Université Mohammed V de Rabat

École nationale supérieure d'informatique et d'analyse des systèmes de Rabat

1

Word2Vec et GloVe en NLP

Introduction

Abdelkarim AGOUJIL

1ère Année Cycle Doctorat

2024/2025

Plan

- Introduction
- Représentations vectorielles
- Présentation du modèle Word2Vec
  - Continuous Bag of Words (CBOW)
  - Skip-Gram
- Présentation du modèle GloVe
- Différence entre Word2Vec et GloVe
- Exemple avec Word2Vec (Skip-Gram, CBOW et GloVe avec NLTK)
- Comparaison des résultats
- Conclusion

Introduction : NLP et Représentation des Mots

Généralement, on peut distinguer deux aspects essentiels à tout problème de ***NLP*** :

- **Partie linguistique :** qui consiste à prétraiter et transformer les informations en entrée en un jeu de données exploitable.

- **Partie apprentissage automatique:** qui porte sur l'application de **modèles de *Machine Learning*** ou **Deep Learning** à ce jeu de données.

Définitions et Pré-processing

- La première étape du traitement des données textuelles est de :
  - Formater les textes pour se rapprocher des **données tabulaires (structurer)**
  - Rendre les textes simple à traiter pour les modèles **ML et DL (normaliser et/ou supprimer des mots)**

Définitions :

Il est important de bien identifier 4 termes incontournables :

- **Corpus :** un ensemble de documents (des textes dans notre cas), regroupés dans une optique ou dans une thématique précise.
- **Document** : la notion de document fait référence à un texte appartenant au corpus, mais indépendant des autres textes. Il peut être constitué d'une ou plusieurs phrases, un ou plusieurs paragraphes.
- **Token** : le terme token désigne généralement un mot et/ou un élément de ponctuation. La phrase "Hello World!" comprend donc 3 tokens.
- **Vocabulaire** : il s'agit de l'ensemble des tokens distincts présents dans l'ensemble du corpus.

Tokenization

Définition :

- **Tokénisation** est le processus qui divise le texte brut (par exemple, une phrase ou un document) en une séquence de **tokens**, tels que des mots ou des sous-mots.
- elle est considéré comme la première étape d'un **pipeline** de traitement **NLP**.
- **Tokens** sont des séquences de texte récurrentes qui sont traitées comme des unités atomiques dans un traitement ultérieur. Il peut s'agir de mots, d'unités de sous-mots appelés morphèmes (par exemple, des préfixes comme "un-" ou des suffixes comme "-ing" en anglais), ou même de caractères individuels.

- Texte : séquence de mots
- Mot : séquence logique de caractères
- **Question :** comment trouver les limites d'un mot ?
- **Réponse :** En français / arabe/ anglais, on peut séparer les mots par les espaces et les ponctuations

Tokenization : définition

- **Tokenization : processus qui sépare une séquence (texte) en une liste de tokens (mots)**

Normalisation : Stemming et lemmatisation

- **Stemming (Racinisation)** : garder la racine d'un terme (souvent, couper à partir d'un caractère)
- **Exemple :** traitera, traitait, traitement, traite…. ⟹ Trait
- **Exemple :** دراسات, دراسة, دَرَسَ, دَرَّاسَة…. ⟹ درس

- **Lemmatisation :** processus qui sépare une séquence (texte) en une liste de tokens (mots)
- **Exemple :** traitera, traitait, traitant, traite…. ⟹ traiter

  traitements, traitement ⟹ traitement
- **Exemple :** دراسات, دراسة, دَرَّاسَة, دَرَسَ…. ⟹ دراسة

Stop-words

- **Les stop-words :** ensemble de mots fréquemment utilisés dans une langue et qui n'apportent pas de signification importante
- **Exemple :** "لا", "و", "عن", "أن", "إلى", "ما", "من", "هي", "هو", "على", "في", "هذا"
- **Objectif : supprimer ces stop-words**

Vectorisation des données textuelles

- **Transformer des documents textuels en données tabulaires (partir des données textuelles tokenizées et normalisées)**

