Mô hình

Mô hình mạng neural bỏ qua gram thực sự đơn giản đáng kinh ngạc ở dạng cơ bản nhất của nó; Tôi nghĩ là do tất cả các chỉnh sửa và cải tiến nhỏ bắt đầu làm xáo trộn lời giải thích.

Hãy bắt đầu với một cái nhìn cấp cao về thứ chúng ta sẽ thảo luận. Word2Vec sử dụng một mẹo bạn có thể đã thấy ở nơi khác trong học máy. Chúng ta sẽ đào tạo một mạng neural đơn giản với một lớp ẩn duy nhất để thực hiện một nhiệm vụ nhất định, nhưng sau đó chúng ta sẽ không sử dụng mạng thần kinh đó cho nhiệm vụ mà chúng ta đã đào tạo! Thay vào đó, mục tiêu thực sự là chỉ để tìm hiểu trọng số của lớp ẩn – chúng ta sẽ thấy rằng những trọng số này thực sự là “vector từ” mà chúng ta đang cố gắng học.

*Một nơi bạn có thể đã thấy thủ thuật này là ở tính năng học không giám sát, nơi bạn luyện tập một tính năng tự động mã hóa để nén một vector đầu vào trong lớp ẩn, và giải nén nó trở lại với bản gốc trong lớp đầu ra. Sau khi đào tạo nó, bạn loại bỏ lớp đầu ra (bước giải nén) và chỉ sử dụng lớp ẩn – đó là một mẹo để học các tính năng hình ảnh tốt mà không có dữ liệu đào tạo được gắn nhãn.*

Nhiệm vụ làm giả

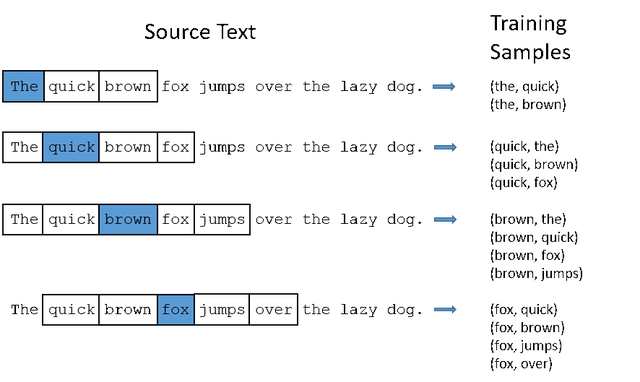
Vì vậy, bây giờ chúng ta cần nói về việc làm “giả” này mà chúng ta sẽ xây dựng mạng neural để thực hiện, và sau đó chúng ta sẽ quay lại làm thế nào gián tiếp lấy lại cho chúng ta những vector từ mà chúng ta thực sự dùng sau đó.

Chúng ta sẽ đào tạo mạng thần kinh để làm như sau. Cho một từ cụ thể ở giữa một câu (từ đầu vào), nhìn vào các từ gần đó và chọn một từ ngẫu nhiên. Mạng sẽ cho chúng ta biết xác suất cho mỗi từ trong từ vựng của chúng ta là “từ gần” mà chúng ta đã chọn.

*Khi tôi nói “gần”, thực sự là có một “window size” cho thuật toán. Kích thước cửa sổ điển hình có thể là 5, có nghĩa là 5 từ phía sau và 5 từ phía trước (tổng cộng 10 từ).*

Các xác suất đầu ra sẽ liên quan đến khả năng nó tìm thấy mỗi từ vựng gần từ đầu vào của chúng ta. Ví dụ, nếu bạn cho mạng được đào tạo từ đầu vào “SoViet”, xác suất đầu ra sẽ cao hơn nhiều với các từ như “Union” và “Russia” so với các từ không liên quan như “watermelon” và “kangaroo”.

Chúng tôi sẽ đào tạo mạng thần kinh để thực hiện điều này bằng cách cho các cặp từ đó tìm thấy trong tài liệu đào tạo của chúng tôi. Ví dụ dưới đây cho thấy một số mẫu huấn luyện (cặp từ), chúng tôi sẽ lấy từ câu “The quick brown fox jumps over the lazy dog.” Tôi đã sử dụng một kích thước cửa sổ nhỏ là 2 chỉ dành riêng cho ví dụ này. Từ đánh dấu màu xanh là từ đầu vào.



