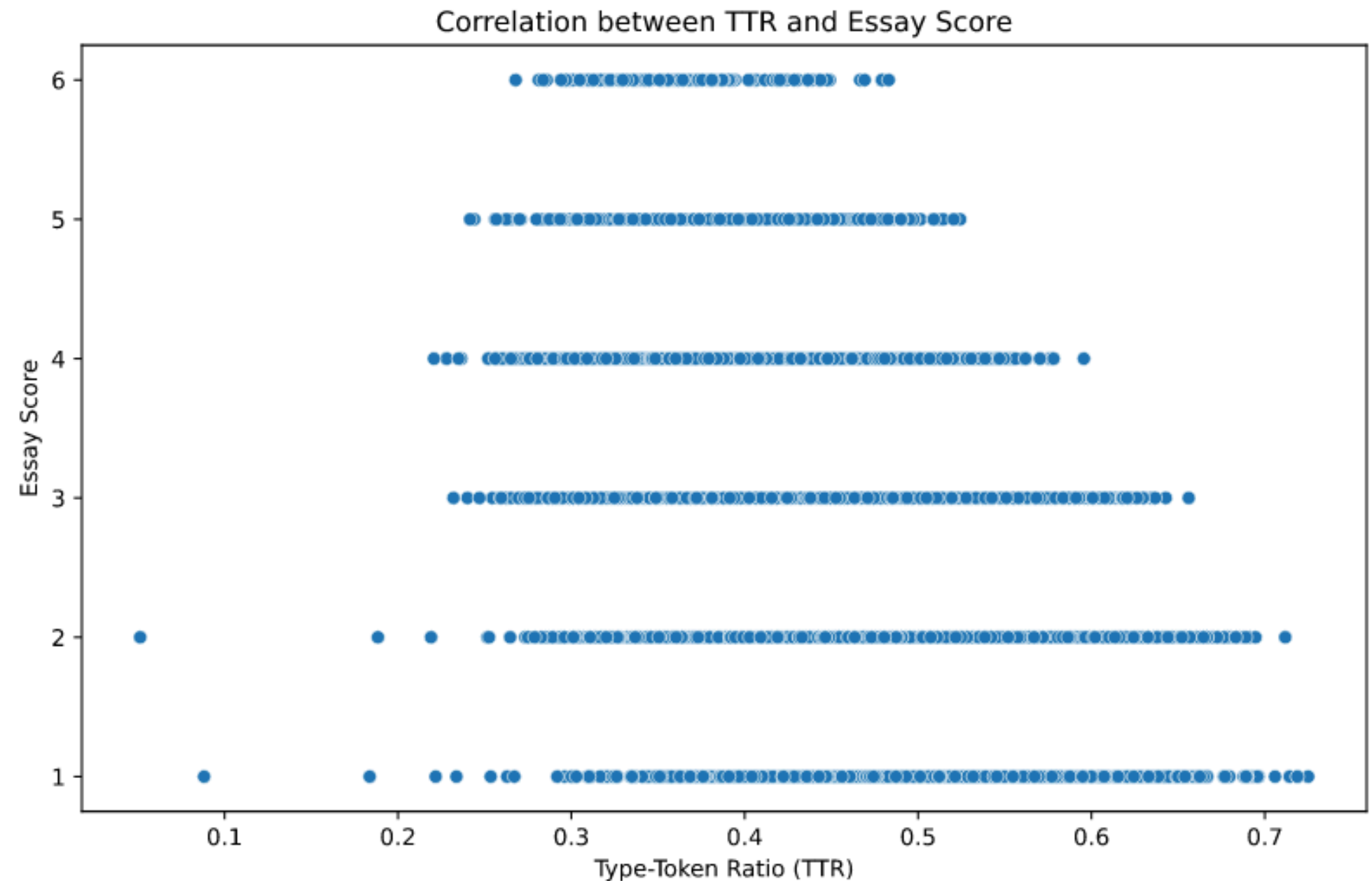
$$\text{TTR} = \frac{\text{NUMBER OF UNIQUE WORDS (TYPES)}}{\text{TOTAL NUMBER OF WORDS (TOKENS)}}$$

=> TTR phân bố nhiều ở khoảng (0.4-0.5)

PHÂN TÍCH ĐỘ ĐA DẠNG TỪ VỰNG

NHẬN XÉT

- Không có mối quan hệ tỉ lệ thuận giữa điểm số và độ đa dạng từ vựng
- Tuy nhiên để đạt điểm cao (từ điểm 5 trở lên) thì hầu như các bài essay phải có chỉ số Simple TTR từ điểm 0.25 trở lên.



