



DATA SCIENCE CONSULTING

Session 3

February 6th, 2023

Agenda



1. **Customer Journey restitution**
2. KPI definition and Customer Journey
3. Steering Committee
4. Back to TF-IDF approach
5. Word embedding definition
6. Word embedding approaches
 - A. Latent Semantic Indexing (LSI) technique
 - B. An advanced embedding method: Word2Vec
 - C. Opening on other techniques: FastText
7. Hands-on session
8. Summary of the session

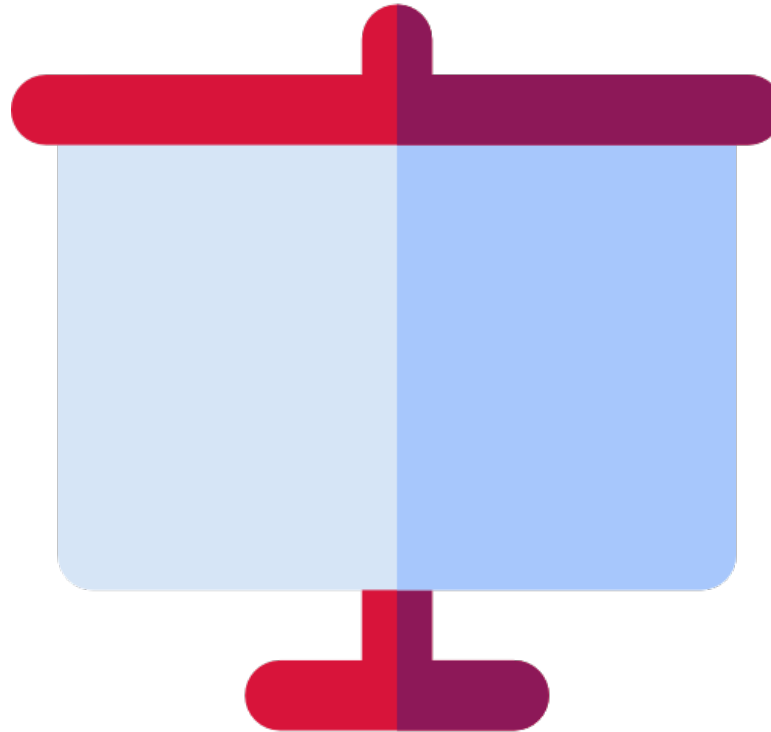


Restitution



10'

Customer Journey Interviews ? What did you find ?



Agenda



1. Customer Journey restitution
- 2. KPI definition and Customer Journey**
3. Steering Committee
4. Back to TF-IDF approach
5. Word embedding definition
6. Word embedding approaches
 - A. Latent Semantic Indexing (LSI) technique
 - B. An advanced embedding method: Word2Vec
 - C. Opening on other techniques: FastText
7. Hands-on session
8. Summary of the session



Adapt your KPIs to your goal



assess the customer relationship strategy of a **French energy supplier** and provide recommendations for each stage of the **customer journey**



- Why do we implement KPIs ?
 - To ensure that the offer we are launching is expected to meet its business goals

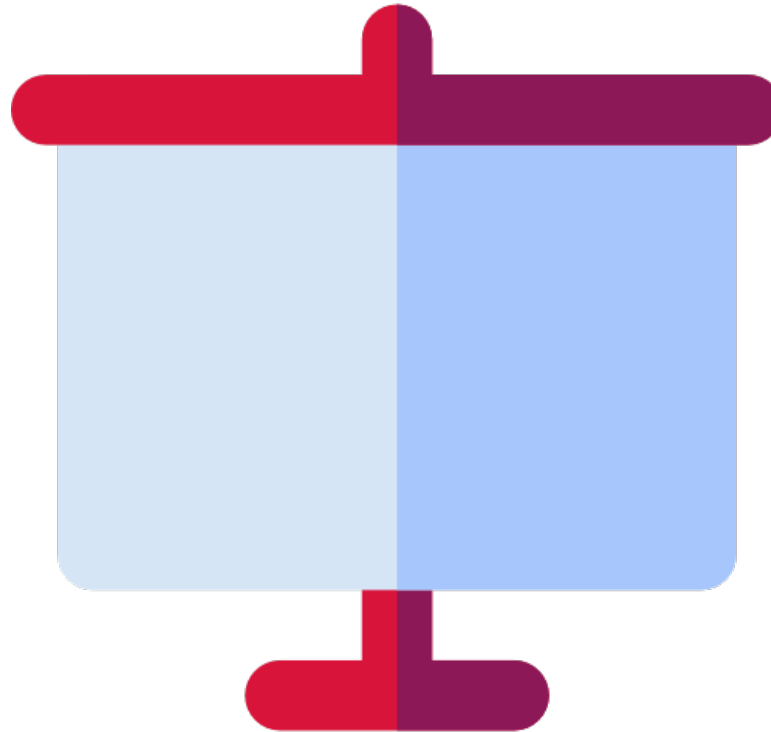
Which KPIs are relevant to our goal ?



Brainstorming



1. Identify KPI categories
2. For each category, define 2-3 KPIs to implement





Example of KPIs



Website Visits

Examples

- Time spent on website
- Number of page visited
- Customer acquisition rate
- Search to website rate

Subscription and consumption

Examples

- Energy consumption analysis
- Customer acquisition rate
- Call center metrics
- Churn rate
- Payment collection rate

Satisfaction and follow up

Examples

- Net promoter score (NPS)
- Number of referrals
- Time to resolution
- Number of customer complaints
- Amount of cross-sell opportunities



Brainstorming



- Suggest KPIs at each step of the customer journey to assess the customer relationship strategy of a French energy supplier
- Identify what you will need to calculate those KPIs
- Propose action plan to gather this data

**Present your results during the
Steering Committee**



Agenda



1. Customer Journey restitution
2. KPI definition and Customer Journey
- 3. Steering Committee**
4. Back to TF-IDF approach
5. Word embedding definition
6. Word embedding approaches
 - A. Latent Semantic Indexing (LSI) technique
 - B. An advanced embedding method: Word2Vec
 - C. Opening on other techniques: FastText
7. Hands-on session
8. Summary of the session



Governance bodies



Governance body



Duration & frequency



Content



Sponsors meeting

- 1 hour
- Once a month

- Review of overall project results
- Validations of :
 - Strategic orientations
 - Cross-segment arbitrations and structuring decisions



Steering committee

- 1 hour
- Once a month

- Review of overall project results (advancements, added value, costs...)
- Breaking points and alerts
- Decision making or submission of arbitrations to the Sponsors



Project committee

- 1 hour
- Once a week

- Operational alignment on achievements and on-going actions
- Identifications of attention points



Core Team stand-up

- 15 minutes
- Each day

- Sharing of what has been done the day before and what is foreseen this day
- Solving of simple problems



Steering Committee : 13/02



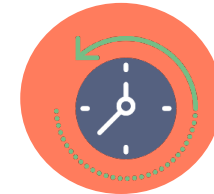
Participants

- Project team
- Client team
- Sponsor



Objectives

- Review overall project results
 - Overall status per stream
 - Latest version of NLP Results
 - Customer journey and KPIs
 - Foreseen next steps



Format

- On a regular basis, all along the project progress
 - On average once a month
- Short and efficient – focused on results & key questions
 - Duration ~ hour

Format of the restitution : 20mn presentation + 10mn questions

Agenda



1. Customer Journey restitution
2. KPI definition and Customer Journey
3. Steering Committee
- 4. Back to TF-IDF approach**
5. Word embedding definition
6. Word embedding approaches
 - A. Latent Semantic Indexing (LSI) technique
 - B. An advanced embedding method: Word2Vec
 - C. Opening on other techniques: FastText
7. Hands-on session
8. Summary of the session



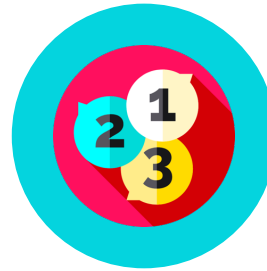
Data pipeline



Data Collection



Data Cleaning



Word Embedding



Topic Extraction



Sentiment Analysis



Back to the TF-IDF approach

Raw data

Document ID	Textual description
1	"Data science is fun"
2	"Artificial intelligence is the future"
3	"Business and artificial intelligence combination is the key"



Cleaned data

Document ID	Cleaned text
1	["data", "science", "fun"]
2	["artificial", "intelligence", "future"]
3	["business", "artificial", "intelligence", "combination", "key"]



