



Unsupervised Learning: Scanner Case study

Parcours Progis

Etudes, Medias, communication, Marketing

Bahareh Afshinpour

30.03.2026

Réduction dimensionnelle

La réduction de la dimension est un processus qui consiste à prendre des données dans un espace de grande dimension et à les remplacer par des données dans un espace de plus petite dimension

Dimension Reduction identifies the combination of attributes (principal components, or directions in the feature space) that account for the most variance in the data.

Réduction dimensionnelle

- We can say some of the features tell us the same information as other features.
- In this case we do not need all features. Maybe we need a little bit of one feature and a little bit of others.
- Or we can say that is a combination of these features that captures the information.
- Linear methods:
 - Principal Component Analysis (PCA)
 - Singular Value Decomposition (SVD)
- Non-Linear Methods
 - T-SNE

PCA (Principal Component Analysis)

- With PCA we figure out, if we have a bunch of features, how many component in the space do we need to explain the data.
- We are not deciding which feature to keep and drop, but we are **creating** new features that it is kind of a little bit of multiple features.
- Covariance matrix : when one variable changes how much does the other variable change
- Eigenvalue decomposition: what direction is the change
- The principal component are those with the largest eigenvalues

PCA is a process of figuring out the most important features, or principal components, that have the most impact on the target variable.

PCA

- Think that each pixel is one feature.
- Some of the pixels do not play any role at all in finding what digit it is.



- For example, these pixels, no matter what number we have, they are always black.



<https://www.youtube.com/watch?v=8klqIM9UvAc>

PCA

What if we eliminate features that aren't essential?

- Faster training and inference
- Data visualization gets simpler.

Visualization helps us. Regarding the final decision-making process.

PCA

Before using PCA, there are a few things to consider.

- Scale features before applying PCA.
- Accuracy might drop (imagine 100D to 2D; we lose lots of information).
- We should split our data if we have a label column.

PCA in Python

```
--  
import matplotlib.pyplot as plt  
from sklearn.decomposition import PCA  
  
pca = PCA(n_components=2)  
X_pca = pca.fit_transform(sessions_embedding)  
  
y=[0] * 61  
# Plot the result  
plt.figure(figsize=(8, 6))  
scatter = plt.scatter(X_pca[:, 0], X_pca[:, 1], c=y, cmap='viridis', edgecolor='k')  
plt.xlabel('Principal Component 1')  
plt.ylabel('Principal Component 2')  
plt.title('PCA - 2D Projection')  
plt.legend(*scatter.legend_elements(), title="Classes")  
plt.grid(True)  
plt.show()
```

Scanner



Scanner case study

- Logs
 - Are records of events that occurred during the running of a software system.
 - Are generated by log statements in software source code.
 - Logs are a primary source for problem diagnosis.
 - **Event** : Units of information in a log are often called events.

Index	Time	Session ID	Object	Action	Input	Output
51	1585070116817	client6	scan12	unlock	[],	0
52	1585070116819	client0	scan0	scan	[3270190022534],	0
53	1585070116820	client1	cashier1	CloseSession	[],	0
54	1585070116820	client2	cashier2	add	[3570590109324],	0
55	1585070116824	client5	scan5	scan	[8718309259938],	0
56	1585070116825	client6	scan12	scan	[3560070139675],	0
57	1585070116837	client0	scan0	scan	[3560070048786],	0
58	1585070117030	client6	scan12	scan	[7640164630021],	-2
59	1585070117073	client6	scan12	delete	[7640164630021],	-2
60	1585070116838	client1	cashier1	pay	[353.06],	0

event

Scanner case study

- *A scanner, or supermarket scanner, is an electronic device designed to detect barcodes of products and import them into a shopping list.*
- *Clients may scan the product's barcode to add it to the purchase list.*
- *A consumer may also remove an item from the purchase list ...*
 - **Self-scanning items by client**
 - **Actions:**
 - Scan, Delete barcodes and Pay, abandon
- Large files from clients' behavior.

Goal:

Clustering



Scanner case study

Client6 **session** from Unlock to Pay

Index	Time	Session ID	Object	Action	Input	Output
51	1585070116817	client6	scan12	unlock,	[],	0
52	1585070116819	client0	scan0	scan,	[3270190022534],	0
53	1585070116820	client1	cashier1	CloseSession,	[],	0
54	1585070116820	client2	cashier2	add,	[3570590109324],	0
55	1585070116824	client5	scan5	scan,	[8718309259938],	0
56	1585070116825	client6	scan12	scan,	[3560070139675],	0
57	1585070116837	client0	scan0	scan,	[3560070048786],	0
58	1585070117030	client6	scan12	scan,	[7640164630021],	-2
59	1585070117073	client6	scan12	delete,	[7640164630021],	-2
60	1585070116838	client1	cashier1	pay,	[353.06],	0
61	1585070116839	client2	cashier2	CloseSession,	[],	0
62	1585070116840	client3	cashier3	add,	[3570590109324],	0
64	1585070117687	client6	scan12	transmission,	[caisse6],	0
65	1585070117687	client6	scan12	abandon,	[],	?
66	1585070117701	client6	cashier4	OpenSession,	[],	0
67	1585070116855	client0	scan0	transmission,	[cashier0],	0
68	1585070116855	client0	scan0	abandon,	[],	?
69	1585070117716	client6	cashier4	add,	[7640164630021],	0
70	1585070117731	client6	cashier4	CloseSession,	[],	0
71	1585070117747	client6	cashier4	Pay,	[260],	9.11

A user session is a series of "user actions" performed by an individual on the website or web application or ...

Log file (test-suites)

Testsuite	Number of clients	Number of events
1026-event	61	1026

We are going to cluster these 61 clients
We treat each client as a sentence (session)

Log partition

- It involves **dividing** a log file or a sequence of log records into separate, manageable partitions based on certain criteria.
 - For example, in a system that collects event logs, log partitioning might split the logs by day or hour, so each partition contains log entries for a specific time period.

