TRƯỜNG ĐẠI HỌC SÀI GÒN

KHOA CÔNG NGHỆ THÔNG TIN

--------*--------**

BÁO CÁO MÔN

XỬ LÝ NGÔN NGỮ TỰ NHIÊN

TÊN ĐỀ TÀI

CÀI ĐẶT VÀ HUẤN LUYỆN MÔ HÌNH SKIP-GRAM TRÊN NGỮ LIỆU TIẾNG VIỆT

Giảng viên hướng dẫn: **PGS. TS. Nguyễn Tuấn Đăng**

Sinh viên thực hiện: **Nguyễn Thiên Thuận**

Thành phố Hồ Chí Minh, năm 2024

Trường Đại học Sài Gòn

Khoa Công Nghệ Thông Tin

Cộng Hoà Xã Hội Chủ Nghĩa Việt Nam

Độc lập – Tự do – Hạnh phúc

Ngày … tháng … năm 2024

BIÊN BẢN ĐÁNH GIÁ ĐỒ ÁN

Học phần: Xử lý ngôn ngữ tự nhiên

Lớp: DCT1223 (chiều thứ 7)

Cá nhân: Nguyễn Thiên Thuận

MSSV: 3122410403

Tên đề tài: Cài đặt và huấn luyện mô hình Skip-gram trên ngữ liệu tiếng Việt

Cán bộ chấm 1		Cán bộ chấm 2		Tổng hợp		Ghi chú
Điểm chữ	Điểm số	Điểm chữ	Điểm số	Điểm chữ	Điểm số	
Ký tên (ghi họ và tên)		Ký tên (ghi họ và tên)				

Mục lục

LỜI MỞ ĐẦU

Dưới tác động của cuộc cách mạng số hóa, xử lý ngôn ngữ tự nhiên (Natural Language Processing - NLP) ngày càng chứng minh được tầm quan trọng trong việc xây dựng các ứng dụng thông minh, như trợ lý ảo, dịch thuật tự động hay phân tích dữ liệu trên mạng xã hội. Một trong những vấn đề cốt lõi của NLP là tìm ra cách biểu diễn từ ngữ để máy tính có thể hiểu và xử lý dễ dàng hơn. Phương pháp word embeddings, với khả năng chuyển đổi từ thành các vector trong không gian số, đã trở thành một công cụ không thể thiếu để giải quyết bài toán này.

Đề tài "*Cài đặt và huấn luyện mô hình Skip-gram trên ngữ liệu tiếng Việt*" hướng đến việc nghiên cứu và triển khai một phương pháp biểu diễn từ hiện đại – Skip-gram – trong ngữ cảnh của tiếng Việt. Với đặc trưng là một ngôn ngữ giàu thanh điệu, đa dạng về hình thái từ và cấu trúc ngữ pháp linh hoạt, tiếng Việt đặt ra nhiều thách thức riêng khi áp dụng các phương pháp xử lý ngôn ngữ tự nhiên. Mục tiêu của đề tài là phát triển một hệ thống học biểu diễn từ tối ưu, hỗ trợ cải thiện hiệu quả của các ứng dụng NLP dành riêng cho tiếng Việt.

Trong quá trình thực hiện, em đã may mắn nhận được sự hướng dẫn tận tâm từ thầy ***Nguyễn Tuấn Đăng***. Thầy không chỉ cung cấp những kiến thức chuyên sâu mà còn truyền động lực và đưa ra những định hướng cụ thể, giúp em vượt qua mọi thử thách trong nghiên cứu. Em xin bày tỏ lòng biết ơn sâu sắc đến thầy vì sự hỗ trợ vô cùng quý giá ấy.

Báo cáo này không chỉ cung cấp cái nhìn chi tiết về phương pháp tiếp cận và quá trình thực hiện đề tài, mà còn đúc kết những bài học kinh nghiệm và những thách thức đã gặp phải trong suốt hành trình nghiên cứu. Em mong rằng tài liệu này sẽ góp phần nhỏ vào sự phát triển của lĩnh vực xử lý ngôn ngữ tiếng Việt, đồng thời khơi

gợi những ý tưởng cho các hướng nghiên cứu mới trong tương lai.Em rất mong nhận được ý kiến đóng góp từ thầy và các bạn để hoàn thiện hơn nữa.

PHẦN 1: TỔNG QUAN

I. Giới thiệu về đề tài

Trong bối cảnh công nghệ thông tin phát triển mạnh mẽ, xử lý ngôn ngữ tự nhiên (Natural Language Processing - NLP) ngày càng khẳng định vai trò quan trọng trong nhiều lĩnh vực như tìm kiếm thông tin, dịch máy tự động, và phát triển trợ lý ảo. Đặc biệt, việc biểu diễn ngữ nghĩa từ ngữ dưới dạng vector trong không gian số đã trở thành nền tảng thiết yếu cho các mô hình học máy hiện đại.

Dự án này tập trung vào nghiên cứu và ứng dụng mô hình ***Skip-gram***, một phương pháp biểu diễn từ hiệu quả và phổ biến. Skip-gram không chỉ hỗ trợ biểu diễn ý nghĩa từ ngữ trong không gian vector mà còn tạo lập mối quan hệ giữa các từ dựa trên ngữ cảnh của chúng. Mô hình này thể hiện rõ khả năng nhóm các từ có ngữ nghĩa tương đồng lại gần nhau trong không gian vector, giúp ích lớn cho các ứng dụng NLP phức tạp như phân tích cảm xúc, phát hiện chủ đề và xây dựng chatbot.

Dự án được triển khai trên cơ sở sử dụng dữ liệu thực tế từ các bài báo tiếng Việt nhằm xây dựng một mô hình ngôn ngữ có thể so sánh mối liên hệ giữa các từ qua ***cosine similarity***. Ngoài ra, dự án cũng đi sâu phân tích các thách thức gặp phải và rút ra bài học kinh nghiệm trong quá trình thực hiện, góp phần vào kho tàng tri thức chung của lĩnh vực NLP.

Với tiềm năng ứng dụng cao và ý nghĩa thực tiễn rõ ràng, dự án không chỉ mang đến hiểu biết sâu sắc về mô hình Skip-gram mà còn mở ra những hướng phát triển mới cho các mô hình học sâu tiên tiến trong tương lai."