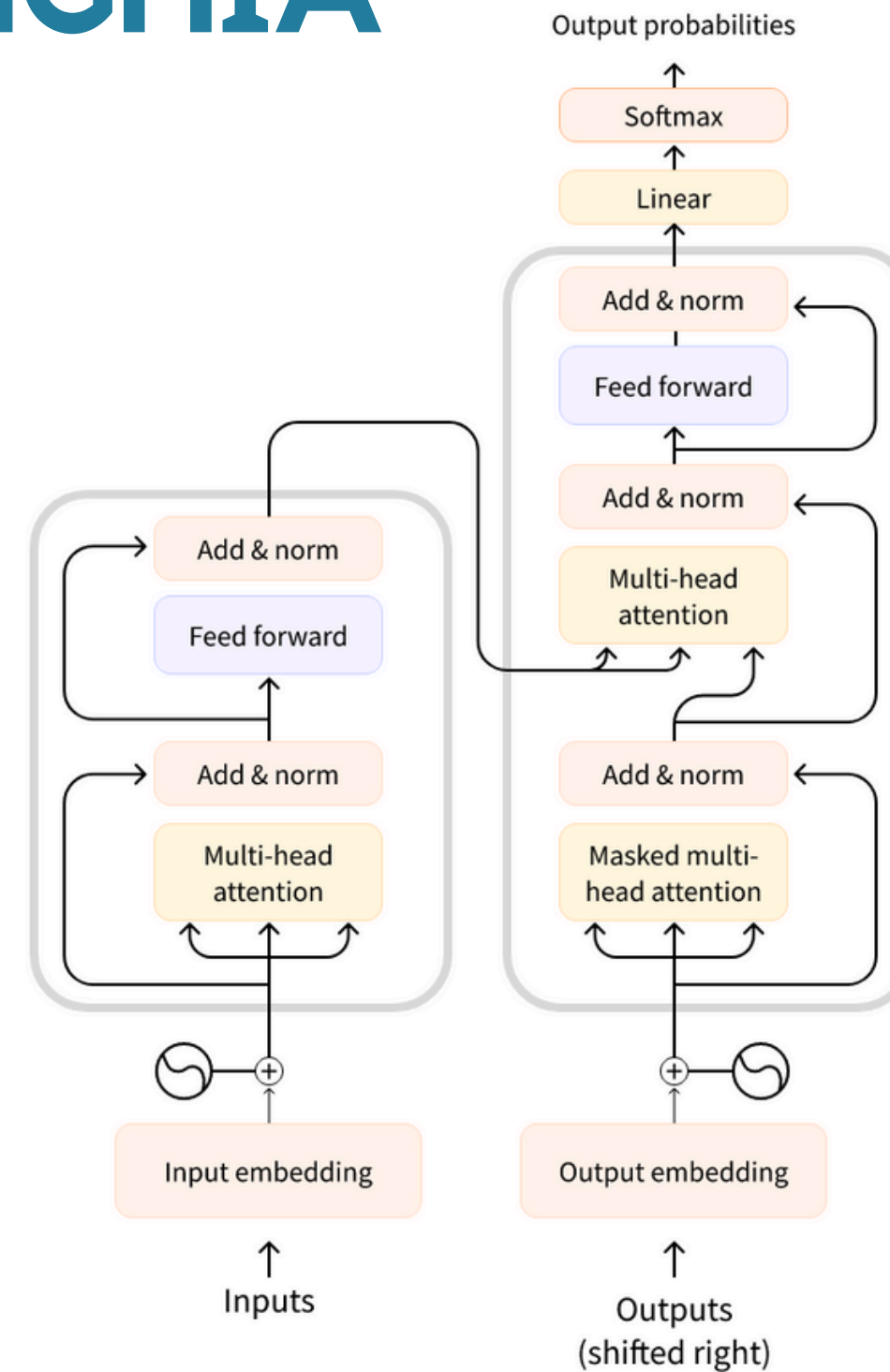


03. THỬ NGHIỆM TRÊN MÔ HÌNH TỰ XÂY DỰNG

- Transformer
 - Mô hình DeBERTaV3
- 

TRANSFOMER - ĐỊNH NGHĨA

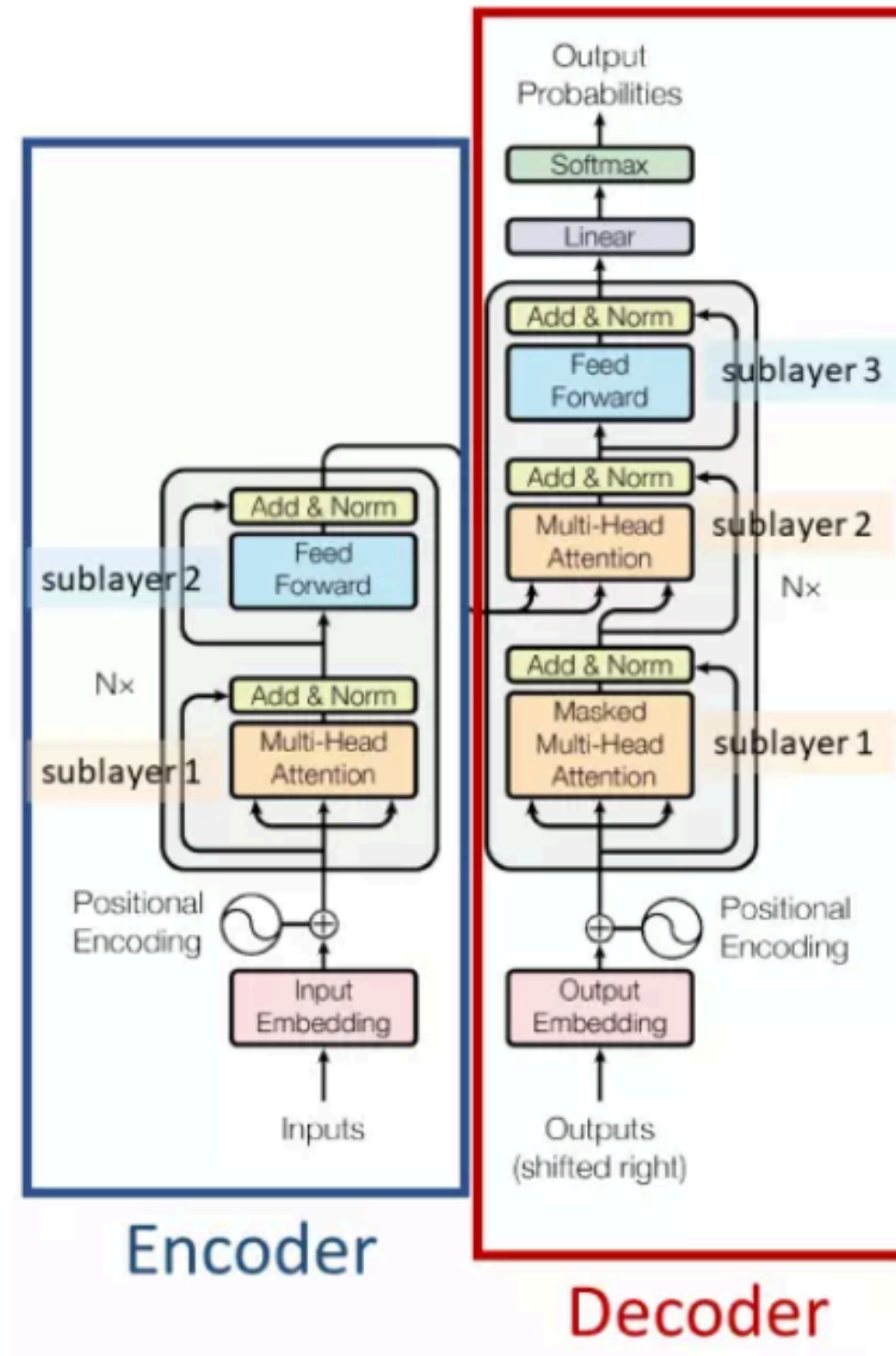
- **Transformer** là một kiến trúc **mạng nơ-ron sâu**, nó đã trở thành một phương pháp chủ chốt trong lĩnh vực xử lý ngôn ngữ tự nhiên (NLP) và dịch máy tự động, cũng như nhiều ứng dụng khác trong học sâu.
- Transformer bao gồm hai thành phần chính: **Bộ mã hóa (Encoder)** và **Bộ giải mã (Decoder)**



