Co-occurrence matrix & Singular Value Decomposition (SVD)

Apar Garg Follow
Mar 14, 2020 · 6 min read

What are Word Embeddings?

Word Embeddings are the texts converted into numbers. There may be different numerical representations of the same text.

Why do we need Word Embeddings?

Many Machine Learning algorithms and almost all Deep Learning Architectures are incapable of processing *strings* or *plain text* in their raw form. They require numbers as inputs to perform any sort of job, be it classification, regression, etc. in broad terms.

What are the different types of Word Embeddings?

The different types of word embeddings can be broadly classified into two categories-

Frequency-based Embedding

There are generally three types of vectors that we encounter under this category: Count Vector, TF-IDF Vector, Co-Occurrence Matrix with a fixed context window.

Prediction-based Embedding

There are generally two types of vectors that we encounter under this category: Continuous Bag of Words(CBOW), Skip-Gram.

In this article, we'll be focusing on the Co-Occurrence Matrix with a fixed context window(one of the Frequency-based techniques).

Before diving into the Co-Occurrence Matrix with a fixed context window, let's discuss the disadvantages of the other two Frequency-based techniques.

Count Vector and TF-IDF do not capture the position in semantics, co-occurrences in the document, etc.

Co-Occurrence Matrix with a fixed context window

The big idea — Similar words tend to occur together and will have a similar context for example — Apple is a fruit. Mango is a fruit.

Apple and mango tend to have a similar context i.e fruit.

- Co-occurrence — For a given corpus, the co-occurrence of a pair of words say w1 and w2 is the number of times they have appeared together in a Context Window.
- Context Window — Context window is specified by a number and the direction.

How to form the Co-occurrence matrix:

- The matrix A stores co-occurrences of words.
- In this method, we count the number of times each word appears inside a window of a particular size around the word of interest.
- Calculate this count for all the words in the corpus.

Let us understand all of this with the help of an example.

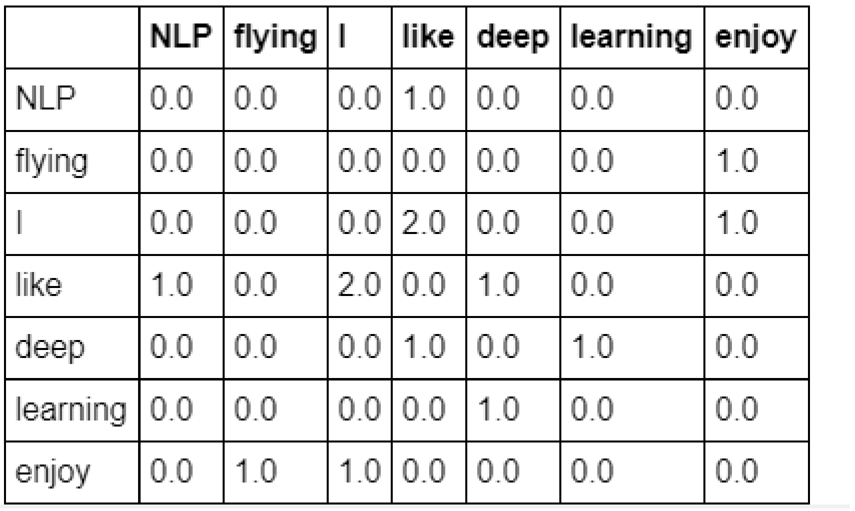
Let our corpus contain the following three sentences:

1. I enjoy flying
2. I like NLP
3. I like deep learning

Let window size =1. This means that context words for each and every word are 1 word to the left and one to the right. Context words for:

- I = enjoy(1 time), like(2 times)
- enjoy = I (1 time), flying(2 times)
- flying = enjoy(1 time)
- like = I(2 times), NLP(1 time), deep(1 time)
- NLP = like(1 time)
- deep = like(1 time), learning(1 time)
- learning = deep(1 time)

