TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**Báo cáo quá trình theo chủ đề**

**VIETNAMESE LANGUAGE MODEL**

**BASED ON NEURAL NETWORK**

***Danh sách thành viên:***

Phạm Dương Thành Long

– 51603190

Trần Ngọc Bảo Duy

– 51702091

MỤC LỤC

[MỤC LỤC 1](#_Toc45895572)

[DANH MỤC CÁC HÌNH VẼ VÀ BẢNG BIỂU 2](#_Toc45895573)

[1. Tổng quan: 3](#_Toc45895574)

[1.1 Tại sao phải dụng mạng neural để xây dựng mô hình ngôn ngữ ? 3](#_Toc45895575)

[1.2 Giải quyết các vấn đề về số chiều trong biểu diễn các words. 3](#_Toc45895576)

[1.3 Quan một word với các words trước đó: 4](#_Toc45895577)

[2. Mô hình neural network: 4](#_Toc45895578)

[2.1 Chi tiết các tham số trong mạng: 4](#_Toc45895579)

[2.2 Quá trình tính toán trong mạng: 7](#_Toc45895580)

DANH MỤC CÁC HÌNH VẼ VÀ BẢNG BIỂU

[Hình 1: Cấu trúc mạng neural . Trong đó g là mạng neural và C (i) là vectơ đặc trưng của word thứ i. 5](#_Toc45359763)

1. Tổng quan:

1.1 Tại sao phải sử dụng mạng neural để xây dựng mô hình ngôn ngữ ?

Có thể thấy với cách xây dựng mô hình ngôn ngữ dựa trên xác suất tuy có hiệu suất rất nhanh và độ chính xác tương đối trong một số bài toán cụ thể. Tuy nhiên điểm bất cập lớn nhất đối với phương pháp này đó là khi một word bất kỳ không nằm trong bộ từ điển của chúng ta thì chắc chắn xác suất của chúng sẽ bằng không. Điều này thường xuyên gặp phải khi ta biểu diễn các words mới và cấu trúc mới. từ đó làm cho mô hình của chúng ta chở nên quá khớp (overfitting) với các ngữ liệu mới được nhận biết dẫn đến mô hình không còn có tính thực tế nữa.

Từ đó có thể thấy ta cần tạo ra một mô hình có khả năng tạo ra một câu tương tự cho các words có khoảng cách gần giống câu đã cho. Trong một thí dụ điển hình ta có một câu : “Một con mèo đang đi vào nhà bếp” trong corpus thì mô hình sẽ giúp ta tạo ra một câu tương tự dựa trên câu đó như: Một con chó đang chạy vào phòng ngủ. Ở đây ta thấy được sự tương đồng về khoảng cách giữa word “mèo” và “chó”( tương tự đối với “đi” –“chạy” và “nhà\_bếp” – "phòng ngủ” những word này có vai trò ngữ nghĩa *(semantic)* và ngữ pháp *(grammatical roles)* tương tự nhau.

1.2 Giải quyết các vấn đề về số chiều trong biểu diễn các words.

Các feature vectors cho các khía cạnh khác nhau của một word: Mỗi word được liên kết với một điểm trong không gian vector. Có thể thấy Số lượng features (*ví dụ m =30,60, ..100)* vẫn luôn nhỏ hơn rất nhiều so với kích thước của bộ từ vựng *(ví dụ word cabulary = 20.000 từ)*. Hàm xác suất của word tiếp theo được biểu thị dưới dạng tích của các xác suất có điều kiện của các words trước nó. ví dụ: sử dụng mạng neural đa tầng để dự đoán word tiếp theo được đưa ra bằng các words trước đó. Các tham số parameter có thể được điều chỉnh lặp đi lặp lại để ***maximize log-likelihood*** của tập huấn luyện. Các vector features có thể liên quan đến mỗi word được học hoặc chúng cũng có thể được tạo ra bằng cách sử dụng các kiến thức liên quan đến các features về ngữ nghĩa đã được học

Ví dụ: ta có thể tạo ra 1 list các câu từ cầu đầu tiên:

Câu đầu tiên: “Con mèo đang đi vào phòng khách”

* Chú chó đang chạy vào phòng
* Chú mèo đã đi vào phòng
* Một con chó đang đi vào phòng khách
* Con chó đã chạy vào phòng,v.v.

