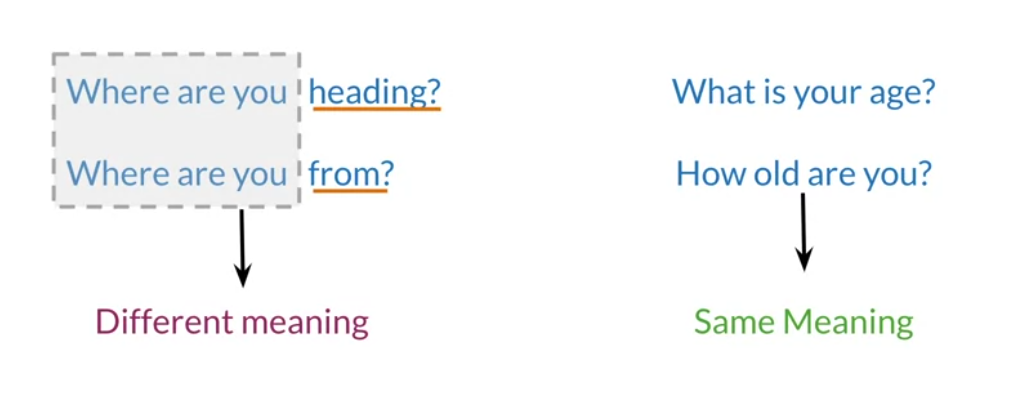
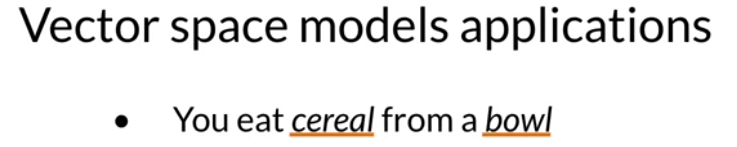
Vector space models capture semantic meaning and relationships between words. You'll learn how to create word vectors that capture dependencies between words, then visualize their relationships in two dimensions using PCA.

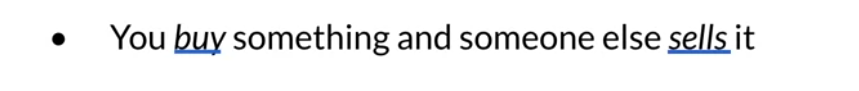


* They can be used to identify similarity for a question answering, paraphrasing and summarization.
* Vector space models will also allow you to capture dependencies between words.

تأكل الحبوب من وعاء.



You can see that the word cereal and the word bowl are related.



The second half of the sentence is dependent on the first half. With vector space models, you will able to capture this and many other types of relationship among different sets of words.

Vector space model using in information extraction to answer in the style of questions of who, what , where and

In machine translation and in chatbots programming.

You will make co-occurrence matrix and extract vector representation for words in your corpus.

* You will be able to get a vector space model using a word by document design using a similar approach.

Relationship between words /documents

You can find relationship between words and vectors,

“ to co-occurrence of two different word”

Number of time they occur together within a certain distance k.

Suppose that your corpus has the following two sentences.

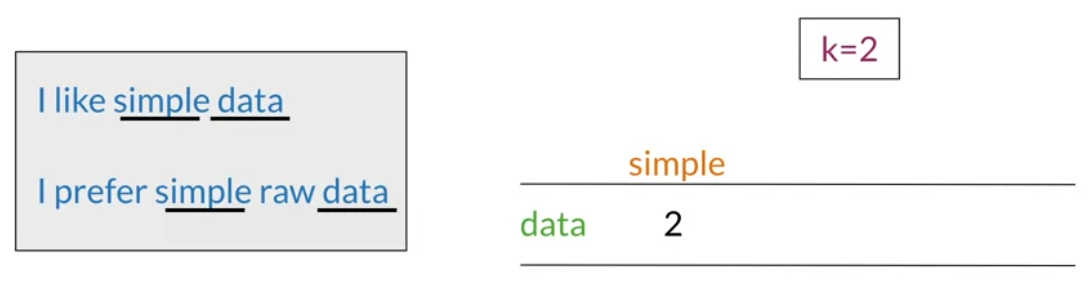


The row of co-occurrence matrix corresponding to the word **data** with a k value equal to 2 would be populated with the following values.

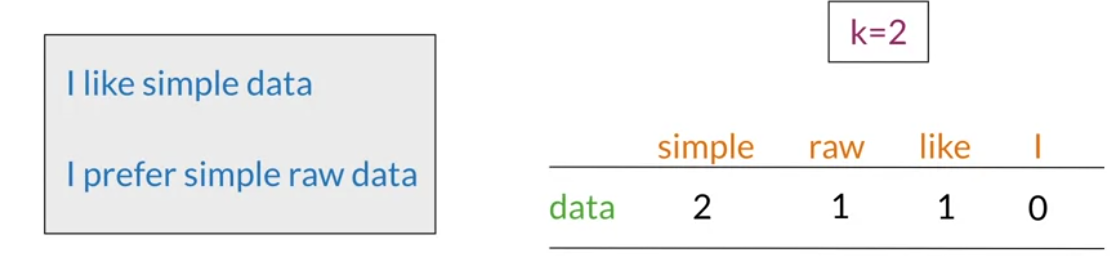
For the column corresponding to the word simple, you would get a value = 2

“ because data and simple co-occur in the first sentence

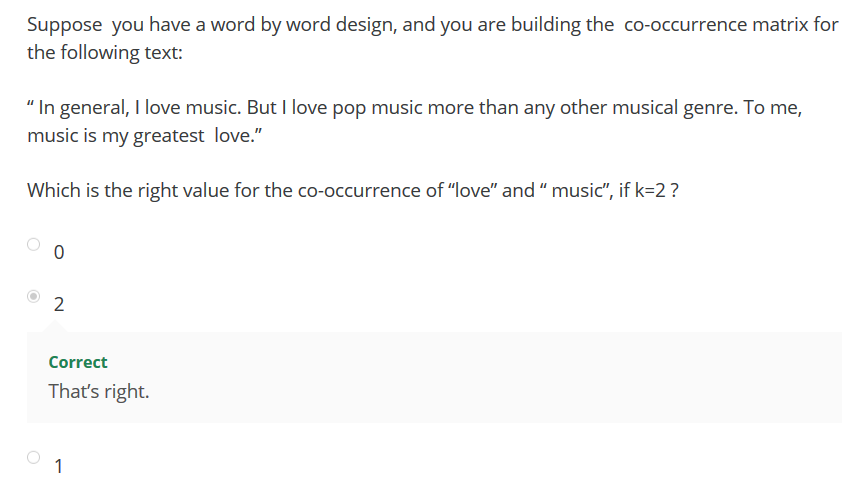
Within a distance of one word, and the second sentence of two words.



The word data world be equal

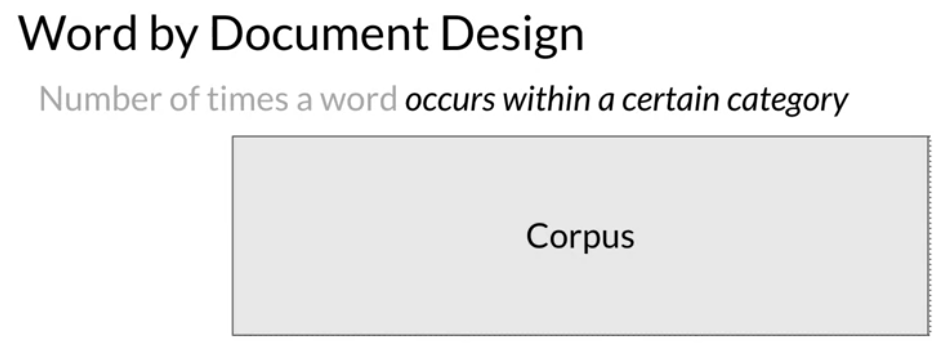


**With word by word design, you can get a representation with n entities with n between one and size of your entire vocabulary.**



For a word by document design, the process is quite similar.

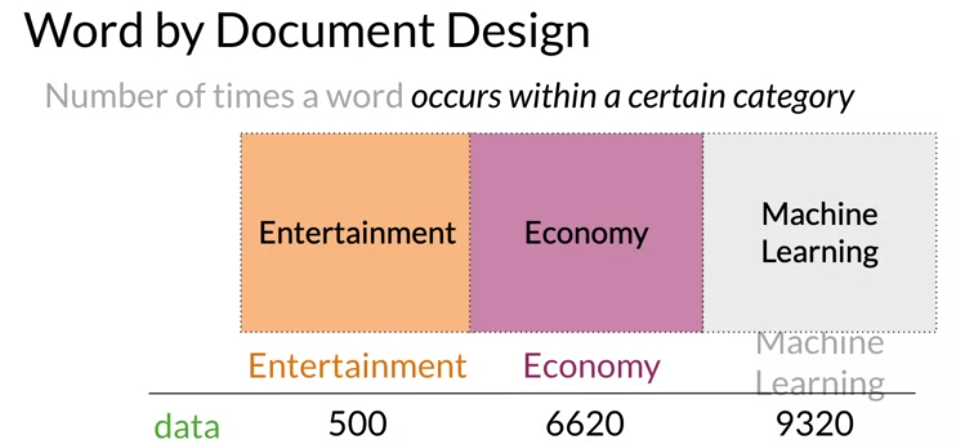
In this case, you will count the times the words from your vocabulary appear in documents that belong to specific categories.



For instance, you could have a corpus consisting of documents between different topics like entertainment, economy ,machine learning.

Example

Suppose that the word data appears



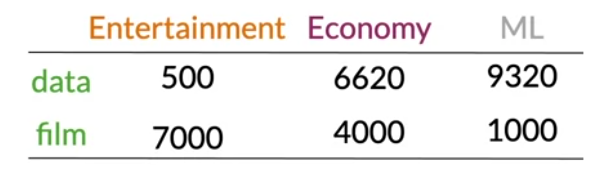
The word film appears in each document’s category 7,000. …………..



Can you get a sense of where this is going already?

Construct the representation: for multiple sets of documents or words,

You will get your vector space.

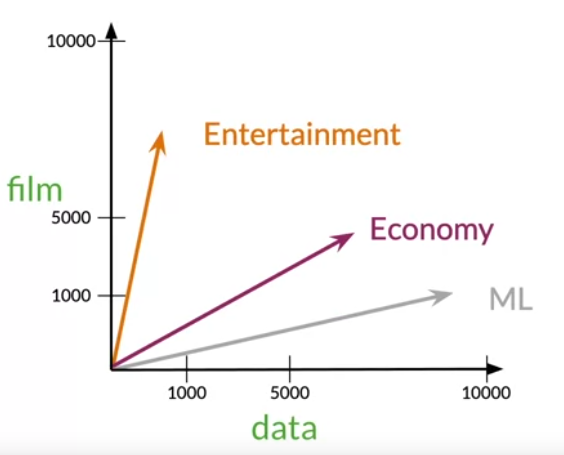


The vector space will have two dimensions.

1. The number of times that the words data and film appear on the type of document.

For the entertainment corpus, you would have the following vector representation.

This one for the economy category,



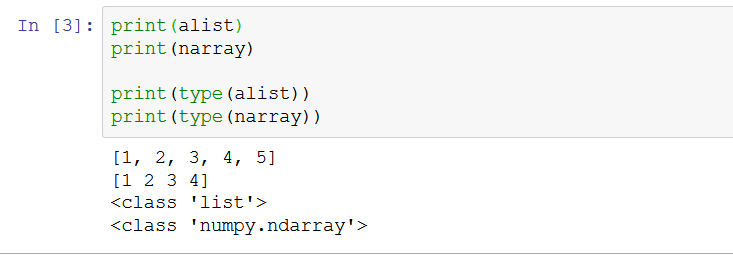
Note that in this space, it easy to see that the economy and machine learning documents are much more similar than they are to the entertainment.

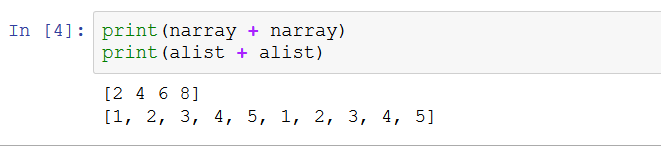
Measures of similarity Angle and Distance.

You will make comparisons between vector representation using the cosine similarity and the Euclidean distance.

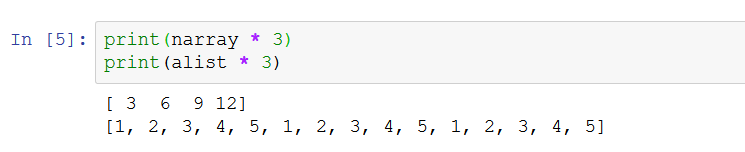
In order to get the angle and distance between them.