TRANSFOMER - QUY TRÌNH HOẠT ĐỘNG

Mã hóa (Encoding):

- Chuỗi đầu vào được đưa qua nhiều lớp mã hóa
- Mỗi lớp mã hóa gồm self-attention và feed-forward network
- Kết quả cuối cùng là chuỗi biểu diễn của đầu vào đã được mã hóa

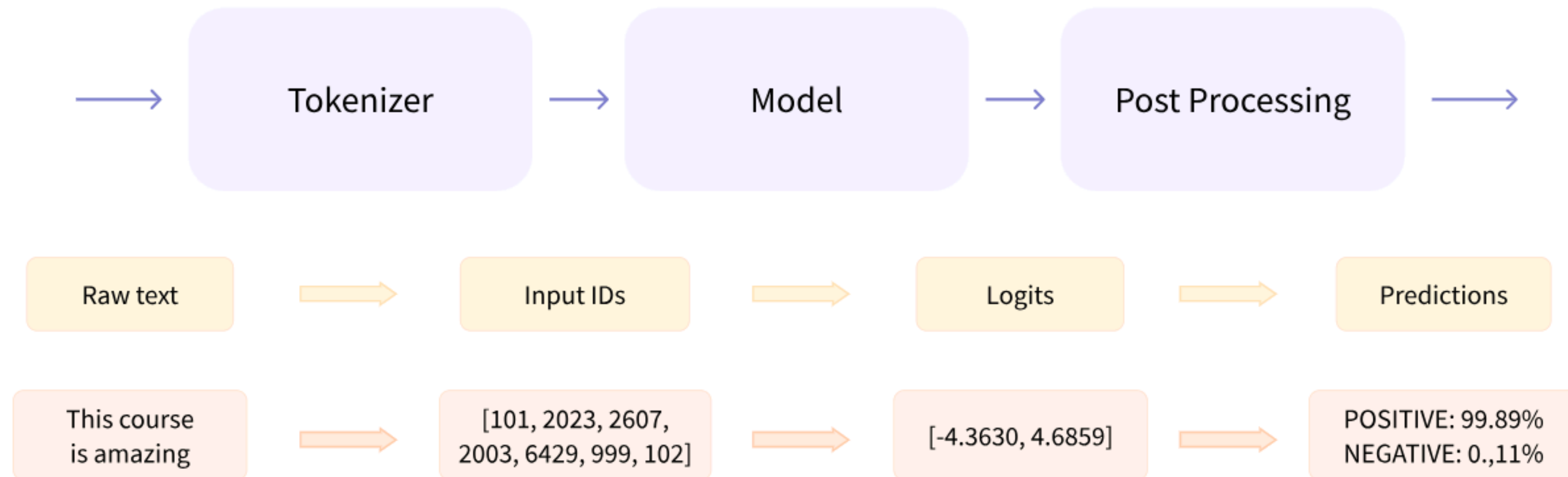


Giải mã (Decoding):

- Chuỗi đầu ra từng phần được đưa qua nhiều lớp giải mã
- Mỗi lớp giải mã gồm masked self-attention, encoder-decoder attention và feed-forward network.
- Kết quả cuối cùng là chuỗi đầu ra đã được giải mã hoàn chỉnh.

