# Unsupervised Learning: Vectorization / Embedding

Parcours Progis

Etudes, Medias, communication, Marketing

Bahareh Afshinpour

02.03.2026

Université Grenoble Alpes

Sciences Po Grenoble UGA

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

# Suggested Reading

- Jurafsky & Martin – "Speech and Language Processing" (3rd ed., Draft), chapter 6

- A Complete Overview of Word Embeddings, https://www.youtube.com/watch?v=5MaWmXwxFNQ

- What is Word2Vec? A Simple Explanation , https://www.youtube.com/watch?v=hQwFeIupNP0

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

# Hard vs soft clustering

- Hard clustering:
  - Eache data point belongs to exactly one cluser
  - More common and easier to use

  *K-means Clustering: When segmenting customers into three groups (e.g., high, medium, and low spenders), each customer is assigned to only one group based on their features.*

- Soft clustering:
  - Each sample is assigned to different clusters with probabilities, rather than 0 and 1.
  - Data point belongs to each cluster with a probability

  *Fuzzy C-Means Clustering: When grouping articles by topics, an article can be 70% related to "Technology" and 30% related to "Health," thus belonging partially to both clusters.*
  *Gaussian Mixture Models: In image segmentation, a pixel may have a 60% probability of belonging to the "background" and a 40% probability of being part of the "object."*

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

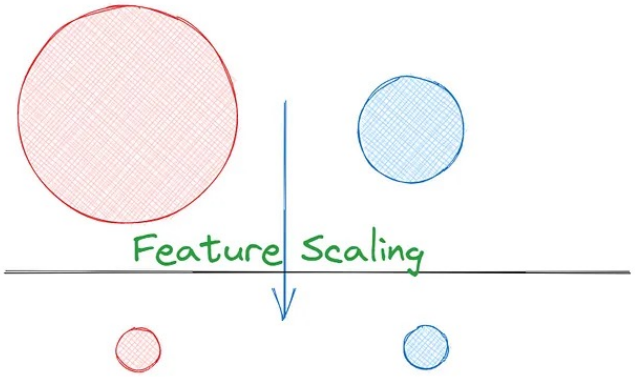
# Scaling:

- scaling is a crucial step because machine learning models are "sensitive to distance.
- Since "Age" (11-17) and "Number of people in house" (1-6) are on different scales, you "scale" them so the model doesn't think Age is more important just because the number is bigger.
- The two main method:
  - 1. Standardization (StandardScaler): This centers the data around 0 with a standard deviation of 1. It is best for **Regression** and **Clustering**.
  - 2. Normalization (MinMaxScaler): This shrinks the data strictly between 0 and 1.

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

# Feature Scaling

- Imagine you’re comparing apples and oranges; it's nonsensical.
- Features measured in vastly different units (like income and age) can skew machine learning algorithms if not addressed.
- Feature scaling addresses this by transforming data into a standard scale, enabling fair comparison between different features.




https://medium.com/@yashbaravaliya206/feature-scaling-and-normalization-ca484a16882a

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

# Feature Scaling

- Feature scaling is a fundamental preprocessing step in machine learning aimed at ensuring that numerical features have a similar scale.
- Apply feature scaling before feeding the data into the machine learning model (K-NN), **except** for algorithms that are scale-invariant, such as **decision trees**.

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

# Common Techniques for Feature Scaling

- **Normalization :**
  - This method scales each feature so that all values are within the range of 0 and 1.
  - x is the value you want to normalize
  - min(X) is the minimum value in your data set
  - max(X) is the maximum value in your data set

$$(x - \min(X)) / (\max(X) - \min(X))$$

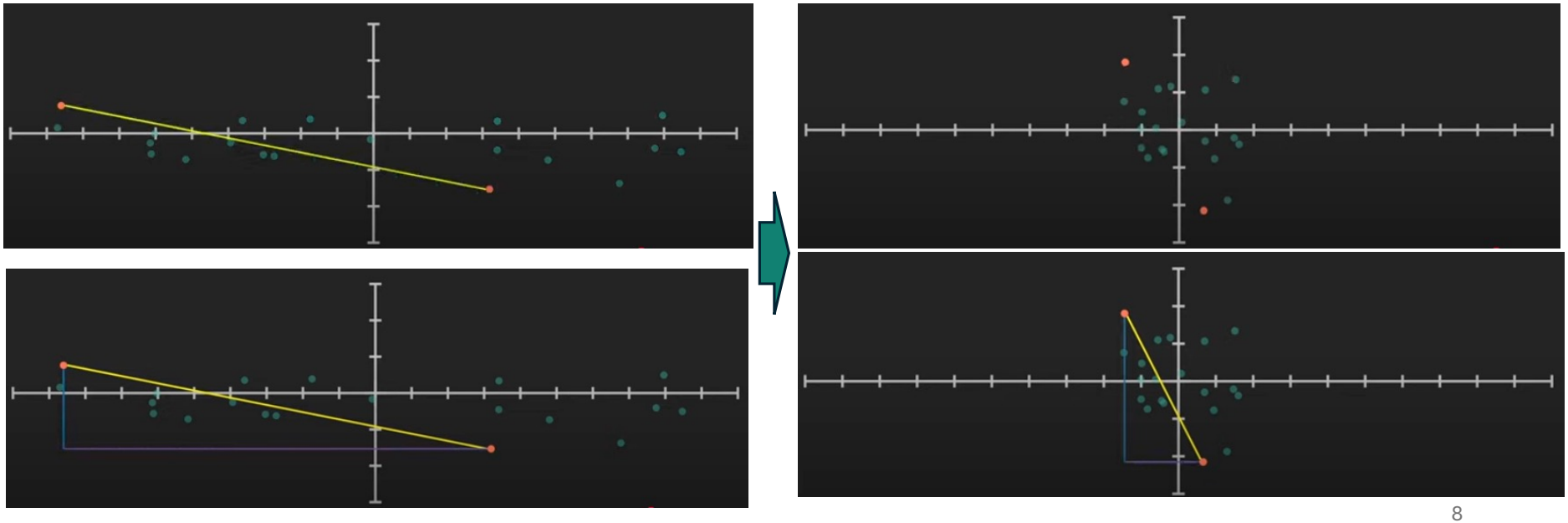
- **Standardization (Z-score Scaling):**
  - Here, each feature is transformed to have a mean of 0 and a standard deviation of 1.
  - This is achieved by subtracting the mean value and dividing by the standard deviation of the feature.
  - x is the original value you want to standardize, μ (mu) is the mean of the data set, σ (sigma) is the standard deviation of the data set

$$Z = (x - \mu) / \sigma$$

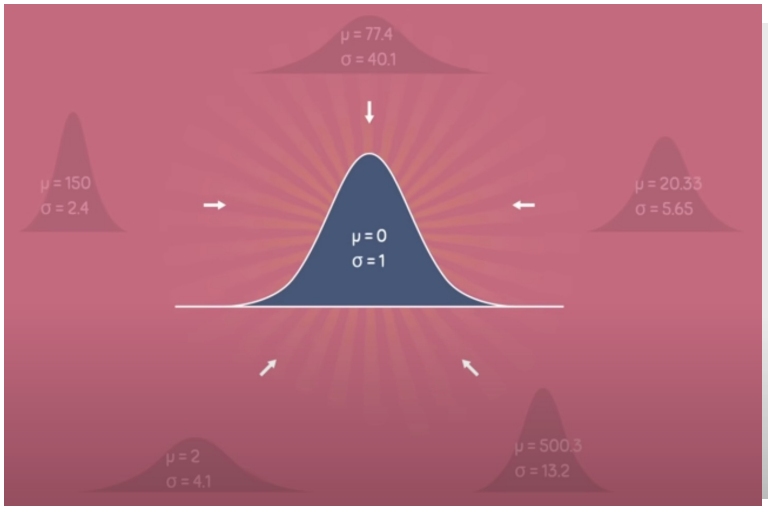
PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