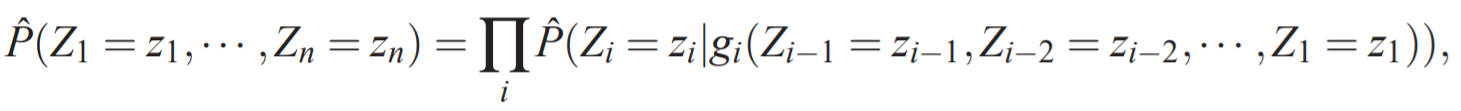
Như ta có thể thấy, mô hình sẽ khái quát hóa các words tương tự nhau thành các feature vectors tương đương ứng với mỗi trường hợp như vậy. Và vì hàm xác suất có chức năng “ smooth” các features này.

* Cho nên, **một thay đổi nhỏ trong các features sẽ tạo ra một thay đổi tương ứng trong xác suất**.

Chính vì lẽ đó, sự tồn tại của bất kỳ một câu nào trong trong dữ liệu huấn luyện cũng sẽ làm tăng hoặc giảm xác suất, không chỉ của câu đó, mà cả tổ hợp của các câu “hàng xóm” gần với nó của nó trong không gian các câu (*được biểu thị bằng các chuỗi feature vectors).*

1.3 Quan một word với các words trước đó:

Ta có tưởng sử dụng các mạng neural để mô hình hóa các phân phối rời rạc cho số chiều ở mức cao có tính ứng dụng trong **joint probability** của *một tập hợp các biến ngẫu nhiên trong đó mỗi biến có thể có các tập features khác nhau).* Công thức trên đuọc biểu diễn như sau:



Trong đó:

là một hàm được biểu thị bởi một mạng neural có kiến ​​trúc đặc biệt từ trái sang phải, với các tham số tính toán của output thứ dùng biểu thị phân phối có điều kiện của với giá trị của trước đó, theo một số thứ tự tùy ý. Một sự khác biệt quan trọng khác là ở đây, tất cả các *(word ở vị trí thứ i)*, đều đề cập đến cùng một loại đối tượng là ***một word***.

2. Mô hình neural network:

2.1 Chi tiết các tham số trong mạng:

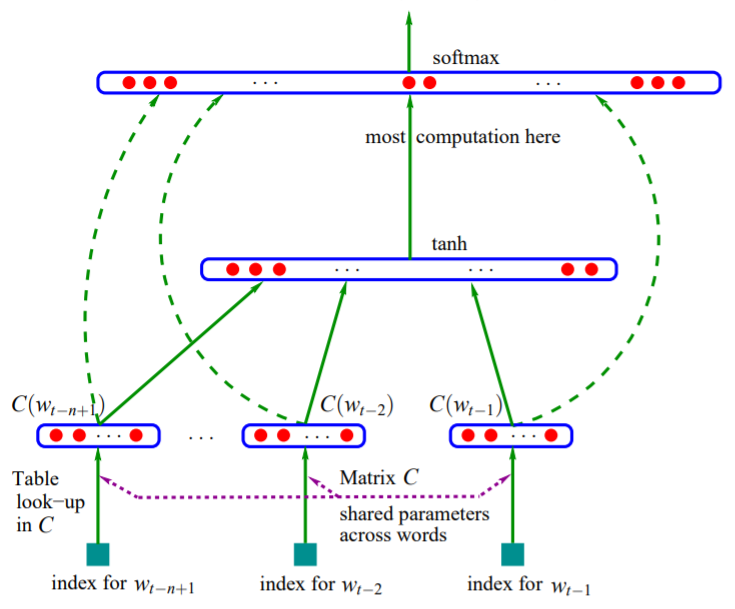
Tập huấn luyện là một chuỗi các words trong đó *.(Với* ***V*** *là một* ***bộ từ vựng*** *kích thước lớn nhưng hữu hạn*). Mục tiêu là tìm ra một mô hình tốt với nghĩa là nó sẽ cho “Out-of-sample” nếu giá trị của likelihood quá cao. Ta có trung bình tích của (còn được gọi là ***perplexity****-sự hỗn loạn),* đó cũng là số mũ âm trung bình của log-likelihood.