Mạng sẽ tìm hiểu số liệu thống kê từ số lần mỗi cặp hiển thị. Vì vậy, ví dụ, mạng có lẽ sẽ nhận được nhiều mẫu đào tạo (“Soviet”, “Union”) hơn là (“Soviet”, “Sasquatch”). Khi đào tạo xong, nếu bạn cho nó từ “Soviet” làm đầu vào, thì nó sẽ tạo ra xác suất cao hơn cho “Union” hay “Russia” hơn là “Sasquatch”.

Chi tiết mô hình

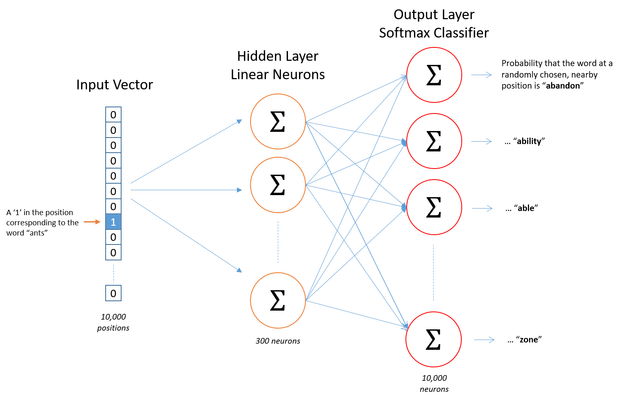
Vậy tất cả điều này được thể hiện như thế nào?

Trước hết, bạn biết rằng bạn không thể nạp một từ giống như chuỗi văn bản vào mạng neural, vì vậy chúng tôi cần một cách để thể hiện các từ đó với mạng. Để làm điều này, trước tiên chúng tôi xây dựng vốn từ vựng của các từ trong tài liệu đào tạo của chúng tôi – giả sử chúng tôi có từ vựng gồm 10.000 từ không trùng lặp.

Chúng tôi sẽ đại diện cho một từ đầu vào như “ants” như một vector one-hot. Vector này sẽ có 10.000 thành phần (một cho mỗi từ trong từ vựng của chúng tôi) và chúng tôi sẽ đặt “1” ở vị trí tương ứng với từ “ants” và 0 ở tất cả các vị trí khác.

Đầu ra của mạng là một vector đơn (cũng với 10.000 thành phần) có chứa, cho mỗi từ trong từ vựng của chúng ta, xác suất mà từ gần đó được chọn là quy tắc từ vựng.

Đây là kiến trúc mạng neural của chúng tôi.



Không có activation function trên các neuron lớp ẩn, nhưng các neuron đầu ra sử dụng softmax. Chúng ta sẽ quay lại sau.

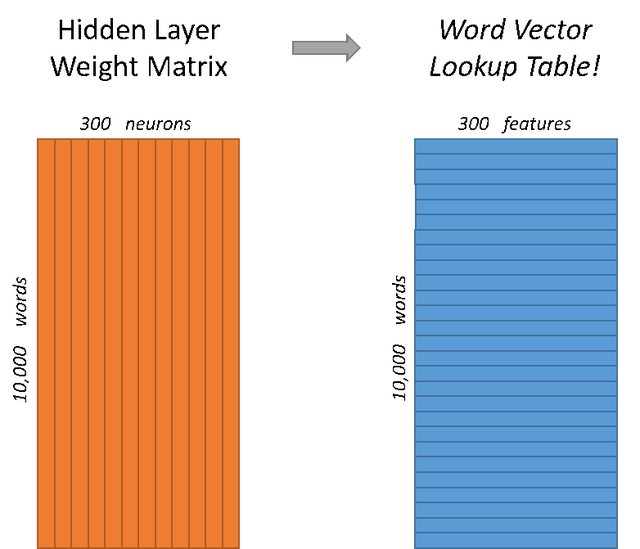
Khi đào tạo mạng này trên các cặp từ, đầu vào là một véc tơ một nóng đại diện cho từ đầu vào và đầu ra đào tạo cũng là một vector one hot đại diện cho từ đầu ra. Nhưng khi bạn đánh giá mạng được đào tạo trên một từ đầu vào, vector đầu ra sẽ thực sự là một phân bố xác suất (nghĩa là, một loạt các giá trị dấu chấm động, không phải là một vector one-hot).