II. Các khái niệm

Skip-gram là một mô hình trong xử lý ngôn ngữ tự nhiên (NLP), được giới thiệu trong Word2Vec bởi Tomas Mikolov và cộng sự. Skip-gram tập trung vào việc học

các biểu diễn vector (word embeddings) của từ sao cho các từ có ngữ nghĩa tương tự sẽ gần nhau trong không gian vector.

Nguyên lý hoạt động của Skip-gram là dựa trên mối quan hệ ngữ cảnh: từ một từ trung tâm (center word), mô hình cố gắng dự đoán các từ xung quanh nó (context words). Điều này giúp Skip-gram học được cách các từ liên kết với nhau trong ngữ cảnh cụ thể, từ đó nắm bắt được ý nghĩa của từ. Ví dụ, trong câu "*Mèo đang nằm ngủ trên ghế*," nếu từ trung tâm là "*nằm*", Skip-gram sẽ cố gắng dự đoán các từ như "*Mèo*", "*đang*", "*ngủ*", và "*trên*". Qua quá trình này, mô hình sẽ học cách biểu diễn từ "*nằm*" dưới dạng một vector có mối quan hệ gần gũi với các từ liên quan trong câu.

Điểm mạnh của Skip-gram là khả năng hoạt động hiệu quả trên các tập dữ liệu lớn và biểu diễn tốt mối quan hệ ngữ nghĩa, ngay cả khi các từ trong ngữ cảnh xuất hiện ít trong dữ liệu huấn luyện. Skip-gram đã trở thành một công cụ mạnh mẽ trong NLP, đóng vai trò nền tảng cho nhiều ứng dụng như phân tích ngữ nghĩa, dịch máy và hệ thống hỏi đáp.

Cấu trúc Skip-gram bao gồm hai thành phần chính:

1. **Từ trung tâm và từ ngữ cảnh:** Mô hình sử dụng một từ trung tâm (center word) để dự đoán các từ xung quanh nó (context words). Bộ dữ liệu được chuẩn bị bằng cách tạo ra các cặp từ trung tâm và từ ngữ cảnh trong một khoảng cách ngữ cảnh cố định (window size).
2. **Hàm mục tiêu:** Skip-gram tối ưu hóa một hàm mục tiêu để tăng xác suất dự đoán đúng các từ ngữ cảnh dựa trên từ trung tâm. Điều này được thực hiện bằng cách sử dụng các phương pháp như hàm *softmax* hoặc *negative sampling* để giảm độ phức tạp tính toán.

Giải thích các khái niệm:

1. Word Embeding (Vector biểu diễn từ)

Word embedding là kỹ thuật chuyển đổi từ ngữ thành các vector số trong không gian nhiều chiều. Trong không gian này, các từ có ý nghĩa gần gũi hoặc tương tự sẽ được biểu diễn gần nhau, trong khi các từ có ý nghĩa khác biệt sẽ nằm xa nhau.

Máy tính không thể trực tiếp hiểu ngôn ngữ tự nhiên của con người, do đó việc chuyển đổi từ ngữ thành dạng số là điều cần thiết. Tuy nhiên, các phương pháp truyền thống như *one-hot encoding* hoặc *bag-of-words* thường không hiệu quả vì chúng không thể biểu diễn được mối quan hệ và ngữ cảnh giữa các từ. Word embedding ra đời để khắc phục vấn đề này bằng cách học các vector biểu diễn từ, chứa thông tin về ngữ nghĩa và ngữ cảnh của chúng trong các câu và văn bản.

Vai trò của Skip-gram trong word embedding:

Skip-gram không chỉ biểu diễn mối quan hệ trực tiếp giữa các từ mà còn có khả năng thể hiện các quan hệ ngữ nghĩa phức tạp thông qua phép toán trong không gian vector. Ví dụ:

- *"man" - "woman" + "girl" ≈ "boy"*

Phép toán này cho thấy cách Skip-gram học được mối quan hệ giới tính trong không gian từ, từ đó biểu diễn thông tin ngữ nghĩa một cách hiệu quả. Điều này làm cho Skip-gram trở thành một mô hình mạnh mẽ trong việc biểu diễn thông tin ngôn ngữ.

2. Context Window (Cửa sổ ngữ cảnh)

Cửa sổ ngữ cảnh là khoảng vùng từ nằm xung quanh một từ trung tâm mà mô hình Skip-gram dựa vào để học và huấn luyện. Phạm vi này xác định các từ ngữ cảnh sẽ được xem xét khi dự đoán mối quan hệ với từ trung tâm trong quá trình học.

Ví dụ:

- Nếu từ trung tâm là *"ăn"*, cửa sổ ngữ cảnh sẽ bao gồm các từ xung quanh nó trong một khoảng nhất định, như *"cơm", "ngon", "bữa", "trưa"*.
- Cửa sổ ngữ cảnh xác định số từ lân cận sẽ được xem xét. Nếu cửa sổ ngữ cảnh có khoảng cách 2 từ, thì các từ trong khoảng từ 2 từ trước và sau từ trung tâm sẽ được đưa vào.

Vai trò trong Skip-gram:

- Cửa sổ ngữ cảnh giúp mô hình Skip-gram học được mối liên kết ngữ nghĩa giữa từ trung tâm và các từ lân cận.

- Một cửa sổ ngữ cảnh rộng hơn có thể nắm bắt mối quan hệ ngữ nghĩa xa hơn, nhưng cũng làm tăng độ phức tạp tính toán.
- Ngược lại, cửa sổ nhỏ sẽ tập trung vào các mối quan hệ gần gũi và cụ thể trong ngữ cảnh hẹp.

Kích thước cửa sổ:

Kích thước của cửa sổ ngữ cảnh phụ thuộc vào đặc điểm của từng loại dữ liệu:

- Dữ liệu ngắn hạn như tiêu đề hoặc câu ngắn: Nên chọn cửa sổ nhỏ (2-5 từ) để tập trung vào các mối quan hệ gần gũi và trực tiếp giữa từ trung tâm và từ lân cận.
- Dữ liệu dài như bài báo hoặc đoạn văn: Cần sử dụng cửa sổ lớn (5-10 từ) để mô hình có thể học được thông tin ngữ cảnh phong phú và mối liên kết dài hạn.