TRANSFOMER - CẤU TRÚC PIPELINE

Công cụ Pipeline: pipeline() là công cụ cơ bản nhất trong thư viện Transformers, kết nối mô hình với các bước tiền xử lý và hậu xử lý cần thiết, cho phép nhập trực tiếp văn bản và nhận câu trả lời có ý nghĩa.



DEBERTA-V3 MODEL - GIỚI THIỆU

- **DeBERTa** là một mô hình encoding cải thiện các mô hình **BERT** và **RoBERTa** bằng cách sử dụng cơ chế tập trung disentangled và bộ giải mã mặt nạ cải tiến.
- **DeBERTaV3** được pre-trained để phục vụ tác vụ Fill-Mask nhưng thư viện Transformers đã cung cấp đầy đủ các công cụ để ta có thể fine-tuning mô hình cho tác vụ classification.
- Ta sẽ sử dụng câu lệnh sau để load model:
`AutoModelForSequenceClassification.from_pretrained(checkpoint, num_labels=1)`



DEBERTA-V3 MODEL - GIỚI THIỆU

Sau khi chạy lệnh này ta sẽ nhận được thông báo

Some weights of DebertaV2ForSequenceClassification were not initialized from the model checkpoint at /kaggle/input/init-aes2/microsoft__deberta-v3-small and are newly initialized: ['classifier.bias', 'classifier.weight', 'pooler.dense.bias', 'pooler.dense.weight']
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.

Có nghĩa là ta sẽ không thể tận dụng được các trọng số đã được pre-trained của mô hình nên ta cần phải fine-tuning một mô hình hoàn toàn mới trên tập dữ liệu của mình.



DEBERTA-V3 MODEL - GIỚI THIỆU

- Mô hình DeBERTaV3 sẽ có 4 phiên bản với độ lớn giảm dần như sau:
 - **DeBERTaV3-large:** mô hình quá lớn gây ra tràn VRAM nên không thể sử dụng được.
 - **DeBERTaV3-base:** Mô hình cho kết quả tốt nhất, được nhóm sử dụng để làm bài nộp. Mỗi epoch của mô hình này mất khoảng 30 phút chạy
 - **DeBERTaV3-small:** Mô hình cho tốc độ chạy nhanh hơn (20 phút/epoch) và kết quả cũng khá tốt nên được nhóm sử dụng chủ yếu để thử nghiệm.
 - **DeBERTaV3-xsmall:** Mô hình nhỏ nhất, được nhóm sử dụng để khởi đầu.



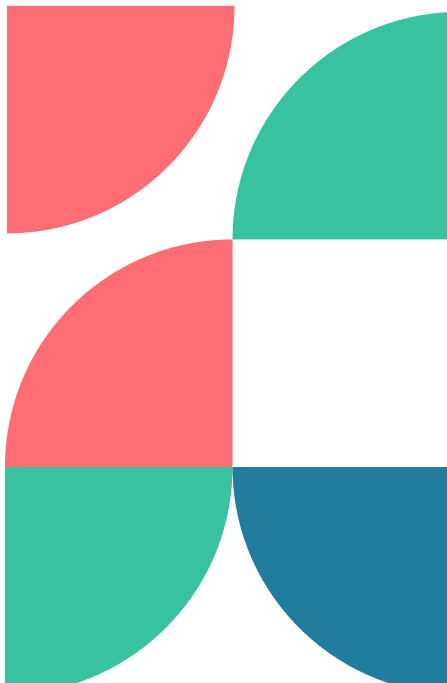


DEBERTA-V3 MODEL - PREPROCESSING

Trong các bài tiểu luận xuất hiện một vài các lỗi có thể gây ảnh hưởng quá trình tokenization nên cần phải xử lý trước khi mã hóa văn bản thành dãy token. Các lỗi đó bao gồm:

- Xuất hiện chuỗi \xa0 (non-breaking space).
- Xuất hiện chuỗi các chuỗi html do web scraping.
- Xuất hiện các dấu chấm liên tiếp.
- Xuất hiện các dấu phẩy liên tiếp.
- Xuất hiện chuỗi " thay vì "