Index	Time	Session ID	Object	Action	Input	Output
51	1585070116817	client6	scan12	unlock	[]	0
52	1585070116819	client0	scan0	scan	[3270190022534]	0
53	1585070116820	client1	cashier1	CloseSession	[]	0
54	1585070116820	client2	cashier2	add	[3570590109324]	0
55	1585070116824	client5	scan5	scan	[8718309259938]	0
56	1585070116825	client6	scan12	scan	[3560070139675]	0
57	1585070116837	client0	scan0	scan	[3560070048786]	0
58	1585070117030	client6	scan12	scan	[7640164630021]	-2
59	1585070117073	client6	scan12	delete	[7640164630021]	-2
60	1585070116838	client1	cashier1	pay	[353.06]	0
61	1585070116839	client2	cashier2	CloseSession	[]	0
62	1585070116840	client3	cashier3	add	[3570590109324]	0
64	1585070117687	client6	scan12	transmission	[caisse6]	0
65	1585070117687	client6	scan12	abandon	[]	?
66	1585070117701	client6	cashier4	OpenSession	[]	0
67	1585070116855	client0	scan0	transmission	[cashier0]	0
68	1585070116855	client0	scan0	abandon	[]	?
69	1585070117716	client6	cashier4	add	[7640164630021]	0
70	1585070117731	client6	cashier4	CloseSession	[]	0
71	1585070117747	client6	cashier4	Pay	[260]	9.11

ID=client60

ID=client10

ID=client2

```

Index Time Session ID Object Action Input Output
51, 1585070116817, client6, scan12, unlock, [ ], 0
52, 1585070116819, client0, scan0, scan, [3270190022534], 0
53, 1585070116820, client1, cashier1, CloseSession, [ ], 0
54, 1585070116820, client2, cashier2, add, [3570590109324], 0
55, 1585070116824, client5, scan5, scan, [8718309259938], 0
56, 1585070116825, client6, scan12, scan, [3560070139675], 0
57, 1585070116837, client0, scan0, scan, [3560070048786], 0
58, 1585070117030, client6, scan12, scan, [7640164630021], -2
59, 1585070117073, client6, scan12, delete, [7640164630021], -2
60, 1585070116838, client1, cashier1, pay, [353.06], 0
61, 1585070116839, client2, cashier2, CloseSession, [ ], 0
62, 1585070116840, client3, cashier3, add, [3570590109324], 0
64, 1585070117687, client6, scan12, transmission, [caisse6], 0
65, 1585070117687, client6, scan12, abandon, [ ], ?
66, 1585070117701, client6, cashier4, OpenSession, [ ], 0
67, 1585070116855, client0, scan0, transmission, [cashier0], 0
68, 1585070116855, client0, scan0, abandon, [ ], ?
69, 1585070117716, client6, cashier4, add, [7640164630021], 0
70, 1585070117731, client6, cashier4, CloseSession, [ ], 0
71, 1585070117747, client6, cashier4, Pay, [260], 9.11
    
```

....

Event embedding

ID=client6

Index	Time	Session ID	Object	Action	Input	Output
51,	1585070116817,	client6,	scan12,	unlock,	[],	0
56,	1585070116825,	client6,	scan12,	scan,	[3560070139675],	0
58,	1585070117030,	client6,	scan12,	scan,	[7640164630021],	-2

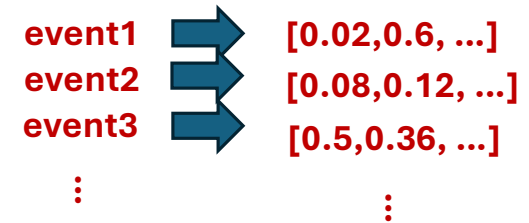
event1 → [0.02,0.6, ...]
event2 → [0.08,0.12, ...]
event3 → [0.5,0.36, ...]
⋮

- Word2vec:
- Is a technique in [natural language processing](#) for obtaining [vector](#) representations of words. These vectors capture information about the meaning of the word based on the surrounding words

Event embedding

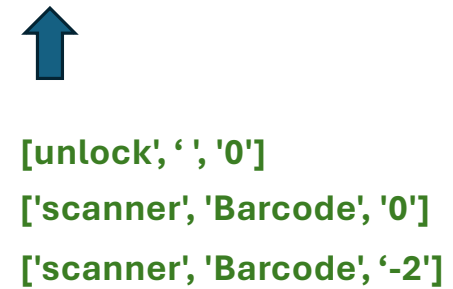
ID=client6

Index	Time	Session ID	Object	Action	Input	Output
51	1585070116817	client6	scan12	unlock	[],	0
56	1585070116825	client6	scan12	scan	[3560070139675],	0
58	1585070117030	client6	scan12	scan	[7640164630021],	-2

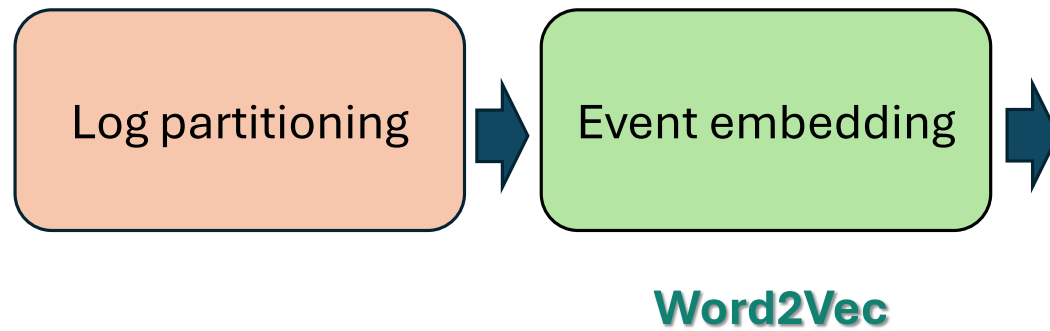


- It is better to code each event.