3. Cosine Similarity (Độ tương đồng Cosine)

Cosine similarity là thước đo được sử dụng để đánh giá mức độ tương đồng giữa hai vector trong không gian.

Công thức tính:

$$\textbf{Cosine Similarity} = \frac{\vec{u}\cdot\vec{v}}{\|\vec{u}\|\cdot\|\vec{v}\|}$$

Trong đó:

- $\vec{u}$ và $\vec{v}$: Hai vector cần so sánh.
- $\|\vec{u}\|$ và $\|\vec{v}\|$: Độ dài của vector $\vec{u}$ và $\vec{v}$

Vai trò trong Skip-gram:

- Được sử dụng để đánh giá mối quan hệ giữa các từ trong không gian vector:
    - Cosine Similarity giúp xác định mức độ gần gũi về mặt ngữ nghĩa giữa các từ thông qua khoảng cách trong không gian vector.

    - Ví dụ: “anh” và “chị” có Cosine Similarity cao, điều đó cho thấy chúng có ý nghĩa tương tự trong không gian biểu diễn.

- Phát hiện từ đồng nghĩa và từ tương tự nghĩa:
    - Thông qua chỉ số Cosine Similarity, mô hình có thể xác định các từ có ngữ nghĩa gần nhau trong cùng ngữ cảnh. Điều này hỗ trợ nhiều bài toán NLP như phân tích cảm xúc, dịch máy hoặc chatbot.
- Đánh giá chất lượng Word Embedding:
    - Nếu các từ gần ngữ nghĩa có Cosine Similarity cao, điều đó thể hiện mô hình Skip-gram đã học được thông tin ngữ nghĩa một cách hiệu quả từ dữ liệu huấn luyện.

III. Các thư viện được sử dụng

1. Json

JSON (JavaScript Object Notation) là một định dạng trao đổi dữ liệu phổ biến, được thiết kế để dễ đọc và xử lý bởi con người cũng như máy tính. Trong Python, thư viện json cung cấp các công cụ để xử lý dữ liệu JSON, như chuyển đổi giữa định dạng JSON và các đối tượng Python (list, dictionary).

Ứng dụng trong đề tài:

- Tải dữ liệu: Sử dụng json để đọc dữ liệu thô từ các tệp JSON chứa văn bản hoặc ngữ liệu.
- Lưu dữ liệu đã xử lý: Sau khi tách từ và loại bỏ các tên riêng, lưu dữ liệu đã qua xử lý về định dạng JSON để tiện sử dụng trong các bước tiếp theo.

2. PyVi

PyVi là một thư viện xử lý ngôn ngữ tự nhiên dành riêng cho tiếng Việt, cung cấp các công cụ như tách từ, gắn nhãn từ loại, và nhận diện thực thể.

Ứng dụng trong đề tài:

- Tách từ: Vì tiếng Việt là ngôn ngữ không có dấu cách giữa các từ ghép, PyVi được sử dụng để phân đoạn văn bản thành các từ riêng biệt, đảm bảo độ chính xác khi xử lý.

- Loại bỏ tên riêng (Np): Dựa vào nhãn từ loại mà PyVi cung cấp, có thể lọc bỏ các từ thuộc danh mục tên riêng (Np) để giảm nhiễu trong tập dữ liệu, đảm bảo các embedding vectors tập trung vào các từ thông dụng hơn.

3. Numpy

NumPy (Numerical Python) là một thư viện mạnh mẽ dành cho tính toán khoa học và xử lý dữ liệu số trong Python. Nó hỗ trợ thao tác với mảng (array) và cung cấp các hàm toán học hiệu quả.

Ứng dụng trong đề tài:

- Huấn luyện mô hình: NumPy được sử dụng để thực hiện các phép tính trên ma trận và vector trong quá trình huấn luyện mô hình Skip-gram, như tính tích vô hướng và cập nhật trọng số.
- Lưu embedding vectors: Sau khi huấn luyện xong, các vector biểu diễn từ (embedding vectors) được lưu dưới dạng mảng NumPy để tái sử dụng hoặc đánh giá.

4. Matplotlib

Matplotlib là một thư viện trực quan hóa dữ liệu mạnh mẽ và linh hoạt trong Python. Nó cho phép tạo ra các biểu đồ, đồ thị, và hình ảnh minh họa một cách dễ dàng. Với sự hỗ trợ từ Matplotlib, người dùng có thể theo dõi và phân tích trực quan các chỉ số trong quá trình huấn luyện mô hình.

Ứng dụng trong đề tài:

- Trực quan hóa quá trình cải thiện hàm mất mát (loss): Matplotlib được sử dụng để vẽ biểu đồ thể hiện sự giảm dần của giá trị hàm mất mát qua các epoch. Điều này giúp đánh giá hiệu quả của việc huấn luyện và kiểm tra xem mô hình có hội tụ hay không.
- Hiển thị các phân tích kết quả: Ngoài loss, có thể dùng Matplotlib để hiển thị các biểu đồ liên quan đến hiệu suất của embedding vectors, chẳng hạn như so sánh khoảng cách ngữ nghĩa giữa các từ.

PHẦN 2: DỮ LIỆU

I. Giới thiệu về bộ dữ liệu

Bộ dữ liệu sử dụng trong đề tài gồm các bài viết tin tức, gồm nhiều danh mục và đề tài khác nhau. Bộ dữ liệu này được lấy từ trang chính của "Kaggle" có tên là "Vietnamese Online News Dataset". Bộ dữ liệu gồm 6000 bài viết, mỗi bài viết chứa 10 trường:

- Id: Khóa chính trong cơ sở dữ liệu, chỉ để nhận dạng.
- Author: Tên của một người/nhóm nhà báo cáo đã viết bài báo.
- Title: Tiêu đề của bài viết.
- Content: Nội dung của bài viết.
- Picture Count: Có bao nhiêu hình ảnh được sử dụng trong bài viết.
- Topic: Chủ đề chính của bài viết.
- URL: URL của bài viết.
- Source: Nguồn tin.
- Crawled At: Bài viết được thu thập dữ liệu khi nào.
- Processed: Có thể bỏ qua.

Trong nghiên cứu này thì chỉ tập trung vào hai trường dữ liệu là ***Id*** và ***Content***. ***Id*** được sử dụng để định danh, trong khi ***Content*** chứa nội dung bài báo – là nguồn thông tin chính cho quá trình phân tích và xử lý. Tổng số từ trong trong trường ***Content*** là 3035540 từ.

