**ỦY BAN NHÂN DÂN THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC SÀI GÒN**

**KHOA CÔNG NGHỆ THÔNG TIN**

…🙢🏶🙠…



**TIỂU LUẬN  
MÔN: SEMINAR CHUYÊN ĐỀ**

**CHUYÊN ĐỀ:**

**Mạng Neuron**

|  |  |
| --- | --- |
| GIẢNG VIÊN HƯỚNG DẪN | SINH VIÊN THỰC HIỆN |
| **PGS.TS. NGUYỄN TUẤN ĐĂNG** | **KOONG CHẤN PHONG** |
|  | **MSSV:** 3121560068 |
|  | **LỚP:** DCT1211  **NĂM HỌC:** 2024 - 2025 |
|  | **TÊN ĐỀ TÀI:** PHÂN TÍCH CẢM XÚC TỪ VĂN BẢN NGẮN SỬ DỤNG RNN |

Tp.Hồ Chí Minh, tháng 05 năm 2025

**MỤC LỤC**

[**MỤC LỤC** 2](#_Toc197285657)

[DANH MỤC BẢNG 4](#_Toc197285658)

[DANH MỤC HÌNH ẢNH 5](#_Toc197285659)

[DANH MỤC TỪ VIẾT TẮT 6](#_Toc197285660)

[LỜI CẢM ƠN 1](#_Toc197285661)

[CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI 2](#_Toc197285662)

[**1.1.** **Đặt vấn đề** 2](#_Toc197285663)

[**1.2.** **Mục tiêu đề tài** 2](#_Toc197285664)

[**1.3.** **Phạm vi đề tài** 2](#_Toc197285665)

[**1.4.** **Nội dung nghiên cứu** 3](#_Toc197285666)

[CHƯƠNG 2. CƠ SỞ LÝ THUYẾT 4](#_Toc197285667)

[**2.1.** **Word Embeddings** 4](#_Toc197285668)

[**2.1.1.** **Định nghĩa và nguyên lý** 4](#_Toc197285669)

[**2.1.2.** **Pre trained Word Embeddings (Glove)** 4](#_Toc197285670)

[**2.1.3.** **Scratch Embeddings** 4](#_Toc197285671)

[**2.1.4.** **Các mô hình mạng Neuron** 5](#_Toc197285672)

[**2.2.** **RNN (Recurrent Neural Network)** 5](#_Toc197285673)

[**2.2.1.** **Định nghĩa và nguyên lý** 5](#_Toc197285674)

[**2.2.2.** **Ưu và nhược điểm của RNN** 6](#_Toc197285675)

[**2.2.3.** **Ứng dụng của RNN trong phân tích cảm xúc.** 6](#_Toc197285676)

[**2.3.** **Một số khái niệm chung** 7](#_Toc197285677)

[**2.3.1.** **Phân tích cảm xúc.** 7](#_Toc197285678)

[**2.3.2.** **Các phương pháp phân tích cảm xúc** 7](#_Toc197285679)

[**2.3.3.** **Các độ đo đánh giá phổ biến** 7](#_Toc197285680)

[CHƯƠNG 3: SƠ ĐỒ KIẾN TRÚC VÀ THUYẾT MINH 9](#_Toc197285681)

[**3.1.** **Sơ đồ kiến trúc** 9](#_Toc197285682)

[**3.2.** **Data.py** 9](#_Toc197285683)

[**3.3.** **Model.py** 13](#_Toc197285684)

[**3.4.** **Train\_eval.py** 16](#_Toc197285685)

[CHƯƠNG 4: KẾT LUẬN 21](#_Toc197285686)

[**4.1. Kết quả** 21](#_Toc197285687)

[**4.2.** **Hướng phát triển** 22](#_Toc197285688)

[CHƯƠNG 5: PHỤ LỤC 23](#_Toc197285689)

[**5.1. Phụ Lục A: data.py** 23](#_Toc197285690)

[**5.2. Phụ Lục B: model.py** 24](#_Toc197285691)

[**5.3. Phụ Lục C: train\_eval.py** 25](#_Toc197285692)

[TÀI LIỆU THAM KHẢO 26](#_Toc197285693)

# DANH MỤC BẢNG

[Bảng 1: Bảng tổng hợp từ Results.json 21](#_Toc197285408)

# DANH MỤC HÌNH ẢNH

[Hình 1: Mô hình RNN mẫu 5](#_Toc197285419)

[Hình 2: Sơ đồ kiến trúc 9](#_Toc197285420)

[Hình 3: Ảnh code data.py 10](#_Toc197285421)

[Hình 4: Thư viện sử dụng cho data.py 11](#_Toc197285422)

[Hình 5: Đọc và chuẩn bị dữ liệu từ file csv 11](#_Toc197285423)

[Hình 6: Tokenizes văn bản 12](#_Toc197285424)

[Hình 7: Xây dựng từ điển 12](#_Toc197285425)

[Hình 8: Hàm to\_indices xây dựng index trong từ điển 12](#_Toc197285426)

[Hình 9: Xây dựng lớp Dataset 13](#_Toc197285427)

[Hình 10: Phân chia tập dữ liệu train và test 13](#_Toc197285428)

[Hình 11: Models.py 14](#_Toc197285429)

[Hình 12: Khởi tạo lớp RNN Models 14](#_Toc197285430)

[Hình 13: Xây dựng Embedding 15](#_Toc197285431)

[Hình 14: Khởi tạo RNN Layer 15](#_Toc197285432)

[Hình 15: Khởi tạo tầng Dense (Fully Connected) 15](#_Toc197285433)

[Hình 16: Hàm Forward Pass 16](#_Toc197285434)

[Hình 17: Khởi tạo mô hình 16](#_Toc197285435)

[Hình 18: Train\_eval.py 17](#_Toc197285436)

[Hình 19: Hàm Train And Evaluate 18](#_Toc197285437)

[Hình 20: Khởi tạo Hàm mất mmats và optimizer cập nhật trọng số 18](#_Toc197285438)

[Hình 21: Vòng lặp huấn luyện 18](#_Toc197285439)

[Hình 22: Evaluate Process 19](#_Toc197285440)

[Hình 23: Results.json 21](#_Toc197285441)

[Hình 24: Phụ lục A data.py 23](#_Toc197285442)

[Hình 25: Phụ lục B models.py 24](#_Toc197285443)

[Hình 26: Phụ lục C train\_eval.py 25](#_Toc197285444)

# DANH MỤC TỪ VIẾT TẮT

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Từ viết tắt** | **Tên Tiếng Anh** | **Tên Tiếng Việt** |
| 1 | SGD | Stochastic Gradient Descent | Giảm dần độ dốc ngẫu nhiên |
| 2 | LSTM | Long Short Term Memory | Bộ nhớ dài ngắn hạn |
| 3 | RNN | Recurrent Neural Network | Mạng nơ-ron hồi quy |

# LỜI CẢM ƠN

Lời đầu tiên, em xin chân thành cảm ơn các thầy cô trong khoa Công nghệ thông tin của trường Đại học Sài Gòn, những người đã trực tiếp giảng dạy và cung cấp kiến thức, phương pháp trong suốt 3 năm qua. Những nền tảng cơ bản và hành trang quý giá này đã giúp em tự tin bước vào con đường sự nghiệp vững vàng hơn trong tương lai.

Để có được thành tích như ngày hôm nay, chúng em xin đặc biệt gửi lời cảm ơn chân thành nhất tới thầy PGS.TS. Nguyễn Tuấn Đăng, người đã quan tâm giúp đỡ, hỗ trợ và đưa ra kế hoạch hướng dẫn em hoàn thành tốt nhất bài tiểu luận môn Seminar trong thời gian qua. Những kiến thức quý báu mà thầy đã truyền đạt không chỉ giúp em hoàn thành tốt luận văn mà còn là hành trang quý giá cho sự nghiệp sau này của em.

Trong quá trình hoàn thành bài tiểu luận, vì chưa có kinh nghiệm thực tế và chỉ dựa vào lý thuyết đã học, cùng với thời gian có hạn, nên bài tiểu luận sẽ không tránh khỏi những sai sót. Kính mong nhận được sự góp ý, nhận xét từ các thầy để kiến thức của em ngày càng hoàn thiện hơn và rút ra được nhiều kinh nghiệm bổ ích có thể áp dụng vào thực tiễn một cách hiệu quả trong tương lai.

Em xin chân thành cảm ơn!

# CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI

* 1. **Đặt vấn đề**

Trong thời đại kỷ nguyên số hiện nay, lượng dữ liệu văn bản trên các nền tảng mạng xã hội, ứng dụng tin nhắn và các dịch vụ trực tuyến ngày càng gia tăng. Người dùng thường xuyên chia sẻ ý kiến, cảm nhận về sản phẩm, dịch vụ hoặc trải nghiệm công việc và học tập qua các đoạn văn bản ngắn. Việc phân tích cảm xúc từ những văn bản này không chỉ giúp hiểu được tâm lý người dùng mà còn có giá trị ứng dụng cao trong nhiều lĩnh vực như chăm sóc khách hàng, marketing, phân tích thương hiệu hay cải thiện môi trường làm việc.