Dictionary

["data", "science", "fun", "artificial", "intelligence", "future", "business", "combination", "key"]



$$TF_{ij} = \frac{N_{ij}}{|d_j|}$$

$$IDF_i = \log \frac{N}{df_i}$$

$$w_{ij} = TF_{ij} \cdot IDF_i$$

TF (Term Frequency)

Word	data	science	fun	artificial	intelligence	future	business	combination	key
Document									
1	0.33	0.33	0.33	0	0	0	0	0	0
2	0	0	0	0.33	0.33	0.33	0	0	0
3	0	0	0	0.2	0.2	0	0.2	0.2	0.2

IDF (Inverse Document Frequency)

Word	data	science	fun	artificial	intelligence	future	business	combination	key
Document									
1	0,48	0,48	0,48	0,18	0,18	0,48	0,48	0,48	0,48
2	0,48	0,48	0,48	0,18	0,18	0,48	0,48	0,48	0,48
3	0,48	0,48	0,48	0,18	0,18	0,48	0,48	0,48	0,48

TF-IDF

Word	data	science	fun	artificial	intelligence	future	business	combination	key
Document									
1	0.16	0.16	0.16	0,00	0,00	0,00	0,00	0,00	0,00
2	0,00	0,00	0,00	0,06	0,06	0.16	0,00	0,00	0,00
3	0,00	0,00	0,00	0,04	0,04	0,00	0,1	0,1	0,1



LIMITS

- ✓ Syntax
- ✓ Semantics (meaning and relationship)

- ✓ Synonyms
- ✓ Sparsity

Agenda



1. Customer Journey restitution
2. KPI definition and Customer Journey
3. Steering Committee
4. Back to TF-IDF approach
- 5. Word embedding definition**
6. Word embedding approaches
 - A. Latent Semantic Indexing (LSI) technique
 - B. An advanced embedding method: Word2Vec
 - C. Opening on other techniques: FastText
7. Hands-on session
8. Summary of the session



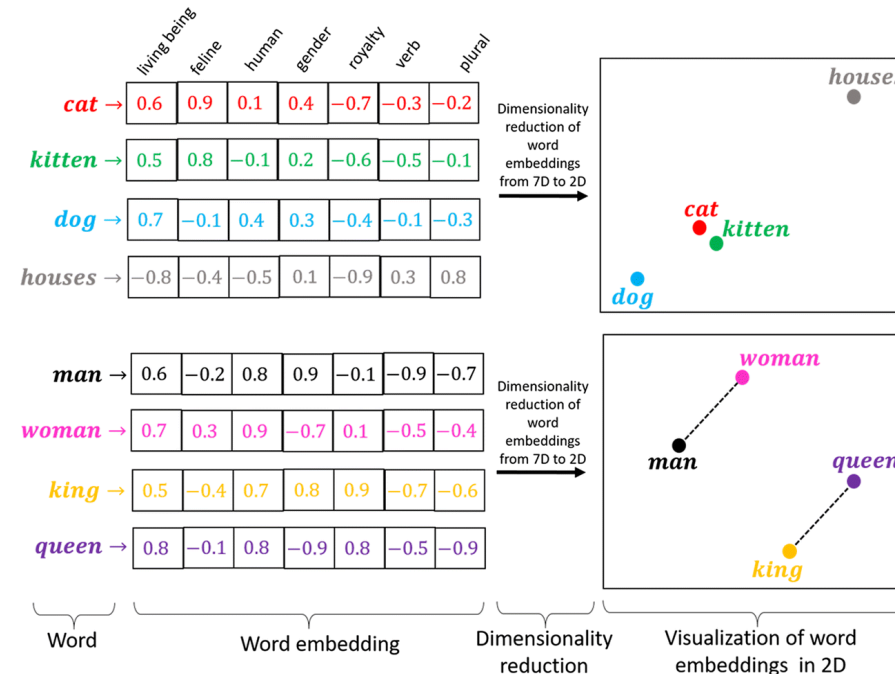
Word embedding definition¹



Word embedding is a type of word representation that allows words with similar meaning to have a similar representation.

Before word embedding

Man != Woman



After word embedding

Man ~ Woman



ADVANTAGES

- ✓ Semantics (meaning and relationship) embedded
- ✓ Dense and low-dimensional vectors

Word embedding definition



"Data science is fun ..."



["data", "science", "fun"]

1	0	0	...
0	1	0	
0	0	1	
⋮			

TRAINING



Transformation matrix
(Latent parameters)



BUILD

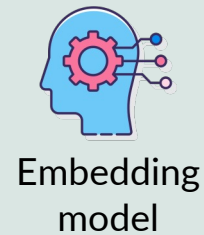


Embedding
model

Most similar words to
"man"

"man"

⋮
1
0
⋮



word	weight
woman	0.76
boy	0.68
girl	0.59
⋮	⋮

Agenda



1. Customer Journey restitution
2. KPI definition and Customer Journey
3. Steering Committee
4. Back to TF-IDF approach
5. Word embedding definition
6. Word embedding approaches
 - A. **Latent Semantic Indexing (LSI) technique**
 - B. An advanced embedding method: Word2Vec
 - C. Opening on other techniques: FastText
7. Hands-on session
8. Summary of the session



Latent Semantic Indexing (LSI) technique (1/7)



Latent Semantic Analysis/Indexing (LSA/LSI) is a mathematical method for modeling the meaning of words by analysing a corpus of texts

- Map each **document** into some **concepts**
- Map each **term** into some **concepts**

Concepts are defined as a set of terms, with corresponding weights

For example, **Sport** concept :

« ball » (0.8)

« player » (0.6)

« run » (0.5)

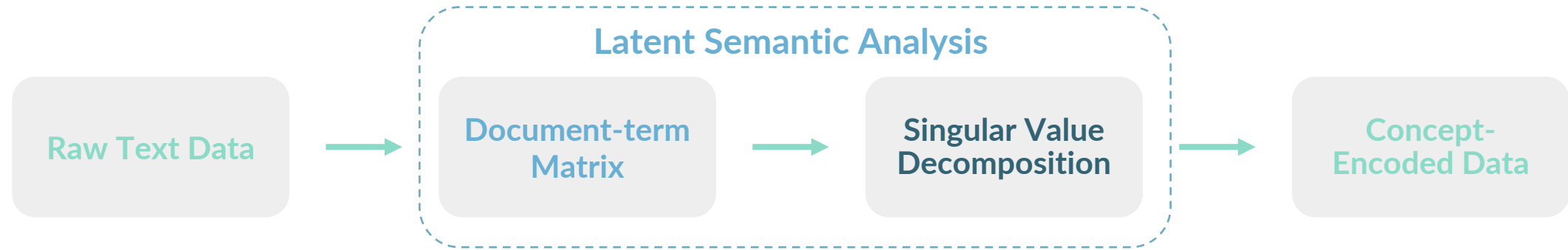
Term-concept matrix

	Sport concept	Cooking concept
ball	0.8	0
player	0.6	0
run	0.5	0
cake	0	0.9
oven	0	0.4

Document-concept matrix

	Sport concept	Cooking concept
Doc 1	0.7	0
Doc 2	0.6	0
Doc 3	0	0.5
Doc 4	0	0.2

Latent Semantic Indexing (LSI) technique (2/7)



Document-term Matrix



- Get a large collection of texts, representative of human language
- Build a matrix with documents as row and terms as columns (TF, TF-IDF...)

