ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
ĐẠI HỌC KHOA HỌC TỰ NHIÊN THÀNH PHỐ HỒ CHÍ MINH

KHOA CÔNG NGHỆ THÔNG TIN
Học thống kê

BÁO CÁO

Công cụ tạo comment code cho ngôn ngữ Python

Giảng viên: Ngô Minh Nhựt
Lê Long Quốc

Nhóm sinh viên thực hiện:

Họ và tên	Lê Ngọc Đức
MSSV	20120059
Họ và tên	Nguyễn Thái Chung
MSSV	20120002

Mục lục

Chương 1

Bộ dữ liệu huấn luyện

1.1 Thông tin cơ bản

CodeSearchNet tập hợp khoảng 6 triệu hàm từ 6 ngôn ngữ khác nhau bao gồm: *java*, *go*, *python*, *javascript*, *ruby*, *php*. Code. CodeSearchNet cũng chứa ngôn ngữ tự nhiên giống như truy vấn được tạo tự động cho 2 triệu chức năng, thu được từ tài liệu chức năng liên quan đến việc loại bỏ và xử lý trước một cách máy móc.

func_code_string (string)	func_code_tokens (sequence)	func_documentation_string (string)	func_documentation_tokens (sequence)	split_name (string)
"def train(train_dir, model_save_path=None, n_neighbors=None, knn_algo='ball_tree',…	["def", "train", "(", "train_dir", ",", "model_save_path", "=", "None", ",",…	"Trains a k-nearest neighbors classifier for face recognition. :param train_dir: directory that…	["Trains", "a", "k", "-", "nearest",…	"train"
"def predict(X_img_path, knn_clf=None, model_path=None, distance_threshold=0.6): """…	["def", "predict", "(", "X_img_path", ",", "knn_clf", "=", "None", ",", "model_path", "=",…	"Recognizes faces in given image using a trained KNN classifier :param X_img_path: path to image t…	["Recognizes", "faces", "in", "given", "image",…	"train"
"def show_prediction_labels_on_image(img_path, predictions): """ Shows the face recognition…	["def", "show_prediction_labels_on_image", "(", "img_path", ",", "predictions", ")", ":",…	"Shows the face recognition results visually. :param img_path: path to image to be recognized…	["Shows", "the", "face", "recognition", "results…	"train"
"def _rect_to_css(rect): """ Convert a dlib 'rect' object to a plain tuple in (top, right, bottom,…	["def", "_rect_to_css", "(", "rect", ")", ":", "return", "rect", ".", "top", "(", ")", ",",…	"Convert a dlib 'rect' object to a plain tuple in (top, right, bottom, left) order :param rect: a…	["Convert", "a", "dlib", "'", "rect", "object", "to",…	"train"
"def _trim_css_to_bounds(css, image_shape): """ Make sure a tuple in (top, right, bottom, left)…	["def", "_trim_css_to_bounds", "(", "css", ",", "image_shape", ")", ":", "return", "max", "(",…	"Make sure a tuple in (top, right, bottom, left) order is within the bounds of the image. :param…	["Make", "sure", "a", "tuple", "in", "(",…	"train"
"def face_distance(face_encodings, face_to_compare): """ Given a list of face…	["def", "face_distance", "(", "face_encodings", ",", "face_to_compare", ")", ":", "if", "len", "…	"Given a list of face encodings, compare them to a known face encoding and get a euclidean distance…	["Given", "a", "list", "of", "face",…	"train"
"def load_image_file(file, mode='RGB'): """ Loads an image file (.jpg, .png, etc) into a numpy arra…	["def", "load_image_file", "(", "file", ",", "mode", "=", "'RGB'", ")", ":", "im", "=", "PIL",…	"Loads an image file (.jpg, .png, etc) into a numpy array :param file: image file name or file…	["Loads", "an", "image", "file", "(", "."jpg"…	"train"
"def _raw_face_locations(img, number_of_times_to_upsample=1, model="hog"): """…	["def", "_raw_face_locations", "(", "img", ",", "number_of_times_to_upsample", "=", "1", ",",…	"Returns an array of bounding boxes of human faces in a image :param img: An image (as a numpy array…	["Returns", "an", "array", "of",…	"train"
"def face_locations(img, number_of_times_to_upsample=1, model="hog"): """…	["def", "face_locations", "(", "img", ",", "number_of_times_to_upsample", "=", "1", ",",…	"Returns an array of bounding boxes of human faces in a image :param img: An image (as a numpy array…	["Returns", "an", "array", "of",…	"train"

Hình 1.1: Bộ dữ liệu được lấy từ Tensorflow [1].