Therefore, the resultant co-occurrence matrix A with fixed window size 1 looks like :



	NLP	flying	I	like	deep	learning	enjoy
NLP	0.0	0.0	0.0	1.0	0.0	0.0	0.0
flying	0.0	0.0	0.0	0.0	0.0	0.0	1.0
I	0.0	0.0	0.0	2.0	0.0	0.0	1.0
like	1.0	0.0	2.0	0.0	1.0	0.0	0.0
deep	0.0	0.0	0.0	1.0	0.0	1.0	0.0
learning	0.0	0.0	0.0	0.0	1.0	0.0	0.0
enjoy	0.0	1.0	1.0	0.0	0.0	0.0	0.0

Co-occurrence matrix A

Problem — For a huge corpus, this co-occurrence matrix could become really complex (high-dimension).

Solution — Singular value decomposition(SVD) and principal component analysis(PCA) are two eigenvalue methods used to reduce a high-dimensional dataset into fewer dimensions while retaining important information.

Singular values of a matrix

Let A be an m*m matrix. The product $A^T A$ is a symmetric matrix.

$$[(A^t A)^t = A^t.(A^t)^t = A^t A]$$

Thus $A^T A$ has n Linearly Independent eigenvectors $v_1, v_2 \ldots v_n$ and real eigenvalues $\lambda_1, \lambda_2 \ldots \lambda_n$.

We know that the eigenvalues of $A^T A$ are also all non-negative.

Let λ be an eigenvalue of $A^T A$ with corresponding eigenvector v. Then,

$$\begin{aligned}\lambda = \lambda \|v\|^2 = \lambda v^t v = v^t \lambda v &= v^t A^t A v \\ &= (Av)^t Av \\ &= \|Av\|^2 \geq 0\end{aligned}$$

.

Label the eigenvectors $v_1, v_2 \ldots v_n$ so that $\lambda_1 \geq \lambda_2 \geq \ldots \lambda_n$. Let $\sigma_i = \sqrt{\lambda_i}$.

Thus, $\sigma_1 \geq \sigma_2 \geq \ldots \sigma_n \geq 0$. The numbers $\sigma_1, \sigma_2, \ldots \sigma_n$ are called the singular values of matrix A.

Singular Value Decomposition (SVD)

Let A be an m*n matrix. Then there exists a factorization of A,

$$A = U \Sigma V^t$$

where U is an m*m orthogonal matrix, V is an n*n orthogonal matrix and Σ is an m*n matrix of the form,

$$\Sigma = \left[\begin{array}{c|c} D & 0 \\ \hline 0 & 0 \end{array}\right], \text{ where}$$

$$D = \begin{bmatrix} \sigma_1 & - & - & 0 \end{bmatrix}$$

where σ1 ≥σ2 ≥… σₙ ≥0

for r ≤m,n

Any such factorization is called the Singular Value Decomposition of matrix A. The matrices U and V are not unique. However, Σ is unique. The elements on the diagonal of D, namely σ1, σ2,… σₙ are the non-zero singular values of A. The columns of U are called left singular vectors of A and columns of V are called right singular vectors of A.

You can find examples for SVD here.

Apply SVD on co-occurrence matrix X

$$\begin{array}{cc} & |V| \\ |V| & \begin{bmatrix} & & \\ & X & \\ & & \end{bmatrix} \end{array} = \begin{array}{cc} & |V| \\ |V| & \begin{bmatrix} | & | & \\ u_1 & u_2 & \cdots \\ | & | & \end{bmatrix} \end{array} \begin{array}{cc} & |V| \\ |V| & \begin{bmatrix} \sigma_1 & 0 & \cdots \\ 0 & \sigma_2 & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} \end{array} \begin{array}{cc} & |V| \\ |V| & \begin{bmatrix} - & v_1 & - \\ - & v_2 & - \\ & \vdots & \end{bmatrix} \end{array}$$

Here, the dimension of X, U, Σ, and V^T is |V|*|V|.