II. Mục tiêu và yêu cầu

Quy trình tiền xử lý dữ liệu trong dự án này hướng đến việc:

- Loại bỏ thông tin thừa hoặc không cần thiết như URL, số, ký tự đặc biệt và thông tin trong các dấu ngoặc.
- Tách từ và gán nhãn từ loại để làm sạch dữ liệu và chỉ giữ các từ có ý nghĩa trong ngữ cảnh học máy.
- Chạy xử lý song song để cải thiện tốc độ khi làm việc với tập dữ liệu lớn.
- Tạo từ điển từ vựng (vocab) dựa trên tần suất xuất hiện của các từ trong tập dữ liệu sau khi tiền xử lý xong.

III. Các bước cụ thể

Bước 1: Làm sạch văn bản:

Đầu tiên, ta cần làm sạch dữ liệu thô từ các thông tin được thu thập. Việc làm sạch bao gồm loại bỏ các URL, các ký tự không cần thiết và chuẩn hóa cấu trúc văn bản để làm dữ liệu sẵn sàng cho các bước tiếp theo.

Hàm clean_text thực hiện nhiệm vụ này:

```
def clean_text(text):
    text = re.sub(r"https?://\S+|www\.\S+", " ", text)  # Loại bỏ URL
    text = re.sub(r"\([^)]*\)", " ", text)  # Loại bỏ nội dung trong ngoặc
    texts = re.split(r'(?<=[.,!?;])\s+', text.strip())  # Tách câu
    text = re.sub(r"\b\w*\d\w*\b", " ", text)  # Loại bỏ số
    texts = [re.sub(r"[^\w\s]", " ", text) for text in texts]  # Loại bỏ ký
tự đặc biệt

    return texts
```

Bước 2: Token hóa và gán nhãn từ loại:

Sau khi hoàn thành bước làm sạch văn bản, dữ liệu sẽ được tách thành các đơn vị nhỏ là token thông qua công cụ *ViTokenizer*. Tiếp theo, các token này sẽ được gán nhãn từ loại bằng công cụ *ViPosTagge*. Để làm giảm độ nhiễu trong dữ liệu, các từ có nhãn là danh từ riêng (NP) sẽ được loại bỏ khỏi tập dữ liệu.

```
# Hàm xử lý tách từ cho một item
def process_item(item):
    # Tách từ cho từng dòng trong 'contents'
    tokenized_contents = [ViTokenizer.tokenize(it)
                          for it in item['contents']]

    item['contents'] = []
    for tokenized_line in tokenized_contents:
        # Gán nhãn từ loại
        words, pos_tags = ViPosTagger.postagging(tokenized_line)

        # Loại bỏ các từ loại 'Np' (danh từ riêng)
        filtered_words = [word
            for word, pos in zip(words, pos_tags)
                if pos != 'Np']
        item['contents'].append(" ".join(filtered_words))  # Gộp lại thành
câu

    return item
```

Bước 3: Xử lý dữ liệu song song:

Dữ liệu thường có kích thước lớn, và việc xử lý tuần tự có thể tiêu tốn nhiều thời gian. Để tối ưu hóa hiệu suất, kỹ thuật xử lý song song được áp dụng thông qua việc sử dụng module *multiprocessing*, giúp phân chia công việc và thực hiện chúng đồng thời.

```
# Hàm chính để xử lý song song
def parallel_processing(data):
    with multiprocessing.Pool(processes=multiprocessing.cpu_count()) as pool:
        results = list(tqdm(pool.imap(process_item, data), total=len(data)))
    return results
```

Bước 4: Chuẩn hóa dữ liệu và lọc thông tin rỗng:

Sau khi hoàn tất việc token hóa và lọc từ loại, dữ liệu sẽ được chuyển đổi về dạng chữ thường để duy trì tính nhất quán và chuẩn hóa văn bản. Đồng thời, các từ rỗng hoặc không mang thông tin sẽ được loại bỏ để đảm bảo chất lượng dữ liệu và giảm tiếng ồn trong quá trình xử lý.

```
# Chuyển thành chữ thường
for item in tqdm(data):
    item["contents"] = [it.lower() for it in item["contents"]]


# Xoá các item không có nội dung hoặc nội dung rỗng
data = [item
      for item in data
            if item["contents"] and all([it
                              for it in item["contents"]])]
```

Bước 5: Tạo từ điển từ vựng:

Tiến trình tiếp theo là tạo từ điển từ vựng từ toàn bộ các từ trong dữ liệu đã làm sạch. Việc tạo từ điển giúp ánh xạ từ ngữ tự nhiên thành chỉ số trong không gian vector để mô hình có thể huấn luyện.

```
# Gộp tất cả các từ lại
all_words = " ".join([it
      for item in tqdm(data)
            for it in item["contents"]]).split()
all_words = [word.strip() for word in all_words]

# Đếm tần suất xuất hiện của các từ
vocab = Counter(all_words)

# Tạo từ điển từ vựng (word2idx)
word_to_idx = {word: idx for idx, word in tqdm(enumerate(vocab.keys()))}
```

Bước 6: Lưu dữ liệu đã xử lý

Cuối cùng, dữ liệu đã làm sạch và từ điển được lưu vào các file JSON để mô hình có thể đọc và sử dụng trong quá trình huấn luyện.

```
# Load data
with open(input_data, 'r', encoding='utf-8') as f:
    data = json.load(f)

with open(input_vocab, 'r', encoding='utf-8') as f:
    vocab = json.load(f)
```

Sau khi xử lý dữ liệu, tổng số từ vựng là 20821 từ.

PHẦN 3: THIẾT KẾ MÔ HÌNH SKIP-GRAM

I. Tổng quan về mô hình

Skip-gram là một kỹ thuật nằm trong mô hình *Word2Vec*, được sử dụng để học các vector biểu diễn từ ngữ nghĩa trong không gian số học. Mô hình *Skip-gram* hoạt động thông qua việc dự đoán các từ trong phạm vi ngữ cảnh lân cận của một từ trung tâm. Điều này cho phép biểu diễn các từ có ý nghĩa tương tự gần nhau trong không gian vector, và khoảng cách giữa các vector này có thể tính toán được.