Index	Time	Session ID	Object	Action	Input	Output
51	1585070116817	client6	scan12	unlock	[],	0
56	1585070116825	client6	scan12	scan	[3560070139675],	0
58	1585070117030	client6	scan12	scan	[7640164630021],	-2



The propose model



Word2Vec on all the Sessions

Event embedding
(word embedding)

The Word2Vec package from the Gensim library

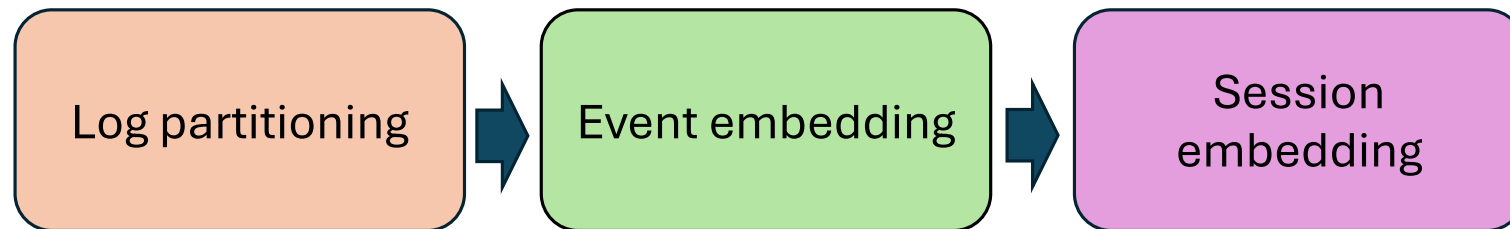
The dimension of the W2V vectors are equal to the number of vocabs

word

```
['debloquer', 'Nothing', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'Barcode', '0'], ['scanner',  
'Barcode', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'Barcode', '0'],  
['scanner', 'Barcode', '0'], ['scanner', 'ErrorBarcode', '0'], ['scanner', 'Barcode', '0'],  
['scanner', 'Barcode', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'Barcode', '0'], ['scanner',  
'Barcode', '0'], ['transmission', 'CaisseNumber', '0'], ['abandon', 'Nothing', 'Error'], ['payer',  
'Price-float', '0']
```

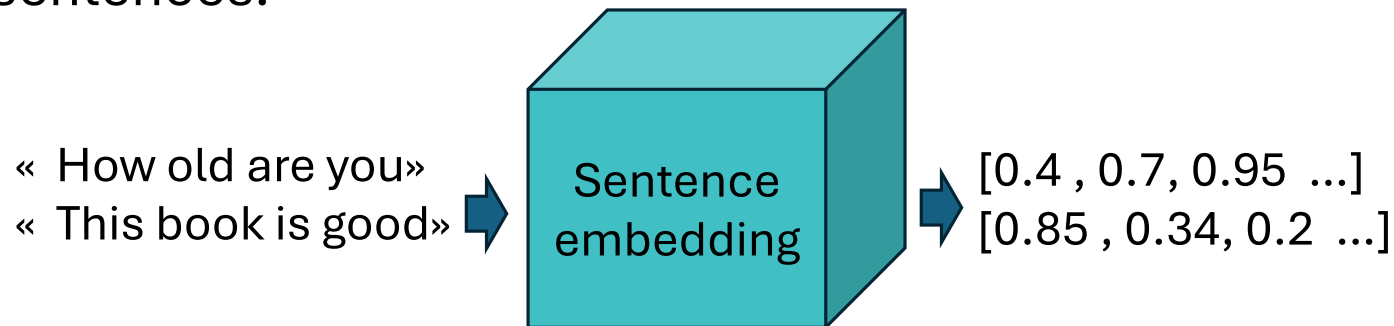
session

The propose model



Sentence embeddings

- The idea behind sentence embeddings is to convert sentences into numerical vectors.
- These numerical vectors represent the sentences in a structured and machine-friendly format.
- The beauty of this vector representation lies in its ability to capture semantic relationships and contextual similarities between sentences.



Comparing two clients (Sessions)

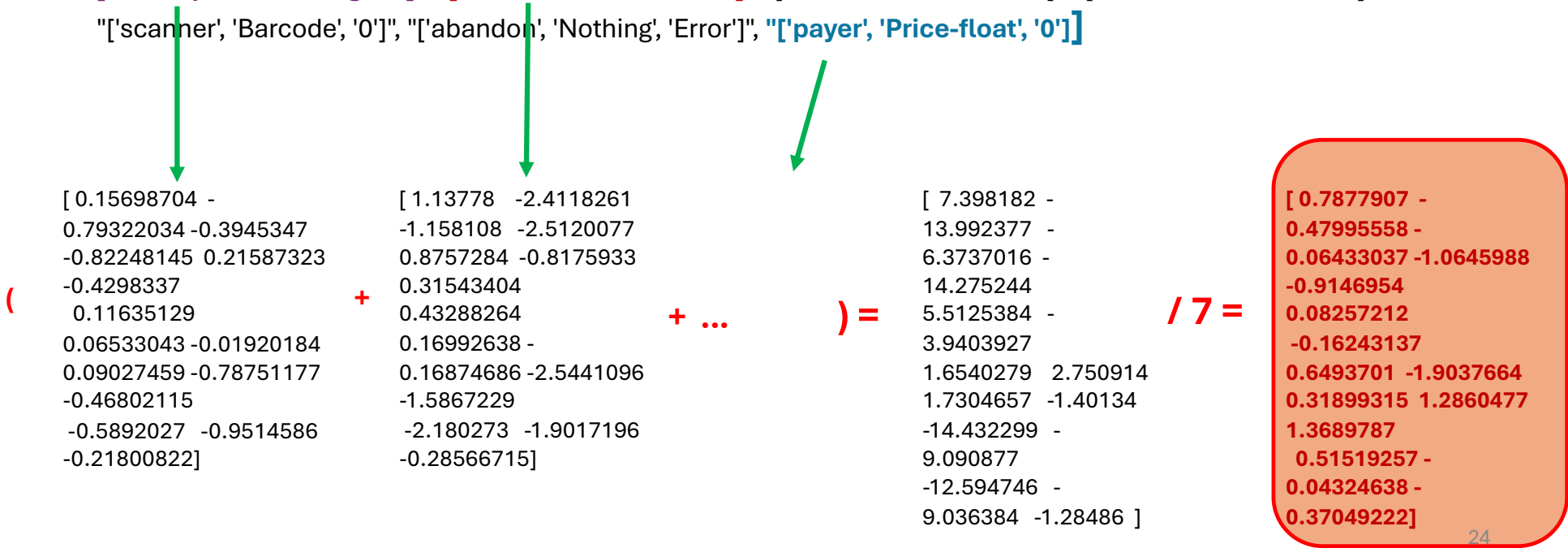
- Need a *measure* for each session ...
 - To compare two sessions and find their similarity
- *Measure*:
 - An average of the Word2Vec vectors in each session

Averaging a session

Session embedding

- client60:

['debloquer', 'Nothing', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'Barcode', '0'],
 ['scanner', 'Barcode', '0'], ['abandon', 'Nothing', 'Error'], ['payer', 'Price-float', '0']



Averaging a session

client60:



```
[ 1.0375887 -0.24739444
-0.92388165 -1.7032957
-0.2800482 0.19364282
0.5891389 -0.38360417
-0.02879436 0.03572838
-1.3417056 0.9103135
-0.83895904 1.4372357
0.19943428]
```

client40:



```
[ 1.0956031 -0.26459935
-0.99176484 -1.8269157
0.2858453 0.22351813
0.62547946 -0.40456355
-0.01479345 0.02950979
1.4409974 0.97218895
-0.90637374 1.5324388
0.19554672]
```

client17:



```
[ 1.1439486 -0.2789368
-1.0483342 -1.9299325
0.29067624 0.24841422
0.6557632 -0.4220297
0.00312602 0.02432763
1.5237406
1.0237519
-0.9625526 1.6117748
0.19230707]
```

Now we have a measure for each session (client).