Reducing dimensionality by selecting the first k singular features.

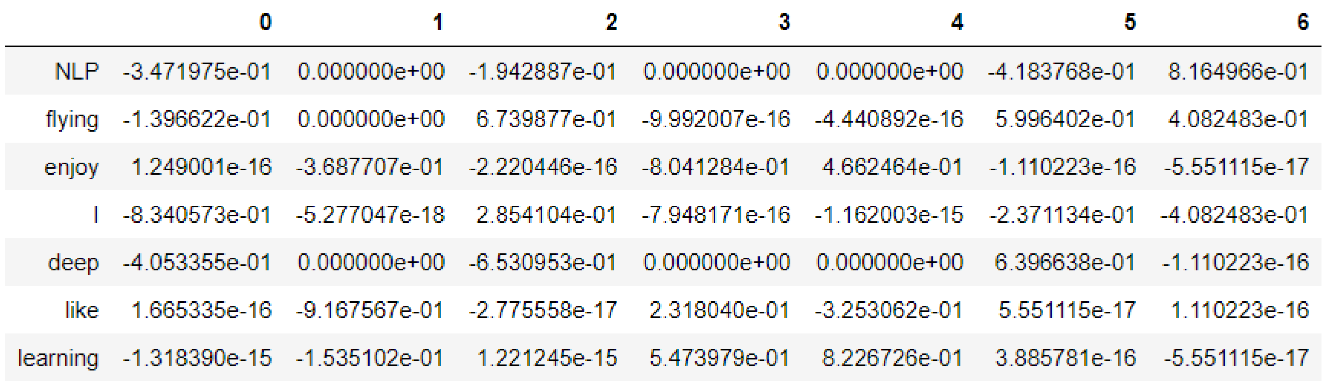
$$\begin{array}{cc} & |V| \\ |V| & \begin{bmatrix} & & \\ & \hat{X} & \\ & & \end{bmatrix} \end{array} = \begin{array}{cc} & k \\ |V| & \begin{bmatrix} | & | & \\ \underbrace{u_1 \quad u_2}_{} & & \cdots \\ | & | & \end{bmatrix} \end{array} \begin{array}{cc} & k \\ k & \begin{bmatrix} \sigma_1 & 0 & \cdots \\ 0 & \sigma_2 & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} \end{array} \begin{array}{cc} & |V| \\ k & \begin{bmatrix} - & v_1 & - \\ - & v_2 & - \\ & \vdots & \end{bmatrix} \end{array}$$

Word embedding matrix

Now,

- The dimension of X is |V|*|V|.
- The dimension of U is |V|*k.
- The dimension of Σ is k*k.
- The dimension of V^T is k*|V|.

Let's understand this by taking the previous example. We have the co-occurrence matrix A of dimension 7*7. Initially, just after the decomposition, the dimension of U, Σ, and V^T is also 7*7.

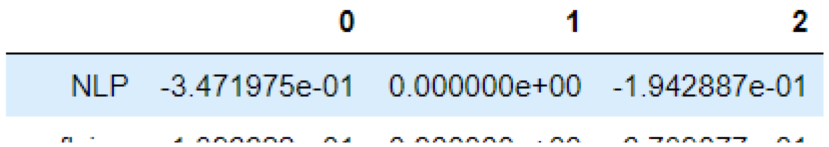


	0	1	2	3	4	5	6
NLP	-3.471975e-01	0.000000e+00	-1.942887e-01	0.000000e+00	0.000000e+00	-4.183768e-01	8.164966e-01
flying	-1.396622e-01	0.000000e+00	6.739877e-01	-9.992007e-16	-4.440892e-16	5.996402e-01	4.082483e-01
enjoy	1.249001e-16	-3.687707e-01	-2.220446e-16	-8.041284e-01	4.662464e-01	-1.110223e-16	-5.551115e-17
I	-8.340573e-01	-5.277047e-18	2.854104e-01	-7.948171e-16	-1.162003e-15	-2.371134e-01	-4.082483e-01
deep	-4.053355e-01	0.000000e+00	-6.530953e-01	0.000000e+00	0.000000e+00	6.396638e-01	-1.110223e-16
like	1.665335e-16	-9.167567e-01	-2.775558e-17	2.318040e-01	-3.253062e-01	5.551115e-17	1.110223e-16
learning	-1.318390e-15	-1.535102e-01	1.221245e-15	5.473979e-01	8.226726e-01	3.885781e-16	-5.551115e-17

