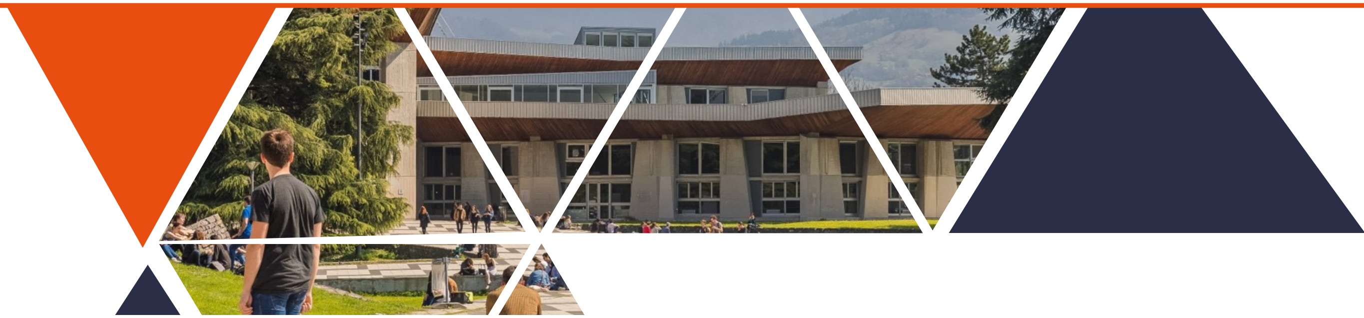
Unsupervised Learning: Word embedding



Parcours Progis

Etudes, Medias, communication, Marketing

Bahareh Afshinpour

19.05.2025

Université Grenoble Alpes

Sciences Po Grenoble UGA

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

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

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
0	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

Book

It is not the most efficient use of space

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

Bag-of-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

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.

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

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

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

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

Challenges

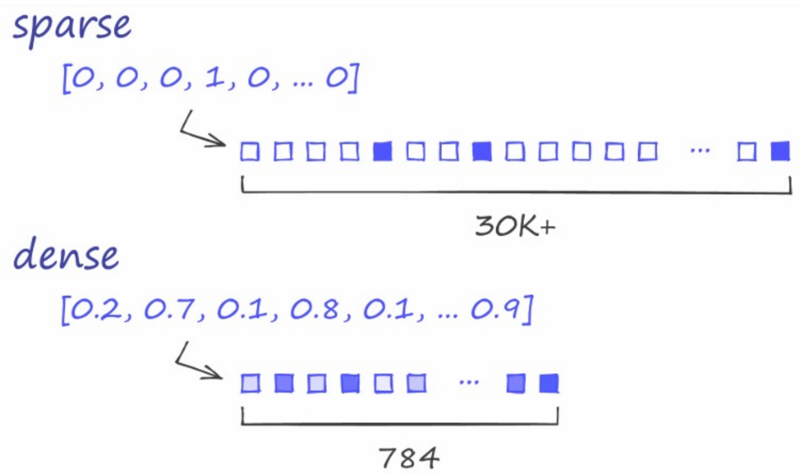
- They have some serious shortcomings:
  - They can not deal with words that they did not see in the training examples
  - They embeding that they produce are very sparse (sparse vector)

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.