The similarity metric is known at the Euclidean distance.

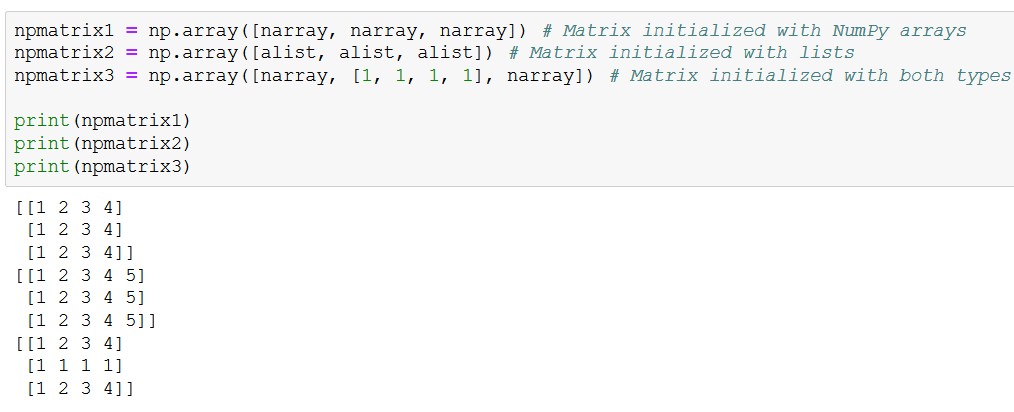




List vs array

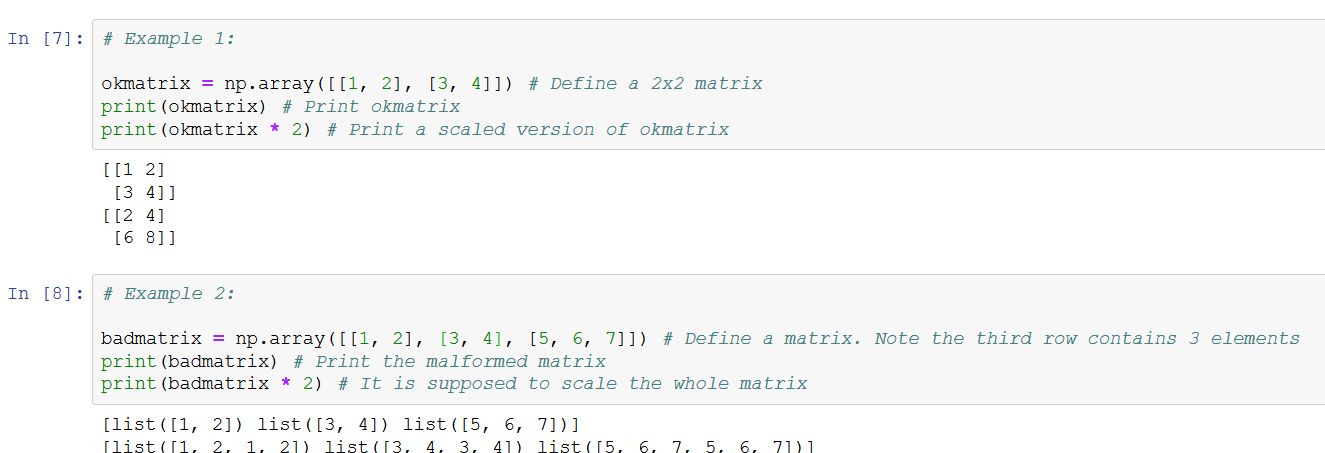


Create matrix

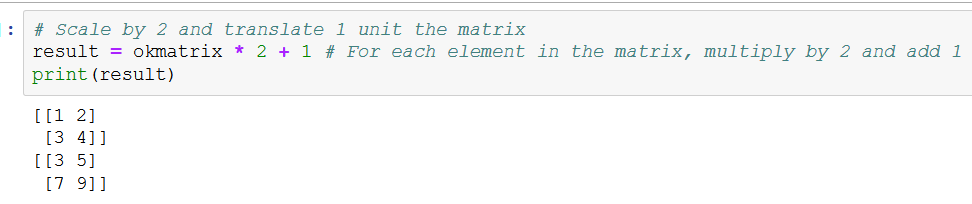


However, when defining a matrix, be sure that all the rows contain the same number of elements. Otherwise, the linear algebra operations could lead to unexpected results.

Take care for number of elements

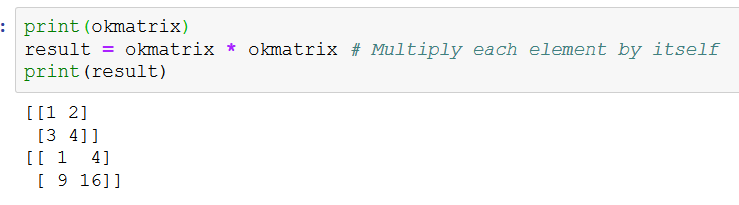


Scaling

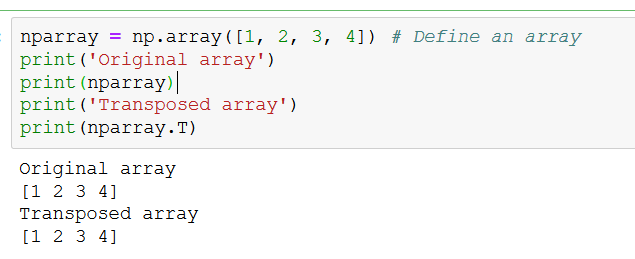


The product operator `\*` when used on arrays or matrices indicates element-wise multiplications.

Do not confuse it with the dot product.

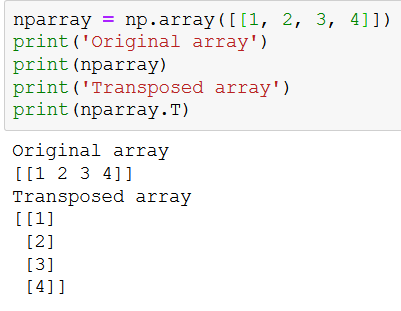


Transpose

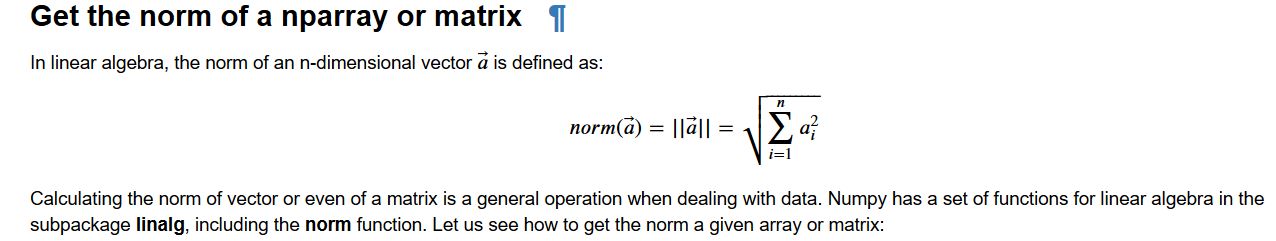


Note

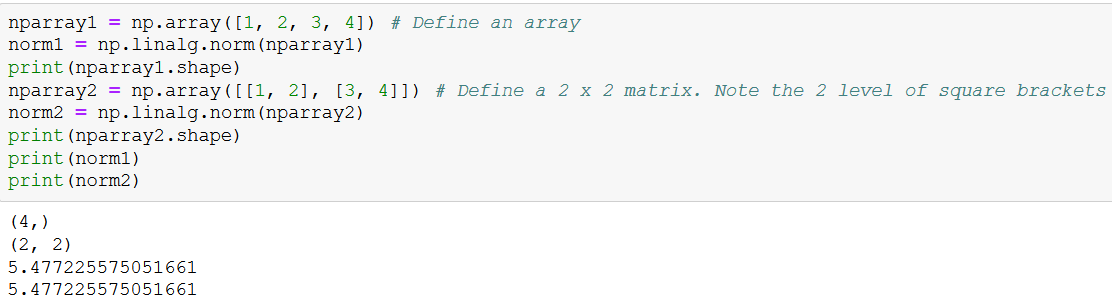
Two [[]]



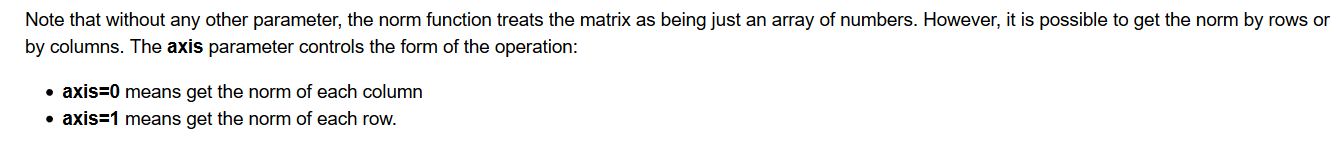
Calculating the norm of vector or even of a matrix is a general operation when dealing with data. Numpy has a set of functions for linear algebra in the subpackage \*\*linalg\*\*, including the \*\*norm\*\* function. Let us see how to get the norm a given array or matrix:



