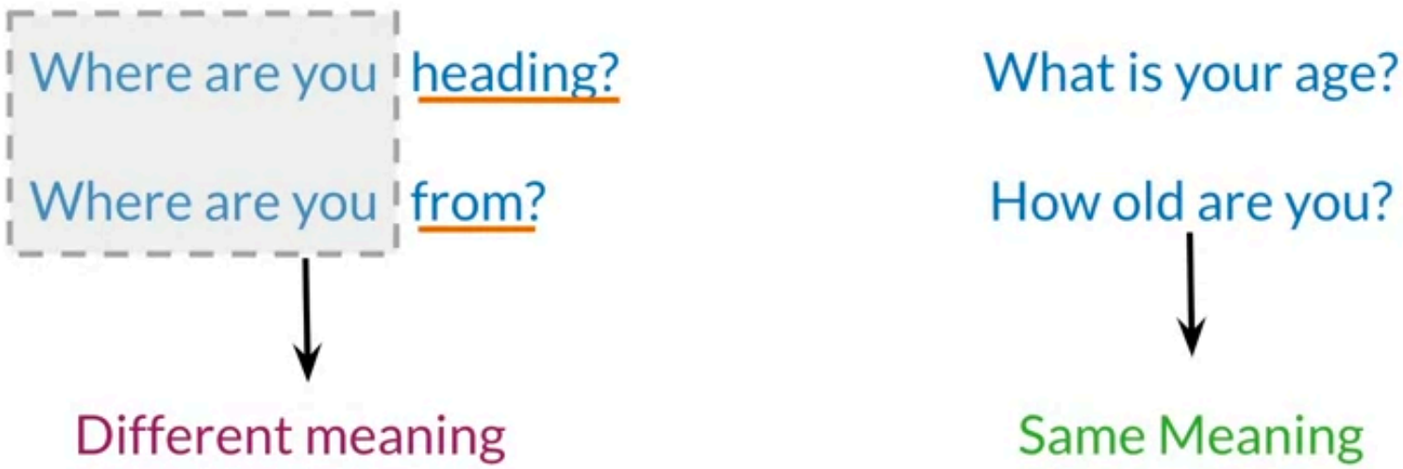
Natural Language Processing with Classification and Vector Spaces

Vector Space Models

2021.02.05

Vector Space Models

- Vector space model identifies similarities and dependencies
  - ex.



Vector Space Models

- Vector space model captures many other types of relationships among different sets of words.
  - Applications: Information Extraction, machine translation, chatbot.
  - The way that representations are made is by identifying the context around each word in the text, and this captures the relative meaning.

- You eat *cereal* from a *bowl*
- You *buy* something and someone else *sells* it

Word by Word & Word by Doc.

- Using co-occurrence matrix → Vector representation
  - **Word by word design**: Number of times they occur together within a certain distance k.

I like simple data

I prefer simple raw data

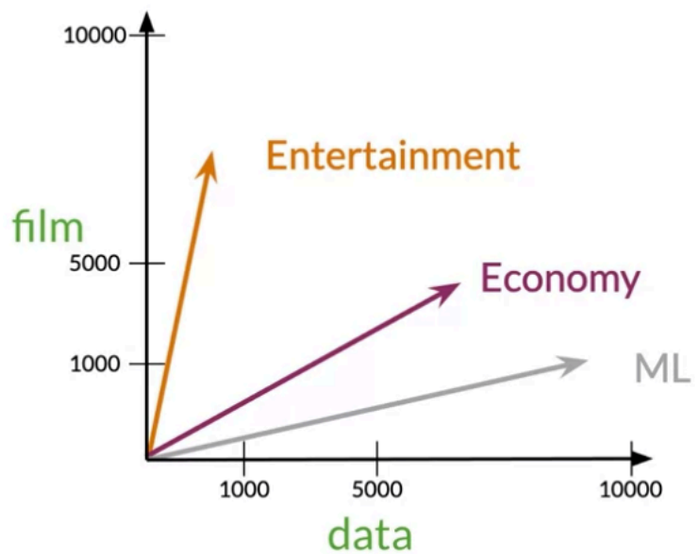
k=2

	simple	raw	like	I
data	2	1	1	0

Word by Word
& Word by Doc.