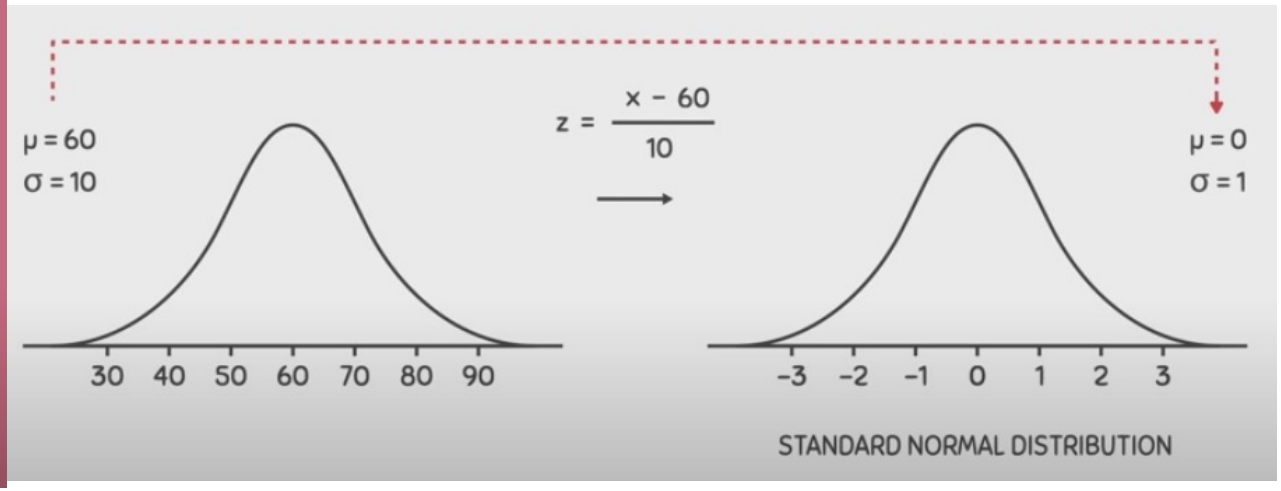
# Feature Scaling

- In the last figure, both feature contribute equally to the distance
- After that, your algorithm won't be affected by the feature with a higher scale.



```
from sklearn.preprocessing import StandardScaler
stdsc=StandardScaler()
X_std=pd.DataFrame(stdsc.fit_transform(X_all), index=X_all.index, columns=X_all.columns)
```




Any normal distribution with any value of mean and standard deviation(sigma) can be transformed into the standard normal distribution where we have a mean of zero and a standars deviation of one.

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

# word embedding

- Word embeddings are techniques that map words or phrases from vocabulary to numerical vectors.

- Applications: Text classification, sentiment analysis, machine translation, question answering

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

# Why do we need word embedding at all

- When we are working with NLP models, we are working with **text**.
- Text is not good for machine learning models:
  - What machine learning methods know what to do with, is **numbers**.
  - So you need to present your text in **numbers** format.

**This is my book.**

| | | | | |
|---|---|---|---|---|
| 0.000 | 0.006 | -0.013 | ... | -0.013 |

Text as vector

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

# One hot encoding

- Create **one vector (**really long **)** that is as long as the number of words that you have in your vocabulary.
- To present **each word**, we fill this vector :
  - with zeros except for the cell that corresponds to the word that we are trying to represent.

**Vocabulary=[I, you, book, cat, flowers,is, this,, they, are, my]**

| | I | you | Book | cat | Flowers | Is | this | they | are | my |
|---|---|---|---|---|---|---|---|---|---|---|
| **Book** | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **cat** | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

**It is not the most efficient use of space**

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

# Bag-of-words

- Is a simple and widely used technique in natural language processing to represent text data. (a collection or "bag" of its words)
- Do not think about order of words in the sentence
- How many times each word occurs

| | Small | dog | cat | and | cute |
|---|---|---|---|---|---|
| **Small dog** | 1 | 1 | 0 | 0 | 0 |
| **Cute cat and cute dog** | 0 | 1 | 1 | 1 | 2 |

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

# TF-IDF

- Keep track of how many times a word occurs in a document or sentence. And how many times this word occurs in **other** documents or sentences throughout the training data.
- Aim: differentiate the words that are commonly used (and, or, is …) and the words that are very important for a certain sentence or document.