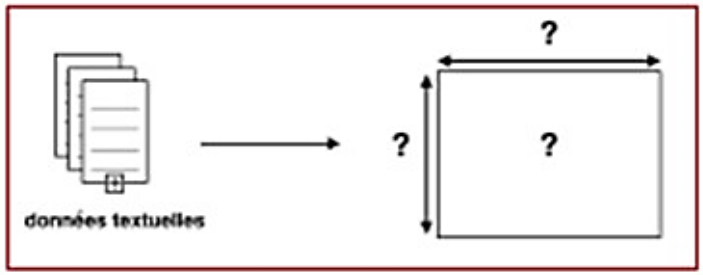
- **Données textuelles => données séquentielles (tenir compte de l'ordre des mots)**

Données textuelles en données tabulaires

Rappel: Processus d'apprentissage pour des **données tabulaires**

Processus d'apprentissage pour des **données textuelles ou speech**.

Problèmes



Approches : TF-IDF et Word Embedding

Vectorisation des textes : principe

corpus → définir le vocabulaire → one-hot encoding

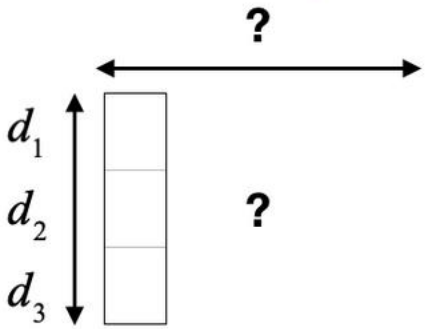
d_1	trouver bonne assurance
d_2	contrat satisfaisant
d_3	changement contrat assurance

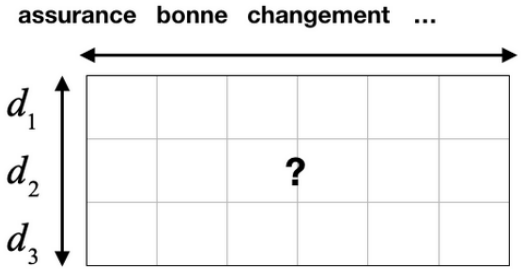
```
V = {
    'assurance'   : 1,
    'bonne'       : 2,
    'changement'  : 3,
    'contrat'     : 4,
    'satisfaisant' : 5,
    'trouver'     : 6
    }
```

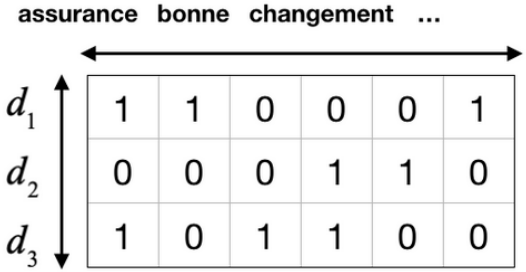
assurance contrat

1	...	0
0		0
0		0
0		1
0		0
0		0

méthode bag-of-words







Vectorisation des textes : approche bag-of-words

- **Exemple :**
- Un modèle **BoW** (Bag of Words) est l'un des algorithmes d'extraction de caractéristiques les plus simples que vous rencontrerez.
- Le nom **"bag-of-words"** vient du fait que l'algorithme cherche simplement le **nombre d'occurrences** d'un token **t** dans le document **d**.
- L'ordre ou le contexte des mots n'est pas analysé ici.
- **Exemple : méthode de comptage**

trouver bonne assurance
contrat satisfaisant
changement contrat assurance

trouver	contrat	assurance	…
1	0	1	…
0	1	0	…
0	1	1	…

- **Problème: pas d'ordre considéré entre les mots**

Approche n-grammes

- **n-grammes** est une approche qui consiste à diviser un texte en séquences contiguës de n mots consécutifs.
- Un N-gramme peut être un mot unique (**uni-gramme)** ou une séquence de plusieurs mots **(bi-gramme, tri-gramme**, etc.).
- Les avantages des N-grammes incluent :

  Capturer la structure locale d'un texte.

  Prise en compte du contexte entourant un mot ou une séquence de mots.

  Amélioration de la compréhension sémantique et syntaxique d'un texte.