Singular Value Decomposition



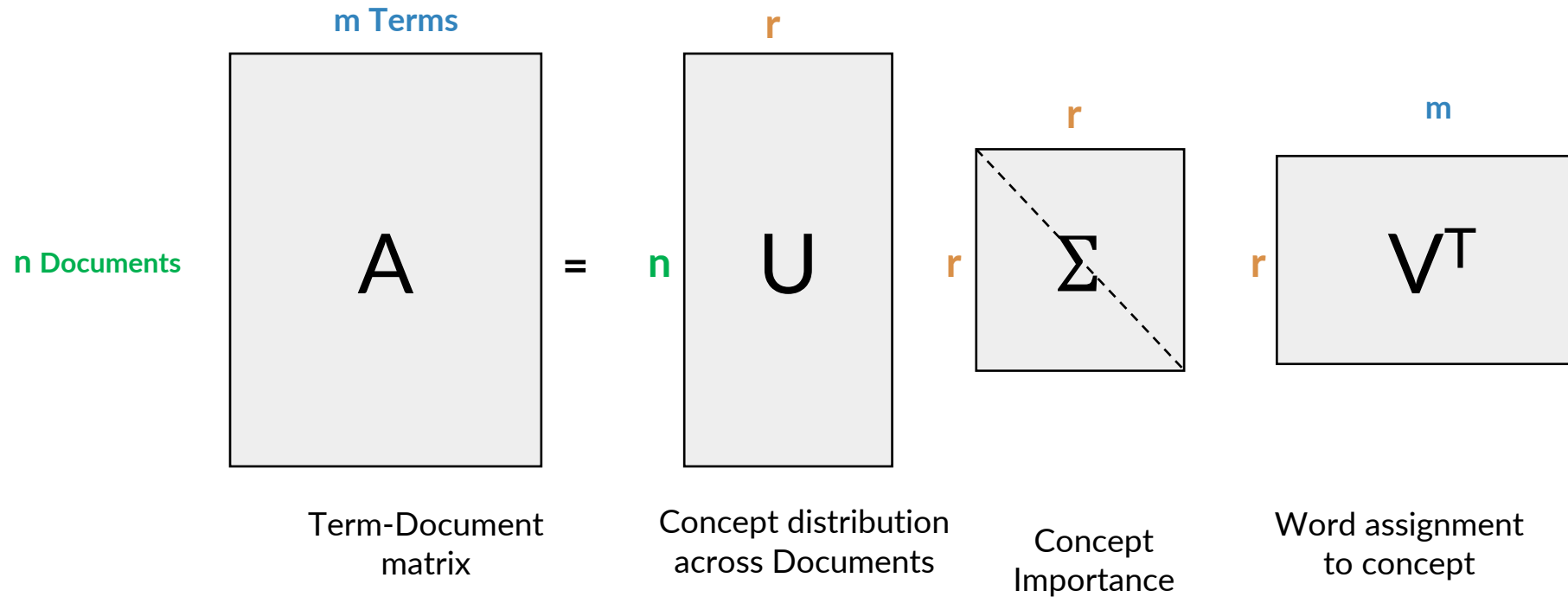
- Document-term matrix is **decomposed** into product of matrices using SVD method
- **Dimensionality reduction** can be applied, keeping the k largest singular values and their associated vectors (LSI-space)



Latent Semantic Indexing (LSI) technique (3/7)



Singular Value Decomposition

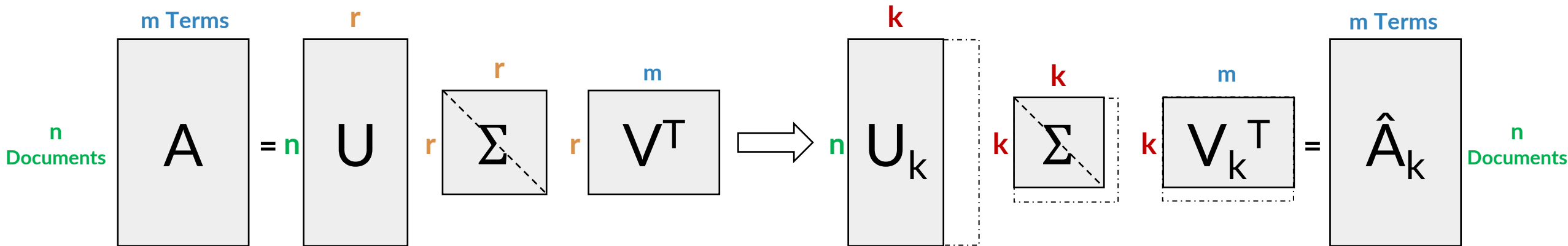


$$A_{[n \times m]} = U_{[n \times r]} \Sigma_{[r \times r]} (V_{[m \times r]})^T$$



Latent Semantic Indexing (LSI) technique (4/7)

●●●
SVD enables dimensionality reduction



We perform a SVD
 U and V are orthogonal
matrices
 Σ is a diagonal matrix

The matrix \hat{A}_k is obtained
by keeping the first k
columns of U and V , and
the k largest elements of
 Σ

To fix an optimal value of
“ k ” we can rely on the
construction error:
 $\text{error}(A, \hat{A})$



Latent Semantic Indexing (LSI) technique (5/7)

Similarity in LSI Space

Term-Term
similarity

$$A^t A = (U \Sigma V^T)^T (U \Sigma V^T)$$

$$A^t A = (V^T \Sigma^T U^{TT}) (U \Sigma V^T)$$

$$A^t A = (V^T \Sigma^T) (\Sigma V^T)$$

$$A^t A = V \Sigma^2 V^T$$

V are the eigenvectors of the covariance matrix $A^t A$.

Document-Document
similarity

$$A A^t = (U \Sigma V^T) (U \Sigma V^T)^T$$

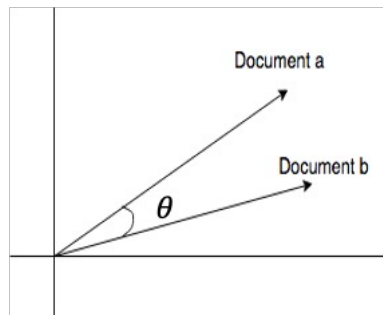
$$A A^t = (U \Sigma V^T) (V^{TT} \Sigma^T U^T)$$

$$A A^t = (U \Sigma) (\Sigma^T U^T)$$

$$A A^t = U \Sigma^2 U^T$$

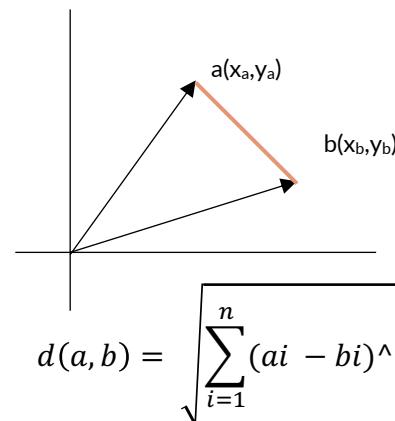
U are the eigenvectors of the Gram matrix $A A^t$.

Cosine similarity

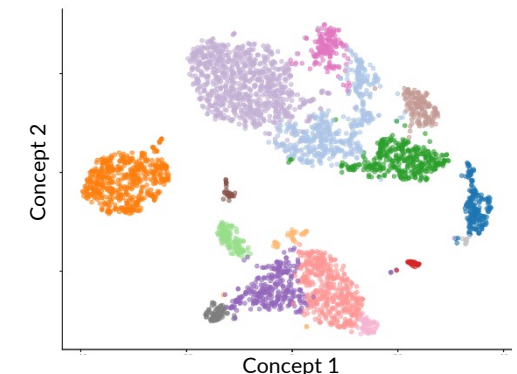


$$\text{sim}(a, b) = \cos(\theta) = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \cdot \|\vec{b}\|}$$

Euclidian norm



Clustering techniques can also be used with the new LSI-space vectors





Latent Semantic Indexing (LSI) technique (6/7)



Advantages

- Powerful and generalizable tool
- Enables dimensionality reduction in an easily interpretable space
- Gives a good way to compute distance or similarity between documents/terms
- Helps for various NLP tasks : search and retrieval, classification, filtering

Limits

- Interpretable meaning of LSI transformation can be complex sometimes even though the mathematical part is valid
- LSI is very sensitive to new type of text that are widely used today² (contracted words, slang...)
- LSI always preserves linear regularities among words³, hard to deal with synonyms



Latent Semantic Indexing (LSI) technique (7/7)



15'

Implement LSI in python to extract hidden features from your text data and perform dimensionality reduction





Break



10'

Feel free to help yourself ! See you at 16.30 !