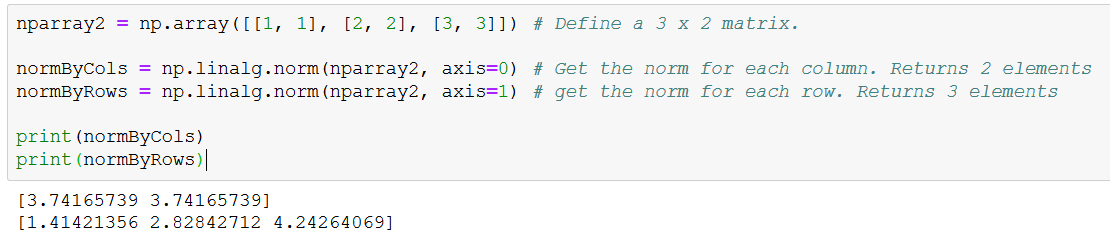
Normalize



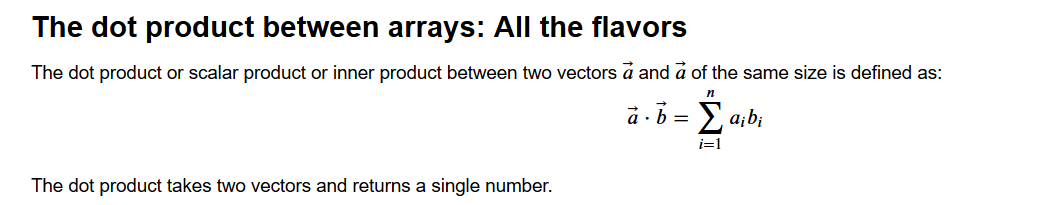
Control norm



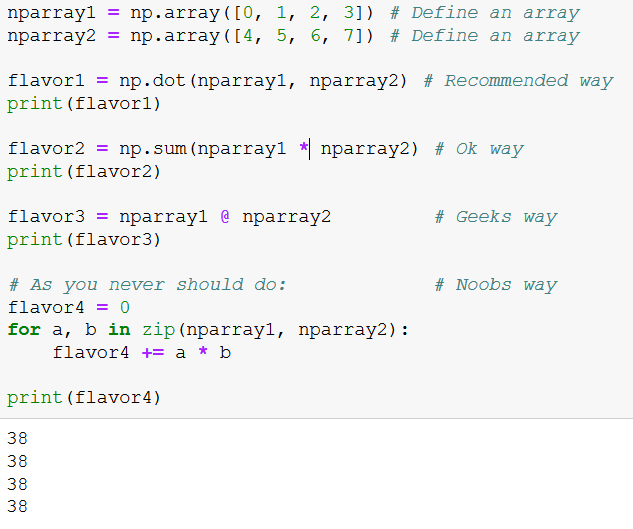
Example



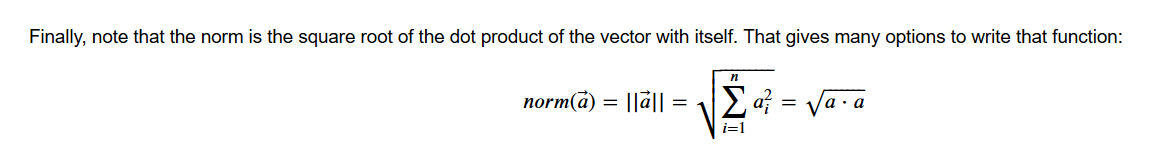
Dot. Product

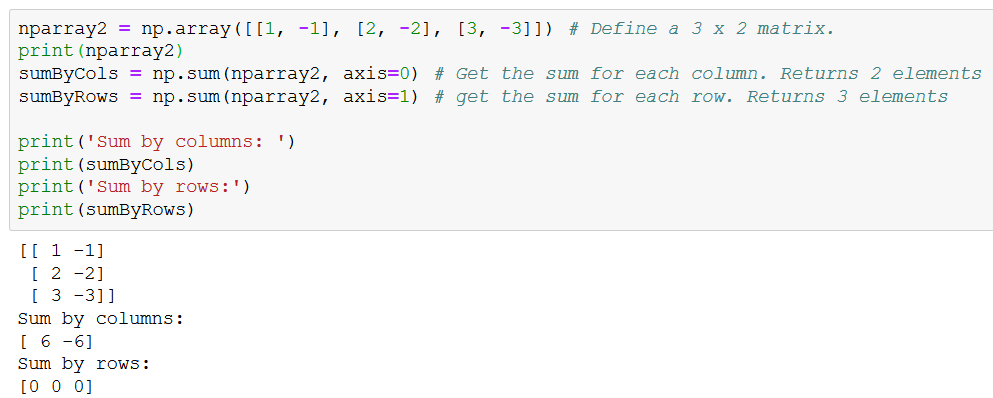


Example

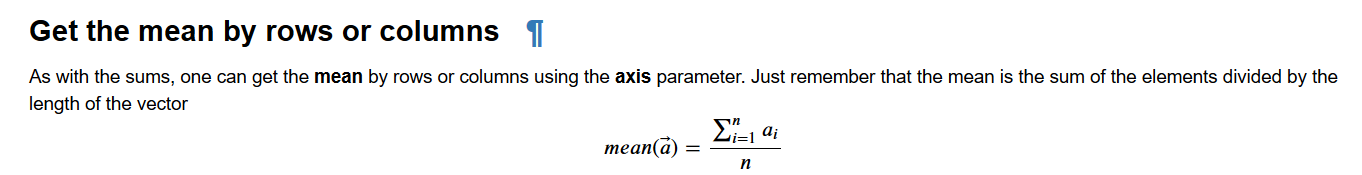


Finally, note that the norm is the square root of the dot product of the vector with itself. That gives many options to write that function:

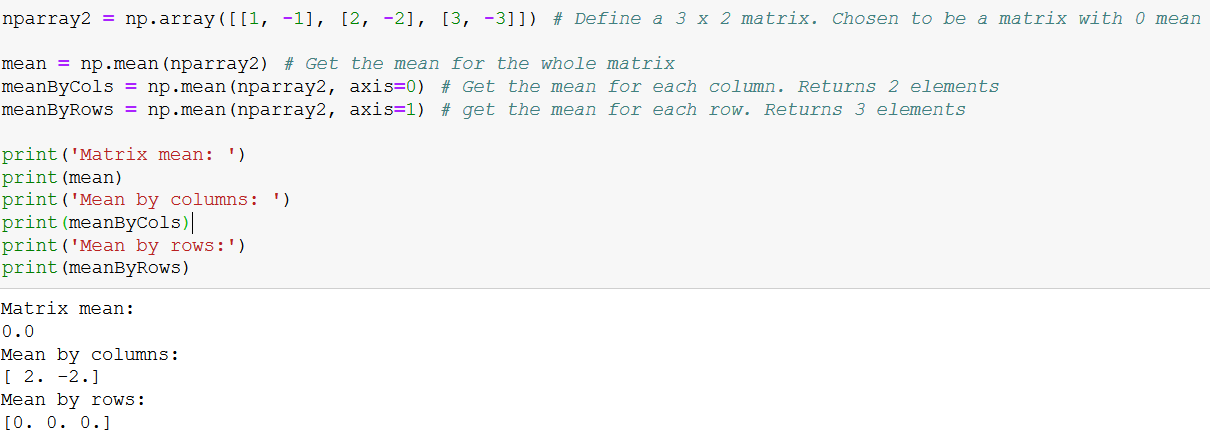




**Mean of row and column**



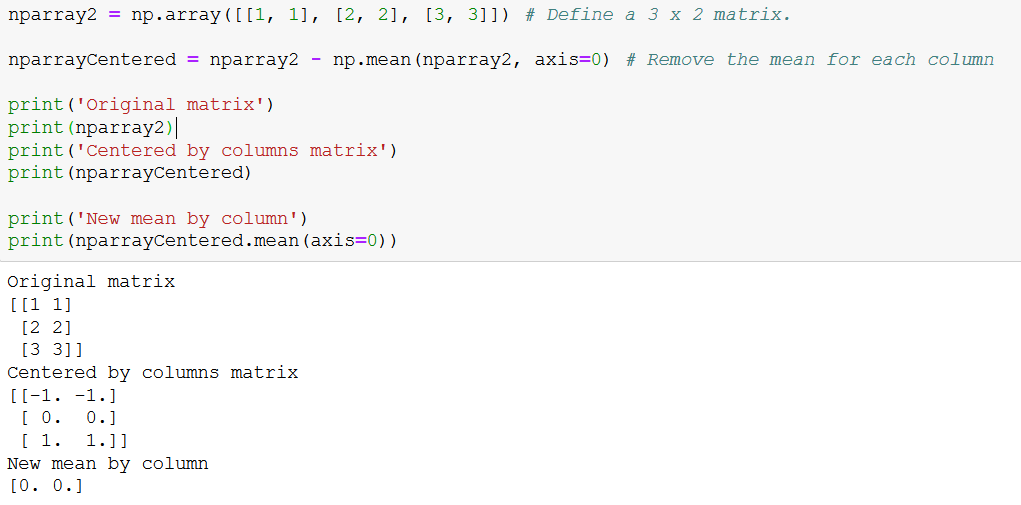
Example



Center

Centering the attributes of a data matrix is **another essential preprocessing** step. Centering a matrix means to remove the column mean to each element inside the column. The sum by columns of a centered matrix is always 0.

Example



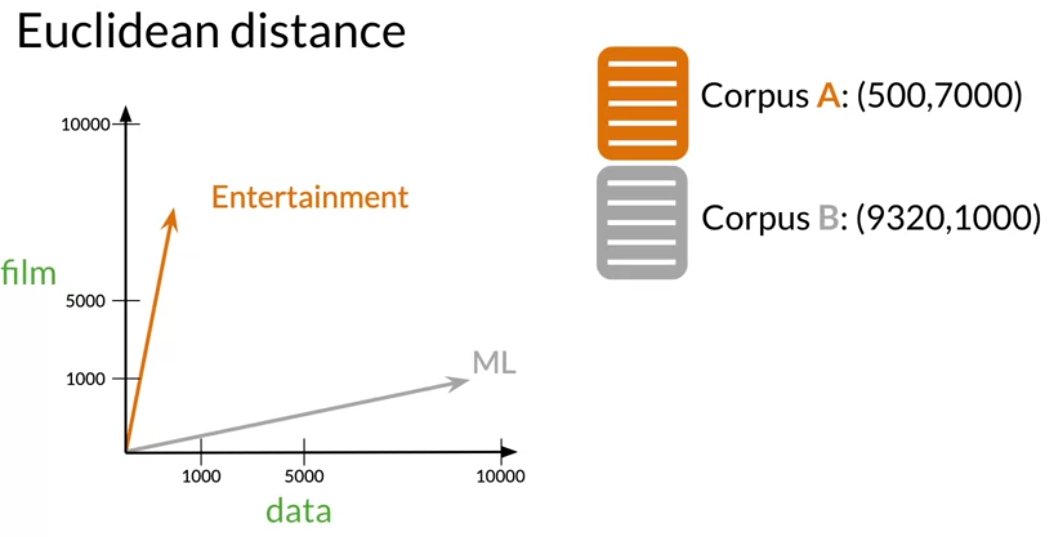
Euclidian distance which is a similarity metric.

“ this metric allows you to identify how far two points or two vector are apart from each other.”

1. You will get the Euclidean distance between documents vector and then generalize that to notion to vector space in height dimensions.

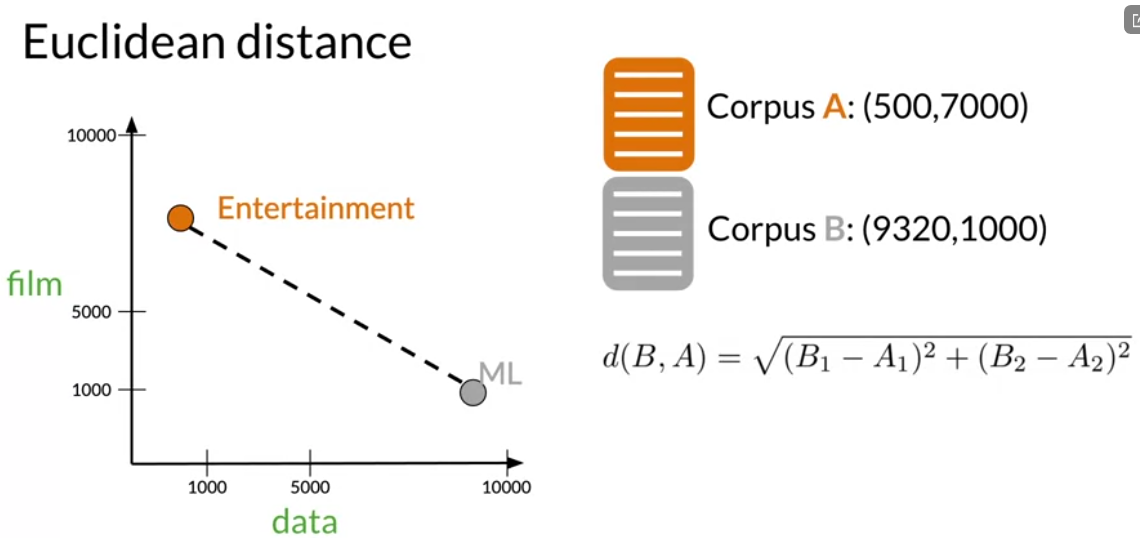
Example

1-the number of times the word data and the word film appearance in the corpus.

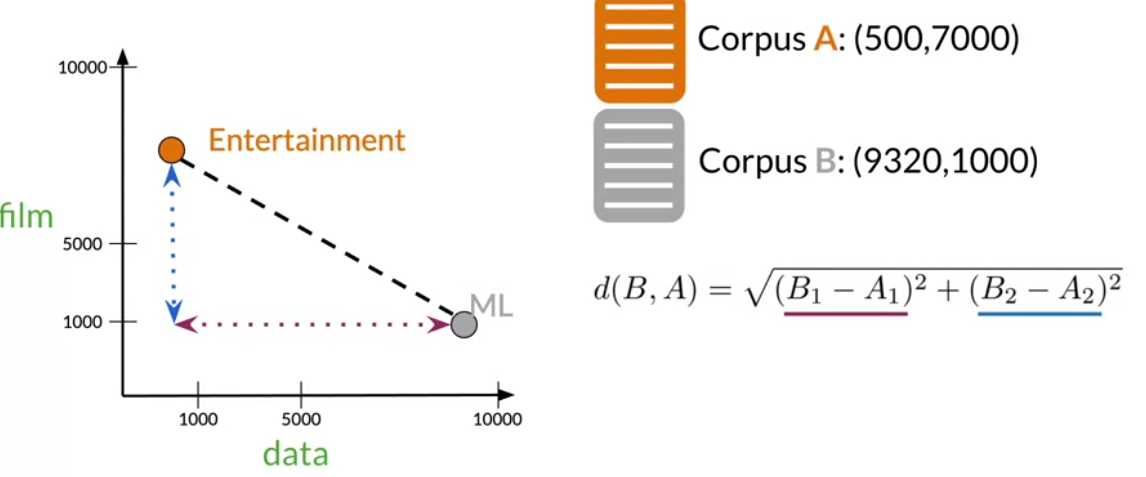


Now let`s represent those vectors as points in the vector space.

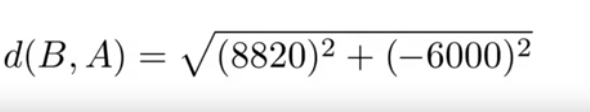
The Euclidian distance is the length of the straight line segment connecting them.



