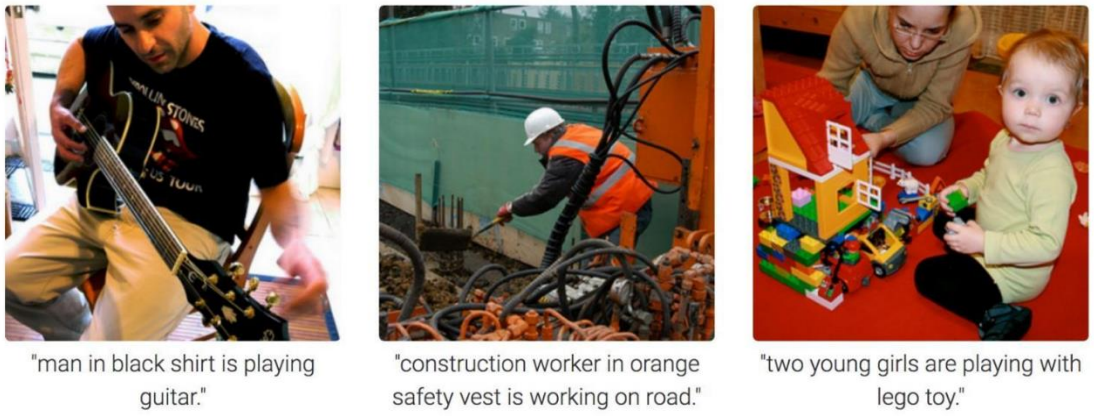
	VIETTEL AI RACE	TD037
	ỨNG DỤNG THÊM MÔ TẢ CHO ẢNH	Lần ban hành: 1

Ở những bài trước tôi đã giới thiệu về mô hình Recurrent Neural Network (RNN) cho bài toán dữ liệu dạng chuỗi. Tuy nhiên RNN chỉ có short term memory và bị vanishing gradient. Tiếp đó tôi đã giới thiệu về Long short term memory (LSTM) có cả short term memory và long term memory, hơn thế nữa tránh được vanishing gradient. Bài này tôi sẽ viết về ứng dụng của LSTM cho ứng dụng image captioning.

1. Ứng dụng



Hình 18.1: Ví dụ image captioning [10]

Ta có thể thấy ngay 2 ứng dụng của image captioning:

- Để giúp những người già mắt kém hoặc người mù có thể biết được cảnh vật xung quanh hay hỗ trợ việc di chuyển. Quy trình sẽ là: Image -> text -> voice.
- Giúp google search có thể tìm kiếm được hình ảnh dựa vào caption.

2. Dataset

Dữ liệu dùng trong bài này là Flickr8k Dataset. Mọi người tải ở đây. Dữ liệu gồm 8000 ảnh, 6000 ảnh cho training set, 1000 cho dev set (validation set) và 1000 ảnh cho test set.

Bạn tải về có 2 folder: Flicker8k_Dataset và Flicker8k_Text. Flicker8k_Dataset chứa các ảnh với tên là các id khác nhau. Flicker8k_Text chứa:

- Flickr_8k.testImages, Flickr_8k.devImages, Flickr_8k.trainImages, Flickr_8k.devImages chứa id các ảnh dùng cho việc test, train, validation.
- Flickr8k.token chứa các caption của ảnh, mỗi ảnh chứa 5 captions.

Ví dụ ảnh ở hình 18.2 có 5 captions:

- A child in a pink dress is climbing up a set of stairs in an entry way.
- A girl going into a wooden building.
- A little girl climbing into a wooden playhouse.

- A little girl climbing the stairs to her playhouse.
- A little girl in a pink dress going into a wooden cabin.

Thực ra 1 ảnh nhiều caption cũng hợp lý vì bức ảnh có thể được mô tả theo nhiều cách khác nhau. Một ảnh 5 caption sẽ cho ra 5 training set khác nhau: (ảnh, caption 1), (ảnh, caption 2), (ảnh, caption 3), (ảnh, caption 4), (ảnh, caption 5). Như vậy training set sẽ có 6000 * 5 = 40000 dataset.

3. Phân tích bài toán

Input là ảnh và output là text, ví dụ "man in black shirt is playing guitar".

Nhìn chung các mô hình machine learning hay deep learning đều không xử lý trực tiếp với text như 'man', 'in', 'black',... mà thường phải quy đổi (encode) về dạng số. Từng từ sẽ được encode sang dạng vector với độ dài số định, phương pháp đấy gọi là word embedding.