1.2 Thành phần bộ dữ liệu

Bộ dữ liệu bao gồm các cột:

- *repository_name*:
- *func_path_in_repository*:
- *func_name*:
- *whole_func_string*:
- *language*:
- *func_code_string*:
- *func_code_tokens*:

- *func_documentation_string*:
- *func_documentation_tokens*:
- *split_name*:
- *func_code_url*:

Cụ thể như hình 1 và 3. Trong mô hình này, chúng em sử dụng bộ dữ liệu cho ngôn ngữ python. Tương tự các ngôn ngữ còn lại, bộ dữ liệu được chia thành ba phần chính:

- *train*: có 412178 mẫu.
- *validation*: có 23107 mẫu.
- *test*: có 22176 mẫu.

1.3 Dữ liệu mẫu

Để hiểu dễ dàng hơn, bộ dữ liệu mẫu được mô tả ở [2].

```
{
    "repository\_name": {
        "dtype": "string",
        "id": null,
        "\_type": "Value"
    },
    "func\_path\_in\_repository": {
        "dtype": "string",
        "id": null,
        "\_type": "Value"
    },
    "func\_name": {
        "dtype": "string",
        "id": null,
        "\_type": "Value"
    },
    "whole\_func\_string": {
        "dtype": "string",
        "id": null,
        "\_type": "Value"
    },
    "language": {
        "dtype": "string",
        "id": null,
        "\_type": "Value"
    },
    "func\_code\_string": {
        "dtype": "string",
        "id": null,
        "\_type": "Value"
    },
    "func\_code\_tokens": {
        "feature": {
            "dtype": "string",
            "id": null,
            "\_type": "Value"
        },
        "length": -1,
        "id": null,
        "\_type": "Sequence"
    },
    "func\_documentation\_string": {
        "dtype": "string",
        "id": null,
        "\_type": "Value"
    },
    "func\_documentation\_tokens": {
        "feature": {
            "dtype": "string",
            "id": null,
            "\_type": "Value"
        },
        "length": -1,
        "id": null,
        "\_type": "Sequence"
    }
}
```


Listing 1: Kiến trúc bộ dữ liệu.

```
{
    "split\_name": {
        "dtype": "string",
        "id": null,
        "\_type": "Value"
    },
    "func\_code\_url": {
        "dtype": "string",
        "id": null,
        "\_type": "Value"
    }
}
```

Listing 2: Kiến trúc bộ dữ liệu (phần sau).

Chương 2

Mô hình huấn luyện

2.1 Truy cập bộ dữ liệu

Tại đây ta sử dụng thư viện hỗ trợ của Tensorflow để truy xuất database từ cơ sở dữ liệu. Diễn giải chi tiết hơn có thể được tìm thấy tại [3]

```
data = tfds.load('huggingface:code_search_net/python', split=spl,
↪  shuffle_files=True)
```

Listing 3: Câu lệnh truy xuất dữ liệu. Tại đây, *'huggingface:code_search_net/python'* chính là đường dẫn đến bộ dữ liệu, *spl* có thể là một trong ba string: 'train', 'validation' và 'test'. shuffle_files mang kiểu dữ liệu bool, có xáo trộn các tệp đầu vào hay không, mặc định là ***False***.

2.2 Đặc trưng mô hình

Mô hình được xây dựng dựa trên nền tảng bộ giải mã Transformer 2.1 (được đề xuất trong bài báo [4]) và bộ mã hóa của mô hình Roberta (được đề xuất trong bài báo [5]). Với bộ tham số được tiền xử lý ở Roberta và Transformer, thời gian đào tạo mô hình được giảm đi đáng kể.