Tuy nhiên, các phương pháp truyền thống vẫn thường gặp hạn chế khi xử lý ngữ cảnh, trình tự từ ngữ và các đặc trưng ngôn ngữ phức tạp. Điều này dẫn đến nhu cầu sử dụng các mô hình học sâu hiện đại có khả năng hiểu và xử lý chuỗi thời gian tốt hơn.

Để giải quyết những thách thức này, thông qua việc kết hợp RNN với các kỹ thuật biểu diễn từ vựng hiện đại như Word Embeddings, chúng ta có thể xây dựng các mô hình phân tích cảm xúc với độ chính xác cao hơn, đặc biệt là với các văn bản ngắn - nơi mỗi từ đều mang trọng lượng lớn và không có nhiều ngữ cảnh để phân tích.

* 1. **Mục tiêu đề tài**

Mục tiêu chính của đề tài này là xây dựng và triển khai một mô hình mạng nơ-ron hồi qui (RNN) để thực hiện phân tích và dự đoán nhãn cảm xúc từ văn bản ngắn của người dùng. Cụ thể, đề tài hướng đến các mục tiêu sau:

* Xây dựng mô hình RNN đơn giản nhưng hiệu quả.
* So sánh hiệu suất giữa hai phương pháp biểu diễn từ vựng.
* Tối ưu hóa mô hình bằng thuật toán SGD.
* Minh họa và giải thích kiến trúc mạng nơ-ron.
* Đánh giá hiệu suất mô hình.
  1. **Phạm vi đề tài**

Phạm vi của đề tài tập trung vào việc phân tích cảm xúc từ văn bản ngắn (dưới 50 từ) liên quan đến công việc và học tập. Mô hình được thiết kế để phân loại văn bản thành ba nhãn cảm xúc: Tích cực (Positive), Tiêu cực (Negative) và Trung tính (Neutral), sử dụng tập dữ liệu gồm ít nhất 500 mẫu có nhãn và được triển khai trên nền tảng Python với PyTorch.

* 1. **Nội dung nghiên cứu**

Nội dung nghiên cứu của đề tài này sẽ tập trung vào việc xây dựng và phân tích cảm xúc từ văn bản ngắn (dưới 50 từ) liên quan đến công việc và học tập. Cụ thể, nội dung nghiên cứu sẽ bao gồm các phần sau:

* **Nghiên cứu lý thuyết:** Tìm hiểu về Word Embeddings và mạng nơ-ron hồi qui (RNN) trong phân tích cảm xúc.
* **Thu thập và xử lý dữ liệu:** Thu thập văn bản ngắn, tiền xử lý và chia tập dữ liệu.
* **Thiết kế và triển khai mô hình:** Xây dựng mô hình RNN với tầng Embedding (Pretrained và Scratch), khối RNN và tầng Dense.
* **Huấn luyện và đánh giá:** Huấn luyện mô hình với SGD, đánh giá hiệu suất và so sánh giữa các phương pháp.
* **Minh họa và giải thích:** Xây dựng sơ đồ kiến trúc mạng nơ-ron và phân tích kết quả.

# CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

* 1. **Word Embeddings**
     1. **Định nghĩa và nguyên lý**

Word Embeddings là kỹ thuật biểu diễn từ ngữ dưới dạng các vector số thực trong không gian nhiều chiều. Khác với biểu diễn one-hot vector truyền thống (vector thưa với kích thước bằng kích thước từ điển), word embeddings tạo ra các vector đặc trưng có kích thước nhỏ hơn (thường 50-300 chiều) nhưng chứa thông tin ngữ nghĩa phong phú.

Nguyên lý cơ bản dựa trên giả thuyết phân phối trong ngôn ngữ học: "Những từ xuất hiện trong cùng ngữ cảnh thường có ý nghĩa liên quan đến nhau". Khi được huấn luyện trên tập dữ liệu lớn, các từ có ý nghĩa tương tự sẽ có vector biểu diễn gần nhau trong không gian đa chiều.

* + 1. **Pre trained Word Embeddings (Glove)**

GloVe (Global Vectors for Word Representation) là phương pháp tạo word embeddings dựa trên thống kê đồng xuất hiện toàn cục từ các corpus lớn. Quá trình tạo vector GloVe tối thiểu hóa hàm mất mát để đảm bảo tích vô hướng của các vector từ gần với logarithm của xác suất đồng xuất hiện.

Ưu điểm của pre-trained embeddings như GloVe:

* Đã được huấn luyện trên tập dữ liệu lớn.
* Nắm bắt được nhiều mối quan hệ ngữ nghĩa phức tạp.
* Hiệu quả ngay cả với dữ liệu huấn luyện hạn chế.
  + 1. **Scratch Embeddings**

Scratch Embeddings là phương pháp tạo word embeddings từ đầu, đồng thời với quá trình huấn luyện mô hình chính. Các embeddings được khởi tạo ngẫu nhiên và cập nhật thông qua quá trình backpropagation.

Ưu điểm:

* Được tối ưu hóa đặc biệt cho tác vụ và miền dữ liệu cụ thể.
* Có thể học những đặc trưng riêng biệt của dữ liệu đang xử lý.

Nhược điểm:

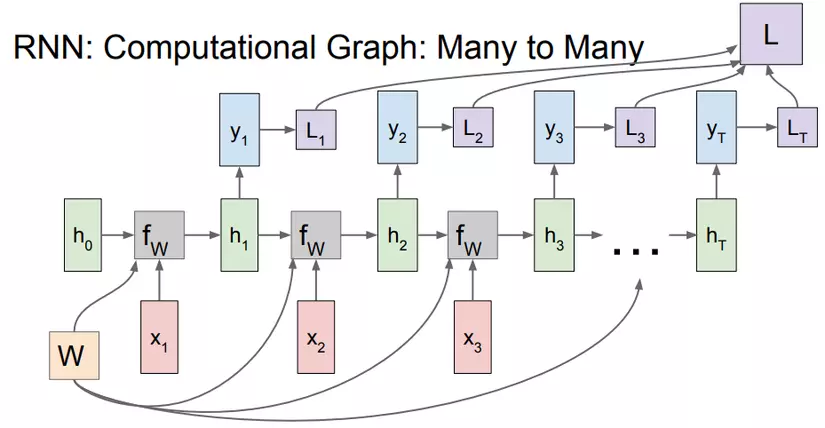
* Yêu cầu lượng dữ liệu huấn luyện lớn để học được các biểu diễn có ý nghĩa.
* Có thể không bắt được các mối quan hệ ngữ nghĩa phức tạp nếu dữ liệu huấn luyện nhỏ.
  + 1. **Các mô hình mạng Neuron**

Trong thế kỷ 21, sự kết hợp giữa phân tích cú pháp truyền thống và học máy đã mang lại những tiến bộ đáng kể. Các mô hình học sâu như mạng nơ-ron hồi quy (RNN) và mạng nơ-ron biến đổi (Transformer) đã được áp dụng vào việc phân tích cú pháp với hiệu suất cao.

* 1. **RNN (Recurrent Neural Network)**
     1. **Định nghĩa và nguyên lý**

Mạng nơ-ron hồi quy (Recurrent Neural Network - RNN) là một kiến trúc mạng nơ-ron được thiết kế đặc biệt để xử lý dữ liệu chuỗi như văn bản, âm thanh hoặc chuỗi thời gian. Khác với mạng nơ-ron truyền thống (feedforward), RNN có khả năng "ghi nhớ" thông tin từ các bước trước đó thông qua cơ chế hồi quy (recurrence).

Nguyên lý cơ bản của RNN là sử dụng cùng một khối tính toán (với cùng một tập tham số) cho mỗi phần tử trong chuỗi, đồng thời duy trì một trạng thái ẩn (hidden state) để lưu trữ thông tin từ các bước trước.



Hình 1: Mô hình RNN mẫu

Công thức tổng quát của RNN:

|  |
| --- |
| ht = tanh(Whx x Xt + Whh x ht-1 + bh)  yt = Why x ht + by |

Trong đó:

* ht: hidden state tại thời điểm t.
* xt: input tại thời điểm t.
* yt: output tại thời điểm t.
* Whx, Whh, Why: Các ma trận trọng số.
* bh, by: các hệ số điều chỉnh (bias).
  + 1. **Ưu và nhược điểm của RNN**

**Ưu điểm:**

* Có khả năng xử lý dữ liệu có độ dài biến đổi.
* Chia sẻ tham số qua các bước thời gian giúp giảm số lượng tham số cần học.
* Có thể nắm bắt được mối quan hệ thời gian và phụ thuộc trong dữ liệu chuỗi.
* Phù hợp với các tác vụ ngôn ngữ như phân loại văn bản, dịch máy, phân tích cảm xúc.