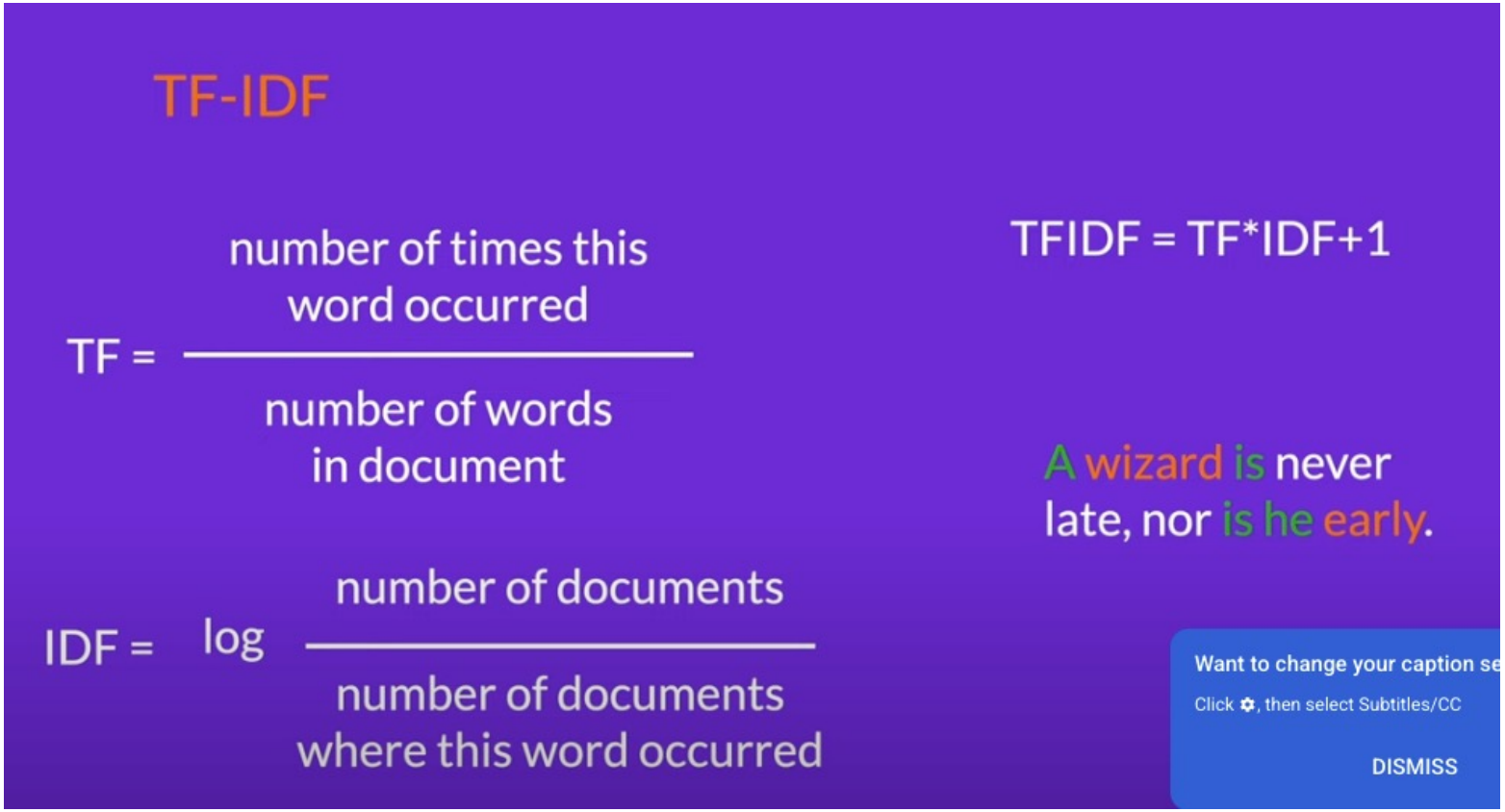
$$TF(t,d) = \frac{(Number\ of\ occurrences\ of\ term\ t\ in\ document\ d)}{(Total\ number\ of\ terms\ in\ the\ document\ d)}$$

$$IDF(t,D) = \log_e \frac{(Total\ number\ of\ documents\ in\ the\ corpus)}{(Number\ of\ documents\ with\ term\ t\ in\ them)}$$

$$TF\text{-}IDF(t,d,D) = TF(t,d) \times IDF(t,D)$$

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

# TF-IDF



PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

# Challenges

- They have some serious shortcomings:
  - They can not deal with words that they did not see in the training examples (new words)
  - They embeding that they produce are very sparse (sparse vector)

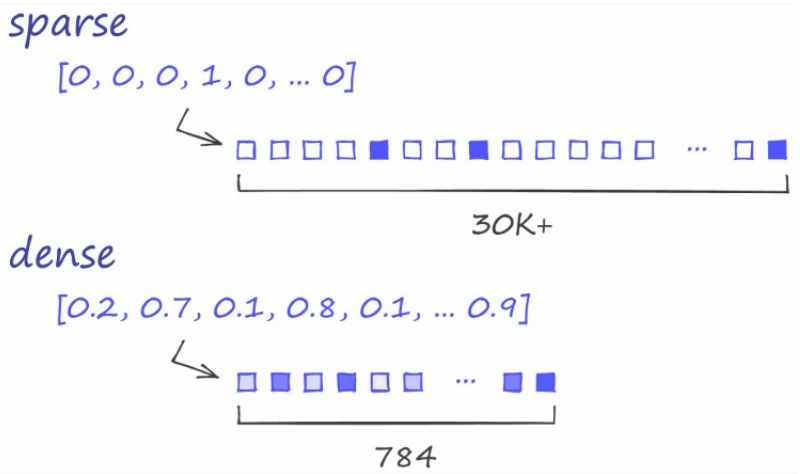
*Sparse vectors are called sparse because vectors are sparsely populated with information. Typically we would be looking at thousands of zeros to find a few ones (our relevant information).*

**Embeddings aim to represent the word in a dense vector while making sure that similar words are close to each other in the embedding space.**

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

# What is a dense vector

- Dense vector means that :
  - ➢The vector representing the word does not mostly consist of 0 and
  - ➢The embedding vector has fewer dimensions that the number of words in vocabulary.

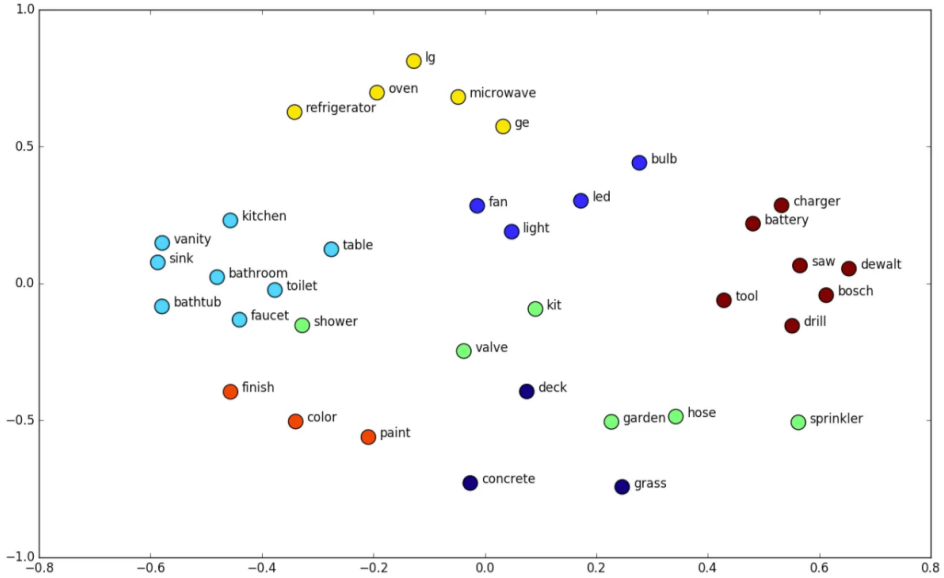



https://www.pinecone.io/learn/series/nlp/dense-vector-embeddings-nlp/

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

# Embedding space

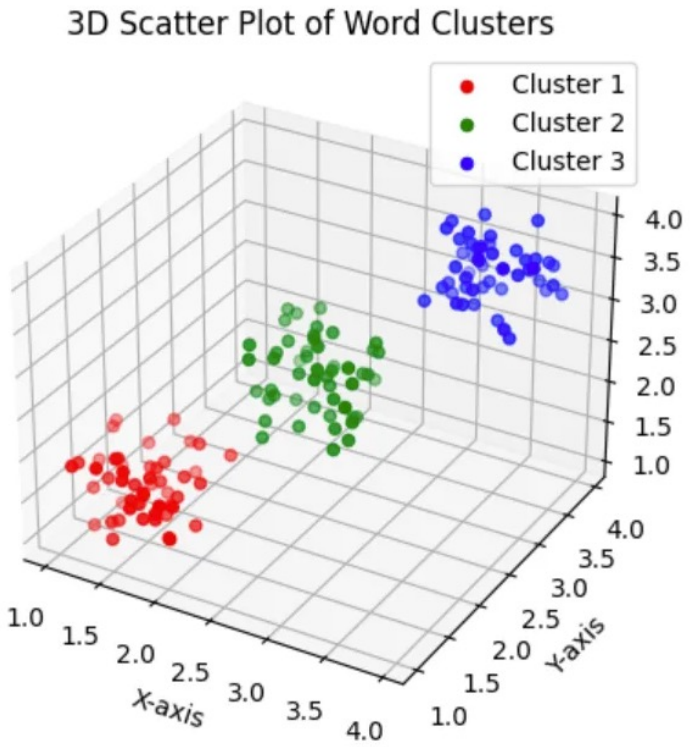
- Embedding space is where your embedded data lives.
- D=2




https://medium.com/opla/how-to-train-word-embeddings-using-small-datasets-9ced58b58fde