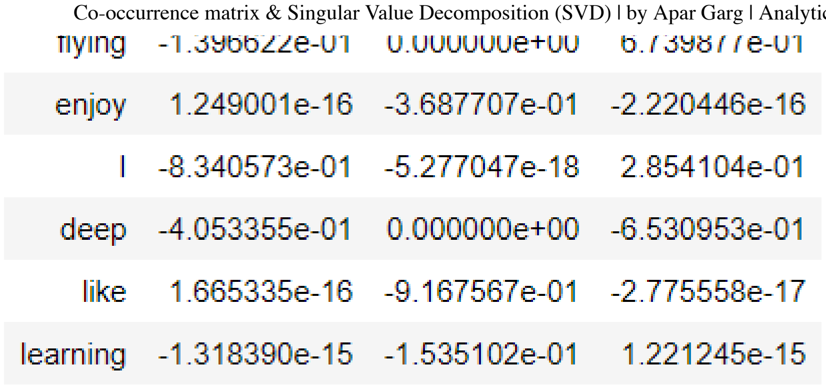
Matrix U (before selecting k singular features)

Now after reducing dimensionality by selecting the first k singular features,

- The dimension of A is 7*7.
- The dimension of U is 7*3.
- The dimension of Σ is 3*3.
- The dimension of V^T is 3*7.



	0	1	2
NLP	-3.471975e-01	0.000000e+00	-1.942887e-01



flying	-1.396622e-01	0.000000e+00	6.739877e-01
enjoy	1.249001e-16	-3.687707e-01	-2.220446e-16
I	-8.340573e-01	-5.277047e-18	2.854104e-01
deep	-4.053355e-01	0.000000e+00	-6.530953e-01
like	1.665335e-16	-9.167567e-01	-2.775558e-17
learning	-1.318390e-15	-1.535102e-01	1.221245e-15

Matrix U (after selecting k singular features)

Advantages of Co-occurrence Matrix

1. It preserves the semantic relationship between words. i.e man and woman tend to be closer than man and apple.

2. It uses SVD at its core, which produces more accurate word vector representations than existing methods.

3. It uses factorization which is a well-defined problem and can be efficiently solved.

4. It has to be computed once and can be used anytime once computed. In this sense, it is faster in comparison to others.

Disadvantages of Co-Occurrence Matrix

1. It requires huge memory to store the co-occurrence matrix.
   But, this problem can be circumvented by factorizing the matrix out of the system for example in Hadoop clusters etc. and can be saved.

Problems with SVD method

1. The dimensions of the matrix change very often (new words are added very frequently and corpus changes in size).

2. The matrix is extremely sparse since most words do not cooccur.

3. The matrix is very high dimensional in general (106 *106)

4. Quadratic cost to train (i.e. to perform SVD)

5. Requires the incorporation of some hacks on X to account for the drastic imbalance in word frequency

You can find the implementation of the Co-occurrence matrix and SVD here.

Sign up for Analytics Vidhya News Bytes

By Analytics Vidhya

Latest news from Analytics Vidhya on our Hackathons and some of our best articles! Take a look.

Your email

Get this newsletter

By signing up, you will create a Medium account if you don't already have one. Review our Privacy Policy for more information about our privacy practices.

Natural Language Svd Co Occurrence Matrix Word Embeddings Pca

About Write Help Legal

Get the Medium app

Download on the App Store GET IT ON Google Play