Session embedding

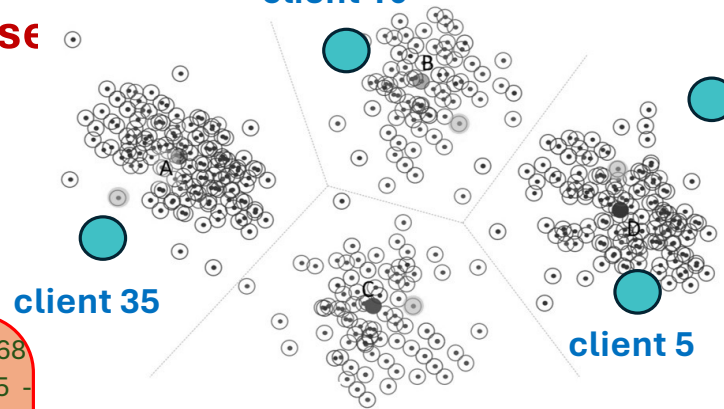
We are going to cluster these 61 clients

```
[ 1.0956031 -0.26459935
-0.99176484 -1.8269157 -
0.2858453 0.22351813
0.62547946 -0.40456355
-0.01479345 0.02950979 -
1.4409974 0.97218895
-0.90637374 1.5324388
0.19554672]
```

client 10

```
[ 1.0375887 -0.24739444 -
0.92388165 -1.7032957 -
0.2800482 0.19364282
0.5891389 -0.38360417 -
0.02879436 0.03572838 -
1.3417056 0.9103135
-0.83895904 1.4372357
0.19943428]
```

client 60



client 35

client 5

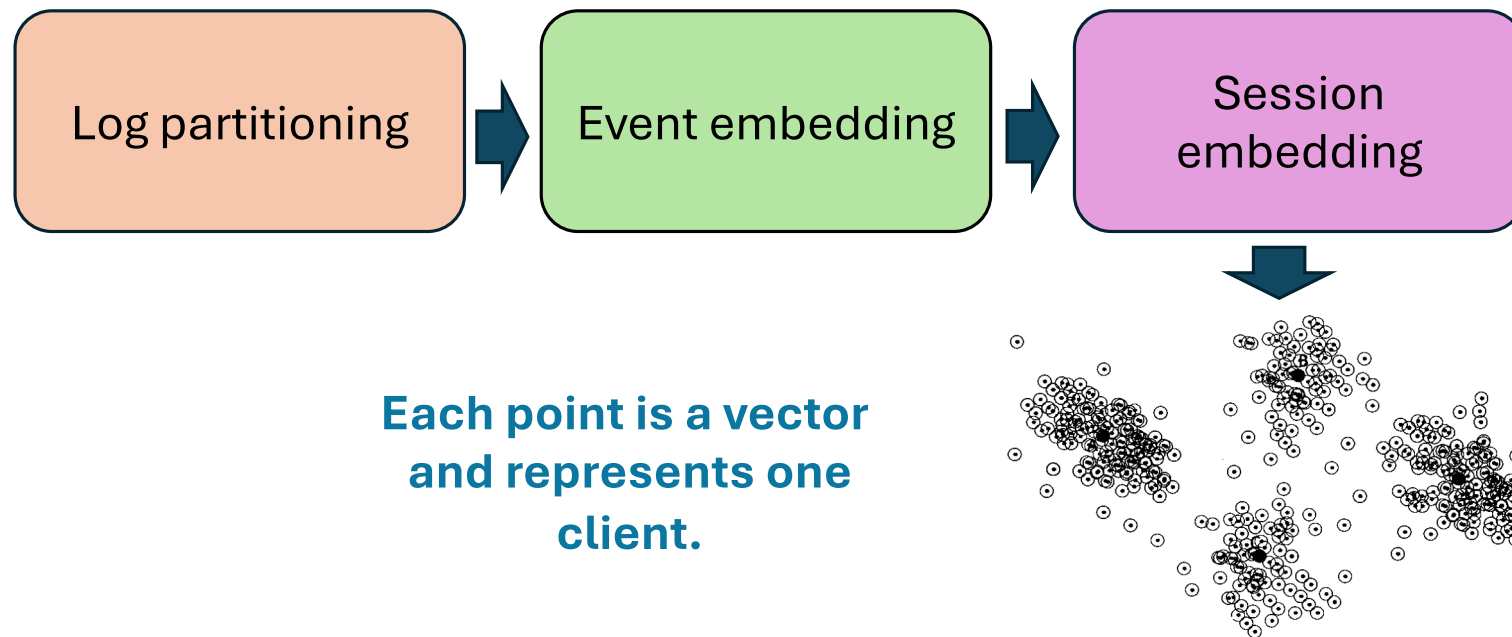
D=15

```
[ 1.1439486 -0.2789368
-1.0483342 -1.9299325 -
0.29067624 0.24841422
0.6557632 -0.4220297 -
0.00312602 0.02432763 -
1.5237406
1.0237519
-0.9625526 1.6117748
0.19230707]
```