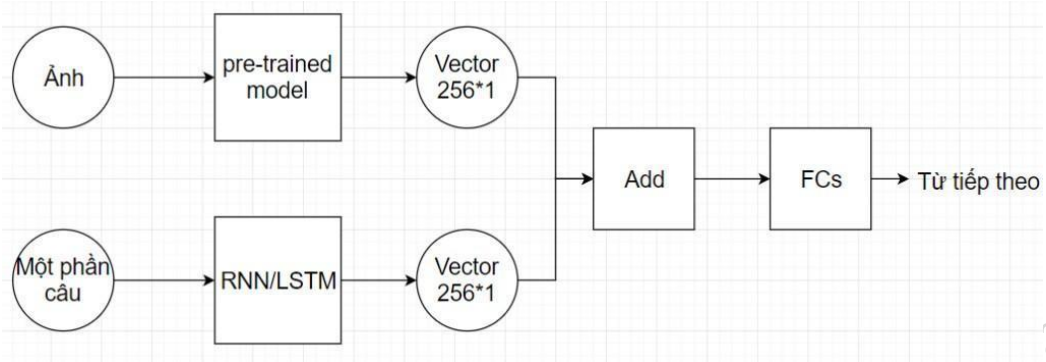
Nhìn thấy output là text nghĩ ngay đến RNN và sử dụng mô hình LSTM.

Input là ảnh thường được extract feature qua pre-trained model với dataset lớn như ImageNet và model phổ biến như VGG16, ResNet, quá trình được gọi là embedding và output là 1 vector.

Ý tưởng sẽ là dùng embedding của ảnh và dùng các từ phía trước để dự đoán từ tiếp theo trong caption.

Ví dụ:

- Embedding vector + A -> girl
- Embedding vector + A girl -> going
- Embedding vector + A girl going -> into
- Embedding vector + A girl going into -> a.
- Embedding vector + A girl going into a -> wooden building .
- Embedding vector + A girl going into a wooden -> building .



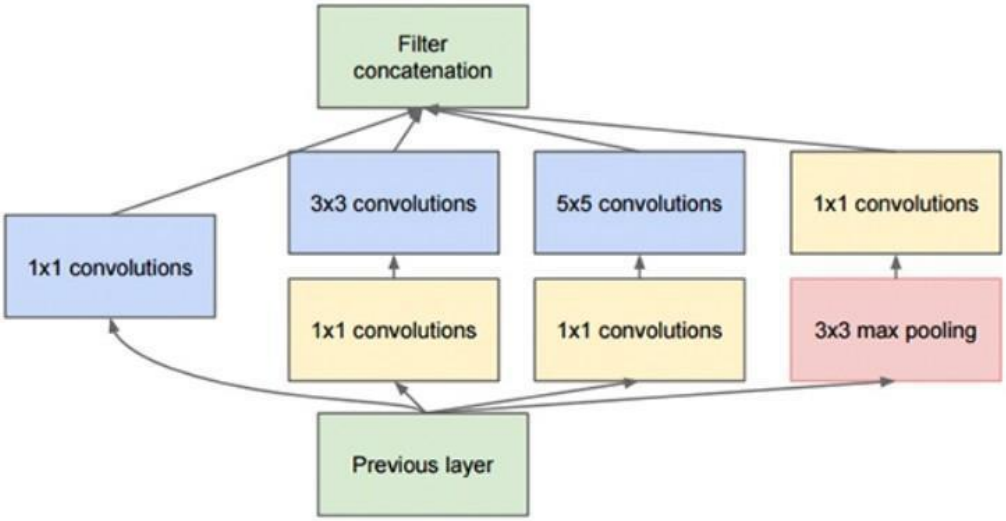
Hình 18.3: Mô hình của bài toán

Để dự đoán từ tiếp theo ta sẽ xây dựng từ điển các từ xuất hiện trong training set (ví dụ 2000 từ) và bài toán trở thành bài toán phân loại từ, xem từ tiếp theo là từ nào, khá giống như bài phân loại ảnh.

4. Các bước chi tiết

4.1 Image embedding với Inception

Có lẽ cái tên GoogLeNet sẽ quen thuộc hơn và gặp nhiều hơn so với Inception, GoogLeNet là version 1 của Inception, hiện giờ mô hình phổ biến là Inception v3.



Hình 18.4: Mô hình Googlenet, Going Deeper with Convolutions, Szegedy et al

Thay vì trong mỗi Conv layer chỉ dùng 1 kernel size nhất định như 3*3, 5*5, thì giờ ở một layer có nhiều kernel size khác nhau, do đó mô hình có thể học được nhiều thuộc tính khác nhau của ảnh trong mỗi layer.