**Nhược điểm:**

* Khó khăn trong việc học các phụ thuộc dài (vanishing gradient problem).
* Quá trình huấn luyện thường chậm do tính tuần tự.
* Tính toán không song song hóa được như các kiến trúc khác.
* Có thể gặp vấn đề "quên" thông tin từ xa trong chuỗi dà.
  + 1. **Ứng dụng của RNN trong phân tích cảm xúc.**

Trong bài toán phân tích cảm xúc, RNN hoạt động theo các bước sau:

* **Biểu diễn từ:** Mỗi từ trong văn bản được chuyển đổi thành vector đặc trưng thông qua embedding layer.
* **Xử lý tuần tự:** RNN xử lý từng từ một theo thứ tự, cập nhật hidden state sau mỗi từ.
* **Tổng hợp thông tin:** Hidden state cuối cùng chứa thông tin tổng hợp từ toàn bộ văn bản.
* **Phân loại:** Hidden state cuối cùng được đưa qua một tầng fully connected để dự đoán nhãn cảm xúc.

RNN đặc biệt hiệu quả trong phân tích cảm xúc vì nó có thể:

* Nắm bắt ngữ cảnh và thứ tự của các từ, điều quan trọng để hiểu cảm xúc.
* Xử lý các cấu trúc ngôn ngữ phức tạp như phủ định, mỉa mai.
* Học được các mẫu ngôn ngữ đặc trưng cho các cảm xúc khác nhau**.**
  1. **Một số khái niệm chung**
     1. **Phân tích cảm xúc.**

Phân tích cảm xúc là một nhánh của xử lý ngôn ngữ tự nhiên (NLP) tập trung vào việc nhận diện, trích xuất và phân loại các ý kiến, cảm xúc và thái độ trong văn bản. Mục tiêu chính là xác định liệu người viết có thái độ tích cực, tiêu cực hay trung lập đối với chủ đề được đề cập.

Ý nghĩa thực tiễn:

* Giúp doanh nghiệp nắm bắt phản hồi của khách hàng về sản phẩm/dịch vụ.
* Hỗ trợ phân tích xu hướng thị trường và ý kiến công chúng.
* Phát hiện các vấn đề tiềm ẩn trong trải nghiệm người dùng.
* Hỗ trợ ra quyết định dựa trên dữ liệu từ mạng xã hội, đánh giá sản phẩm.
  + 1. **Các phương pháp phân tích cảm xúc**
* **Phương pháp dựa trên từ điển (Lexicon-based):** Sử dụng các từ điển cảm xúc đã được gán nhãn sẵn.
* **Phương pháp học máy truyền thống:** Sử dụng các mô hình như SVM, Naive Bayes, Decision Trees.
* **Phương pháp học sâu (Deep Learning):** Sử dụng RNN, LSTM, GRU, Transformer.
  + 1. **Các độ đo đánh giá phổ biến**

**Accuracy (Độ chính xác):**

* Tỷ lệ dự đoán đúng trên tổng số mẫu.
* Công thức: (TP + TN) / (TP + TN + FP + FN).
* Phù hợp khi dữ liệu cân bằng giữa các lớp.

**F1-Score:**

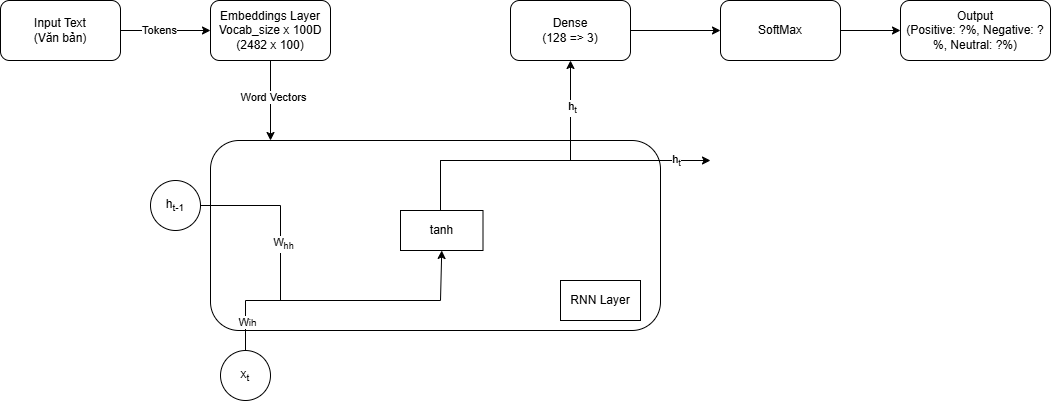
* Trung bình điều hòa của precision và recall.
* Công thức: 2 \* (precision \* recall) / (precision + recall).
* Phù hợp với dữ liệu mất cân bằng.

**Macro F1-Score:**

* Trung bình cộng của F1-score cho mỗi lớp.
* Đánh giá khách quan hơn với dữ liệu mất cân bằng.
* Phản ánh hiệu suất trên tất cả các lớp, kể cả lớp thiểu số.

# CHƯƠNG 3: SƠ ĐỒ KIẾN TRÚC VÀ THUYẾT MINH

* 1. **Sơ đồ kiến trúc**



Hình 2: Sơ đồ kiến trúc

**Luồng hoạt động của sơ đồ:**

* **Văn bản → Tokens:** Văn bản đầu vào được tách thành các token riêng biệt.
* **Tokens → Word Vectors:** Mỗi token được chuyển thành vector 100 chiều qua tầng Embedding.
* **Word Vectors → RNN Processing:** Các vector từ được xử lý theo thứ tự thời gian để tạo ht (hidden state) tại mỗi bước.
* **ht (Hidden State) → Dense Layer:** Hidden state cuối cùng được đưa qua một lớp để thực hiện fully connected chuyển đổi từ 128 units thành 3 units (tương ứng với 3 nhãn cảm xúc là Positive, Negative, Neutral).
* **Logits → SoftMax → Output:** Áp dụng hàm kích hoạt SoftMax để chuyển đổi logits thành xác suất cho ra tỷ lệ phần trăm cho mỗi nhãn cảm xúc.

**Ký hiệu:**

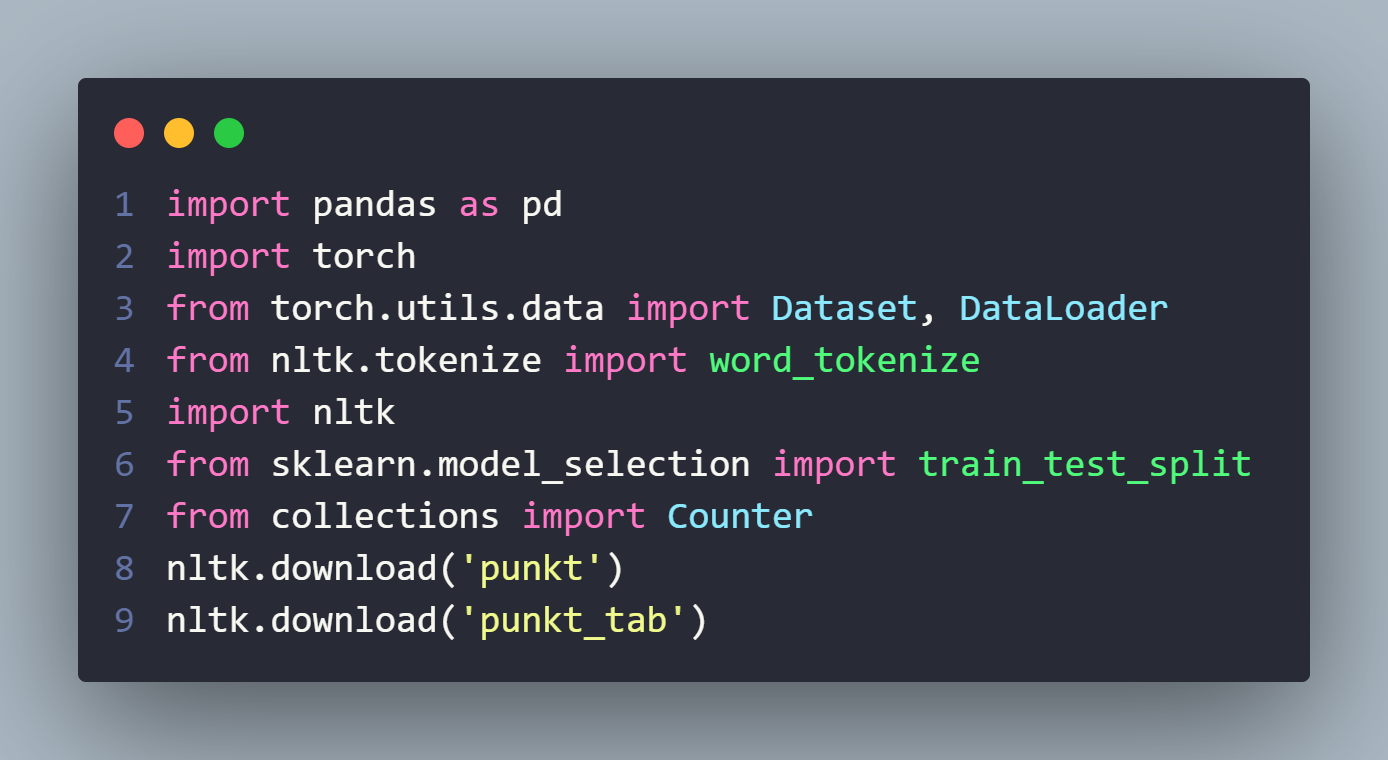
* **ht-1:** Hidden state tại bước thời gian t-1, lưu trữ thông tin ngữ cảnh từ các từ trước đó.
* **ht:** Hidden state tại bước thời gian t, kết quả của việc xử lý từ hiện tại và ngữ cảnh trước đó.
* **xt:** Vector word tại vị trí t trong chuỗi đầu vào.
* **Whh:** Ma trận trọng số cho hidden-to-hidden connections, Weight từ trạng thái trước đó (ht-1) đến trạng thái hiện tại (ht).
* **Wih:** Ma trận trọng số cho input-to-hidden connections, Weight từ đầu vào hiện tại (xt) đến trạng thái hiện tại (ht).
  1. **Data.py**

