IA et Nous

TALN

Traitement Automatique Du Langage Naturel

NLP Natural Language Processing

Plan

1. *TALN c'est quoi ?*
2. *Brève histoire*
3. *Séquences*
4. *Prétraitements*
    - *Codage*
    - *Tokenisation*
    - *Word Embedding*
5. *Réseaux de neurones récurrents RNN et LSTM*
6. *Encodeur Décodeur - Seq2seq*
7. *Mécanisme d'attention*
8. *Transformer*
9. *Modèles de langage*

IA et Nous
TALN
1ère partie

TALN traitement automatique du Langage Naturel

NLP Natural Language Processing

Objectif

Comprendre le langage ***humain dit « naturel »***
– qu'il soit écrit ou oral –

afin qu'une ***machine*** *puisse répondre de façon pertinente à une requête basée sur des mots-clés.*

1. TALN

Définitions

Traitement

Manipulation d'un objet d'entrée aboutissant à la modification de cet objet en un objet de sortie.

Vise

- *à transformer des données linguistiques existantes : correction, extraction d'information, résumé, traduction...*
- *à en construire : génération de textes à partir d'informations.*

Automatique

Les données linguistiques doivent être appréhendées de façon totalement ***explicite, cohérente et opératoire*** *- d'où le recours à divers types de* ***formalismes et de techniques informatiques***

TALN

Comprend

- ***La parole***
  - *Reconnaissance vocale :* transcription automatique de l'oral vers l'écrit
  - *Synthèse vocale :* l'émission d'une suite de sons correspondant à des phrases fournies à l'écrit
- ***L'écrit***

Objet de cet exposé

TALN

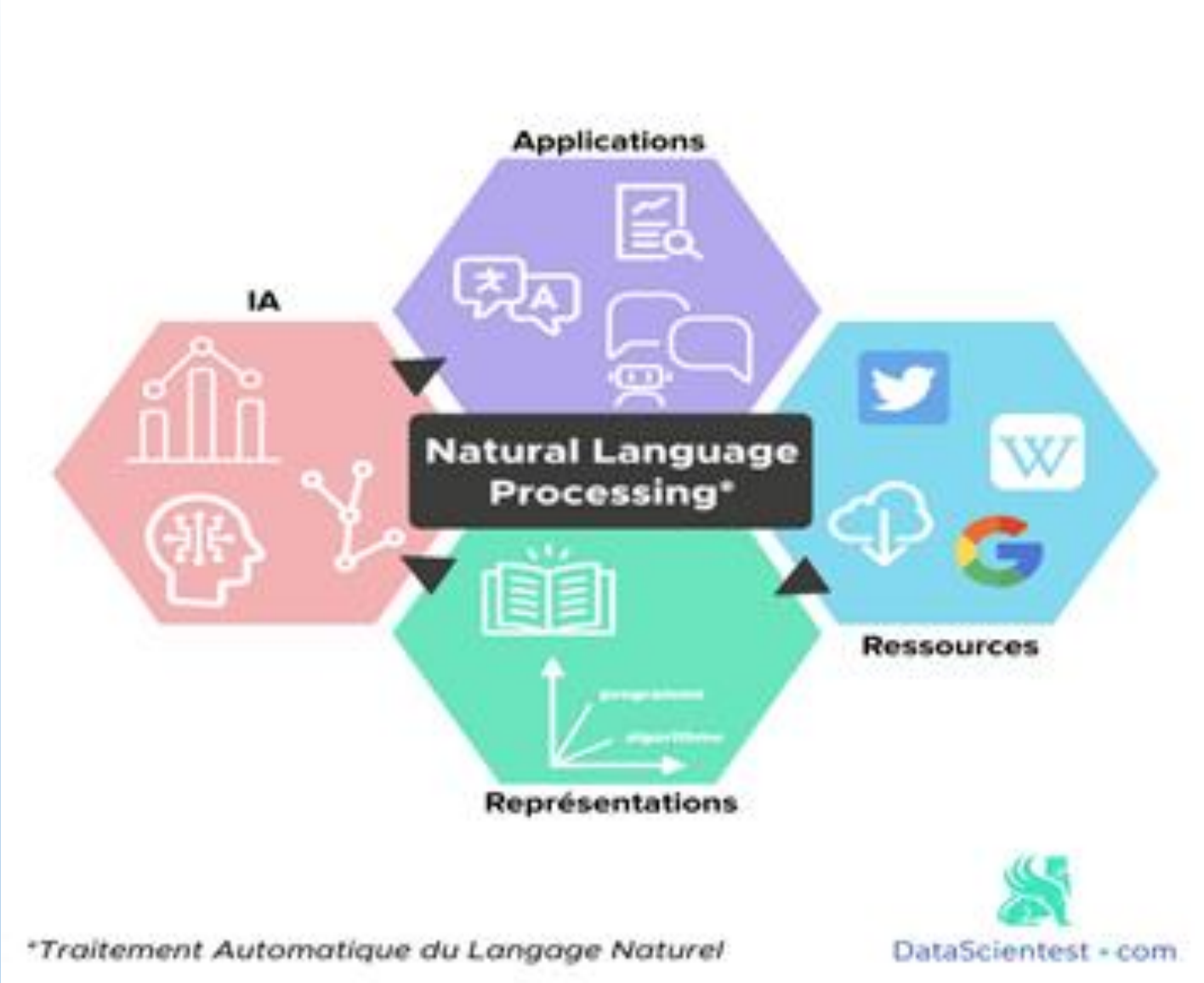
Approche symbolique : *Repose sur les règles de langage communément adoptées dans une langue donnée, qui sont définies et enregistrées par des spécialistes du lexique afin que le système informatique puisse les suivre.*

Approche statistique : *Basée sur des illustrations notables et récurrentes de manifestations linguistiques.*

Approche connexionniste : *Mélange de l'approche symbolique et de l'approche statistique.*

- *Part des règles du langage communément admises et les adapte à des applications particulières à partir des données obtenues par **inférence statistique**.*
- *Utilisation des réseaux de neurones et du Deep Learning*

NLP TALN

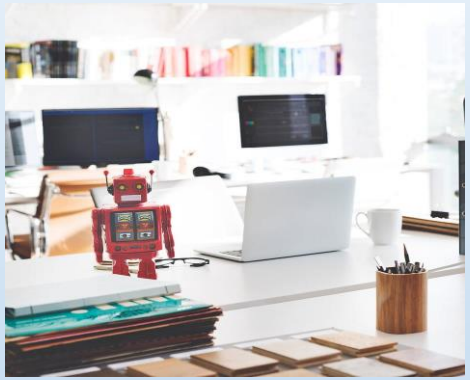

Applications
IA
Natural Language Processing*
Ressources
Représentations
*Traitement Automatique du Langage Naturel
DataScientest.com

TALN

Le traitement automatique du langage naturel est déjà partout

Historiquement tourné vers des applications de traduction automatique,
Aujourd'hui :

- *Extraction d'informations Fouille de texte, recherche d'information, Émotions, sentiments, mots clé, résumés, Moteurs de recherche*
- *Assistants personnels vocaux (Siri, Alexa, Cortana, Google Assistant,…)*
- *Assistants virtuels (chatbot)*
- *Génération automatique de textes (GAT) - Exemples bulletins météo, rapports automatisés, ….*
- *Correction orthographique,….*



➔ *Grands modèles de langage LLM :*

GPT, LlaMa, PALM, HuggingFace, Bloom, Mistral, Baidu etc.

2. Brève histoire duTALN

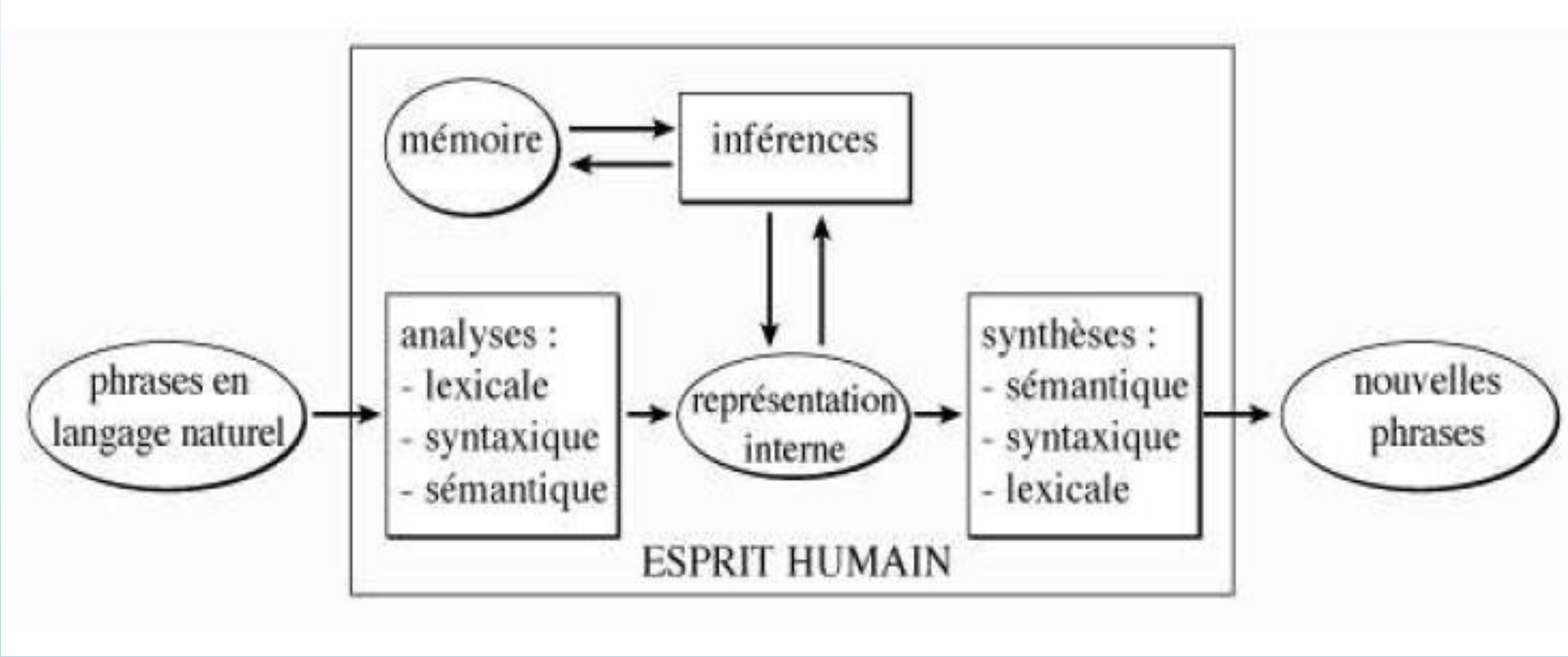
Jusqu'aux années 1980, les recherches en TALN étaient concentrées sur la traduction automatique dans un contexte politique de Guerre Froide où ces travaux visaient la traduction instantanée des « écoutes » des adversaires.

Dès 1954, « Le cerveau » a été le premier programme informatique capable de traduire une soixantaine de phrases du russe vers l'anglais dans le cadre de l'expérience Georgetown-IBM.

Les années 60 ont été témoins de l'apparition des premiers chatbots de l'histoire, avec par exemple ELIZA en 1964 qui était capable de simuler une psychothérapie (reformulation des phrases du « patient » et questions contextuelles).

