ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

~~~~~📖~~~~~

BÁO CÁO MÔN HỌC THỐNG KÊ

NỘI DUNG: MACHINE TRANSLATION

Thực hiện:

1. Nguyễn Ngọc Thúy Quỳnh 20120176
2. Phan Phong Lưu 20120326
3. Lương Văn Triều 20120604
4. Khúc Xuân Trường 20120610

Giáo viên hướng dẫn: Thầy Lê Long Quốc

TP Hồ Chí Minh, ngày 29 tháng 06 năm 2023
~~~~~

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

~~~~~📖~~~~~

BÁO CÁO MÔN HỌC THỐNG KÊ

NỘI DUNG: MACHINE TRANSLATION

Thực hiện:

1. Nguyễn Ngọc Thúy Quỳnh 20120176
2. Phan Phong Lưu 20120326
3. Lương Văn Triều 20120604
4. Khúc Xuân Trường 20120610

TP Hồ Chí Minh, ngày 29 tháng 06 năm 2023
~~~~~

MỤC LỤC

I. BẢNG ĐÁNH GIÁ THÀNH VIÊN

MSSV	Họ Và Tên	Phần Trăm Đóng Góp	Mức Độ Hoàn Thành
20120176	Nguyễn Ngọc Thúy Quỳnh	25%	Đầy đủ, tốt
20120326	Phan Phong Lưu	25%	Đầy đủ, tốt
20120604	Lương Văn Triều	25%	Đầy đủ, tốt
20120610	Khúc Xuân Trường	25%	Đầy đủ, tốt

II. NỘI DUNG

1. Model

1.1. Mô tả chi tiết tập dữ liệu

- Tập dữ liệu sử dụng gồm các cặp câu song ngữ Anh và Việt cách nhau một khoảng tab (tab space) lưu trong file txt, khi ta huấn luyện cho mô hình sẽ đưa các file txt này để huấn luyện mô hình.

1.2. Kiến trúc Transformer

- **Kiến trúc Transformer** hoạt động bằng cách sử dụng cơ chế self-attention và kiến trúc mã hóa-giải mã. Cách hoạt động tổng quan của mô hình Transformer.
- **Kiến trúc mã hóa (Encoder):**

  + Dữ liệu đầu vào, chẳng hạn như một câu văn bản, được biểu diễn dưới dạng các vector từ (word embeddings).
  + Mỗi từ trong câu văn bản được truyền qua một layer self-attention để tạo ra một biểu diễn kết hợp (contextualized representation) cho từ đó. Cơ chế self-attention cho phép mô hình tập trung vào các từ quan trọng và xác định mối quan hệ giữa chúng.

+ Sau layer self-attention, một layer feed-forward network được áp dụng để tạo ra biểu diễn mới cho mỗi từ.
+ Quá trình trên được lặp lại nhiều lần để tạo ra các layer mã hóa (encoder layers). Mỗi layer mã hóa có thể học được các biểu diễn phức tạp hơn của câu văn bản.