****

Hình 3: Ảnh code data.py

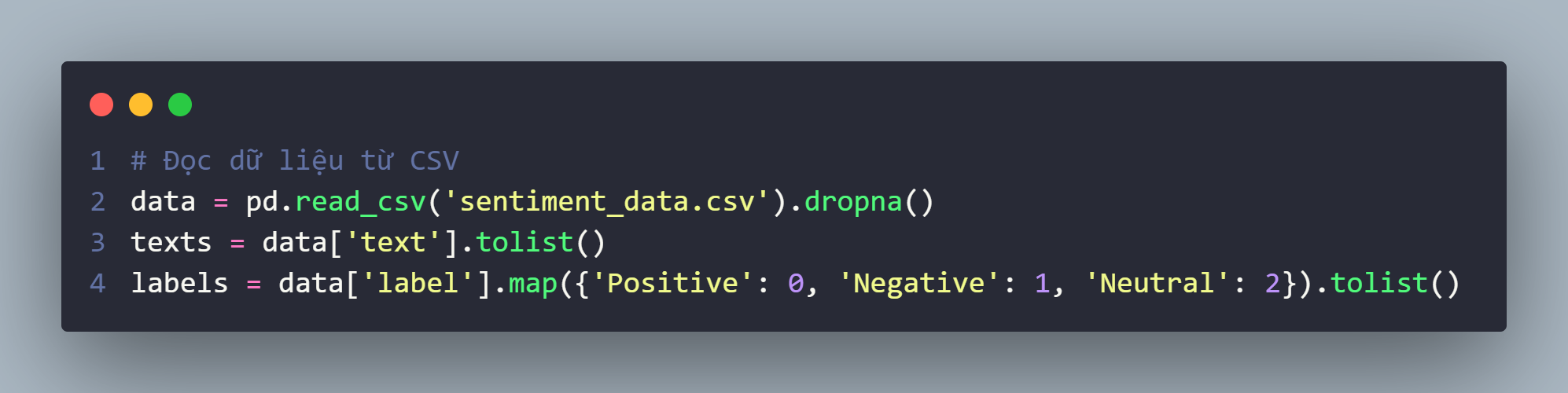
File data.py đảm nhận vai trò quan trọng trong việc xử lý dữ liệu đầu vào cho mô hình phân tích cảm xúc.

* **Thư viện sử dụng:**



Hình 4: Thư viện sử dụng cho data.py

* + Pandas: Đọc và xử lý dữ liệu dạng bảng từ file csv.
  + Torch: Framework deep learning chính cho mô hình.
  + DataSet và DataLoader: Các lớp hỗ trợ quản lý và xử lý dữ liệu trong PyTorch.
  + Word\_tokenize: Hàm từ NLTK để tách văn bản thành các token (từ).
* **Đọc và chuẩn bị dữ liệu:**



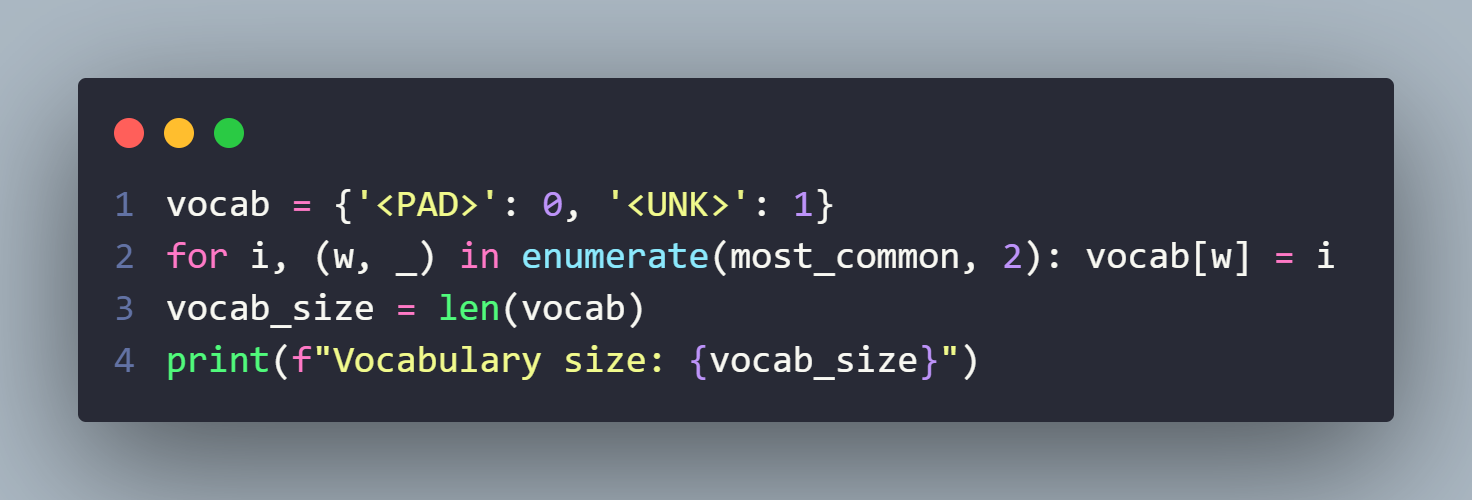
Hình 5: Đọc và chuẩn bị dữ liệu từ file csv

* + Đọc dữ liệu từ file sentiment\_data.csv và loại bỏ các dòng có giá trị null.
  + Lấy danh sách các văn bản từ cột “text”.
  + Chuyển đổi nhãn từ dạng chuỗi (“Positive”, “Negative”, “Neutral”) sang dạng số (0, 1, 2).
* **Tokenize văn bản và xây dựng từ điển:**



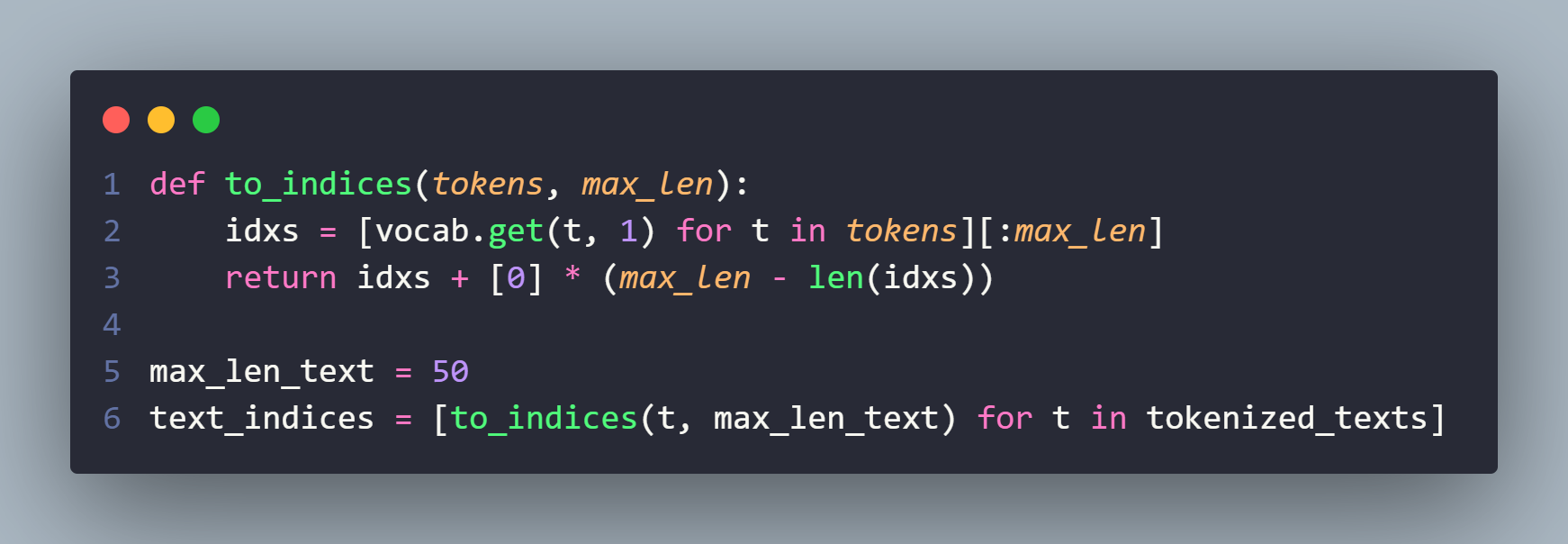
Hình 6: Tokenizes văn bản

* + Tokenize các văn bản bằng cách chuyển đổi mỗi câu thành danh sách các từ (token).
  + Chuyển tất cả văn bản thành chữ thường để chuẩn hóa.
  + Tạo danh sách tất cả các từ xuất hiện trong dữ liệu.