- **Exemple : d'Utilisation des Trigrammes (3-grammes)**

On considère le texte suivant :

- « Le traitement automatique du langage naturel est une branche de l'intelligence artificielle. »
- **Solution :**
- « Le traitement automatique » , « traitement automatique du » , « automatique du langage »

  « du langage naturel » , « langage naturel est » , « naturel est une » , « est une branche » ,

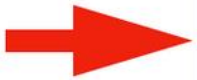
  « une branche de », « branche de l'intelligence », « de l'intelligence artificielle »

Approche n-grammes

- Les trigrammes capturent des informations sur la structure locale du texte et permettent de considérer des contextes de trois mots à la fois.

- **Exemple : méthode de comptage (1,2)-grammes**

trouver bonne assurance
contrat satisfaisant
changement contrat assurance



trouver	assurance	contrat assurance	…
1	1	0	…
0	0	0	…
0	1	1	…

- **Problème: trop de variables**
- **Solution : supprimer les stop-words et quelques n-grammes (très hautes et très basses fréquences)**

Vectorisation des textes : approche TF-IDF

- **Term Frequency : nombre d'occurrences d'un token/n-grams t dans le document d**

$$tf(t,d) = f_{t,d}$$

- **Variantes :** $\frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}$ **ou** $\mathbb{1}(t \in d)$ **ou** $(1 + \log f_{t,d})$
- **Inverse Document Frequency : mesure l'importance du token/n-grams t dans l'ensemble du corpus**

$$idf(t, D) = log_2(\frac{|D|}{|\{d|t \in d\}|})$$

- **Term Frequency- Inverse Document Frequency (TF-IDF) :**

$$tfidf(t, d, D) = tf(t, d).idf(t, D)$$

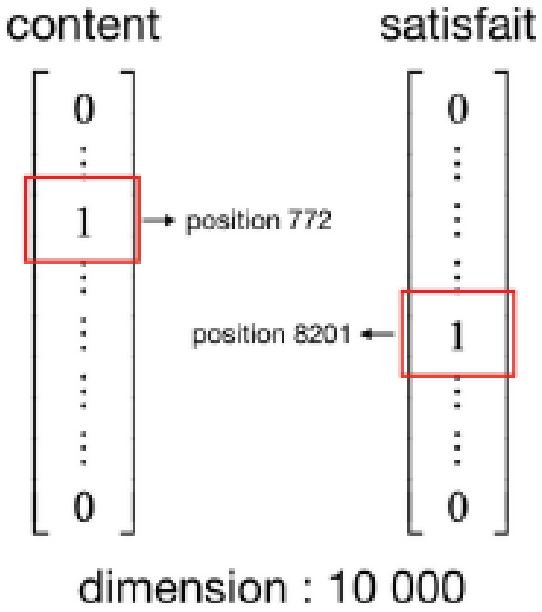
Tfidf mesure le poids de token t dans le document d.

Représentation vectorielle des mots : one-hot-encoding ?

Objectif:

- **Texte à apprendre :** mon client est content de son assurance → 🙂
- **Texte à prédire :** l'assuré est satisfait de notre contrat → ?

V = {'a' : 1, …
'assurance' : 59, …
'assuré' : 62, …
'client' : 698, …
'content' : 772, …
'contrat' : 899, …
'satisfait' : 8 201, …
'zoo' : 9 999, …
<UNK> : 10 000 }



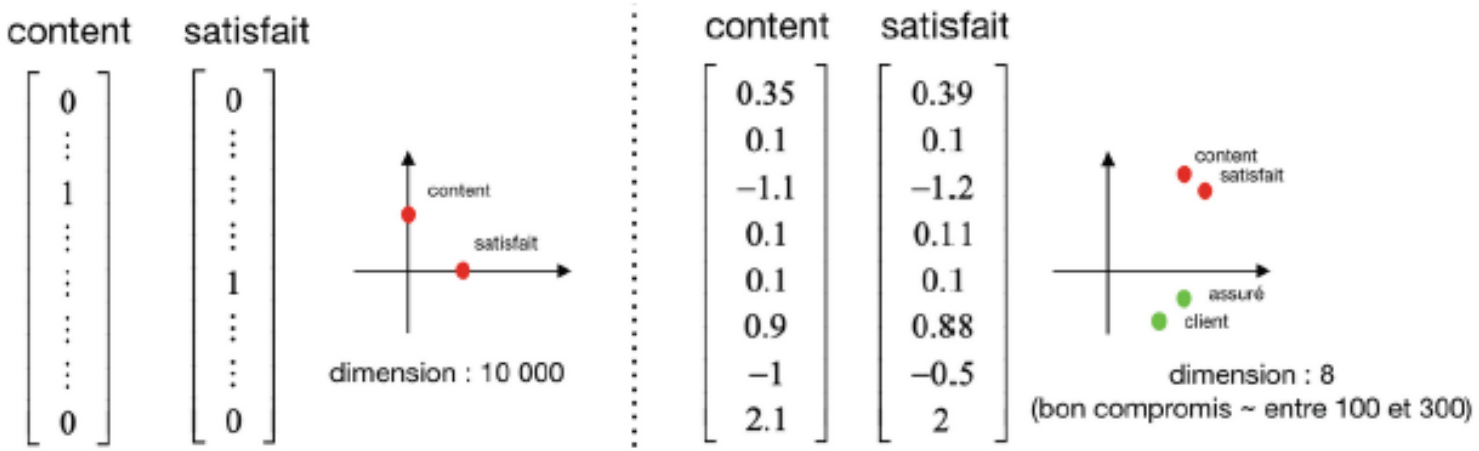
Approches naïves : One-hot-encoding or TF-IDF