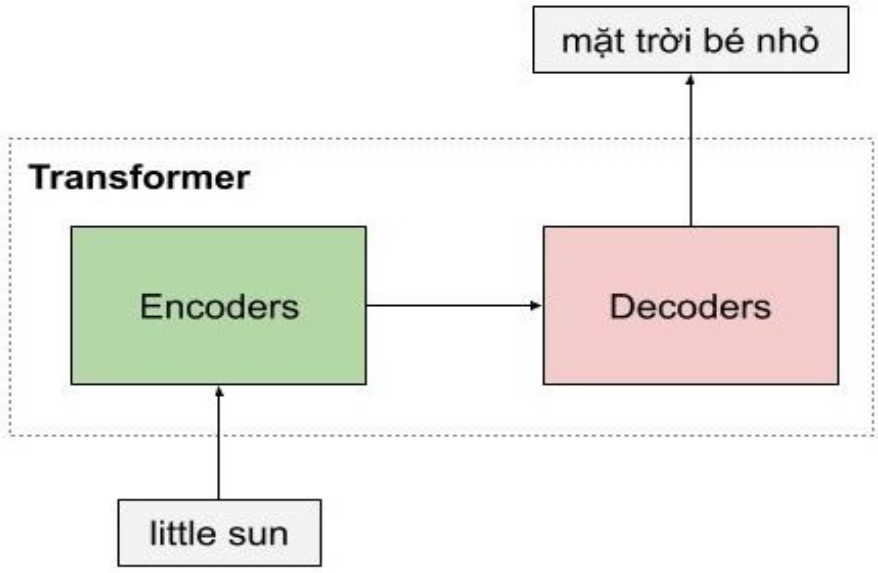
- **Kiến trúc giải mã (Decoder):**

+ Mô hình sử dụng một layer self-attention trong quá trình giải mã để tạo ra một biểu diễn kết hợp cho từ đang được dự đoán. Layer self-attention này giúp mô hình biết được các từ đã được dự đoán trước đó.
+ Sau layer self-attention, mô hình sử dụng một layer attention khác để tìm hiểu mối quan hệ giữa các từ trong câu văn bản đầu vào (source sentence) và các từ đã được dự đoán trước đó.
+ Từ biểu diễn kết hợp của layer attention, một layer feed-forward network được áp dụng để tạo ra biểu diễn mới cho từ đang được dự đoán.
+ Quá trình trên được lặp lại nhiều lần để tạo ra các layer giải mã (decoder layers). Mỗi layer giải mã sử dụng thông tin từ quá trình dự đoán trước đó để cải thiện dự đoán tiếp theo.

- **Tạo dự đoán:**

+ Đầu ra của mô hình Transformer là các biểu diễn của các từ đã được giải mã. Các biểu diễn này được đưa qua một fully connected layer (fc_out) và softmax để tính xác suất của các từ trong bộ từ vựng đích.
+ Dựa trên xác suất này, từ có xác suất cao nhất được chọn là từ được dự đoán tiếp theo.
+ Mô hình Transformer được huấn luyện bằng cách tối thiểu hóa sai số giữa các từ đã được dự đoán và các từ thực tế trong quá trình huấn

luyện. Quá trình huấn luyện này thường sử dụng giải thuật lan truyền ngược (backpropagation) và thuật toán tối ưu hóa gradient descent.



- **Một trong những điểm mạnh của mô hình Transformer** là khả năng xử lý song song cho các từ trong quá trình huấn luyện và dự đoán. Điều này làm cho Transformer trở thành một lựa chọn phổ biến trong các tác vụ dịch máy và xử lý ngôn ngữ tự nhiên khác. Dưới đây là các khía cạnh liên quan đến khả năng xử lý song song trong mô hình Transformer:
  + Cơ chế self-attention: Cơ chế self-attention trong Transformer cho phép mô hình tập trung vào các từ quan trọng trong câu và xác định mối quan hệ giữa chúng. Quá trình này có thể được thực hiện song song đối với các từ khác nhau trong câu, mà không cần phải tuần tự xử lý từng từ một. Điều này giúp tăng tốc độ tính toán và khả năng xử lý song song của mô hình.
  + Attention đa đầu (Multi-head attention): Mô hình Transformer sử dụng attention đa đầu, cho phép mô hình tập trung vào các khía cạnh khác nhau của câu văn đầu vào. Mỗi đầu attention có thể hoạt động độc

lập, xử lý một phần của dữ liệu đầu vào. Điều này cho phép mô hình chia sẻ tải tính toán và thực hiện xử lý song song cho các phần khác nhau của câu.

+ Paralelism trong quá trình mã hóa-giải mã: Mô hình Transformer có thể thực hiện quá trình mã hóa và giải mã song song. Điều này đồng nghĩa với việc nhiều câu văn bản có thể được mã hóa và giải mã cùng một lúc trên các luồng tính toán độc lập. Điều này giúp tăng tốc độ tính toán và hiệu suất xử lý của mô hình trong các tác vụ song song.

- **Tóm lại**, Kiến trúc Transformer được thiết kế với khả năng xử lý song song trong việc xử lý các từ và câu văn bản. Các cơ chế self-attention, attention đa đầu và khả năng mã hóa-giải mã song song đóng vai trò quan trọng trong việc tăng tốc độ tính toán và cải thiện hiệu suất của mô hình trong các tác vụ học máy và xử lý ngôn ngữ tự nhiên.

