TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH

KHOA ĐIỆN ĐIỆN TỬ

BỘ MÔN KỸ THUẬT MÁY TÍNH - VIỄN THÔNG



BÁO CÁO TIỂU LUẬN

**XỬ LÝ NGÔN NGỮ TỰ NHIÊN**

**BÀI TOÁN SINH TỪ**

**NGÀNH CÔNG NGHỆ KỸ THUẬT ĐIỆN – ĐIỆN TỬ**

Giáo viên hướng dẫn:

**PGS.TS TRƯƠNG NGỌC SƠN**

Sinh viên: **BẠCH ĐÌNH HIỀN**

MSSV: 18142286

**ĐỖ QUỐC VIỆT**

MSSV: 18142417

TP. HỒ CHÍ MINH – 06/2021

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH

KHOA ĐIỆN ĐIỆN TỬ

BỘ MÔN KỸ THUẬT MÁY TÍNH - VIỄN THÔNG



BÁO CÁO TIỂU LUẬN

**XỬ LÝ NGÔN NGỮ TỰ NHIÊN**

**BÀI TOÁN SINH TỪ**

**NGÀNH CÔNG NGHỆ KỸ THUẬT ĐIỆN – ĐIỆN TỬ**

Giáo viên hướng dẫn:

**PGS.TS TRƯƠNG NGỌC SƠN**

Sinh viên: **BẠCH ĐÌNH HIỀN**

MSSV: 18142286

**ĐỖ QUỐC VIỆT**

MSSV: 18142417

TP. HỒ CHÍ MINH – 06/2021

BẢN NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

| Stt | Nội dung thực hiện | Nhận xét |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Nhận xét tổng quát:

………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………

…………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………

**TÓM TẮT**

Trí tuệ nhân tạo hay còn gọi là trí thông minh được thể hiện bằng máy móc, trái ngược với trí thông minh tự nhiên của con người. Thông thường, thuật ngữ "trí tuệ nhân tạo" thường được sử dụng để mô tả các máy móc (hoặc máy tính) có khả năng bắt chước các chức năng "nhận thức" mà con người thường phải liên kết với tâm trí, như "học tập" và "giải quyết vấn đề. Mặc dù được hình thành và phát triển từ thế kỷ 20, song trong những năm gần đây trí tuệ nhân tạo mới phát triển vượt bậc và đạt được những thành tựu to lớn. Thêm một lý do nữa để trí tuệ nhân tạo phát triển mạnh mẽ đến như vậy là vì những năm trở lại đây, nghành công nghiệ vi mạch phát triển, các hệ thống máy tính được mặt tốc độ xử lý và khả năng lưu trữ thông tin, tạo tiền đề to lớn để trí tuệ nhân tạo phát triển.

Xử lý ngôn ngữ tự nhiên (natural language processing) là một nhánh của trí tuệ nhân tạo tập trung vào các ứng dụng trên ngôn ngữ của con người. Với một lĩnh vực tìm năng và đầy triển vọng như vậy. Nên nhóm sinh viên đã quyết định tìm hiểu về chủ đề này.

Bố cục cuốn báo cáo bao gồm 4 chương:

Chương 1: Giới thiệu tổng quát về trí tuệ nhân tạo, xử lý ngôn ngữ tự nhiên. Các mục tiêu, giới hạn đề tài, phương pháp nghiên cứu, đối tượng và phạm vi nghiên cứu, bố cục quyển báo cáo.

Chương 2: Giới thiệu các cơ sở lý thuyết để phục vụ cho bài toán sinh từ. Bao gồm: Xử lý ngôn ngữ tự nhiên (NLP) - Tokenizations and Sequence, Word Embedding, mô hình ngôn ngữ và mạng nơ ron hồi quy (RNN), Long Short-Term Memory (LSTM) và BI-DIRECTIONAL LSTM.

Chương 3: Thiết kế mô hình bài toán sinh từ. Đầu tiên chúng ta xẽ lên ý tưởng thiết kế, đây là bước khá quan trọng để chúng ta có thể mô hình hóa và khái quát những gì chúng ta xẽ làm. Sau đó là chuẩn bị dữ liệu (bộ từ điển), xây dựng mô hình với độ chính xác trên 80% và dự đoán 10 từ tiếp theo.

Chương 4: Kết luận và hướng phát triển. Trong phần này chúng ta xẽ bàn về ưu, nhược điểm của bài toán. Các kiến thức đã được học và áp dụng cho bài toán này. Tiếp đó là hướng phát triển của bài toán xử lý ngôn ngữ tự nhiên (NLP).

Phần phụ lục: Đưa ra toàn bộ Source Code của chương trình.

Cuối cùng, tuy đã rất cố gắng để tìm kiếm thông tin học hỏi và chỉnh sữa nhưng cuốn báo cáo không tránh khỏi những sai sót, nhóm sinh viên mong thầy cô và các bạn bổ xung và đóng góp để cuốn báo cáo này được hoàn thiện hơn. Xin chân thành cảm ơn.

**MỤC LỤC**

[DANH MỤC HÌNH 7](#_heading=h.gjdgxs)

[CÁC TỪ VIẾT TẮT 10](#_heading=h.1fob9te)

[CHƯƠNG 1 GIỚI THIỆU 11](#_heading=h.3znysh7)

[1.1.](#_heading=h.2et92p0) GIỚI THIỆU 11

[1.2.](#_heading=h.tyjcwt) MỤC TIÊU ĐỀ TÀI 11

[1.3.](#_heading=h.3dy6vkm) GIỚI HẠN ĐỀ TÀI 11

[1.4.](#_heading=h.1t3h5sf) PHƯƠNG PHÁP NGHIÊN CỨU 11

[1.5.](#_heading=h.4d34og8) ĐỐI TƯỢNG VÀ PHẠM VI NGHIÊN CỨU 12

[1.6.](#_heading=h.2s8eyo1) BỐ CỤC QUYỂN BÁO CÁO 12

[CHƯƠNG 2 CƠ SỞ LÝ THUYẾT 13](#_heading=h.17dp8vu)

[2.1.](#_heading=h.3rdcrjn) XỬ LÝ NGÔN NGỮ TỰ NHIÊN (NPL) 13

[2.2.](#_heading=h.1ksv4uv) WORD EMBEDDING 17

[2.2.1. WORD EMBEDDING LÀ GÌ? 17](#_heading=h.44sinio)

[2.2.2. TẠI SAO CHÚNG TA LẠI CẦN EMBEDDING? 18](#_heading=h.2jxsxqh)

[2.2.3. VẬY LÀM THẾ NÀO ĐỂ BIỂU DIỄN EMBEDDING? 19](#_heading=h.37m2jsg)

[2.3.](#_heading=h.3whwml4) MÔ HÌNH NGÔN NGỮ VÀ MẠNG NƠ RON HỒI QUY 23

[2.3.1. LANGUAGE MODEL 23](#_heading=h.2bn6wsx)

[2.3.2. RECURRENT NEURAL NETWORK 24](#_heading=h.qsh70q)

[2.3.3. LOSS FUNCTION 26](#_heading=h.3as4poj)

[2.3.4. GIẢI QUYẾT GRADIENT QUÁ NHỎ HOẶC QUÁ LỚN 27](#_heading=h.1pxezwc)

[2.4.](#_heading=h.49x2ik5) LONG SHORT-TERM MEMORY (LSTM) 27

[2.4.1. Ý TƯỞNG CỐT LÕI ĐẰNG SAU LSTM 29](#_heading=h.2p2csry)

[2.4.2. CÁC BƯỚC CỦA LSTM 29](#_heading=h.147n2zr)

[2.5.](#_heading=h.3o7alnk) BI-DIRECTIONAL LSTM 31

[CHƯƠNG 3 THIẾT KẾ MÔ HÌNH BÀI TOÁN SINH TỪ 33](#_heading=h.23ckvvd)

[3.1.](#_heading=h.ihv636) Ý TƯỞNG THIẾT KẾ 33

[3.2.](#_heading=h.32hioqz) CHUẨN BỊ BỘ TỪ ĐIỂN 34

[3.3.](#_heading=h.41mghml) XỬ LÝ DỮ LIỆU, CHIA FEATURES VÀ LABLE 36

[3.4.](#_heading=h.2grqrue) XÂY DỰNG MODEL VỚI ĐỘ CHÍNH XÁC TRÊN 80% 37

[3.5.](#_heading=h.vx1227) DỰ ĐOÁN 10 TỪ TIẾP THEO 39

[CHƯƠNG 4 KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 41](#_heading=h.3fwokq0)

[4.1 KẾT LUẬN 41](#_heading=h.1v1yuxt)

[4.2](#_heading=h.4f1mdlm) HƯỚNG PHÁT TRIỂN 41

[PHỤ LỤC 42](#_heading=h.2u6wntf)

[SOURCE CODE CỦA BÀI TOÁN SINH TỪ: 42](#_heading=h.19c6y18)

[TÀI LIỆU THAM KHẢO 44](#_heading=h.28h4qwu)

**DANH MỤC HÌNH**

Hình 1a và 1b miêu tả vị trí đứng của các từ trong câu

Hình 2. Sắp xếp theo số lượng xuất hiện của từ trong câu có OOV

Hình 3. Biểu diễn dữ liệu

Hình 4. Không gian vecto biểu diễn

Hình 5. Mô hình skip-gram quan tâm đến xác suất có điều kiện sinh ra các từ ngữ cảnh với một từ đích trung tâm cho trước.