Problèmes : vecteurs creux de grandes dimensions sauvegardent des mots sans prendre en compte le contexte :

vect (« content ») ≠ **vect** (« satisfait »)

Word Embedding

- **Word embedding** (plongement de mot en français) : vectorisation des mots de sorte que les mots apparaissant dans des contextes similaires ont des significations apparentées



One-hot-encoding or TF-IDF

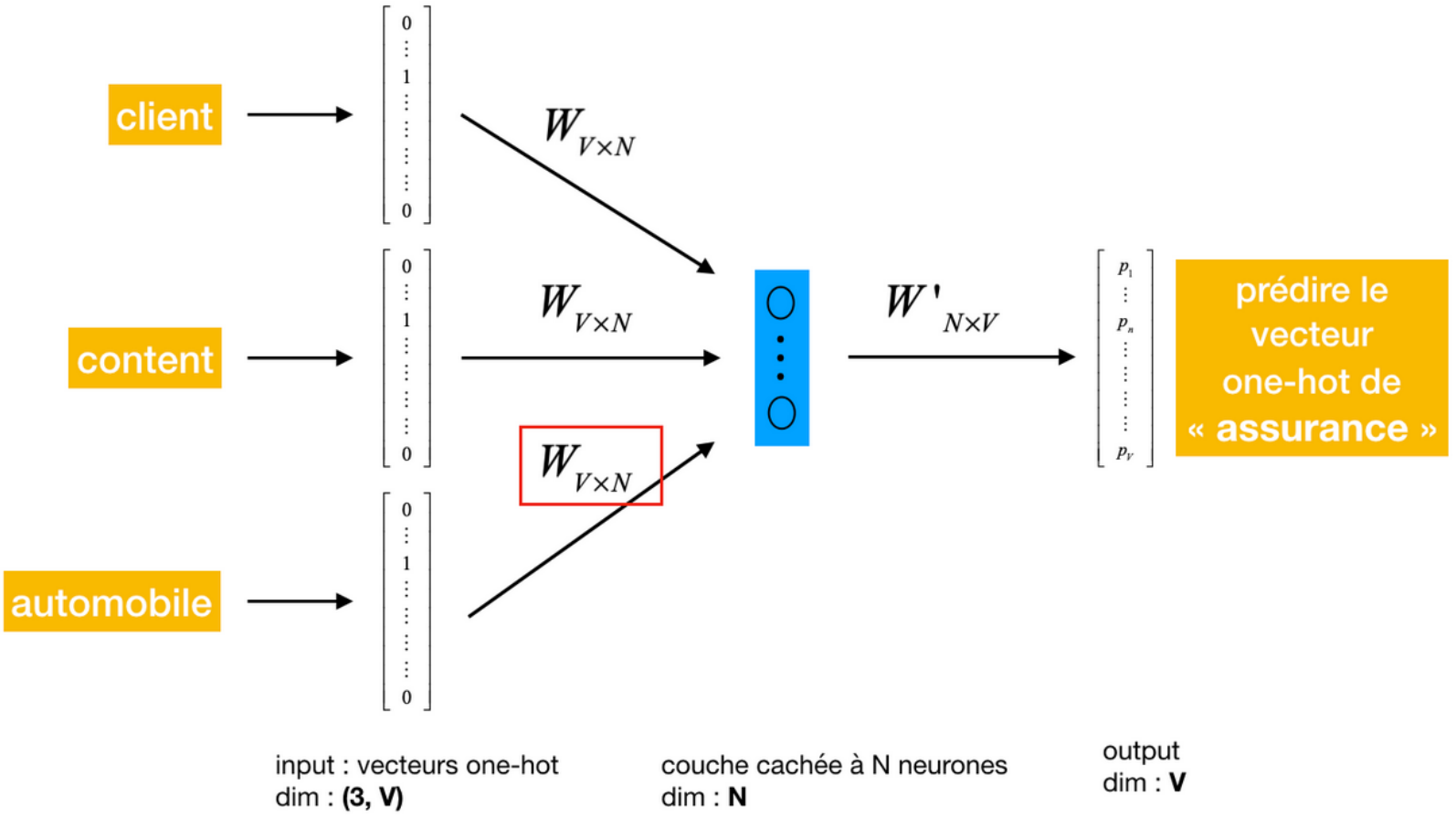
Word embedding

- Vecteurs **denses** de plus **petites dimensions**

Mot1 ≈ mot2 ⟹ **word_embedding(mot1) ≈ word_embedding(mot2)**

Word2Vec : modèle CBOW

Architecture CBOW : prédit un mot cible à partir de son contexte.



CBOW : pseudo-code