Trong dự án này, mô hình *Skip-gram* được thiết kế và triển khai từ đầu với cấu trúc đơn giản sử dụng hai ma trận trọng số như sau:

- **W1 (Trọng số từ trung tâm):** Biểu diễn thông tin liên quan đến từ trung tâm.
- **W2 (Trọng số ngữ cảnh):** Biểu diễn thông tin của các từ nằm trong ngữ cảnh lân cận từ trung tâm.

II. Giới thiệu về mô hình sẽ được thiết kế

Mô hình Skip-gram được triển khai trong dự án sử dụng hai ma trận trọng số, W1 và W2. Quá trình huấn luyện của mô hình dựa trên việc tối ưu hóa hàm mất mát bằng *Softmax* để dự đoán các từ trong ngữ cảnh của từ trung tâm và cải thiện các trọng số thông qua phép lan truyền ngược (backpropagation).

Các bước chính trong mô hình bao gồm:

1. **Forward Pass:** Dự đoán xác suất của các từ ngữ cảnh dựa vào từ trung tâm thông qua hàm Softmax.
2. **Backward Pass:** Tính toán sai số và cập nhật trọng số thông qua lan truyền ngược.
3. **Huấn luyện mô hình:** Dùng các tập dữ liệu về các cặp từ (center-context) để huấn luyện mô hình.

1. Ma trận trọng số (Weight Matrices)

Mô hình Skip-gram sử dụng hai ma trận trọng số chính, được ký hiệu là W1 và W2. Chúng đóng vai trò quan trọng trong việc học biểu diễn từ:

1.1. Ma trận W1 (Trọng số trung tâm)

- Đây là ma trận biểu diễn thông tin từ trung tâm trong mô hình.
- Kích thước của ma trận là **(***Vocabulary Size x Embedding Dimension***)**, với *Vocabulary Size* là tổng số từ trong từ điển và *Embedding Dimension* là chiều dài của vector biểu diễn từ.
- W1 được sử dụng để ánh xạ các chỉ số của từ trung tâm thành các vector trong không gian số học.

1.2. Ma trận W2 (Trọng số ngữ cảnh)

- Biểu diễn thông tin ngữ cảnh lân cận tương ứng.
- Kích thước của ma trận là *(Embedding Dimension x Vocabulary Size).*
- W2 dùng để tính toán xác suất các từ ngữ cảnh lân cận thông qua phép nhân với từ trung tâm.

Hai ma trận này được khởi tạo ngẫu nhiên và cập nhật thông qua quá trình huấn luyện.

2. Softmax Function

Softmax là một hàm kích hoạt quan trọng trong mô hình Skip-gram. Nó chuyển đổi các giá trị thô (logits) thành xác suất, mô phỏng xác suất xuất hiện của các từ ngữ cảnh dựa vào từ trung tâm.

Công dụng chính:

- Biến đổi các đầu ra thô từ mô hình thành một xác suất trong khoảng [0, 1].
- Tổng của tất cả xác suất này sẽ bằng 1, đảm bảo tính xác suất hợp lý trong dự đoán của mô hình.

Softmax thường được áp dụng sau phép nhân giữa từ trung tâm và trọng số ngữ cảnh W2.

3. Forward Pass

Forward pass là quá trình tính toán từ đầu vào thông qua mô hình để dự đoán các xác suất ngữ cảnh lân cận của từ trung tâm:

1. Dùng trọng số W1 để lấy thông tin từ trung tâm và ánh xạ thành vector nhúng.
2. Tính toán logits thông qua phép nhân giữa vector từ trung tâm và W2.
3. Áp dụng hàm *softmax* để chuyển *logits* thành các xác suất, thể hiện xác suất dự đoán các từ trong ngữ cảnh.

Forward pass cung cấp thông tin dự đoán của mô hình cho mỗi từ trung tâm.

4. Backward Pass và Gradient Calculation

Backward pass là bước quan trọng để cập nhật trọng số thông qua việc tính toán gradient và tối ưu hóa sai số:

4.1. Tính sai số (Error)

- Sai số được tính bằng cách lấy hiệu giữa xác suất dự đoán từ softmax và một vector one-hot biểu diễn thông tin ngữ cảnh thực tế.

4.2. Gradient

- Dựa vào sai số và các trọng số W1, W2 để tính toán gradient.

- Gradient được tính thông qua phép nhân giữa sai số và thông tin từ mô hình.

4.3. Cập nhật trọng số

- Gradient được tính toán và nhân với tốc độ học để cập nhật W1 và W2.

Cập nhật trọng số này sẽ giúp mô hình học dần dần sao cho xác suất dự đoán các từ ngữ cảnh ngày càng chính xác hơn.

5. Huấn luyện mô hình (Training)

Quá trình huấn luyện diễn ra trong nhiều epoch với các cặp từ trung tâm-ngữ cảnh được tạo từ dữ liệu đầu vào:

- **Epochs:** Là số vòng lặp qua toàn bộ tập huấn luyện.
- **Batch size:** Dùng để chia dữ liệu thành các tập nhỏ và huấn luyện theo từng batch.

Trong huấn luyện, các trọng số W1 và W2 được cập nhật thông qua gradient và tối ưu hóa sai số qua mỗi batch.

6. Embedding và tính khoảng cách (Cosine Similarity)

6.1. Vector biểu diễn (Embedding)

- Dùng trọng số W1 sau khi huấn luyện để lấy thông tin biểu diễn từ ngữ trong không gian số học.
- Mỗi từ sẽ có một vector biểu diễn duy nhất.

6.2. Cosine Similarity

- Được tính giữa các vector biểu diễn để đánh giá khoảng cách ngữ nghĩa giữa hai từ.
- Nếu hai từ có cosine similarity gần 1, chúng gần về mặt ngữ nghĩa; nếu gần -1, chúng xa về mặt ngữ nghĩa.

7. Lưu và tải mô hình

Mô hình Skip-gram cung cấp chức năng lưu các trọng số (W1 và W2) sau khi huấn luyện xong để tái sử dụng sau này. Điều này giúp tiết kiệm thời gian và tài nguyên trong các giai đoạn sau của dự án hoặc trong các tác vụ phân tích.