Hình 6. Mô hình CBOW quan tâm đến xác suất có điều kiện tạo ra từ đích trung tâm dựa trên các từ ngữ cảnh cho trước

Hình 7. Biểu diễn đo lường xác xuất

Hình 8. Mô hình RNN

Hình 9. Mô hình đầy đủ

Hình 10. Biểu diển sơ đồ của lớp hồi quy

Hình 11a và 11b. Mô hình lớp RNN và lớp LSTM

Hình 12. Mô-đun lặp lại trong RNN tiêu chuẩn chứa một lớp duy nhất

Hình 13. Mô-đun lặp lại trong một LSTM chứa bốn lớp tương tác

Hình 14. Mô tả thông tin trôi theo dòng không đổi

Hình 15. Cấu trúc cổng

Hình 16. Cổng quyết định cho thông tin đi qua hay không

Hình 17. Tạo bản cập nhật cho trạng thái

Hình 18. Cập nhật lại Ct

Hình 19. Quyết định những gì xuất ra thông qua Ot

Hình 20. Mô hình Bi-Birectional LSTM

Hình 21. Từ điển khi load về

Hình 22. Kết quả sau khi training

Hình 23. Kết quả sau sinh ra 10 tiếp theo**DANH MỤC BẢNG**

Bảng 1. Mô hình hóa các từ thành vecto

**CÁC TỪ VIẾT TẮT**

NLP: Natural Language Processing

CBOW: Continuous Bag-of-Words

RNN: Recurrent Neural Network

CNN: Convolutional Neural Networks

LSTM: Long Short-Term Memory

BI-DIRECTIONAL LSTM: BI-DIRECTIONAL Long Short-Term Memory

AI: Artificial Intelligence

# CHƯƠNG 1 Giới thiệu

## Giới thiệu

Xử lý ngôn ngữ tự nhiên ra đời từ những năm 1940, với rất nhiều công trình nghiên cứu theo hai hướng chính là: 1) ô-tô-mát (automaton) và các mô hình xác suất (probabilistic models) vào những năm 1950; 2) các phương pháp dựa trên ký hiệu (symbolic) và các phương pháp ngẫu nhiên (stochastic) vào những năm 1970. Giai đoạn tiếp theo (1970-1983) chứng kiến sự bùng nổ trong nghiên cứu về xử lý tiếng nói và ngôn ngữ. Ngày nay với sự phát triển nhanh chóng, học máy (machine learning) đã trở thành trung tâm của phần lớn các lĩnh vực thuộc khoa học máy tính, bao gồm xử lý ảnh và thị giác máy tính (computer vision), tin sinh học (bioinformatics), các hệ tư vấn (recommender systems), kỹ nghệ phần mềm, và cả xử lý ngôn ngữ tự nhiên.

Xử lý ngôn ngữ tự nhiên (NLP) là một nhánh của Trí tuệ nhân tạo, tập trung vào việc nghiên cứu sự tương tác giữa máy tính và ngôn ngữ tự nhiên của con người. Mục tiêu của lĩnh vực này là giúp máy tính hiểu và thực hiện hiệu quả những nhiệm vụ liên quan đến ngôn ngữ của con người như: tương tác giữa người và máy, cải thiện hiệu quả giao tiếp giữa con người với con người, hoặc đơn giản là nâng cao hiệu quả xử lý văn bản và lời nói.

## mục tiêu đề tài

Với đề tài này, chúng em hướng đến thiết kế một hệ thống có thể xử lý, nhận diện, phân biệt những nguồn dữ liệu về đối tượng dạng văn bản, ngôn ngữ… và từ đó có thể thực hiện các chức năng mô hình hóa ngôn ngữ (dự đoán từ tiếp theo) dựa trên một tập dữ liệu có sản.

## Giới hạn đề tài

Đề tài được thực hiện xuyên suốt trong quá trình học tập. Kiến thức nghiên cứu được giới hạn ở mức cơ bản cho sinh viên.

## Phương pháp nghiên cứu

Có rất nhiều hướng tiếp cận bài toán phân loại văn bản đã được nghiên cứu như: tiếp cận bài toán phân loại dựa trên lý thuyết đồ thị, cách tiếp cận thống kê… Tuy nhiên, tất cả các phương pháp trên đều dựa vào các phương pháp chung là học máy đó là: học có giám sát, không có giám sát và học tăng cường.

Vấn đề phân loại văn bản theo phương pháp thống kê dựa trên kiểu học có giám sát được mô tả gồm 2 giai đoạn: giai đoạn huấn luyện và giai đoạn phân lớp.

Dựa vào các kiến thức đã được học về các mạng như RNN, LSTM và nghiên cứu thêm các mạng khác như BI-DIRECTIONAL LSTM, mô hình xử lý ngôn ngữ tự nhiên,…

## Đối tượng và phạm vi nghiên cứu

Đối tượng của nghiên cứu các loại dữ liệu như văn bản, excel hoặc các bình luận…

## Bố cục quyển báo cáo

Bố cục của bài báo cáo gồm các phần:

* **CHƯƠNG 1: GIỚI THIỆU**
* **CHƯƠNG 2: CƠ SỞ LÝ THUYẾT**
* **CHƯƠNG 3: THIẾT KẾ MÔ HÌNH BÀI TOÁN SINH TỪ**
* **CHƯƠNG 4: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN**

Qua các chương ta sẽ tìm hiểu thế nào là xử lý ngôn ngữ tự nhiên (NLP), các yếu tố để cấu thành cũng như tìm hiểu cách tạo nên một chương trình xử lý ngôn ngữ tự nhiên. Đánh giá lại kết quả của chương trình đồng thời nêu kên hướng phát triển của hệ thống.

# chương 2 CƠ SỞ LÝ THUYẾT

Chương này sẽ giới thiệu về các lý thuyết liên quan tới tất cả các vấn đề cần giải quyết trong đề tài về xử lý ngôn ngữ tự nhiên - Tokenizations and Sequence, Word Embedding, Language model, RNN, LSTM, BI-DIRECTIONAL LSTM. Sử dụng các mô hình và cơ sở lý thuyết để vận dụng giải quyết bài toán sinh từ ở chương 3.

## Xử lý ngôn ngữ tự nhiên (NPL) - Tokenizations and Sequence

Dữ liệu của ngôn nhữ tự nhiên có một phần khác biệt so với dữ liệu trong thị giác máy tính. Trong thị giác máy tính thì dữ liệu nằm trong luôn trong từng Pixel của từng ảnh một, chúng ta xẽ thao tác luôn với Pixel đó. Nhưng trong xử lý ngôn ngữ tự nhiên chúng ta cần phải mô hình hóa được các từ và mối quan hệ giữa các từ với nhau. Trong phần này chúng ta cùng tìm hiểu từ một câu, chúng ta phải tách từ ra và sau khi tách từ ra chúng ta dùng từ đó để chuyển thành số mới đưa vào mạng noron.

* Encode words

Sentences = [

‘I love Vietnam’,

‘Viennamese people are pretty friendly’,

‘My mom loves cooking’,

‘I am Vietnamese’

| **0** | **0** | **2** | **4** | **5** |
| --- | --- | --- | --- | --- |
| **3** | 6 | 7 | 8 | 9 |
| **0** | 10 | 11 | 12 | 13 |
| **0** | 0 | 2 | 14 | 3 |

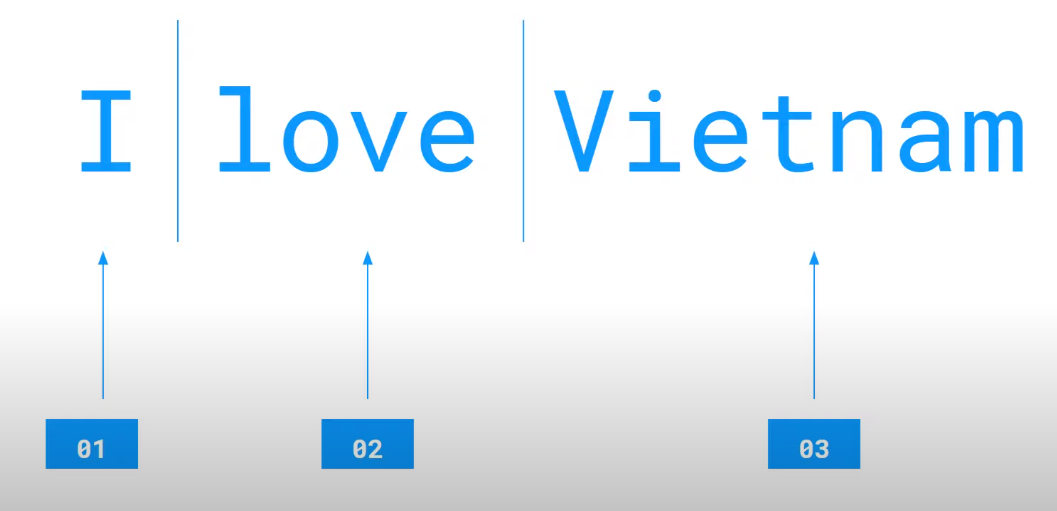
]



*Bảng 1. Mô hình hóa các từ thành vecto*

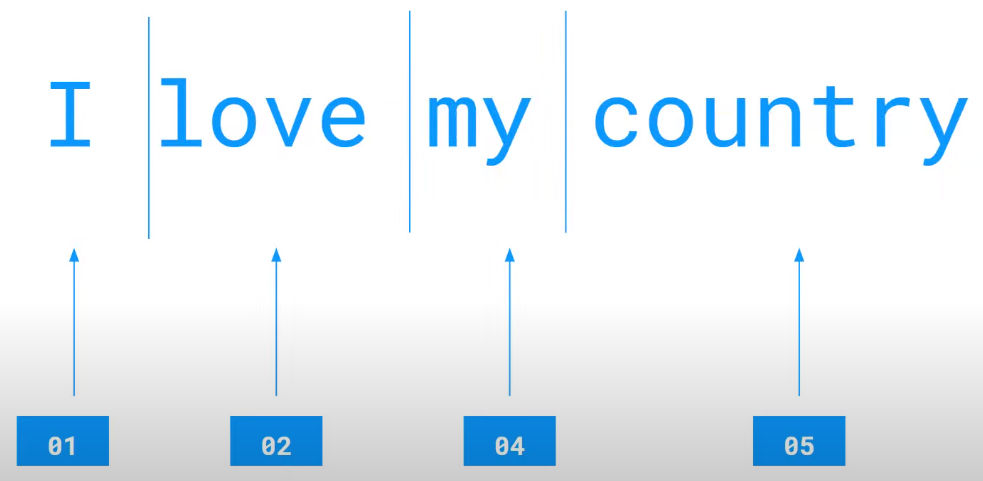
Chúng ta có 4 câu trong sentences. Từ những câu trên chúng ta mô hình hóa chúng thành một ma trận. Mỗi câu xẽ tương ứng với một vecto. Ví dụ ‘I love Vietnam’ xẽ tương ứng với vecto [0 0 2 4 5], ‘Viennamese people are pretty friendly’ tương ứng với vecto [3 6 7 8 9],…

Tách từ được hiểu như là chúng ta chia một câu, đoạn văn, toàn bộ chữ trong văn bản thành nhiều phần nhỏ ta gọi là units, nó có thể là từ hoặc có thể là cụm từ. Mỗi một unit gọi là một Token, một đơn vị nhỏ nhất.