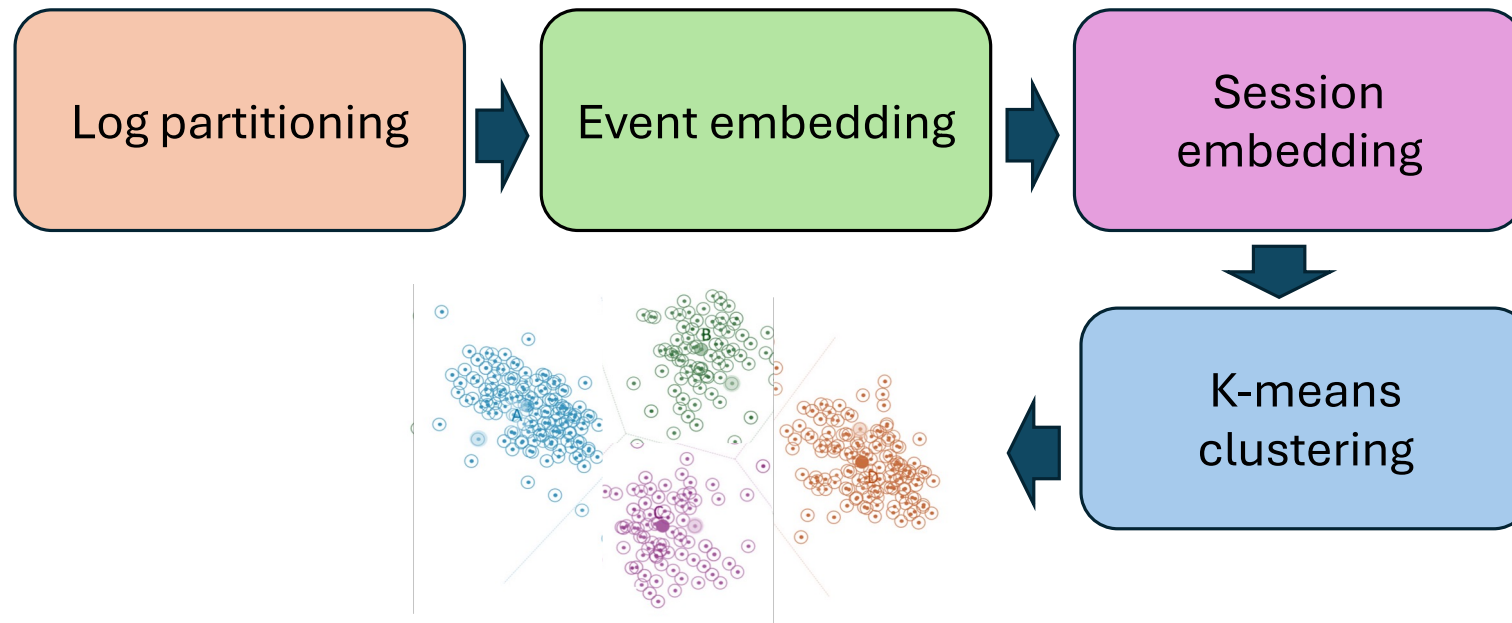
D=15

```
[ 1.1439486 -0.2789368
-1.0483342 -1.9299325 -
0.29067624 0.24841422
0.6557632 -0.4220297 -
0.00312602 0.02432763 -
1.5237406
1.0237519
-0.9625526 1.6117748
0.19230707]
```

The propose model

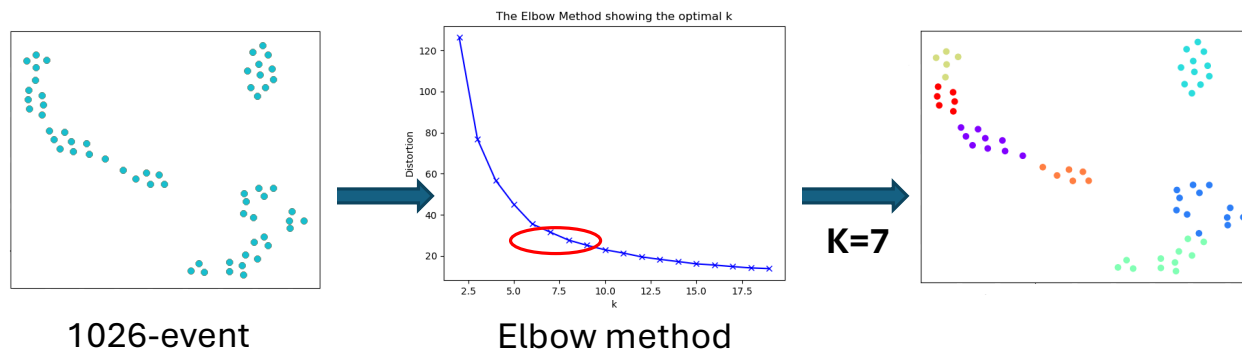


The propose model



Clustering

- K-Means & Elbow method
 - K-means needs to know number of cluster K
 - Elbow method finds the optimal K
- Elbow tries different K and calculates *distortion*[3]



TP

Range() function

- The `range()` function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and ends at a specified number.

Loop

```
for i in range(6):  
    print(i)
```

→ {0,1,2,3,4,5}

0
1
2
3
4
5

Len() function

- The len() function in Python is used to **get the number of items:**

```
text = "hello"  
print(len(text)) # Output: 5
```

```
numbers = [1, 2, 3, 4]  
print(len(numbers)) # Output: 4
```


Dictionary

- is a built-in data structure that **stores data in key-value pairs**.

```
person = {  
    "name": "Alice",  
    "age": 30,  
    "city": "Paris"  
}
```

keys values

Here, "**person**" refers to the name of the dictionary.

Dictionary

- Key
- value

```
thisdict = {  
    "brand": "Ford",  
    "model": "Mustang",  
    "year": 1964  
}
```

Dictionaries are used to store data values in **key:value** pairs.

A dictionary is a collection which is changeable and do not allow duplicates.

```
{ ' client0': [['debloquer', 'Nothing', '0'], ['scanner', 'Barcode', '0  
code', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'Barcode', '0'],  
arcod', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'Barcode', '0'  
' client1': [['debloquer', 'Nothing', '0'], ['scanner', 'Barcode', '0'  
ode', '0'], ['transmission', 'CaisseNumber', '0'], ['abandon', 'Nothin  
anner', 'Barcode', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'Bar  
r', '0'], ['abandon', 'Nothing', 'Error'], ['ouvrirSession', 'Nothing'  
lient3': [['debloquer', 'Nothing', '0'], ['scanner', 'Barcode', '0'],
```

Iterate over dictionary keys: keys()

```
for sent in Sessions:  
    print(sent)
```

```
client0  
client1  
client2  
client3  
client4  
client5  
client6  
client7  
client8  
client9  
client10  
client11  
client12  
client13  
client14  
client15  
client16  
client17  
client18  
client19
```