What is a dense vector

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

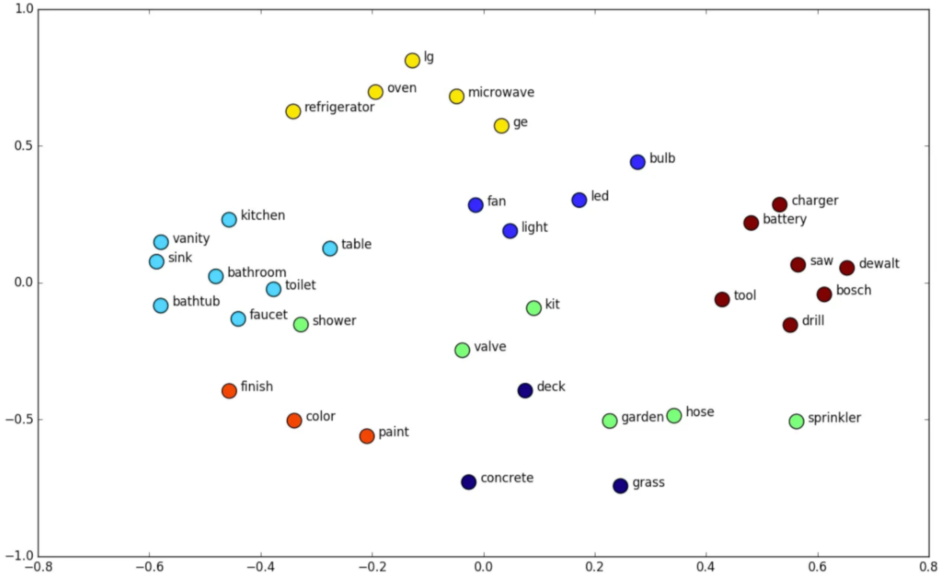


https://www.pinecone.io/learn/series/nlp/dense-vector-embeddings-nlp/

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

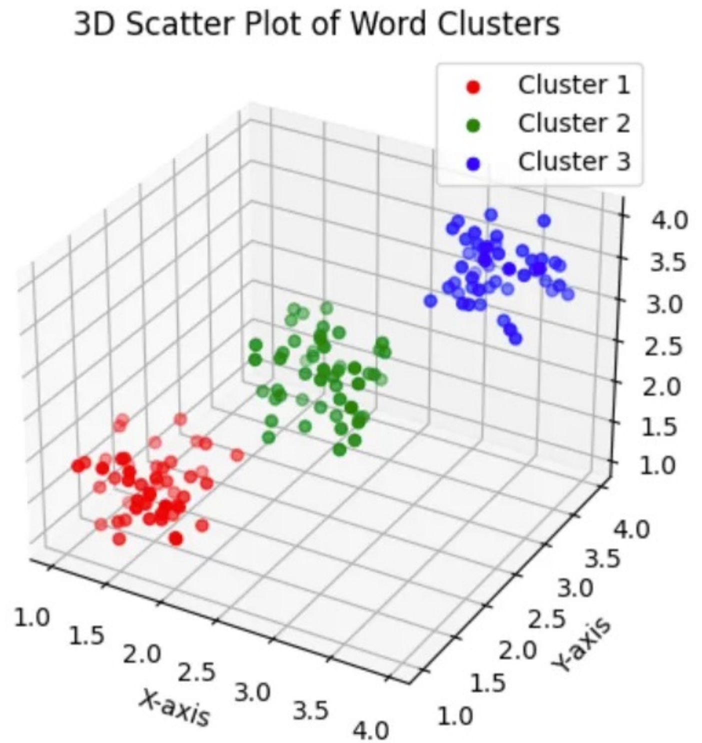
Embedding space

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



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

D=3



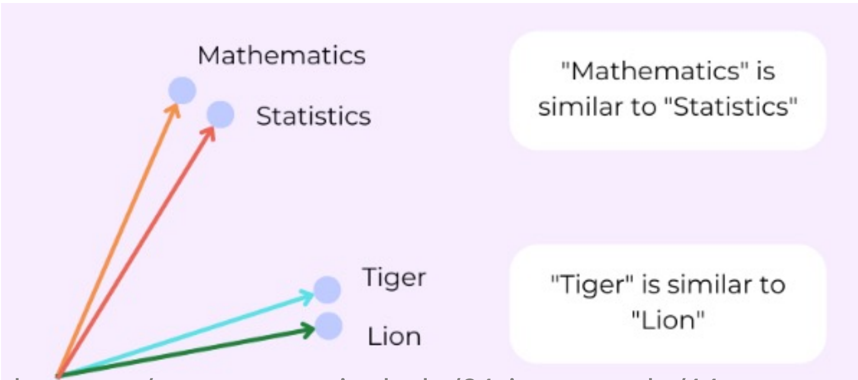
We can not visualize the embedding in 20 dimension space, but we can calculate the similarity

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

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

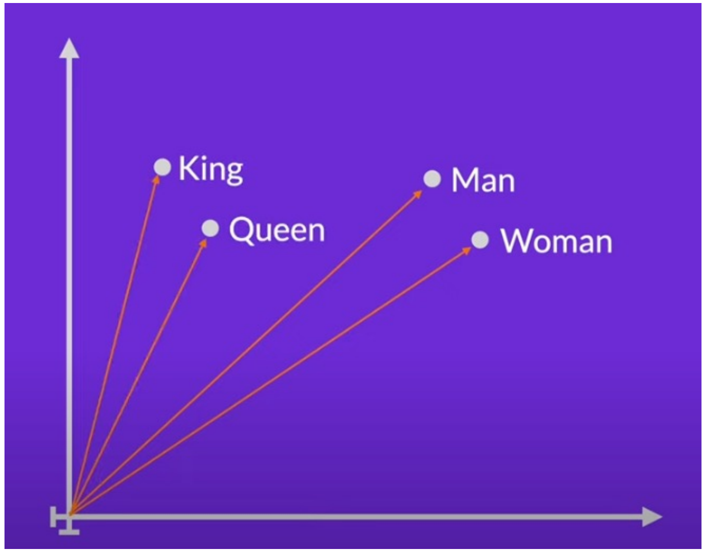
What are similar words?

- 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.



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.

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.