Hình 7: Xây dựng từ điển

* + Tạo từ điển Vocabulary ánh xạ từ mỗi từ sang một chỉ số:
    - <PAD> (Index 0): Token đặc biệt dùng để đệm câu ngắn.
    - <UNK> (Index 1): Token đặc biệt cho các từ không có trong từ điển.
* **Chuyển đổi văn bản thành chuỗi số**



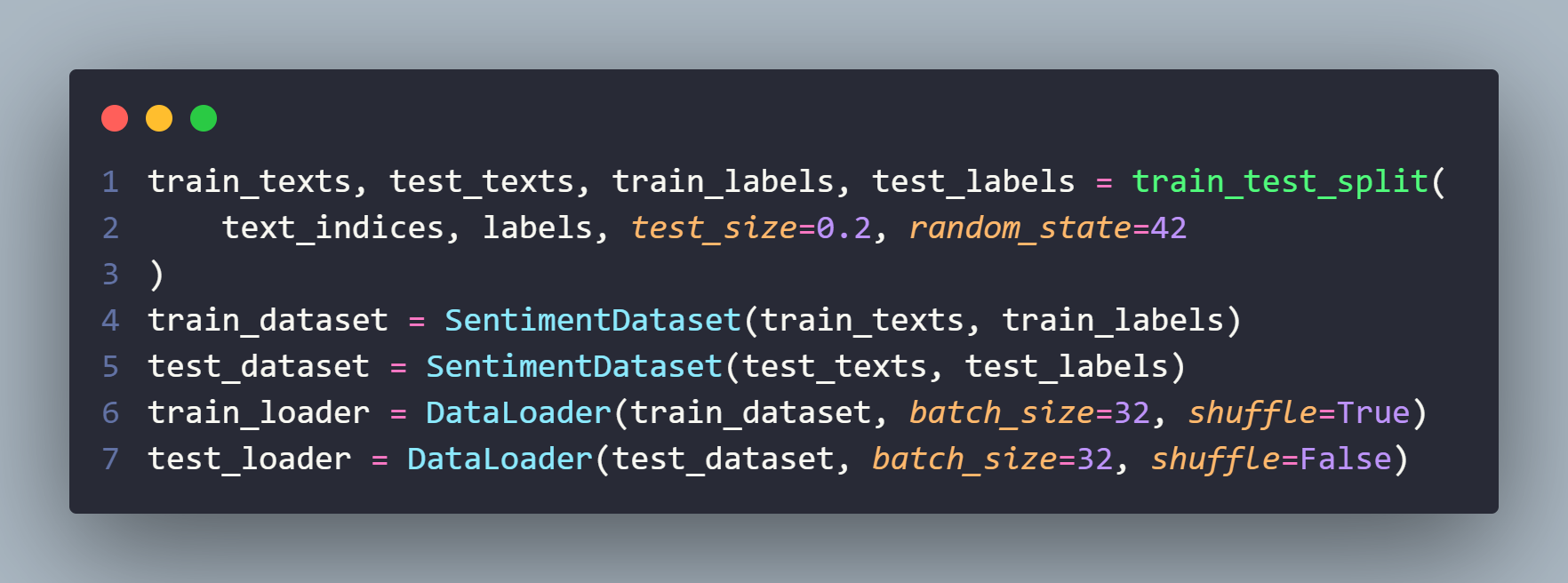
Hình 8: Hàm to\_indices xây dựng index trong từ điển

* + Hàm to\_indices() thực hiện nhiệm vụ chuyển đổi danh sách token thành danh sách số nguyên (Index trong từ điển).
  + Nếu một từ không có trong từ điển sẽ gán cho Index 1 (token <UNK>).
  + Nếu văn bản có độ dài lớn hơn max\_lex là 50 thì sẽ thực hiện cắt bớt câu và văn bản sau khi cắt bớt hay văn bản bé hơn max\_len sẽ được đệm thêm phần dư là 0 cho đến khi đạt đủ max\_len.
* **Tạo lớp Dataset**



Hình 9: Xây dựng lớp Dataset

* + Định nghĩa một lớp SentimentDataset kế thừa từ torch.utils.data.Dataset để tổ chức dữ liệu theo cách mà PyTorch có thể sử dụng hiệu quả.
* **Phân chia và chuẩn bị data cho việc huấn luyện**



Hình 10: Phân chia tập dữ liệu train và test

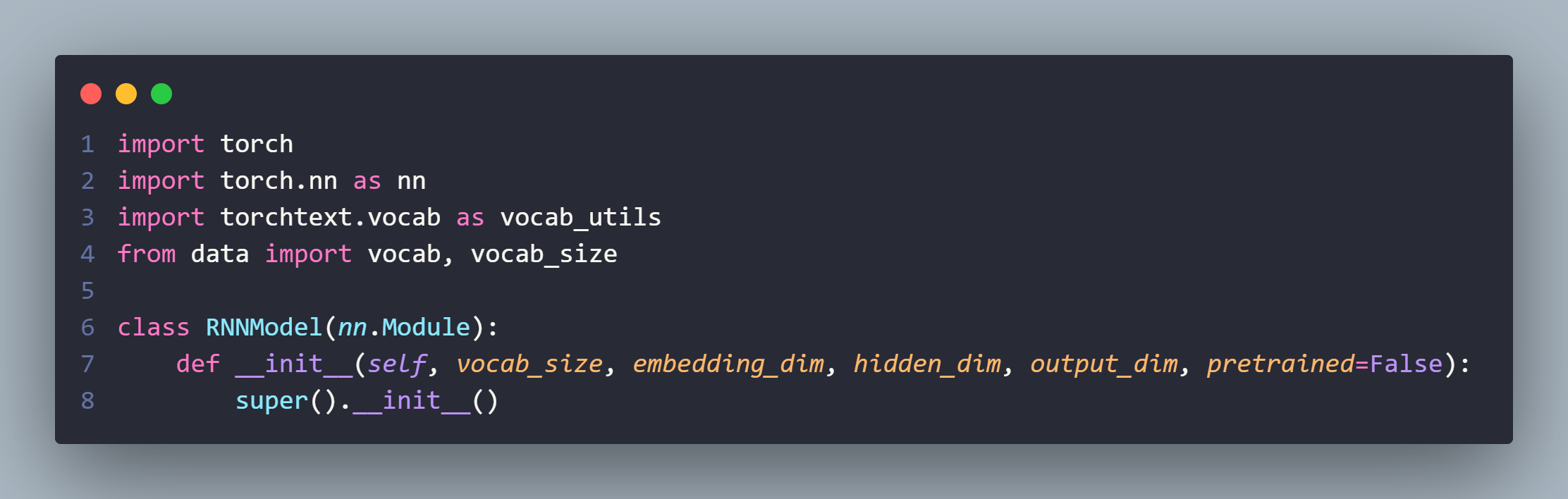
* + Phân chia dữ liệu thành tập huấn luyện (80%) và tập kiểm tra (20%).
  + Random\_state = 42 đảm bảo tính tái lập, mỗi lần chạy đều cho kết quả phân chia giống nhau.
  + Tạo các object SentimentDataset cho tập huấn luyện và tập kiểm tra.
  + Tạo các DataLoader để cung cấp dữ liệu theo Batch, với batch\_size = 32 mỗi lần xử lý 32 mẫu và shuffle = True sẽ xáo trộn tập data train và không xáo trộn cho tập data test.
  1. **Model.py**

File models.py định nghĩa kiến trúc mạng neuron hồi quy (RNN) được sử dụng cho bài toán phân tích cảm xúc.