Iterate over dictionary values: values()

```
for sent in Sessions.values():  
    print(sent)
```

```
[['debloquer', 'Nothing', '0'], ['scanner', 'Barcode', '0'],  
 ['scanner', 'Barcode', '0'], ['scanner', 'Barcode', '0'], ['s  
'0'], ['scanner', 'Barcode', '0'], ['scanner', 'Barcode', '0']  
 [['debloquer', 'Nothing', '0'], ['scanner', 'Barcode', '0'],  
 ['transmission', 'CaisseNumber', '0'], ['abandon', 'Nothing',  
 [['debloquer', 'Nothing', '0'], ['scanner', 'Barcode', '0'],  
 '0'], ['transmission', 'CaisseNumber', '0'], ['abandon', 'Not  
 ['payer', 'Price-float', '0']]
```

Iterate over dictionary key-value pairs: items()

```
for c,sent in Sessions.items():  
    print(c,sent)
```

```
client0 [['debloquer', 'Nothing', '0'], ['scanner',  
e', '0'], ['scanner', 'Barcode', '0'], ['scanner',  
ode', '0'], ['scanner', 'Barcode', '0'], ['scanner',  
client1 [['debloquer', 'Nothing', '0'], ['scanner',  
e', '0'], ['transmission', 'CaisseNumber', '0'], [  
client2 [['debloquer', 'Nothing', '0'], ['scanner',  
arcode', '0'], ['transmission', 'CaisseNumber', '0'],  
g', '0'], ['payer', 'Price-float', '0']]  
client3 [['debloquer', 'Nothing', '0'], ['scanner',  
e', '0'], ['scanner', 'Barcode', '0'], ['scanner',  
ber']]
```

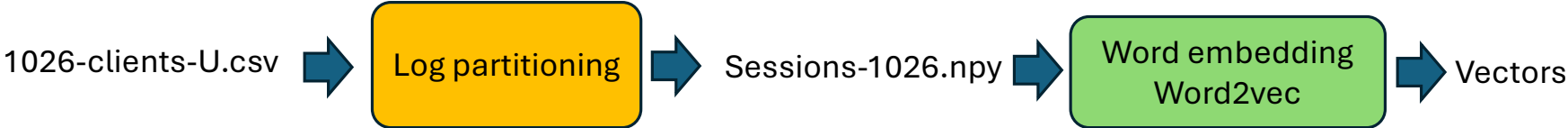
Change event to string

```
: Sentences=[]
all_clients_string=[]
for c,sent in Sessions.items():
    Sentences.append(sent)
    SentencesString=[]
    for i in sent:
        SentencesString.append(str(i))
    all_clients_string.append(SentencesString)
print(all_clients_string)
```

[['debloquer', 'Nothing', '0'], ['scanner', 'Barcode', '0']]

["['debloquer', 'Nothing', '0']", "['scanner', 'Barcode', '0']"]

Log partitioning based on ClientTID



Index	Time	Session ID	Object	Action	Input	Output
51	1585070116817	client6	scan12	unlock	[],	0
52	1585070116819	client0	scan0	scan	[3270190022534],	0
53	1585070116820	client1	cashier1	CloseSession	[],	0
54	1585070116820	client2	cashier2	add	[3570590109324],	0
55	1585070116824	client5	scan5	scan	[8718309259938],	0
56	1585070116825	client6	scan12	scan	[3560070139675],	0
57	1585070116837	client0	scan0	scan	[3560070048786],	0
58	1585070117030	client6	scan12	scan	[7640164630021],	-2
59	1585070117073	client6	scan12	delete	[7640164630021],	-2
60	1585070116838	client1	cashier1	pay	[353.06],	0

```

    {' client0': [['debloquer', 'Nothing', '0'], ['scanner', 'Barcode', '0',
    code', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'Barcode', '0'],
    arcode', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'Barcode', '0'
    ' client1': [['debloquer', 'Nothing', '0'], ['scanner', 'Barcode', '0'
    ode', '0'], ['transmission', 'CaisseNumber', '0'], ['abandon', 'Nothin
    anner', 'Barcode', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'Bar
    r', '0'], ['abandon', 'Nothing', 'Error'], ['ouvrirSession', 'Nothing'
    lient3': [['debloquer', 'Nothing', '0'], ['scanner', 'Barcode', '0'],
  
```

Client0: [0.5,0.36, ...]
Client1: [0.07,0.41, ...]

Word2Vec

```
import multiprocessing
from gensim.models import Word2Vec
cores = multiprocessing.cpu_count()
w2v_model = Word2Vec(min_count=1,window=3,vector_size=15,sample=0,alpha=0.03,min_alpha=0.0007,negative=2,workers=cores-1)
w2v_model.build_vocab(all_clients_string)
w2v_model.train(all_clients_string, total_examples=w2v_model.corpus_count, epochs=10, report_delay=1)
```

```
index2word_set = set(w2v_model.wv.index_to_key) # Our Vocab
print("the word that we have are: ",index2word_set)
```

The word (different event) that we have are: 15

```
{['supprimer', 'Barcode', '0'], ['payer', 'Price-float', '0'], ['ajouter', 'Barcode', '0'], ['payer', 'Price-integer', 'Float Number'], ['scanner', 'ErrorBarcode', '0'], ['scanner', 'Barcode', '-2'], ['fermerSession', 'Nothing', '0'], ['abandon', 'Nothing', 'Error'], ['debloquer', 'Nothing', '0'], ['scanner', 'Barcode', '0'], ['ouvrirSession', 'Nothing', '0'], ['transmission', 'CaisseNumber', '0'], ['transmission', 'CaisseNumber', 'Integer Number'], ['supprimer', 'ErrorBarcode', '0'], ['supprimer', 'Barcode', '-2']}
```

Vector of each event

For example, what is the vector that represents **event: [payer, price-float, 0]**?

```
w2v_model.wv["['payer', 'Price-float', '0']"]
```

```
array([-0.208438 , -0.17140363,  0.58019   , -0.8334066 , -0.4235734 ,  
       0.43202496,  0.16815467,  0.51377785, -0.1497903 ,  0.54200566,  
       0.1650343 ,  0.39274028,  0.00320614, -0.7347753 ,  0.13456789],  
      dtype=float32)
```