Brève histoire duTALN

L'approche du TALN était alors dite « symbolique » *car le savoir linguistique était codé sous forme de grammaire et de bases de données lexicales développées manuellement*



Brève histoire duTALN

A partir des années 1980**, l'approche est devenue **«statistique»
*c'est-à-dire basée sur des **algorithmes** et non plus des **listes de mots**.*

*Pour cela, le texte est transformé en **input numérique***
*Représentation sous forme de **vecteurs***

*Utilisation des techniques du **Machine Learning***

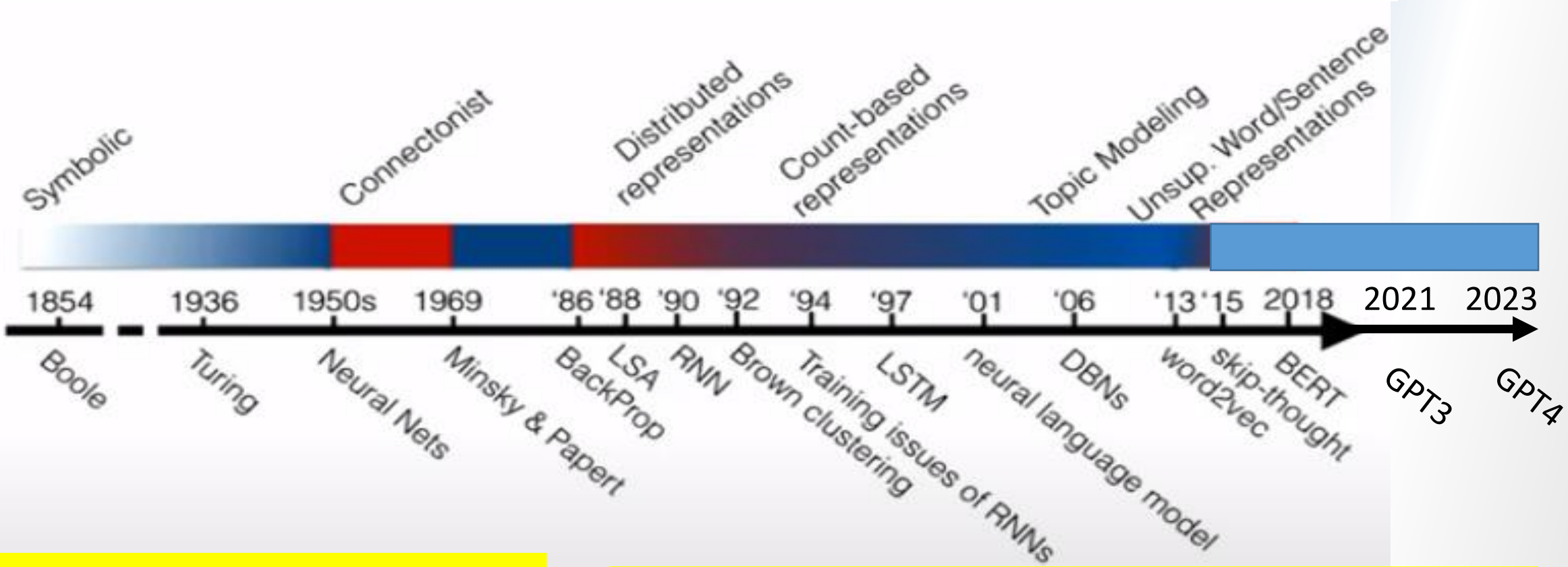
Brève histoire duTALN

Les années 1990 *appartiennent au* ***Deep Learning.***

Les travaux de ***Yann LeCun*** *sur l'apprentissage profond au sein des laboratoires Bells ont donné naissance au premier système basé sur les réseaux de neurones convolutifs entrainés à la reconnaissance de la parole puis à celle des caractères.*

Si le premier champ d'application a été la lecture des chèques bancaires, ***Yann LeCun*** *et ses collaborateurs ont tout simplement initié le fonctionnement des assistants vocaux comme Siri d'Apple.*

Brève histoire du TALN



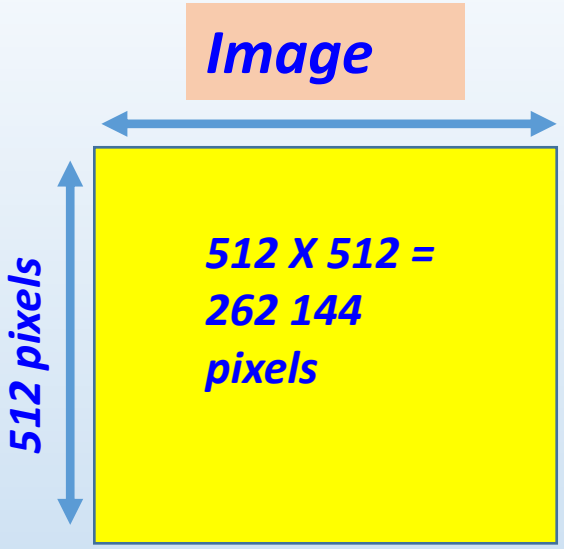
RNN Recurrent Neural Network
LSTM Long Short Term Memory

BERT Modèle de langage à base de transformers Google
GPT3 Modèle de langage à 175 milliards de paramètres Open AI

Source : Unsupervised Deep Learning - https://media.neurips.cc/Conferences/NIPS2018/Slides/graves-deeplearning2.pdf

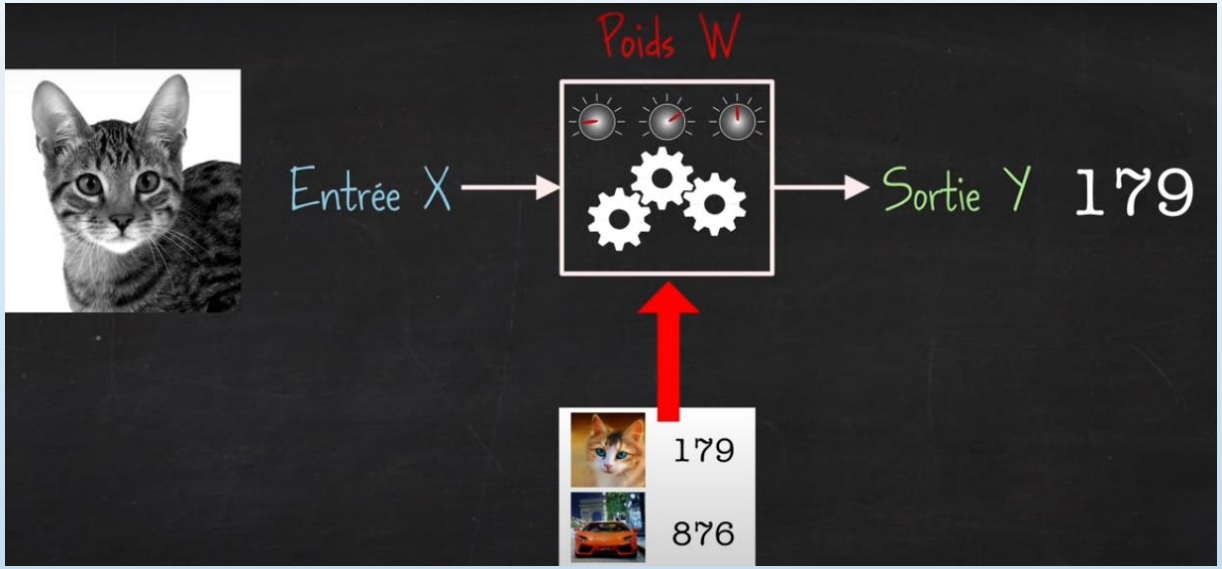
3. Séquences vs Images

Images



Si entrée diff de 512X512
Mise au format

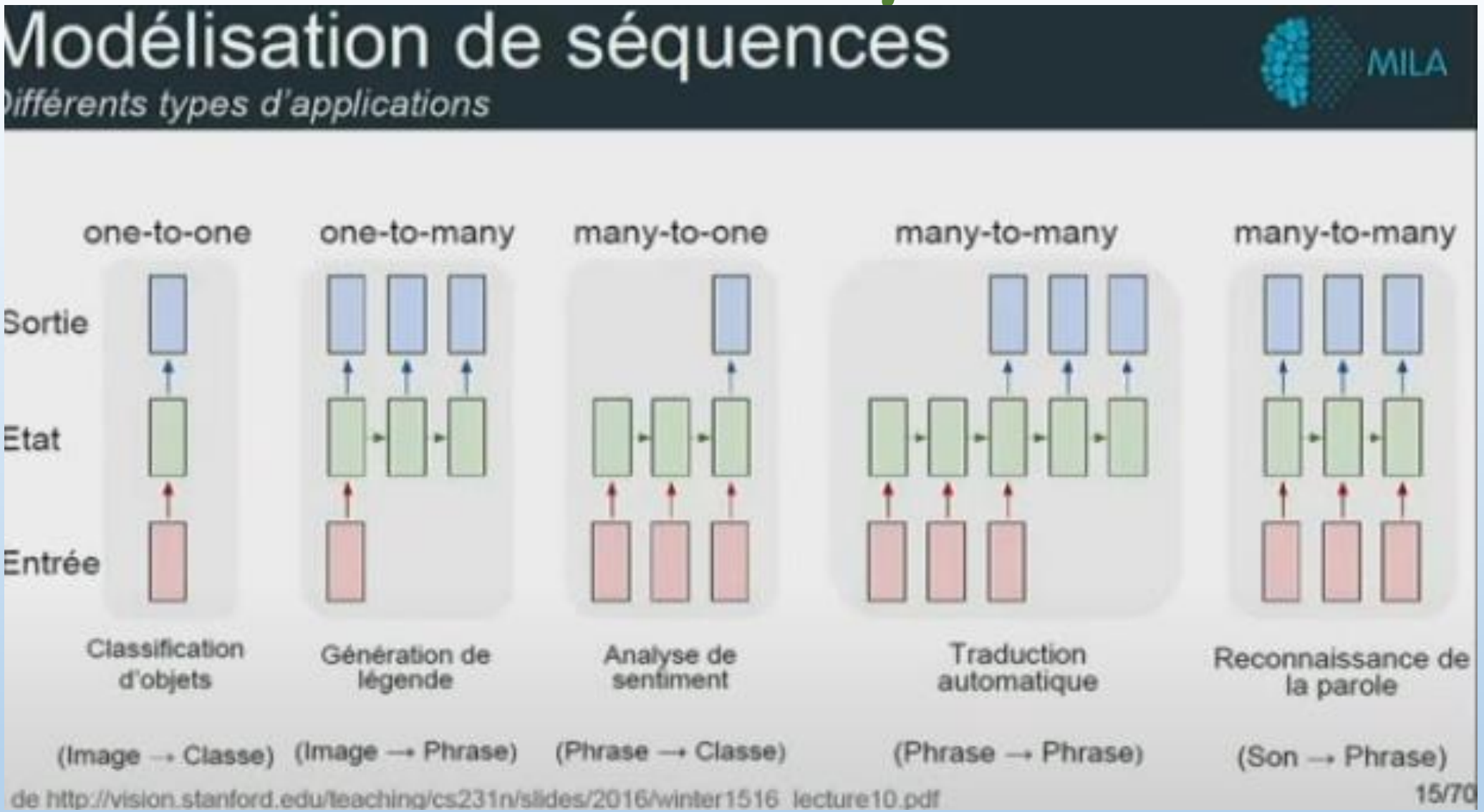
Données fixes



0,1
,,,
0,24
,,,,
0,94
,,,
0,01
,,,

Vecteur

Traitement de séquences



Langage naturel

Compositionnel

« La signification d'une expression complexe est définie par les significations des expressions la composant et par les règles employées pour les combiner » Wikipédia

Phrase est composée de N mots

Données séquentielles
Chaque phrase est de longueur différente
La connaissance de tous les mots de la phrase est nécessaire
La connaissance de la position des mots est importante

Exemple : analyse de sentiments ou émotions

« Moi qui pensait que ce serait mauvais, eh bien, je me suis trompé »

4. Prétraitements

Les algorithmes ne savent pas manipuler des mots

1. ***Codage des caractères***
2. ***Tokenisation***
3. ***Word Embedding***

4.1 Codage Caractères

1. ASCII

- *7 bits codent **128 caractères***
- *Exemple M se code 4D soit 77*
- *Insuffisant aujourd'hui : caractères spéciaux, emojis,…..*

2. Unicode

- *Unicode 15.0 comprend plus de **149 000 caractères**.*
- *Les 128 premiers caractères sont identiques à ceux de la table ASCII.*
- *Unicode implémenté dans normes : UTF-8 et UTF-16*
- *1 à 4 octets*
  - *UTF-8 : 4 octets, 0xF0 0x9F 0x90 0xB6*
  - *UTF-16 : 4 octets, 0xD83D 0xDC36*

4.2 Tokenisation

- *Unités atomiques*
- *Définies au niveau*
  - *des caractères : un ou plusieurs éléments*
  - *des mots : un ou plusieurs éléments*
  - *des phrases*

"I saw a girl with a telescope."

"I", "saw", "a", "girl", "with", "a", "telescope", "."

N-gramme

Un n-gramme est "une sous-séquence de n éléments construite à partir d'une séquence donnée." (Wikipédia).