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

# D=3



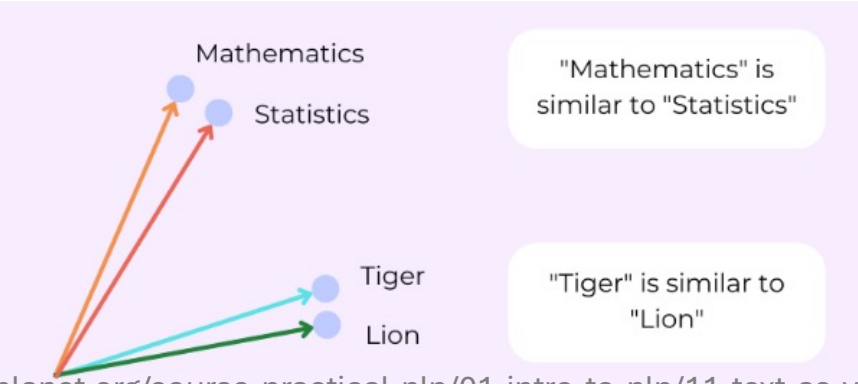

We can not visualize the embedding in 20 dimension space, but we can calculate the similarity

https://honnuanand.medium.com/navigating-the-world-of-vector-embeddings-from-basics-to-visualization-6cf3f9cc2533

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

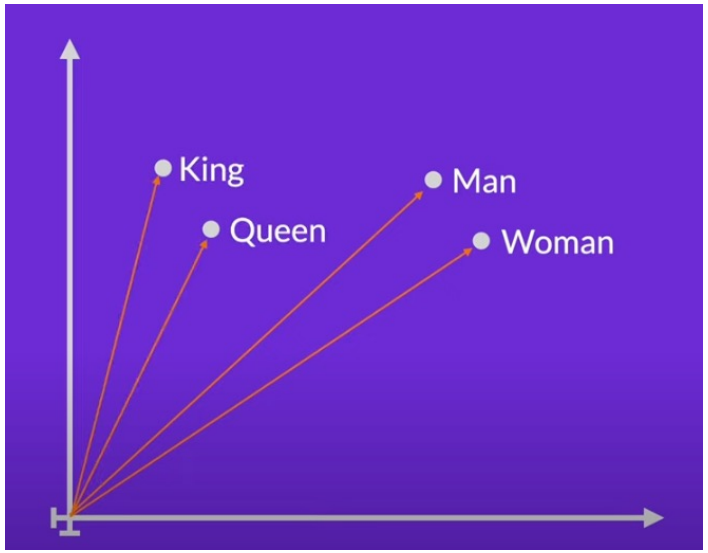
# What are similar words?

- The words that are used in similar or **same context** most of the time.
- They being used around same word.
  - For example: tea and coffee. (they are similar words)
  - But tea and pea they are not similar. Even though they are spelled really similarly. Since they used in vastly different contexts.




https://www.nlplanet.org/course-practical-nlp/01-intro-to-nlp/11-text-as-vectors-embeddings

- For some cases, it is even possiblle to make sure that the relative distances between word represent contextual information.



PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

# Word Similarity

- Knowing how **similar two words** are can help in computing how **similar the meaning** of two phrases or sentences are.
- One way of getting values for word similarity is to ask humans to judge how similar one word is to another. A number of datasets have resulted from such experiments

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

# Word embedding

- Vectors for representing words are called embeddings


-Words with similar meanings are nearby in space.
-Notice the distinct regions containing positive words, negative words, and neutral function words.

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

# Pourquoi les word embeddings ?

**Problème avec les représentations traditionnelles** (ex : one-hot encoding)

- Pas de notion de similarité sémantique
- Vecteurs très grands et creux (sparse)

- **Mais des embeddings** : représenter les mots par des vecteurs **denses** qui capturent la **sémantique**

Goal of embeddings: to represent words with dense vectors that capture semantics

*“chat” et “chien” proches, “banane” loin.*

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

# Méthodes classiques de word embedding

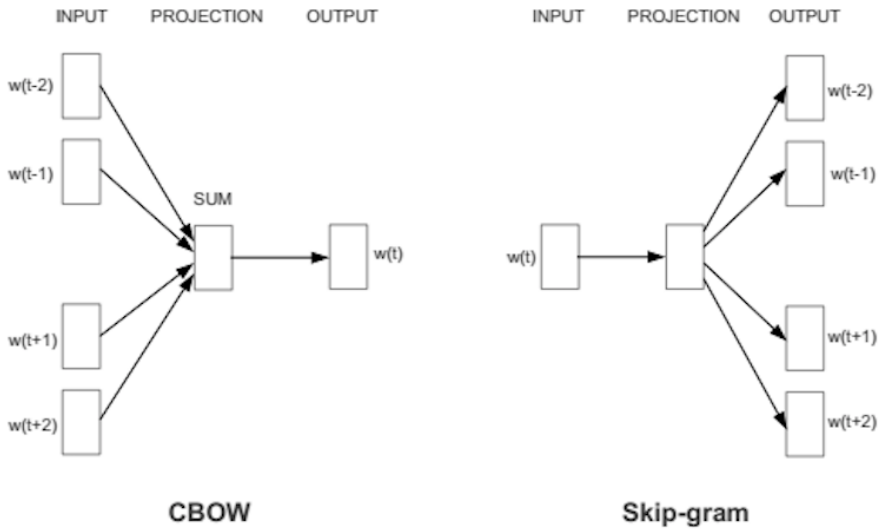
| Méthode | Description rapide |
|---|---|
| **Word2Vec (CBOW & Skip-gram)** | Prédit un mot à partir du contexte ou l'inverse |
| **GloVe** | Utilise des co-occurrences globales |
| **FastText** | Utilise les sous-mots (caractères) → gère mieux les mots rares |

Exemples visuels avec des schémas/animations aident beaucoup ici.

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

# Word2vec

- Predict words using context
- Two versions: CBOW (continuous bag of words) and Skip-gram




https://skymind.ai/wiki/word2vec

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

# Word2vec

- Word2vec is a revolutionary invention in the field of computer science that allows you to represent words in a vector in a very accurate way so that you can do mathematics with it.
- Given a text (lot of sentences), we divide these sentences into groups of n words (windows) and feed it to neaural network.