1.3. Mô hình trong bài toán

- Dựa trên kiến trúc Transformer mà để thực hiện mô hình này.
- Ở đây mô hình được xây dựng bằng Pytorch.
- Quá trình:

  + Đầu vào là một câu nguồn (source sequence) và câu đích (target sequence). Cả hai sau đó sẽ được tách thành các từ riêng biệt và mỗi từ sẽ mang một con số riêng (biểu diễn số hóa). Từ đó tạo thành từ vựng nguồn (source vocab) và từ vựng đích (target vocab).

  + Tạo một mặt nạ nguồn để chỉ ra vị trí padding trong câu nguồn và tạo một mặt nạ đích sử dụng phương thức generate_square_subsequent_mask của module nn.Transformer.

  + Tiếp theo, câu nguồn sẽ đi qua lớp nhúng nguồn (source embedding) Câu đích sẽ đi qua lớp nhúng đích (target embedding). Mô hình sau đó áp dụng cơ chế chú ý tự động và các lớp tiếp diễn truyền (feed-forward)

để nắm bắt các phụ thuộc giữa các từ nguồn đã được nhúng (source word embeddings) và các từ đích đã được nhúng (target word embeddings). Kết quả đầu ra của mô hình là biểu diễn dịch của câu đích.
+ Biểu diễn dịch sau đó được truyền qua lớp tuyến tính để tạo ra điểm số cho mỗi từ trong từ vựng đích. Những từ có điểm số cao nhất sẽ được chọn làm dự đoán cho mỗi vị trí trong câu đích. Sau đó, dựa vào từ vựng đích (target vocab) để chuyển các số này thành các từ, cuối cùng tạo thành một câu hoàn chỉnh là câu được dịch từ câu nguồn.

- **Ngoài ra**, mô hình còn áp dụng dropout cho các từ đã được nhúng của câu nguồn và câu đích nhằm ngăn ngừa overfitting.

1.4. Cách tổ chức và phân chia dữ liệu

```
# Tạo dataset từ example
dataset = Dataset(examples, fields=[('src', source), ('trg', target)])

# Chia train, valid, test
train_data, valid_data, test_data = dataset.split(split_ratio=[0.7, 0.15, 0.15])
```

- **Đầu tiên ta tạo dataset từ example**

+ dataset(examples, fields=[('src', source), ('trg', target)]): Đây là cách để tạo một đối tượng dataset từ các ví dụ (examples). Đối tượng dataset này được tạo bằng cách sử dụng lớp Dataset và nhận hai tham số chính:

+ examples: Một danh sách các ví dụ, mỗi ví dụ là một từ điển (dictionary) với các cặp key-value tương ứng với các trường dữ liệu và giá trị tương ứng.

+ fields: Một danh sách các cặp (field_name, field) xác định các trường dữ liệu trong ví dụ và các xử lý (processing) tương ứng cho từng trường. Trong trường hợp này, có hai cặp (src, source) và (trg, target) được sử

dụng. Đây là cách để xác định rằng trường 'src' trong ví dụ tương ứng với trường 'source' và trường 'trg' tương ứng với trường 'target'.

- **Chia train, valid, test**

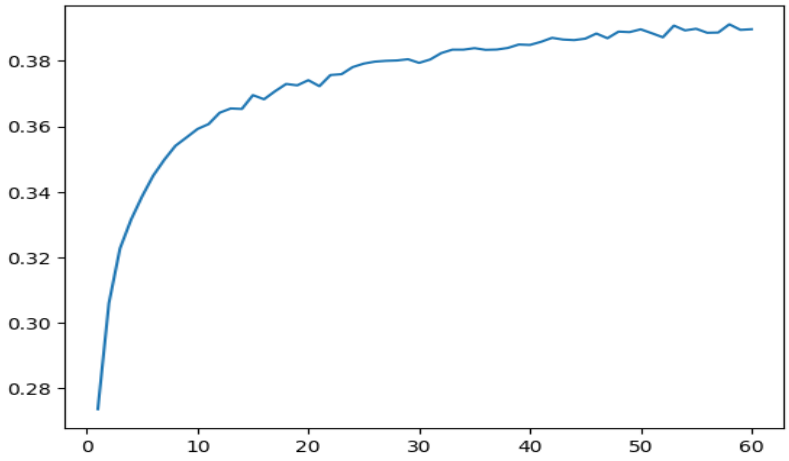
  + dataset.split(split_ratio=[0.7, 0.15, 0.15]): Đoạn code này chia dataset thành ba phần: tập train (train_data), tập validation (valid_data) và tập test (test_data) theo tỷ lệ phần trăm tương ứng được xác định trong split_ratio.