Il s'agit d'une séquence de taille n, piochée dans une séquence de taille plus grande que n.

Exemples

- *de est un bigramme de* ***demain,***
- *mai un trigramme*
- *main un quadrigramme.*

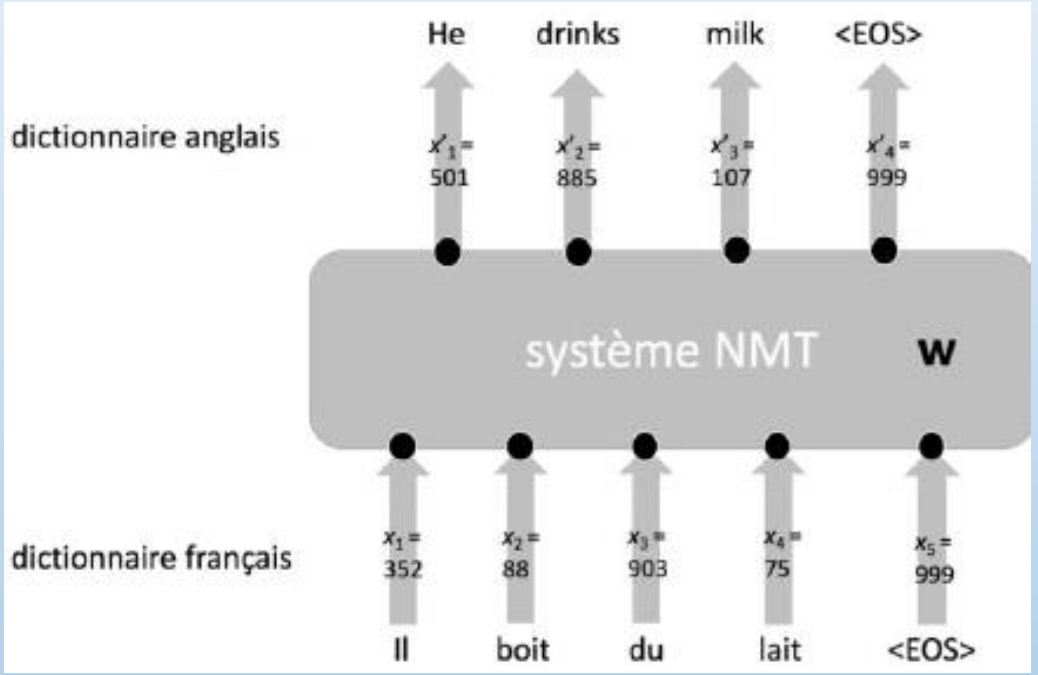
Tokenisation

Exemple

- *Chaque mot est numéroté* ***xi,***
- *une phrase de n mots est une suite* ***(x1, x2, … , xn)***

*n entiers xj (ou token- entité) avec 1< **xi** < 100 000*

.



Exemple de Tokenisation

Représentation des tokens

- Hypothèse: token = 1 mot
- Utilisation d'un vecteur binaire par token (one-hot vector)
  - Exemple: chat = [0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 ... 0]
- Similarité entre tokens:

  chat = [0 0 1 0 0 0 0 0 0 0 0 0 0 0 0...0]
  chien = [0 0 0 0 0 0 0 0 0 1 0 0 0 0...0]
  maison = [0 0 0 0 0 0 0 0 0 0 0 0 1 0...0]

- Tous les mots différents sont à une distance de 2.

EH2018-12 - Application : Traitement des langues naturelles - Arsène Fansi
Université Montréal Joshua Bengio

Exemple de Tokenisation

Représentation des phrases

- Bag of words
  - Exemple:
    Le chat est dans la maison avec le chien:
    [0 0 1 0 0 0 0 0 0 1 0 0 1 0...0]

    chat = [0 0 1 0 0 0 0 0 0 0 0 0 0 0 0...0]
    chien = [0 0 0 0 0 0 0 0 0 1 0 0 0 0...0]
    maison = [0 0 0 0 0 0 0 0 0 0 0 0 0 1 0...0]
- On perd l'ordre des mots

Tokéniseurs

Source Loîck Bourdois

Tokéniseurs basés sur des règles

*Les tokenizers à base de **règles** nous permettent de tokeniser plus intelligemment au cas par cas.*

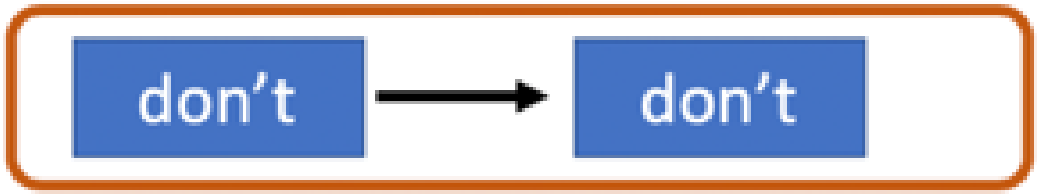
1 Spacy

Largement utilisé : il est rapide, fournit des valeurs par défaut raisonnables et est facilement personnalisable;

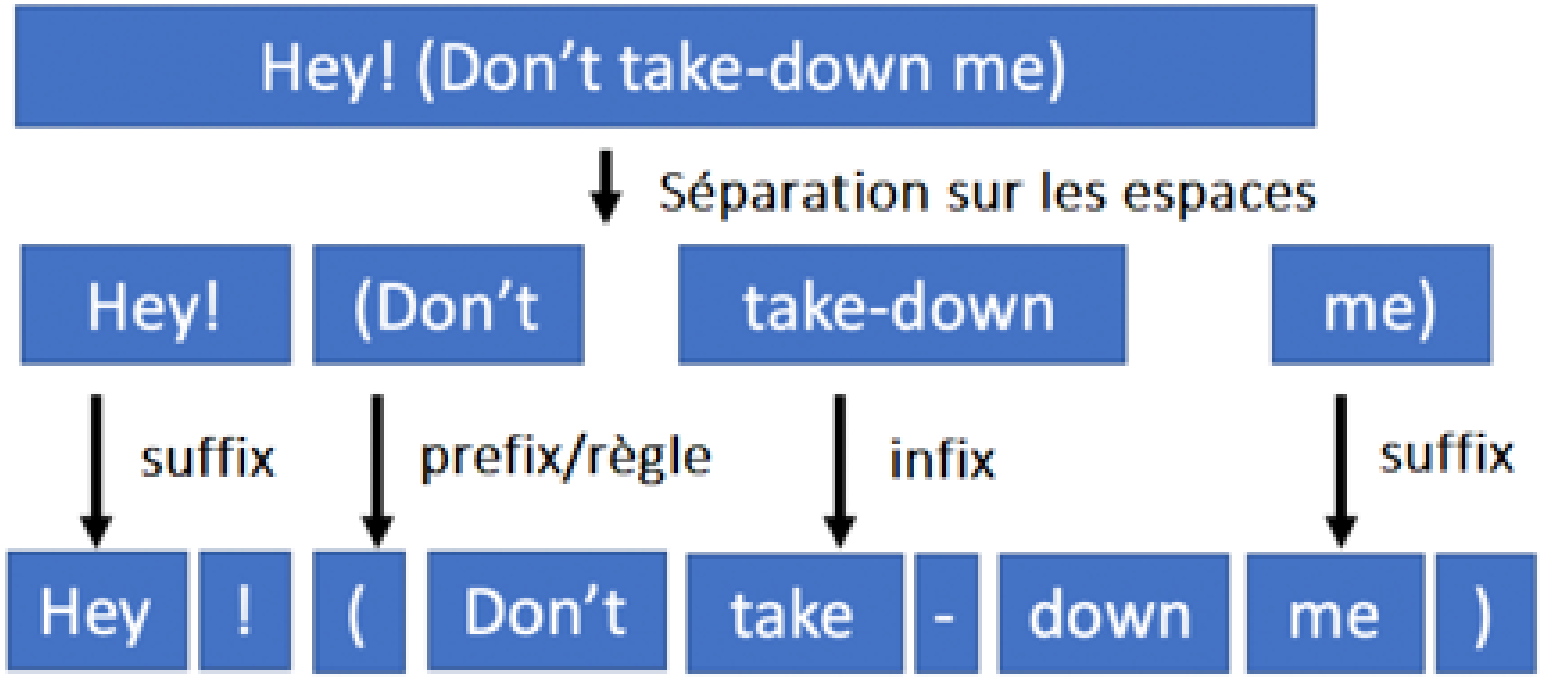
Permet de spécifier des tokens spéciaux
Exemple :
les points de suspension peuvent définis par une règle comme un seul token,

Spacy

Règle choisie :



Traitement :



Exemple de fonctionnent du *tokenizer Spacy*

Moses

2. Moses

Classique, plus ancien que Spacy, largement utilisé en traduction automatique.

Comparé à Spacy il est moins personnalisable.

Remplace en interne certains tokens spéciaux (par exemple des points de suspension) par des tokens personnalisés et est un bon exemple de la façon dont la normalisation et la tokenisation ne sont pas toujours proprement divisées.

Moses fonctionne assez bien sur une langue simple,

Mais problèmes avec certaines entrées comme les émoticônes.

Limitations des tokéniseurs basés sur des règles

- *Capacité relativement limitée à gérer efficacement **les mots rares***.
  - *Le mot « structurally » est relativement rare, mais le mot « structural » est commun, ce qui nous permet de déduire le sens de « structurellement » à partir d'un mot plus fréquent.*
  - *Exemple d'un mot allemand faisant plus de **63 lettres***
  - *Conduit à spécifier chaque mot rare comme une règle spéciale, mais cela devient clairement beaucoup trop complexe très rapidement.*
- *Autre problème majeur : toutes les langues ne divisent pas les mots via des espaces blancs . Le chinois et le japonais etc.*
  - *Ces langues exigent donc des règles beaucoup plus sophistiquées ce qui signifie plus de complexité et potentiellement d'erreurs*

- ***Athazagoraphobie (peur d'être oublié) se décompose en ath, az, agora et phobie***

Si on rencontre ces mots dans un teste, il vient ***<unk>***

Tokénisation en sous mots
Sub-word tokenisation

Idée fondamentale :
Les mots les plus fréquents ont des identificateurs uniques.
Les mots moins fréquents sont décomposés en sous-mots qui conservent le mieux leur signification.

Exemples :

« wonderfully » , fréquent codé comme un seul mot
« structurally » , peu fréquent en « structural » et « ly »

4 tokéniseurs majeurs :

- ***le byte-pair encoding (BPE),***
- ***le wordpiece,***
- ***l'unigram language model,***
- ***et le sentencepiece.***

BPE

Byte-pair encoding

par Sennrich, Haddow et Birch en 1994

Tire ses racines dans la théorie de l'information et compression
Représenter les mots fréquents avec moins de symboles, et les mots moins fréquents avec plus de symboles

Utilisé par Open Ai dans GPT

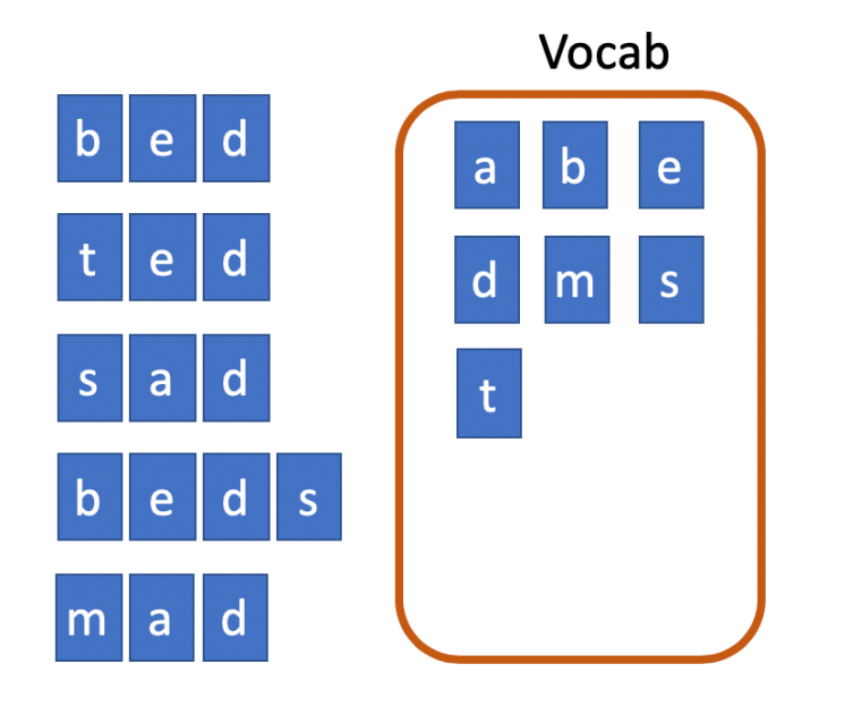
Source : Philip Gage, *Un nouvel algorithme pour la compression de données* . "Journal du Dr Dobbs"