Méthodes classiques de word embedding

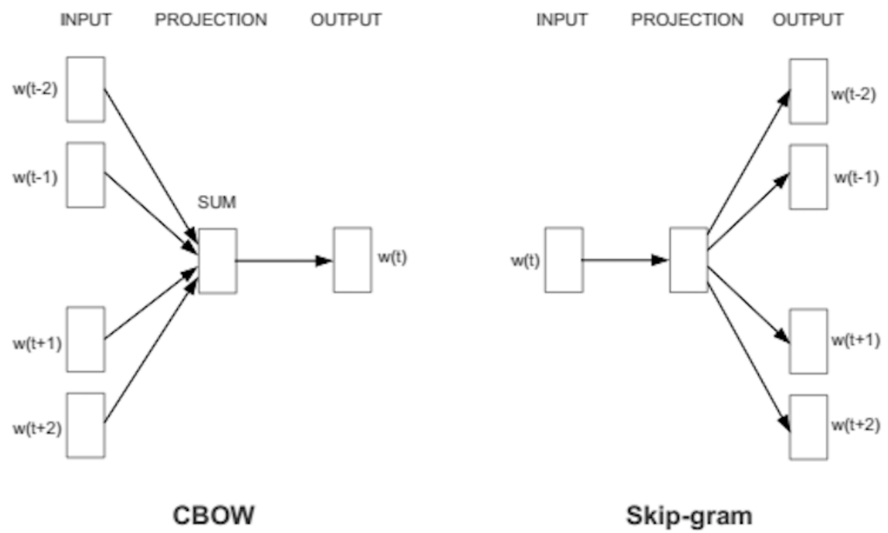
Méthode	Description rapide
Word2Vec (CBOW & Skip-gram)	Prédit un mot à partir du contexte ou l'inverse
GloVe	Utilise des co-occurrences globales
FastText	Utilise les sous-mots (caractères) → gère mieux les mots rares

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

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.

CBOW : Continuous Bag Of Words

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

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

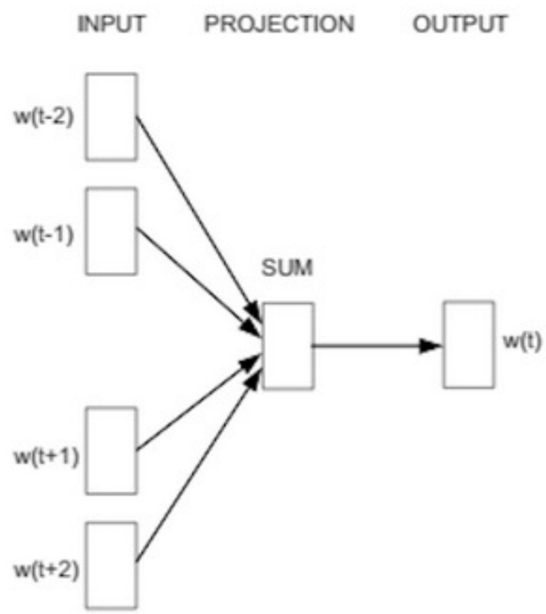
Example:

King ordered his

context

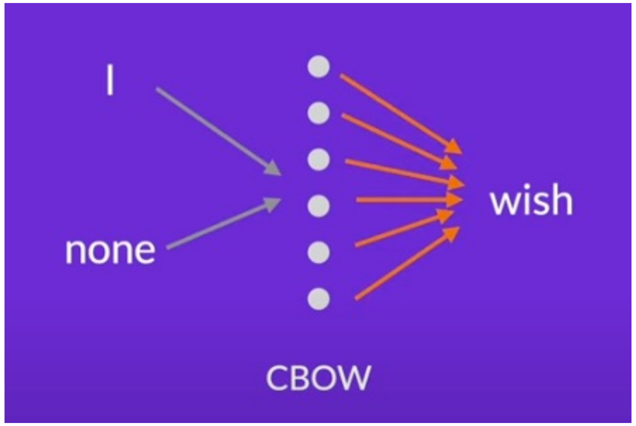
target

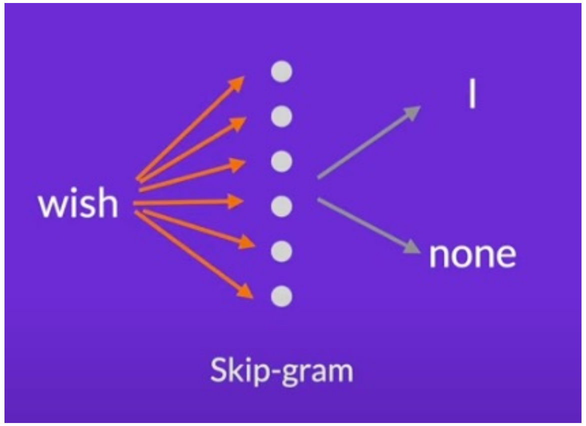
Here, we took a window size 3. it could be window of size 4 or five depends on how you want to experiment.