- Fonction **Entraîner_CBOW**(corpus, vector_size, window, learning_rate, nombre_iterations)

Initialisation des vecteurs de mots

- Initialiser les vecteurs de mots aléatoirement
- Pour iter de 1 à nombre_iterations
  - Pour chaque phrase dans le corpus
    - Pour chaque mot de la phrase

Récupérer les mots de contexte

✓ context_words = Mots de la fenêtre autour du mot cible

Calcul de la moyenne des vecteurs de contexte

✓ average_context_vector = Moyenne des vecteurs des mots de contexte

Prédiction du mot cible

✓ predicted_word_vector = Calculer la prédiction en utilisant le modèle CBOW

Calcul de l'erreur de prédiction

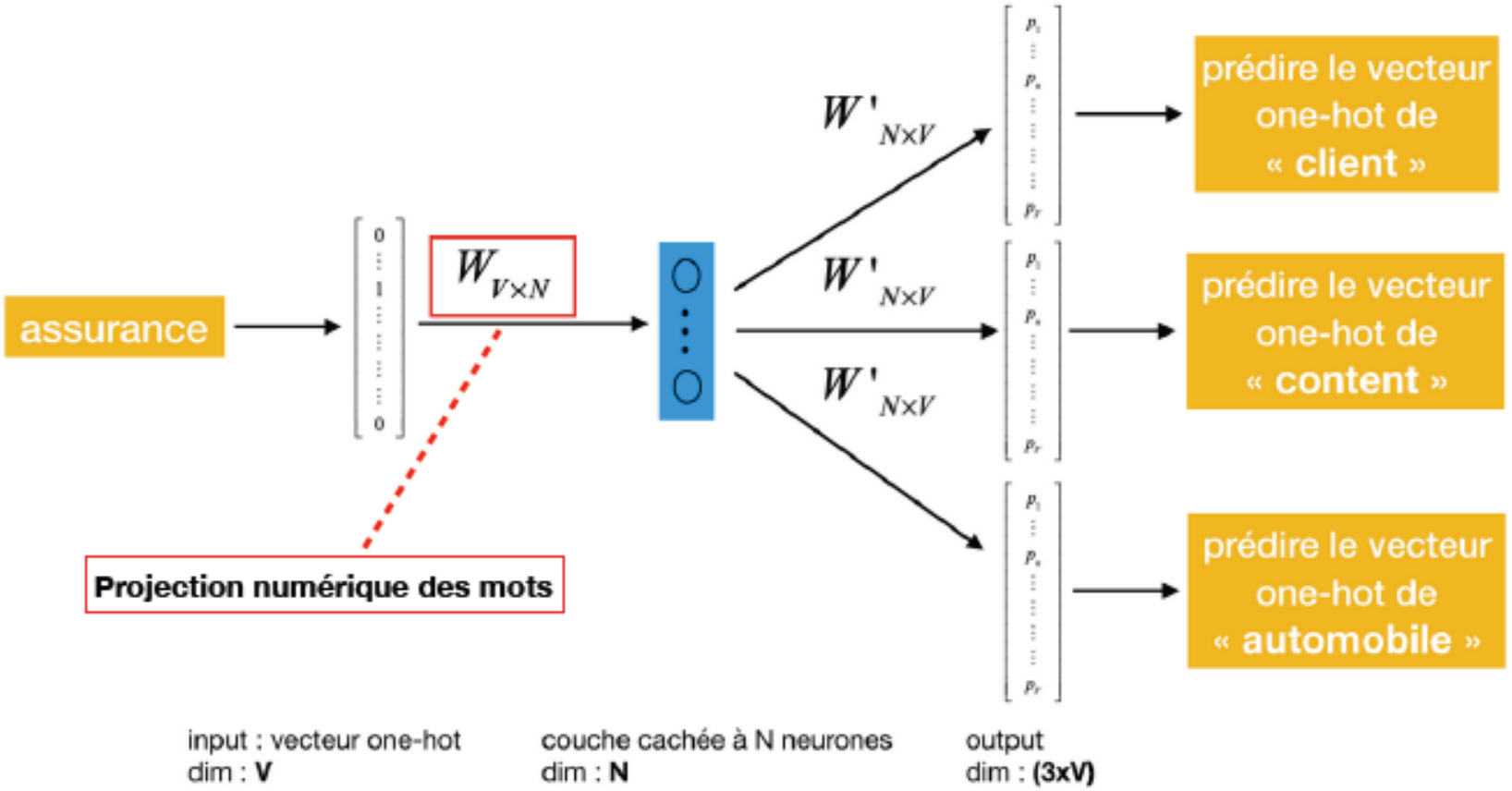
✓ error = Différence entre le vecteur réel du mot cible et la prédiction