Ta sẽ sử dụng pre-trained model Inception v3 với dataset Imagenet. Do là pre-trained model yêu cầu ảnh đầu vào là 229*229 nên ra sẽ resize ảnh về kích thước này. Sau khi qua pre-trained model ta sẽ lấy được embedding vector của ảnh, kích thước 256*1.

4.2 Text preprocessing

Ta xử lý text qua một số bước cơ bản.

- Chuyển chữ hoa thành chữ thường, "Hello" -> "hello"
- Bỏ các kí tự đặc biệt như "
- Loại bỏ các chữ có số như hey199

Sau đó ta sẽ thêm 2 từ "startseq" và "endseq" để biểu thị sự bắt đầu và kết thúc của caption. Ví dụ: "startseq a girl going into a wooden building endseq". "endseq" dùng khi test ảnh thì biết kết thúc của caption.

Ta thấy có 8763 chữ khác nhau trong số 40000 caption. Tuy nhiên ta không quan tâm lắm những từ mà chỉ xuất hiện 1 vài lần, vì nó giống như là nhiễu vậy và không tốt cho việc học và dự đoán từ của model, nên ta chỉ giữ lại những từ mà xuất hiện trên 10 lần trong số tất cả các caption. Sau khi bỏ những từ xuất hiện ít hơn 10 lần ta còn 1651 từ.

Tuy nhiên do độ dài các sequence khác nhau, ví dụ: "A", " A girl going", " A girl going into a wooden", nên ta cần padding thêm để các chuỗi có cùng độ dài bằng với độ dài của chuỗi dài nhất là 34. Do đó số tổng số từ (từ điển) ta có là 1651 + 1 (từ dùng để padding).

4.3 Word embedding

Để có thể đưa text vào mô hình deep learning, việc đầu tiên chúng ta cần làm là số hóa các từ đầu vào (embedding). Ở phần này chúng ta sẽ thảo luận về các mô hình nhúng từ (word embedding) và sự ra đời của mô hình word2vec rất nổi tiếng được google giới thiệu vào năm 2013. Trước đó ta thảo luận một số phương pháp cổ điển

4.3.1 One hot encoding

Phương pháp này là phương pháp đơn giản nhất để đưa từ về dạng số hóa vector với chiều bằng với kích thước bộ từ điển. Mỗi từ sẽ được biểu diễn bởi 1 vector mà giá trị tại vị trí của từ đó trong từ điển bằng 1 và giá trị tại các vị trí còn lại đều bằng 0.

Ví dụ: Ta có 3 câu đầu vào: "Tôi đang đi học", "Mình đang bận nhé", "Tôi sẽ gọi lại sau". Xây dựng bộ từ điển: "Tôi, đang , đi, học, Mình, bận, nhé, sẽ , gọi, lại, sau". Ta có các biểu diễn one hot encoding của từng từ như sau:

Tôi:[1,0,0,0,0,0,0,0,0,0,0],

đang: [0,1,0,0,0,0,0,0,0,0,0],

...

Mình: [0,0,0,0,1,0,0,0,0,0,0],

...
sau: [0,0,0,0,0,0,0,0,0,0,1].

Cách biểu diễn này rất đơn giản, tuy nhiên ta có thể nhận thấy ngay các hạn chế của phương pháp này. Trước hết, one hot encoding không thể hiện được thông tin về ngữ nghĩa của từ, ví dụ như khoảng cách(vector(Tôi) - vector(Mình)) = khoảng cách(vector(Tôi) - vector(đang)), trong khi rõ ràng từ "Tôi" và từ "Mình" trong ngữ cảnh như trên có ý nghĩa rất giống nhau còn từ "Tôi" và từ "đang" lại khác nhau hoàn toàn. Tiếp nữa, mỗi từ đều được biểu diễn bằng một vector có độ dài bằng kích thước bộ từ điển, như bộ từ điển của google gồm 13 triệu từ, thì mỗi one hot vector sẽ dài 13 triệu chiều. Cách biểu diễn này tốn rất nhiều tài nguyên nhưng thông tin biểu diễn được lại rất hạn hẹp.

=> Cần một cách biểu diễn từ ít chiều hơn và mang nhiều thông tin hơn.