2.2.1 Tham số mô hình

Mô hình được tạo có đến hàng triệu tham số như được trình bày trong hình 2.1. Các thiết lập cấu hình của mô hình được lấy từ mô hình đã được đào tạo từ trước là *codebert-base*. Cả ba mô hình đều được đào tạo với $learningrate = 0.00001$, tuy vậy, có xây dựng early_stop để kịp thời kết thúc quá trình đào tạo của epoch hiện tại. Các mô hình được đào tạo qua 5 epoch, tùy theo mô hình mà batch_size được cài đặt riêng, cụ thể, có ba giá trị batch_size được cài đặt là 16, 32 và 64.

2.2.2 Tách từ

Ở đây em thực hiện đào tạo trên hai mô hình với hai bộ tokenizer trên nền tảng của hai mô hình khác nhau là: *deepset/roberta-base-squad2* [6] và *stsb-roberta-base* [7]. Cả hai trong số đó đều được xây dựng trên mô hình nền tảng là *roberta*.

```
   | Name         | Type                | Params
-------------------------------------------------
0  | encoder      | RobertaModel        | 124 M
1  | decoder      | TransformerDecoder  | 47.3 M
2  | dense_layer  | Linear              | 590 K
3  | generator    | Linear              | 38.6 M
4  | dropout      | Dropout             | 0
5  | log_softmax  | LogSoftmax          | 0
-------------------------------------------------
172 M     Trainable params
0         Non-trainable params
172 M     Total params
690.014   Total estimated model params size (MB)
```

Hình 2.1: Số lượng các tham số mô hình.

Và do đó, có thể nói rằng chúng đều có bản chất là *transformer* hoặc có một phần bản chất là *transformer*.

2.3 Khắc phục overfitting

Có nhiều phương pháp để khắc phục overfitting [8]. Trong đó, trong mô hình này, chúng em sử dụng đến *early stopping* và *kiểm tra độ chính xác của mô hình trên dữ liệu xác thực*.

2.4 Ảnh hưởng của batch size đến quá trình huấn luyện

2.4.1 Thời gian

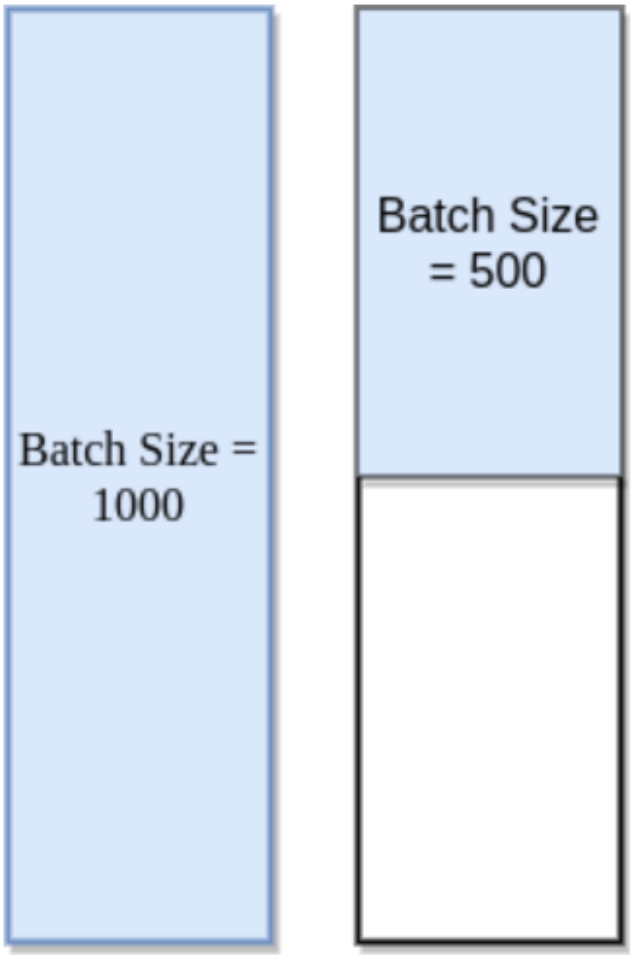
Theo như phần 6 trong sách của Goodfellow [9], *"các lô nhỏ có thể mang lại hiệu quả chính quy hóa (Wilson và Martinez, 2003), có lẽ do nhiễu mà chúng thêm vào quá trình học tập. Lỗi tổng quát hóa thường là tốt nhất đối với kích thước lô là 1. Huấn luyện với kích thước lô nhỏ như vậy có thể yêu cầu tốc độ học nhỏ để duy trì sự ổn định do phương sai cao trong ước tính độ dốc. Tổng thời gian chạy có thể rất cao do cần phải thực hiện nhiều bước hơn, cả do tốc độ học tập giảm và do cần nhiều bước hơn để quan sát toàn bộ tập huấn luyện."*.