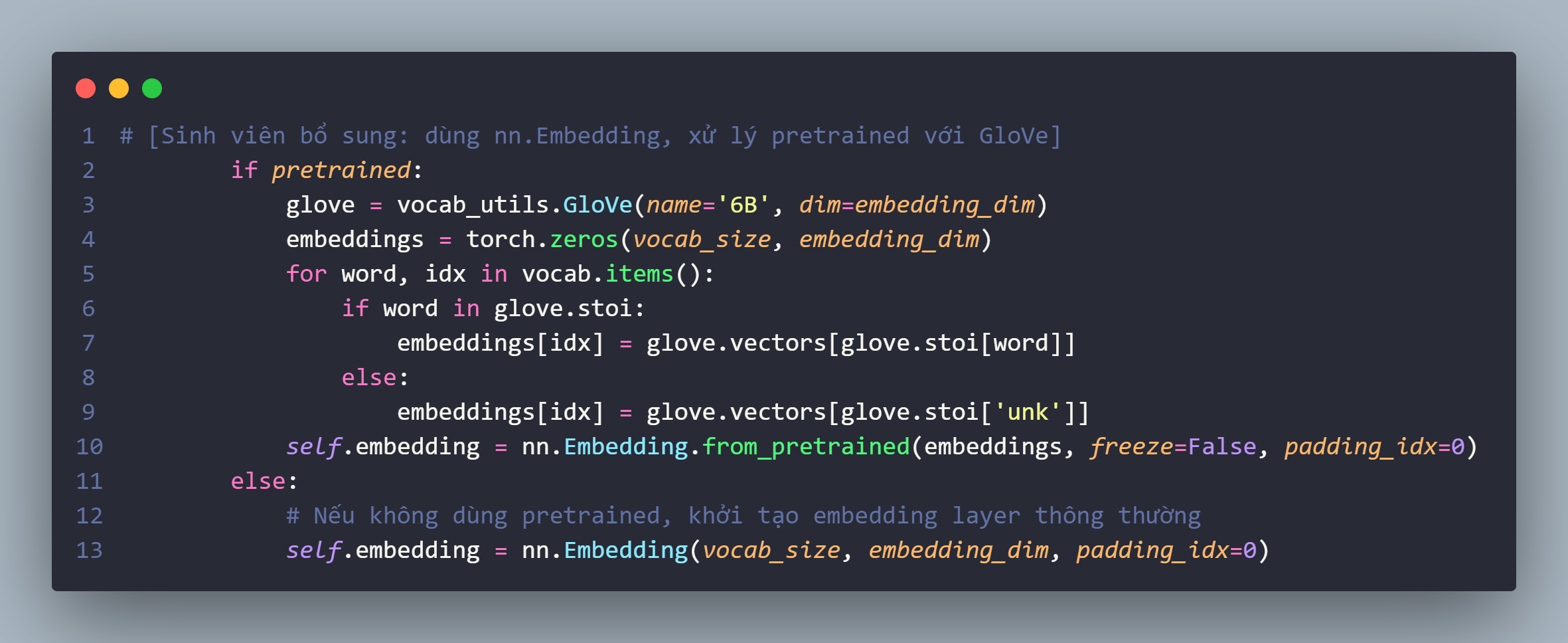
Hình 11: Models.py

* **Định nghĩa lớp RNN Model**



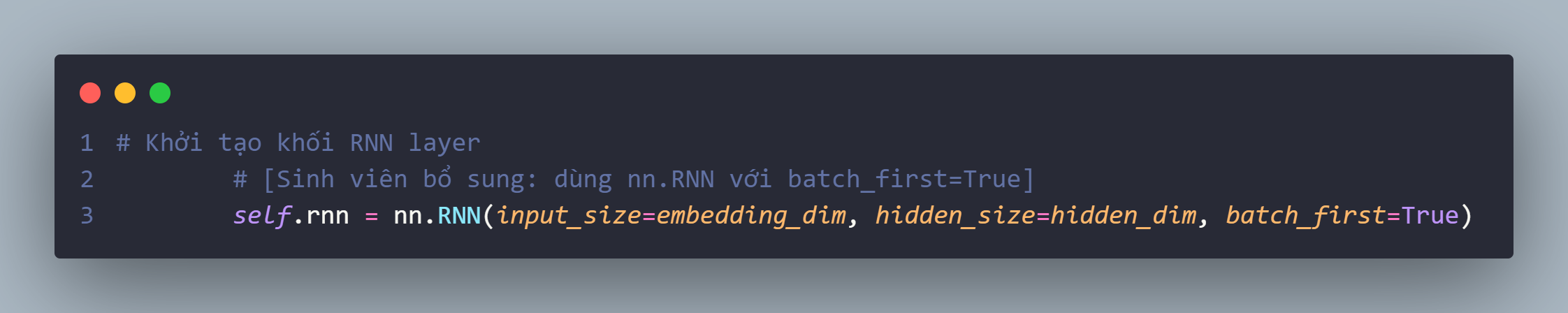
Hình 12: Khởi tạo lớp RNN Models

* + **Hàm khởi tạo xử lý embedding:**



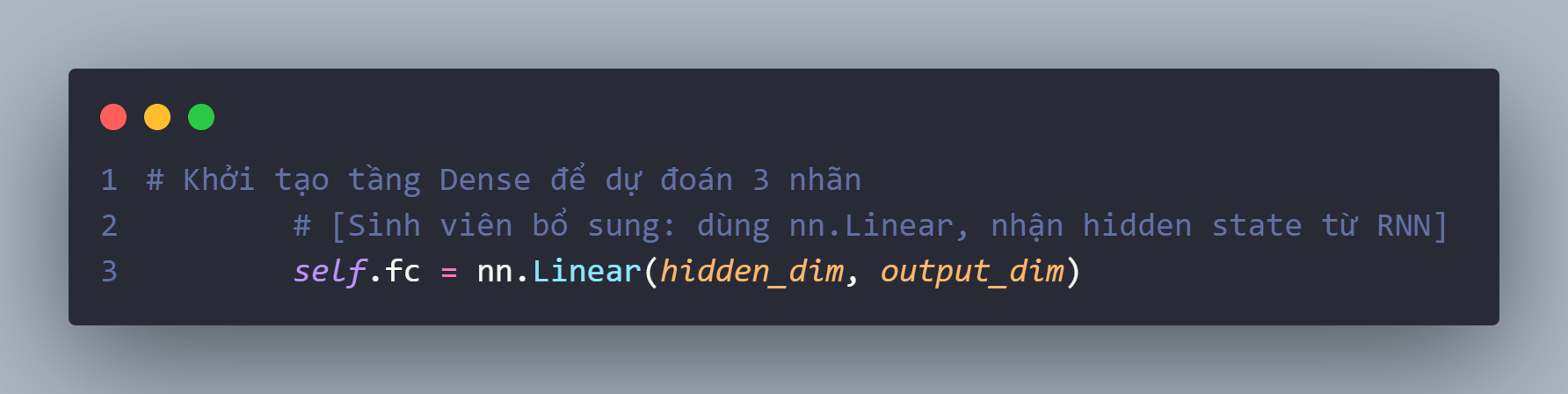
Hình 13: Xây dựng Embedding

* + - Đoạn code này cấu hình Embedding theo 2 cách khi pretrained = True và False, khi True sẽ xài mô hình đã train sẵn là Glove. Với mỗi từ tồn tại trong Glove sẽ lấy vector tương ứng còn không sử dụng vector của token <UNK>. Khi Pretrained = False thì tạo embedding với trọng số ngẫu nhiên, các trọng số sẽ được học từ đầu (scratch) trong quá trình huấn luyện.
  + **Khởi tạo RNN Layer:**