  + split_ratio: Đây là một danh sách các phần trăm xác định tỷ lệ chia cho mỗi tập dữ liệu. Trong trường hợp này, tỷ lệ là [0.7, 0.15, 0.15], tức là 70% dữ liệu được chia vào tập train, 15% vào tập validation và 15% vào tập test.

  + Sau khi thực hiện hai đoạn code trên, bạn sẽ có ba biến train_data, valid_data, và test_data, mỗi biến chứa tập dữ liệu tương ứng đã được chia theo tỷ lệ đã xác định.

1.5. Kết quả đánh giá mô hình

- Dưới đây là biểu đồ độ chính xác sau 60 Epoch:

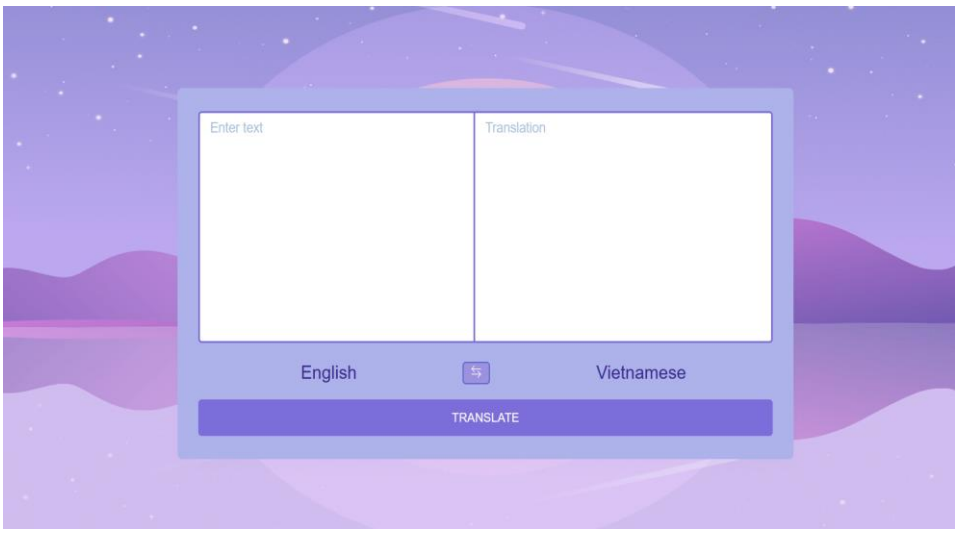


- **Độ chính xác chưa được cao, một phần lý do dữ liệu train chưa được nhiều (thiếu CPU).**

2. Web

2.1. Frontend

- Phần giao diện Website, sử dụng thư viện React của ngôn ngữ Javascript để thực hiện.
- Giao diện sẽ gồm các thông tin, nút bấm như sau:



+ Ô text bên trái sẽ chứa thông tin mà mình muốn dịch.

+ Ô text bên phải sẽ chứa thông tin kết quả sau khi dịch xong.

+ Nút chuyển ngôn ngữ, ta có thể lựa chọn dịch từ tiếng Anh qua tiếng Việt hoặc là từ tiếng Việt qua tiếng Anh (dịch được hai chiều).

+ Nút bấm Translate, khi ta bấm vào nút này thì nó sẽ bắt đầu dịch.