PHẦN 4: HUẤN LUYỆN MÔ HÌNH, KẾT QUẢ VÀ ĐÁNH GIÁ

I. Triển khai mô hình và lý do chọn các tham số

Tham số	Giá trị	Lý do lựa chọn
Kích thước vector nhúng (embedding dimension)	100	Kích thước 100 đủ lớn để biểu diễn thông tin ngữ nghĩa cơ bản của các từ, đồng thời tiết kiệm tài nguyên tính toán và bộ nhớ hơn so với các kích thước lớn hơn như 300.
Batch size	1024	Kích thước này đủ lớn để giảm nhiễu trong gradient, đảm bảo quá trình huấn luyện ổn định mà không cần tăng số lần cập nhật quá nhiều.
Epochs	12	Huấn luyện trong 12 vòng lặp để đảm bảo mô hình học đủ thông tin từ dữ liệu mà không bị overfitting.
Window size	4	Tiếng việt thường có các cụm từ hoặc mối quan hệ ngữ nghĩa trong phạm vi 4 từ.
Learning rate	0.01	Giá trị 0.01 đủ nhỏ để đảm bảo mô hình học ổn định, tránh việc bỏ qua điểm cực trị hoặc giao động quanh điểm tối ưu.

II. Quy trình huấn luyện

1. Tạo các cặp dữ liệu Skip-gram

- Dùng hàm skipgram_pairs để tạo các cặp dữ liệu trung tâm và ngữ cảnh.

- **Window size = 4**: Kích thước cửa sổ 4 đủ để nắm bắt ngữ cảnh gần gũi xung quanh từ trung tâm mà không làm mất đi các mối quan hệ quan trọng. Tiếng Việt thường có các cụm từ hoặc mối quan hệ ngữ nghĩa trong phạm vi 4 từ.

2. Huấn luyện mô hình

- Dùng thuật toán **Skip-gram model** để huấn luyện với cặp dữ liệu được tạo ở trên.
- Huấn luyện với batch size = 1024 và lặp qua 12 epochs.

3. Loss trong quá trình huấn luyện

- Loss giảm đều qua từng epoch: Loss trung bình giảm từ 7.3954 ở epoch đầu tiên xuống 6.5972 ở epoch cuối cùng. Điều này cho thấy mô hình đang học hiệu quả và cập nhật các trọng số phù hợp để tối ưu hóa.

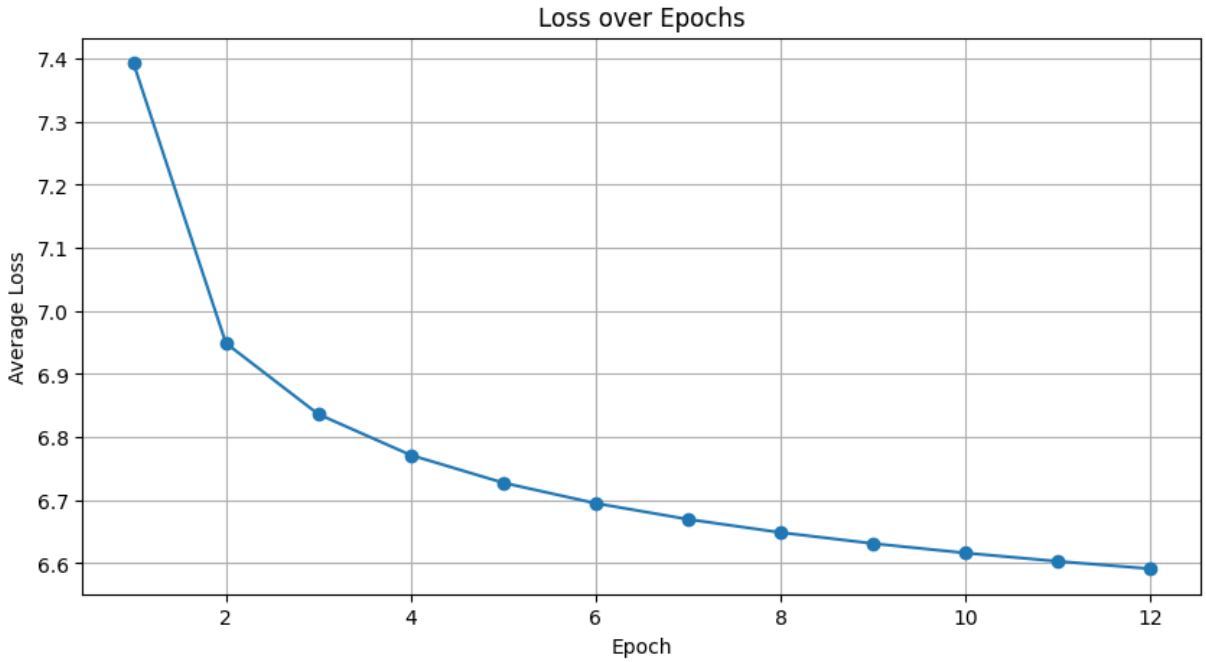
4. Tính toán cosine similarity

- Sau khi huấn luyện xong, tính toán **cosine similarity** giữa các cặp từ cụ thể để đánh giá xem mô hình có học được các mối quan hệ ngữ nghĩa tốt không.
- Ví dụ:
    - Tính cosine similarity giữa *"bệnh viện"* và *"y tế"*, *"mua"* và *"bán"*, *"tối"* và *"xấu"*,...

III. Kết quả thu được từ huấn luyện

1. Loss trong quá trình huấn luyện

Loss giảm đều qua 12 vòng lặp với các thông tin như sau:



Epoch1 = 7.3954 và Epoch2 = 6.5972 cho ta thấy Loss giảm đều qua mỗi Epoch, cho ta thấy được mô hình đang học và cải thiện thông qua quá trình huấn luyện. Tuy vậy, mô hình vẫn chưa tối ưu hoàn toàn và có thể cần thêm thời gian huấn luyện và tinh chỉnh. Giá trị Loss chưa hoàn toàn ổn định và còn có xu hướng giảm nếu Epoch nhiều hơn. Do đó, mô hình vẫn chưa hoàn toàn hội tụ.