*Hình 1a*

Một cách đơn giản chúng ta có thể dựa vào vị trí của từ đó. Ví dụ “I love Vietnam”, ‘I’ là giá đầu tiên 01, ‘love’ là 02, và ‘Vietnam’ là 03.

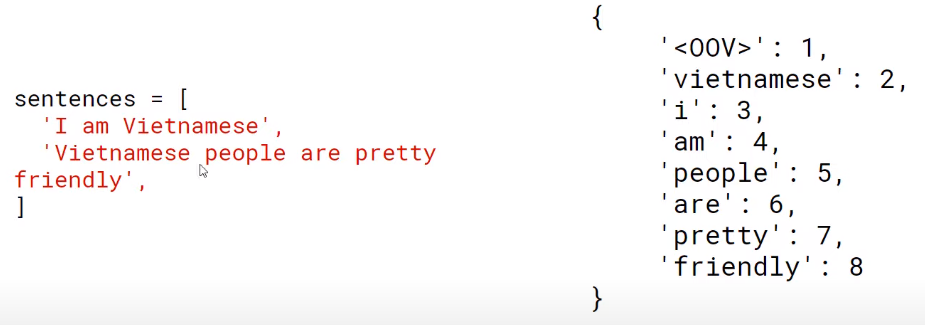


*Hình 1b*

*Hình 1a và 1b miêu tả vị trí đứng của các từ trong câu*

Ví dụ xuất hiện những từ mới ‘I’ và ‘love’ đã xuất hiện rồi thì giá trị là 01 và 02, ‘my’, ‘country’ chưa xuất hiện thì 04 và 05. Vì chữ ‘Vietnam’ đã là 03 rồi nên phải tăng lên.

Nhưng mà trong một câu xuất hiện nhiều từ giống nhau, mục tiêu chúng ta phải chuyển thành một Dictionary. Chúng ta xẽ sắp xếp theo số lượng xuất hiện của từ đó trong câu.



*Hình 2. Sắp xếp theo số lượng xuất hiện của từ trong câu có OOV*

Ví dụ trong câu từ ‘Vietnamese’ xuất hiện hai lần (nhiều nhất) là 2, vị trí trên cùng là <OOV> không nằm trong từ điển là 1. Tiếp đến ‘i’, ‘am’, ‘people’, ‘are’, ‘pretty’, ‘friendly’ số lượng xuất hiện bằng nhau thì chúng ta sắp xếp theo thứ tự xuất hiện của nó.

* **Ví dụ cách chúng ta xây dựng nên từ điển đơn giản:**

Bước đầu tiên chúng ta phải gọi thư viện Keras để dễ dàng cho ví dụ này.



Định nghĩa câu. Đây là lúc chúng ta tự tạ cho mình một từ điển đơn giản theo bằng cách thực hiện như sau.



Chuyển thành một từ điển giới hạn số lượng từ độc nhất là 100 (num\_words=100) và đặt những từ không thuộc từ điển của chúng ta (oov\_token= "<OOV>").



Build từ điển từ những câu trong sentences và in ra xác suất dựa theo số lần xuất hiện và thứ tự xuất hiện của từ.



Kết quả:

{'<OOV>': 1,

'am': 15,

'are': 6,

'bach': 9,

'dinh': 10,

'do': 12,

'friendly': 8,

'hien': 11,

'i': 2,

'love': 4,

'pretty': 7,

'quoc': 13,

'viet': 14,

'vietnam': 5,

'vietnamese': 3}

Nhận xét kết quả:

* ‘i’ và ‘vietnamese’ xuất hiện nhiều nhất và theo thứ tự xuất hiện thì ‘i’ là 2 và ‘vietnamese’ là 3, OOV sẽ luôn là 1. Các từ còn lại theo thứ tự 4, 5, 6, …, 15.
* Đây là cách chúng ta xây dựng từ điển. Sau khi xây dựng xong từ điển rồi, khi có một câu mới thì chúng ta dựa vào từ điển này để tra ra những giá trị của từ.
* **Ví dụ có câu mới:**

Trong phần này chúng ta có 2 câu mới. Nhiệm vụ của chúng ta là đưa các câu mới này vào trong tập từ điển. Và sau đó tính xác suất các từ suất hiện trong từ điển.



Đưa câu mới vào từ điển và in kết quả ra màn hình



**Kết quả:** [[2, 4, 1], [2, 1, 1, 1]]

Ta thấy từ ‘dog’, ‘live’, ‘in’, ‘HCM’ chưa có trong từ điển nên nó có giá trị là 1. Nhưng thực tế hai câu có độ dài khác nhau, để dể dàng chúng ta phải cân bằng độ dài của câu này dựa trên độ dài của câu kia. Sử dụng padding để thực hiện điều này để giúp cho việc training của chúng ta dể dàng hơn.



Kết quả: array([[0, 2, 4, 1],

[2, 1, 1, 1]], dtype=int32)

Câu đầu tiên đã thêm một phần tử 0 vào, do đó độ dài hai câu đã bằng nhau, giúp dể dàng cho việc training.

* 1. Word Embedding
     1. **Word Embedding là gì?**

Ngôn ngữ tự nhiên là một hệ thống phức tạp mà con người sử dụng để diễn đạt ngữ nghĩa. Trong hệ thống này, từ là đơn vị cơ bản của ngữ nghĩa. Như tên gọi của nó, một vector từ (*word vector*) là một vector được sử dụng để biểu diễn một từ. Vector từ cũng có thể được xem là vector đặc trưng của một từ. Kỹ thuật ánh xạ từ ngữ sang vector số thực còn được gọi là kỹ thuật embedding từ (*word embedding*). Trong vài năm gần đây, embedding từ dần trở thành kiến thức cơ bản trong xử lý ngôn ngữ tự nhiên.

* + 1. **Tại sao chúng ta lại cần Embedding?**

Chúng ta thử so sánh với một cách biểu diễn khác mà chúng ta thường dùng trong các bài toán multi-label, multi-task là one-hot encoding. Nếu sử dụng *one-hot encoding*, dữ liệu mà chúng ta biểu diễn sẽ có dạng như sau:

Table

Description automatically generated

*Hình 3. Biểu diễn dữ liệu*

Nhìn vào bảng bên trên, ta thấy có 3 vấn đề khi ta biểu diễn dữ liệu dạng text dưới dạng one-hot:

* Chi phí tính toán lớn:
* Nếu data có 100 từ, độ dài của vector one-hot là 100. Nếu data có 10000 từ, độ dài của vector one-hot là 10000.
* Tuy nhiên, để mô hình có độ khái quát cao thì trong thực tế dữ liệu có thể lên đến hàng triệu từ, lúc đó độ dài vector one-hot sẽ phình to gây khó khăn cho việc tính toán, lưu trữ.
* Mang ít giá trị thông tin:
* Các vector hầu như toàn số 0. Và các bạn có thể thấy, đối với dữ liệu dạng text thì giá trị chứa trong các pixel (nếu input dạng ảnh) hay các dạng khác là rất ít. Nó chủ yếu nằm trong vị trí tương đối giữa các từ với nhau và quan hệ về mặt ngữ nghĩa. Tuy nhiên, one-hot vector không thể biểu diễn điều đó vì nó chỉ đánh index theo thứ tự từ điển đầu vào chứ không phải vị trí các từ trong một context cụ thể.
* Để khắc phục điều đó, trong mô hình thường dùng một lớp RNN hoặc LSTM để nó có thể trích xuất được thông tin về vị trí. Có một cách khác như trong mô hình transformer, được bỏ hoàn toàn lớp word embeddig hay RNN và thêm vào đó lớp positional encoding và self-attention.
* **Độ khái quát yếu:**
* **Ví dụ ta có ba từ cùng chỉ *người mẹ:* mẹ, má, bầm. Tuy nhiên, từ bầm tương đối hiếm gặp trong tiếng Việt. Khi biểu diễn bằng one-hot encoding, khi đưa vào model train thì từ bầm mặc dù cùng nghĩa so với hai từ kia nhưng lại bị phân vào class khác nhau do cách biểu diễn khác nhau.**
* **Còn nếu dùng word embedding, do biểu diễn được cả thông tin về vị trí, ngữ nghĩa nên từ bầm sẽ có vị trí gần với hai từ kia. Đúng như mục đích embedding của mình.**
  + 1. **Các loại Word Embedding**

***Word Embedding*** được phân chủ yếu thành 2 loại:

* Frequency-based embedding.
* Prediction-based embedding.
  + - 1. **Frequency-based embedding**

Đúng như tên gọi của nó, Frequency-based Embedding dựa vào tần số xuất hiện của các từ để tạo ra các vector từ, trong đó có 3 loại phổ biến nhất:

* Count Vector: Count Vector là dạng đơn giản nhất của Frequencey-

based Embedding, giả sử ta có D documents d1, d2, … dD và N là độ dài của từ điển, vector biểu diễn của một từ là một vector số nguyên và có độ dài là D, ở đó phần tử tại vị trí i chính là tần số của từ đó xuất hiện trong document di. Trong một số trường hợp, ta có thể lượt bớt các từ có tần số xuất hiện thấp hoặc thay đổi mục nhập của vector (thay vì tần số có thể thay bằng một giá trị nhị phân biểu thị sự xuất hiện của từ) tùy vào mục đích cụ thể.

* tf-idf Vector: Khác với Count Vector chỉ xét đến tần số xuất hiện của từ

trong một document, tf-idf Vector quan tâm cả tần số xuất hiện của từ trong toàn bộ tập dữ liệu, chính do đặc điểm này mà tf-idf Vector có tính phân loại cao hơn so với Count Vector. tf-idf (Term Frequency-Inverse Document Frequency) Vector là một vector số thực cũng có độ dài D với D là số văn bản, nó được tính bằng tích của 2 phần bao gồm tf và idf, công thức của mỗi phần tử của vector được tính như sau:

Trong đó:

i: 1…D

ni: tần số xuất hiện của từ trong văn bản i

Ni: tổng số từ trong văn bản i

Trong đó:

D: tổng số documents trong tập dữ liệu

d: số lượng documents có sự xuất hiện của từ

tf-idf Vector có tính phân loại cao hơn so với Count Vector chính là bởi nó

được điều chỉnh bởi trọng số idf, dựa trên công thức của nó ta có thể hiểu rằng nếu từ xuất hiện ở càng nhiều văn bản (tính phân loại thấp) thì giá trị của nó càng nhỏ, từ đó kết quả cuối cùng sẽ bị nhỏ theo.