Agenda



1. Customer Journey restitution
2. KPI definition and Customer Journey
3. Steering Committee
4. Back to TF-IDF approach
5. Word embedding definition
6. Word embedding approaches
 - A. Latent Semantic Indexing (LSI) technique
 - B. An advanced embedding method: Word2Vec**
 - C. Opening on other techniques: FastText
7. Hands-on session
8. Summary of the session



Word2Vec is a classification model based on the context



Word2vec

- ✓ Reduced continuous (dense) word representations trained on large **unlabeled corpora**



Corpus example: “**The** **cat** **purred** **loudly**”
{“The”, “cat”, “loudly”} => **context words**
{“purred” } => **target word**

“**The** **cat** _____ **loudly**” → predict that “**purred**” is the target word

“_____ **purred** _____” → predict the context : {“**The**”, “**cat**”, “**loudly**”}

[4] [Dhruvil Karani 2018, Introduction to word embedding and word2vec](#)

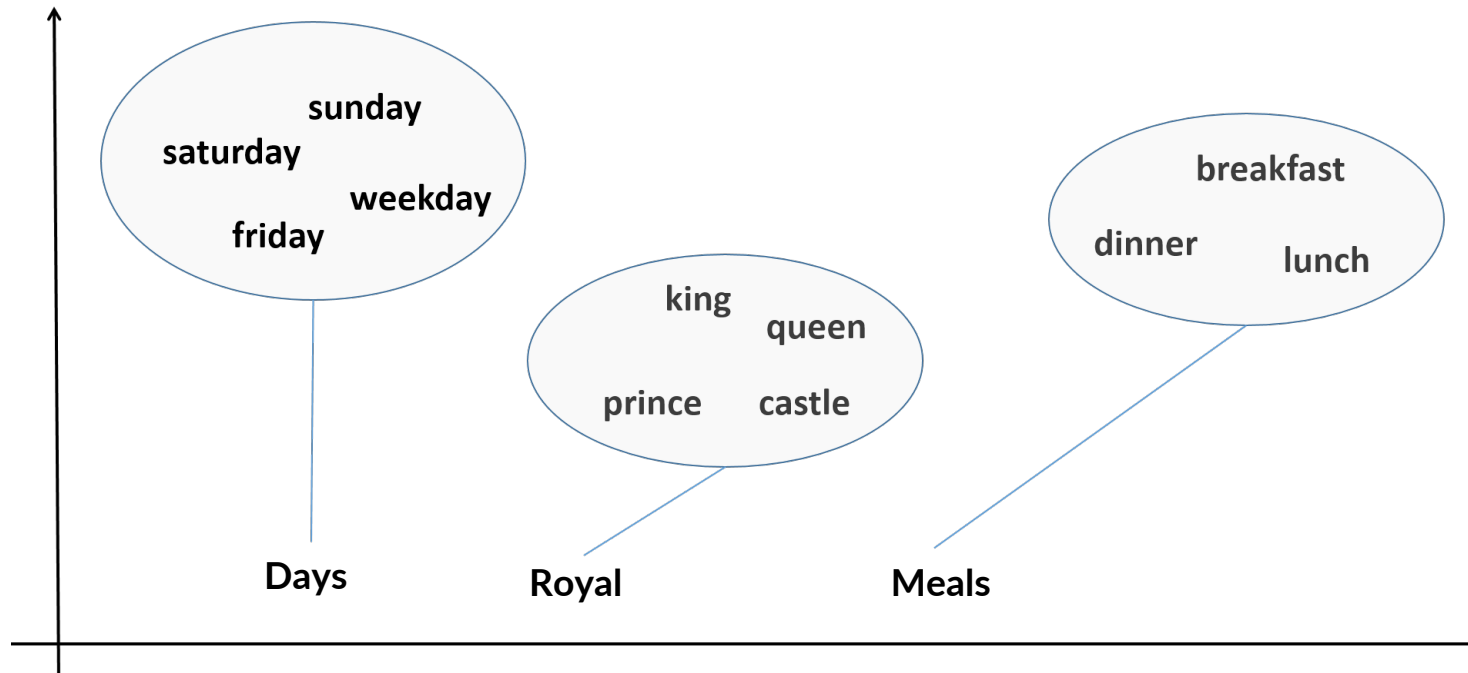


Word2Vec is a classification model based on the context



Word2vec

- ✓ Reduced continuous (dense) word representations trained on large **unlabeled corpora**
- ✓ Word2Vec is a representation of document vocabulary in which words with **similar context occupy close spatial positions**



[4] <https://samyzaf.com/ML/nlp/nlp.html>



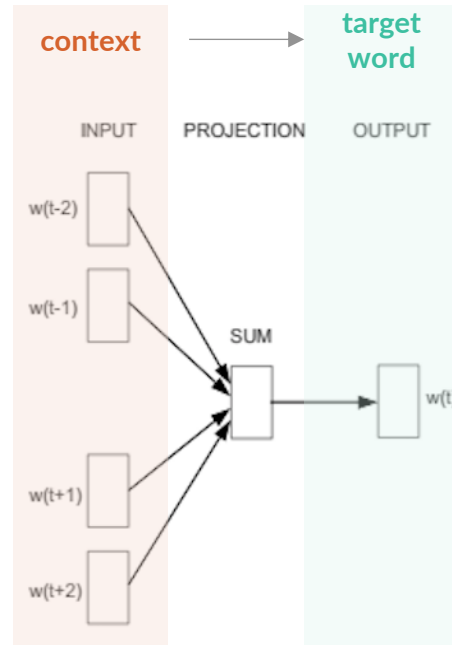
Word2vec has 2 possible architectures : CBOW and Skipgram



Corpus example: “The cat purred loudly”
{“The”, “cat”, “loudly”} => **context words**
{“purred” } => **target word**

Continuous Bag of words (CBOW)

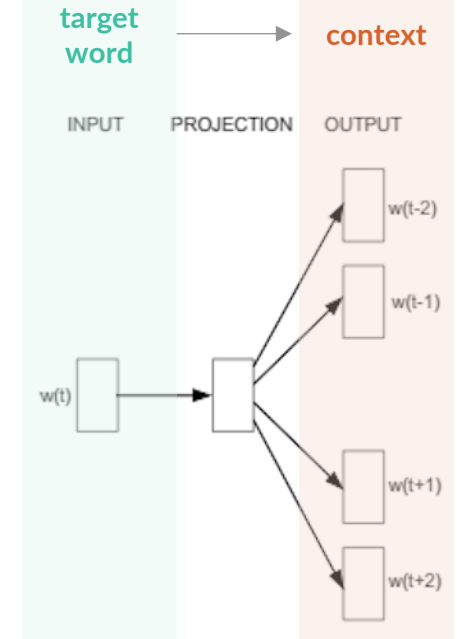
Takes the **context** of each word as the input and tries to predict the word corresponding to the context being the **target**



Predict “purred” knowing the context words {“The”, “cat”, “loudly”}

Skipgram

Takes the **target** word and tries to predict the **context** words of that target word and produce representations



Predict {“The”, “cat”, “loudly”} knowing the target word “purred”



Zoom on CBOW: One-word context₇(1/2)



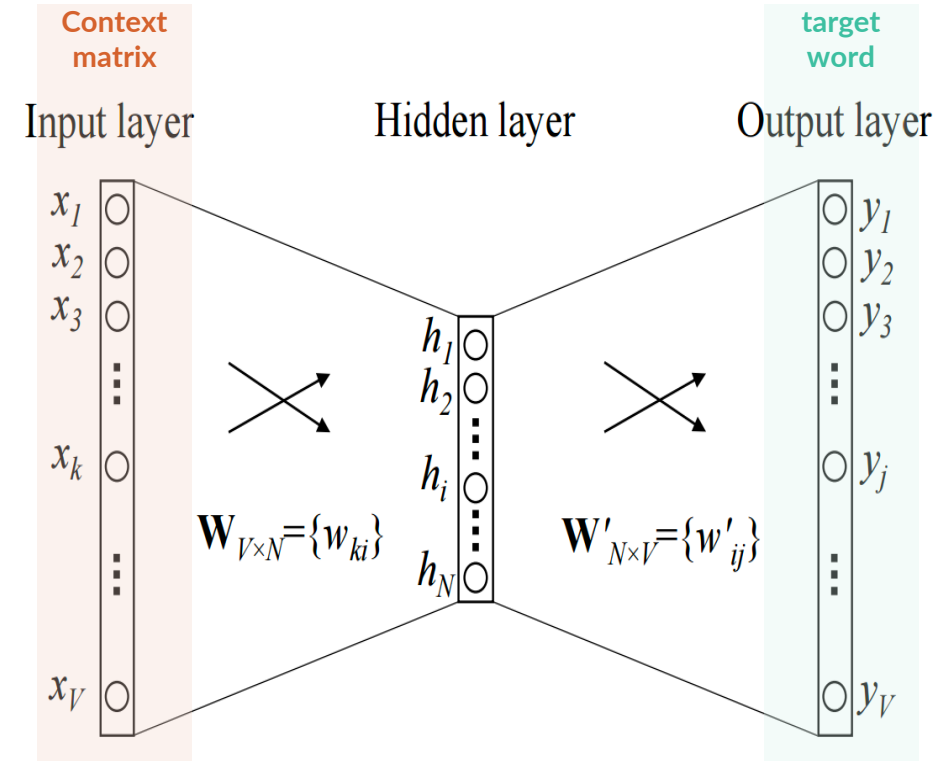
$$\mathbf{h} = \mathbf{W}^T \mathbf{x} = \mathbf{W}^T_{(k,.)}$$