2. Tính cosine similarity

Các kết quả tính cosine similarity cho thấy mối quan hệ ngữ nghĩa giữa một số từ quan trọng:

- Similarity giữa *"công ty"* và *"doanh nghiệp"*: 0.6762
  Đây là một giá trị khá cao, cho thấy hai từ này có sự tương đồng ngữ nghĩa trong ngôn ngữ hàng ngày và pháp luật.
- Similarity giữa *"bệnh viện"* và *"y tế"*: 0.5854
  Cho thấy hai từ này có sự tương đồng ngữ nghĩa ở mức vừa phải, *"bệnh viện"* là một phần trong lĩnh vực *"y tế"*.

- Similarity giữa *"mua"* và *"bán": 0.7204*
  Đây là một giá trị khá cao, cho thấy hai từ này có mức độ tương đồng ngữ nghĩa đáng kể, hai từ này không phải là hai khái niệm hoàn toàn tách biệt mà có mối quan hệ bổ sung lẫn nhau trong các ngữ cảnh kinh tế và thương mại.
- Similarity giữa *"tối"* và *"xấu":* 0.2104
  Đây là một giá trị khá thấp, cho thấy chúng có sự tương đồng ngữ nghĩa yếu. Hai từ này không liên quan về mặt ngữ nghĩa, nhưng vẫn có thể liên quan đến cảm xúc tiêu cực hoặc trạng thái không mong muốn.
- Similarity giữa *"đại dương"* và *"hạnh phúc":* 0.1172
  Đây là một giá trị rất thấp, cho thấy hai từ này có sự tương đồng ngữ nghĩa yếu, gần như không liên quan. Chúng có thể có sự tương đồng trong ngữ cảnh văn học hoặc cảm xúc, không phải mối quan hệ ngữ nghĩa rõ ràng.
- Similarity giữa *"giáo viên"* và *"cửa sổ":* -0.0313
  Đây là một giá trị nhỏ hơn 0, cho thấy chúng không có bất kì mối quan hệ ngữ nghĩa nào.

IV. Phân tích và đánh giá kết quả

- **Loss giảm đều qua các epochs**

Loss giảm cho thấy mô hình đang học thông tin từ dữ liệu và đang tối ưu hóa các trọng số một cách hiệu quả. Loss trung bình kết thúc tại mức 6.5972 sau 12 epochs.

- **Vector nhúng có thể phản ánh ngữ nghĩa**

Thông qua tính toán cosine similarity, mô hình đã học được các mối quan hệ ngữ nghĩa giữa các từ, như *"công_ty"* gần với *"doanh_nghiệp", "mua"* có mối liên kết với *"bán "*. Điều này khẳng định mô hình đã học được thông tin ngữ nghĩa và biểu diễn chúng trong không gian vector.

Các từ có độ tương đồng cao

- *"công_ty"* và *"doanh_nghiệp"* có similarity = 0.6762, hai từ này có mối quan hệ tương đồng cao, cho thấy mô hình học đúng trong mối quan hệ xã hội và pháp luật.
- *"mua"* và *"bán"* có similarity = 0.7204, hai từ này có độ tương đồng khá cao, cho thấy mô hình học đúng trong mối quan hệ kinh tế và thương mại.

Tuy nhiên, mô hình còn gặp khó khăn với các từ sau:

- *"tối"* và *"xấu"* có similarity = 0.2104, cho thấy mô hình không học được ngữ nghĩa hoặc chưa phát hiện được mối liên kết hợp lí trong ngữ cảnh.
- *"đại_dương"* và *"hạnh_phúc"* có similarity = 0.1127, cho thấy mô hình phải cần được thể hiện liên tưởng trong ngữ cảnh văn học hoặc ngữ cảnh ẩn dụ.

PHẦN 5: KHÓ KHĂN, BÀI HỌC KINH NGHIỆM VÀ KẾT LUẬN

I. Khó khăn

1. Dữ liệu không đủ đa dạng

Dữ liệu được thu thập và xử lý có hạn chế về ngữ cảnh và từ vựng. Điều này dẫn đến mô hình học không đủ thông tin để biểu diễn đầy đủ ý nghĩa ngữ nghĩa của từ, ảnh hưởng đến khả năng biểu diễn và khả năng tính toán similarity giữa các từ.

Giải pháp trong tương lai là mở rộng bộ dữ liệu, thêm nhiều văn bản từ nhiều nguồn ngôn ngữ và ngữ cảnh khác nhau.

2. Tốc độ xử lý tách từ khi chạy đơn luồng chậm

Quá trình tách từ và tạo các cặp dữ liệu Skip-gram thông qua các câu lệnh xử lý đơn luồng tốn nhiều thời gian. Việc xử lý dữ liệu lớn với hàng chục nghìn câu khiến tốc độ trở nên chậm.

Điều này cho thấy cần thiết kế lại các thuật toán xử lý dữ liệu để hỗ trợ **đa luồng (multithreading)** hoặc các phương pháp tối ưu hóa.

3. Bộ nhớ không đủ để lưu dữ liệu pairs

Với hàng chục nghìn câu và hàng chục nghìn từ vựng, việc lưu trữ tất cả các cặp từ trung tâm và ngữ cảnh (word pairs) trong bộ nhớ tốn một lượng bộ nhớ lớn.

Giải pháp tiềm năng là áp dụng các kỹ thuật lưu trữ dữ liệu hiệu quả hơn, chẳng hạn lưu trực tiếp vào ổ đĩa hoặc chia nhỏ dữ liệu thành các batch nhỏ hơn để xử lý.

4. Tốc độ huấn luyện mô hình chậm

Mô hình Skip-gram huấn luyện tốn rất nhiều thời gian với các thông số như batch size lớn, window size và vocab size. Thêm vào đó, việc lặp qua 12 epoch với dữ liệu lớn làm tăng đáng kể thời gian huấn luyện.

Giải pháp trong tương lai có thể bao gồm tối ưu hóa thuật toán huấn luyện, áp dụng đa luồng hoặc sử dụng GPU để tăng tốc độ tính toán.

5. Hạn chế của mô hình

Mô hình học chưa tốt trong việc phản ánh mối quan hệ ngữ nghĩa trong một số ngữ cảnh cụ thể. Mô hình cũng chưa nắm bắt được một số liên kết ngữ nghĩa trừu tượng hoặc ẩn dụ trong ngôn ngữ tự nhiên.

II. Bài học kinh nghiệm

1. Dữ liệu là yếu tố quyết định

Dữ liệu không đủ đa dạng làm giảm chất lượng mô hình. Vì vậy, khi xây dựng mô hình học từ ngữ nghĩa, cần đầu tư thêm vào thu thập và làm phong phú bộ dữ liệu.