Lớp ẩn

Ví dụ của chúng ta, chúng ta sẽ nói rằng chúng ta đang học vectơ từ với 300 đặc trưng. Vì vậy, các lớp ẩn sẽ được đại diện bởi một ma trận trọng số với 10.000 hàng (một cho mỗi từ trong từ vựng của chúng tôi) và 300 cột (một cho mỗi tế bào thần kinh ẩn).

*300 đặc trưng là những gì Google sử dụng trong mô hình được xuất bản của họ được đào tạo trên bộ dữ liệu tin tức của Google (bạn có thể tải xuống từ đây). Số lượng các đặc trưng là một “siêu tham số” mà bạn sẽ cần điều chỉnh cho ứng dụng của bạn (có nghĩa là, hãy thử các giá trị khác nhau và xem những gì mang lại kết quả tốt nhất).*

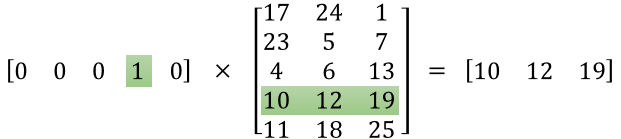
Nếu bạn nhìn vào các hàng của ma trận trọng số này, đây thực sự sẽ là vector từ của chúng ta!



Vì vậy, mục tiêu cuối cùng của tất cả điều này thực sự chỉ là để tìm hiểu ma trận trọng số lớp ẩn này – lớp đầu ra mà chúng ta sẽ chỉ đưa ra khi hoàn tất!

Tuy nhiên, hãy quay trở lại để làm việc với định nghĩa về mô hình mà chúng ta sẽ đào tạo.

Bây giờ, bạn có thể tự hỏi mình – “Đó là vector one-hot gần như tất cả các số là không … hiệu ứng của nó là gì?” Nếu bạn nhân một vectơ one-hot 1 x 10.000 bằng một ma trận 10.000 x 300, nó sẽ chỉ cần chọn hàng ma trận tương ứng với “1”. Dưới đây là một ví dụ nhỏ để cung cấp cho bạn hình ảnh. Ảnh hưởng của phép nhân ma trận với vector one-hot



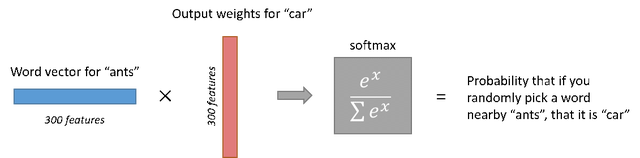
Điều này có nghĩa là lớp ẩn của mô hình này thực sự chỉ hoạt động như một bảng tra cứu. Đầu ra của lớp ẩn chỉ là “word vector” cho từ đầu vào.

Lớp đầu ra

Vector word 1 x 300 cho “ants” sau đó được nạp vào lớp đầu ra. Lớp đầu ra là một trình phân cụm hồi quy softmax. Có một hướng dẫn sâu về Softmax Regression ở đây, nhưng ý chính của nó là mỗi nơron đầu ra (mỗi từ trong từ vựng của chúng ta!) Sẽ tạo ra một đầu ra giữa 0 và 1, và tổng của tất cả các giá trị đầu ra này sẽ cộng lại được 1.

Cụ thể, mỗi nơron đầu ra có một vector trọng số mà nó nhân với vector từ ở lớp ẩn, sau đó nó áp dụng hàm exp(x) cho kết quả. Cuối cùng, để có được kết quả đầu ra tổng là 1, chúng tôi chia kết quả này cho tổng tất cả kết quả 10.000 nút đầu ra.

Dưới đây là một minh họa về tính toán kết quả của neuron đầu ra cho từ “car”.