- Using co-occurrence matrix → Vector representation
  - **Word by document design**: Number of times a word occurs within a certain category.

	Entertainment	Economy	ML
data	500	6620	9320
film	7000	4000	1000

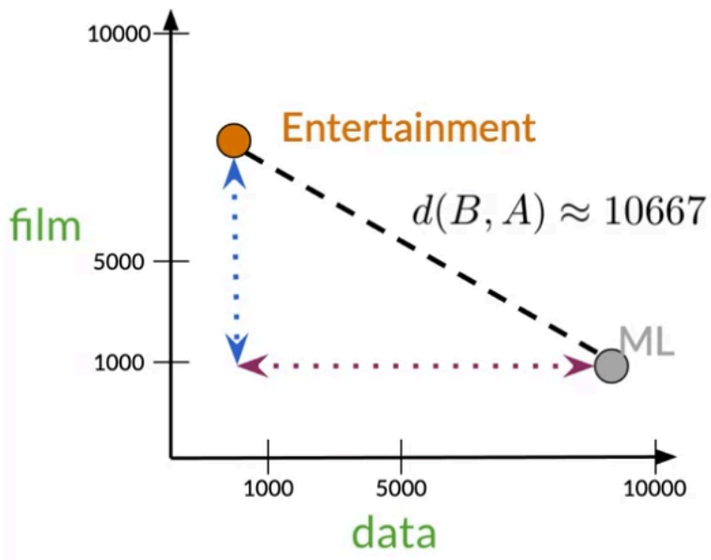


Euclidean Distance

- Similarity metric: the length of the straight line between points.
  -

Corpus A: (500,7000)

Corpus B: (9320,1000)



$$d(B, A) = \sqrt{(B_1 - A_1)^2 + (B_2 - A_2)^2}$$

$$c^2 = a^2 + b^2$$

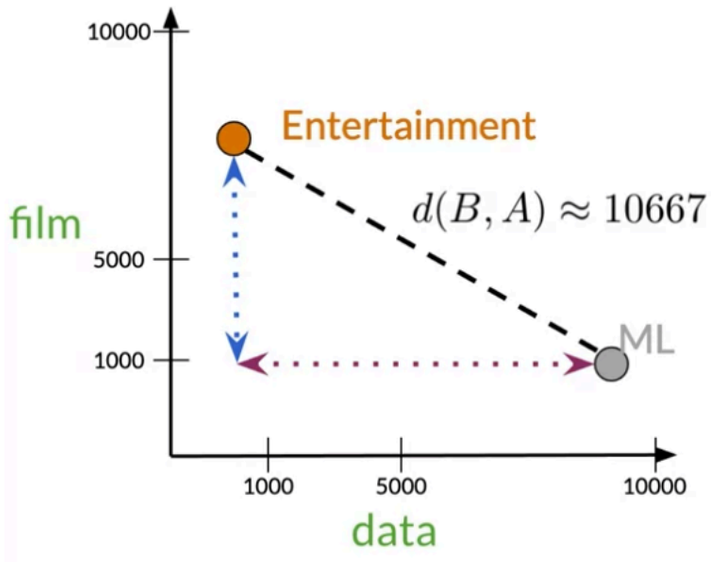
$$d(B, A) = \sqrt{(8820)^2 + (-6000)^2}$$

Euclidean Distance

- Similarity metric: the length of the straight line between points.

Corpus A: (500,7000)

Corpus B: (9320,1000)



$$d(B, A) = \sqrt{(B_1 - A_1)^2 + (B_2 - A_2)^2}$$

$$c^2 = a^2 + b^2$$

$$d(B, A) = \sqrt{(8820)^2 + (-6000)^2}$$

Euclidean Distance

- For n-dimensional vectors.