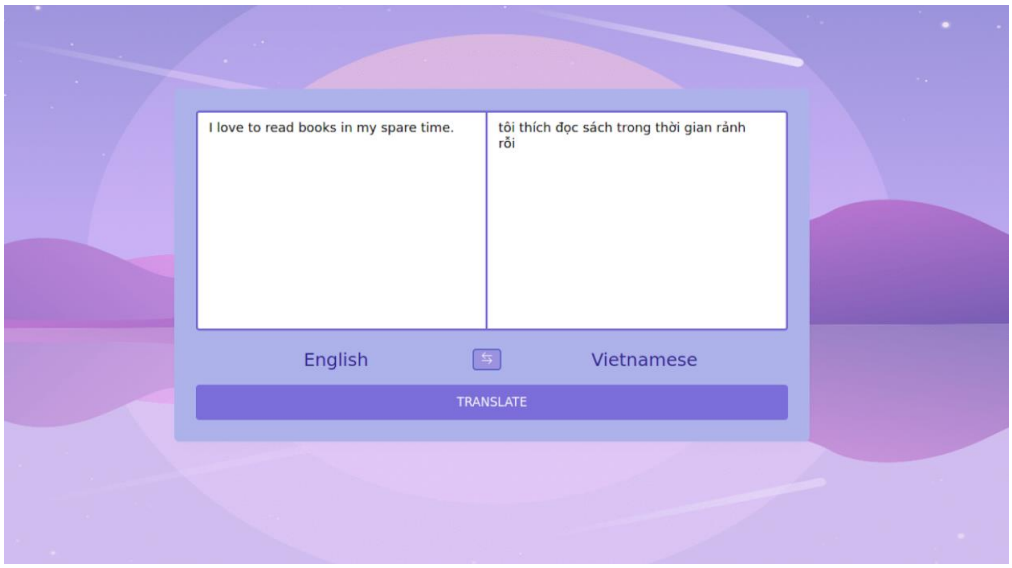
2.2. Backend

- Sử dụng framework Flask Python để thực hiện phần backend, ở backend ta sẽ thực hiện tạo ra các API để cho bên giao diện (frontend) sử dụng. Phần giao diện (frontend) sẽ gửi yêu cầu kèm theo dữ liệu cần dịch tới backend và backend sẽ trả lại dữ liệu sau khi đã dịch thành công. Chúng giao tiếp với nhau thông qua API.