4.3.2 Co-occurrence Matrix

Năm 1957, nhà ngôn ngữ học J.R. Firth phát biểu rằng: "Bạn sẽ biết nghĩa của một từ nhờ những từ đi kèm với nó.". Điều này cũng khá dễ hiểu. Ví dụ nhắc đến Việt Nam, người ta thường có các cụm từ quen thuộc như "Chiến tranh Việt Nam", "Cafe Việt Nam", "Việt Nam rừng vàng biển bạc", dựa vào những từ xung quanh ta có thể hiểu hoặc mường tượng ra được "Việt Nam" là gì, như thế nào. Co-occurrence Matrix được xây dựng dựa trên nhận xét trên, co-occurrence đảm bảo quan hệ ngữ nghĩa giữa các từ, dựa trên số lần xuất hiện của các cặp từ trong "context window". Một context window được xác dựa trên kích thước và hướng của nó, ví dụ của context window:

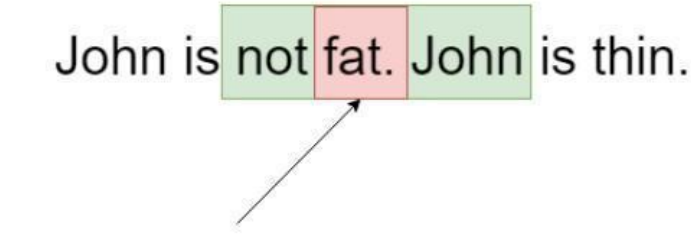


Figure 1-Ví dụ về Context Window

Hình 18.5: Ví dụ về context window

Co-occurrence matrix là một ma trận vuông đối xứng, mỗi hàng, mỗi cột sẽ làm vector đại diện cho từ tương ứng. Từ ví dụ trên ta tiếp tục xây dựng 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

Figure 2-Ví dụ về Co-occurence Matrix

Hình 18.6: Ví dụ về co-occurrence matrix

Trong đó, giá trị tại ô [i, j] là số lần xuất hiện của từ i nằm trong context window của từ j. Cách biểu diễn trên mặc dù đã giữ được thông tin về ngữ nghĩa của một từ, tuy vẫn còn các hạn chế như sau:

- Khi kích thước bộ từ điển tăng, chiều vector cũng tăng theo.

- Lưu trữ co-occurrence matrix cần rất nhiều tài nguyên về bộ nhớ.
- Các mô hình phân lớp bị gặp vấn đề với biểu diễn thưa (có rất nhiều giá trị 0 trong ma trận).

Để làm giảm kích thước của co-occurrence matrix người ta thường sử dụng phép SVD (Singular Value Decomposition) để giảm chiều ma trận. Ma trận thu được sau SVD có chiều nhỏ hơn, dễ lưu trữ hơn và ý nghĩa của từ cũng cô đọng hơn. Tuy nhiên, SVD có độ phức tạp tính toán cao, tăng nhanh cùng với chiều của ma trận ($O(mn^2)$ với m là chiều của ma trận trước SVD, n là chiều của ma trận sau SVD và $n < m$) , ngoài ra phương pháp này cũng gặp khó khăn khi thêm các từ vựng mới vào bộ từ điển.

=> Cần phương pháp khác lưu trữ được nhiều thông tin và vector biểu diễn nhỏ.

4.3.3 Word to vec (Word2vec)

Với tư tưởng rằng ngữ cảnh và ý nghĩa của một từ có sự tương quan mật thiết đến nhau, năm 2013 nhóm của Mikolov đề xuất một phương pháp mang tên Word2vec.

Ý tưởng chính của Word2vec:

- Thay thế việc lưu thông tin số lần xuất hiện của các từ trong context window như co-occurrence matrix, word2vec học cách dự đoán các từ lận cận.
- Tính toán nhanh hơn và có thể transfer learning khi thêm các từ mới vào bộ từ điển.
- Phương pháp:

Với mỗi từ t trong bộ từ điển ta dự đoán các từ lân cận trong bán kính *m* của nó.