*Lưu ý rằng mạng nơron không biết gì về độ lệch của từ đầu ra liên quan đến từ đầu vào. Nó không học được xác suất khác nhau cho từ đầu vào trước so với từ sau. Để hiểu được ý nghĩa, hãy nói rằng trong kho dữ liệu đào tạo của chúng tôi, mỗi lần xuất hiện duy nhất của từ ‘York’ được bắt đầu bằng từ ‘New’. Đó là, ít nhất là theo các dữ liệu đào tạo, có một xác suất 100% rằng ‘New’ sẽ nằm trong vùng lân cận của ‘York’. Tuy nhiên, nếu chúng ta lấy 10 từ trong vùng lân cận của ‘York’ và chọn ngẫu nhiên một trong số chúng, xác suất của nó là ‘New’ không phải là 100%; bạn có thể đã chọn một trong những từ khác trong vùng lân cận.*

Trực giác

Ok, bạn đã sẵn sàng cho một chút hiểu biết thú vị về mạng này chưa?

Nếu hai từ khác nhau có ngữ cảnh rất giống nhau (nghĩa là, những từ nào có khả năng xuất hiện xung quanh), thì mô hình của chúng ta cần đưa ra kết quả rất giống nhau cho hai từ này. Và một cách để mạng xuất ra các dự đoán ngữ cảnh tương tự cho hai từ này là các vector tương tự nhau. Vì vậy, nếu hai từ có ngữ cảnh tương tự, thì mạng của chúng ta được thúc đẩy để học các vectơ từ tương tự cho hai từ này! Ta da!

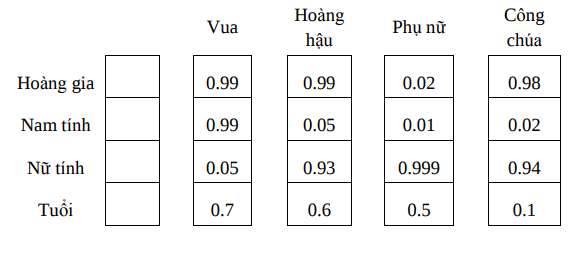
Và hai từ có ngữ cảnh tương tự có ý nghĩa gì? Tôi nghĩ bạn có thể mong rằng các từ đồng nghĩa như ““intelligent” and “smart” sẽ có những ngữ cảnh rất giống nhau. Hoặc những từ đó có liên quan, như “engine” and “transmission”, có lẽ cũng có những ngữ cảnh tương tự.

Điều này cũng có thể xử lý bắt nguồn từ bạn – mạng có khả năng sẽ tìm hiểu các vector tương tự cho các từ “ant” và “ants” bởi vì chúng phải có ngữ cảnh tương tự.

Thay vì đếm và xây dựng ma trận đồng xuất hiện, word2vec học trực tiếp word vector có số chiều thấp trong quá trình dự đoán các từ xung quanh mỗi từ. Đặc điểm của phương pháp này là nhanh hơn và có thể dễ dàng kết hợp một câu một văn bản mới hoặc thêm vào từ vựng.  
  
Word2vec là một mạng neural 2 lớp với duy nhất 1 tầng ẩn, lấy đầu vào là một corpus lớn và sinh ra không gian vector(với số chiều khoảng vài trăm), với mỗi từ duy nhất trong corpus được gắn với một vector tương ứng trong không gian.

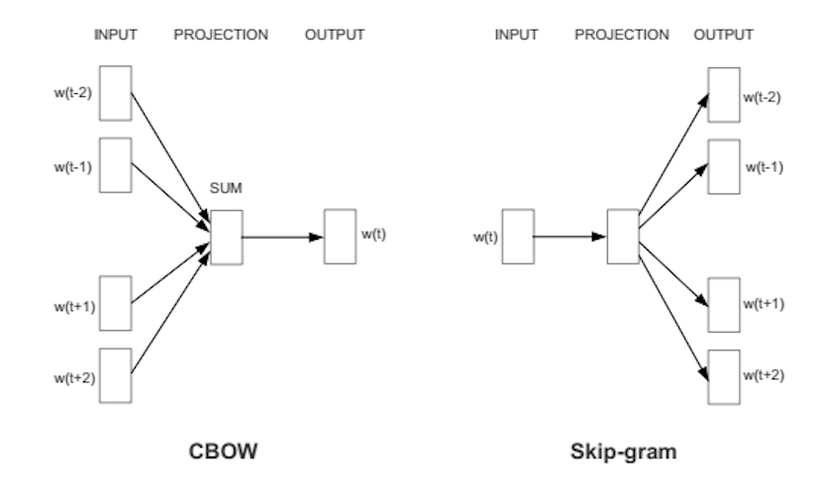
Các word vectors được xác định trong không gian vector sao cho những từ có chung ngữ cảnh trong corpus được đặt gần nhau trong không gian. Dự đoán chính xác cao về ý nghĩa của một từ dựa trên những lần xuất hiện trước đây.