Cụ thể trong quá trình huấn luyện mô hình, chúng em nhận thấy rằng với cùng một số lượng *epoch* và tỉ lệ *learning rate*, *batch size* lớn hơn thì mô hình sẽ huấn luyện nhanh hơn (thời gian huấn luyện mỗi epoch giảm đi). Thay vào đó, có một sự đánh đổi về bộ nhớ, cụ thể là mô hình với batch size bằng 64 yêu cầu khoảng

22 GB GPU RAM, trong khi đó, giá trị đó chỉ là 13.3 GB GPU RAM ở mức batch size bằng 32.

Để giải thích cho vấn đề này, chúng ta nhận thấy rằng, thời gian thực hiện một batch size lớn có lớn hơn, nhưng độ chênh lệch rất nhỏ thời gian thực hiện một batch size nhỏ hơn. Khi batch size lớn thì số lượng batch được thực hiện sẽ ít hơn2.2, điều này giúp cho thời gian thực hiện mỗi epoch giảm đi gần như tương đương với tỷ lệ độ tăng batch size.

Tuy vậy, một vấn đề cần quan tâm đó là, thời gian huấn luyện nhanh hơn không đồng nghĩa với khả năng hội tụ nhanh hơn. Cụ thể là, hệ số lỗi của mô hình có $batchsize = 32$ từ $loss = 4.911$ ở epoch đầu tiên đã giảm xuống $loss = 0.753$ ngay ở epoch thứ 2. Trong khi đó, mô hình có $batchsize = 64$ từ $loss = 5.633$ ở epoch đầu tiên đã giảm xuống $loss = 4.794$ ở epoch thứ 2.



Hình 2.2: Mối liên hệ giữa batch size và thời gian thực hiện epoch.

2.4.2 Khả năng tổng quát hóa

Theo bài báo [10], *"trong thực tế, người ta đã quan sát thấy rằng khi sử dụng một lô lớn hơn, chất lượng của mô hình sẽ bị suy giảm đáng kể, được đo bằng khả năng khái quát hóa của nó. Đã có một số nỗ lực điều tra nguyên nhân của sự sụt giảm tổng quát hóa này trong chế độ lô lớn, tuy nhiên câu trả lời chính xác cho hiện tượng này vẫn chưa được biết. Trong bài báo này, chúng tôi trình bày bằng*

chứng bằng số phong phú ủng hộ quan điểm rằng các phương pháp lô lớn có xu hướng hội tụ thành các cực tiểu sắc nét của các chức năng đào tạo và kiểm tra – và cực tiểu sắc nét đó dẫn đến khả năng khái quát hóa kém hơn. Ngược lại, các phương pháp lô nhỏ luôn hội tụ thành các bộ giảm thiểu phẳng và các thử nghiệm của chúng tôi ủng hộ quan điểm thường được cho rằng điều này là do nhiễu cố hữu trong ước tính độ dốc. Chúng tôi cũng thảo luận về một số chiến lược thực nghiệm giúp các phương pháp lô lớn loại bỏ khoảng cách tổng quát hóa và kết luận bằng một loạt các ý tưởng nghiên cứu trong tương lai và các câu hỏi mở.".