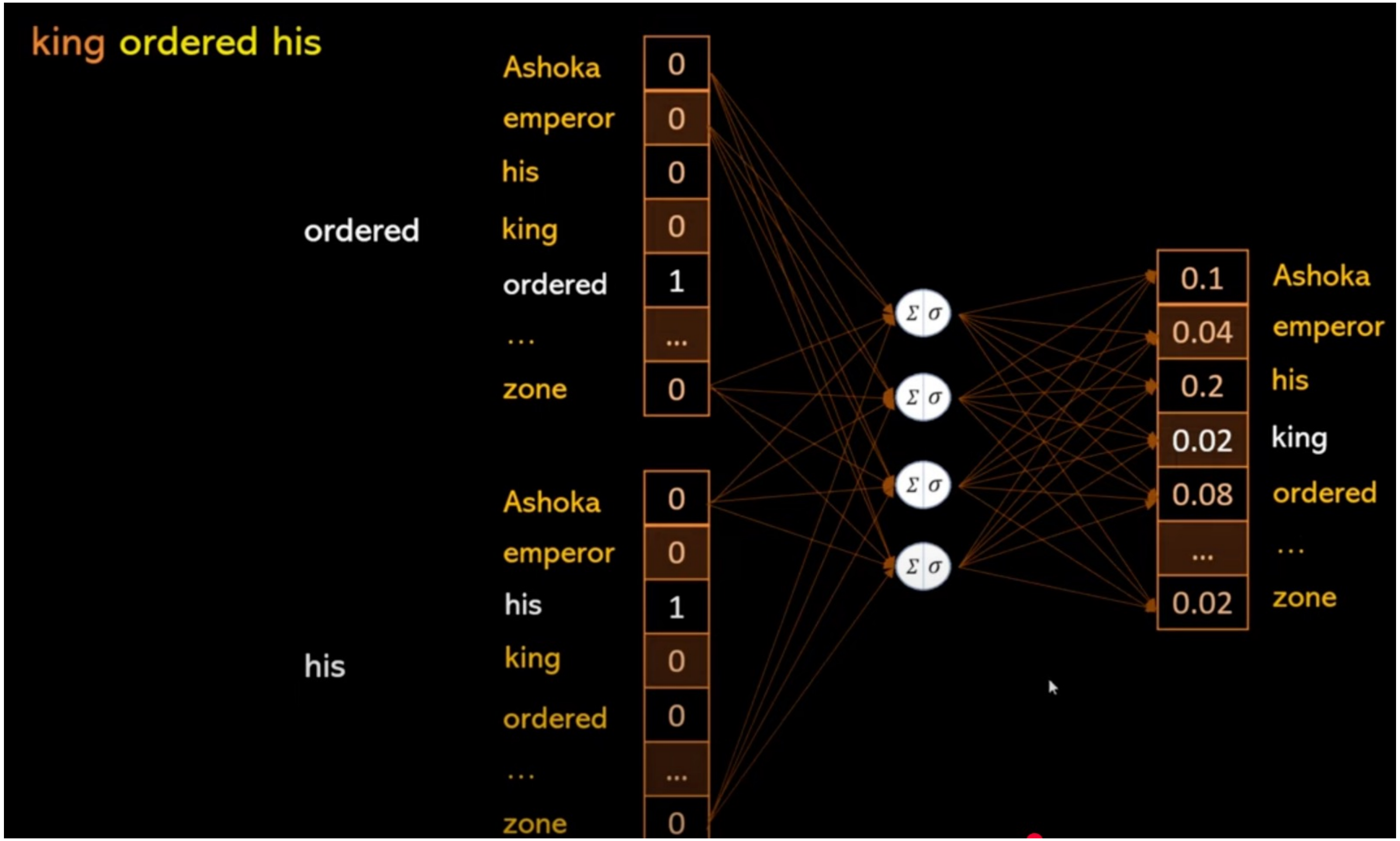
CBOW

- This model has **only one** hidden layer.
- The number of neurones in this hidden layer is the size of the embedding.
- Once the network has good performance, we can extract the embedded words from it.