Mise à jour des vecteurs de mots

✓ Mise à jour des vecteurs des mots de contexte et du mot cible en utilisant l'erreur **Fin Pour Fin Fonction**

Word2Vec : modèle Skip-Gram

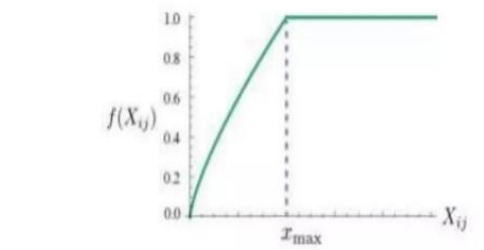
- **Architecture Skip-Gram :** prédit les mots contextuels à partir d'un mot cible.
- Utile pour des mots rares ou spécifiques.

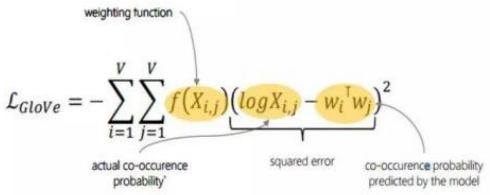


Modèle GloVe

- Basé sur des statistiques globales (co-occurrence des mots).
- Utilise une matrice de co-occurrence factorisée pour générer des vecteurs.
- Avantages : capture les relations globales et sémantiques.