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

# CBOW : Continuous Bag Of Words

-Given context words predict **target** word.

-We try to guess the word that should be in the middle.

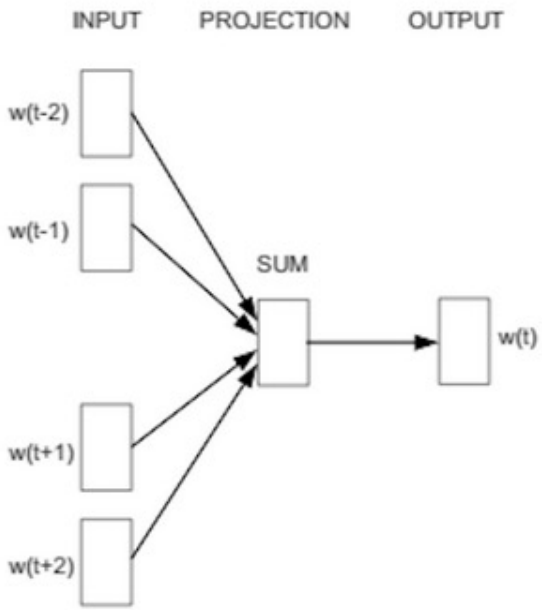
Example:

King ordered his

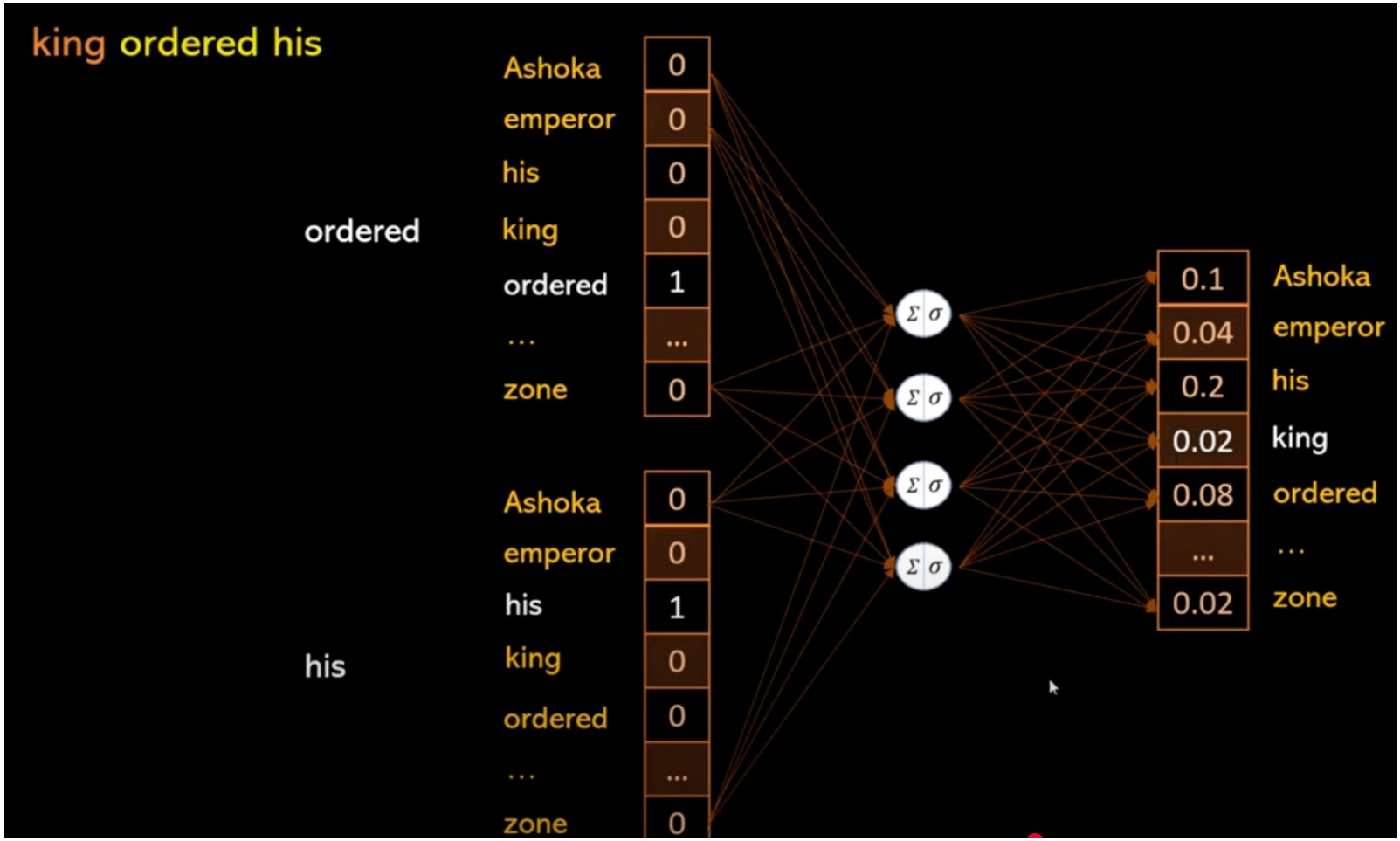
**context**

**target**

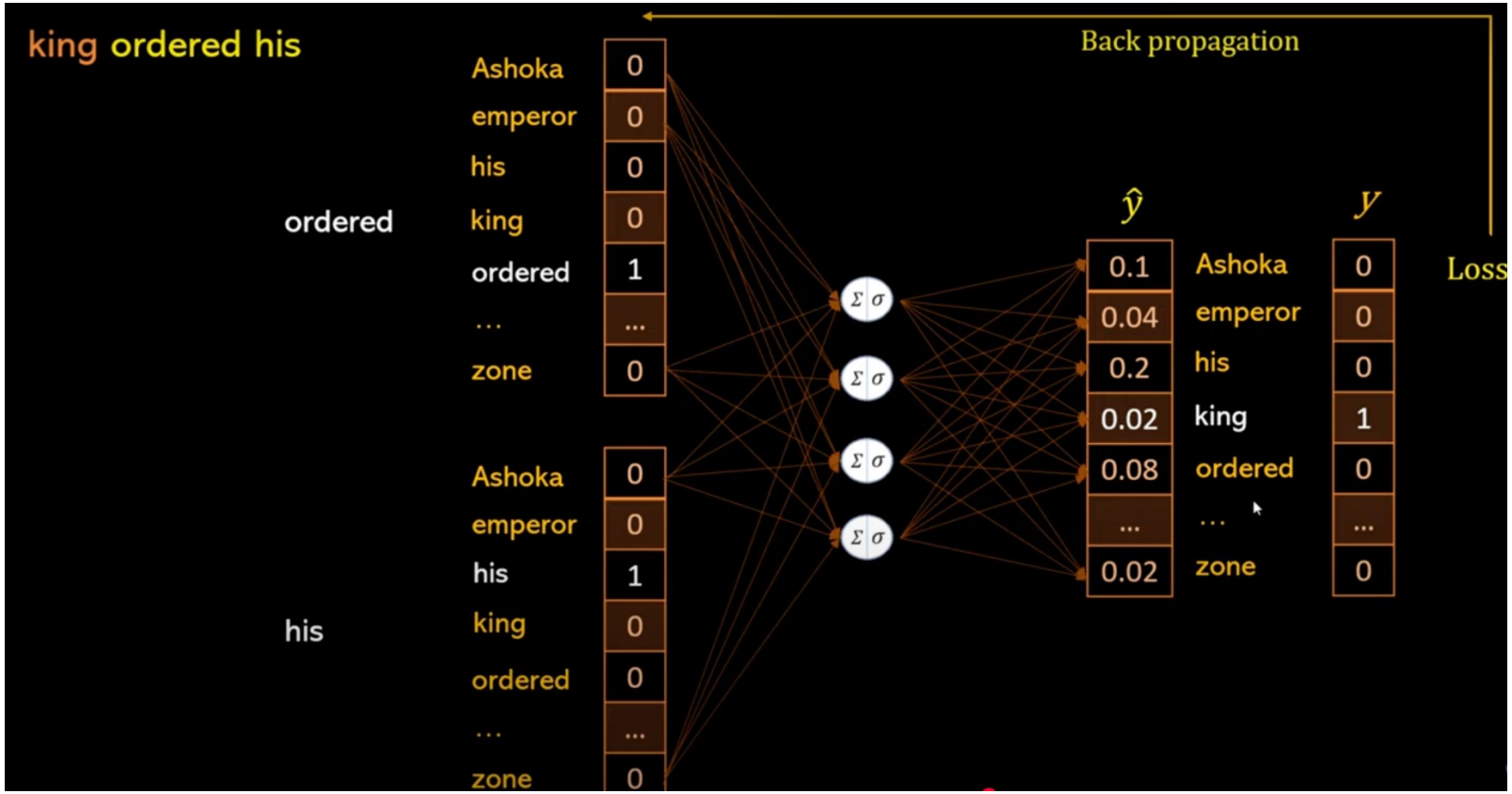
Here, we took a window size 3. it could be window of size 4 or five depends on how you want to experiment.




CBOW


king ordered his
ordered
his
Ashoka
emperor
his
king
ordered
…
zone
0
0
0
0
1
…
0
Ashoka
emperor
his
king