BPE

par Sennrich, Haddow et Birch en 1994

1. *Trouvez le symbole bigramme le plus fréquent (paire de symboles)*
2. *Fusionnez ces symboles pour créer un nouveau symbole et ajoutez-le au vocabulaire. Ceci augmente la taille du vocabulaire de 1.*

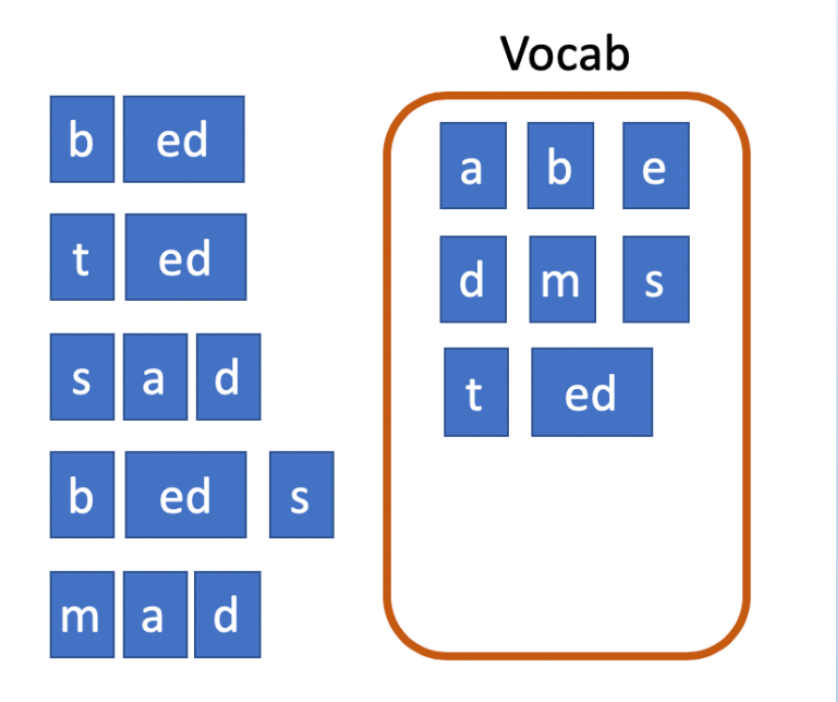
Décomposer en caractères unicodes



Mots

Caractères

En unicode

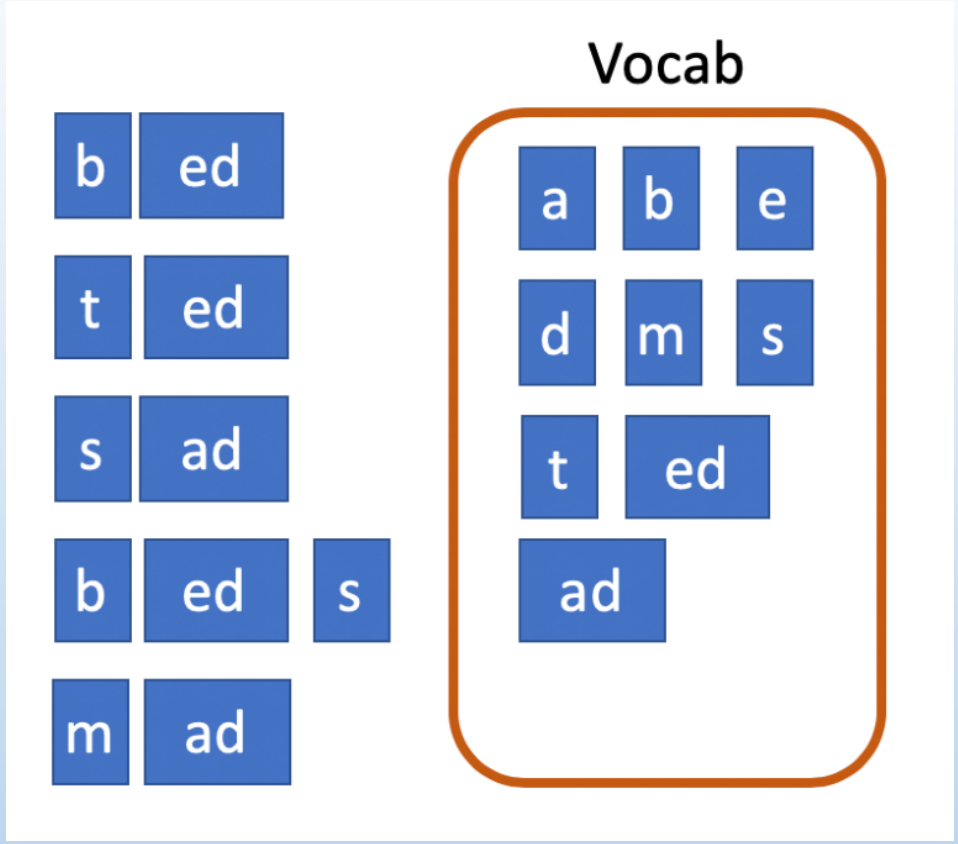


Intégration du bigramme le + fréquent

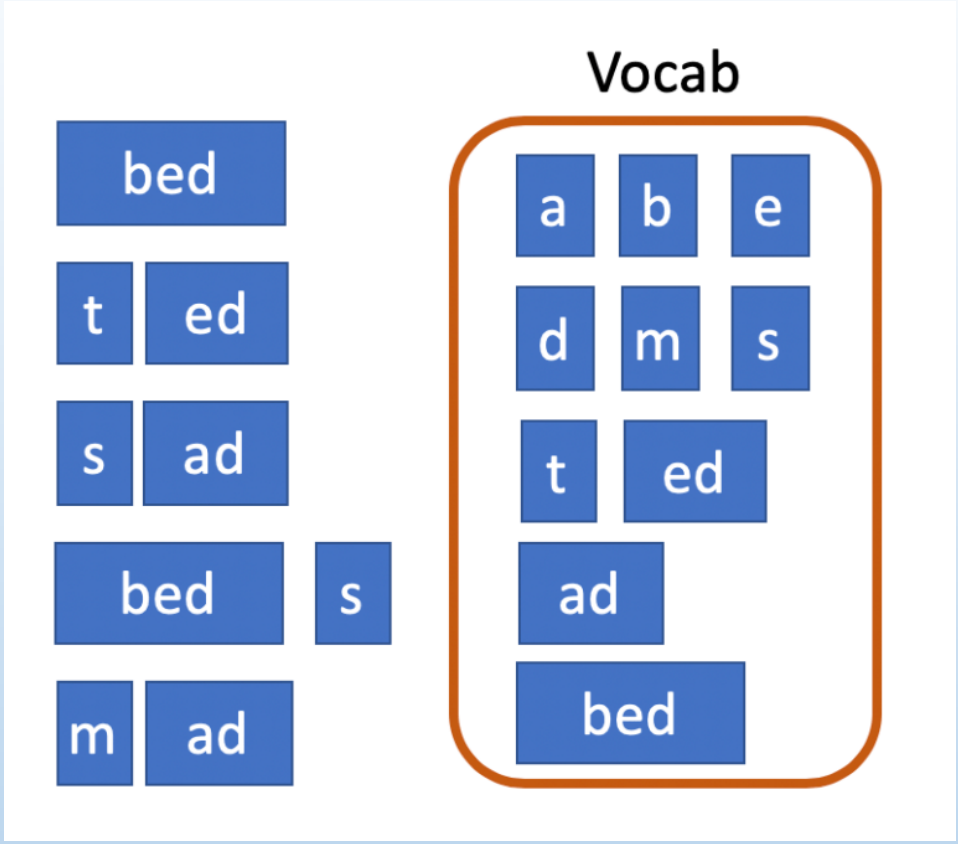
On distinguera ed et ##ed

BPE

par Sennrich, Haddow et Birch en 1994



Intégration du bigramme le + fréquent



Fusion et intégration de la paire de symboles apparaissant 2 fois

Wordpiece *Schuster et Kaisuke*

Utilisé dans BERT

Algorithme de tokenisation en sous-mots largement utilisé. Pratiquement identique à BPE.

La seule différence est qu'au lieu de fusionner le bigramme de symbole le plus fréquent, le modèle fusionne le bigramme qui, une fois fusionné, augmenterait la probabilité d'un modèle de langage unigramme entrainé sur les données d'entraînement.

Unigram Language Model

L'algorithme de tokenization de l'unigram language model a été proposé à l'origine dans cet article par Taku Kudo. Bien qu'il utilise des principes similaires aux méthodes décrites précédemment, il est en fait entraîné très différemment dans la pratique.

L'idée de base de ce tokenizer est d'entraîner un modèle de langage en unigramme, en supposant que tous les mots se produisent indépendamment les uns des autres. Il utilise ensuite ce modèle pour trouver la segmentation la plus probable de chaque mot. L'avantage de cette méthode est qu'elle utilise un modèle probabiliste, ce qui signifie qu'en plus de trouver la segmentation la plus probable, vous pouvez échantillonner des segmentations à partir d'une distribution de probabilités.

Sentencepiece

Résout le problème des langages sans le caractère « Espace ».

Sentencepiece *traite l'entrée comme un flux brut de caractères unicode.*

Il utilise ensuite soit le codage BPE, soit le codage de l'unigram language model au niveau des caractères pour construire le vocabulaire approprié.

Cela signifie que les espaces sont inclus dans la tokenisation.

Par exemple, avec l'unigram language model,

I like natural language processing peut être tokénissé comme

"I", "_like", "_natural", "_lang", "uage", "_process", "ing"

où le caractère espace est remplacé par le trait de soulignement (" _ ") pour plus de clarté.
Notez la distinction avec BPE, où la séquence ci-dessus avec les mêmes sous-mots est tokénisée par

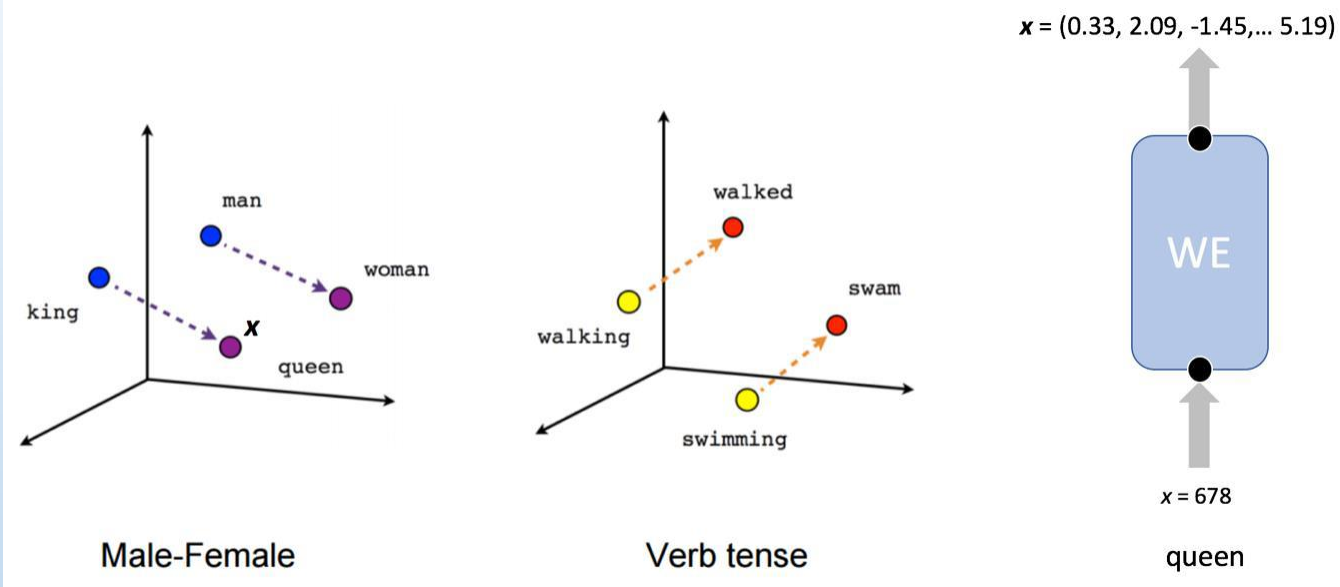
"I", "like", "natural", "lang", "##uage", "process", "##ing"

où les sous-mots sont précédés d'un marqueur spécial. L'ajout de sous-mots avec un marqueur spécial n'a de sens qu'avec un modèle de prétokenisation, puisque sentencepiece ne connaît pas les limites des mots.