Ta sẽ sử dụng hàm preprocessor để xử lý các lỗi trên



DEBERTA-V3 MODEL - TOKENIZER

- **Tokenizer** của mô hình sẽ chuyển văn bản thành một dãy các token. Dãy token này sẽ có 3 thành phần để biểu diễn:
 - **input_ids**: Các token đã được mã hóa thành dãy các số nguyên ứng với từ điển của mô hình.
 - **attention_mask**: Một dãy số 0 và 1, 1 ở vị trí token, 0 ở vị trí padding (các token thêm vào để đảm bảo độ dài của dãy token bằng nhau).
 - **token_type_ids**: Dùng để phân biệt các token trong nhiều input khác nhau. Ở bài toán này, chỉ có một input nên giá trị của token_type_ids tất cả là 0.

"Many people have car where they live."

Sử dụng hàm `tokenizer.tokenize(text)`:

=> ['__Many', '__people', '__have', '__car', '__where', '__they', '__live', '.']

Sử dụng hàm `tokenizer(text, truncation = True, max_length = 20)`:

=> {'input_ids': [1, 1304, 355, 286, 640, 399, 306, 685, 260, 2],
 'token_type_ids': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
 'attention_mask': [1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}

DEBERTA-V3 MODEL - XỬ LÝ DỮ LIỆU

```
def data_preprocessing_test(path, tokenizer, max_length):  
    data = pd.read_csv(path)  
    data['full_text'] = data['full_text'].apply(preprocessor)  
    dataset = Dataset.from_pandas(data)  
    dataset = dataset.map(lambda x: tokenizer(x['full_text'],  
                                             truncation = True,  
                                             max_length = max_length),  
                          batched = True)  
  
    # Remove unnecessary columns  
    columns_to_remove = ['essay_id', 'full_text']  
    dataset = dataset.remove_columns(columns_to_remove)  
    return dataset, data
```



DEBERTA-V3 MODEL - THAM SỐ

Mô hình DeBERTaV3:

- **num_labels:** Số lượng label cần dự đoán.
- **hidden_dropout_prob:** dropout ở lớp ẩn.
- **attention_probs_dropout_prob:** dropout ở lớp attention.



DEBERTA-V3 MODEL - THAM SỐ

TrainingArguments:

- **learning_rate:** Hệ số học.
- **warmup_ratio:** Tỷ lệ warmup.
- **num_train_epochs:** Số epoch huấn luyện.
- **per_device_train_batch_size:** Kích thước batch huấn luyện trên mỗi thiết bị.
- **per_device_eval_batch_size:** Kích thước batch đánh giá trên mỗi thiết bị.
- **fp16:** Sử dụng mixed precision training.
- **lr_scheduler_type:** Loại scheduler sử dụng cho hệ số học.
- **weight_decay:** Hệ số weight decay.
- **load_best_model_at_end:** Load mô hình tốt nhất ở cuối quá trình huấn luyện.
- **optim:** Optimiser sử dụng.





DEBERTA-V3 MODEL - THAM SỐ

Trainer:

- **model:** Mô hình cần huấn luyện.
- **args:** Các tham số huấn luyện.
- **train_dataset:** Dữ liệu huấn luyện.
- **eval_dataset:** Dữ liệu đánh giá.
- **data_collator:** Hàm tạo batch từ dữ liệu.
- **tokenizer:** Tokenizer sử dụng để mã hóa dữ liệu.
- **compute_metrics:** Hàm tính toán các metric đánh giá mô hình.



DEBERTA-V3 MODEL - ĐÁNH GIÁ VỚI QUADRATIC WEIGHTED KAPPA

- **Quadratic weighted kappa** là một metric được sử dụng để đánh giá mức độ đồng thuận giữa hai chuỗi số. Metric này thường được sử dụng trong các bài toán dự đoán điểm số.
- Có giá trị nằm trong khoảng $[0,1]$, với 1 là giá trị tốt nhất. Kappa càng gần 0 có nghĩa là random agreement, còn kappa càng gần 1 có nghĩa là complete agreement.