2.5 Thước đo đánh giá mô hình

2.5.1 BLUE

BLEU (cơ quan đánh giá song ngữ) là một thuật toán để đánh giá chất lượng văn bản đã được máy dịch từ ngôn ngữ tự nhiên này sang ngôn ngữ tự nhiên khác. Chất lượng được coi là sự tương ứng giữa đầu ra của máy và đầu ra của con người: "bản dịch máy càng gần với bản dịch chuyên nghiệp của con người thì càng tốt" – đây là ý tưởng chính đằng sau BLEU.[1] Được phát minh tại IBM vào năm 2001,[1] BLEU là một trong những thước đo đầu tiên khẳng định mối tương quan cao với đánh giá của con người về chất lượng,[2][3] và vẫn là một trong những thước đo tự động và rẻ tiền phổ biến nhất [11]. Trong ngữ cảnh này, BLUE đánh giá khả năng 'dịch' từ ngôn ngữ máy (code) thành ngôn ngữ con người('text').

Điểm số được tính cho các phân đoạn đã dịch riêng lẻ—thường là các câu—bằng cách so sánh chúng với một tập hợp các bản dịch tham khảo có chất lượng tốt. Những điểm số đó sau đó được tính trung bình trên toàn bộ kho văn bản để đạt được ước tính về chất lượng tổng thể của bản dịch. Tính dễ hiểu hoặc tính đúng ngữ pháp không được tính đến.[cần dẫn nguồn] Đầu ra của BLEU luôn là một số trong khoảng từ 0 đến 1. Giá trị này cho biết mức độ tương tự của văn bản ứng viên với văn bản tham chiếu, với các giá trị càng gần 1 biểu thị văn bản tương tự hơn. Rất ít bản dịch của con người sẽ đạt được điểm 1, vì điều này cho thấy rằng ứng viên giống hệt với một trong các bản dịch tham khảo. Vì lý do này, không nhất thiết phải đạt điểm 1. Vì có nhiều cơ hội khớp hơn nên việc thêm các bản dịch tham khảo bổ sung sẽ làm tăng điểm BLEU [11].

2.5.2 Độ tương tự (sentence similarity)

Mô hình *cross-encoder/stsb-roberta-base* được đào tạo trên tập dữ liệu STS-B, bao gồm các cặp câu được chú thích với điểm số tương đồng nằm trong khoảng từ 0 đến 5. Mô hình này được đào tạo để tối đa hóa điểm số tương đồng cho các cặp văn bản được tương đương hoặc tương tự về mặt ngữ nghĩa và giảm thiểu điểm số cho các cặp không giống nhau.

Biến đầu ra trong mã ví dụ chứa một giá trị vô hướng biểu thị mức độ giống nhau giữa hai văn bản đầu vào, với các giá trị càng cao biểu thị mức độ giống nhau càng lớn.