Tuy nhiên, nhược điểm của cả hai phương pháp trên chính là việc nó chỉ chú

trọng đến tần số xuất hiện của một từ, dẫn tới nó hầu như không mang ý nghĩa gì về mặt ngữ cảnh, Co-occurrence Matrix phần nào giải quyết vấn đề đó.

* Co-occurrence Matrix: Co-occurrence Matrix có ưu điểm là bảo tồn mối

quan hệ ngữ nghĩa giữa các từ, được xây dựng dựa trên số lần xuất hiện của các cặp từ trong Context Window. Một Context Window được xác định bởi kích thước và hướng của nó. Hình dưới đây là một ví dụ của Context Window:

Diagram

Description automatically generated

Hình 4. Ví dụ về Context Window

Thông thường, Co-occurrence Matrix là một ma trận vuông đối xứng, mỗi

hàng hoặc mỗi cột sẽ chính là vector biểu thị của từ tương ứng. Tiếp tục ví dụ trên ta sẽ có ma trận Co-occurrence Matrix:

|  | John | is | not | fat | thin |
| --- | --- | --- | --- | --- | --- |
| John | 0 | 2 | 0 | 1 | 0 |
| is | 2 | 0 | 1 | 0 | 1 |
| not | 0 | 1 | 0 | 1 | 0 |
| fat | 1 | 0 | 1 | 0 | 0 |
| thin | 0 | 1 | 0 | 0 | 0 |

Tuy nhiên, trong thực tế, do số lượng từ vựng nhiều, ta thường chọn cách bỏ

đi một số từ không cần thiết (ví dụ như các stopwords) hoặc sử dụng phân tách SVD (Singular Value Decomposition) để giảm kích thước của vector từ nhằm giúp cho biểu diễn của từ được rõ ràng hơn đồng thời tiết kiệm bộ nhớ dùng để lưu trữ Co-occurrence Matrix (do các Co-occurrence Matrix có kích thước rất lớn).

GloVe là một trong những phương pháp mới để xây dựng vectơ từ (giới thiệu vào năm 2014), nó được xây dựng dựa trên ma trận đồng xảy ra (Co-occurrence Matrix). GloVe dựa trên ý tưởng tính tỉ lệ xác xuất:

Với P(k|i) là xác suất từ k xuất hiện trong ngữ cảnh của từ i, tương tự vậy với P(k|j).



Công thức tính của P(k|i):



Xik: là số lần xuất hiện của từ k trong ngữ cảnh của từ i (và ngược lại).

Xi: là số lần xuất hiện của từ i trong ngữ cảnh của toàn bộ các từ còn lại ngoại trừ i.

Ý tưởng chính của GloVe được tính dựa trên độ tương tự ngữ nghĩa giữa hai từ i, j được xác định thông qua độ tương tự ngữ nghĩa giữa từ k với mỗi từ i, j, những từ k có tính xác định ngữ nghĩa tốt chính là những từ làm cho giá trị được tính từ công thức (1) nằm trong khoảng [0 ,1]

Ví dụ, nếu từ i là “table”, từ j là “cat” và từ k là “chair” thì công thức (1) sẽ cho giá trị tiệm cận đến 1 do “chair” có nghĩa gần hơn với “table” hơn là “cat”, ở trường hợp khác, nếu ta thay từ k là “ice cream” thì giá trị công thức (1) sẽ xấp xỉ bằng 0 do “ice cream” hầu như chẳng lên quan gì tới “table” và “cat”.

Dựa trên tầm quan trọng của công thức (1), GloVe khởi đầu bằng việc là nó sẽ tìm một hàm F sao cho nó ánh xạ từ các vec-tơ từ trong vùng không gian V sang một giá trị tỉ lệ tính theo công thức (1). Việc tìm F không đơn giản, tuy nhiên, sau nhiều bước đơn giản hóa cũng như tối ưu, ta có thể đưa nó về bài toán hồi quy với việc tính hàm chi phí tối thiểu (minimum cost function) sau:



Wi, Wj là các vector từ.  
bi, bj là các bias tương ứng (được thêm vào ở các bước đơn giản hóa và tối ưu).

xij: mục nhập tương ứng với cặp từ i, j trong ma trận đồng xảy ra

Hàm f(x) được gọi là hàm trọng số (weighting function), được thêm vào để giảm bớt sự ảnh hưởng của các cặp từ xuất hiện quá thường xuyên, hàm này thỏa mãn 3 tính chất:

* Có giới hạn tại 0.
* Là hàm không giảm.
* Có giá trị nhỏ khi x rất lớn.

Thực tế, có nhiều hàm số thỏa các tính chất trên, nhưng ta sẽ lựa chọn hàm số sau:



Với α=3/4

Việc thực hiện tính hàm chi phí tối thiểu J để tìm ra các vec-tơ từ thể được thực hiện bằng nhiều cách, trong đó cách tiêu chuẩn nhất là sử dụng tìm cực tiểu hàm số theo thuật toán Gradient Descent.

* + - 1. **Predictive methods (Word2Vec)**

Khác so với Count-based method, Predictive method tính toán sự tương đồng ngữ nghĩa giữa các từ để dự đoán từ tiếp theo bằng cách đưa qua một mạng neural network có một hoặc vài layer dựa trên input là các từ xung quanh (context word). Một context word có thể là một hoặc nhiều từ. Ví dụ cũng với hai câu ở trên, ban đầu hai từ cơm và cá có thể được khởi tạo ở khá xa nhau nhưng để tối thiểu loss giữa hai từ đó và context word ("Mèo" và "ăn") thì vị trí của hai từ cơm và cá trong không gian vector phải gần nhau. Có 2 phương pháp *predictive method* phổ biến đó chính là:

* Continuous Bag-of-Words (CBOW)
* Skip-gram
  + - 1. **Skip-gram**

Mô hình skip-gam giả định rằng một từ có thể được sử dụng để sinh ra các từ xung quanh nó trong một chuỗi văn bản. Ví dụ, giả sử chuỗi văn bản là “the”, “man”, “loves”, “his” và “son”. Ta sử dụng “loves” làm từ đích trung tâm và đặt kích thước cửa sổ ngữ cảnh bằng 2. Như mô tả ở dưới, với từ đích trung tâm “loves”, mô hình skip-gram quan tâm đến xác suất có điều kiện sinh ra các từ ngữ cảnh (“the”, “man”, “his” và “son”) nằm trong khoảng cách không quá 2 từ:

Ta giả định rằng, với từ đích trung tâm cho trước, các từ ngữ cảnh được sinh ra độc lập với nhau. Trong trường hợp này, công thức trên có thể được viết lại thànhDiagram

Description automatically generated with medium confidence

*Hình 5. Mô hình skip-gram quan tâm đến xác suất có điều kiện sinh ra các từ ngữ cảnh với một từ đích trung tâm cho trước*

Trong mô hình skip-gam, mỗi từ được biểu diễn bằng hai vector d-chiều để tính xác suất có điều kiện. Giả sử chỉ số của một từ trong từ điển là ii, vector của từ được biểu diễn là vi ∈  khi từ này là từ đích trung tâm và là ui ∈  khi từ này là một từ ngữ cảnh. Gọi c và o lần lượt là chỉ số của từ đích trung tâm wc và từ ngữ cảnh wo trong từ điển. Có thể thu được xác suất có điều kiện sinh ra từ ngữ cảnh cho một từ đích trung tâm cho trước bằng phép toán softmax trên tích vô hướng của vector:

Trong đó, tập chỉ số trong bộ từ vựng là V= {0, 1, …, |V|−1}. Giả sử trong một chuỗi văn bản có độ dài T, từ tại bước thời gian t được ký hiệu là . Giả sử rằng các từ ngữ cảnh được sinh độc lập với từ trung tâm cho trước. Khi kích thước cửa sổ ngữ cảnh là m, hàm hợp lý (*likelihood*) của mô hình skip-gam là xác suất kết hợp sinh ra tất cả các từ ngữ cảnh với bất kỳ từ trung tâm cho trước nào

Ở đây, bất kỳ bước thời gian nào nhỏ hơn 1 hoặc lớn hơn T đều có thể được bỏ qua.

* + - 1. **Continuous Bag-of-Words (CBOW)**

Mô hình túi từ liên tục (*Continuous bag of words* - CBOW) tương tự như mô hình skip-gram. Khác biệt lớn nhất là mô hình CBOW giả định rằng từ đích trung tâm được tạo ra dựa trên các từ ngữ cảnh phía trước và sau nó trong một chuỗi văn bản. Với cùng một chuỗi văn bản gồm các từ “the”, “man”, “loves”, “his” và “son”, trong đó “love” là từ đích trung tâm, với kíc q qh thước cửa sổ ngữ cảnh bằng 2, mô hình CBOW quan tâm đến xác suất có điều kiện để sinh ra từ đích “love” dựa trên các từ ngữ cảnh “the”, “man”, “his” và “son” như sau:

A picture containing text, clipart

Description automatically generated

*Hình 6. Mô hình CBOW quan tâm đến xác suất có điều kiện tạo ra từ đích trung tâm dựa trên các từ ngữ cảnh cho trước*