1. The first term is their horizontal distance squared
2. The second term is their vertical distance squared

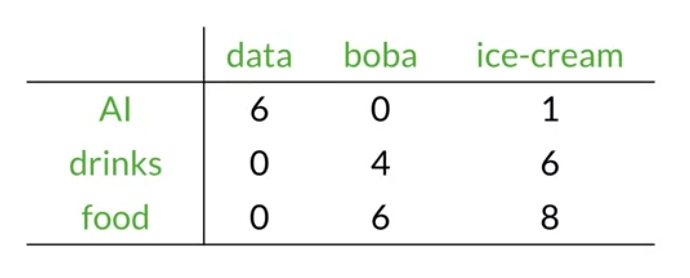


As you see, this formula is an example of the Pythagorean theorem.

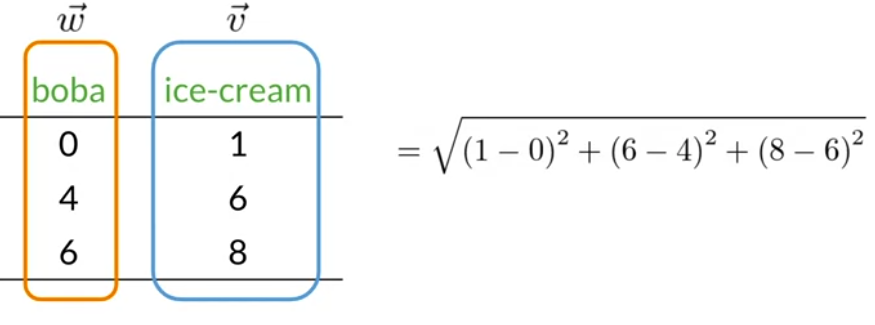


When you have higher dimensions, the Euclidean distance is not much more difficult.

Example

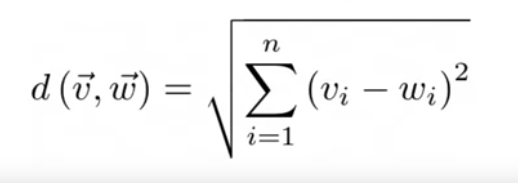


To get ice-cream and boba, start to get difference between them



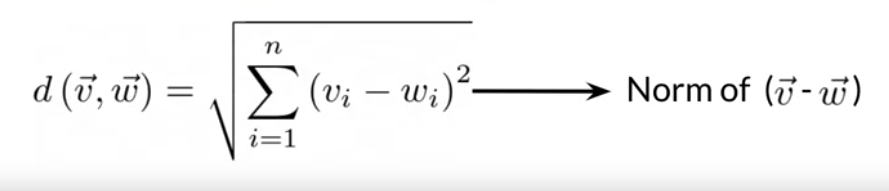
Each of their dimensions,

This is generalize of formula

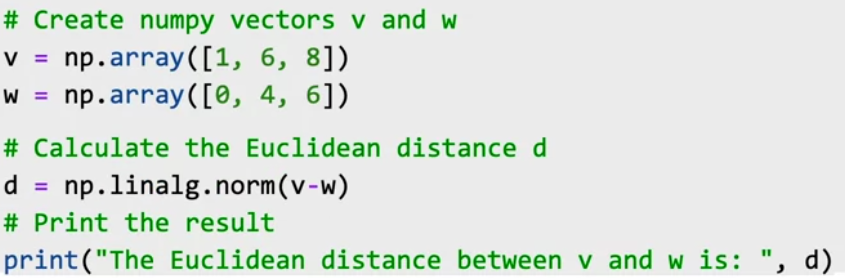


This is the formula that you would use to get the Euclidean distance between vector representation on at n-dimensional vector space.

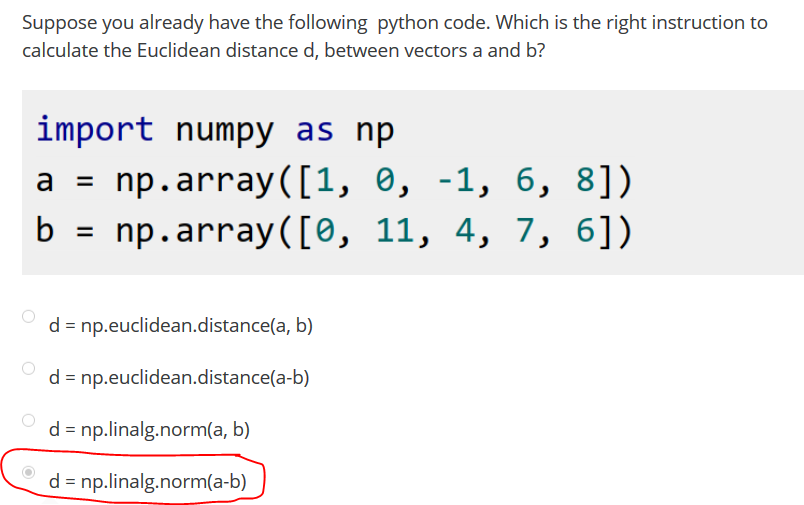
1. The you remember from algebra, this formula is known as the norm of the difference between the vectors that you are comparing.



Euclidean distance in python.



Quiz



Cosine similarity: it basically makes use of the cosine of the angle between two vectors.

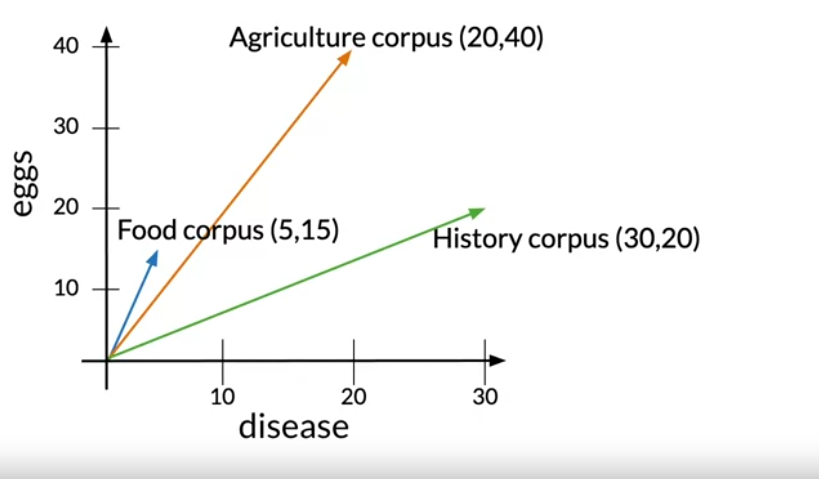
It tell you whether two vectors are close or not.

Problem with Euclidean distance.

Overcome that problem.

Example

this is the representation of three

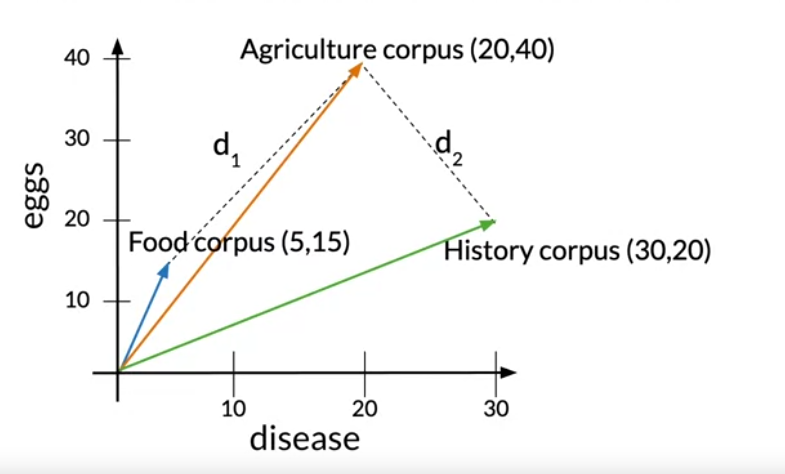


Each of these corpora have text related to that subject. But you know that the word totals in the corpora differ from one another.

* In the fact, the agriculture and the history corpus have a similar number words, while the food corpus has a relatively small number.

Let`s define the Euclidean distance between the food and the agriculture corpus as d1.

And let the Euclidean distance between the history and the agriculture corpus as d2.

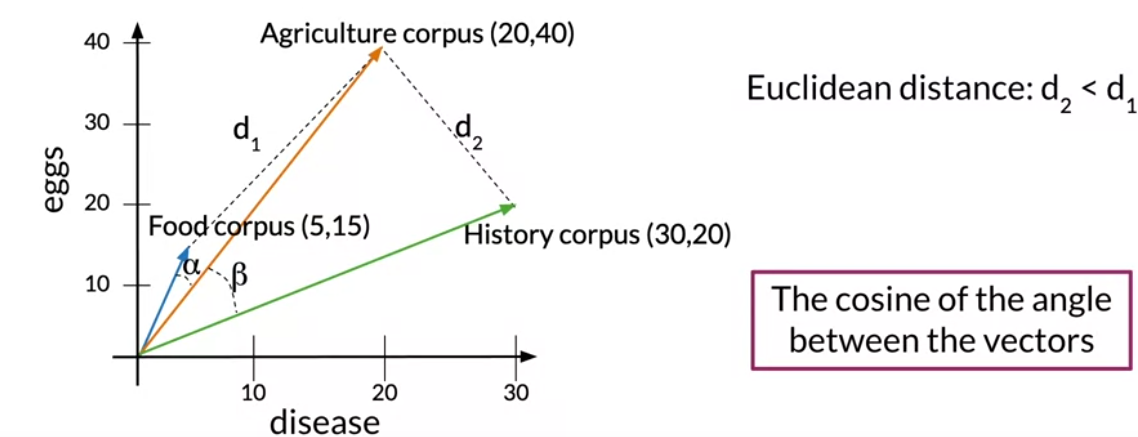


Euclidean distance d2 < d1, which would suggest that the agriculture and history corpora are more similar than the agriculture and food corpora.

Another common method for determining the similar between vectors

* If the angle is small, the cosine would be close to one,
* And as the angle approaches 90 degrees, the cosine approaches zero.

As you can see here, the angle alpha between food and agriculture is smaller than the angle beta between agriculture and history.



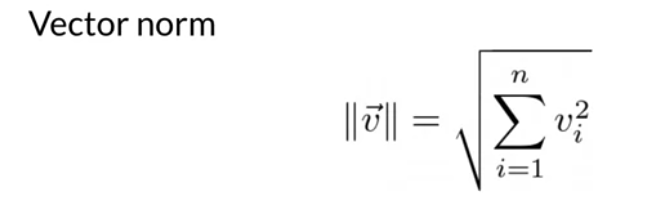
The cosine of those angles is a better proxy of similarly between these vector representations than their Euclidean distance.



Remember, that the main advantage of this metric over the Euclidean distance is that it isn`t instead by size difference between the representations.

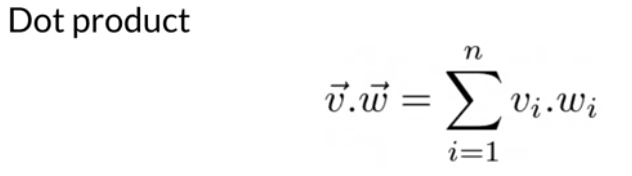
Outline

Relation of this metric to similarity: is related to the similarity of the directions of two vectors.



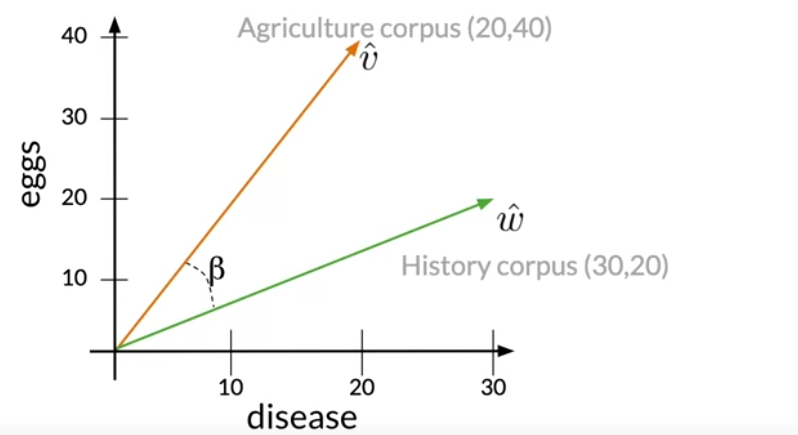
The norm of a vector or its magnitude is written like this.

It`s defined to be the square root of the sum of its elements squared.



The dot product between two vector is the sum of the products between their elements in each dimension of the vector space.