4.3 Word Embedding - Plongement

Pourquoi ?

• *Chaque mot ainsi codé ne rend pas compte de la proximité sémantique entre deux termes*

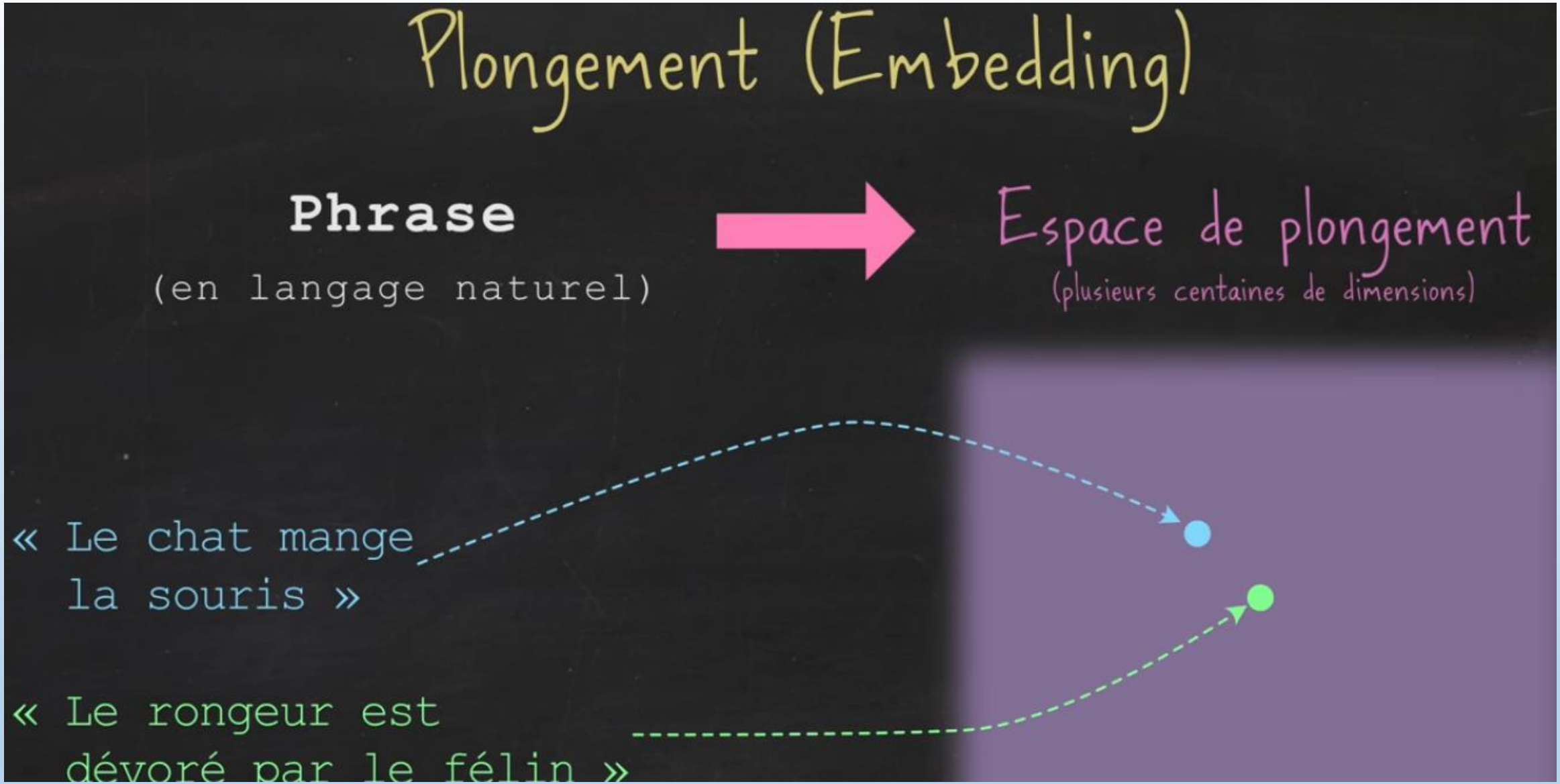


Principe

• *A chaque mot on associe un vecteur xj (dimension par ex 300), représentant la proximité sémantique avec d'autres mots*

• *Matrice de 10 000 ou 100 000 vecteurs par exemple .*

Word Embedding - Plongement



Source « Science étonnante »

Evaluation d'un Embedding

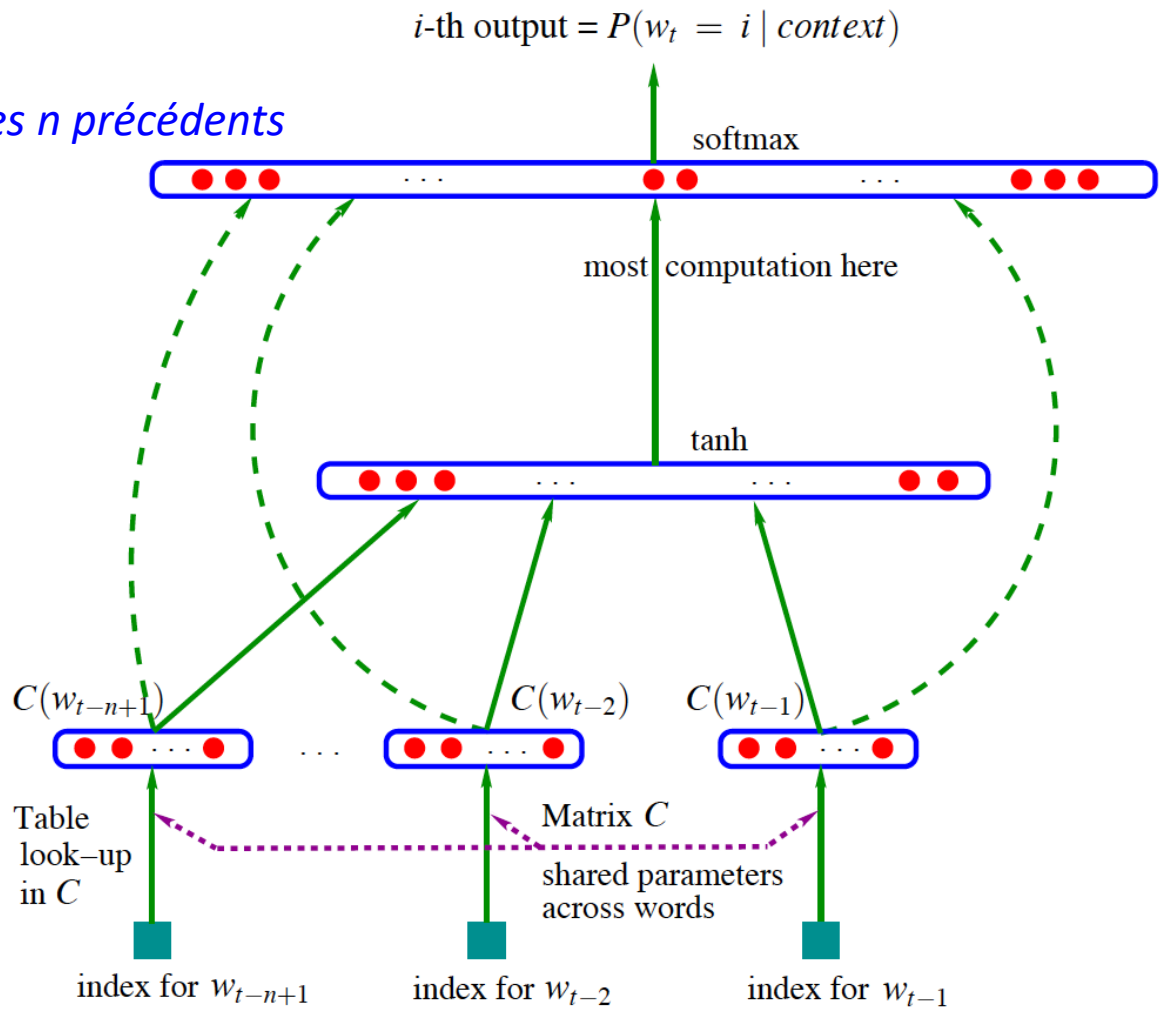
Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

Source: https://opensource.googleblog.com/2013/08/learning-meaning-behind-words.html