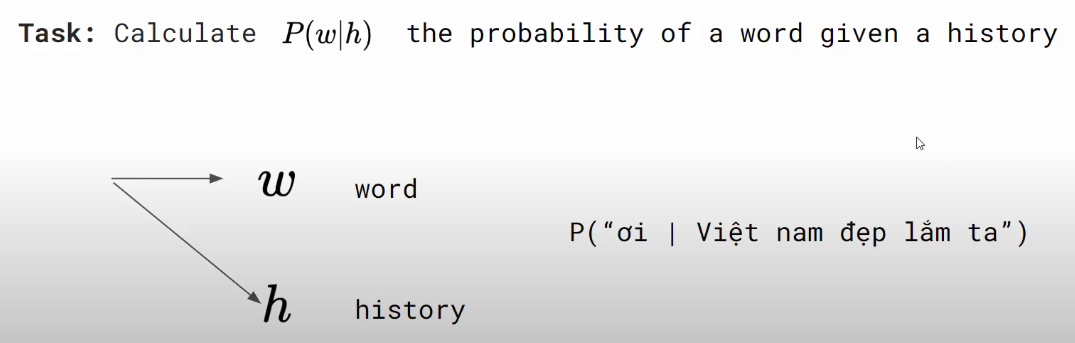
Vì có quá nhiều từ ngữ cảnh trong mô hình CBOW, ta sẽ lấy trung bình các vector từ của chúng và sau đó sử dụng phương pháp tương tự như trong mô hình skip-gram để tính xác suất có điều kiện. Giả sử vi ∈ và ui ∈ là vector từ ngữ cảnh và vector từ đích trung tâm của từ có chỉ số i trong từ điển (lưu ý rằng các ký hiệu này ngược với các ký hiệu trong mô hình skip-gram). Gọi c là chỉ số của từ đích trung tâm wc, và o1, …, o2m là chỉ số các từ ngữ cảnh wo1, …, wo2m trong từ điển. Do đó, xác suất có điều kiện sinh ra từ đích trung tâm dựa vào các từ ngữ cảnh cho trước là:

Để rút gọn, ký hiệu Wo= {wo1, …, wo2m} và o= (vo1+…, +vo2m)/(2m). Phương trình trên được đơn giản hóa thành

Cho một chuỗi văn bản có độ dài T, ta giả định rằng từ xuất hiện tại bước thời gian t là w(t), và kích thước của cửa sổ ngữ cảnh là mm. Hàm hợp lý của mô hình CBOW là xác suất sinh ra bất kỳ từ đích trung tâm nào dựa vào những từ ngữ cảnh.

* 1. **Mô hình ngôn ngữ và mạng nơ ron hồi quy (language model and recurrent neural network)**
     1. **Language model**

Mục đích của Lanuage Model là đo lường xác xuất xảy ra của một từ khi cho những từ phía trước.

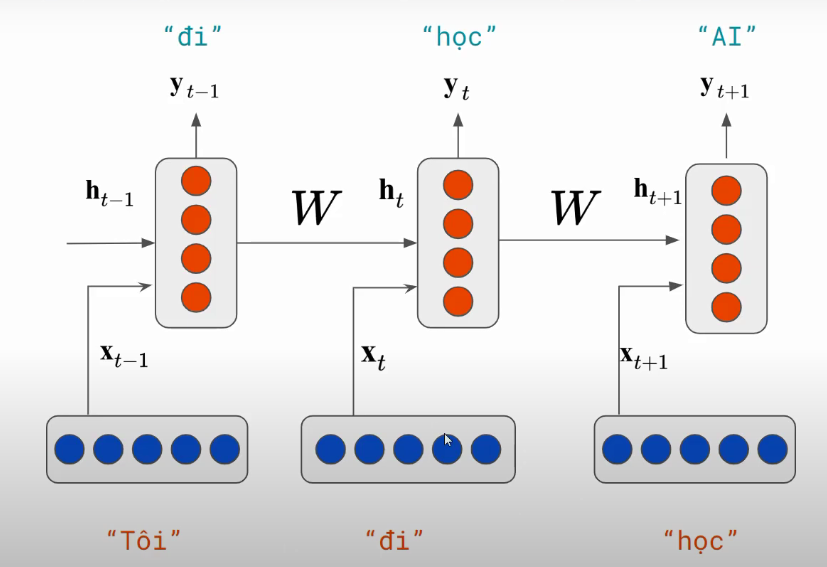


*Hình 7. Biểu diễn đo lường xác xuất*

Tính xác xuất của w khi cho h. Ví dụ cho “Việt nam đẹp lắm ta” thì chúng ta phải đo lường xác xuất xảy từ “ơi” là bao nhiêu.

* + 1. **Recurrent neural network**

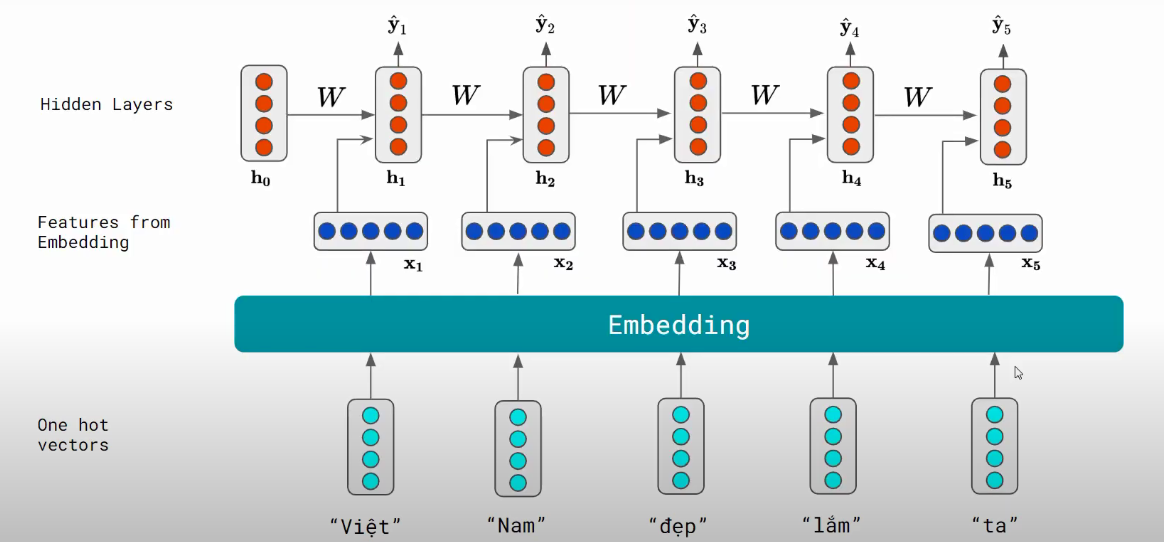
**Mô hình:**



*Hình 8. Mô hình RNN*

Đầu tiên dữ liệu của chúng ta xẽ theo thời gian. xt-1, xt, xt+1 là ba từ liên tiếp nhau, xt-1 là thời gian phía trước của xt và xt+1 là thời gian phía sau của xt. Người ta tạo ra vecto h để học lịch sử của các từ. Đầu tiên khởi tạo bằng 0, sau đó học giá trị của từ “tôi” và lưu lại. Sau khi có giá trị của từ “tôi” rồi thì xẽ output ra xác xuất của tất cả các từ trong từ điển để xem từ nào có khả năng xuất hiện sau từ “tôi” là cao nhất. Sau khi ht-1 lưu lại được thông tin của xt-1 xẽ được đưa qua ht và ht xẽ lưu thông tin của ht-1 và xt (xt lúc này là từ “đi”), mà ht-1 lại lưu thông tin của từ tôi. Nghĩa là ht xẽ lưu thông tin của từ “tôi” và từ “đi” và xẽ cho ra xác xuất trên toàn bộ từ điển từ nào có xác xuất có thể đứng sau từ “tôi” và từ “đi” là cao nhất ht-1, trong trường hợp này xẽ là từ “học”. Tương tự như vậy ht+1 xẽ lưu lại thông tin của xt+1, xt, xt-1. Do đó, ht+1 xẽ lưu thông tin của từ “tôi”, “đi”, “học” và xác xuất lần này trong từ điển có thể là từ “Ai”. Như vậy RNN đã hoàn thành nhiệm vụ mô hình Language Model.

**Quy trình xây dựng mô hình đầy đủ:**



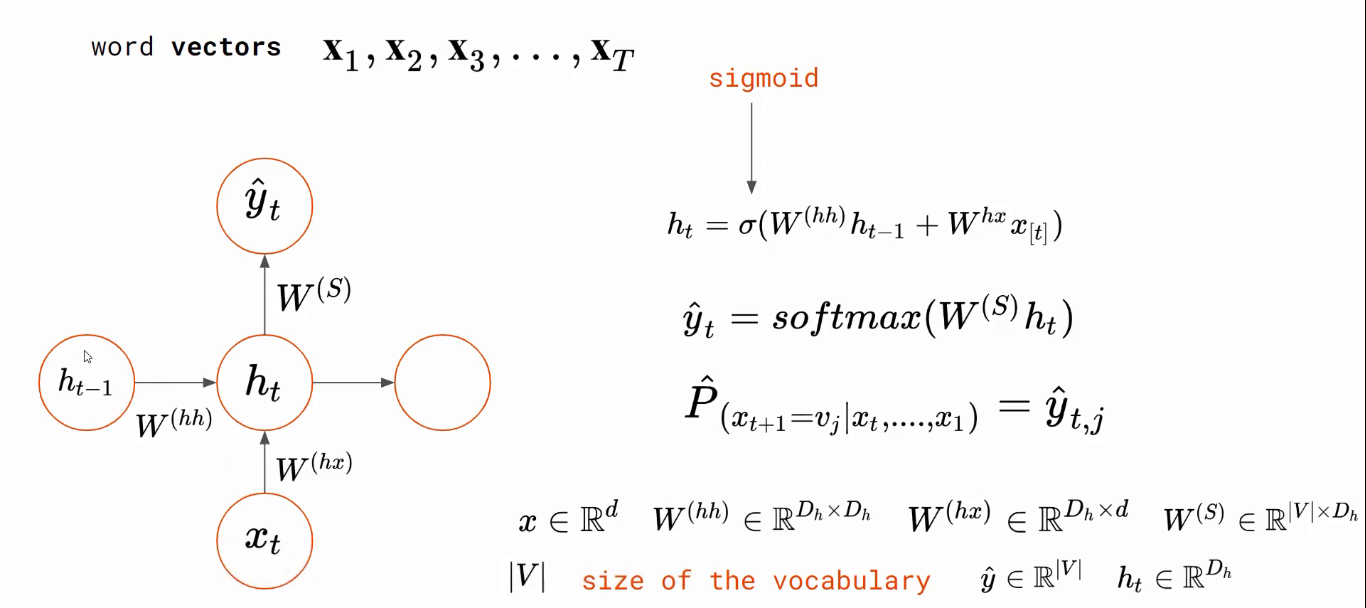
*Hình 9. Mô hình đầy đủ*

= P(x2| “Việt”)

= P(x6| “Việt Nam đẹp lắm ta”)

Đầu tiên chúng ta có những vecto one-hot của “Việt”, “Nam”, “đẹp”, “lắm”, “ta” đưa qua một Embedding lấy được vecto Embedding của từng từ đó. Sau đó đưa vào mạng. Từ “Việt” (x1) kết hợp với h0 ban đầu (giá trị ban đầu bằng 0) thì xẽ ra h1, h1 xẽ lưu trữ thông tin của từ “Việt” và xẽ output ra một xác xuất trên toàn bộ tập từ điển xem là khả năng từ nào đi sau từ “Việt” là cao nhất, trong trường hợp này xẽ là từ “Nam” là cao nhất. Sau đó h1 được đưa sang bên kết hợp với từ “Nam”. Tương tự như trên h2 xẽ lưu thông tin của “Việt”, “Nam”, tương tự như vậy h5 xẽ lưu thông tin của “Viet”, “Nam”, “đẹp”, “lắm”, “ta”, “ơi” và xẽ output ra . Lưu ý tất cả các bước sử dụng chung một W.