Nếu ta gán nhãn các thuộc tính cho một vector từ giả thiết, thì các vector được biểu diễn theo word2vec sẽ có dạng như sau:



Có 2 cách xây dựng word2vec:

* Sử dụng ngữ cảnh để dự đoán mục tiêu(CBOW).
* Sử dụng một từ để dự đoán ngữ cảnh mục tiêu(skip-gram)(cho kết quả tốt hơn với dữ liệu lớn).



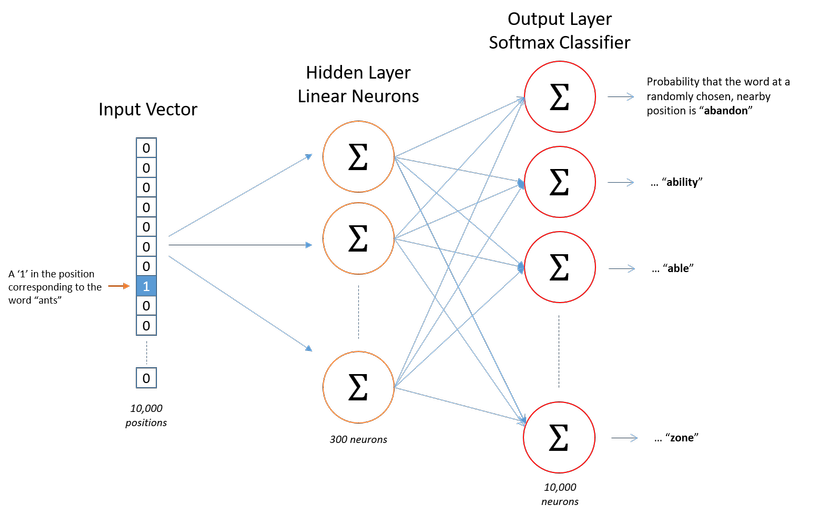
Do những ưu điểm của Skip-gram Model vượt trội hơn người anh em của nó, nên trong bài viết này, mình chỉ đi trọng tâm vào mô hình này.

#### Skip-gram Model

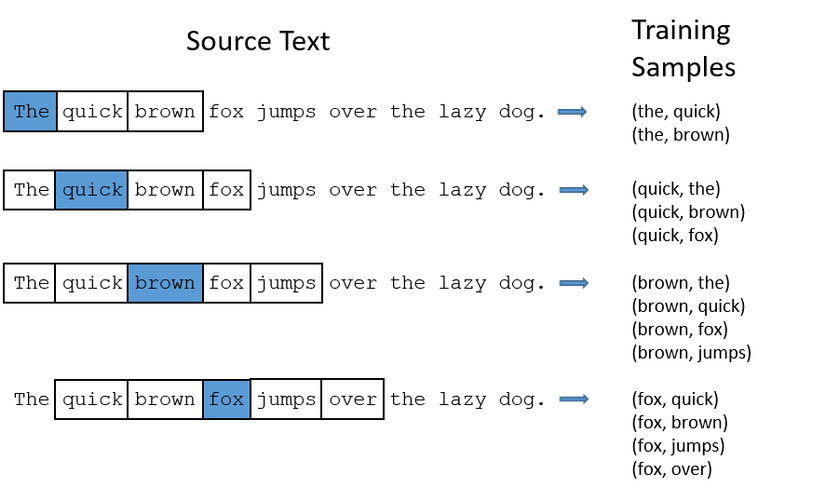
**Mục tiêu**: Học trọng số các lớp ẩn, các trọng số này là các words vector

**Cách thức**: Cho một từ cụ thể ở giữa câu(input word), nhìn vào những từ ở gần và chọn ngẫu nhiên. Mạng neural sẽ cho chúng ta biết xác suất của mỗi từ trong từ vựng về việc trở thành từ gần đó mà chúng ta chọn.

Dưới đây là mô hình kiến trúc của mạng Skip-gram và cách xây dựng training data.