$\mathbf{u}_j = \mathbf{v}'_{w_j}{}^T \mathbf{h}$ where \mathbf{v}'_{w_j} is the j -th column of \mathbf{W}'

$$y_j = \frac{\exp(u_j)}{\sum_{k=1}^V \exp(u_k)}$$

Such as:

- \mathbf{x} : one-hot encoded vector of a word
- \mathbf{W} : weight matrix of size $V * N$
- \mathbf{W}' : weight matrix of size $N * V$
- V : size of the vocabulary
- N : hidden layer size





Zoom on CBOW: One-word context₇ (2/2)



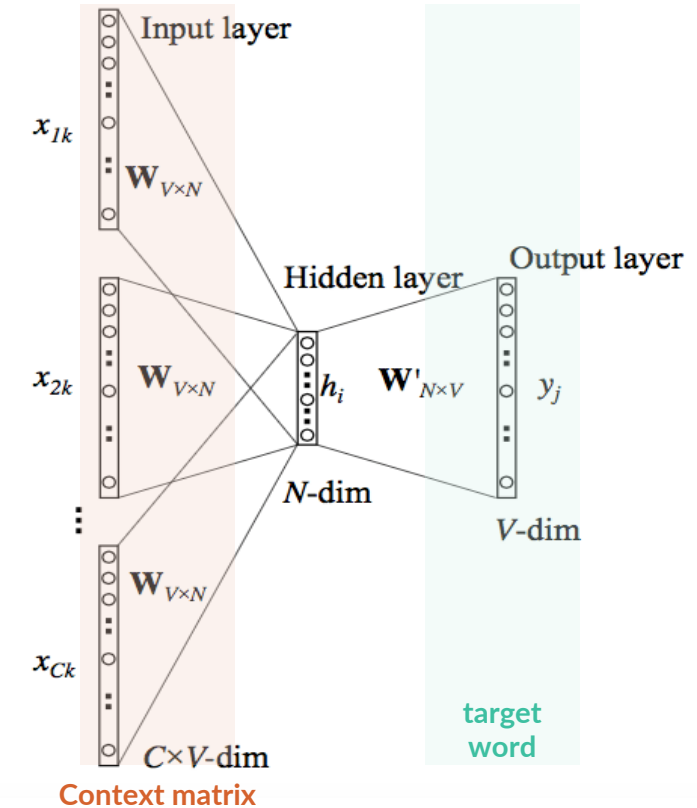
$$h = \frac{1}{C} W^T (x_1 + x_2 + \dots + x_C)$$

$$h = \frac{1}{C} (vw_1 + vw_2 + \dots + vw_C)^T$$

Such as:

- **C** : number of the context word
- **V** : size of the vocabulary
- **N** : hidden layer size

Score function and softmax output as previously





Zoom on Skipgram₇



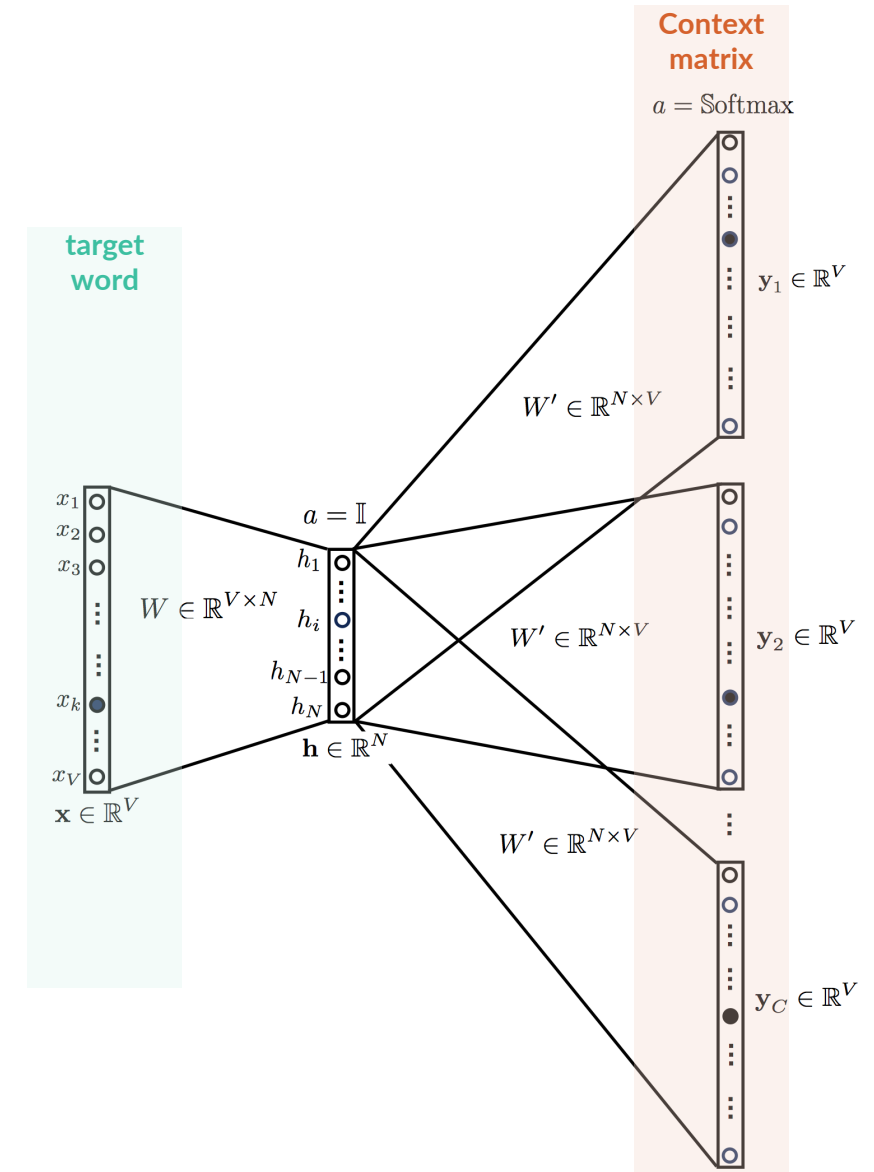
$$\mathbf{h} = \mathbf{W}\mathbf{T}\mathbf{x} = \mathbf{W}^T(\mathbf{k}_{\cdot})$$

$\mathbf{u}_{c,j} = \mathbf{v}'_{w_j}{}^T \mathbf{h}$ for c in $1, \dots, C$ where \mathbf{v}'_{w_j} is the j -th column of \mathbf{W}'

$\mathbf{u}_{c,j} = \mathbf{u}_j$ because the output layer share the same weight matrix

$$y_{c,j} = \frac{\exp(\mathbf{u}_{c,j})}{\sum_{j'=1}^V \exp(\mathbf{u}_{j'})}$$

- V : size of the vocabulary
- N : hidden layer size





Step-by-step example of word2vec



Contexts and targets

#1	natural	language	processing	and	machine	learning	is	fun	and	exciting	#1
	X_k	$Y(c=1)$	$Y(c=2)$								
#2	natural	language	processing	and	machine	learning	is	fun	and	exciting	#2
	$Y(c=1)$	X_k	$Y(c=2)$	$Y(c=3)$							
#3	natural	language	processing	and	machine	learning	is	fun	and	exciting	#3
	$Y(c=1)$	$Y(c=2)$	X_k	$Y(c=3)$	$Y(c=4)$						
#4	natural	language	processing	and	machine	learning	is	fun	and	exciting	#4
		$Y(c=1)$	$Y(c=2)$	X_k	$Y(c=3)$	$Y(c=4)$					
#5	natural	language	processing	and	machine	learning	is	fun	and	exciting	#5
			$Y(c=1)$	$Y(c=2)$	X_k	$Y(c=3)$	$Y(c=4)$				
#6	natural	language	processing	and	machine	learning	is	fun	and	exciting	#6
				$Y(c=1)$	$Y(c=2)$	X_k	$Y(c=3)$	$Y(c=4)$			
#7	natural	language	processing	and	machine	learning	is	fun	and	exciting	#7
					$Y(c=1)$	$Y(c=2)$	X_k	$Y(c=3)$	$Y(c=4)$		
#8	natural	language	processing	and	machine	learning	is	fun	and	exciting	#8
						$Y(c=1)$	$Y(c=2)$	X_k	$Y(c=3)$	$Y(c=4)$	
#9	natural	language	processing	and	machine	learning	is	fun	and	exciting	#9
							$Y(c=1)$	$Y(c=2)$	X_k	$Y(c=3)$	
#10	natural	language	processing	and	machine	learning	is	fun	and	exciting	#10
								$Y(c=1)$	$Y(c=2)$	X_k	


What is the size of the Window=?