Word Embedding

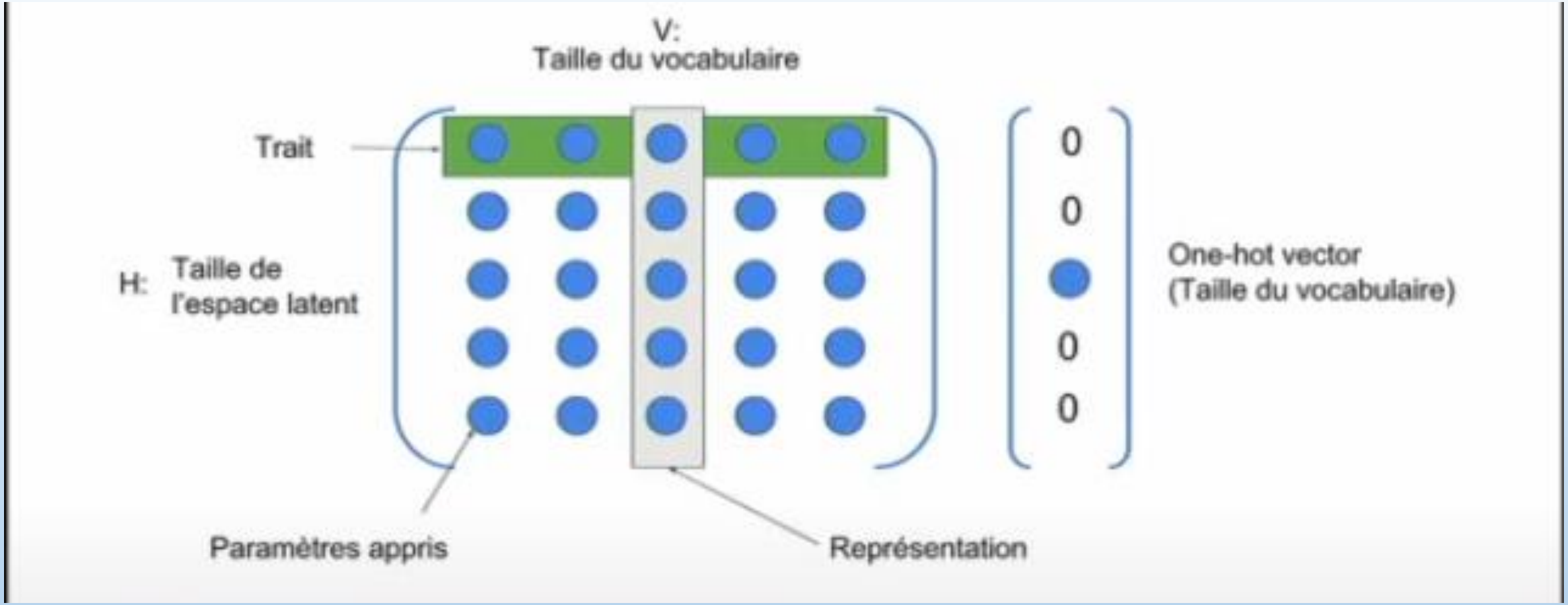
Bengio en 2003

Prédire un mot sachant les n précédents



Bengio et al A neuronal probabilistic langage model

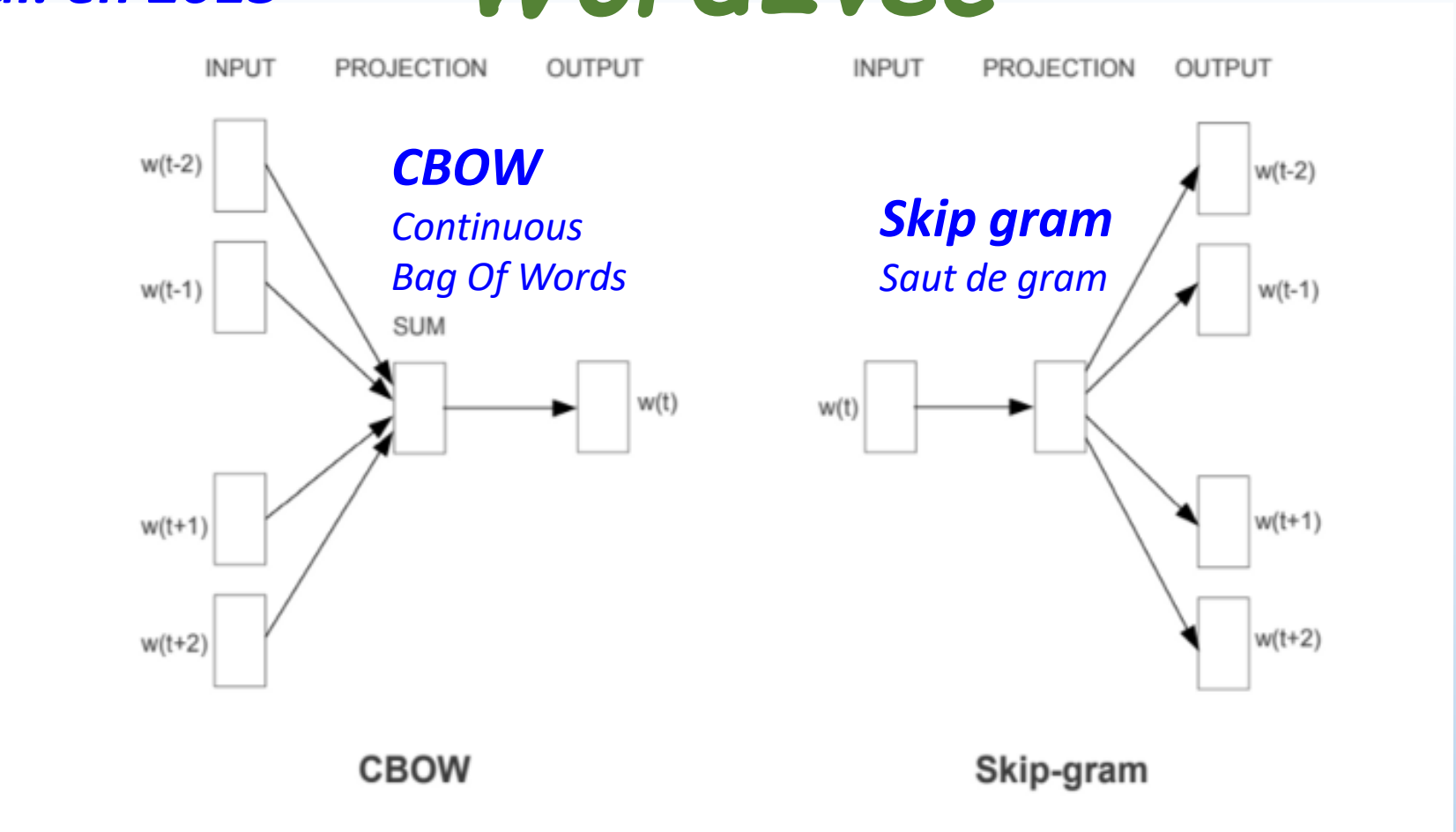
Word Embedding



Word Embedding Existants

Word2vec

Mikolov et al. en 2013

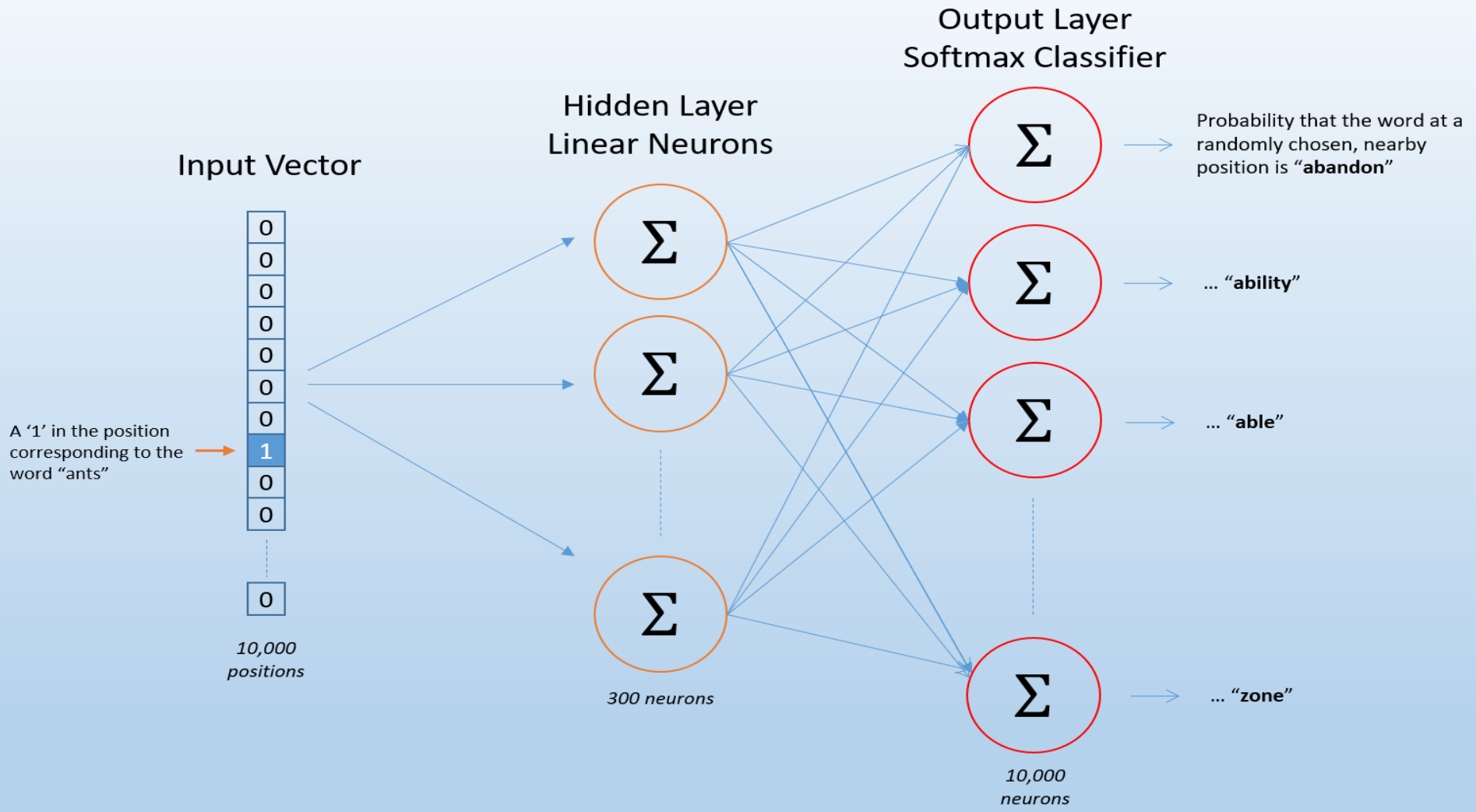


CBOW *modèle, les représentations distribuées du contexte (ou des mots environnants) sont combinées pour* ***prédire le mot du milieu***.

Skip-gram *modèle, la représentation distribuée du mot d'entrée est utilisée pour* ***prévoir le contexte***.