		$\vec{w}$	$\vec{v}$
	data	boba	ice-cream
AI	6	0	1
drinks	0	4	6
food	0	6	8

$$= \sqrt{(1-0)^2 + (6-4)^2 + (8-6)^2}$$

$$= \sqrt{1+4+4} = \sqrt{9} = 3$$

$$d(\vec{v}, \vec{w}) = \sqrt{\sum_{i=1}^{n} (v_i - w_i)^2} \longrightarrow \text{Norm of } (\vec{v} - \vec{w})$$

- In Python,

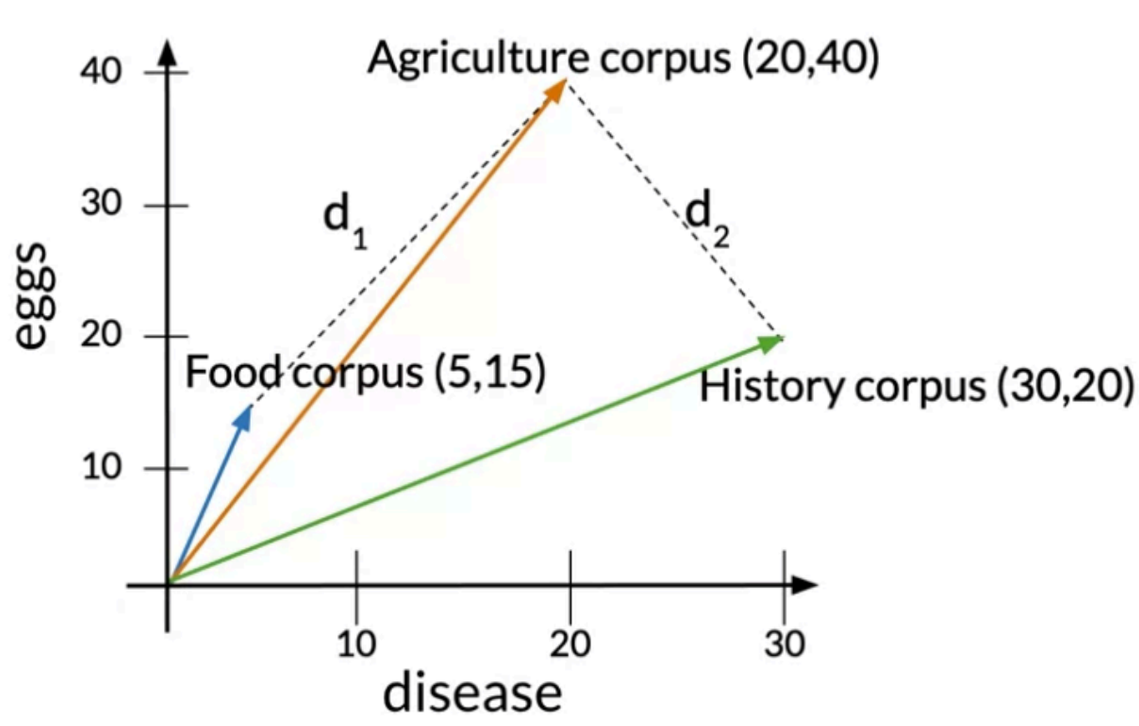
```
# Create numpy vectors v and w
v = np.array([1, 6, 8])
w = np.array([0, 4, 6])

# Calculate the Euclidean distance d
d = np.linalg.norm(v-w)
# Print the result
print("The Euclidean distance between v and w is: ", d)
```

```
The Euclidean distance between v and w is: 3
```