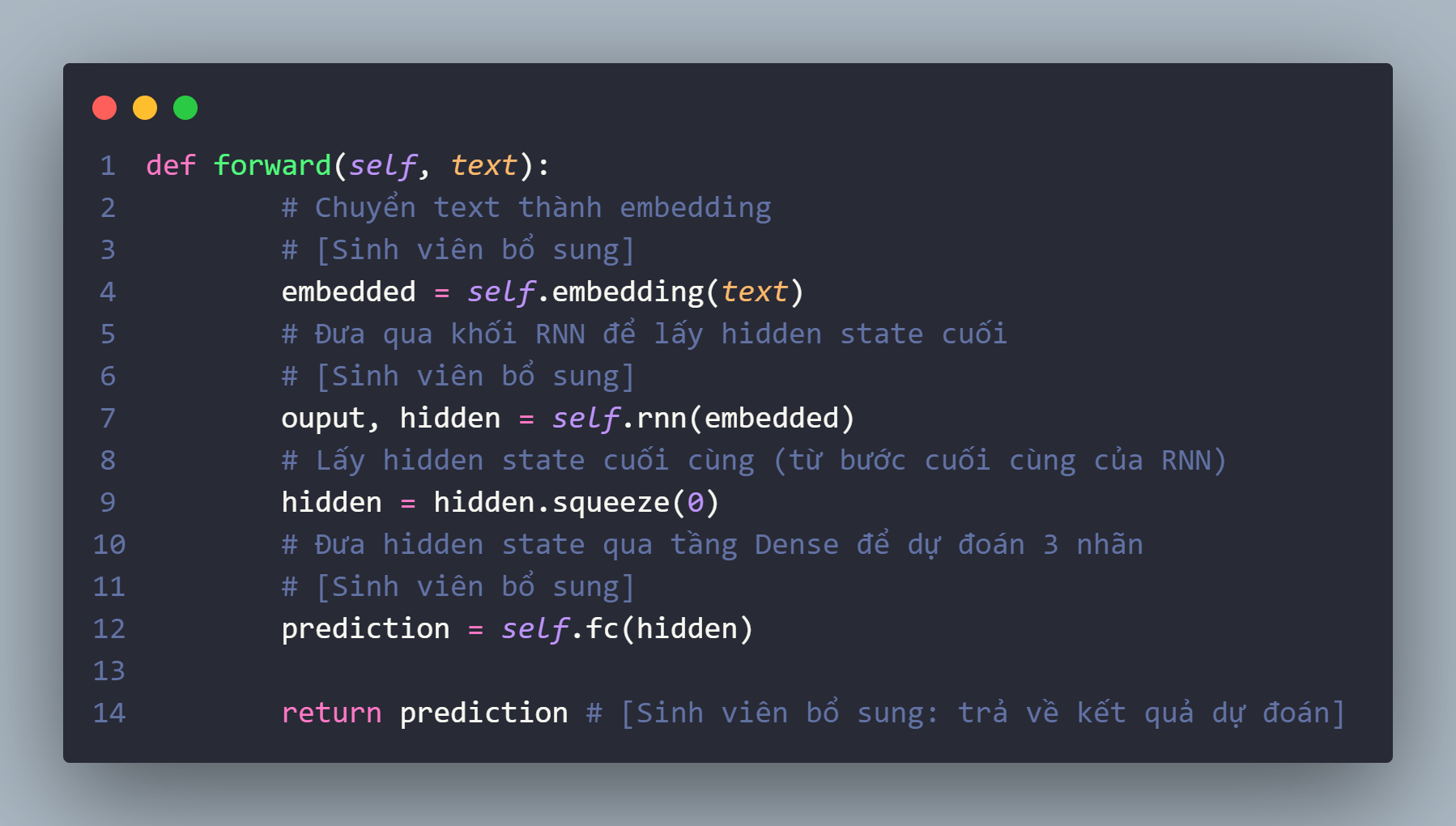
Hình 14: Khởi tạo RNN Layer

* + - Tạo một khối RNN cơ bản với input đầu vào là input\_size = embedding\_dim là 100. Hidden\_size = hidden\_dim nghĩa là kích thước hidden state là 128 và với batch\_first = True.
  + **Khởi tạo tầng Dense (Fully Connected):**



Hình 15: Khởi tạo tầng Dense (Fully Connected)

* + - Tạo một tầng Fully Connected với hidden state là 128 và đầu ra là output\_dim là 3 tương ứng với 3 nhãn cảm xúc.
    - Tầng này chuyển đổi hidden state từ RNN thành các logits cho mỗi nhãn cảm xúc.
* **Hàm Forward (Quá trình Forward trong mô hình)**



Hình 16: Hàm Forward Pass

* + Embedding: Chuyển đổi các chỉ số word thành vector embedding.
  + RNN Processing: Đưa embedded vectors qua RNN để xử lý theo thứ tự.
  + Lấy hidden state sau cùng.
  + Classification: Đưa hidden state cuối cùng qua tầng Fully Connected để dự đoán.
* **Khởi tạo mô hình**

****

Hình 17: Khởi tạo mô hình

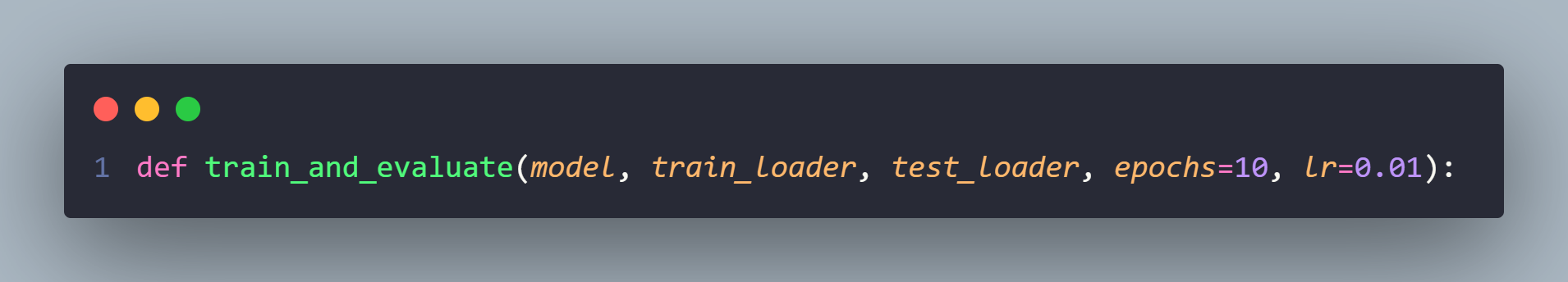
* + Tạo một instance của mô hình RNN với các tham số:
    - Vocab\_size: kích thước từ điển trong file data.py.
    - Embedding\_dim=100: Mỗi từ được biểu diễn bằng vector 100 chiều.
    - Hidden\_dim=128: Hidden state của RNN có 128 units.
    - Output\_dim=3: Tương ứng với 3 lớp đầu ra (Positive, Negative, Neutral).
    - Pretrained = True: Sử dụng pretrained Glove Embeddings.
  1. **Train\_eval.py**

File train\_eval.py có vai trò quan trọng trong việc thực hiện huấn luyện, đánh giá mô hình và so sánh hiệu suất giữa 2 phương pháp embedding khác nhau là Pretrained với Scratch.



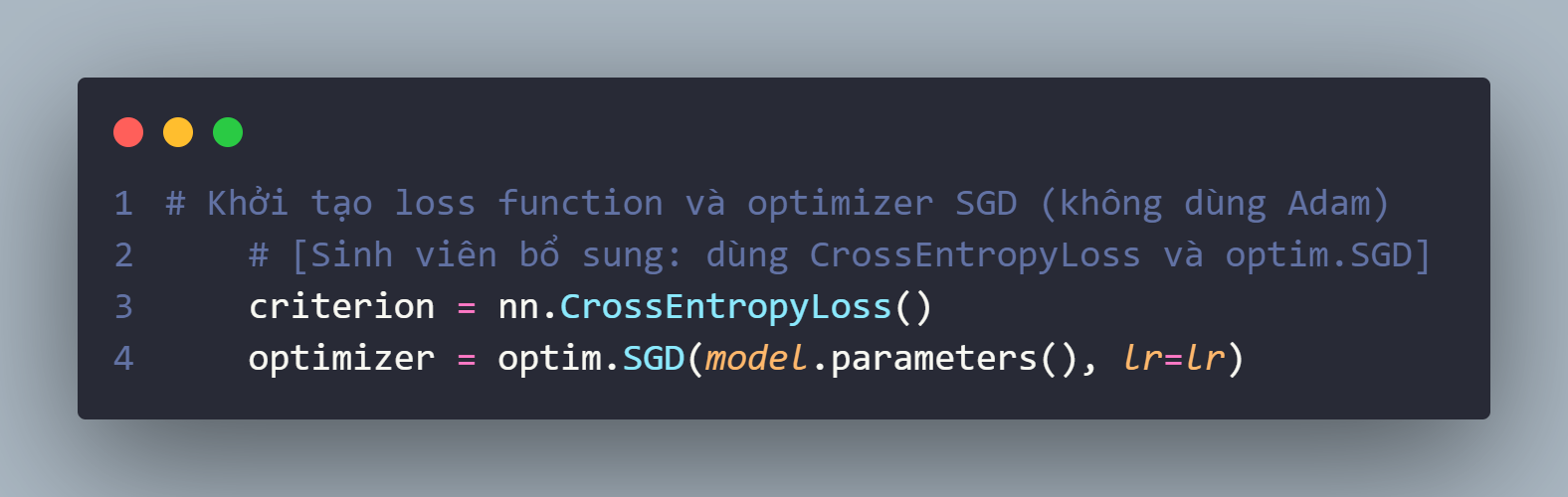
Hình 18: Train\_eval.py

* **Hàm train\_and\_evaluate**



Hình 19: Hàm Train And Evaluate

* + **Hàm này thực hiện toàn bộ quá trình train và đánh giá mô hình.**
  + **Khởi tạo Loss Function và Optimizer SGD**