2. Cần tối ưu hóa tốc độ xử lý dữ liệu

Thay vì chạy các tác vụ xử lý dữ liệu tuần tự, nên áp dụng các kỹ thuật ***đa luồng (multithreading)*** hoặc tối ưu hóa thuật toán để giảm thời gian chờ và đẩy nhanh tốc độ xử lý dữ liệu.

3. Quản lý bộ nhớ hiệu quả

Lưu trữ và xử lý dữ liệu lớn cần thiết kế bộ nhớ hiệu quả hơn. Thay vì giữ toàn bộ dữ liệu trong bộ nhớ, nên lưu trữ từng batch nhỏ hoặc tối ưu hóa kích thước dữ liệu.

4. Đa dạng hóa mô hình và tối ưu hóa tốc độ huấn luyện

Thời gian huấn luyện lâu dài do mô hình và dữ liệu lớn. Do đó, trong các dự án tiếp theo, cần thử nghiệm các kỹ thuật như **GPU acceleration** hoặc các chiến lược huấn luyện tối ưu để cải thiện tốc độ huấn luyện.

III. Kết luận

Trong quá trình huấn luyện mô hình Skip-gram và tính toán cosine similarity, em đã hoàn thành việc biểu diễn từ và khai thác thông tin ngữ nghĩa từ dữ liệu một cách thành công. Kết quả đạt được cho thấy mô hình đã nắm bắt thông tin ngữ nghĩa trong không gian vector và thực hiện hiệu quả việc tính toán các mức độ tương đồng giữa các từ.

Tuy nhiên, một số thách thức như dữ liệu còn thiếu tính đa dạng, tốc độ huấn luyện và dung lượng bộ nhớ cần được cải thiện là những vấn đề cần được giải quyết trong các nghiên cứu và dự án trong tương lai. Những bài học kinh nghiệm từ quá trình phát triển và triển khai mô hình sẽ là nền tảng quan trọng để cải tiến và phát triển các mô hình học từ sau này.

Việc lưu trữ thành công mô hình và các trọng số đã đạt kết quả tích cực, mở ra cơ hội để tiếp tục nghiên cứu và áp dụng vào các bài toán liên quan đến ngôn ngữ tự nhiên và học máy.

TÀI LIỆU THAM KHẢO

1. **Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013).**
   Efficient Estimation of Word Representations in Vector Space.
   ArXiv.org.
2. **Goldberg, Y., & Levy, O. (2014).**
   Word2Vec Explained: Deriving Mikolov's Skip-Gram Model with Negative Sampling.
   Google Research.

3. **Jurafsky, D., & Martin, J. H. (2020).**
Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.
Prentice Hall.
4. **Ruder, S. (2016).**
An Overview of Gradient Descent Optimization Algorithms.
Retrieved from https://ruder.io/
5. **Nguyen, P., et al. (2020).**
Vietnamese Sentiment Analysis Using Word Embeddings.
Journal of Computational Linguistics.
6. **Nguyen, P., et al. (2020).**
Vietnamese Sentiment Analysis Using Word Embeddings.
Computational Linguistics Journal.VanderPlas, J. (2016).
7. **McCormick, C. (2019).**
Building Word2Vec from Scratch in Python.
Retrieved from https://www.shu.ai/blog

PHỤ LỤC

I. Mô tả thuật toán Skip-gram

Input và Output:

- Input: Một từ trung tâm W_t và một cửa sổ ngữ cảnh có kích thước c.
- Output: Xác suất các từ ngữ cảnh xuất hiện gần từ trung tâm.

Các bước thực hiện:

- Bước 1: Biểu diễn từ trung tâm W_t dưới dạng one-hot vector:
    - Từ trung tâm được biểu diễn thông qua vector one-hot, tức là một vector có một giá trị là 1 tại chỉ số của từ đó và các giá trị còn lại là 0.
- Bước 2: Tính vector nhúng từ trung tâm bằng cách nhân one-hot vector với ma trận nhúng W_1 :
    - Ma trận W_1 được nhân với vector one-hot để tạo ra vector nhúng đầu tiên cho từ trung tâm.
- Bước 3: Dùng ma trận W_2 và hàm softmax để tính xác suất đầu ra

    - Vector nhúng được nhân với ma trận W_2 , sau đó áp dụng hàm softmax để tính xác suất của các từ ngữ cảnh dựa trên mô hình dự đoán.
- Bước 4: Tính hàm mất mát cross-entropy để đo lường sai lệch giữa dự đoán và từ ngữ cảnh thực tế:
    - Cross-entropy đo lường khoảng cách giữa xác suất dự đoán và xác suất thực tế của các từ ngữ cảnh trong cửa sổ.
- Bước 5: Cập nhật các trọng số W_1 và W_2 thông qua lan truyền ngược và gradient descent:
    - Sử dụng kỹ thuật lan truyền ngược và thuật toán gradient descent để tối ưu các trọng số, từ đó cải thiện hiệu quả mô hình qua từng vòng lặp huấn luyện.

II. Ví dụ minh họa

Dataset (tập dữ liệu): "Bầu trời hôm nay nhiều sao."

Vocab (từ vựng): {'Bầu', 'trời', 'hôm', 'nay', 'nhiều', 'sao'}

Dữ liệu huấn luyện (Context Window = 2):

Từ trung tâm	**Từ ngữ cảnh**
bầu	trời
bầu	hôm
trời	bầu
trời	hôm
trời	này
hôm	nay
hôm	nhiều
hôm	trời
...	...

III. Đánh giá

Công thức tính cosine similarity giữa hai vector $\vec{u}$ và $\vec{v}$:

$$\textbf{Cosine Similarity} = \frac{\vec{u} \cdot \vec{v}}{\|\vec{u}\| \cdot \|\vec{v}\|}$$

Trong đó:

- $\vec{u}$ và $\vec{v}$: Hai vector cần so sánh.
- $\|\vec{u}\|$ và $\|\vec{v}\|$: Độ dài của vector $\vec{u}$ và $\vec{v}$

IV. Toàn bộ mã của dự án và Dataset

Github: thienthuan25/NLP

Dataset: Vietnamese Online News Dataset