2.6 Đánh giá kết quả đào tạo qua các biến thể

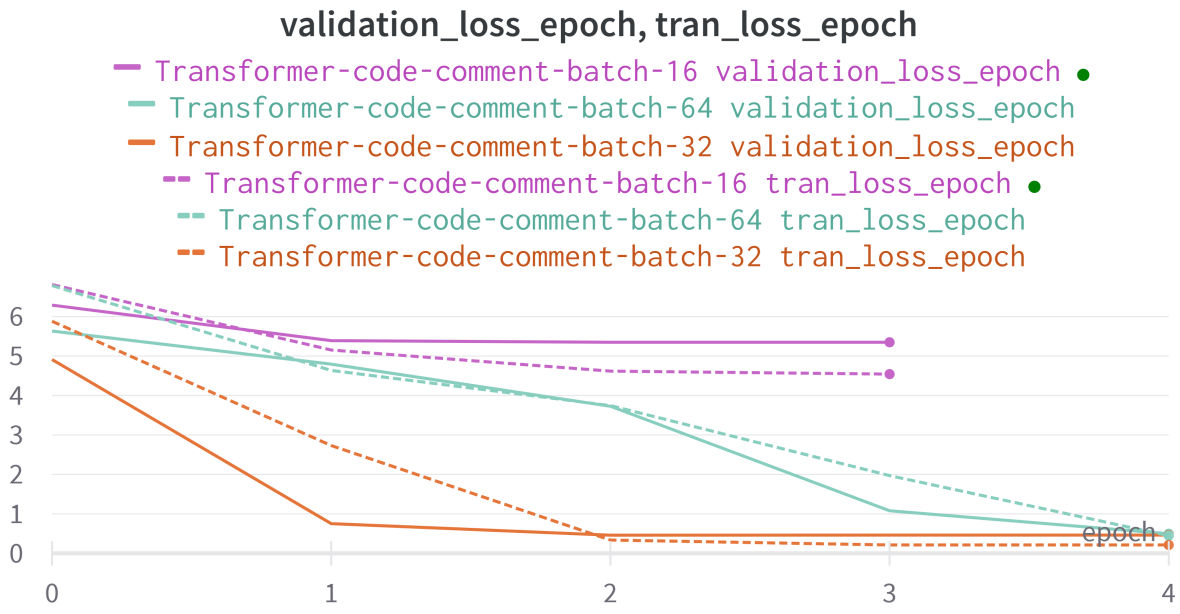
2.6.1 Đặc trưng

- Đường màu xanh biểu diễn cho mô hình có $batch\_size = 64$ và bộ phân tách từ của $deepset/roberta - base - squad2$.
- Đường màu cam biểu diễn cho mô hình có $batch\_size = 32$ và bộ phân tách từ của $stsb - roberta - base$.
- Đường màu tím biểu diễn cho mô hình có $batch\_size = 16$ và bộ phân tách từ của $stsb - roberta - base$.

2.6.2 Mối quan hệ giữa $validation\_loss\_epoch$ và $tran\_loss\_epoch$

Dựa vào hình 2.3, ta có được:

- Đường màu tím cho thấy hiện tượng overfitting ngay trong quá trình đào tạo epoch đầu tiên.
- Đường màu cam cho thấy hiện tượng overfitting bắt đầu tại epoch thứ 2 ($validation\_loss\_epoch > tran\_loss\_epoch$).
- Trong khi đó, có một sự bất ổn được thể hiện trên mối tương quan giữa validation_loss_epoch, tran_loss_epoch của đường màu xanh.

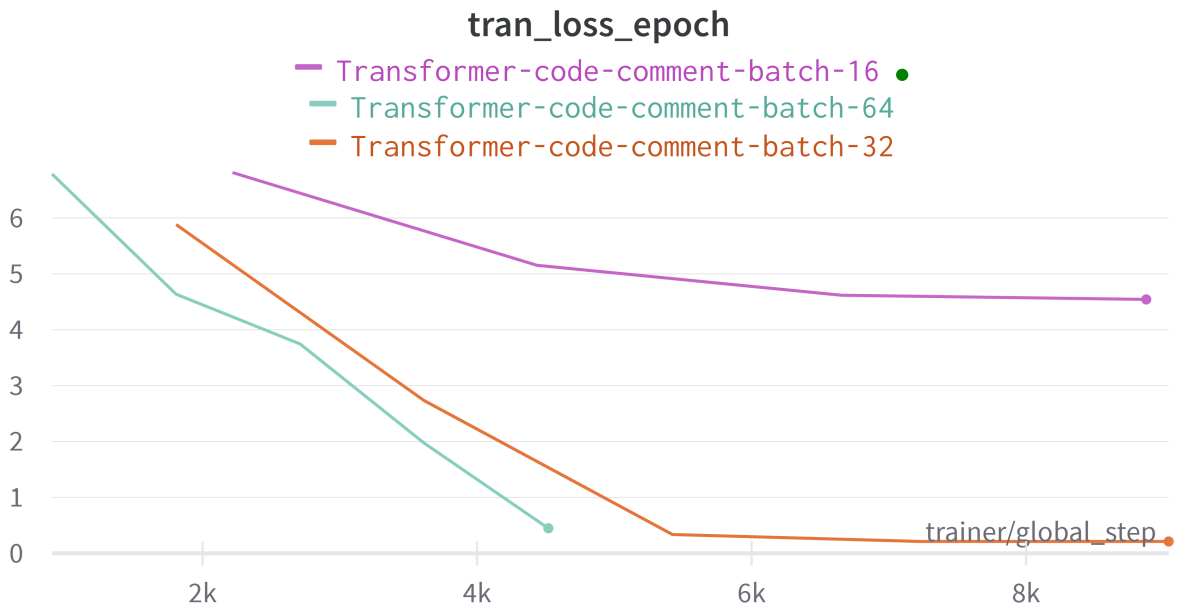


Hình 2.3: Mối liên hệ giữa validation_loss_epoch và tran_loss_epoch.