A vector concerning the client0

Session
embedding

```

print(all_clients_string[0]) ← List of Actions of client0=session of client0

feature_vec = np.zeros((15, ), dtype='float32')
number_words = 0
for word in all_clients_string[0]:
    number_words = number_words+1 ← Counting the number of actions that client0 did
    feature_vec = np.add(feature_vec, w2v_model.wv[word])

feature_vec = np.divide(feature_vec, number_words)
print(feature_vec)

[["['debloquer', 'Nothing', '0']", ["['scanner', 'Barcode', '0']", ["['scanner', 'Barcode', '0']", ["['scanner', 'Barcode', '0']", ["['scanner', 'Barcode', '0']", ["['scanner', 'Barcode', '0']", ["['scanner', 'Barcode', '0']", ["['scanner', 'Barcode', '0']", ["['scanner', 'Barcode', '0']", ["['scanner', 'Barcode', '0']", ["['scanner', 'Barcode', '0']", ["['scanner', 'Barcode', '0']", ["['scanner', 'Barcode', '0']", ["['scanner', 'Barcode', '0']"], ["['payer', 'Price-float', '0']"]]
[ 0.20394726  0.1303705   0.4773702   2.5052588  -0.05649199 -1.7890764
  0.95154786  0.04148404 -0.29469082 -1.2973394   0.07733324 -0.4541708
 -1.4051955   1.7533631  -0.36733612]

```

Range() function

- The `range()` function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and ends at a specified number.

Loop

```
for i in range(6):  
    print(i)
```

→ {0,1,2,3,4,5}

0
1
2
3
4
5

61 vectors concerning all clients

```
sessions_embedding=[]
for i in range(61):
    feature_vec = np.zeros((15, ), dtype='float32')
    number_words = 0
    for word in all_clients_string[i]:
        number_words = number_words+1
        feature_vec = np.add(feature_vec, w2v_model.wv[word])

    feature_vec = np.divide(feature_vec, number_words) #vector for one client
    print(i,":",feature_vec)
    sessions_embedding.append(feature_vec)
print(sessions_embedding)
```

61 vectors concerning all clients

```
sessions_embedding=[]
for i in range(61):
    feature_vec = np.zeros((15, ), dtype='float32')
    number_words = 0
    for word in all_clients_string[i]:
        number_words = number_words+1
        feature_vec = np.add(feature_vec, w2v_model.wv[word])

    feature_vec = np.divide(feature_vec, number_words) #vector for one client
    print(i,":",feature_vec)
    sessions_embedding.append(feature_vec)
print(sessions_embedding)
```

By using a loop (for), we can calculate the average for all 61 clients

61 vectors concerning all clients

```
sessions_embedding=[] To keep all the vectors
for i in range(61):
    feature_vec = np.zeros((15, ), dtype='float32')
    number_words = 0
    for word in all_clients_string[i]:
        number_words = number_words+1
        feature_vec = np.add(feature_vec, w2v_model.wv[word])

    feature_vec = np.divide(feature_vec, number_words) #vector for one client
    print(i,":",feature_vec)
    sessions_embedding.append(feature_vec) .append can add new vector
print(sessions_embedding)
```

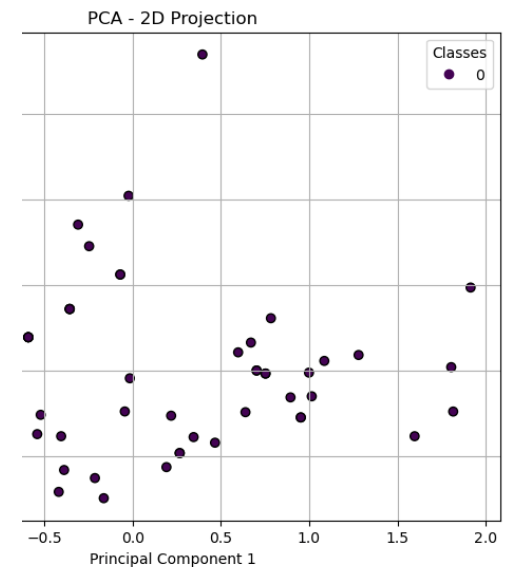
Visualisation de l'espace vectoriel (PCA)

15D → 2D

```
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA

pca = PCA(n_components=2)
X_pca = pca.fit_transform(sessions_embedding)

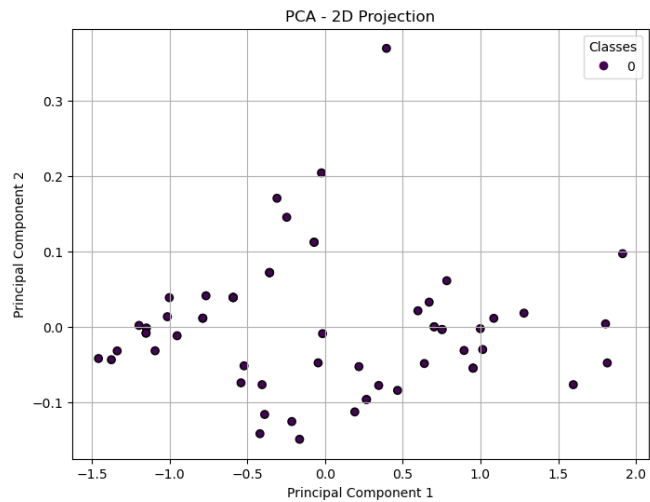
y=[0] * 61
# Plot the result
plt.figure(figsize=(8, 6))
scatter = plt.scatter(X_pca[:, 0], X_pca[:, 1], c=y, cmap='viridis', edgecolor='k')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('PCA - 2D Projection')
plt.legend(*scatter.legend_elements(), title="Classes")
plt.grid(True)
plt.show()
```



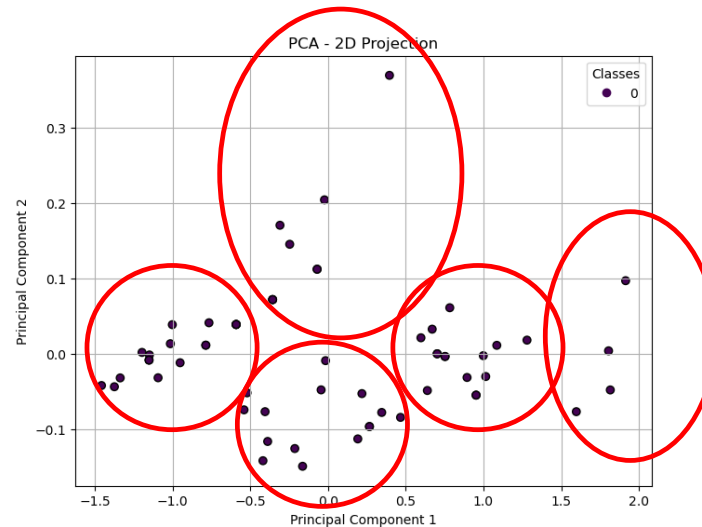
Clustering

K-means clustering

- K-Means & Elbow method
 - K-means needs to know number of cluster K
 - Elbow method finds the optimal K