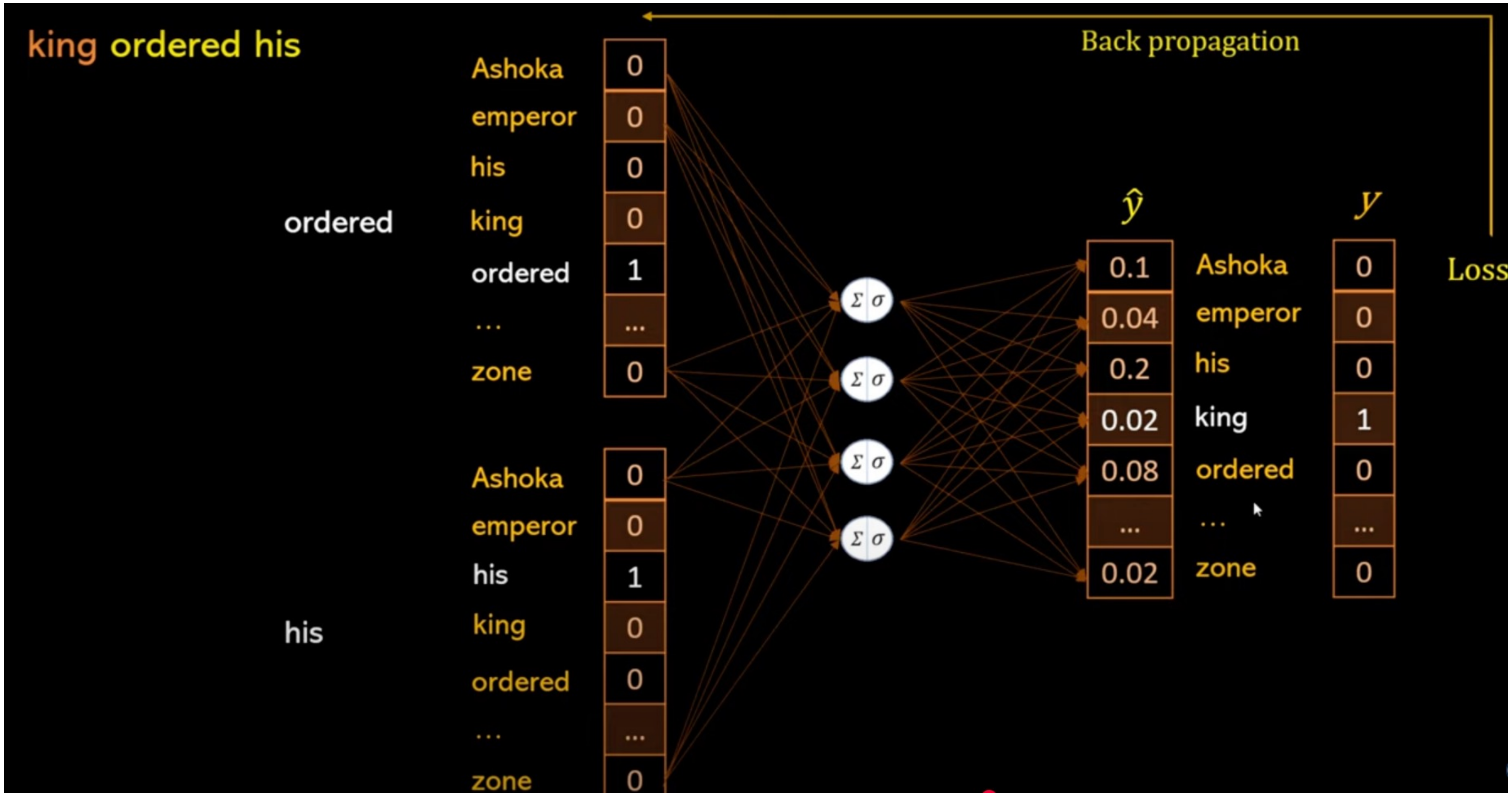




PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING



By doing backpropagation we are adjusting all these weights

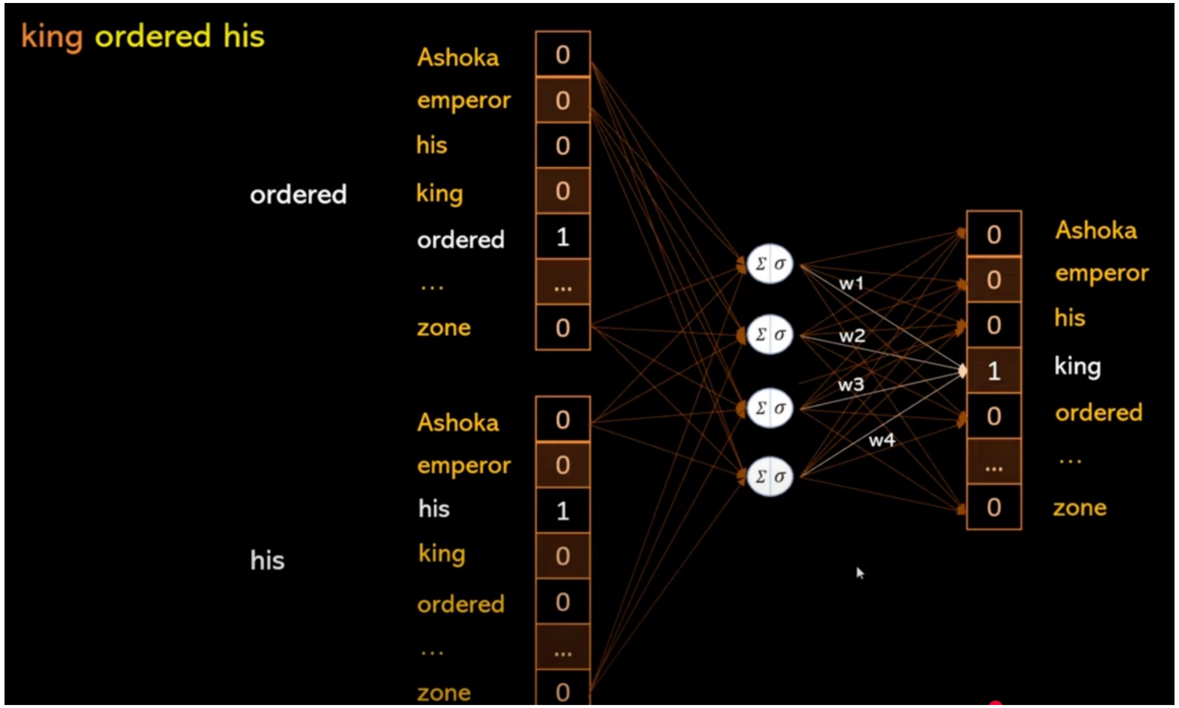