Sử dụng hàm `cohen_kappa_score` từ thư viện `sklearn` để tính quadratic weighted kappa

```
from sklearn.metrics import cohen_kappa_score
y_true = [1, 2, 3, 4, 3]
y_pred = [2, 2, 4, 4, 5]
kappa = cohen_kappa_score(y_true, y_pred, weights='quadratic')
print(kappa)
```



DEBERTA-V3 MODEL - ĐƯA MÔ HÌNH VỀ DỰ ĐOÁN REGRESSION

Có 2 hướng tổ chức kết quả để mô hình dự đoán 6 điểm số

- Hướng classification: Các điểm số sẽ được mã hóa về các vector sau để đảm bảo tính thứ tự:
 - 1: [1, 0, 0, 0, 0, 0]
 - 2: [1, 1, 0, 0, 0, 0]
 - 3: [1, 1, 1, 0, 0, 0]
 - 4: [1, 1, 1, 1, 0, 0]
 - 5: [1, 1, 1, 1, 1, 0]
 - 6: [1, 1, 1, 1, 1, 1]
- Hướng regression: bởi vì các điểm số này có tính thứ tự tự nhiên ($1 < 2 < 3, \dots$) nên ta có thể dùng bài toán regression để dự đoán.

=> Sau khi thử nghiệm và đánh giá thì kết quả ở mô hình theo hướng regression cho điểm số tốt hơn hướng classification

DEBERTA-V3 MODEL - ĐƯA MÔ HÌNH VỀ DỰ ĐOÁN REGRESSION

Để cho mô hình có thể dự đoán regression ta cần một vài điều chỉnh đặt biệt:

- Đưa số lượng label về 1
- Tắt dropout đi

```
model = AutoModelForSequenceClassification.\  
    from_pretrained(cfg.checkpoint,  
    num_labels=1,  
    hidden_dropout_prob = 0,  
    attention_probs_dropout_prob = 0,)
```

DEBERTA-V3 MODEL - TRAIN MÔ HÌNH SỬ DỤNG STRATIFIEDKFOLD

STRATIFIEDKFOLD LÀ MỘT PHƯƠNG PHÁP CHIA DỮ LIỆU THÀNH CÁC FOLDS SAO CHO TỈ LỆ CỦA CÁC LABELS TRONG MỖI FOLD KHÔNG THAY ĐỔI SO VỚI TỈ LỆ CỦA TOÀN BỘ DỮ LIỆU.

```
X, y = np.ones((50, 1)), np.hstack(([0] * 45, [1] * 5))
skf = StratifiedKFold(n_splits=5)
for train, test in skf.split(X, y):
    print('train - {} | test - {}'.format(
        np.bincount(y[train]), np.bincount(y[test])))
```

Output

```
# train - [30 3] | test - [15 2]
# train - [30 3] | test - [15 2]
# train - [30 4] | test - [15 1]
```



DEBERTA-V3 MODEL - TRAIN MÔ HÌNH SỬ DỤNG STRATIFIEDKFOLD

Hàm Train mô hình gồm các bước:

- Load tokenizer từ checkpoint.
- Khởi tạo data collator.
- Gọi hàm data_preprocessing lấy dữ liệu đã được tiền xử lý.
- Chia dữ liệu thành 5 fold sử dụng StratifiedKFold.
- Với mỗi split (gồm 4 folds train và 1 fold validation):
 - Chọn dữ liệu train và dữ liệu validation từ dataset với index lấy từ split.
 - Load mô hình DeBERTaV3 từ checkpoint và thêm head ForSequenceClassification
 - Khởi tạo TrainingArguments.
 - Khởi tạo Trainer.
 - Huấn luyện mô hình và đánh giá trên dữ liệu validation cuối mỗi epoch.
 - Lưu mô hình.





DEBERTA-V3 MODEL - TRAIN MÔ HÌNH SỬ DỤNG STRATIFIEDKFOLD

Với hàm test (dùng để tạo file submission.csv):

- Tạo một list chứa kết quả dự đoán từ 5 mô hình.
- Với mỗi mô hình:
- Load mô hình và tokenizer đã được lưu từ quá trình huấn luyện.
- Khởi tạo data collator.
- Gọi hàm data_preprocessing_test để lấy dữ liệu test đã được tiền xử lý.
- Khởi tạo Trainer với model, tokenizer và data_collator.
- Dự đoán trên dữ liệu test và lưu kết quả vào list.
- Tính trung bình của 5 kết quả dự đoán.
- Lưu kết quả dự đoán vào file submission.csv.