Mỗi bước như trên là một Step và có mô hình như sau:



*Hình 10. Biểu diển sơ đồ của lớp hồi quy*

Cho ht-1 đằng trước vào kết hợp với từ hiện tại ra ht và ht có khả năng output ra xác suất trên toàn tập từ điển từ nào có khả năng xuất hiện đằng sau từ có lịch sử là ht-1. Đầu tiên chúng ta có các vecto x1, x2, x3, …, xt chính là những từ. Chiều của W(hh) chính bằng chiều của hai layer ht-1 và ht, tương tự với W(hx) và W(S). Xác xuất của từ nào trên tập từ điển chính là ouput của từng cell.

* ht =

ht xẽ bằng sigmoid của

* = *softmax*()

bằng *softmax của*

* *= t,j*
* x Rd có nghĩa là khi đưa các vecto qua Rd (chiều của Embedding) từ của chúng ta có chiều theo Embedding.
* W(hh) nghĩa là W(hh) có chiều theo ht và ht-1
* W(hx)  , W(hx) có chiều theo ht và xt
* W(S) là chiều của từ điển nhân với ht
* , chiều của cũng chính là chiều của từ điển
* ht

Trong đó V là cở của từ điển

* + 1. **Loss function**

Ngõ ra ở mỗi thời điểm là hàm Softmax của lớp V, trong đó V là kích thướt của tập từ điển. Sai số tương ứng cho mỗi thời điểm là giá trị âm của logarit sai số trên tất cả tập từ điển. Hàm lỗi tại mỗi điểm là:

C(t) (W) = -

Sai số trên tổng tất cả các bước được tính bằng tổng sai số trên mỗi bước. Giá trị trung bình của C(t) (W) được sử dụng trong quá trình giảm độ dốc ngẩu nhiên. Tổng của sai số trên tất cả các bước được tính như sau:

C = -

* **Ưu điểm:**
* W được sử dụng ở mọi timestep. Nên hoạt động tốt trên tất cả chiều

dài của các câu. Khác với mạng CNN, ví dụ như khi ảnh chúng ta lớn lên thì W cũng xẽ lớn lên.

* Vì trong RNN thì W phụ thuộc vào h mà h (định nhĩa ban đầu) nên cố định. Do đó, W cũng không thay đổi và hoạt động được với tất cả chiều dài các câu.
* **Nhược điểm:**
* Muốn tính ht chúng ta phải tính xong ht-1, việc tính toán theo tuần tự chứ

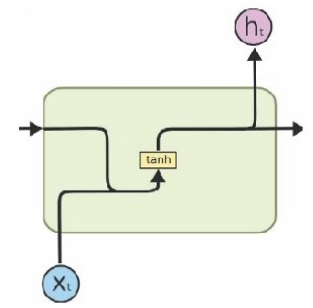
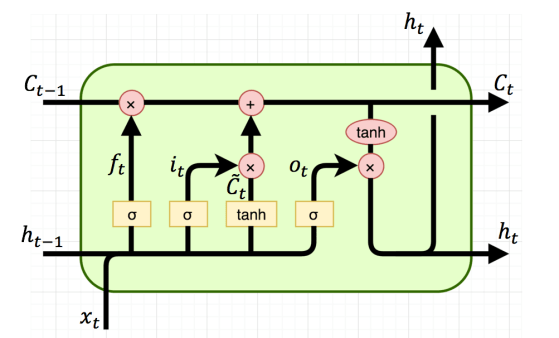
không thể song song được như CNN. Do đó việc training rất chậm. Nếu độ dài câu quá dài thì xẽ ra hiện tượng Gradient quá nhỏ (sắp xỉ 0), hoặc quá lớn.

* + 1. **Giải quyết hiện tượng Gradient quá nhỏ hoặc quá lớn**

Sử dụng phương pháp kẹp được thực hiện đơn giản bằng cách khống chế giá trị các gradient trong một ngưỡng nhất định, không cho nó vượt quá giới hạn. Nếu giá trị của gradient vượt qua ngưỡng xác định trước, độ lớn của các vecto gradient xẽ được thiết lập về mức ngưỡng trong khi chiều của gradient là không đổi. Trong quá trình lan truyền ngược tại thời điểm t, nếu gradient của hàm lỗi J đối với vector w vượt mức ngưỡng k thì gradient g sẽ được sử dụng cho quá trình lan truyền để cập nhật trọng số. Trong đó, g được tính như sau:

* 1. **Long Short-Term Memory (LSTM)**

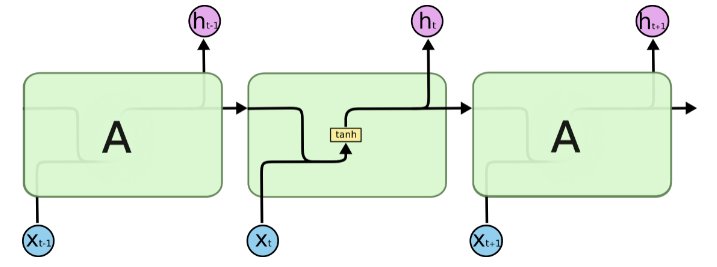
Long Short-Term Memory (LSTM) là một cải tiến của RNN. ra đời để hạn chế gradient quá nhỏ hoặc quá lớn trong mạng hồi quy RNN. Được đề xuất bởi Hochreite và Schmidhuber. Hoạt động rất hiệu quả trong nhiều vấn đề và hiện đang được sử dụng rộng rãi. Mô hình một lớp trong mạng hồi quy tại một thời điểm được biểu diễn bởi một khối xử lý với ngỏ ra là hàm Tanh. LSTM phức tạp hơn, nhưng vẫn dựa trên các kết nối hôi quy. Mô hình LSTM và RNN như hình 11a và 11b.





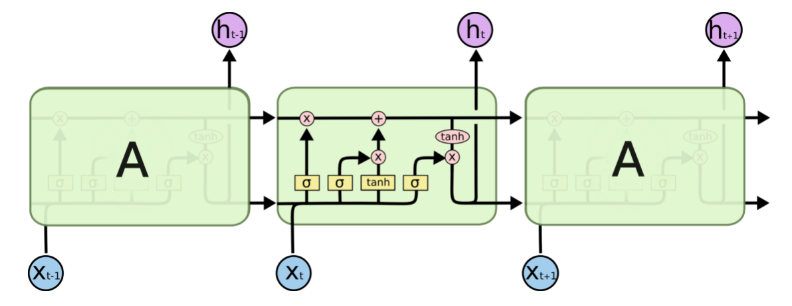
*Hình 11a. Recurrnet Neural Network cell Hình 11b. (b) LSRM cell*

*Hình 11a và 11b. Mô hình lớp RNN và lớp LSTM*



*Hình 12. Mô-đun lặp lại trong RNN tiêu chuẩn chứa một lớp duy nhất*

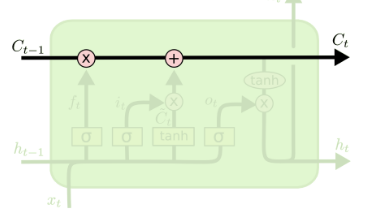
Các LSTM cũng có cấu trúc giống như RNN, nhưng mô-đun lặp lại có cấu trúc khác. Thay vì có một lớp mạng nơ-ron duy nhất thì LSTM có bốn lớp, tương tác theo một cách rất đặc biệt.



*Hình 13. Mô-đun lặp lại trong một LSTM chứa bốn lớp tương tác*

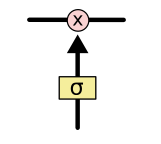
* + 1. **Ý tưởng cốt lõi đằng sau LSTM**

Chìa khóa của LSTM là trạng thái cell, đường nằm ngang chạy qua đầu sơ đồ. Trạng thái cell giống như một băng chuyền. Nó chạy thẳng xuống toàn bộ chuỗi, chỉ với một số tương tác tuyến tính nhỏ. Rất dễ dàng để thông tin trôi theo dòng không thay đổi.



*Hình 14. Mô tả thông tin trôi theo dòng không đổi*

LSTM có khả năng **loại bỏ hoặc thêm thông tin vào trạng thái cell**, được

điều chỉnh cẩn thận bởi **các cấu trúc được gọi là cổng.**

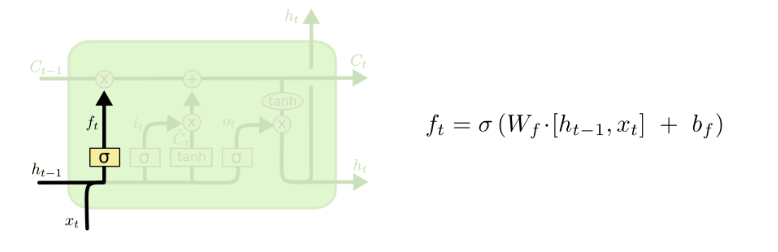
*Hình 15. Cấu trúc cổng*

Lớp sigmoid xuất ra các con số từ 0 và 1, mô tả mức độ thông qua của mỗi thành phần. Giá trị bằng 0 có nghĩa là "không có gì thông qua", trong khi giá trị bằng một có nghĩa là "để mọi thứ thông qua".

Một LSTM có ba trong số các cổng này, để bảo vệ và kiểm soát trạng thái.