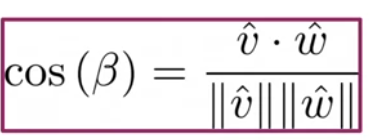
The angle between those vector representation is denoted by beta.



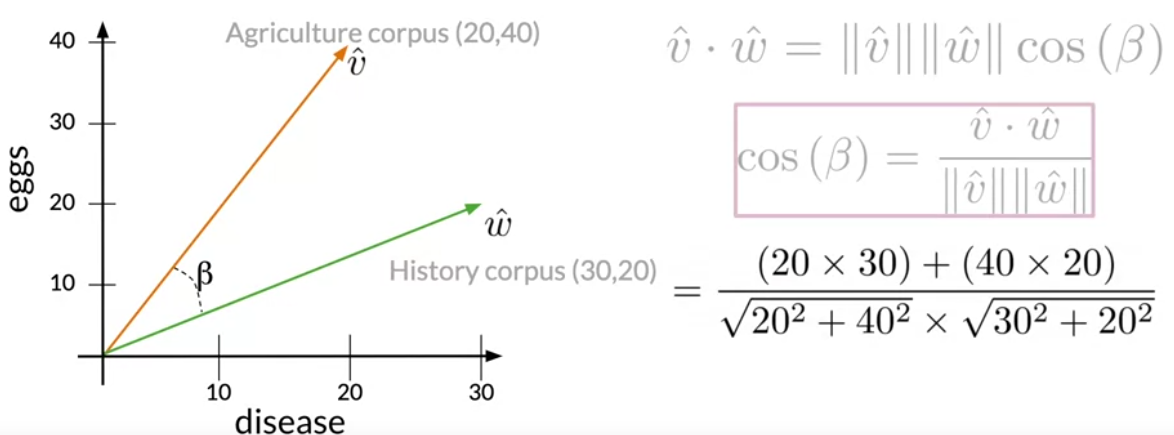
The dot products between two vectors.



The cosine of te angle beta is equal.



Example



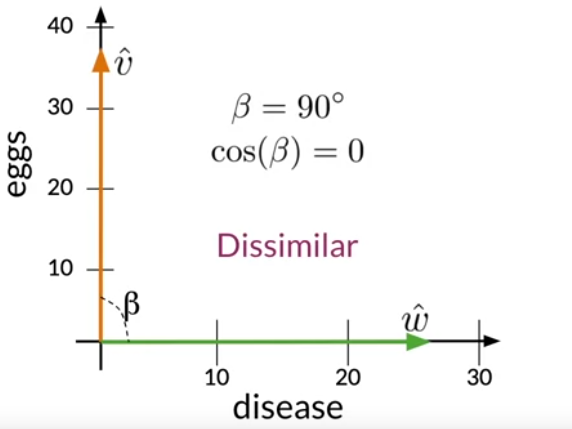
= 0.87

But what the metric tell you about the similarity of two different vectors?

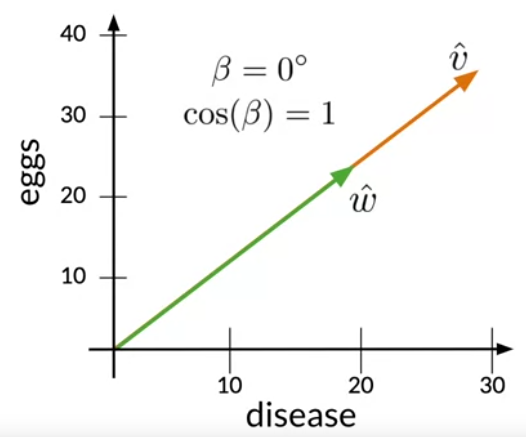
Consider when two vectors are orthogonal in the vector spaces that you know so far.

* It is only possible to have positive values for any dimension, so maximum angle between pair of vectors is 90 digress.

In that case, the cosine would be equal to zero, and it would mean that the two have orthogonal directions or that they are maximally the similar.

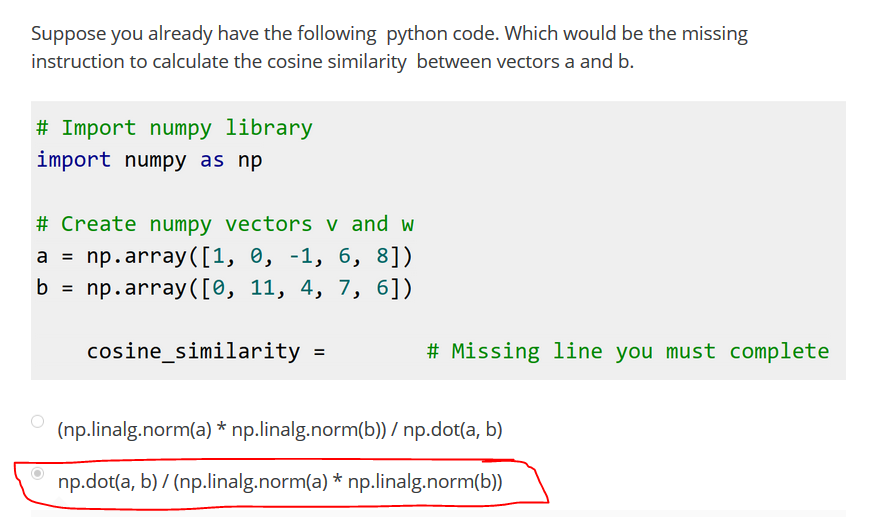


Now, let`s look at the case where the vectors have the same directions.



Are you cane see, as the cosine of the angle between two vectors approaches one, their closer

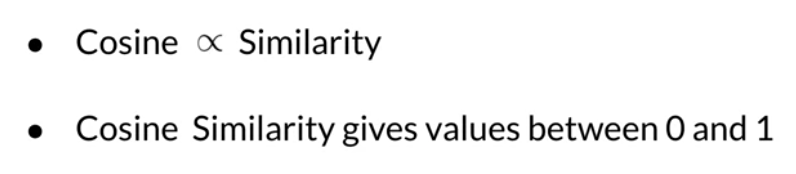
QUIZ



Note

An important takeaway is that, this metric is proportional to the similarity between the directions of the vectors that you are comparing.

And that for the vector spaces you have seen so

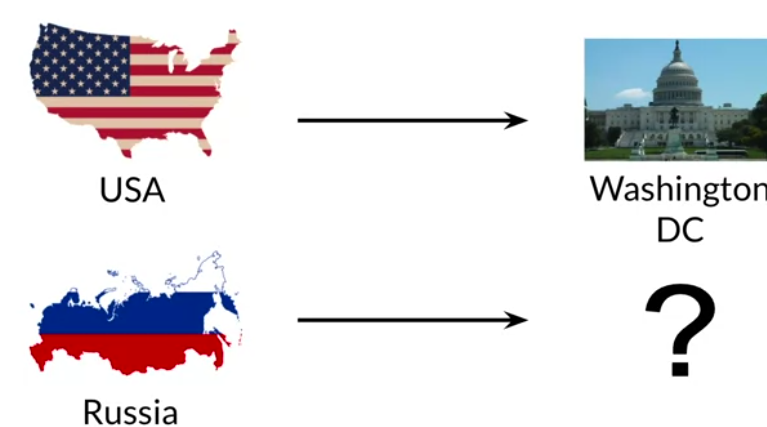


Manipulating Words in Vector Spaces

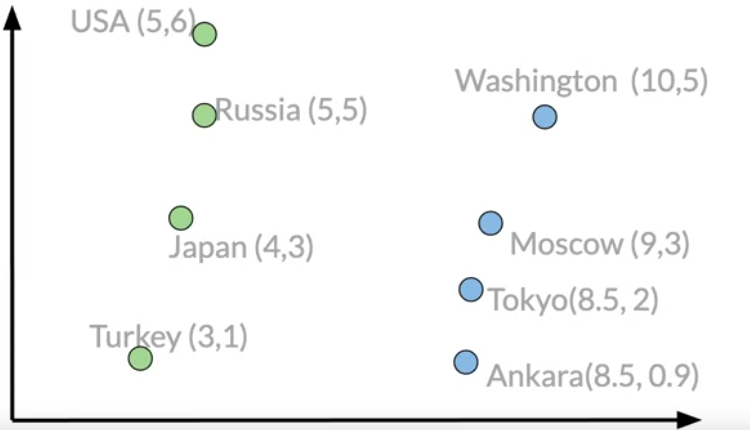
How to use vector representations in order to infer unknow relations among words.

Suppose you have a vector space with countries and their capital cities.

* You know that the capital of the united state is Washington Dc and don`t know the capital of Russia, but you would like to use the relationship known between Washington dc and the USA to figure out.

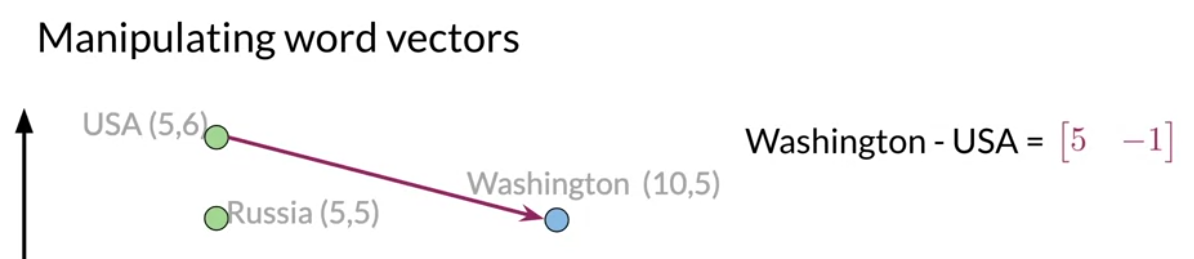


A hypothetical two-dimensional vector space that has different representing for different countries and capital cities.



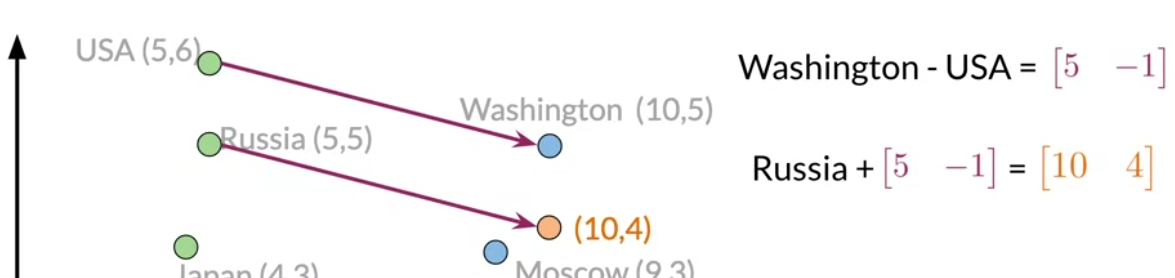
First, you will have to find the relationship between wash – dc and USa vector representation.

1. In other words, which vector connects them? To do that get the difference between the two vectors.



How many units on each dimension you should in order to find a country`s capital in that vector space.

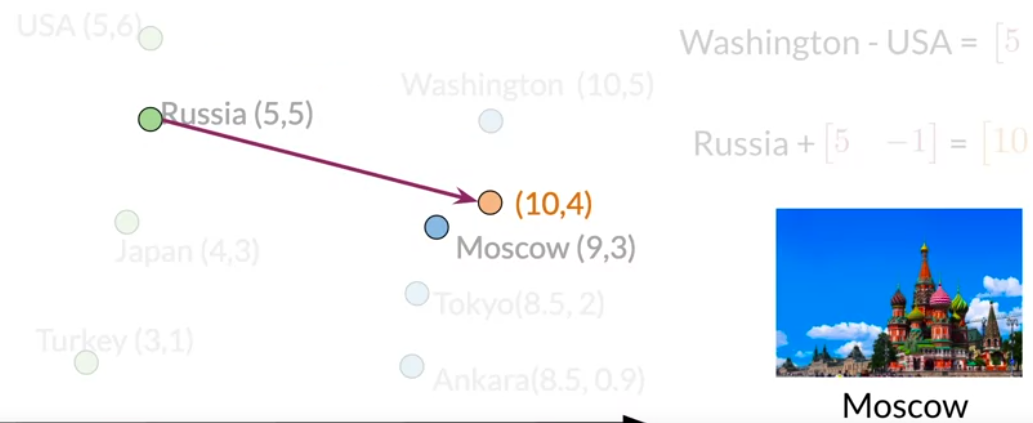
So to find the capital city of Russia, you will have to sum it`s vector presentation with the vector that you also got in the last step.