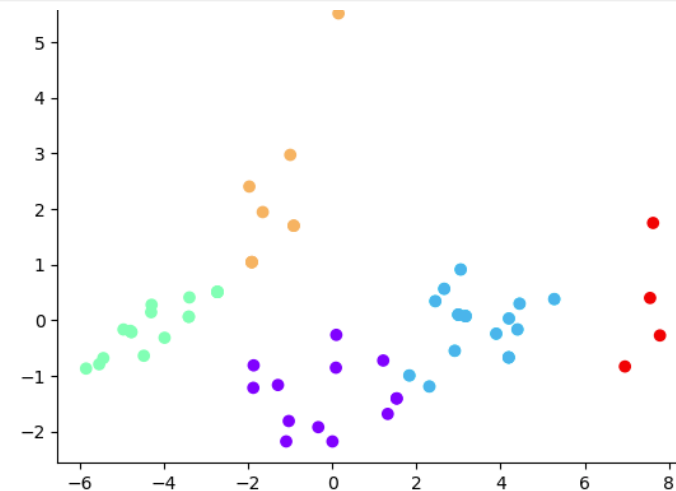


K=5

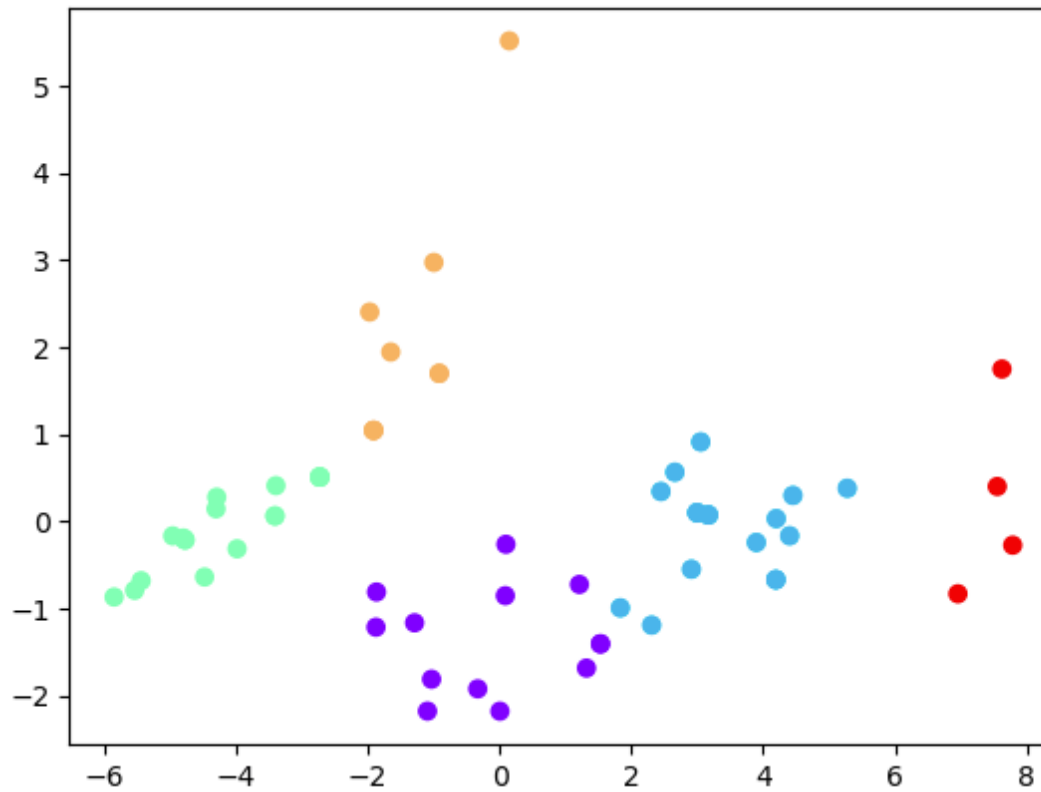


Clustering

```
from sklearn.cluster import KMeans  
kmeans = KMeans(n_clusters=5)  
kmeans.fit(X_pca)  
plt.scatter(X_pca[:,0],X_pca[:,1], c=kmeans.labels_, cmap='rainbow')  
plt.show()
```



Clustering



Clustering

```
clusters=dict()
for l in range(5):
    vocab_elem=[]
    OneCluster = np.where(kmeans.labels_ == l)
    print("Cluster ",l,":")
    for j in OneCluster[0]:
        print("client",j,":")
        print(all_clients_string[j])
        vocab_elem.append(j)
    clusters[l]=vocab_elem
print("*****")
```

Cluster 0 :

client 4 :

```
["['debloquer', 'Nothing', '0']", "['scanner', 'Barcode'
r', 'Barcode', '0']", "['scanner', 'Barcode', '0']", "['
de', '0']", "['scanner', 'Barcode', '0']", "['scanner',
['scanner', 'Barcode', '0']", "['scanner', 'Barcode', '0
['abandon', 'Nothing', 'Error']", "['ouvrirSession', 'No
'0']"]
```

client 5 :

```
["['debloquer', 'Nothing', '0']", "['scanner', 'Barcode'
canner', 'Barcode', '0']", "['scanner', 'Barcode', '0']"
arcod', '0']", "['scanner', 'Barcode', '-2']", "['scann
'0']", "['scanner', 'Barcode', '0']", "['transmission',
code', '0']", "['ajouter', 'Barcode', '0']", "['fermerSe
```

client 11 :

```
["['debloquer', 'Nothing', '0']", "['scanner', 'Barcode'
er', 'Barcode', '0']", "['scanner', 'Barcode', '0']", "[
```

End

bahareh.afshinpour@univ-grenoble-alpes.fr