2.6.3 $tran\_loss\_epoch$

Dựa vào hình 2.4, ta có được:

- Đường màu tím cho thấy giá trị loss có xu hướng tiến về hằng số ngay khi vừa kết thúc epoch đầu tiên.
- Ta nhận thấy giá trị ở đường màu cam có dấu hiệu không giảm sau khi kết thúc epoch thứ 2. Thật vậy, đây cũng là điểm mà giá trị $validation\_loss\_epoch$ bắt đầu vượt qua $tran\_loss\_epoch$.
- Đường màu xanh chỉ thể hiện chiều giảm và chưa cho thấy xu hướng dừng lại. Điều này có thể xảy ra do quá trình đào tạo hoặc bộ dữ liệu chưa đủ.



Hình 2.4: Kết quả hàm loss trong quá trình huấn luyện dữ liệu train.

Chương 3

Ứng dụng web

3.1 Công nghệ sử dụng

Về front-end, nhóm sử dụng framework Nextjs và Tailwind để xây dựng giao diện. Về back-end, nhóm sử dụng Flask/Python để tải checkpoint và tạo api endpoint cho việc dự đoán comment của đoạn code

3.2 Mô tả ứng dụng

Ứng dụng được xây dựng như là một code editor trên nền web, cho phép lập trình Python, nhận input đầu vào và thực thi, sau đó trả về các thông số như kết quả đầu ra, comment được dự đoán, thời gian và bộ nhớ được sử dụng.

3.3 Cấu trúc mã nguồn ứng dụng

Mã nguồn của ứng dụng bao gồm:

- *src*: Chứa mã nguồn front-end
- *back-end*: Chứa mã nguồn back-end
  - *api*: Thư mục chứa checkpoint và mã nguồn khởi chạy back-end server

Để sử dụng ứng dụng, chạy câu lệnh **yarn start** để khởi chạy đồng thời back-end và front-end, sau đó truy cập vào *http://localhost:3000* để sử dụng ứng dụng. Dưới đây là hình ảnh từ ứng dụng:

3.4 Sử dụng ứng dụng

- Cài đặt node, yarn và các dependencies của dự án (**yarn install**)
- Cài đặt python3 và các module cần thiết cho back-end
- Tải checkpoint file model-batch-16.pt và đặt vào thư mục back-end/api
  - ID drive: 1_V972LTW7wQ1IQm7neCc9oKMnC_do7rw

- Tạo file .env để cấu hình endpoint của back-end, ví dụ: BACK_END_ENDPOINT = "http://127.0.0.1:3001"
- Chạy cả back-end và front-end: yarn start
- Truy cập http://localhost:3000 (front-end) để kiểm thử ứng dụng, back-end là http://localhost:3001

#Codecom Code comment generator

Python (3.8.1) Select Theme

```python
def bubble_sort(arr):
    n = len(arr)

    for i in range(n - 1):
        # Last i elements are already in place
        for j in range(n - i - 1):
            # Compare adjacent elements
            if arr[j] > arr[j + 1]:
                # Swap elements if they are in the wrong o
                arr[j], arr[j + 1] = arr[j + 1], arr[j]

    return arr

a = list(map(int, input().split()))
print(bubble_sort(a))
```

Hình 3.1: Bubble sort

Tài liệu tham khảo

[1] [Online]. Available: https://www.tensorflow.org/datasets/community_catalog/huggingface/code_search_net

[2]

[3] [Online]. Available: https://www.tensorflow.org/datasets/api_docs/python/tfds/load

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017.

[5] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," 2019.

[6]

[7]

[8] A. Soleymani, "Your validation loss is lower than your training loss? this is why!" Apr 2022. [Online]. Available: https://towardsdatascience.com/what-your-validation-loss-is-lower-than-your-training-loss-this-is-why-5e92e0b1747

[9] Goodfellow. [Online]. Available: https://www.deeplearningbook.org/contents/optimization.html

[10] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, "On large-batch training for deep learning: Generalization gap and sharp minima," 2017.

[11] Jun 2023. [Online]. Available: https://en.wikipedia.org/wiki/BLEU