You should deduce that the capital of Russia has a vector representation of 10, 4

However, there is are no cities with that representation, so you will have to take the one that is the most similar to its by comparing each vector with the Euclidean distances or cosine similarities.

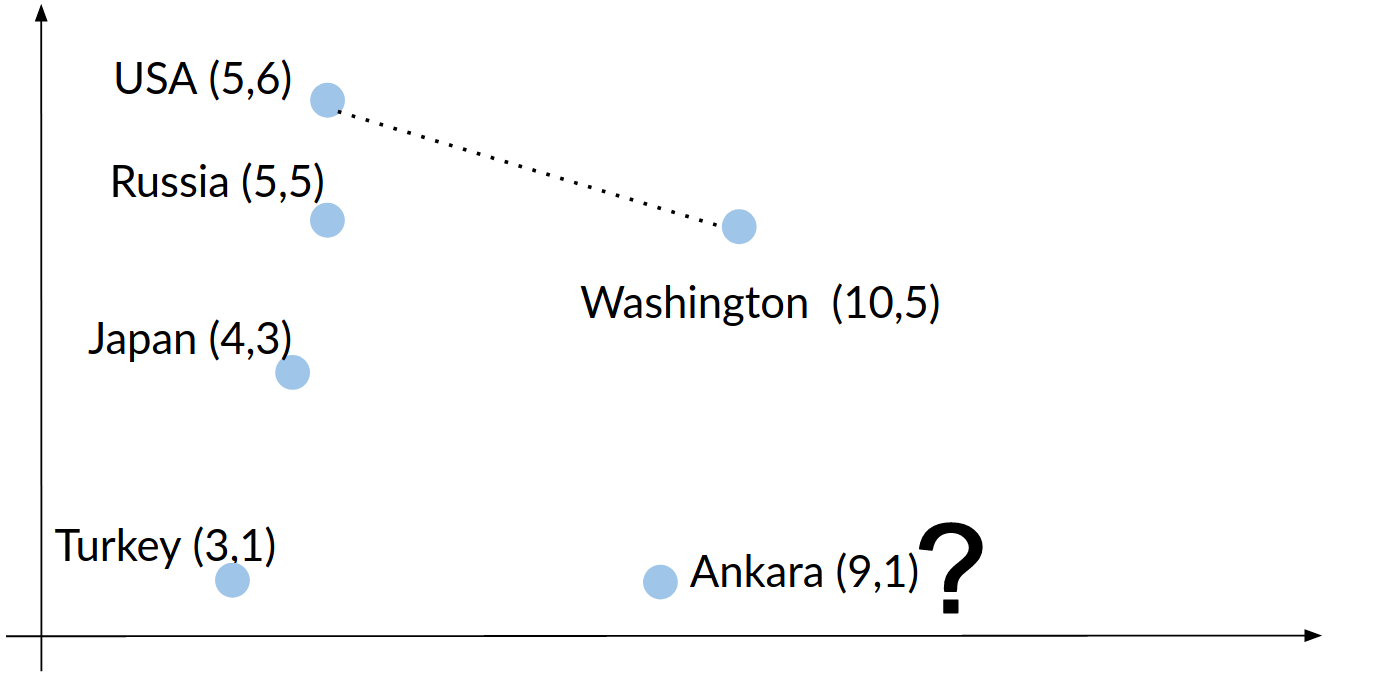
In this case, the vector representation that is closest to the 10,



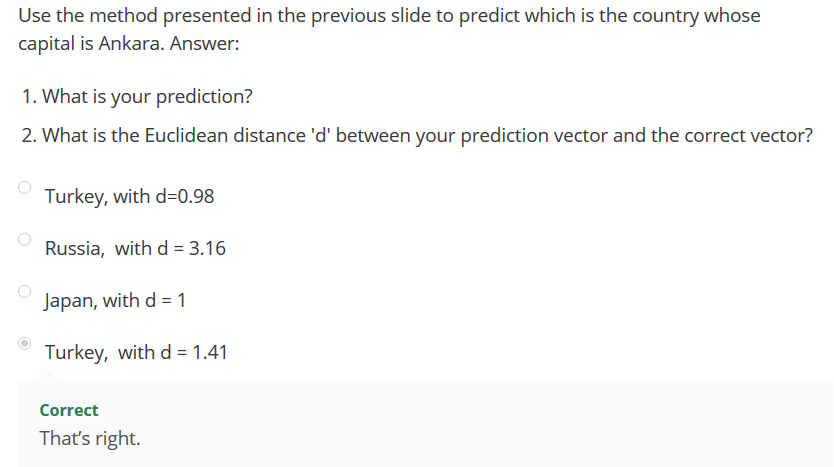
Using the simple process, you could have predict the capital of Russia if you knew the capital of the USA.

You only need here, a vector space where the representations capture the relative meaning of words.

Example



Answer:

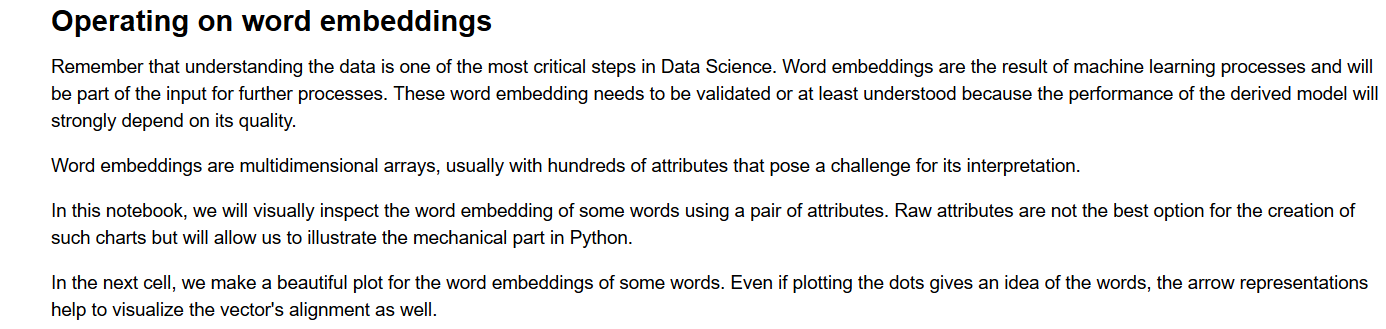


To get unknown relationships between words by use of known relationships between others.

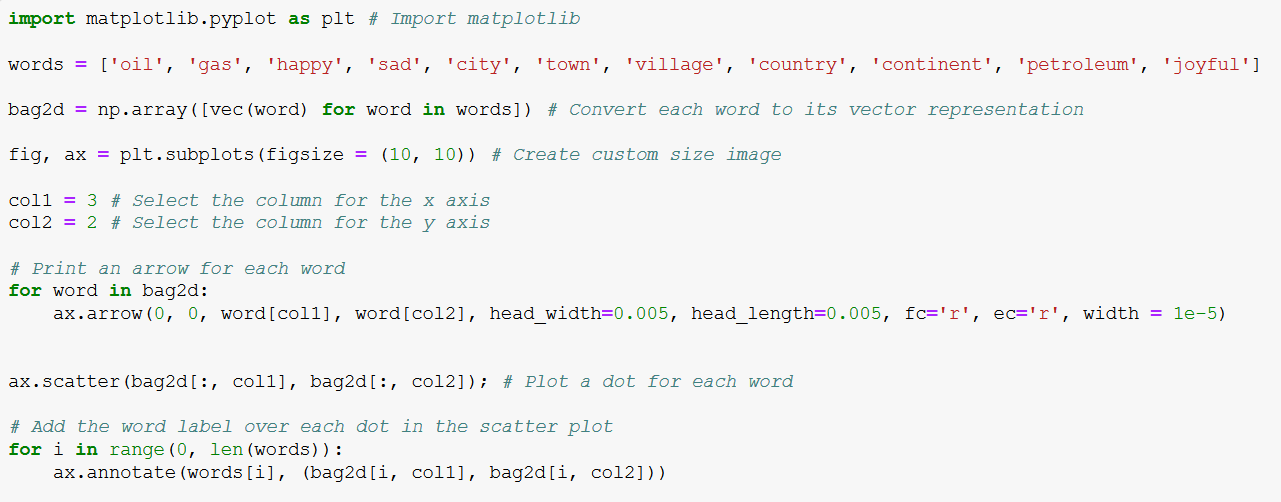
You now know the importance of having vectors spaces where the representation of words captures the relative meaning in natural

Example: to represent vector

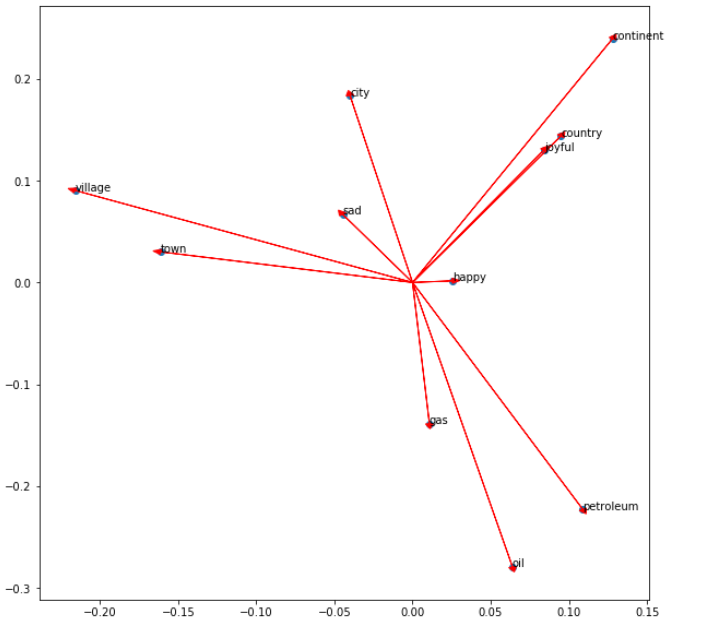
If the word doctor and were the closest words that are closest to it by computing cosine similarity, nurse , ………………….



Example



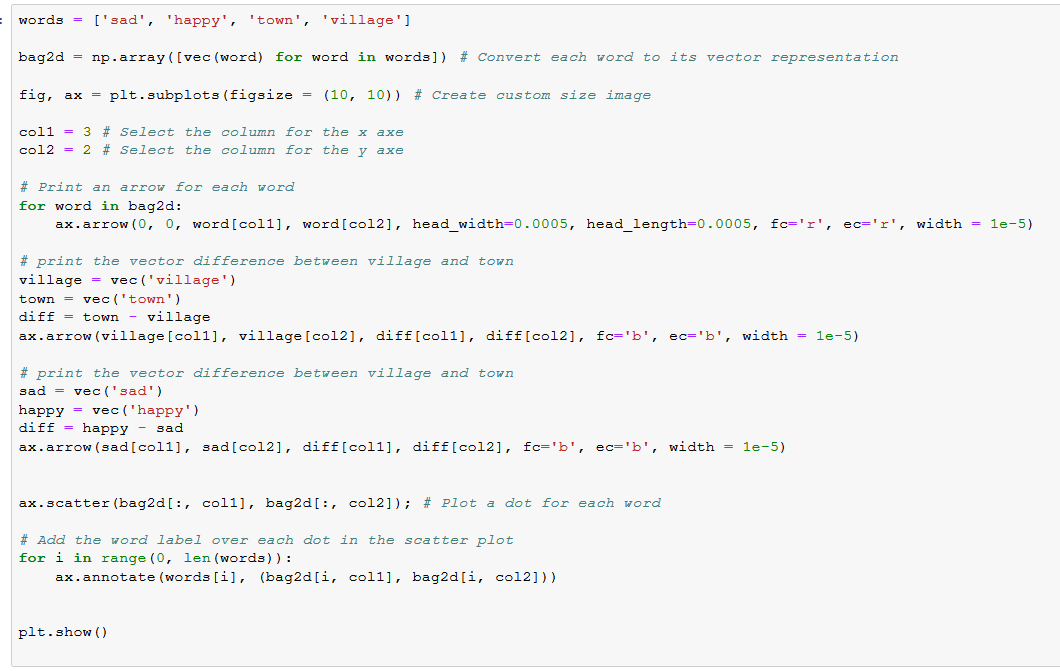
Print



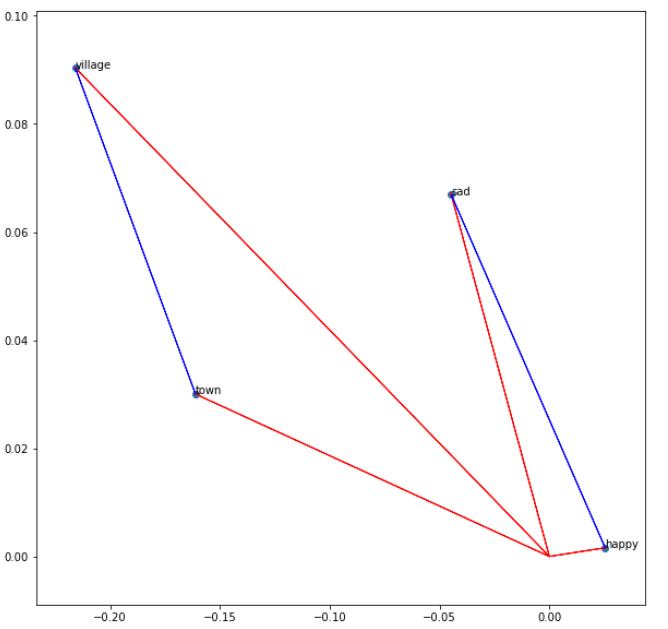
Note that similar words like 'village' and 'town' or 'petroleum', 'oil', and 'gas' tend to point in the same direction. Also, note that 'sad' and 'happy' looks close to each other; however, the vectors point in opposite directions.