Compare actual output (y) with predicted output (y hat).

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

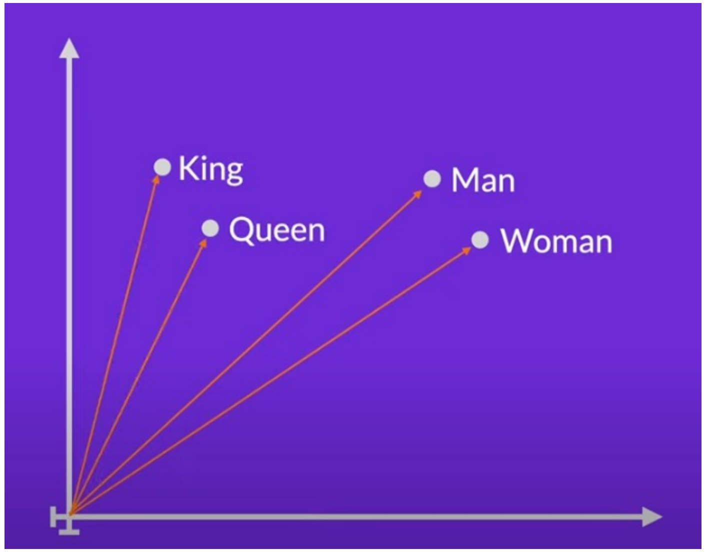
Do backpropagate

- 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)