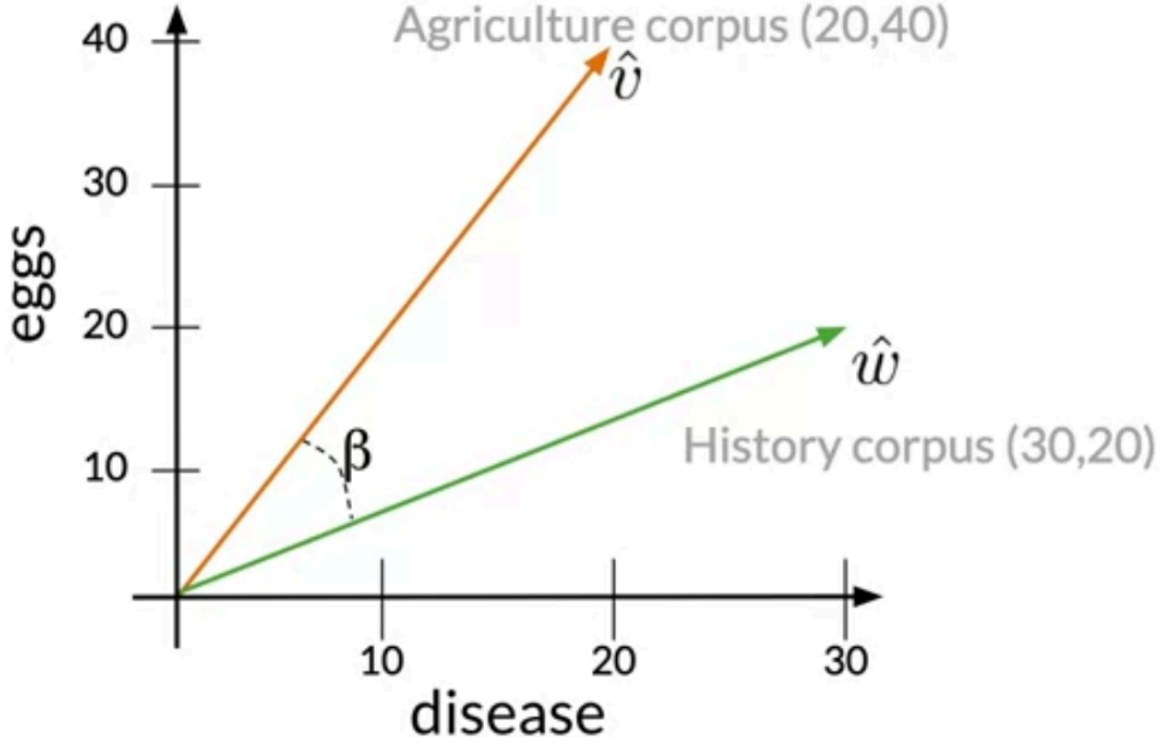
Cosine Similarity

- Problem of Euclidean distance
  - It can be misleading when corpora have different sizes.



Cosine Similarity

- As a similarity metric, it isn't biased by the size difference.