In this chart, one can figure out the angles and **distances between the words. Some words are close in both kinds of distance metrics**.

Now plot the words 'sad', 'happy', 'town', and 'village'. In this same chart, display the vector from 'village' to 'town' and the vector from 'sad' to 'happy'. Let us use NumPy for these linear algebra operations.



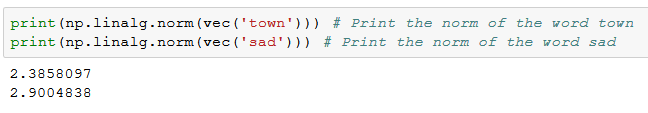
Graph



## Linear algebra on word embeddings

In the lectures, we saw the analogies between words using algebra on word embeddings. Let us see how to do it in Python with Numpy.

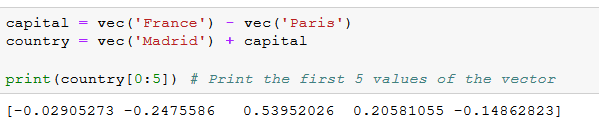
To start, get the \*\*norm\*\* of a word in the word embedding.



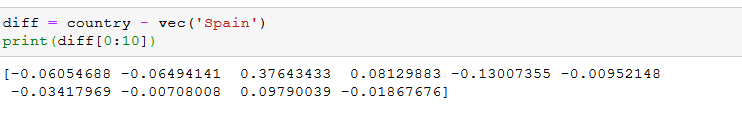
Predict capital

Now, applying vector difference and addition, one can create a vector representation for a new word. For example, we can say that the vector difference between 'France' and 'Paris' represents the concept of Capital.

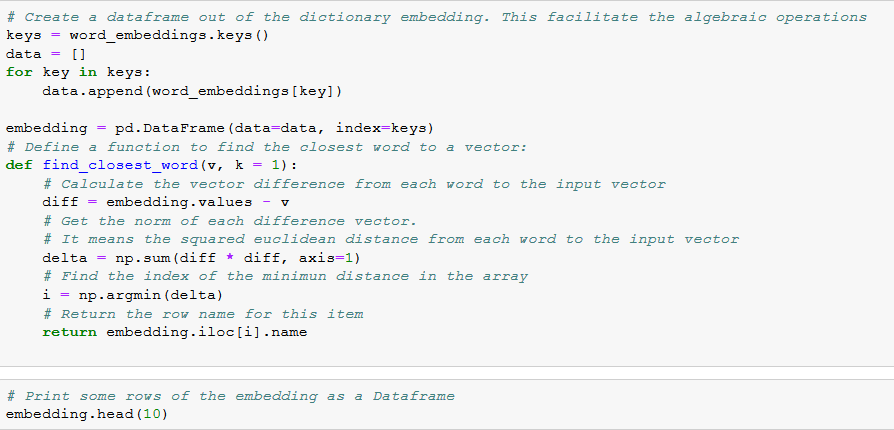
One can move from the city of Madrid in the direction of the concept of Capital, and obtain something close to the corresponding country to which Madrid is the Capital.



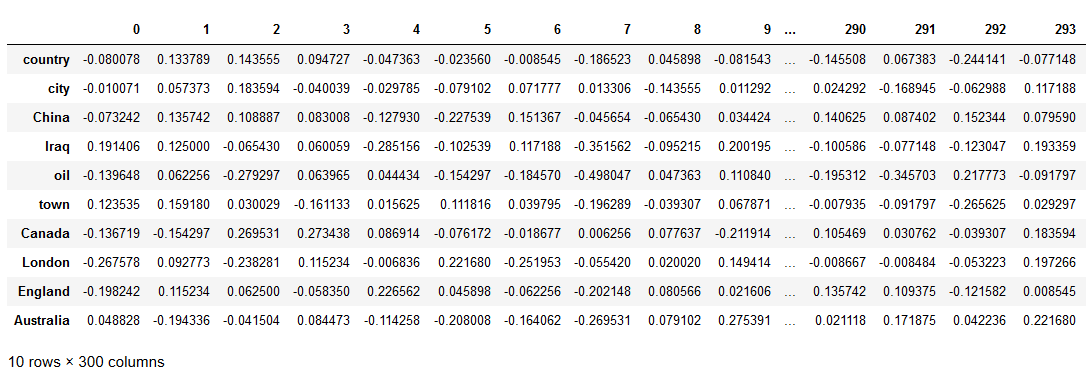
We can observe that the vector 'country' that we expected to be the same as the vector for Spain is not exactly it.



Complete



Data

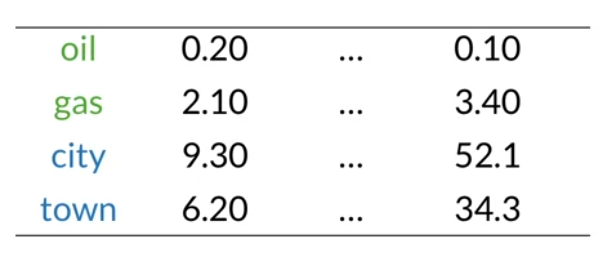


Complete

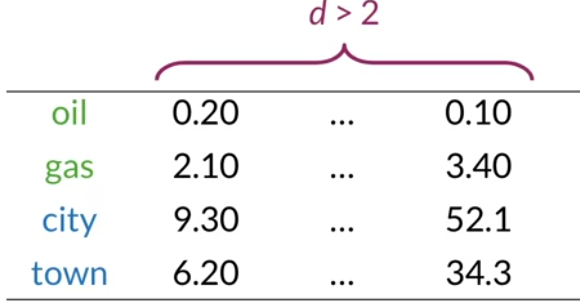


Visualization and PCA

* Visualizing vector presentation of words.
* Principal Component analysis (how to used for dimensionality reduction)



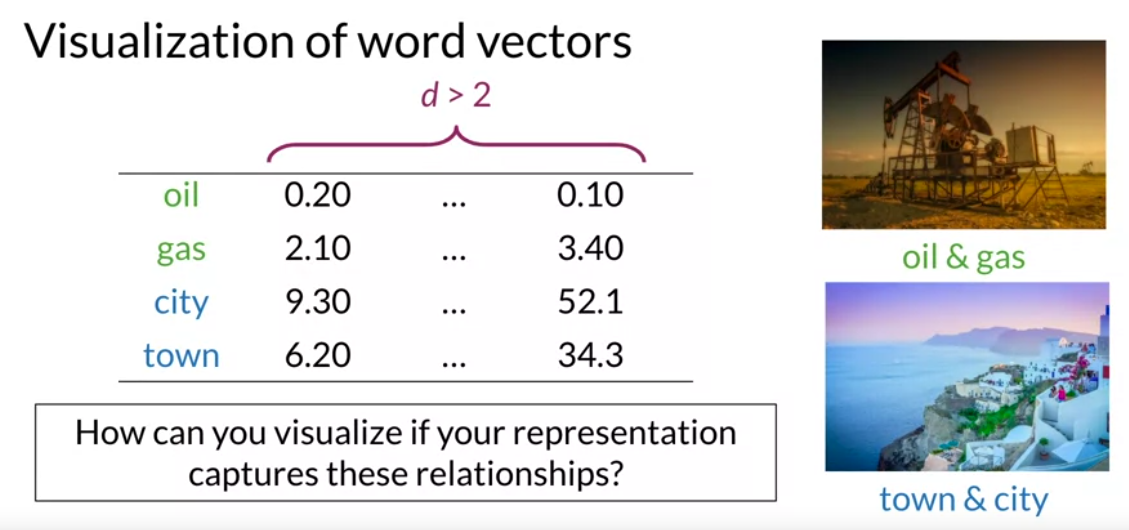
Imagine, you have word, you vector space dimension is higher than tw.



You know that the words oil and gas, and city and town are related .

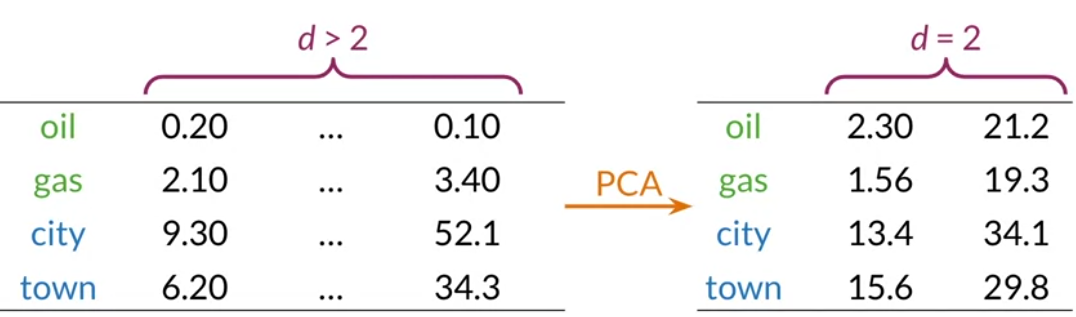
And you want to see if that relationship is captured by the representation of your words.

So how could you visualize your words in order to see this and other possible relationships?



Dimensionality reduction is perfect choice for this task. When you have a representation of your words in a high dimensional space.

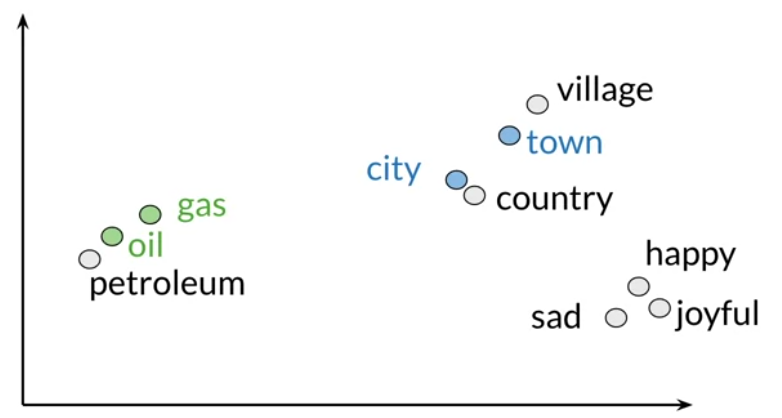
You can use algorithm like PCA



To get a representation on a vector space with fewer dimensions.

If you want to visualize your data, you can get reduced representation with three or two fewer features.

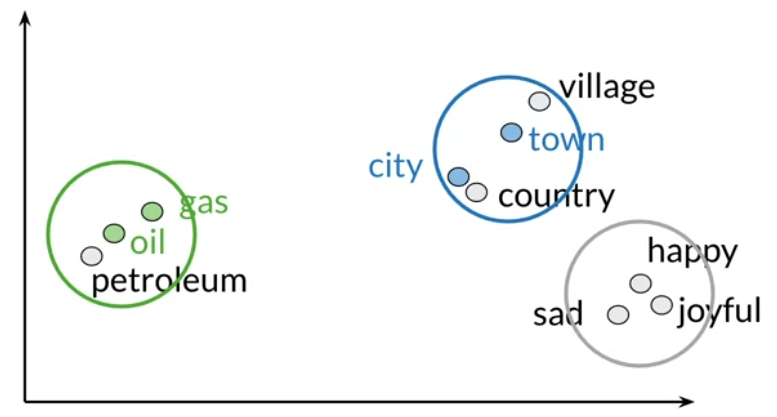
If you perform PCA on your data and get a 2-dim you can then plot visual of your words.