Hạn chế duy nhất trên mô hình là đối với bất kỳ lựa chọn nào thì , với . Bởi tích của các xác suất có điều kiện, người ta có được một mô hình xác suất kết hợp *(joint probability)* của chuỗi các words*.*

Ta sẽ phân rã hàm thành 2 phần:

1. Một ánh xạ **C** từ bất kỳ phần tử **i** nào thuộc **V** sang một vector thực , sẽ đại diện cho các feature vector phân phối đến mỗi word trong bộ từ vựng. Trên thực tế, C đại diện bằng một ma trận kích thướccác free parameters.
2. Hàm xác suất trên các words, được biểu thị bằng **C**: một hàm **g** ánh xạ một chuỗi đầu vào là các feature vectors cho các words trong ngữ cảnh, *,* đến phân phối xác suất có điều kiện trên các words trong **V** cho word tiếp theo . Đầu ra cũng là một vector có phần tử thứ ước lượng xác suất bằng như hình 1:

Ta có: hàm là một tổ hợp của 2 ánh xạ (C và g), với C được chia sẻ trên tất cả các words trong ngữ cảnh. Với mỗi hai phần như thế sẽ được liên kết bởi một số tham số. Các đầu ra thứ i:



Hình 1: Cấu trúc mạng neural . Trong đó g là mạng neural và C (i) là feature vector của word thứ i.

Tham số của ánh xạ C đơn giản là các features vectors của chúng, được đại diện bởi một ma trận C kích thước **,** trong đó dòng thứ i là các features ***C(i)*** cho word thứ i. Hàm ***g()*** có thể được lập trình bằng một mạng *feed-forward* hoặc *recurrent neural network*. Các bộ tham số tổng thể là

Trainning đạt được bằng cách tìm kiếm nhầm tối đa hóa log-likelihood của training corpus:

Trong đó là một ***weight decay*** áp dụng cho các trọng số của mạng neural và ma trận C, không áp dụng cho các biases.

Trong mô hình trên, số lượng free parameters sẽ phân chia theo tỷ lệ tuyến tính với *số lượng word trong bộ từ vựng*. Ngoài ra nó còn phân chia theo tỷ lệ tuyến tính với thứ tự ***n***: hệ số tỷ lệ có thể được giảm xuống thành tuyến tính phụ *(sub-linear)* nếu nhiều phần tử chia sẻ được giới thiệu,

*ví dụ: sử dụng mạng time-delay neural network hoặc recurrent neural network (hoặc kết hợp cả 2)....*

Trong hầu hết các thử nghiệm, mạng neural có một hidden layer vượt khỏi word features mapping và các liên kết trực tiếp từ các features đến output

Do đó, thực sự có hai hidden layer: Layer C dùng để chia sẽ giữa các word features và một hidden layer thông thường. Chính xác hơn, mạng neural tính toán hàm sau, với activation ở output là softmax *(đảm bảo đầu ra =1)*

Xác suất log của tại mỗi output thứ i, được tính như sau, với các tham số ,

Trong đó:

* ***tanh*** làm một hàm tiếp tuyến được áp dụng cho từng phần tử,
* ***W*** tùy ý bằng 0 (no direct connections)
* ***x*** là word features layer activation vector, giúp nối các word features đầu vào của ma trận C

Lúc này ta sẽ có thêm một số đại lượng là ***h*** là số ***hidden units*** *và* ***m*** *số features liên kết với mỗi word.* Khi không có kết nối trực tiếp từ các *features* đến đầu ra mong muốn, ma trận ***W*** được *set bằng 0.*

Các free parameters của mô hình là:

* Các *output* ***biases b*** *(với |V| phần tử)*
* Các *biases* tại *hidden layer* ***d*** *(với h phần tử )*
* Các hidden-to-output weights
* Các word features cho output weights
* Các hidden layer weights
* Các word features **C** (a |V| × *m* matrix)

Ta có:

Số lượng các free parameters là Yếu tố chi phối là . Lưu ý rằng trên lý thuyết, nếu có một *weight decay* trên các *weight*s ***W*** và ***H*** nhưng không có trên ***C***, thì *Y* và *H* có thể hội tụ về 0 trong khi *C* sẽ nở ra *(blow up).* Và trong thực tế, Ta không quan sát được trường này khi training với stochastic gradient ascent.