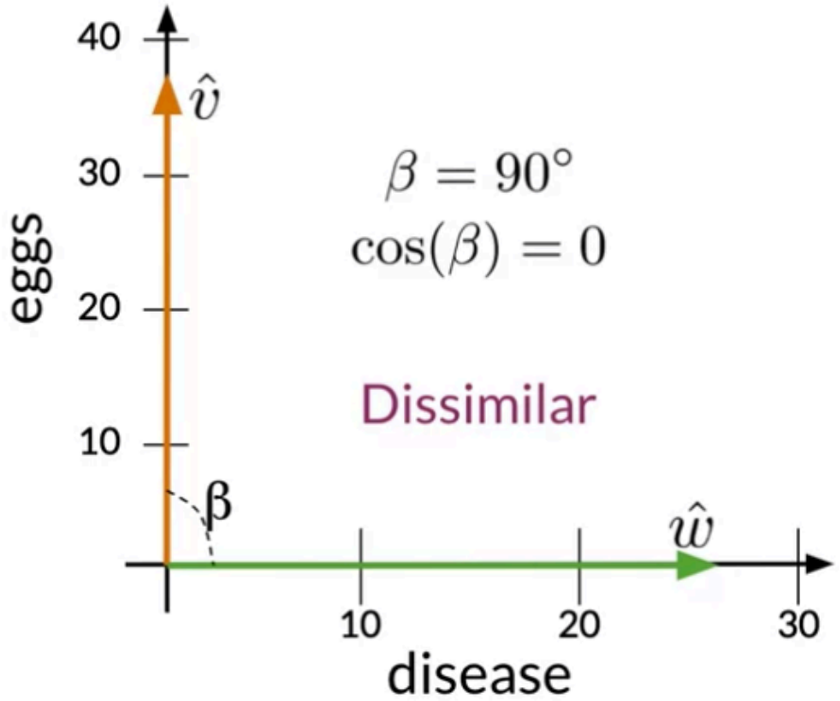


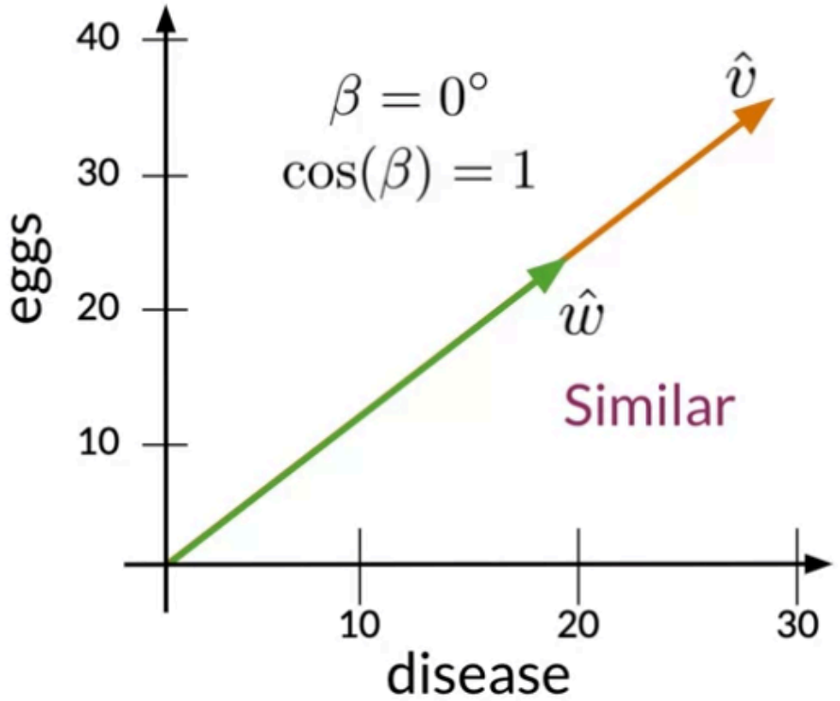
$$\hat{v} \cdot \hat{w} = \|\hat{v}\| \|\hat{w}\| \cos(\beta)$$

$$\cos(\beta) = \frac{\hat{v} \cdot \hat{w}}{\|\hat{v}\| \|\hat{w}\|}$$

Cosine Similarity

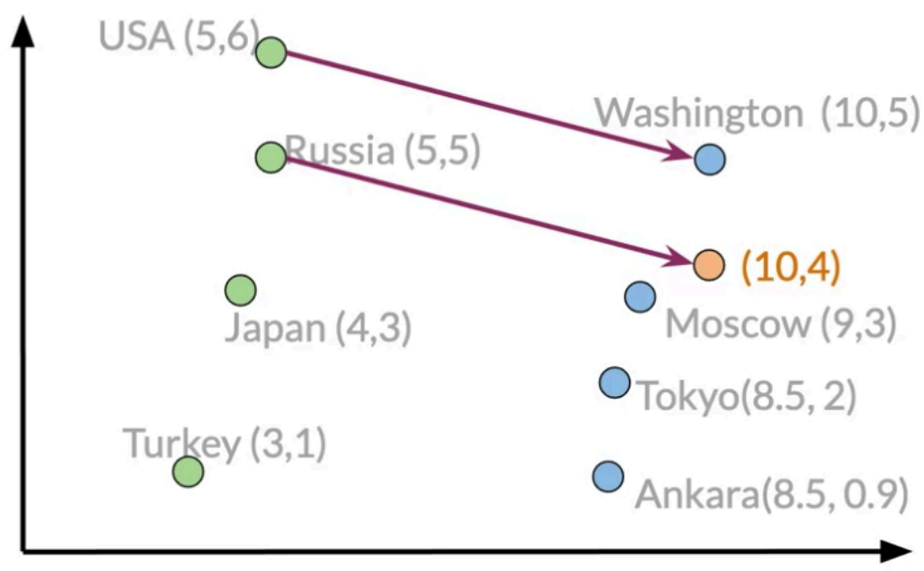
- As a similarity metric, it isn't biased by the size difference.





Manipulating Words in Vector Space

- By performing some simple vector arithmetic, we are able to manipulate vectors.



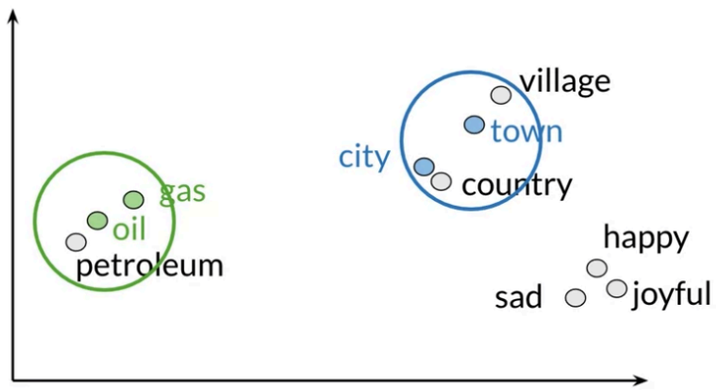
Washington - USA = $\begin{bmatrix} 5 & -1 \end{bmatrix}$