* + 1. **Các bước của LSTM**

Bước đầu tiên trong LSTM của chúng ta là **quyết định thông tin nào chúng ta sẽ loại bỏ khỏi trạng thái cell**. Quyết định này được thực hiện bởi một lớp sigmoid được gọi là “lớp cổng quên”. Nó xem xét ht-1 và xt, và xuất ra một số từ 0 và 1 cho mỗi số ở trạng thái ô Ct-1. Số 1 đại diện cho "để mọi thứ thông qua" trong khi số 0 đại diện cho "không có gì thông qua."

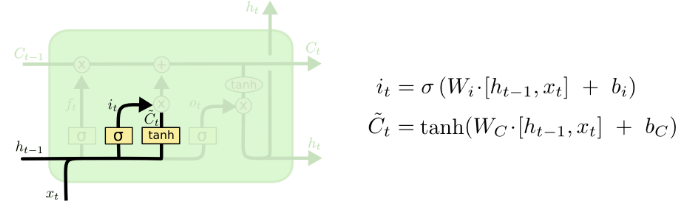




*Hình 16. Cổng quyết định cho thông tin đi qua hay không*

Bước tiếp theo là **quyết định thông tin mới mà chúng ta sẽ lưu trữ ở trạng thái cell.** Điều này có hai phần. **Đầu tiên**, một lớp sigmoid được gọi là “lớp cổng đầu vào” quyết định những giá trị nào chúng ta sẽ cập nhật. Tiếp theo, một lớp **tanh tạo ra một vectơ có các giá trị mới**, t , có thể được thêm vào trạng thái.

Trong bước tiếp theo, chúng ta sẽ **kết hợp hai điều này để tạo bản cập nhật cho trạng thái.**

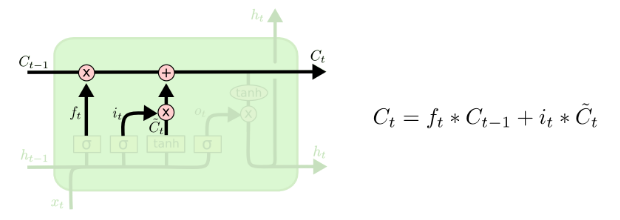


*Hình 17. Tạo bản cập nhật cho trạng thái*

**Bây giờ đã đến lúc cập nhật trạng thái cell cũ, Ct-1, thành trạng thái cell mới Ct.** Các bước trước đã quyết định những gì cần làm.

Chúng ta nhân trạng thái cũ với ft, quên đi những thứ mà chúng ta đã quyết định quên trước đó. Sau đó, chúng ta thêm it ∗ t. Đây là các giá trị mới, được chia tỷ lệ theo mức độ mà chúng ta quyết định cập nhật từng giá trị trạng thái.

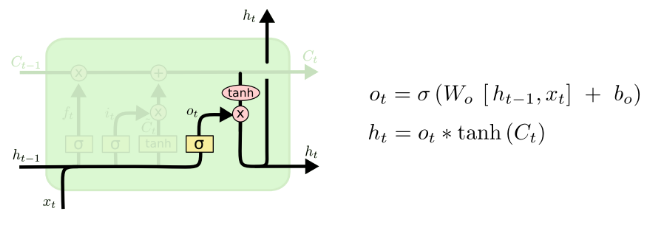
Trong trường hợp của mô hình ngôn ngữ, đây là nơi chúng ta thực sự loại bỏ thông tin của chủ thể cũ và thêm thông tin mới, như chúng ta đã quyết định ở các bước trước.



*Hình 18. Cập nhật lại Ct*

**Cuối cùng, chúng ta cần quyết định những gì chúng ta sẽ xuất ra.** Đầu ra này sẽ dựa trên trạng thái cell của chúng ta, nhưng sẽ là một phiên bản được lọc. **Đầu tiên,** chúng ta **chạy một lớp sigmoid quyết định phần nào của trạng thái cell mà chúng ta sẽ xuất ra.** **Sau đó,** **chúng ta đặt trạng thái cell thông qua hàm tanh** (để đẩy các giá trị nằm trong khoảng từ −1 đến 1) và nhân nó với đầu ra của cổng sigmoid, để chúng ta chỉ xuất các phần mà chúng ta đã quyết định.

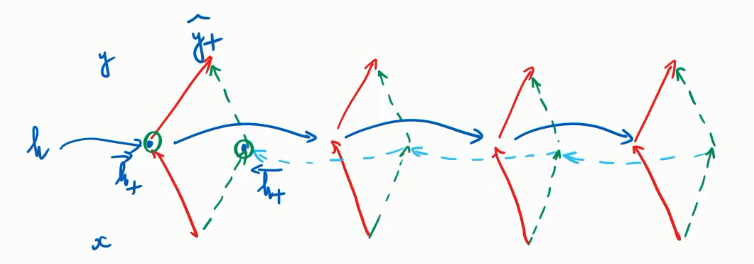
Đối với ví dụ về mô hình ngôn ngữ, vì nó vừa nhìn thấy một chủ đề, nó có thể muốn xuất ra thông tin liên quan đến một động từ, trong trường hợp đó là những gì sắp xảy ra tiếp theo. Ví dụ: nó có thể xuất ra liệu chủ ngữ là số ít hay số nhiều, để chúng ta biết động từ nên được chia thành dạng nào nếu đó là những gì tiếp theo sau.



*Hình 19. Quyết định những gì xuất ra thông qua Ot*

* 1. **BI-DIRECTIONAL LSTM**

Việc nhận dạng chính xác tên riêng trong một đoạn văn bản phụ thuộc không chỉ vào các thông tin phía trước của từ đang xét mà còn cả các thông tin phía sau. Tuy nhiên, một kiến trúc LSTM truyền thống với một lớp duy nhất chỉ có thể dự đoán nhãn của từ hiện tại dựa trên thông tin có được từ các từ nằm trước đó. Bidirectional LSTM (BiLSTM) đã được tạo ra để khắc phục điểm yếu trên. Một kiến trúc BiLSTM thường chứa 2 mạng LSTM đơn được sử dụng đồng thời và độc lập để mô hình hoá chuỗi đầu vào theo 2 hướng: từ trái sang phải (forward LSTM)

và từ phải sang trái (backward LSTM) Hình 4.

*Hình 20. Mô hình Bi-Birectional LSTM*

Là mạng có hai chiều. Việc tính toán giống như mạng RNN thuần nhưng chúng ta xẽ có hai chiều, đó là chiều từ trái qua phải và chiều từ phải qua trái. Công thức của chúng ta vẫn tương tự như mạng RNN.

Mục đích của chúng ta là tính được lịch sử h.

* Xét chiều từ trái sang phải

= f ()

bằng hàm activation function của (để chuyển xt thành vecto có chiều giống h) cộng với (để chuyển h ở phía trước thành h hiện tại) và cộng thêm bias.

* Chiều từ phải đến trái thì chúng ta có theo chiều ngược lại

= f ()

* Để tính được (xác xuất của một từ nào đấy trên một tập từ điển) ta nối hai vecto h lại với nhau.

= g ()

bằng hàm activation function nào đó của (để ghép hai vecto ) cộng với bias c.

# Chương 3 thiết kế MÔ HÌNH bài toán sinh từ

* 1. **Ý tưởng thiết kế**

Ví dụ: Bài toán chúng ta mong muốn có kết quả là “I Love Ai So Much”. Đầu tiên chúng ta xẽ đưa từ “I” vào thì xẽ sinh ra từ “Love” ta được “I Love”. “I Love” xẽ sinh ra “Ai” thì được “I Love Ai”. Tương tự như vậy ta có quá trình như sau:

I Love

I Love Ai

I Love Ai So 

I Love Ai So Much

Kết quả thu được “I Love Ai So Much”.

Đây là ví dụ điển hình về mô hình ngôn ngữ, ta phải tính làm sao để xác suất của một từ là cao nhất trong lịch sử. Thì từ tập dữ liệu thì làm sao chúng ta có thể biến các câu thành bộ train của chúng ta. Dựa vào tính chất cảu mô hình ngôn ngữ, dựa vào lịch sử, ví dụ như trên “I” thì xẽ ra “Love”, lịch sử trước đó là “I” thì xẽ ra “Love”. Feature của chúng ta là “I” thì nhãn của chúng ta là “Love”, feature là “I Love” thì nhãn là “Ai”,….

* Lựa chọn phương pháp thiết kế:
* Xử lý ngôn ngữ tự nhiên (NLP) - tokenizations and sequence
* Word Embedding
* Mô hình ngôn ngữ và mạng nơ ron hồi quy (Language Model and Recurrent Neural Network)
* Long Short-Term Memory (LSTM)
* Bi-Directional LSTM
* Kết hợp với Google Colab để thực hiện phần code

## chuẩn bị bộ từ điển

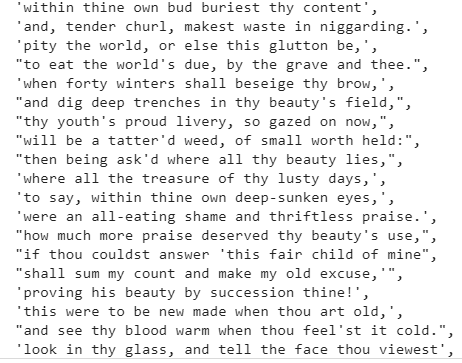
* **Load và import những thư viện cần thiết:**

Load tập dữ liệu:

Thư viện Keras hỗ trợ thiết kế các mạng nơ-ron thông qua mô hình các lớp nổi tiếp (Sequential). Trong bài này, chúng ta sử dụng lớp LSTM để xây dựng mô hình mạng nơ-ron hồi quy. Khai báo các mô đun trong thư viện để sử dụng các lớp cần thiết cho thiết kế ron hồi quy.



Cuối cùng chúng ta đã load được tập từ điển như hình 13. Vì tập từ điển khá dài nên chỉ xét một đoạn nhỏ của từ điển.



*Hình 21. Từ điển khi load về*

* **Yêu cầu cụ thể của bài toán: Sinh từ tiếp theo khi gõ bất cứ một câu nào có chiều dài bất kỳ.**

Ví dụ như: despite of wrinkles this the golden time to heart's sight 'must '. Khi ta gõ “despite of wrinkles” thì xẽ sinh ra “this the golden time to heart's sight ' must '”.