In this case, you might find that your initial representation captured the relationship between the words oil and gas and city and town.

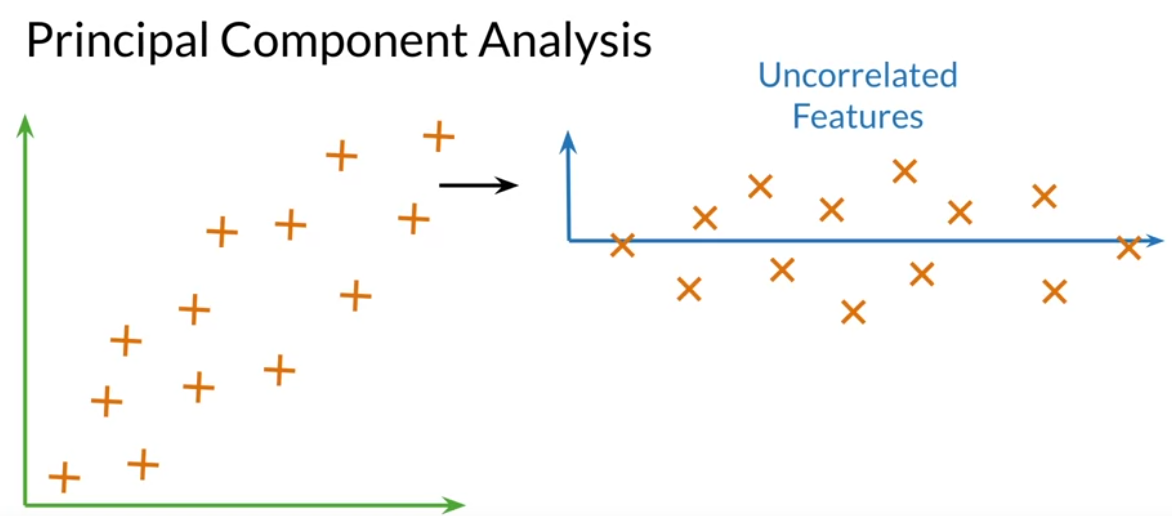
Because in your 2-dim space they appear to be clustered with related words.

You can even find other relationships among your words that your didn`t expect, which is a fun and useful possibility.

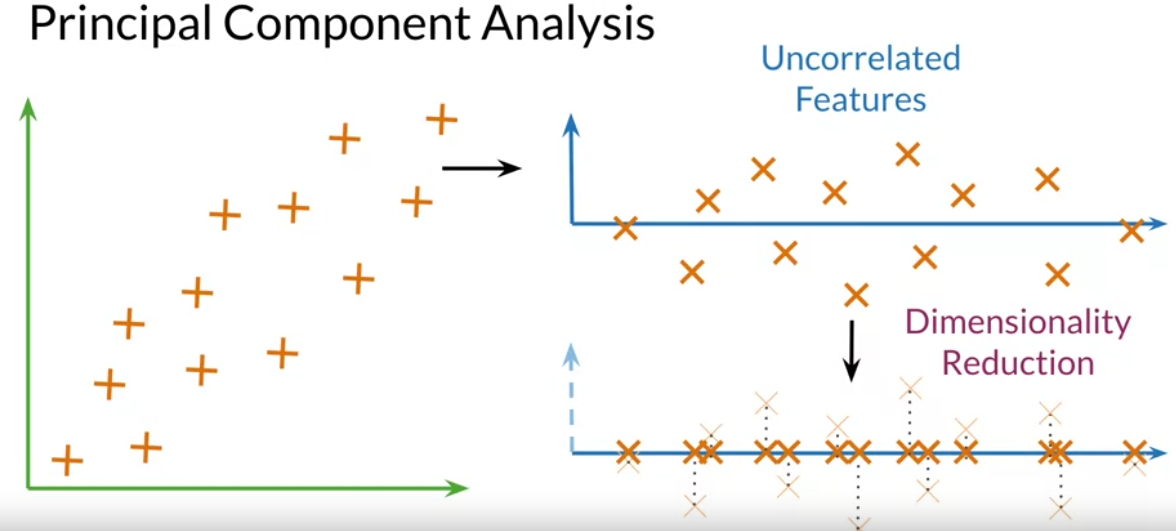


For the sake of simplicity, I will begin with a two dim vector space.

1. Say that you want your data to be represented by one feature instead . Using PCA, first you will find a set of uncorrelated features.



1. And then projects your data to a 1-dim space, trying to retain as much information as possible.



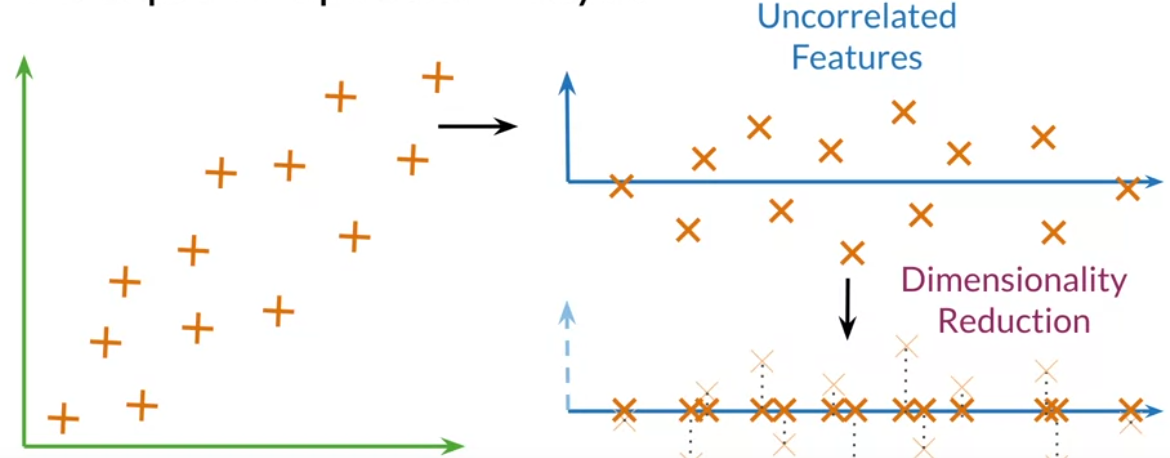
As you can see, this process is quite straightforward.

PCA : it is algorithm used for dimensionality reduction that can find uncorrelated features for your data. It`s very helpful for visualize your data to check if your representation is capturing relationships among words.

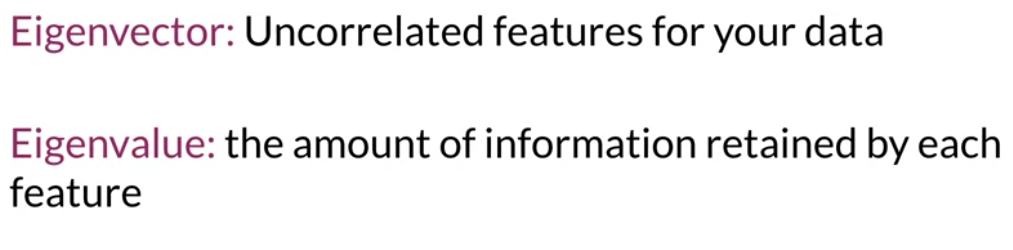
You will learn about eigenvalues and eigenvectors.

To perform dim reduction using PCA, begin with your original vector space. Then you get uncorrelated features for your data.

And finally, you projection you data to a number of desired features that retain the most information.



You may recall from algebra that metrices have eigenvectors and eigenvalues.

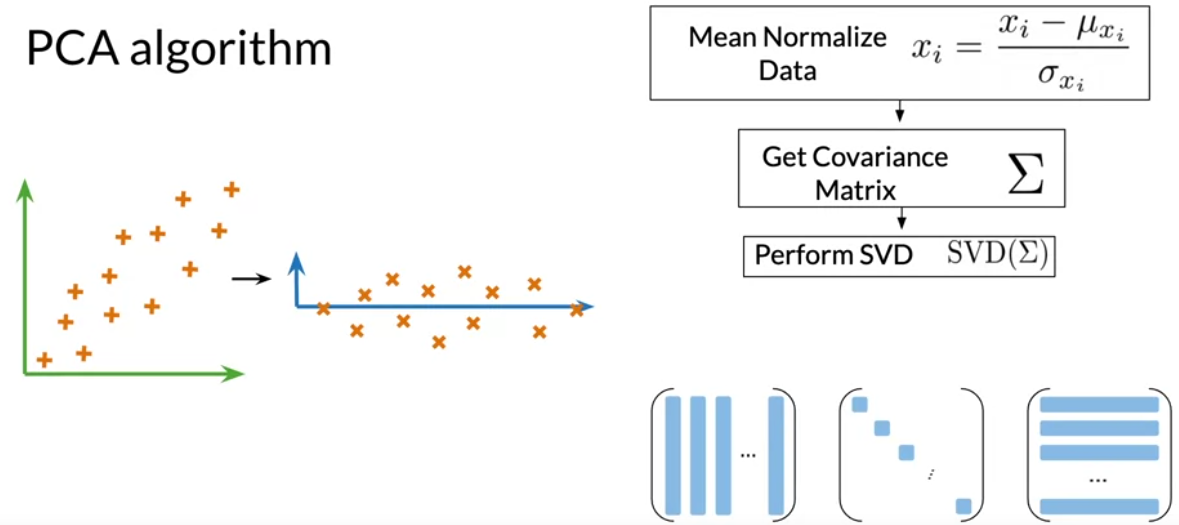


Eigenvectors of the covariance matrix from your data, they give direction of uncorrelated features.

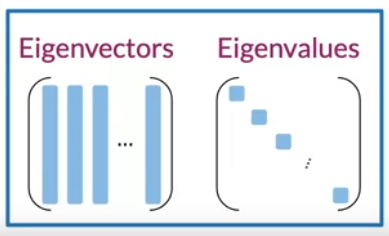
Eigenvalues: are the variance of your data sets in each of those new features

Need to get Eigenvectors and Eigenvalues from the covariance matrix of your data.

1. The first step to get a set of uncorrelated features for this step you will mean normalize your data, then get the covariance matrix and finally a singular value decompression to get set of three metrices.



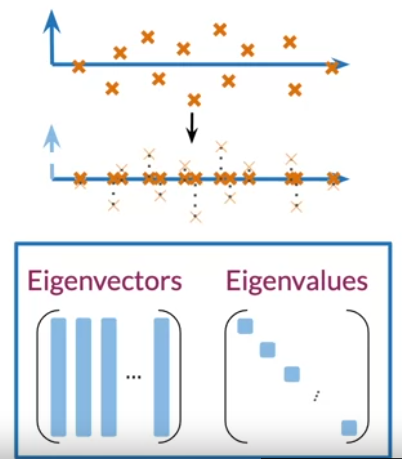
The first of those metrices contain the eigenvector and eigenvalues staked column wise.



On second one has the eigenvalues on the diagonal.

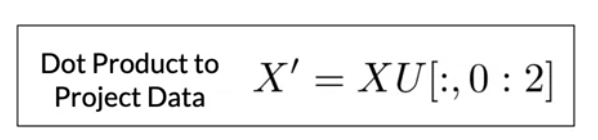
The singular vector decomposition is already implemented in many programming libraries.

The next step is to project your data to a new set of features. You will be using the eigenvectors and eigenvalues in this step.

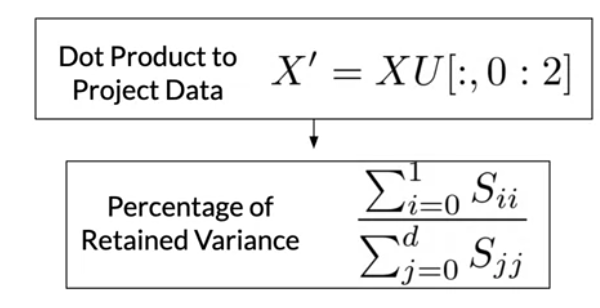


Let`s denote the eigenvectors with U, and the eigenvalues with S.

First, you will perform the dot products between the matrix containing your word embeddings and the first and columns of the U matrix, where n equals the number of dim that you want to have at the end.



For visualization it`s common practice to have 2-dim



Then you will get percentage of variance retained in the new vector space.

As important side note your eigenvectors and eigenvalues should be organized according to the eigenvalues in descending order.

**“ this condition will ensure that you retain as much information as possible from your original embedding . “**

However, most libraries order those matrices for you.