Russia + $\begin{bmatrix} 5 & -1 \end{bmatrix}$ = $\begin{bmatrix} 10 & 4 \end{bmatrix}$

Visualization and PCA

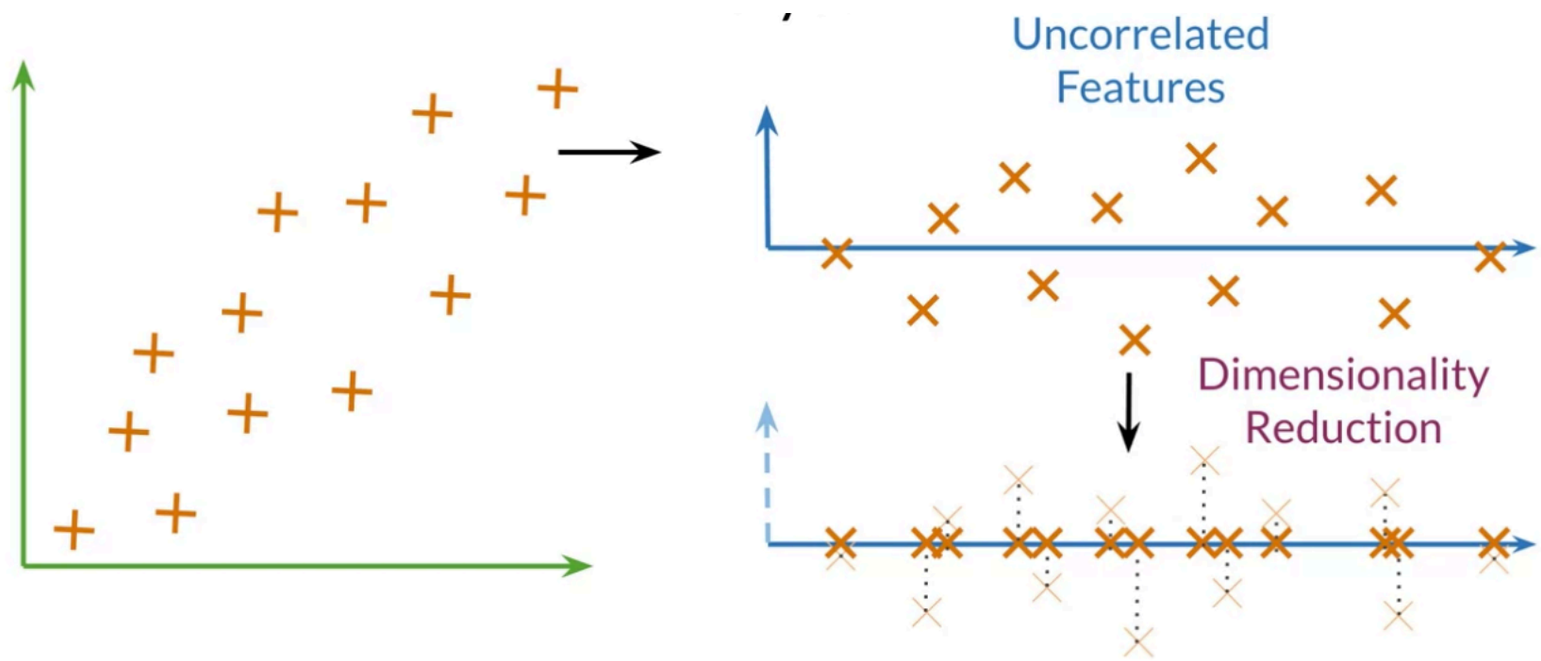
- Dimensionality reduction
  - A way to reduce the dimension of high dimensional vectors to 2-dimensions, so plot it on an XY axis.

- We can find that initial representation captured the relationship between them. (Clustered with related words)



Visualization and PCA

- Principal Component Analysis
  - 2-dimensional vector space → one feature.
  - Find a set of uncorrelated features, then projects data to lower dimensional space.
- While trying to retain as much information as possible.