Step-by-step example of word2vec



Contexts and targets



#1	natural	language	processing	and	machine	learning	is	fun	and	exciting	#1
	X_k	$Y(c=1)$	$Y(c=2)$								
#2	natural	language	processing	and	machine	learning	is	fun	and	exciting	#2
	$Y(c=1)$	X_k	$Y(c=2)$	$Y(c=3)$							
#3	natural	language	processing	and	machine	learning	is	fun	and	exciting	#3
	$Y(c=1)$	$Y(c=2)$	X_k	$Y(c=3)$	$Y(c=4)$						
#4	natural	language	processing	and	machine	learning	is	fun	and	exciting	#4
		$Y(c=1)$	$Y(c=2)$	X_k	$Y(c=3)$	$Y(c=4)$					
#5	natural	language	processing	and	machine	learning	is	fun	and	exciting	#5
			$Y(c=1)$	$Y(c=2)$	X_k	$Y(c=3)$	$Y(c=4)$				
#6	natural	language	processing	and	machine	learning	is	fun	and	exciting	#6
				$Y(c=1)$	$Y(c=2)$	X_k	$Y(c=3)$	$Y(c=4)$			
#7	natural	language	processing	and	machine	learning	is	fun	and	exciting	#7
					$Y(c=1)$	$Y(c=2)$	X_k	$Y(c=3)$	$Y(c=4)$		
#8	natural	language	processing	and	machine	learning	is	fun	and	exciting	#8
						$Y(c=1)$	$Y(c=2)$	X_k	$Y(c=3)$	$Y(c=4)$	
#9	natural	language	processing	and	machine	learning	is	fun	and	exciting	#9
							$Y(c=1)$	$Y(c=2)$	X_k	$Y(c=3)$	
#10	natural	language	processing	and	machine	learning	is	fun	and	exciting	#10
								$Y(c=1)$	$Y(c=2)$	X_k	

What is the size of the Window=?



Step-by-step example of word2vec



One-hot encoding pass

Target *Context 1* *Context 2*

#	Token	#1			#2			#3			#4			#5		
0	natural	1	0	0	0	1	0	0	1	0	0	0	0	0	0	0
1	language	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0
2	processing	0	0	1	0	0	1	1	0	0	0	1	0	0	1	0
3	and	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1
4	machine	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0
5	learning	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0
6	is	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
7	fun	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	exciting	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		X _k	Y(c=1)	Y(c=2)	X _k	Y(c=1)	Y(c=2)	Y(c=3)	X _k	Y(c=1)	Y(c=2)	Y(c=3)	Y(c=4)	X _k	Y(c=1)	Y(c=2)

#	Token	#6			#7			#8			#9			#10		
0	natural	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	language	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	processing	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	and	0	1	0	0	0	0	0	0	1	1	0	0	0	0	1
4	machine	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0
5	learning	1	0	0	0	0	1	0	0	1	0	0	0	0	0	0
6	is	0	0	0	1	0	1	0	0	0	1	0	0	0	0	0
7	fun	0	0	0	0	1	0	0	1	0	0	1	0	0	1	0
8	exciting	0	0	0	0	0	0	0	1	0	0	1	1	1	0	0
		X _k	Y(c=1)	Y(c=2)	Y(c=3)	Y(c=4)	X _k	Y(c=1)	Y(c=2)	Y(c=3)	Y(c=4)	X _k	Y(c=1)	Y(c=2)	Y(c=3)	Y(c=4)

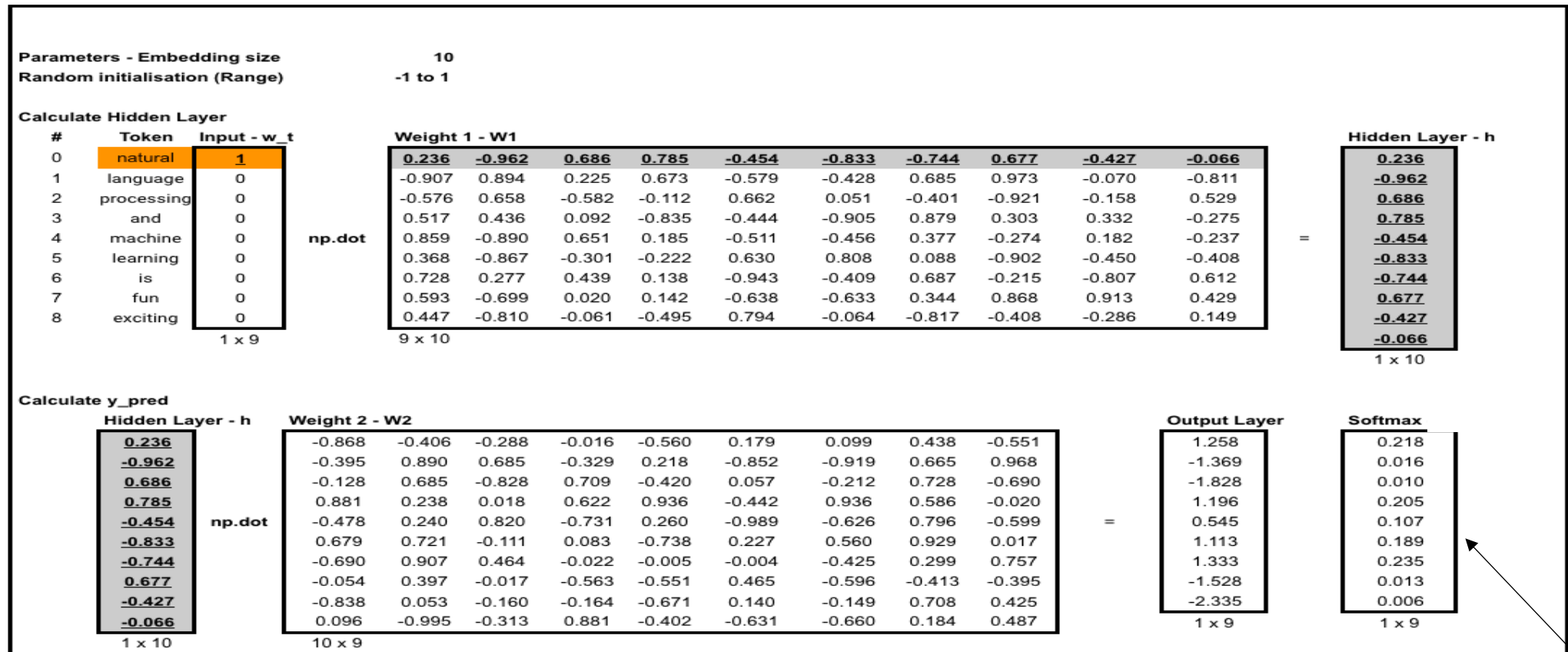
What is the total number of word in the corpus=?



Step-by-step example of word2vec



Forward propagation



What is the type of architecture (skipgram or CBOW)?



Step-by-step example of word2vec



Error calculation

y_pred w_c = 1				y_pred w_c = 2				EI	
Softmax		Token	Diff	Softmax		Token	Diff	Sum of Diff	
0.218	0	natural	0.218	0.218	0	natural	0.218	0.436	
0.016	1	language	-0.984	0.016	0	language	0.016	-0.968	
0.010	0	processing	0.010	0.010	1	processing	-0.990	-0.980	
0.205	0	and	0.205	0.205	0	and	0.205	0.411	
0.107	0	machine	0.107	0.107	0	machine	0.107	0.214	
0.189	0	learning	0.189	0.189	0	learning	0.189	0.378	
0.235	0	is	0.235	0.235	0	is	0.235	0.471	
0.013	0	fun	0.013	0.013	0	fun	0.013	0.027	
0.006	0	exciting	0.006	0.006	0	exciting	0.006	0.012	