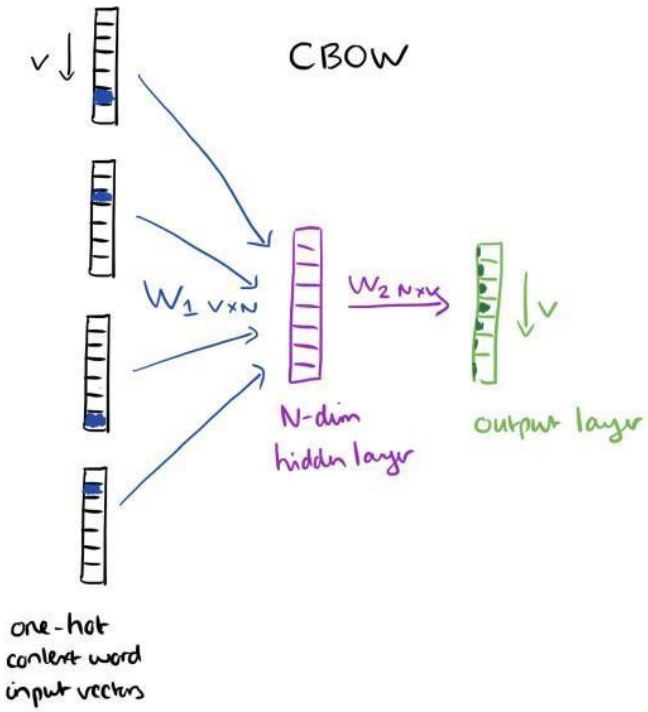
Hàm mục tiêu nhằm tối ưu xác suất xuất hiện của các từ ngữ cảnh (context word) đối với từ đang xét hiện tại:

Có 2 kiến trúc khác nhau của word2vec, là CBoW và Skip-Gram:

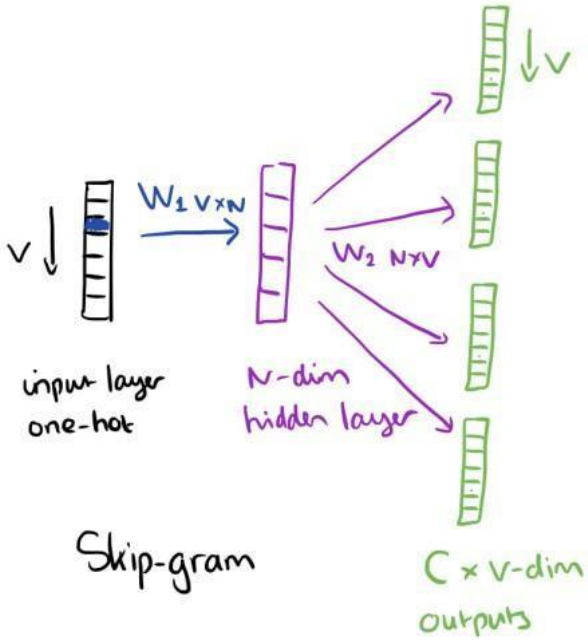
- Cbow: Cho trước ngữ cảnh ta dự đoán xác suất từ đích. Ví dụ: "I ... you", với đầu vào là 2 từ "I" và "you" ta cố gắng dự đoán từ còn thiếu, chẳng hạn "love".
- Skip-Gram: Cho từ đích ta dự đoán xác suất các từ ngữ cảnh (nằm trong context window) của nó. Ví dụ: "... love ...", cho từ "love" ta dự đoán các từ là ngữ cảnh của nó, chẳng hạn "I", "you".

Trong bài báo giới thiệu word2vec, Mikolov và cộng sự có so sánh và cho thấy 2 mô hình này cho kết quả tương đối giống nhau.

Chi tiết mô hình:



Hình 18.7: Mô hình Cbow

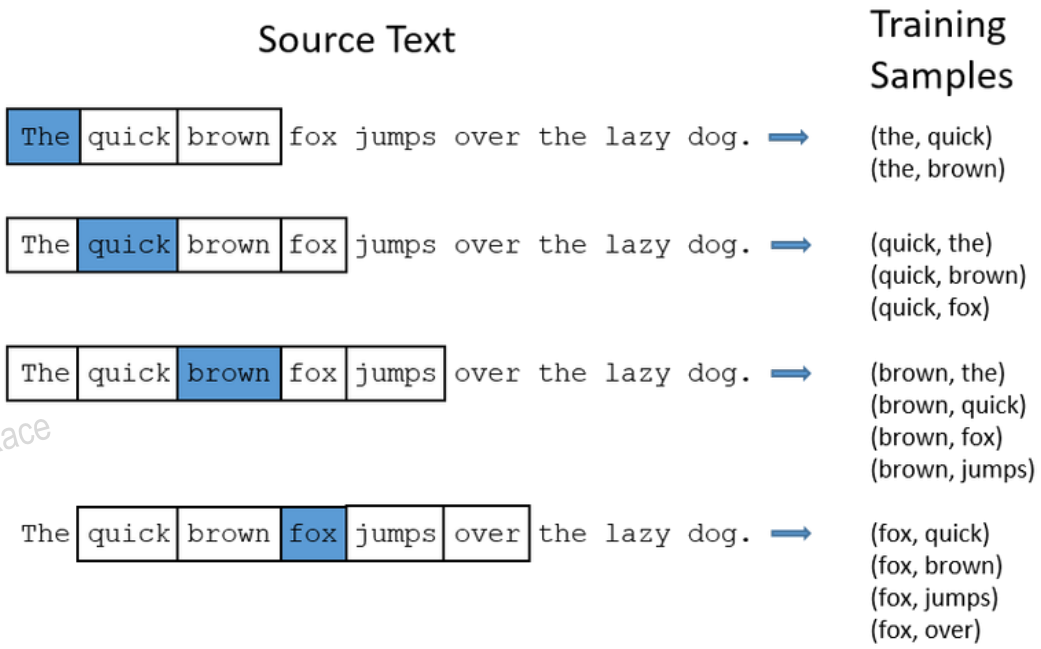


Hình 18.8: Mô hình Skip-Gram

Do 2 kiến trúc khá giống nhau nên ta chỉ thảo luận về Skip-Gram.

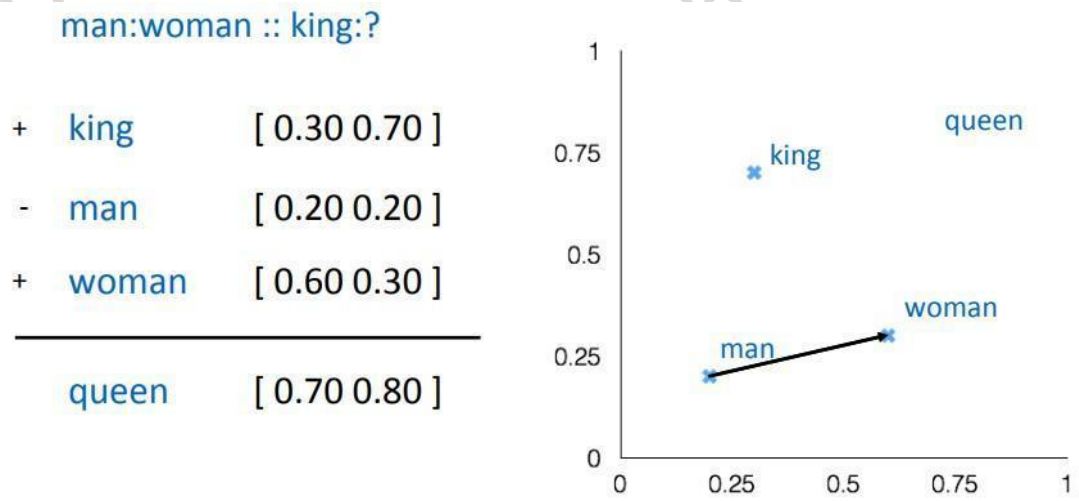
Mô hình Skip-Gram sẽ input từ đích và dự đoán ra các từ ngữ cảnh. Thay vì input từ đích và output ra nhiều từ ngữ cảnh trong 1 mô hình, họ xây dựng model để input từ đích và output ra 1 từ ngữ cảnh.

Mô hình là một mạng neural network 2 lớp, với chỉ 1 hidden layer. Input là một từ trong từ điển đã được mã hóa thành dạng one hot vector chiều $V*1$ với V là kích thước từ điển. Hidden layer không sử dụng activation function có N node, trong đó N chính là độ dài vector embedding của mỗi từ. Output layer có V node, sau đó softmax activation được sử dụng để chuyển về dạng xác suất. Categorical cross entropy loss function được học để dự đoán được từ ngữ cảnh với input là từ đích. Ví dụ của xây dựng training data:



Hình 18.9: Ví dụ của xây dựng training data, window size = 2, tức là lấy 2 từ bên trái và 2 từ bên phải mỗi từ trung tâm làm từ ngữ cảnh (context word)

Một số kết quả của Word2vec:



Hình 18.11: Vị trí các word vector trong không gian

Ví dụ trên là ví dụ kinh điển của Word2vec cho thấy các vector biểu diễn tốt quan hệ về mặt ngữ nghĩa của từ vựng như thế nào.

4.4 Output

Bài toán là dự đoán từ tiếp theo trong chuỗi ở input với ảnh hiện tại, nên output là từ nào trong số 1652 từ trong từ điển mà ta có. Với bài toán phân loại thì softmax activation và categorical_crossentropy loss function được sử dụng.