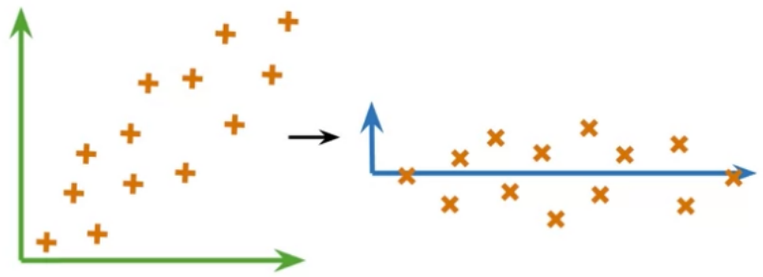


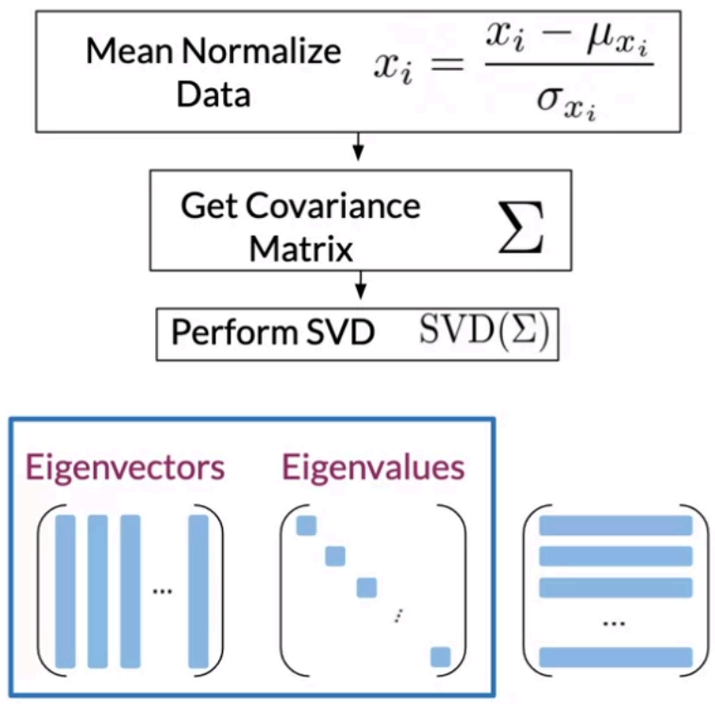
PCA Algorithm

- Two algebra
  - Eigenvector: Uncorrelated features for data.
  - Eigenvalue: the amount of information retained by each feature.
- To perform PCA,
  - Get the eigenvectors and eigenvalues from the covariance matrix of data.

PCA Algorithm

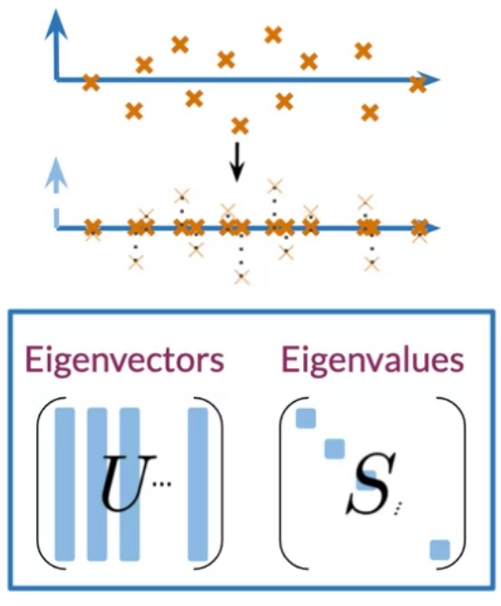
- Perform PCA
  - Get a set of uncorrelated features:
  - Mean normalize data, get covariance matrix, then perform a singular value decomposition to get a set of 3 matrices.





PCA Algorithm

- Perform PCA
  - Project data to a new set of features.
  - Perform the dot products between the matrix containing word embeddings and the 1st and columns of the U matrix.
  - Then get the percentage of variance retained in the new vector space.



Dot Product to Project Data $X' = XU[:, 0:2]$

Percentage of Retained Variance $\frac{\sum_{i=0}^{1} S_{ii}}{\sum_{j=0}^{d} S_{jj}}$