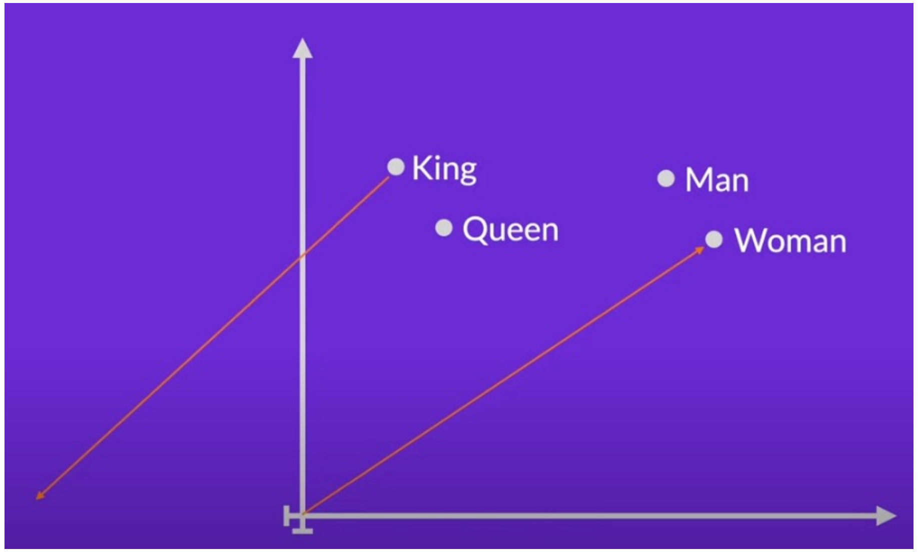
Some interesting results

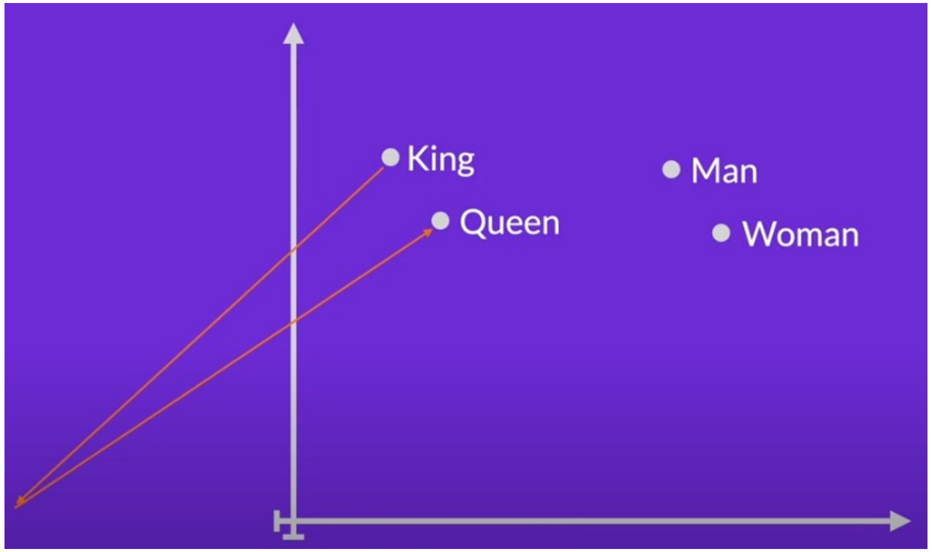
- For some cases, it is even possiblle to make sure that the relative distances between word represent contextual information.



PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

Some interesting results





King-man

King-man+woman

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

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.

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]

Sentence embedding

- **FastSent**
- **Sent2Vec**
- **Sentence-Transformers (all-MiniLM-L6-v2)**

Scanner


Scan as you shop
1ST CAPE SHIRZ 5.50
BANANAS 5 PACK 1.35
LARGE STIR FRY 1.70
2 SOUP 2.30
2 CARROT SOUP 2.30
2 SALMON 4.60
3 W 400G W/M MED 2.55
2 MUESLI 6.00
Items 14
Savings 0.00
26.30
Item
Remove

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

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

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

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

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

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

Log file (test-suites)

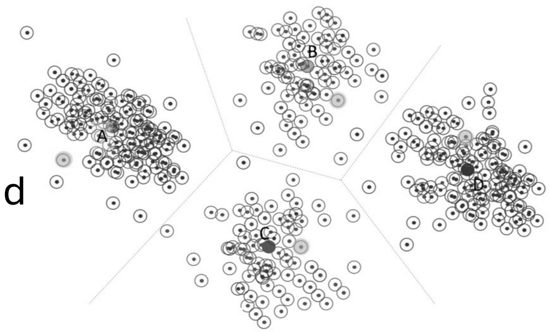
Testsuite	Number of clients	Number of events
1026-event	61	1026

We are going to cluster these 61 clients

We treat each client as a sentence (session)

The propose model

Log partitioning → Event embedding → Session embedding → K-means clustering



Each point is a vector and represents one client.

Word2Vec on all the Sessions

The Word2Vec package from the Gensim libray

The dimension of the W2Vvectors 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']"]

One session=one client

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

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

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

Averaging a session

- **client60:**

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

([0.15698704 -0.79322034 -0.3945347 -0.82248145 0.21587323 -0.4298337 0.11635129 0.06533043 -0.01920184 0.09027459 -0.78751177 -0.46802115 -0.5892027 -0.9514586 -0.21800822] + [1.13778 -2.4118261 -1.158108 -2.5120077 0.8757284 -0.8175933 0.31543404 0.43288264 0.16992638 -0.16874686 -2.5441096 -1.5867229 -2.180273 -1.9017196 -0.28566715] + ...) = [7.398182 -13.992377 -6.3737016 -14.275244 5.5125384 -3.9403927 1.6540279 2.750914 1.7304657 -1.40134 -14.432299 -9.090877 -12.594746 -9.036384 -1.28486] / 7 = **[0.7877907 -0.47995558 -0.06433037 -1.0645988 -0.9146954 0.08257212 -0.16243137 0.6493701 -1.9037664 0.31899315 1.2860477 1.3689787 0.51519257 -0.04324638 -0.37049222]**

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

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:

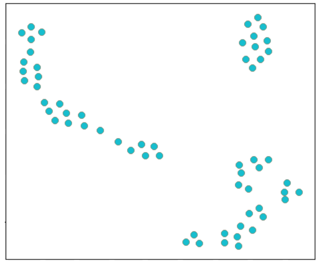
[1.1439486 -0.2789368
-1.0483342 -1.9299325 -
0.29067624 0.24841422
0.6557632 -0.4220297 -
0.00312602 0.02432763 -
1.5237406
1.0237519
-0.9625526 1.6117748
0.19230707]

Now we have a measure for each session.