Global statistics of co-occurrence probability

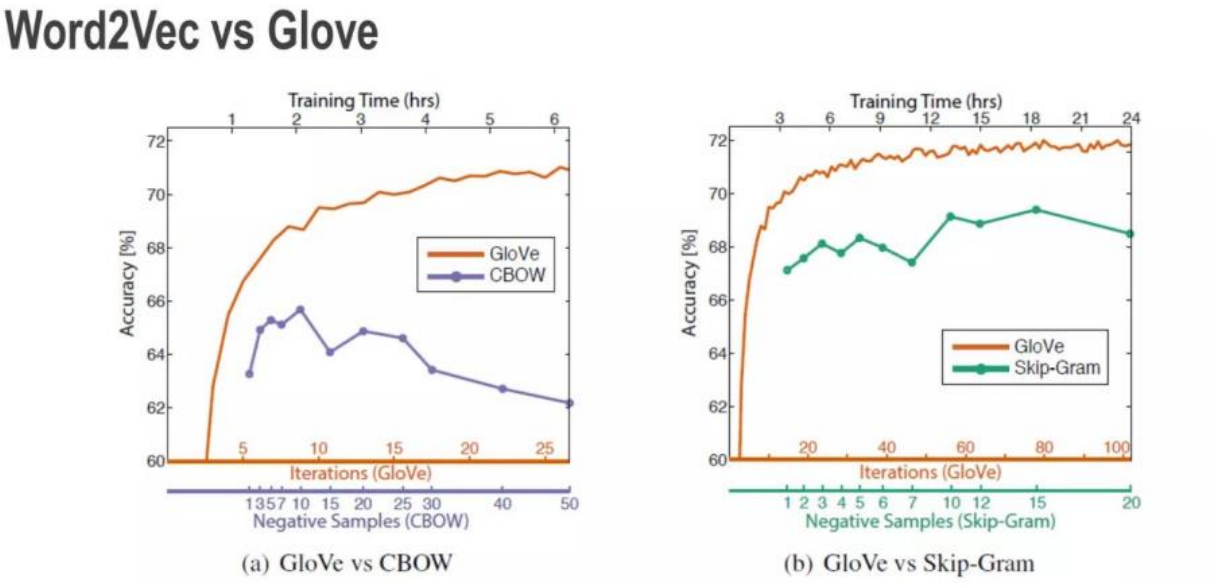




	w_0	w_1	w_2	...	w_j	...	$w_{\|V\|}$
w_0							
w_1							
w_2							
...							
w_i					X_{ij}		
...							
$w_{\|V\|}$							

Différences entre Word2Vec et GloVe

- Méthodologie : **Word2Vec** est prédictif (CBOW/Skip-Gram), tandis que **GloVe** est basé sur des statistiques globales.
- Taille des données : **GloVe** est plus efficace pour des grands corpus.
- Sémantique : **Word2Vec** excelle dans la capture des relations contextuelles proches, **GloVe** excelle dans les relations globales.



Word2Vec : Préparation de dataset

- Dataset = """ الذكاء الاصطناعي هو مجال يهتم بتطوير الأنظمة التي يمكنها أداء المهام التي تتطلب عادةً الذكاء البشري تتضمن هذه المهام مثل التعرف على الصور، معالجة اللغة الطبيعية، يعمل الذكاء الاصطناعي على تحسين الكفاءة وجودة العمل في مختلف الصناعات والتنبؤ"""

Exemple avec CBOW

Le Modèle CBOW

```
import numpy as np
import re
import nltk
from nltk.tokenize import word_tokenize
nltk.download('punkt')
```

[185] Python

```
# (['الأنظمة', 'بتطوير', 'مجال', 'الاصطناعي'], 'يهتم')
context = ['الأنظمة', 'بتطوير', 'مجال', 'الاصطناعي']
context_reordered = context[::-1]
predict(context_reordered)
```

[219] Python

... 'يهتم' ← La prédiction

Exemple avec Skip-Gram

Le Modèle CSkip-Gram

```
import numpy as np
import re
import nltk
from nltk.tokenize import word_tokenize
nltk.download('punkt')
```

[118] Python

```
input_word = 'الذكاء'
input_ind = word_to_id[input_word]
output_words = [id_to_word[output_ind] for output_ind in top_sorted_inds[::-1, input_ind]]
print("{}'s skip-grams: {}".format(input_word, output_words))
```