Probabilities

One hot encoding of context word 1

One hot encoding of context word 2



Step-by-step example of word2vec



Backpropagation

Hidden Layer									
0.236	-0.962	0.686	0.785	-0.454	-0.833	-0.744	0.677	-0.427	-0.066
10 x 1									
EI									
Sum of Diff									
0.436	-0.968	-0.980	0.411	0.214	0.378	0.471	0.027	0.012	
9 x 1									
Delta for W2									
0.103	-0.228	-0.231	0.097	0.050	0.089	0.111	0.006	0.003	
-0.420	0.931	0.943	-0.395	-0.206	-0.363	-0.453	-0.026	-0.012	
0.299	-0.664	-0.672	0.282	0.147	0.259	0.323	0.018	0.008	
0.343	-0.760	-0.769	0.322	0.168	0.296	0.370	0.021	-0.005	
-0.198	0.440	0.445	-0.186	-0.097	-0.171	-0.214	-0.012	-0.005	
-0.364	0.807	0.817	-0.342	-0.178	-0.315	-0.392	-0.022	-0.010	
-0.325	0.721	0.729	-0.306	-0.159	-0.281	-0.350	-0.020	-0.009	
0.295	-0.655	-0.663	0.278	0.145	0.256	0.319	0.018	0.008	
-0.186	0.413	0.418	-0.175	-0.091	-0.161	-0.201	-0.011	-0.005	
-0.029	0.063	0.064	-0.027	-0.014	-0.025	-0.031	-0.002	-0.001	
10 x 9									
Weight (W2)									
-0.868	-0.406	-0.288	-0.016	-0.560	0.179	0.099	0.438	-0.551	
-0.395	0.890	0.685	-0.329	0.218	-0.852	-0.919	0.665	0.968	
-0.128	0.685	-0.828	0.709	-0.420	0.057	-0.212	0.728	-0.690	
0.881	0.238	0.018	0.622	0.936	-0.442	0.936	0.586	-0.020	
-0.478	0.240	0.820	-0.731	0.260	-0.989	-0.626	0.796	-0.599	
0.679	0.721	-0.111	0.083	-0.738	0.227	0.560	0.929	0.017	
-0.690	0.907	0.464	-0.022	-0.005	-0.004	-0.425	0.299	0.757	
-0.054	0.397	-0.017	-0.563	-0.551	0.465	-0.596	-0.413	-0.395	
-0.838	0.053	-0.160	-0.164	-0.671	0.140	-0.149	0.708	0.425	
0.096	-0.995	-0.313	0.881	-0.402	-0.631	-0.660	0.184	0.487	
10 x 9									
EI									
Sum of Diff									
0.436	-0.968	-0.980	0.411	0.214	0.378	0.471	0.027	0.012	
9 x 1									
np.dot(W2, EI)									
0.290	-2.518	0.227	0.882	-2.142	-0.042	-1.829	-0.861	-0.465	1.050
10 x 1									
np.outer									
1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
9 x 10									
Delta for W1									
0.290	-2.518	0.227	0.882	-2.142	-0.042	-1.829	-0.861	-0.465	1.050
9 x 10									
np.dot(w_t, np.dot(W2, EI))									
0.290	-2.518	0.227	0.882	-2.142	-0.042	-1.829	-0.861	-0.465	1.050
9 x 10									

Weights update

Weight (W1)										Learning Rate		Delta for W1										Updated Weight (W1)	
0.23583 -0.96174 0.68575 0.78487 -0.45389 -0.83338 0.74431 0.67674 -0.42654 -0.06554										0.01		0.28989 -2.51804 0.22702 0.88178 -2.14232 -0.04230 -1.82874 -0.86144 -0.46461 1.05026										0.23293 -0.93656 0.68348 0.77605 -0.43246 -0.83295 -0.72603 0.68536 -0.42189 -0.07604	
-0.90686 0.89372 0.22517 0.67294 -0.57859 -0.42823 0.68545 0.97326 -0.06998 -0.81135												0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000										-0.90686 0.89372 0.22517 0.67294 -0.57859 -0.42823 0.68545 0.97326 -0.06998 -0.81135	
-0.57642 0.65794 -0.58247 -0.11187 0.66228 0.05126 -0.40076 -0.92055 -0.15796 0.52942												0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000										-0.57642 0.65794 -0.58247 -0.11187 0.66228 0.05126 -0.40076 -0.92055 -0.15796 0.52942	
0.51692 0.43591 0.09182 -0.83478 -0.44396 -0.90529 0.87882 0.30314 0.33212 -0.27488												0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000										0.51692 0.43591 0.09182 -0.83478 -0.44396 -0.90529 0.87882 0.30314 0.33212 -0.27488	
0.85921 -0.88984 0.65071 0.18497 -0.51058 -0.45623 0.37692 -0.27448 0.18150 -0.23724												0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000										0.85921 -0.88984 0.65071 0.18497 -0.51058 -0.45623 0.37692 -0.27448 0.18150 -0.23724	
0.36818 -0.86733 -0.30110 -0.22212 0.63036 0.80755 0.08793 -0.90154 -0.44968 -0.40764												0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000										0.36818 -0.86733 -0.30110 -0.22212 0.63036 0.80755 0.08793 -0.90154 -0.44968 -0.40764	
0.72789 0.27735 0.43892 0.13763 -0.94310 -0.40912 0.68705 -0.21547 -0.80684 0.61176												0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000										0.72789 0.27735 0.43892 0.13763 -0.94310 -0.40912 0.68705 -0.21547 -0.80684 0.61176	
0.59291 -0.69651 0.02039 0.14180 -0.63794 -0.63330 0.34375 0.86814 0.91304 0.42875												0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000										0.59291 -0.69651 0.02039 0.14180 -0.63794 -0.63330 0.34375 0.86814 0.91304 0.42875	
0.44697 -0.80979 -0.06090 -0.49508 0.79350 -0.06352 -0.81698 -0.40769 -0.28581 0.14893												0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000										0.44697 -0.80979 -0.06090 -0.49508 0.79350 -0.06352 -0.81698 -0.40769 -0.28581 0.14893	
9 x 10												9 x 10										9 x 10	

Weight (W2)										Learning Rate		Delta for W2										Updated Weight (W2)	
-0.86836 -0.40570 -0.28847 -0.01552 -0.56012 0.17876 0.09944 0.43818 -0.55050										0.01		0.10293 -0.22838 -0.23113 0.09683 0.05046 0.08907 0.11104 0.00635 0.00283										-0.86939 -0.40342 -0.28616 -0.01649 -0.56062 0.17787 0.09833 0.43812 -0.55053	
-0.39517 0.88950 0.68481 -0.32862 0.21801 -0.85194 0.91851 0.66522 0.96813												-0.41977 0.93138 0.94256 -0.39487 -0.20580 -0.36322 -0.45285 -0.02589 -0.01155										-0.39097 0.88019 0.67539 -0.32470 0.22007 -0.84830 -0.91398 0.66548 0.96824	
-0.12834 0.68467 -0.82843 0.70860 -0.42024 0.05711 -0.21171 0.72769 -0.68958												0.29930 -0.66410 -0.67207 0.28155 0.14674 0.25898 0.32289 0.01846 0.00824										-0.13133 0.69132 -0.82171 0.70578 -0.42170 0.04542 -0.21494 0.72750 -0.68966	
0.88141 0.23787 0.01788 0.62214 0.93608 -0.44215 0.93589 0.58585 -0.02000												0.34257 -0.76009 -0.76921 0.32225 0.16795 0.29642 0.36956 0.02112 -0.00545										0.87799 0.24547 0.02557 0.61891 0.93440 -0.44512 0.93219 0.58564 -0.01995	
-0.47817 0.23975 0.81963 -0.73062 0.26039 -0.98893 0.62572 0.79596 -0.59899												-0.19811 0.43956 0.44483 -0.18635 -0.09713 -0.17142 -0.21372 -0.01222 -0.00545										-0.47619 0.23536 0.81518 -0.72877 0.26136 -0.98722 -0.62559 0.79606 -0.59894	
0.67935 0.72147 -0.11143 0.08312 -0.73809 0.22704 0.55970 0.92906 0.01680												-0.36374 0.80707 0.81675 -0.34216 -0.17833 -0.31474 -0.39241 -0.02243 -0.01001										0.68299 0.71340 -0.11960 0.08654 -0.73631 0.20319 0.56362 0.92928 0.01680	
-0.68984 0.90715 0.46368 -0.02233 -0.00475 -0.00444 0.42466 0.29852 0.75657												-0.32487 0.72081 0.72947 -0.30560 -0.15928 -0.28110 -0.35047 -0.02003 -0.00894										-0.68659 0.89994 0.45639 -0.01927 -0.00316 -0.00163 -0.42119 0.29872 0.75666	
-0.05432 0.39730 -0.01734 0.56336 -0.55088 0.46473 0.59556 -0.41257 -0.39543												0.29537 -0.65538 -0.66324 0.27785 0.14482 0.25558 0.31865 0.01821 0.00813										-0.05728 0.40385 -0.01071 -0.56617 -0.55233 0.46217 -0.59877 -0.41276 -0.39551	
-0.83774 0.05329 -0.15983 -0.16387 -0.67081 0.13953 -0.14870 0.70762 0.42471												-0.18617 0.41307 0.41803 -0.17513 -0.09128 -0.16109 -0.20084 -0.01148 -0.00512										-0.83588 0.04916 -0.16402 -0.16212 -0.66990 0.14114 -0.14669 0.70773 0.42476	
0.09638 -0.99509 -0.31341 0.88094 -0.40153 -0.63114 0.66011 0.18356 0.48690												-0.02861 0.06347 0.06423 -0.02691 -0.01402 -0.02475 -0.03086 -0.00176 -0.00079										0.09667 -0.99572 -0.31405 0.88121 -0.40139 -0.63089 -0.65982 0.18356 0.48691	
10 x 9												10 x 9										10 x 9	



Weaknesses of Word2Vec₈



- **Do not consider the morphology of words (subwords information) in the representation**

For example, we can deduce the relationship between “dog”, “dogs”, and “dogcatcher” by their spelling. The use of another method can handle that: FastText.