Đầu tiên chúng ta xẽ tạo ra một Tokenizer: tokenizer = Tokenizer()

Sau đó chúng ta xẽ biến tất cả các từ này thành từ điển. Các chữ xẽ trở thành các số chứ không còn là từ thông thường nữa.

tokenizer.fit\_on\_texts(corpus)

Tổng số lượng các từ để chúng ta training:

total\_words = len(tokenizer.word\_index) + 1

Chuyển các câu thành một chuỗi số:

token\_list = tokenizer.texts\_to\_sequences([line])[0]

Bắt đầu từ vị trí một trở đi, chúng ta bắt đầu cắt từ vị trí 0 đến vị trí i hiện tại để lấy được nhãn. Lấy ví dụ một ma trận như bên dưới sau khi chúng ta tạo chuỗi đầu vào.

| [153,5,174]  [153,5,174,3]  [153,5,174,3,856]  [153,5,174,3,856,857] |  |  |
| --- | --- | --- |

## Xử lý dữ liệu, chia features và lable

Đầu tiên chúng ta xẽ pad sequences. Padding ở đằng trước để chúng ta có thể đẩy được các nhãn của chúng ta về đằng sau thì cắt xẽ dể hơn.



Sau khi padding xong thì chúng ta lấy tất cả giá trị ở trước từ cuối cùng và từ cuối cùng chúng ta xẽ để là nhãn ở đây.



Lable của chúng ta là một vecto: [ 417 877 166 ... 3210 15 14]

Sau khi làm xong thì chúng ta xẽ in các features ra thì ta thấy đằng trước là các số 0 và đằng là các từ của chúng ta.



Kết quả:

array([[ 0, 0, 0, ..., 0, 0, 34],

[ 0, 0, 0, ..., 0, 34, 417],

[ 0, 0, 0, ..., 34, 417, 877],

...,

[ 0, 0, 0, ..., 3209, 493, 493],

[ 0, 0, 0, ..., 493, 493, 3210],

[ 0, 0, 0, ..., 493, 3210, 15]], dtype=int32)

Rõ ràng lúc này đầu vào tất cả các câu của chúng ta có chiều dài bằng nhau.

Điều này giúp cho quá trình training trở nên dể dàng hơn.

## Xây dựng model với độ chính xác trên 80%

Sau khi xử lý dữ liệu, chia features và lable xong, để xây dựng model với độ chính xác trên 80% chúng ta xẽ thực hiện các bước như sau:



Tạo một Sequential(): model = Sequential()

Thêm một lớp Embedding:

model.add(Embedding(total\_words, 100, input\_length=max\_sequence\_len-1))

Thêm một lớp Bidirectional:

model.add(Bidirectional(LSTM(150, return\_sequences = True)))

Thêm một lớp Dropout tắt những thứ khi cần thiết:

model.add(Dropout(0.2))

Thêm một LSTM, LSTM xẽ nhận h từ lớp Bidirectional:

model.add(LSTM(100))

Sau đó đi qua một lớp Dense. Vì đây là bài toán xem xác suất của từ nào trong từ điển là cao nhất nên lớp Dense cuối cùng ta lấy tất cả các từ trong từ điển:

model.add(Dense(total\_words/2, activation='relu', kernel\_regularizer=regularizers.l2(0.01)))

model.add(Dense(total\_words, activation='softmax'))

Loss function là categorical\_crossentropy, optimizer là adam và metrics là accuracy:

model.compile(loss='categorical\_crossentropy', optimizer='adam',metrics=['accuracy'])

In ra summary:

print(model.summary())

**Kết quả:**

Model: "sequential"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

embedding (Embedding) (None, 10, 100) 321100

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

bidirectional (Bidirectional (None, 10, 300) 301200

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout (Dropout) (None, 10, 300) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

lstm\_1 (LSTM) (None, 100) 160400

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense (Dense) (None, 1605) 162105

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_1 (Dense) (None, 3211) 5156866

=================================================================

Total params: 6,101,671

Trainable params: 6,101,671

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

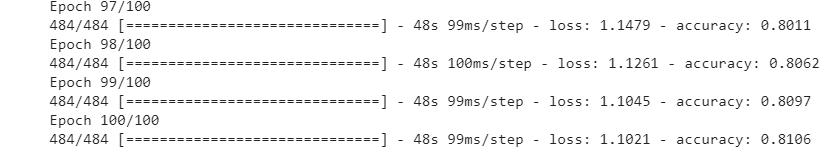
None

Sau khi có Summary thì chúng ta xẽ tiến hành in kết quả training ra.



Vì chúng ta có 6,101,671 params nên quá trình train của chúng ta khá lâu. Đây là một nhược điểm của RNN so với CNN như đã đề cập ở phần trước.

Vì epoch của chúng ta là 100 lần, nên chúng ta chỉ xem xét các epoch cuối. Kết quả training là 81.06% đạt yêu cầu bài toán đặt ra.

Kết quả sau quá trình training:

*Hình 22. Kết quả sau khi training*

## Dự đoán 10 từ tiếp theo

Sau khi thực hiện xong quá trình training. Tiếp theo chúng ta xẽ dự đoán 10 từ tiếp theo.



Chúng ta xẽ nhập 3 từ đầu tiên ‘despite of wrinkles’ và nhiệm vụ của chúng ta là sinh ra 10 từ tiếp theo dựa trên tập từ điển:

test\_seq = 'despite of wrinkles'

next\_words = 10

Chúng ta xẽ cho lặp 10 lần và đầu tiên xẽ đưa câu câu vào ‘despite of wrinkles’ trong model của chúng ta. Sau đó padding ở đường như bình thường. Và đưa vào model của chúng ta dự đoán ra một từ.

for \_ in range(next\_words):

  token\_list = tokenizer.texts\_to\_sequences([test\_seq])[0]

  token\_list = pad\_sequences([token\_list], maxlen=max\_sequence\_len-1, padding='pre')

  predicted = model.predict\_classes(token\_list, verbose=0)

  output\_word = ""

  for word, index in tokenizer.word\_index.items():

    if index == predicted:

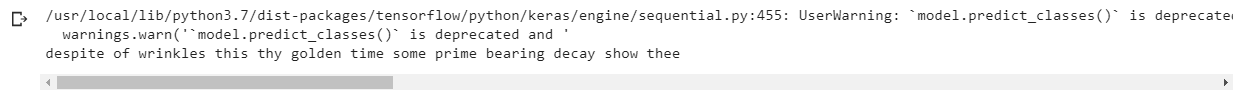
      output\_word = word

      break

Sau khi dự đoán ra từ rồi ta tiếp tục nối từ này vào cấu trúc và tiếp tục đưa câu này vào trong model, đúng như tính chất của mô hình ngôn ngữ.

  test\_seq += " " + output\_word

print(test\_seq)

Và đây chính là kết quả khi chạy xong. Từ 3 từ ban đầu ‘despite of wrinkles’ sau khi đi qua mô hình thì sinh thêm 10 từ như kết quả trên, kết quả của chúng ta dựa vào tập từ điển đã tạo.

*Hình 23. Kết quả sau sinh ra 10 tiếp theo*

# CHƯƠNG 4 KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Sau thời gian học tập và làm việc cuối cùng đề tài cũng đã hoàn thành đúng thời gian quy định. Đề tài giúp sinh viên ứng dụng lý thuyết vào giải quyết bài toán cụ thể, nhằm giúp cho sinh viên nắm chắc hơn những kiến thức đã được học.

## 4.1 Kết luận

Ưu điểm của bài toán này là đã quyết được các yêu cầu của bài toán sinh từ thực tế mà chúng ta hay thấy ở các ứng dụng của Google Search hay các mô hình ngôn ngữ khác. Ví dụ trong Google Search khi ta nhập vào một vài từ muốn tìm kiếm thì xẽ hiện ra thêm các từ khác để giúp ta tìm kiếm dể dàng hơn.

Một nhược điểm của bài toán này là quá trình training lâu do đó cần những siêu máy tính với hệ thống sử lý và lưu trữ lớn mới có thể chính xác và nhanh chóng.

Bài toán đã ứng dụng được mạng LSTM, BI-DIRECTIONAL LSTM và các mô hình xử lý ngôn ngữ tự nhiên. Cách mà chúng ta xử dụng Embedding để thêm các phần tử trong câu trước bằng với phần tử trong câu sau để dể dàng cho việc training

## Hướng phát triển

Với sự phát triển như vũ bảo của trí tuệ nhân tạo kéo thì sự phát triển của mô hình xử lý ngôn ngữ tự nhiên. Ngoài bài toán sinh từ trên chúng ta có rất nhiều ứng dụng hữu hiệu của xử lý ngôn ngữ tự nhiên như: Trích xuất thông tin chọn lọc, phản hồi câu hỏi, tóm tắt văn bản, dịch máy. Trong đó phản hồi khách hàng một cách tự động sử dụng chatbot là một đi với tiềm năng lớn đi mở rộng và phát triển, bên cạch đó cũng có thể ứng dụng mô hình này để phân tích cảm xúc của câu,…

PHỤ LỤC

Source Code của bài toán sinh từ:

* Load data và khai báo thư viện









* Sinh từ tiếp khi gõ bất cứ một câu nào có chiều dài bất kỳ.
* Xử lý dữ liệu. Chia features, label



* Xây dựng model Yêu cầu độ chính xác: 80%



* Dự đoán 10 từ tiếp theo

TÀI LIỆU THAM KHẢO

[1] Trương Ngọc Sơn, *Trí Tuệ Nhân Tạo và Cơ Sở Ứng Dụng*, Đại Học Quốc Gia Thành Phố Hồ Chí Minh, năm xuất bản 2020.

[2] <https://quantrimang.com/ai>

[3] <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>

[4] <https://d2l.aivivn.com/chapter_natural-language-processing> pretraining/word2vec\_vn.html?fbclid=IwAR204FTYsav\_0Gzy7QEKaqmQi0CUev0\_0rXjtnomxUc54\_YTFMxCWWpshVg

[6] https://en.wikipedia.org/wiki/Bidirectional\_recurrent\_neural\_networks

[5]<https://www.youtube.com/watch?v=FT1ZZdcur5A>

[6]https://www.youtube.com/watch?v=akRbuXokLSo

[7] <https://www.youtube.com/watch?v=-gyb1dpeGCA>