**Ví dụ:** Xây dựng training data với windows size = 2. Ở đây windows được hiểu như một cửa sổ trượt qua mỗi từ. Windows size = 2 tức là lấy 2 từ bên trái và bên phải mỗi từ trung tâm.



**Model details**

* Xây dựng bộ từ vựng
* Biểu diễn mỗi từ thành các one-hot-vector
* Đầu ra là một vector duy nhất, có kích thước bằng kích thước của bộ từ vựng, thể hiện xác suất của mỗi từ được là lân cận của từ đầu vào.
* Không có hàm kích hoạt trên tầng ẩn
* Hàm kích hoạt trên tầng output là softmax
* Trong quá trình training, input là 1 one-hotvector, ouput cũng là 1 one-hot-vector
* Trong quá trình đánh giá sau khi training, đầu ra phải là 1 phân bố xác suất.

**Vấn đề:**

* Mạng Neural lớn: Giả sử words vector với 300 thuộc tính, và từ vựng là 10k từ. Mạng neural có ma trận trọng số lớn, kích thước của ma trận trọng số là 300\*10000 bằng 3 triệu giá trị.
* Chạy Gradient Descent sẽ rất chậm.

#### Word2vec cải tiến

Có 3 cải tiến cơ bản cho mô hình word2vec truyền thống:

* Xử lý các cặp từ thông dụng hoặc cụm từ như là một từ đơn
* Loại bỏ các từ thường xuyên lặp lại để giảm số lượng các ví dụ huấn luyện
* Sửa đổi mục tiêu tối ưu hóa bằng một kỹ thuật gọi là “Negative Sampling”.

**Cải tiến 1:** Xử lý cụm từ như một từ đơn

Ví dụ các từ như “thành\_phố\_Cảng” có nghĩa khác nhau với từng từ “thành\_phố” và “cảng”,...

Chúng ta sẽ coi như đó là một từ duy nhất, với word vector của riêng mình.

Điều này sẽ làm tăng kích thước từ vựng. (Tìm hiểu thêm về word2phrase)  
  
**Cải tiến 2:** Loại bỏ các từ thường xuyên lặp lại

Các từ thường xuyên lặp lại như “các”, “những”,… không cho chúng ta biết thêm nhiều hơn về ý nghĩa của những từ đi kèm nó, và chúng cũng xuất hiện trong ngữ cảnh của khá nhiều từ.

Chúng ta sẽ xác định xác suất loại bỏ, giữ lại một từ trong từ vựng thông qua tần suất xuất hiện của nó.  
  
**Cải tiến 3:** Negative Sampling

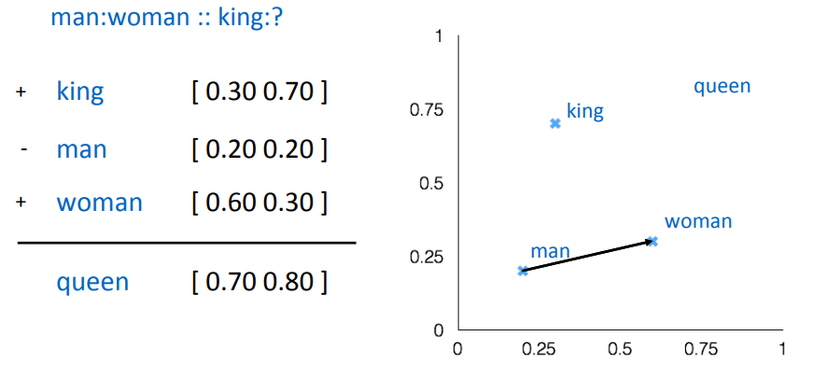
Mỗi mẫu huấn luyện chỉ thay đổi một tỷ lệ phần trăm nhỏ các trọng số, thay vì tất cả chúng.

Nhớ lại: Khi huấn luyện mạng với 1 cặp từ, đầu ra của mạng sẽ là 1 one-hot vector, neural đúng thì đưa ra 1 còn hàng ngàn neural khác thì đưa ra 0.

Chọn ngẫu nhiên 1 số lượng nhỏ các neural “negative” kết hợp với neural “positive” để cập nhật trọng số.(chọn là 5-20 hoạt động tốt với các bộ dữ liệu nhỏ, 2-5 với bộ dữ liệu lớn).