DEBERTA-V3 MODEL - KẾT QUẢ MÔ HÌNH

- Cross validation: 0.83
- Điểm leader board: 0.8
- Thời gian chạy: 8h

Training fold 1 ...

[5193/5193 1:38:24, Epoch 3/3]

Epoch	Training Loss	Validation Loss	Quad Kappa
1	0.408600	0.372352	0.784291
2	0.251500	0.272572	0.833029
3	0.195900	0.265628	0.840505

Training fold 3 ...

[5193/5193 1:38:14, Epoch 3/3]

Epoch	Training Loss	Validation Loss	Quad Kappa
1	0.332100	0.359282	0.775900
2	0.297400	0.294791	0.819457
3	0.211200	0.283701	0.826127

Training fold 0 ...

[5193/5193 1:37:51, Epoch 3/3]

Epoch	Training Loss	Validation Loss	Quad Kappa
1	0.364200	0.319130	0.775340
2	0.274600	0.292317	0.805738
3	0.221900	0.273240	0.833589

Training fold 2 ...

[5193/5193 1:38:00, Epoch 3/3]

Epoch	Training Loss	Validation Loss	Quad Kappa
1	0.329100	0.341107	0.784351
2	0.296400	0.313927	0.798267
3	0.237600	0.284492	0.823459

Training fold 4 ...

[5193/5193 1:38:07, Epoch 3/3]

Epoch	Training Loss	Validation Loss	Quad Kappa
1	0.353700	0.472789	0.704830
2	0.290600	0.288312	0.822021
3	0.189700	0.280221	0.829067



04. PROMPT ENGINEERING VỚI META LLAMA 3 8B INSTRUCT

- Giới thiệu mô hình
 - Các bước thực hiện
- 

GIỚI THIỆU MÔ HÌNH META LLAMA 3 8B INSTRUCT

01

- Là một mô hình ngôn ngữ lớn với khoảng 8 tỷ tham số.
- Thuộc họ Encoder-decoder models.
- Được sử dụng chủ yếu cho tác vụ Text Generation.

02

- Được áp dụng vào bài toán bằng cách:
- Đưa bài luận và các hướng dẫn chấm điểm vào mô hình.
 - Yêu cầu mô hình sinh ra một đoạn văn bản chấm điểm cho bài luận đó.

CÁC BƯỚC THỰC HIỆN

01 - PIPELINE

02 - PROMPT ENGINEERING

03 - LẶP LẠI

- Pipeline được khởi tạo với task là text-generation và model là Meta-Llama-3-8B-Instruct

```
pipe = pipeline('text-generation', 'meta-llama/Meta-Llama-3-8B-Instruct',  
                device_map='auto')
```

CÁC BƯỚC THỰC HIỆN

01 - PIPELINE

02 - PROMPT ENGINEERING

03 - LẶP LẠI

Prompt Engineering gồm các vai trò với nội dung tương ứng như sau:

- **system:** Cho biết ngữ cảnh và chỉ dẫn ban đầu của prompt.
- **user:** Yêu cầu đặt ra. Ở đây yêu cầu chấm điểm từ 1-6 và kèm theo đoạn văn.
- **assistant:** Phản hồi mà người dùng mong muốn.

```
prompt = [  
    {'role': 'system', 'content': "As a teacher, you are going to grade your student's essay."},  
    {'role': 'user', 'content': instruction + essay},  
    {'role': 'assistant', 'content': f'\n\nThe score is: '}]
```

CÁC BƯỚC THỰC HIỆN

01 - PIPELINE

02 - PROMPT ENGINEERING

03 - LẶP LẠI

- Với mỗi bài luận trong dataset, thực hiện các bước trên để sinh ra điểm số cho bài luận đó.
- Để thuận tiện, có thể dùng method map của pandas để áp dụng hàm cho từng dòng trong dataset.

Phân tích dữ liệu thông minh

CẢM ƠN

Nhóm 15