[150] Python

... الذكاء's skip-grams: ['عادة', 'معالجة', 'البشري', 'التعرف'] ← La prédiction

Exemple avec GloVe

Le Modèle GloVe

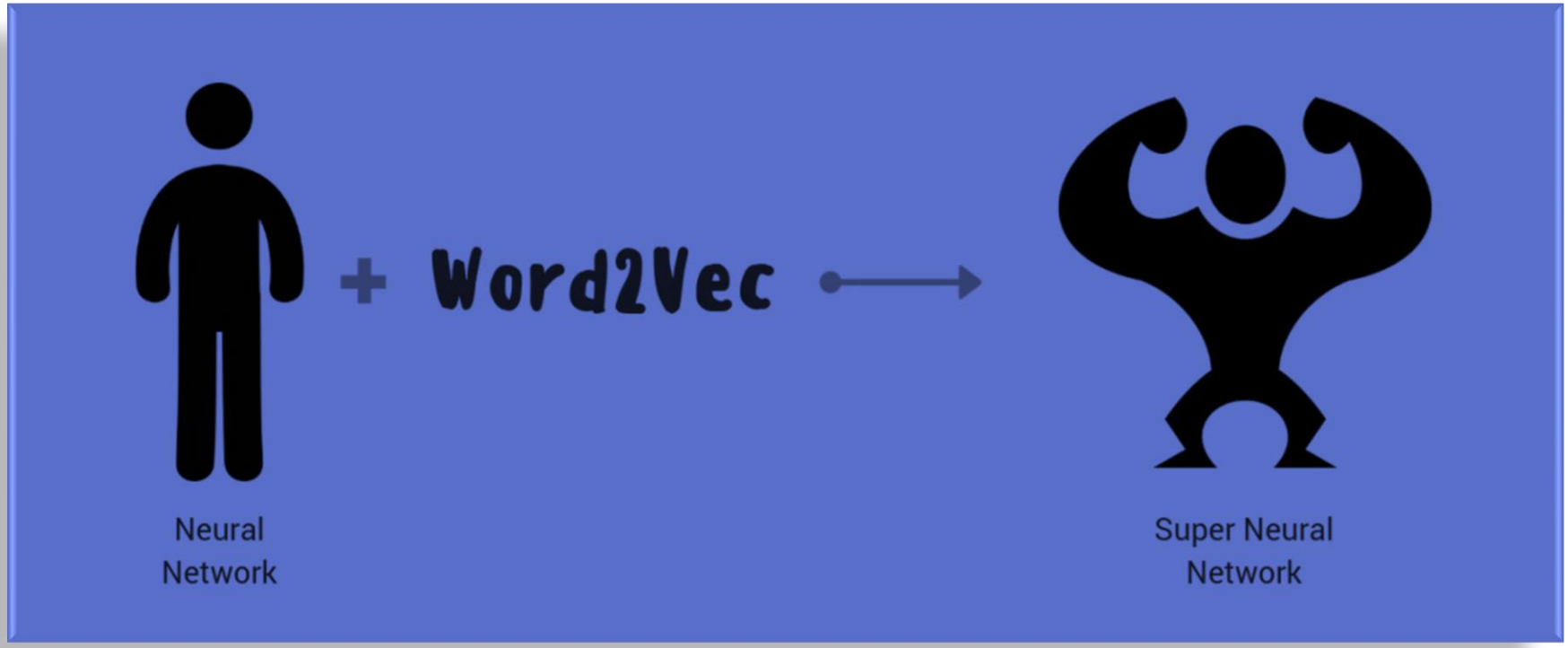
```
import numpy as np
import re
from nltk.tokenize import word_tokenize
from collections import Counter
import nltk
```

```
Test de similarité entre 'الذكاء' et 'الاصطناعي':
Similarité cosinus entre 'الذكاء' et 'الاصطناعي' :0.3590775039820202

Test d'analogie 'يعمل' + 'الاصطناعي' - 'الذكاء':
Les mots les plus similaires à l'analogie 'يعمل + الاصطناعي - الذكاء':
Mot: والتنبؤ, Similarité: 0.7383184306079075
Mot: الكفاءة, Similarité: 0.7197392514055865
Mot: الذكاء, Similarité: 0.7050008422013012
Mot: هذه, Similarité: 0.6789067174990286
Mot: معالجة, Similarité: 0.6575608758963057
```

+ Code + Markdown

Conclusion



Reference

- Glove: Global Vectors for Word Representation
- January 2014 DOI: 10.3115/v1/D14-1162
- Conference: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)
- Lien: https://www.researchgate.net/publication/284576917_Glove_Global_Vectors_for_Word_Representation

- Efficient Estimation of Word Representations in Vector Space
- Subjects: Computation and Language (cs.CL)
- Cite as: arXiv:1301.3781 [cs.CL] (or arXiv:1301.3781v3 [cs.CL] for this version) link : https://arxiv.org/abs/1301.3781
- **A Dummy's Guide to Word2Vec :** https://medium.com/@manansuri/a-dummys-guide-to-word2vec-456444f3c673