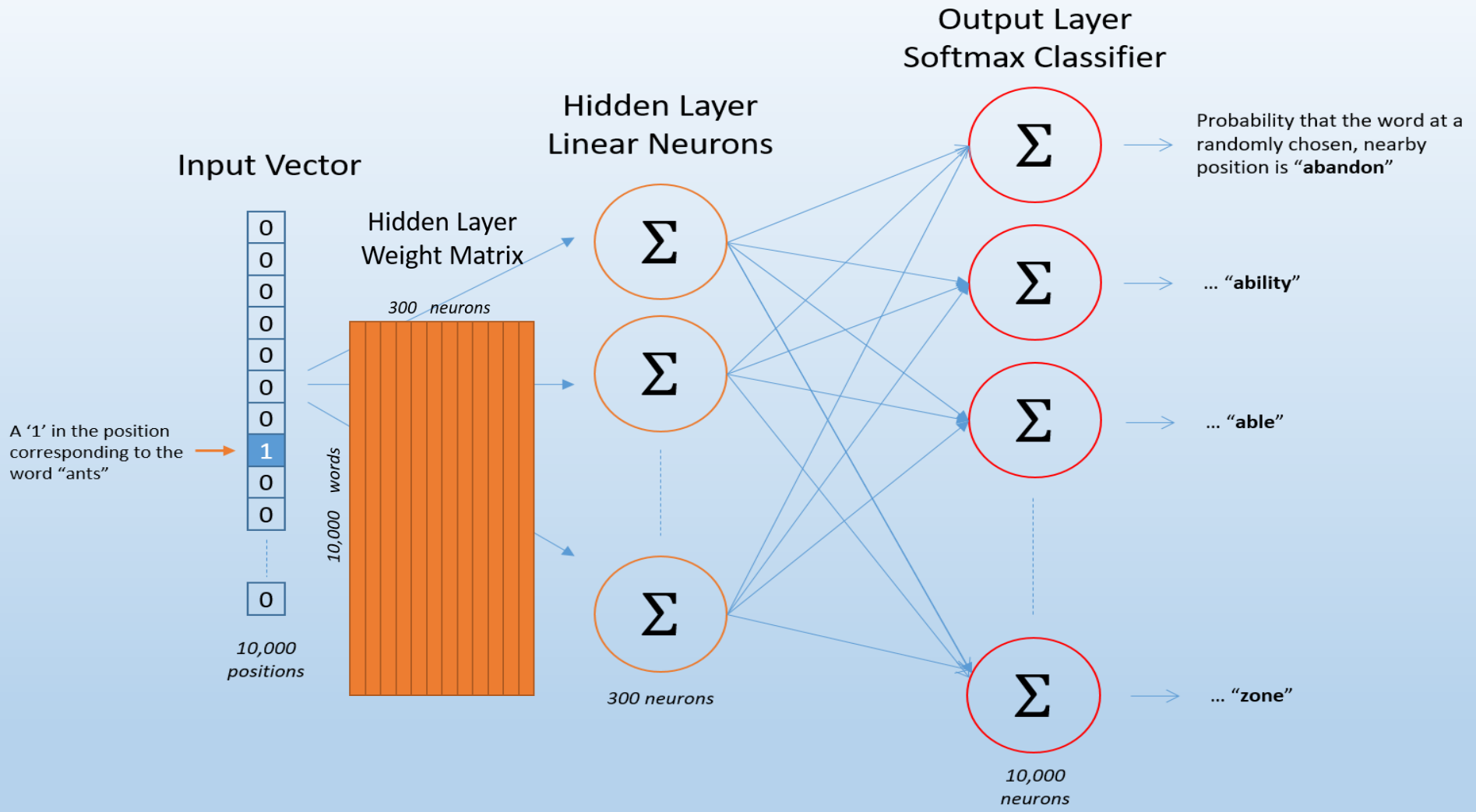
Word2vec

Skip Gram Saut de gram



Word2vec

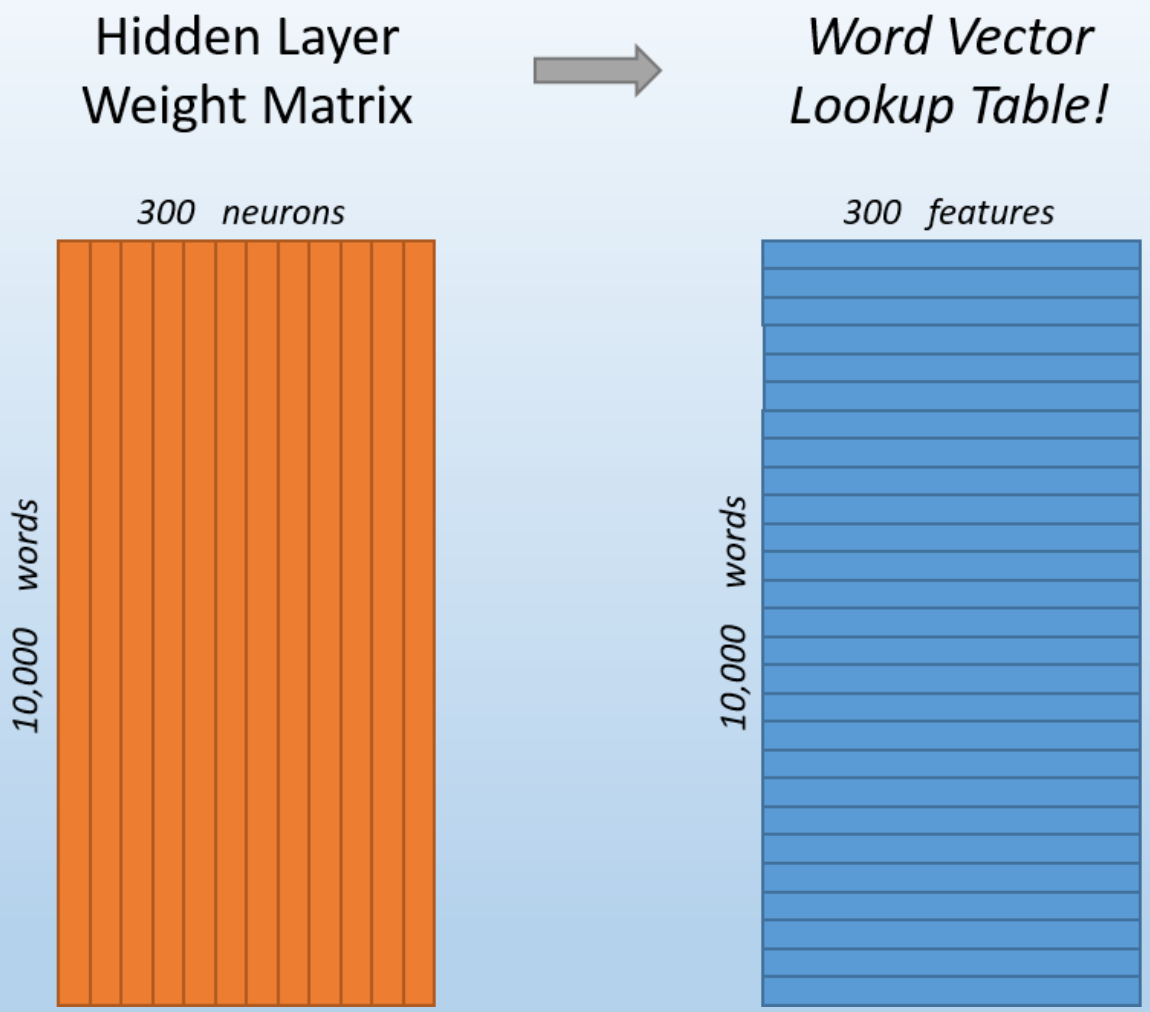
Skip Gram Saut de gram



Word2Vec

Skip Gram Saut de gram

L'objectif final est d'apprendre cette matrice de poids de couche cachée -



Word2vec

Si on multiplie un vecteur 1 x 10 000 par une matrice 10 000 x 300, on obtient le vecteur de ce mot

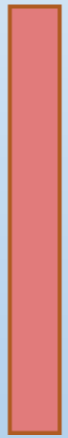
$$\underbrace{[0 \quad 0 \quad 0 \quad 1 \quad 0]}_{10\,000} \times \underbrace{\begin{bmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ 4 & 6 & 13 \\ 10 & 12 & 19 \\ 11 & 18 & 25 \end{bmatrix}}_{10\,000 \times 300} = \underbrace{[10 \quad 12 \quad 19]}_{300}$$

Output weights for "car"

Word vector for "ants"

300 features

×



300 features

softmax

$$\frac{e^x}{\sum e^x}$$

=

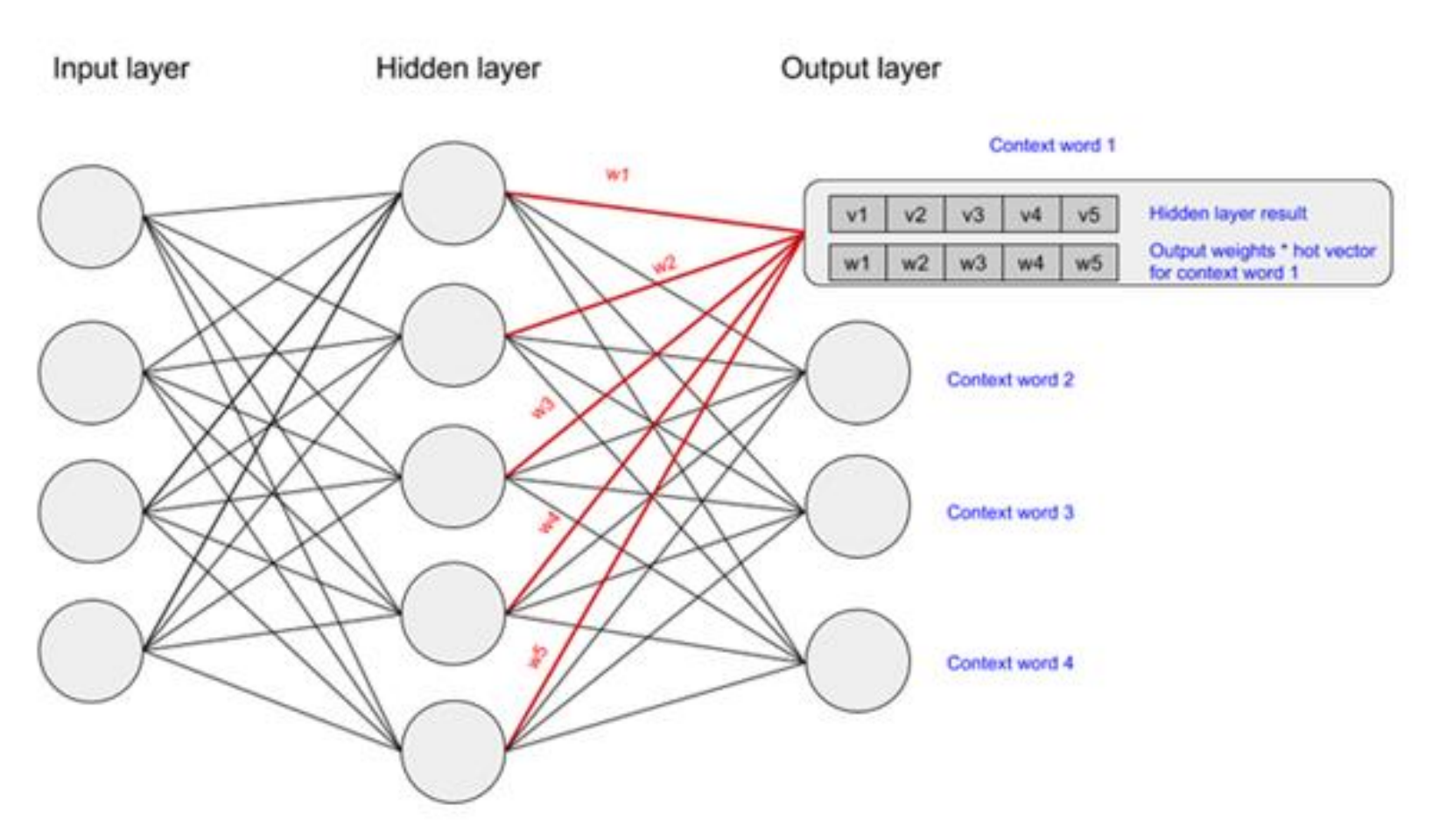
Probability that if you randomly pick a word nearby "ants", that it is "car"

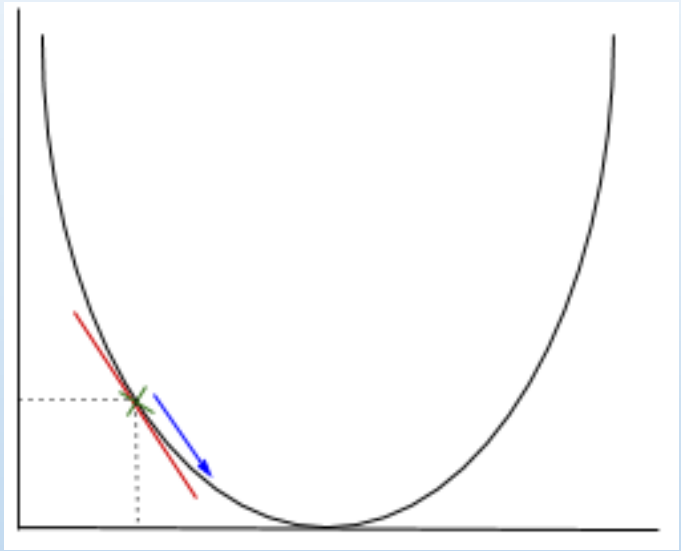
Skip-Gram - Entrainement

Etant donné un mot, nous essaierons de prédire ses mots voisins. Nous allons définir un mot voisin par la taille de la fenêtre — un hyper-paramètre, ici = 2

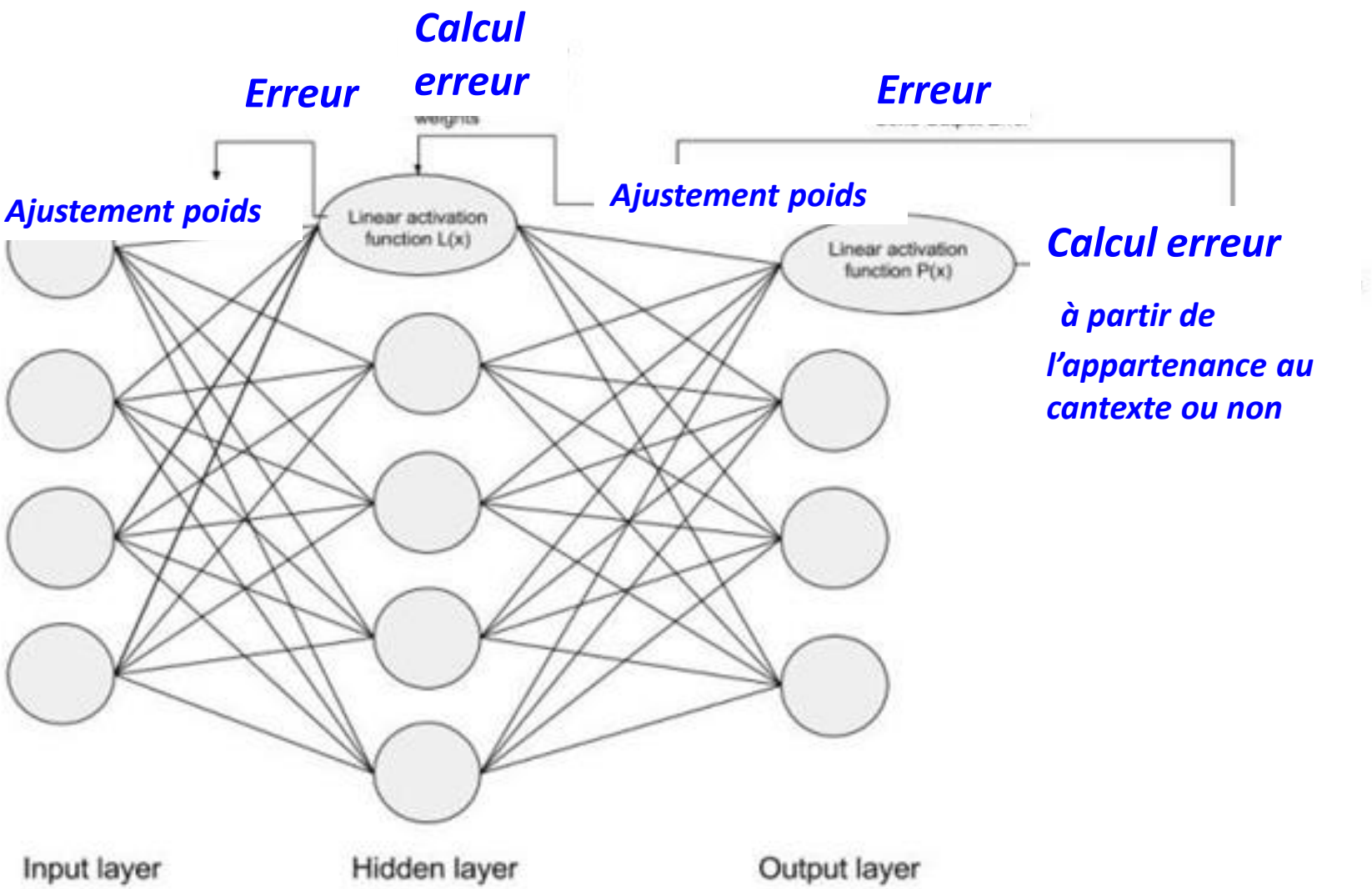
Source Text	Training Samples generated from source text			
I will have orange juice and eggs for breakfast	(will, I)	(will, have)	(will, orange)	
I will have orange juice and eggs for breakfast	(have, I)	(have, will)	(have, orange)	(have, juice)
I will have orange juice and eggs for breakfast	(orange, will)	(orange, have)	(orange, juice)	(orange, and)
I will have orange juice and eggs for breakfast	(juice, have)	(juice, orange)	(juice, and)	(juice, eggs)
I will have orange juice and eggs for breakfast	(and, orange)	(and, juice)	(and, eggs)	(and, for)
I will have orange juice and eggs for breakfast	(eggs, juice)	(eggs, and)	(eggs, for)	(eggs, breakfast)
I will have orange juice and eggs for breakfast	(for, and)	(for, eggs)	(for, breakfast)	