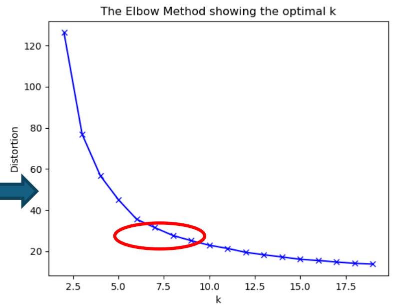
PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

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]*

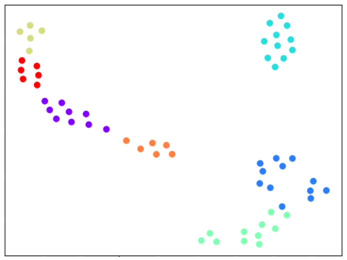


1026-event



Elbow method

K=7



PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

TP

```python
import pandas as pd
data='./1026-clients-U.csv'
df=pd.read_csv(data)
df.head()
```

	1	1584454655792	client0	scan0_0	debloquer	[]	0
0	2	1584454655801	client0	scan0_0	scanner	[8718309259938]	0
1	3	1584454656089	client1	scan1_1	debloquer	[]	0
2	4	1584454656095	client0	scan0_0	scanner	[3560070976478]	0
3	5	1584454656105	client1	scan1_1	scanner	[3560070048786]	0
4	6	1584454656127	client2	scan2_2	debloquer	[]	0

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

Log partitioning based on ClienTID

1026-clients-U.csv → Log partitioning → Sessions-1026.npy

client0': [['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']],
' **client1**': [['debloquer', 'Nothing', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'Barcode', '0'], ['transmission', 'CaisseNumber', '0'], ['abandon', 'Nothing', 'Error'], ['payer', 'Price-integer', 'Float Number']],
' **client2**': [['debloquer', 'Nothing', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'Barcode', '-2'], ['scanner', 'Barcode', '0'], ['scanner', 'ErrorBarcode', '0'], ['transmission', 'CaisseNumber', '0'], ['abandon', 'Nothing', 'Error'], ['ouvrirSession', 'Nothing', '0'], ['ajouter', 'Barcode', '0'], ['fermerSession', 'Nothing', '0'], ['payer', 'Price-float', '0']],

```
import numpy as np
Sessions = np.load('Sessions-1026.npy',allow_pickle='TRUE').item()
print(Sessions)
```


```
{' client0': [['debloquer', 'Nothing', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'Barc
code', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'ErrorBarcode', '0'], ['scanner', 'Ba
arcode', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'Barcode', '0'], ['transmission', 'CaisseNumber', '0'], ['abando
' client1': [['debloquer', 'Nothing', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'Barco
ode', '0'], ['transmission', 'CaisseNumber', '0'], ['abandon', 'Nothing', 'Error'], ['payer', 'Price-integer', 'Float Nu
anner', 'Barcode', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'Barcode', '-2'], ['scanner', 'Barcode', '0'], ['scann
r', '0'], ['abandon', 'Nothing', 'Error'], ['ouvrirSession', 'Nothing', '0'], ['ajouter', 'Barcode', '0'], ['fermerSessi
lient3': [['debloquer', 'Nothing', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'Barcode'
e', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'Barcode', '0'], ['transmission', 'CaisseNumber', '0'], ['abandon', 'I
ber']], ' client4': [['debloquer', 'Nothing', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'Barcode', '0'], ['scanner'
r', 'Barcode', '0'], ['scanner', 'Barcode', '0'], ['supprimer', 'Barcode', '0'], ['scanner', 'Barcode', '-2'], ['scanner
r', 'Barcode', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'Barcode', '0'], ['scanner',
'Barcode', '0'], ['supprimer', 'Barcode', '0'], ['scanner', 'Barcode', '0'], ['transmission', 'CaisseNumber', '0'], ['ab
'0'], ['ajouter', 'Barcode', '0'], ['fermerSession', 'Nothing', '0'], ['payer', 'Price-float', '0']], ' client5': [['deb
['scanner', 'Barcode', '0'], ['scanner', 'ErrorBarcode', '0'], ['scanner', 'Barcode', '-2'], ['scanner', 'Barcode', '0']
```

Dictionary

- Key
- value

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'], ['a
 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']]
```

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

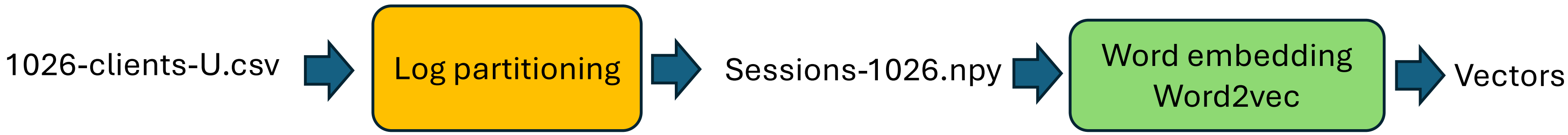
Change event to string

```
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 ClienTID



PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

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 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 word

```
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)
```

End