#### Kết quả thú vị của Word2Vec:

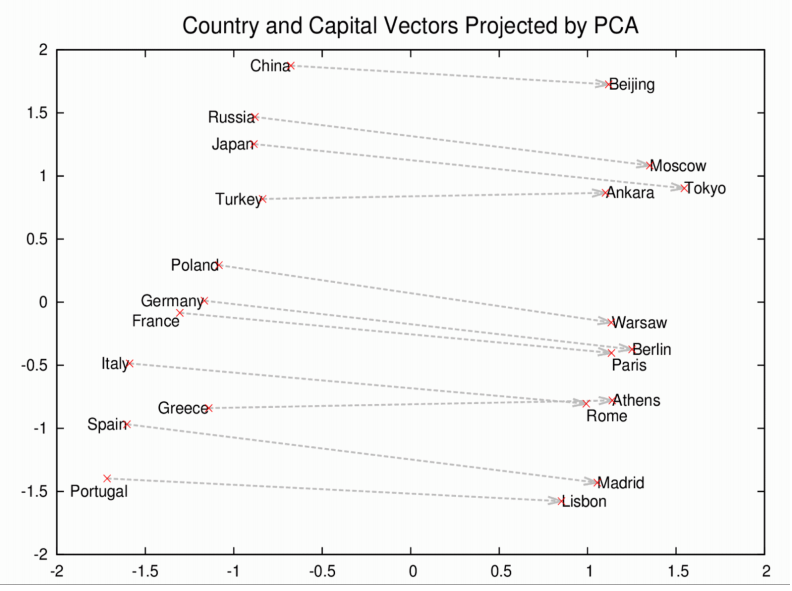
Chúng ta cùng phân tích qua một vài kết quả thú vị sử dụng mô hình word2vec. Ví dụ dưới đây là một ví dụ cực kỳ kinh điển của mô hình word2vec. Word vector học từ mô hình word2vec rất phù hợp để tìm ra quan hệ giữa các từ về mặt ngữ nghĩa. Chúng ta có thể tìm mối quan hệ giữa các từ thông qua các phép toán vector.



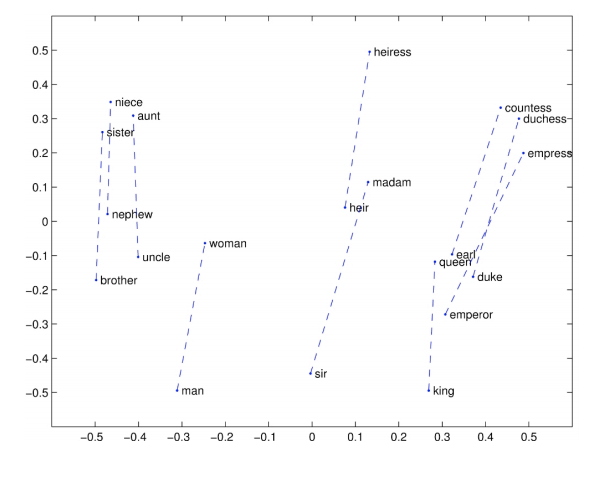
Word vector học từ mô hình word2vec rất phù hợp để trả lời cho câu hỏi:

***Nếu A là B thì C là ...***

Như ví dụ dưới đây, nếu thủ đô của China là Bắc Kinh thì thủ đô của Nga là thành phố Moscow,...



Hay như trong trường hợp này là mối quan hệ về giới tính. Các kết quả này hoàn toàn lấy từ mô hình word2vec.



Trong phần cuối của bài viết này, chắc hẳn các bạn đang nghĩ rằng tại sao chúng ta phải train lại mô hình trong khi đã có các mô hình có sẵn đúng không?

Đơn giản là trong nhiều bài toán, chúng ta train dữ liệu trong một miền cụ thể sẽ cho kết quả tốt hơn nhiều so với dùng mô hình train sẵn. Ví dụ trong bài toán tính độ tương đồng giữa các quốc gia, mình đã thử so sánh việc dùng pre-train model có sẵn của tác giả [Trần Việt Trung](https://github.com/trungtv/vivi_spacy) và việc train dữ liệu chỉ từ các bài báo quốc tế. Kết quả là mô hình mình train lại tốt hơn do word vector được học từ dữ liệu phù hợp với bài toán hơn