Stochastic gradient ascent trên neural network bao gồm thực hiện việc update tuần tự sau khi biễu diễn *tth*word trong corpus

Trong đó:

* ε là learning rate. Lưu ý rằng một phần lớn các tham số không cần phải được cập nhật hoặc duyệt sau mỗi ví dụ:
* word features của tất cả các words j không xuất hiện trong input window

2.2 Quá trình tính toán trong mạng:

COMPUTATION FOR PROCESSOR i, example t

1. ***Giai đoạn duyệt tiến*** *(feed-forward phase)*
2. Thực hiện tính feedforward cho *word features layer:*
3. Thực hiện tính feedforward cho hidden layer:
4. Thực hiện tính feedforward cho output units tại vị trí tại khối thứ i-th:

Si← 0

Vòng lặp từ j tại khối thứ i-th:

1. Tính toán và chia sẽ **S =** trong bộ xử lý . Điều này có thể dễ dàng thực hiện với MPI ***Allreduce*** operation, điều này giúp tính toán và chia sẽ hiệu quả tổng trên.
2. Chuẩn hóa xác suất:

Vòng lặp từ j tại khối thứ i-th:

1. Cập nhật log-likelihood.

* Nếu rơi vào block của , thì CPU i gửi đến *CPU 0.*
* *CPU 0* tính và bám sát theo tổng của log-likelihood.

1. ***Giai đoạn duyệt lùi*** *(backward / update phase),* ***với learning rate ε.***
   1. Thực hiện tính backward gradient cho các output units trong khối i-th: clear gradient vectors và . Vòng lặp từ j tại khối thứ i-th:

* 1. Perform backward Sum and share and across processors. This can easily be achieved with an MPI *Allreduce* operation
  2. Back-propagate through and update hidden layer weights: Loop over *k* between *1* and *h*,
  3. ) Update word feature vectors for the input words: Loop over k between 1 and *n−1:*

where is the k-th block (of length m) of the vector

2.3 Cách tạo feature từ dữ liệu thô cho các câu:

2.3.1 Phương pháp One-Word-In, One-Word-Out Sequences:

Ví dụ ta có một câu: “Hôm nay tôi ăn sushi”

Thì lúc này ta sẽ lấy một từ trong câu theo thứ tự từ đầu đến cuối làm input và từ tiếp theo của nó làm output:

**X y**

* Hôm nay
* nay tôi
* tôi ăn
* ăn sushi.

2.3.2 Phương pháp Line-by-Line Sequence:

Ví dụ một câu khác: “ Tôi và cô ấy cùng đến trường”

Giả sử, ở đây ta sẽ đồng nhất các câu có độ dài là 6 trước khi xây dựng data.

|  |  |
| --- | --- |
| **X** | **y** |
| \_, \_, \_, \_, \_, Tôi | và |
| \_, \_, \_, \_, Tôi, và | cô |
| \_, \_, \_, Tôi, và, cô | ấy |
| \_, \_, Tôi, và, cô, ấy | cùng |
| \_, Tôi, và, cô, ấy, cùng | đến |
| Tôi, và, cô, ấy, cùng, đến | trường |

Cách tiếp cận trên sẽ giúp cho mô hình nắm bắt được chọn vẹn các ngữ cảnh đang diễn ra trong câu giảm được sự mơ hồ mà phương pháp one-word-in-and-out gặp phải.

Cách tiếp cận trên cũng phù hợp cho ta trong việc tạo ra một mô hình hóa các ngữ liệu hiện tại và sinh ngữ liệu (generating text).

2.3.3 Phương pháp Two / three Words-In, One-Word-Out Sequence:

Đây là cách tiếp cận tương tự như phương pháp 1 nhưng sẽ giúp ta bao hàm tốt hơn ý nghĩa của các từ lân cận thay vì phải sử dụng cả 1 đoạn thật dài để giải quyết bài toán. Ngoài ra nó còn giúp trực quan hóa các cụm từ nếu cần khai thác trong một dữ liệu lớn và có tính hệ thống. phù hợp cho nhận diện các cấu trúc ngữ pháp, các cụm từ hay gặp.

2.4 Tiền xử lý dữ liệu:

Bước 1: Sau khi chọn phương pháp tạo câu cho các đầu vào ta sẽ tiến hàng lower case tất cả các ký tự trong câu.

Bước 2: Tokenizer các ký tự trong câu

Bước 3: lọc các từ không dùng đến hoặc ký tự đặc biệt ngoài bản mã Alpha

Bước 4: Tạo vocabulary từ các ký tự trong câu

Bước 5: Encoding các ký tự trong thành dạng số interger để giảm bớt quá trình lưu trữ.