ordered
…
zone
0
0
1
0
0
…
0
Σ σ
Σ σ
Σ σ
Σ σ
0.1
0.04
0.2
0.02
0.08
…
0.02
Ashoka
emperor
his
king
ordered
…
zone

**By doing backpropagation we are adjusting all these weights**

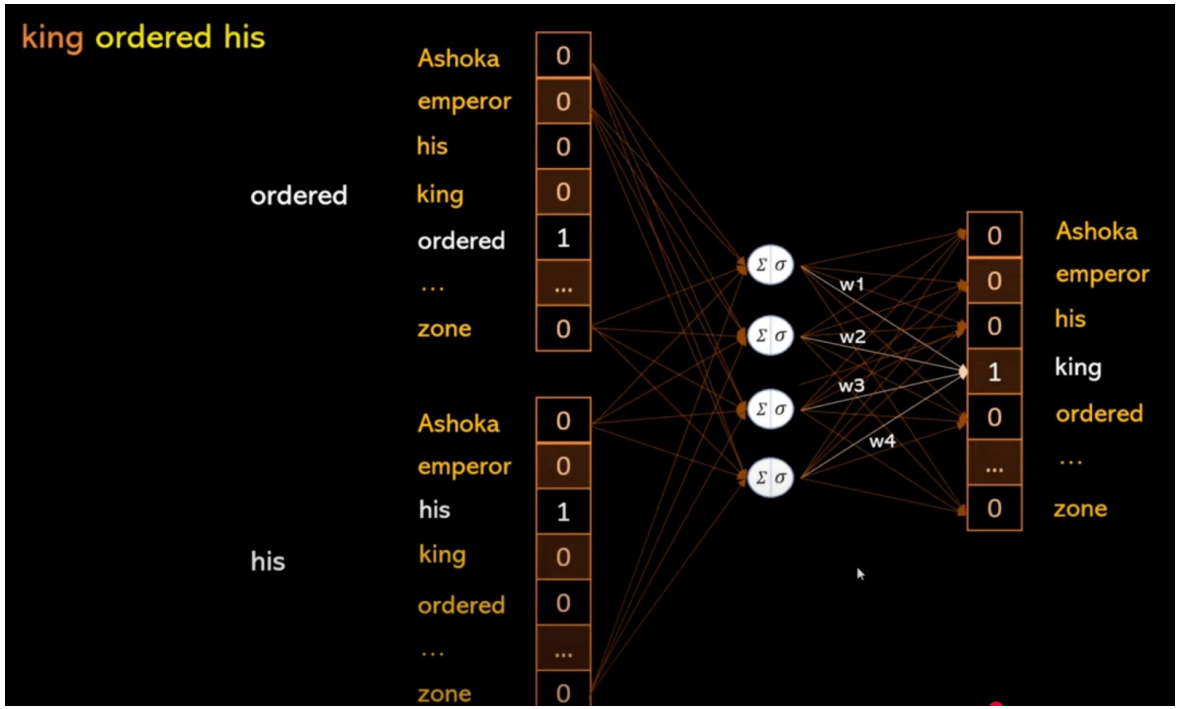



**Compare actual output (y) with predicted output (y hat).**

**Take a loss (difference between actual output and predicted output)**

**Do backpropagate**

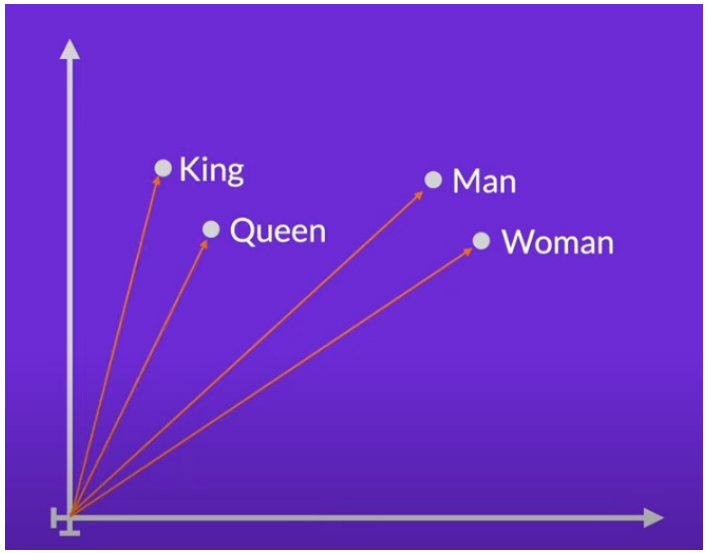
- When you have done feeding your about 1 million elements. Then your neural network is strong.
- At the end, the word vector for king would be these weights. (w1,w2,w3,w4)



PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

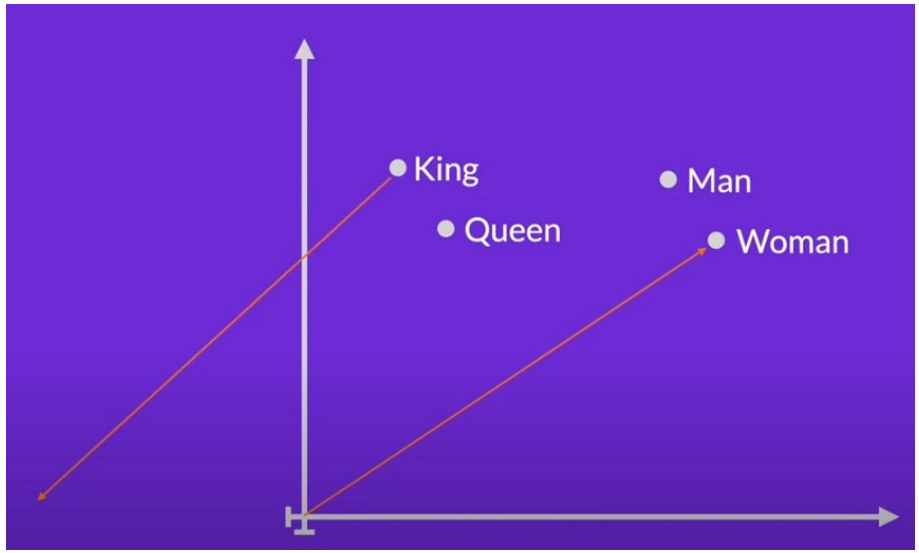
# Some interesting results

- For some cases, it is even possiblle to make sure that the relative distances between word represent contextual information.

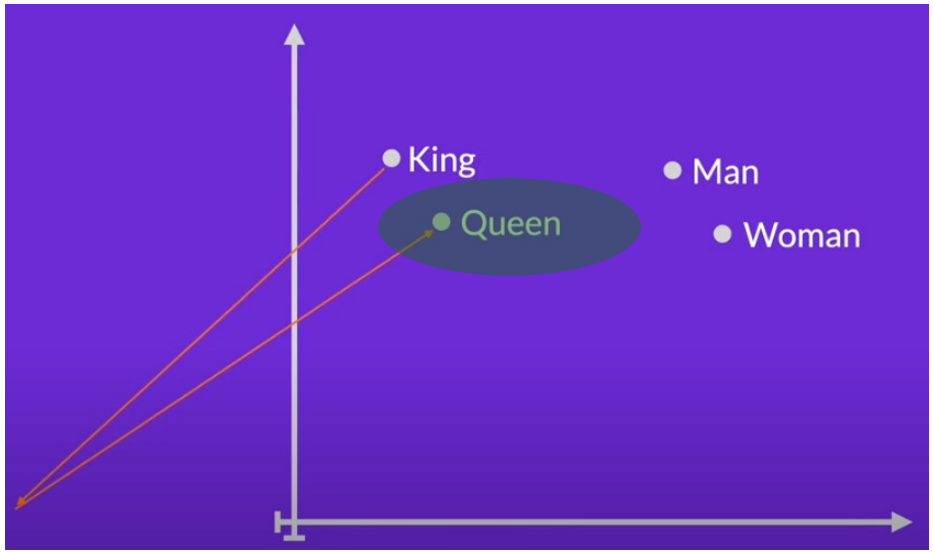


PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

# Some interesting results




**King-man**




**King-man+woman**

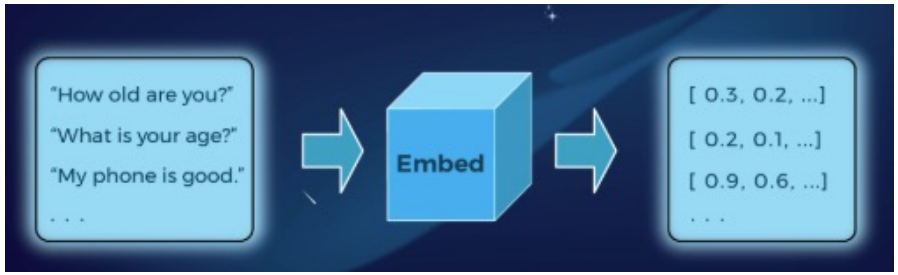
PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

# Sentence embedding

- **Sentence embedding** is a technique in natural language processing (NLP) where an entire sentence is converted into a fixed-size vector (a list of numbers) that captures the *meaning* of the sentence.

Sentence embeddings let us do things like:

- Compare two sentences for **similarity**
- **Feed** sentences into machine learning models



PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

# Sentence embedding

- Consider these two sentences:
  - ❖"I love playing football."
  - ❖"Playing soccer is fun. »

- A good sentence embedding model will generate similar vectors for both, because they mean similar things—even though the words are different.

The **averaging method** can be used to create sentence embeddings—it's one of the simplest and most intuitive approaches.

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

# Sentence embedding

- If a sentence has 3 word embeddings like:
  **[0.2,0.4], [0.1,0.6], [0.3,0.5]**

The sentence embedding by using averaging method would be:

**( [0.2,0.4]+[0.1,0.6]+[0.3,0.5] )/ 3**
**[ (0.2+ 0.1+ 0.3)/3, (0.4+ 0.6+ 0.5)/3 ] = [0.2, 0.5]**

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

# Sentence embedding

- **Popular Methods:**

  - ✓**Averaging Word Embeddings:**
  - ✓**Doc2Vec:**
  - ✓**FastSent**
  - ✓**Sent2Vec**
  - ✓**Sentence-Transformers (all-MiniLM-L6-v2)**

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

# TP7

```
data='./Mall_Customers.csv'
df=pd.read_csv(data)
df.head()
```

| | CustomerID | Genre | Age | Annual Income (k$) | Spending Score (1-100) |
|---|---|---|---|---|---|
| **0** | 1 | Male | 19 | 15 | 39 |
| **1** | 2 | Male | 21 | 15 | 81 |
| **2** | 3 | Female | 20 | 16 | 6 |
| **3** | 4 | Female | 23 | 16 | 77 |
| **4** | 5 | Female | 31 | 17 | 40 |

```
df.shape
```

```
(200, 5)
```