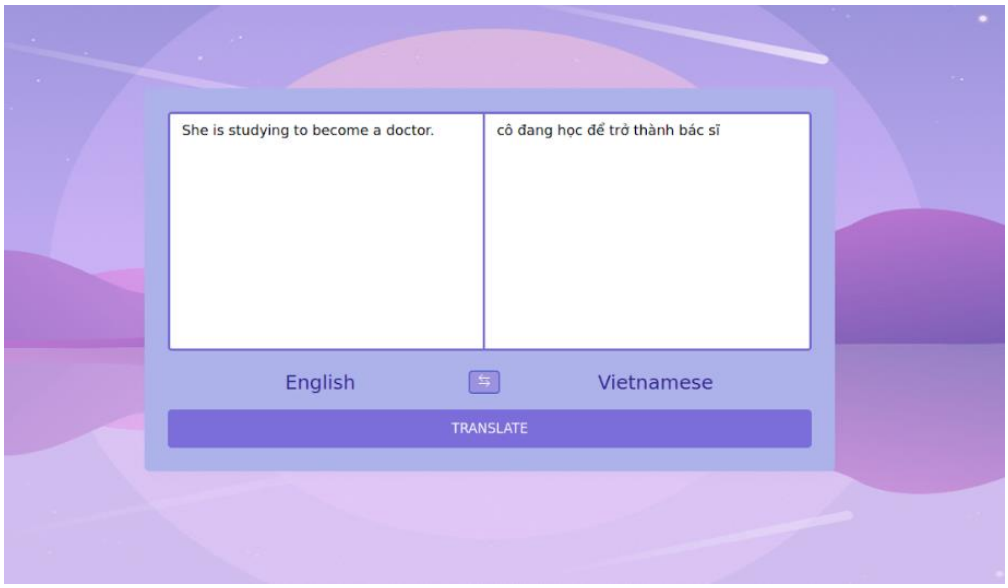
3. Test case

3.1. Dịch từ tiếng Anh sang tiếng Việt

- **Test case 1**



- **Test case 2**



- **Test case 3**

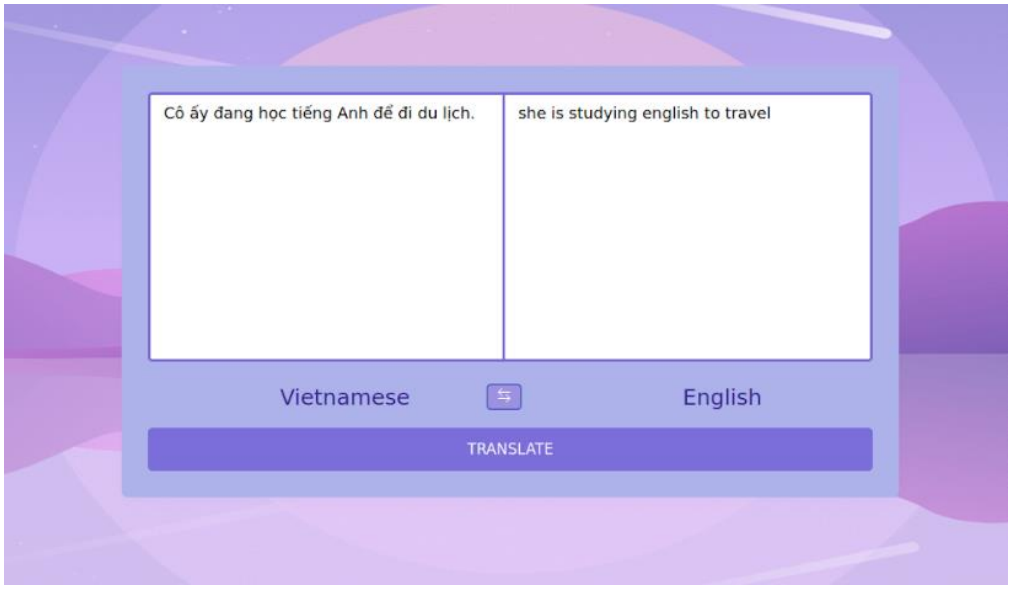
- **Test case 4**

- **Test case 5**

3.2. Dịch từ tiếng Việt sang tiếng Anh

- **Test case 1**

- **Test case 2**



- **Test case 3**

- **Test case 4**

- **Test case 5**

4. Các thư viện và framework sử dụng

- **Model:**

+ Torchtext: Thư viện hỗ trợ xử lý dữ liệu văn bản và tạo dataset cho việc huấn luyện mô hình học máy trong PyTorch.

+ Spacy: Thư viện xử lý ngôn ngữ tự nhiên (NLP) với các chức năng như tách từ, phân tích cú pháp, nhận diện thực thể, v.v.

+ Pyvi: Một công cụ xử lý tiếng Việt dựa trên pyvi.vn, cung cấp các chức năng như tách từ, gắn thẻ từ loại, gắn thẻ ngữ cảnh, v.v.

+ En_core_web_sm: Một phiên bản của thư viện spacy cho ngôn ngữ tiếng Anh, được sử dụng để xử lý văn bản tiếng Anh.

+ Langdetect: Một thư viện để nhận diện ngôn ngữ của văn bản dựa trên các đặc trưng ngôn ngữ.

+ Contractions: Một thư viện để mở rộng từ viết tắt trong tiếng Anh thành các từ đầy đủ tương ứng.

+ Dill: Một thư viện hỗ trợ serialize và deserialize các đối tượng Python.

Ngoài ra, còn sử dụng một số Mô-đun, thư viện và framework khác.

- **Website:**

+ ReactJS: Thư viện sử dụng để tạo giao diện website nhanh chóng.

+ Bootstrap: Framework sử dụng để phát triển giao diện nhanh chóng.

+ Flask: Framework sử dụng để làm backend dễ dàng và tiện lợi.

5. Hướng phát triển trong tương lai

- Cải thiện model để cho nó dịch nhanh và chính xác hơn.
- Trên website sẽ có thêm một số chức năng như lưu lại lịch sử dịch, dự đoán các nghĩa tương tự.

III. TÀI LIỆU THAM KHẢO

1. https://github.com/pbcquoc/transformer?fbclid=IwAR0RcIkMxcCTqR_5s7WLmqkjNhrOmiM8c2QCJGdiUU1unkPUYQCTqnrp9l8
2. https://viblo.asia/p/tim-hieu-ve-kien-truc-transformer-Az45byM6lxY
3. https://www.kaggle.com/code/nageshsingh/neural-machine-translation-attention-mechanism/notebook
4. https://www.analyticsvidhya.com/blog/2020/01/first-text-classification-in-pytorch/
5. https://electronicsmarket.org/best-budget-car-stereos/

6. https://www.youtube.com/watch?v=U0s0f995w14&t=2522s
7. https://www.youtube.com/watch?v=M6adRGJe5cQ&t=1781s