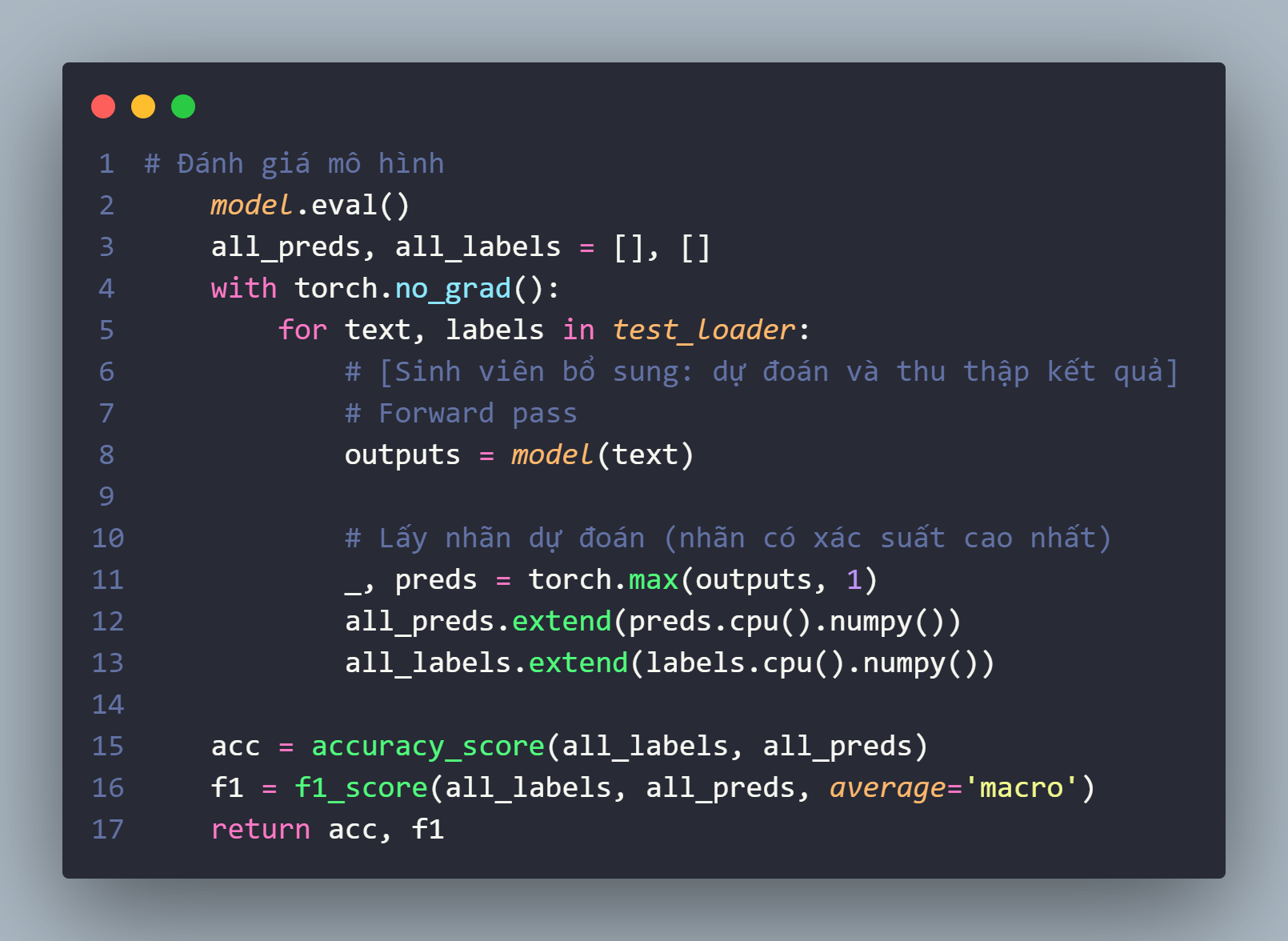
Hình 20: Khởi tạo Hàm mất mmats và optimizer cập nhật trọng số

* + - Sử dụng CrossEntropyLoss vì đây là cách tính hàm mất mát phổ biến cho các bài toán phân loại nhiều lớp.
    - SGD một thuật toán tối ưu hóa gradient trong việc cập nhật trọng số. Thuật toán này cập nhật trọng số qua từng mẫu.
  + **Vòng lặp huấn luyện**



Hình 21: Vòng lặp huấn luyện

* + - Model.train(): Bật chế độ train cho mô hình các cơ chế như dropout sẽ được bật.
    - Vòng lặp đầu tiên sẽ lặp qua lại nhiều lần trên toàn bộ tập dữ liệu, mỗi lần lặp như vậy gọi là một Epoch.
    - Vòng lặp 2 với mỗi Epoch ta sẽ không xử lý toàn bộ dữ liệu cùng lúc mà sẽ chia thành các batch, train\_loader trong data.py sẽ cung cấp các batch này.
    - Quy trình học cho mỗi batch:
      * Optimizer.zero\_grad(): Trước khi tính Loss mới ta sẽ xóa Loss cũ.
      * predictions = model(text): Thực hiện Forward pass cho mô hình và trả về kết quả dự đoán.
      * Loss = criterion(predictions, labels): Tính lỗi có nghĩa là so sánh kết quả thực tế với kết quả dự kiến.
      * Loss.backward(): Tính gradients các trọng số cần cập nhật.
      * Optimizer.step(): Thực hiện cập nhật trọng số.
      * Sau đó là cập nhật lại loss tổng.
  + **Đánh giá mô hình:**



Hình 22: Evaluate Process

* + - Model.eval() chuyển sang chế độ đánh giá, báo cho mô hình biết bây giờ chỉ test tắt các tính năng chỉ dùng khi học (train) như Dropout.
    - With torch.no\_grad(): Tắt tính toán gradients vì bây giờ chỉ dự đoán không cần cập nhật mô hình nữa.
    - Sau đó ta sẽ duyệt qua từng batch dữ liệu test lấy từ test\_loader trong data.py.
    - Ouputs = model(text) lấy kết quả dự đoán từ mô hình với 3 nhãn cảm xúc sẽ có xác suất khác nhau.
    - \_, preds = torch.max(outputs, 1) có nghĩa là chọn nhãn có xác suất cao nhất làm dự đoán cuối cùng.
    - all\_preds.extend(...), all\_labels.extend(...): Lưu lại tất cả các dự đoán của mô hình (all\_preds) và các nhãn đúng thực tế (all\_labels) từ toàn bộ dữ liệu test.
    - acc = accuracy\_score(...), f1 = f1\_score(...): Dùng các nhãn đã thu thập để tính toán:
      * accuracy (acc): Tỷ lệ dự đoán đúng trên tổng số dự đoán.
      * f1\_score (f1): Một độ đo khác (trung bình hài hòa của precision và recall), thường tốt hơn accuracy khi dữ liệu các lớp không cân bằng.

# CHƯƠNG 4: KẾT LUẬN

**4.1. Kết quả**

Đây là kết quả sau khi train huấn luyện chương trình với tệp data gồm 8144 rows dữ liệu.

**File Results.json:**



Hình 23: Results.json

**Bảng tổng hợp từ Results.json**

|  |  |  |  |
| --- | --- | --- | --- |
| Thử nghiệm | Accuracy | F1-Score | Ghi chú |
| Pretrained | 32.72% | 0.1644 | Sử dụng Glove Embeddings |
| Scratch | 32.60% | 0.1639 | Sử dụng Random Embeddings |

Bảng 1: Bảng tổng hợp từ Results.json

**Nhận xét:** Về hiệu suất cho thấy rằng mặc dù pretrained sử dụng Gloves cho kết quả tốt hơn Scratch nhưng thực sự không kém nhiều. Nguyên nhân có thể là vì Gloves được huấn luyện trên tập dữ liệu là tiếng anh mà data test lại là tiếng việt nên vẫn chưa thực sự hiệu quả.

* 1. **Hướng phát triển**
* Sử dụng pretrained embeddings dành riêng cho tiếng việt như PhoBERT hoặc Vietnamese fastText thay vì Gloves.
* Sử dụng tokenize phiên bản hỗ trợ tiếng việt sẽ phân loại từ tốt hơn.
* Thử nghiệm với các kiến trúc mạnh hơn như LSTM, Transformer để cho ra kết quả tốt hơn.
* Áp dụng các kỹ thuật xử lý mất cân bằng dữ liệu để cải thiện F1-Score.
* Tăng số chiều Embedding.

# CHƯƠNG 5: PHỤ LỤC

## **5.1. Phụ Lục A: data.py**



Hình 24: Phụ lục A data.py

## **5.2. Phụ Lục B: model.py**



Hình 25: Phụ lục B models.py

## **5.3. Phụ Lục C: train\_eval.py**



Hình 26: Phụ lục C train\_eval.py

# TÀI LIỆU THAM KHẢO

1. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. Advances in Neural Information Processing Systems, 26, 3111-3119.
2. Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global vectors for word representation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 1532-1543. <https://doi.org/10.3115/v1/D14-1162>.
3. Prachi Gopalani, A. (2022, January 10). LSTM vs RNN Confusion Cleared. PlainEnglish.io. <https://ai.plainenglish.io/lstm-vs-rnn-confusion-cleared-772d43b4206f>.
4. Shipra Saxena. (2025, May 01). Introduction to softmax function for neural network. Analytics Vidhya. <https://www.analyticsvidhya.com/blog/2021/04/introduction-to-softmax-for-neural-network/>
5. Leon, T. (2022). Synthetic Vietnamese Students Feedback Corpus [Data set]. Kaggle. <https://www.kaggle.com/datasets/toreleon/synthetic-vietnamese-students-feedback-corpus/data>.