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

# Checking the missing value

```
df.isna().sum()
```

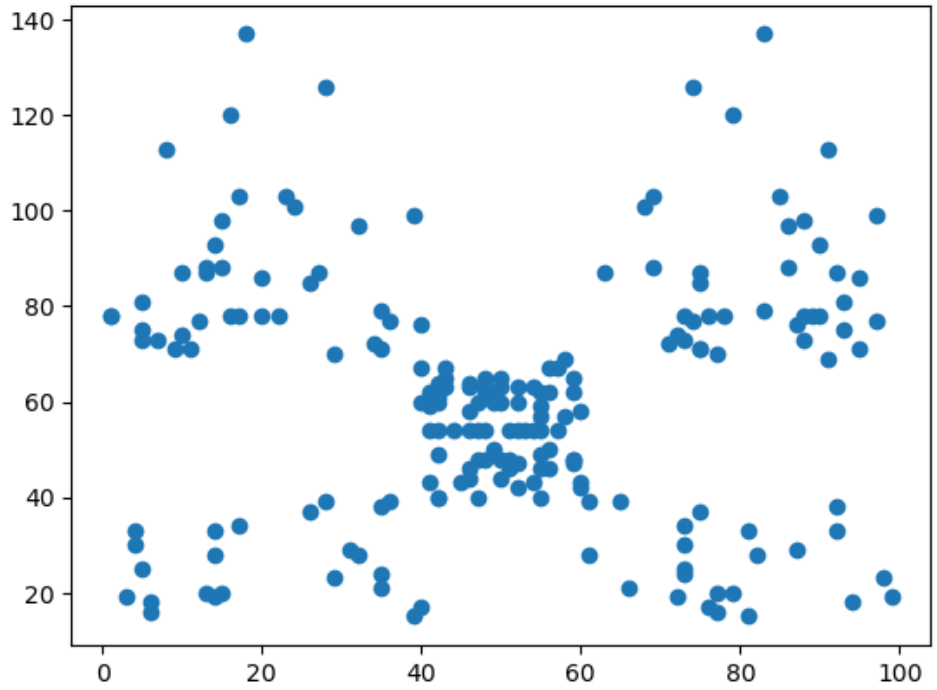
```
CustomerID                0
Genre                     0
Age                       0
Annual Income (k$)        0
Spending Score (1-100)    0
cluster                   0
dtype: int64
```

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

# Plot the data

```
plt.scatter(df["Spending Score (1-100)"],df["Annual Income (k$)"])
```

```
<matplotlib.collections.PathCollection at 0x157d9c110>
```



# K-means clustering

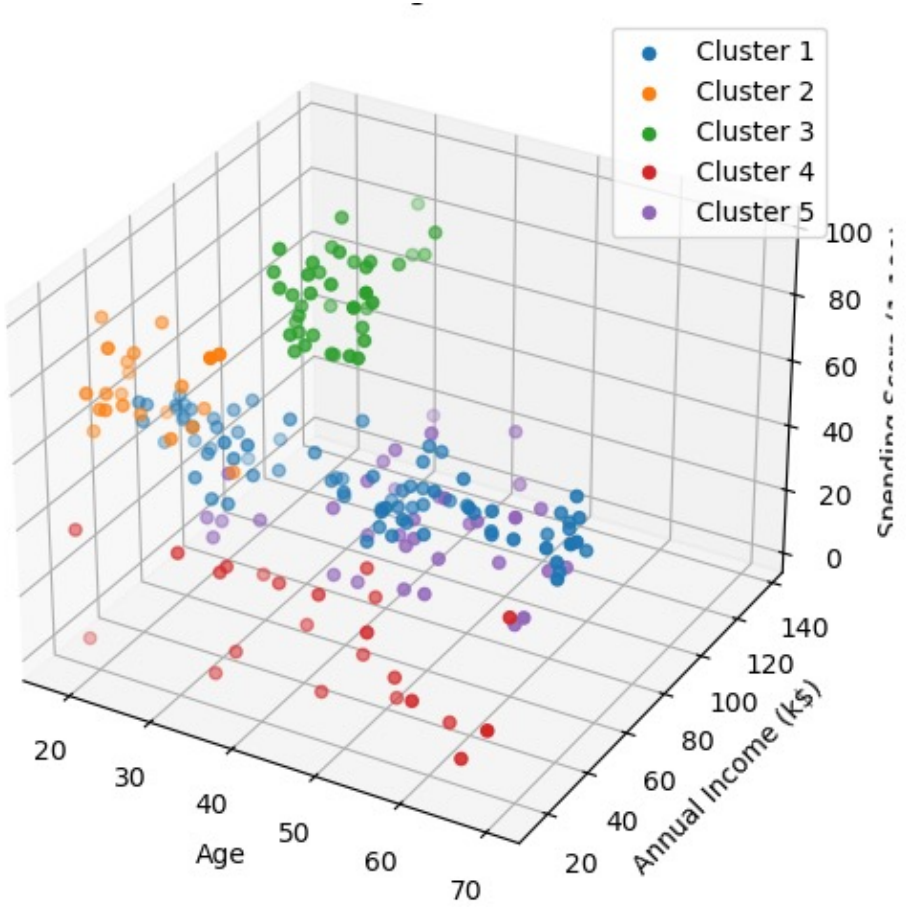
```
km=KMeans(n_clusters=5)
y_predicted=km.fit_predict(df[["Spending Score (1-100)","Annual Income (k$)"]])
```

```
df["cluster"]=y_predicted
df.head()
```

| | CustomerID | Genre | Age | Annual Income (k$) | Spending Score (1-100) | cluster |
|---|---|---|---|---|---|---|
| **0** | 1 | Male | 19 | 15 | 39 | 3 |
| **1** | 2 | Male | 21 | 15 | 81 | 1 |
| **2** | 3 | Female | 20 | 16 | 6 | 3 |
| **3** | 4 | Female | 23 | 16 | 77 | 1 |
| **4** | 5 | Female | 31 | 17 | 40 | 3 |

```
import seaborn as sns
plt.figure(figsize=(10,6))
sns.scatterplot(x = 'Annual Income (k$)',y = 'Spending Score (1-100)',hue="cluster",
                palette=['green','orange','brown','dodgerblue','red'], legend='full',data = df  ,s = 60 )
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.title('Spending Score (1-100) vs Annual Income (k$)')
plt.show()
```

# 3D


Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Age
20
30
40
50
60
70
Annual Income (k$)
20
40
60
80
100
120
140
0
20
40
60
80
100

# 3D

```python
from mpl_toolkits.mplot3d import Axes3D

# Plotting the 3D scatter plot
fig = plt.figure(figsize=(10, 6))
ax = fig.add_subplot(111, projection='3d')

# Scatter plot for each cluster
for i in range(5):
    ax.scatter(df[df['cluster'] == i]['Age'],
               df[df['cluster'] == i]['Annual Income (k$)'],
               df[df['cluster'] == i]['Spending Score (1-100)'],
               label=f'Cluster {i+1}')

ax.set_xlabel('Age')
ax.set_ylabel('Annual Income (k$)')
ax.set_zlabel('Spending Score (1-100)')
ax.set_title('Customer Segmentation')

plt.legend()
plt.show()
```

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

# Silhouette score

```
from sklearn.metrics import silhouette_score
scores = []
for k in range(2, 8):
    kmeans = KMeans(n_clusters=k, random_state=42)
    labels = kmeans.fit_predict(df[["Spending Score (1-100)","Annual Income (k$)"]])
    scores.append(silhouette_score(df[["Spending Score (1-100)","Annual Income (k$)"]], labels))
print(scores)
```

K=2, 0.2968969162503008,
K=3. 0.467613581587754 35,
K=4. 0.4931963109249047,
**K=5. 0.553931997444648,**
K=6. 0.53976103063432,
K=7. 0.5264283703685728]

# END