Cao Chánh Dương @caochanhduong Follow

★ 52 2 1

Published Feb 10th, 2019 6:34 PM - 14 min read

5.2K 3 4

SƠ LƯỢC WORD EMBEDDING

Machine Learning Deep Learning

- Tác giả: Cao Chánh Dương.
- Sinh viên năm 3 Khoa Khoa học và kĩ thuật Máy tính Đại học Bách khoa TPHCM.
- Deep Research Atomic Group Of New-Age (DRAGON).

1. Khái niệm

Word Embedding là tên gọi chung của các mô hình ngôn ngữ và các phương pháp học theo đặc trưng trong Xử lý ngôn ngữ tự nhiên(NLP), ở đó các từ hoặc cụm từ được ánh xạ sang các vector số (thường là số thực). Đây là một công cụ đóng vai trò quan trọng đối với hầu hết các thuật toán, kiến trúc Machine Learning, Deep Learning trong việc xử lý Input ở dạng text, do chúng chỉ có thể hiểu được Input ở dạng là số, từ đó mới thực hiện các công việc phân loại, hồi quy,vv...

2. Các loại Word Embedding

Word Embedding được phân chủ yếu thành 2 loại:

- Frequency-based embedding.
- Prediction-based embedding.

2.1 Frequency Base 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.
- tf-idf Vector.
- Co-occurrence Matrix.

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***

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.

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:

$$tf_i = \frac{n_i}{N_i}$$

Trong đó:
i : 1 .. D
n_i : tần số xuất hiện của từ trong văn bản i.
N_i : tổng số từ trong văn bản i.

$$idf_i = \log_2 \frac{D}{d}$$

Trong đó:
D: tổng số documents trong tập dữ liệu.
d: số lượng documents có sự xuất hiện của từ.

$$tfidf_i = tf_i \times idf_i$$

Như đã đề cập ở trên, ***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*** 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***:

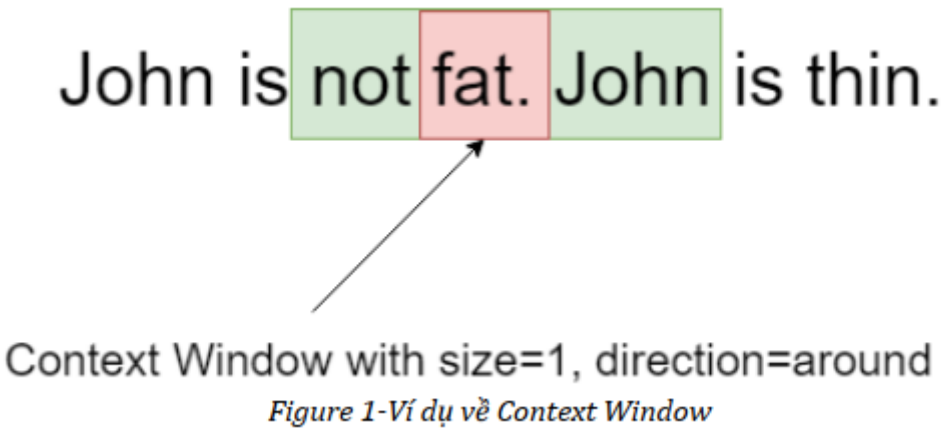

Figure 1-Ví dụ về Context Window

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

Figure 2-Ví dụ về Co-occurence Matrix

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 (Global Vector) là một trong những phương pháp mới để xây dựng vec-tơ từ (được giới thiệu vào năm 2014), nó thực chất được xây dựng dựa trên ***Co-occurrence Matrix***. ***GloVe*** có bản chất là xác suất, ý tưởng xây dựng phương pháp này đến từ tỉ số sau:

$$\frac{P(k|i)}{P(k|j)} \qquad (1)$$

Trong đó:

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

Công thức của ***P(k|i)*** :

$$P(k|i) = \frac{X_{ik}}{X_i} = \frac{X_{ik}}{\sum_m X_{im}}$$

Trong đó:

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

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

(Các giá trị này chính là các mục nhập của ***Co-occurrence Matrix***)

Ý tưởng chính của ***GloVe***: độ tương tự ngữ nghĩa giữa hai từ ***i***, ***j*** có thể 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 ***(1)*** >>1 hoặc xấ

là “cat”, ở trường hợp khác, nếu ta thay ***k*** là “ice cream” thì ***(1)*** sẽ xấp xỉ bằng 1 do “ice cream” hầu như chẳng lên quan gì tới “table” và “cat”.
Dựa trên tầm quan trọng của ***(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 với ***(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 minimum ***cost function*** sau:

$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2$$

Trong đó:

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 ***Co-occurrence Matrix***.