- **Do not separate some opposite word pairs**

For example, “good” and “bad” are usually located very close to each other in the vector space, which may limit the performance of word vectors in NLP tasks like sentiment analysis.

- **Takes a lot of computation time on huge text corpora**
- **New words are not handled**
- **Misspelled words are not handled**
- **Hard to get good representation of non-common words**

Agenda



1. Customer Journey restitution
2. KPI definition and Customer Journey
3. Steering Committee
4. Back to TF-IDF approach
5. Word embedding definition
6. Word embedding approaches
 - A. Latent Semantic Indexing (LSI) technique
 - B. An advanced embedding method: Word2Vec
 - C. Opening on other techniques: FastText**
7. Hands-on session
8. Summary of the session



Opening on other techniques: FastText



FastText principle

Improve word representation by using character level (n-gram) information.
Incorporate information about structure by representing words as a bag of character n-grams.

Long n-grams (n=6) are good to capture semantic information whereas shorter n-grams (n=3) are good to capture syntactic information





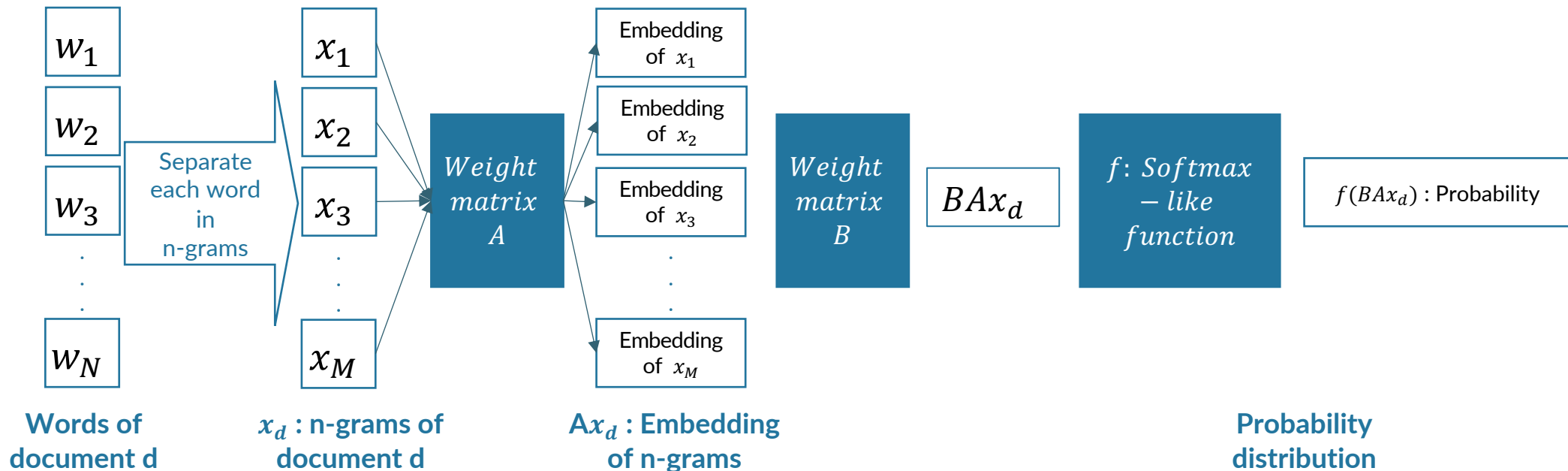
Opening on other techniques: FastText



FastText is an extension to the skipgram model in which words are represented as sum of characters n-grams⁹

Preprocessing

- A word is a bag of character. We **split a word in sequence of characters of n-grams** (bigrams/trigrams)
- G : set of all n-grams; G_w is the set of n-grams appearing in the word w
- Get the vector representation z_g of each n-gram as in the skipgram model before the softmax activate



Agenda



1. Customer Journey restitution
2. KPI definition and Customer Journey
3. Steering Committee
4. Back to TF-IDF approach
5. Word embedding definition
6. Word embedding approaches
 - A. Latent Semantic Indexing (LSI) technique
 - B. An advanced embedding method: Word2Vec
 - C. Opening on other techniques: FastText
- 7. Hands-on session**
8. Summary of the session



Hands-on 2



15'

Let's see Word2Vec applied to your data during this practical session.



Agenda



1. Customer Journey restitution
2. KPI definition and Customer Journey
3. Steering Committee
4. Back to TF-IDF approach
5. Word embedding definition
6. Word embedding approaches
 - A. Latent Semantic Indexing (LSI) technique
 - B. An advanced embedding method: Word2Vec
 - C. Opening on other techniques: FastText
7. Hands-on session
- 8. Summary of the session**



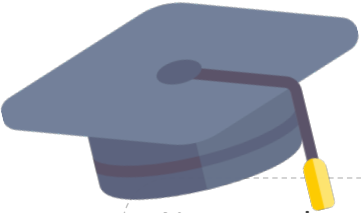
Summary of the session – To remember



- An embedding is a representation in which a word is associated **to a vector of numeric**
- We can perform **calculations and operations** on embedding matrices
- **LSI** is a technique of dimension reduction performed with **SVD for text meaning analysis**
- **Word2Vec** is an embedding method which embark a neural networks with **context and target word notions**
- **Cbow (context → target)** and **Skipgram (target → context)** are various ways of applying Word2Vec
- **FastText** is an advanced form of Skigram model in which characters **n-grams** of a word are considered rather than the word itself. A word embedding vector is the sum/average of its inner characters embedding vectors



Work for next sessions



Business

Next week you'll make a restitution about what you've seen until now.

- You are working in a consulting firm
- We are your clients
- According to what is mentionned in **Slide 10**

Thank you to send us your presentation
by Sunday 12th evening
to naomi.serfaty@capgemini.com and
sami.mhirech@capgemini.com

As a reminder, this presentation we'll be evaluated and be 20% of your total final mark.

Data-Science

To practice what we learnt today, for next session, you'll have to :

- Apply Word2Vec on your scraped data
- Carry out the same analysis using the gensim package
- Using PCA or TSNE, plot the embedding matrix on a two-dimensional figure.
- Find the most similar words for each review
- Using a similarity function, analyze the similarity between reviews.

We expect you to send your notebook file by **by Sunday 12th evening**
to cadmos.kahale-abdou@capgemini.com and
mayssae.zidani@frog.co

If you have any questions, feel free to contact us through the slack channel or by email.



References



- [1] Goldberg, Yoav. Neural network methods for natural language processing. Synthesis Lectures on Human Language Technologies, 2017, vol. 10, no 1, p. 1-309.
- [2] Thomas Landauer and Susan Dumais (2008) Latent semantic analysis. Scholarpedia, 3(11):4356
- [3] Mikolov, Yih, Zweig, 2013, Linguistic Regularities in Continuous Space Word Representations
- [4] Dhruvil Karani 2018, [Introduction to word embedding and word2vec](#)
- [5] [Ronxin Demo](#)
- [6] Mikolov, Chen, Corrado, Dean 2013a, Efficient estimation of Words Representations in Vector Space
- [7] Xin Rong 2016, Word2Vec Parameter Learning Explained
- [8] Derek Chai 2018, [An implementation guide to Word2Vec using Numpy and Google Sheets](#)
- [9] Bojanowski, Grave, Joulin, Mikolov 2017, Enriching word vectors with subword information



Course evaluation



Did you like that third course ? It's time to share your feedbacks !

Chapter 3 - X-HEC NLP Bootcamp
2023





Thank you for your attention

See you next week

GOODBYE !

February 6th, 2023