Hàm ***f*** được gọi là ***weighting function***, được thêm vào để giảm bớt sự ảnh hưởng của các cặp từ xuất hiện quá thường xuyên, hàm này thỏa 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:

$$f(x) = \begin{cases} (x/x_{\max})^{\alpha} & \text{if } x < x_{\max} \\ 1 & \text{otherwise} \end{cases}$$

  Với α=3/4

  Việc thực hiện minimum ***cost function J*** để tìm ra các vec-tơ từ ***wi***, ***wj*** thể được thực hiện bằng nhiều cách, trong đó cách tiêu chuẩn nhất là sử dụng ***Gradient Descent***.

2.2 Prediction Base Embedding

Prediction-based Embedding xây dựng các vector từ dựa vào các mô hình dự đoán. Tiêu biểu nhất chính là ***Word2vec***, nó là sự kết hợp của 2 mô hình: ***CBOW (Continous Bag Of Words)*** và ***Skip-gram***. Cả hai mô hình này đều được xây dựng dựa trên một mạng neuron gồm ***3*** lớp:1 ***Input Layer***,1 ***Hidden Layer*** và 1 ***Output Layer***. Mục đích chính của các mạng neuron này là học các trọng số biểu diễn vector từ.

(với ***V*** là độ lớn của từ điển), output sẽ là một vector xác suất cũng với chiều dài ***V*** của từ liên quan hoặc còn thiếu,

Hidden Layer có chiều dài ***N***, ***N*** cũng chính là độ lớn của vector từ biểu thị. Dưới đây là mô hình ***CBOW*** với ngữ cảnh là 1 từ đơn:

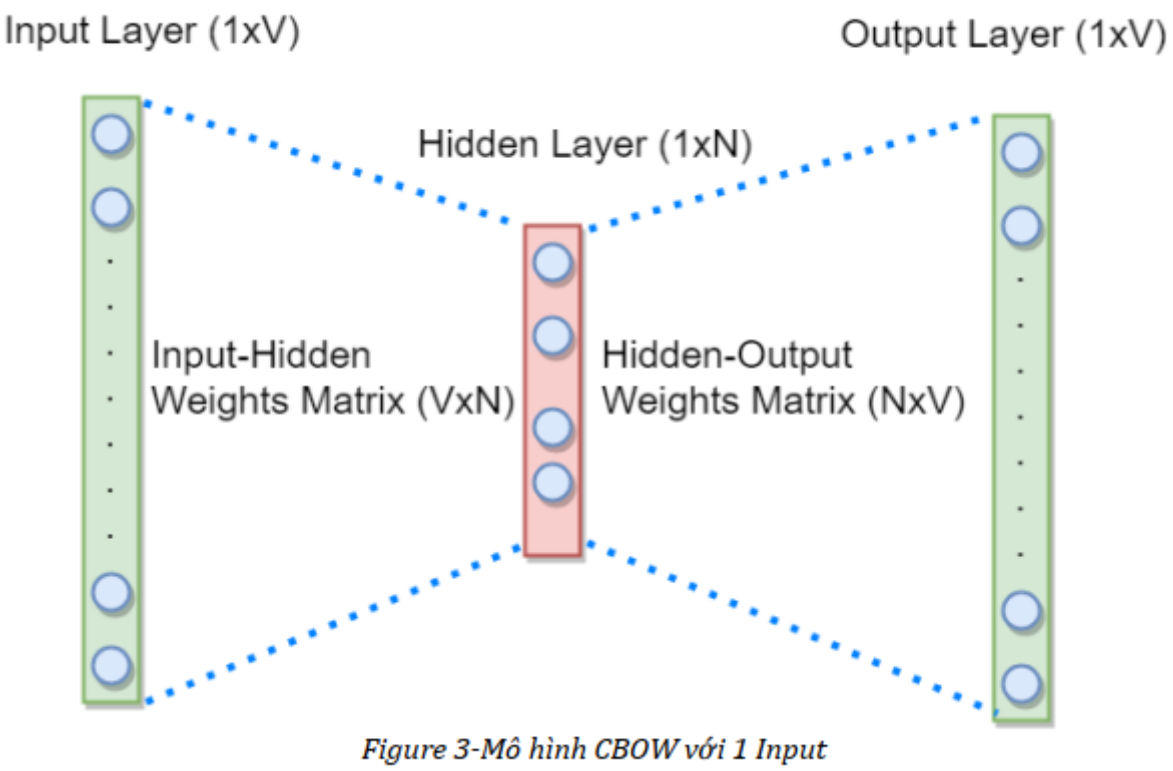

Figure 3-Mô hình CBOW với 1 Input

Về bộ dữ liệu dùng để train, Input sẽ bao gồm các bộ ***One-hot vectors*** ngữ cảnh và các ***One-hot vectors*** của từ mong muốn.

Về cách thức hoạt động, ban đầu hai ma trận trọng số ***Input-Hidden Weights Matrix*** và ***Hidden-Output Weights Matrix*** được khởi tạo ngẫu nhiên, Input sẽ được nhân với ***Input-Hidden Weights Matrix*** ra được một kết quả gọi là ***Hidden Activation***, kết quả này sẽ được nhân tiếp với ***Hidden-Output Weights Matrix*** và cuối cùng được đưa vào một hàm ***softmax*** để ra được Output là 1 vector xác suất, Output này sẽ được so sánh với Output mong muốn và tính toán ***độ lỗi***, dựa vào ***độ lỗi*** này mà mạng neuron sẽ lan truyền ngược trở lại để cập nhật các giá trị của các ma trận trọng số. Đối với mô hình ***CBOW*** nhiều Input, các thức hoạt động là tương tự, chỉ khác ở chỗ các kết quả thu được khi nhân các Input với ***Input-Hidden Weights Matrix*** sẽ được lấy trung bình để ra được ***Hidden Activation*** cuối cùng. Các trọng số của ***Hidden-Output Weights Matrix*** sau khi học xong sẽ được lấy làm biểu diễn của các vector từ.

Mô hình ***Skip-gram*** có cấu trúc tương tự như ***CBOW***, nhưng mục đích của nó là dự đoán ngữ cảnh của một từ đưa vào. Dưới đây là hình ảnh của mô hình ***Skip-gram***:

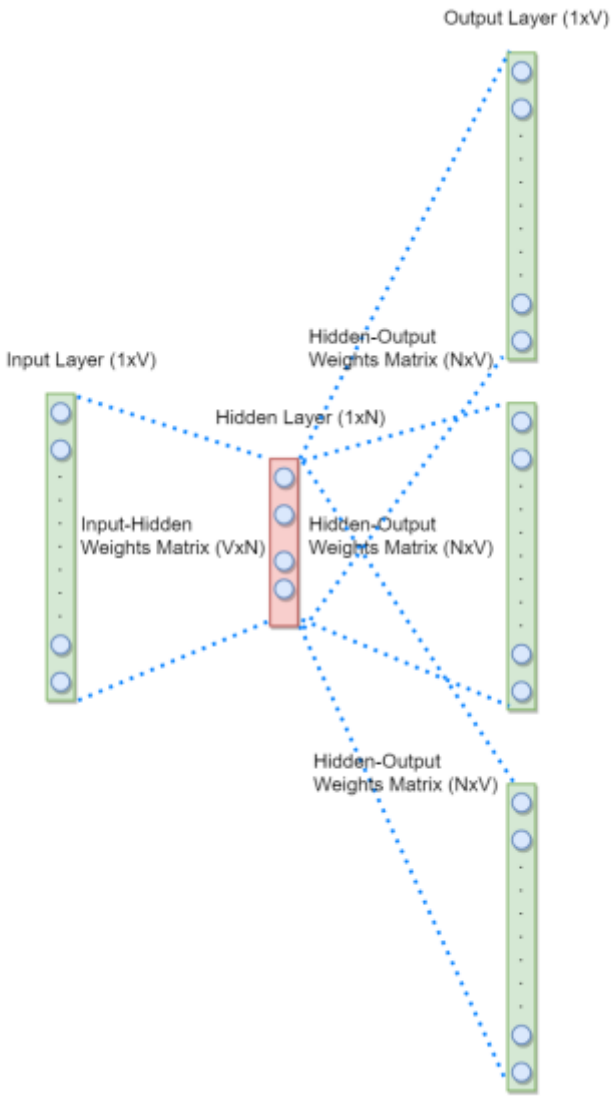

Figure 4-Mô hình Skip-gram

Về bộ dữ liệu dùng để train cũng như cách thức hoạt động của mô hình ***Skip-gram*** hoàn toàn tương tự với mô hình ***CBOW*** 1 Input, chỉ khác ở điểm thay vì chỉ có 1 ***độ lỗi***, ta có nhiều ***độ lỗi*** do có nhiều vector Output, các độ lỗi này sẽ được tổng hợp lại thành 1 ***độ lỗi*** cuối cùng để lan truyền ngược trở lại cập nhật các trọng số. Các trọng số của ***Input-Hidden Weights Matrix*** sau khi học xong sẽ được lấy làm biểu diễn của các vector từ.

Ưu điểm của ***CBOW*** là nó không tốn nhiều bộ nhớ để lưu trữ các ma trận lớn, cũng như do nó có bản chất là xác suất cho nên việc hiện thực được cho là vượt trội hơn so với phương pháp khác, tuy nhiên vẫn còn tồn tại khuyết điểm các từ giống nhau nhưng nghĩa khác nhau vẫn chỉ được biểu diễn bằng 1 vec-tơ từ duy nhất.

3. Một chút so sánh giữa Word2vec và GloVe

Về bản chất, rõ ràng ***Word2vec*** và ***GloVe*** khác nhau do thuộc 2 loại ***Embedding*** khác nhau nhưng đều bắt nguồn từ ***Context Window***, ***Word2vec*** sử dụng ***Context Window*** để tạo ra các tập train cho mạng neuron còn ***GloVe*** sử dụng nó để tạo ra ***Co-occurrence Matrix***. Để ý kĩ một chút, ta thấy rằng ***GloVe*** mang tính "toàn cục" hơn là ***Word2vec*** vì ***GloVe*** tính toán xác suất từ dựa trên ***toàn bộ*** tập dữ liệu còn ***Word2vec*** học dựa trên các ngữ cảnh ***đơn lẻ***, cũng chính vì lý do này mà ***GloVe*** có trội hơn ***Word2vec*** cũng như vài mô hình khác trong một số task về ngữ nghĩa, nhận dạng thực thể có gắn tên,vv... Ngoài ra, ***GloVe*** có ***độ ổn định trung bình*** tốt hơn ***Word2vec***, ***độ ổn định*** ở đây chính là độ biến thiên của kết quả giữa hai lần ta thực hiện việc học với cùng một điều kiện xác định (cùng bộ dữ liệu, cùng

4. Ứng dụng

Related

[Một vài hiểu nhầm khi mới học Machine Learning](#)

Phạm Văn Toàn
15 min read
9851 12 3 28

[10 thuật toán học máy mà các kỹ sư cần biết](#)

Vương Hưng
16 min read
9595 11 1 8

[Recurrent Neural Network(Phần 1): Tổng quan và ứng dụng](#)

Do Duong
9 min read
8334 3 2 9

[Một số phương pháp làm mịn trong mô hình trong mô hình N-gram](#)

Nguyễn Phương Lan
10 min read
1728 3 0 5

Comments

Login to comment

Cao Chánh Dương @caochanhduong — Feb 11th, 2019 9:08 PM

Mình vừa cập nhật thêm mô hình GloVe cũng như thêm phần so sánh giữa GloVe và Word2vec. Cảm ơn các bạn đã xem.

0 | Reply Share

Phát Võ @tamio — Mar 2nd, 2019 4:37 PM

Các trọng số của Hidden-Output Weights Matrix sau khi học xong sẽ được lấy làm biểu diễn của các vector từ.

Input-Hidden mới đúng nhé bạn

0 | Reply Share

Cao Chánh Dương @caochanhduong — Mar 3rd, 2019 9:16 PM

Theo như mình tìm hiểu thì Skip-gram mới lấy Input-Hidden còn CBOW thì lấy Hidden-Output ấy bạn. Cảm ơn bạn đã góp ý.

0 | Reply Share