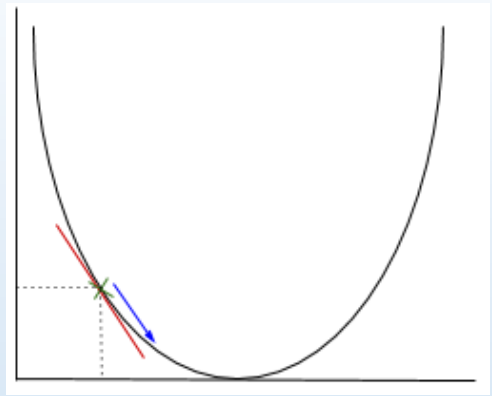
Skip-Gram - Entrainement



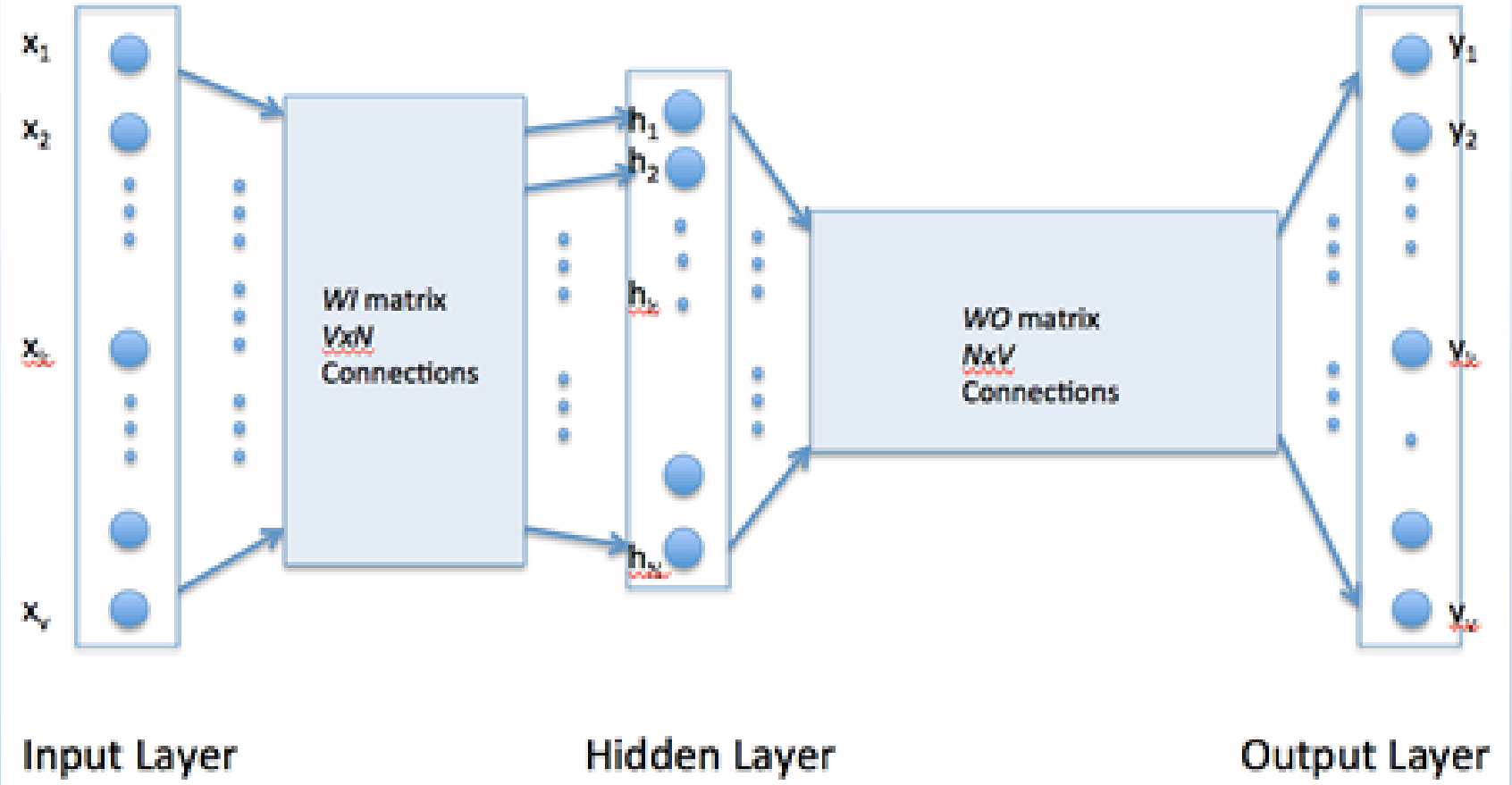


Skip-Gram - Entrainement





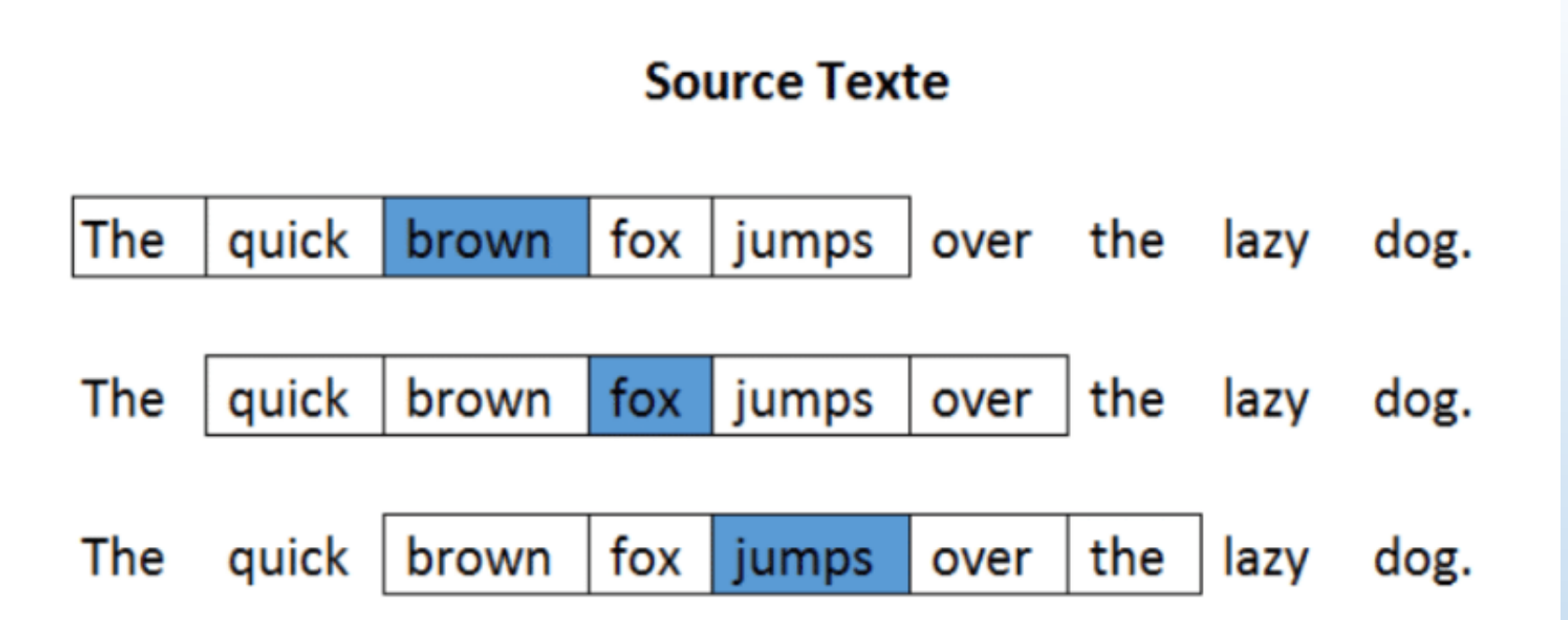
Skip-Gram - Entrainement



Matrice WI de taille VxN avec chaque ligne représentant un mot de vocabulaire

Matrice WO de taille NxV avec chaque colonne représentant un mot du vocabulaire donné.

CBOW

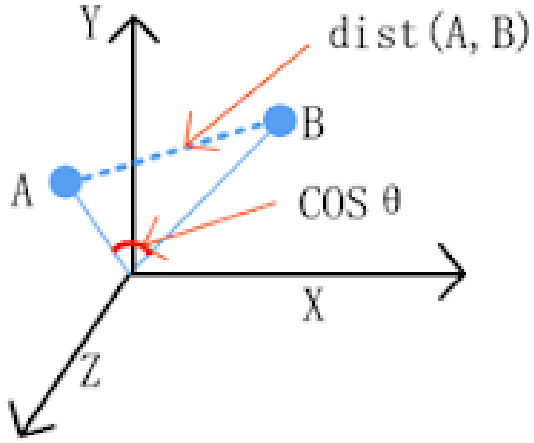


***CBOW : Le modèle est nourri par le contexte**, et prédit le mot cible.*
Le résultat de la couche cachée est la nouvelle représentation du mot

Word2vec

Similarité des mots

sémantiques, syntaxiques ou thématiques



$$dist(A, B) = cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \, \|\mathbf{B}\|}$$

$$dist(A, B) = \frac{\sum_{i=1}^{N} A_i B_i}{\sqrt{\sum_{i=1}^{N} A_i^2} \sqrt{\sum_{i=1}^{N} B_i^2}}$$

Avec cette métrique et dans ce sous-espace vectoriel, les 5 mots ***les proches de « body »*** *sont :*

intestines — 0.3054816126823425 3
bodies — 0.2691531181335449
arm — 0.24878980219364166
chest — 0.2261650413274765
leg — 0.2193179428577423

Entrainement Word2vec

Avec 300 fonctionnalités et un vocabulaire de 10 000 mots, cela fait 3 Millions de poids dans la couche cachée et la couche de sortie chacune!

L'entraînement sur un plus grand vocabulaire et un grand ensemble de données serait prohibitif

Les auteurs de Word2Vec ont apporté des solutions

1. ***Sous-échantillonnage** des mots fréquents pour diminuer le nombre d'exemples de formation.*
2. *Modifier l'objectif d'optimisation avec une technique qu'ils ont appelée **« échantillonnage négatif »,** qui fait que chaque échantillon d'entraînement ne met à jour qu'un faible pourcentage des poids du modèle.*

Glove

Global Vectors

Université Stanford

GloVe est un algorithme d'apprentissage non supervisé pour obtenir des représentations vectorielles pour les mots. La formation est effectuée sur des statistiques globales agrégées de ***cooccurrence*** *mot-mot à partir d'un corpus,*

L'objectif de formation de GloVe est d'apprendre des vecteurs de mots tels que leur produit scalaire soit égal au logarithme de la probabilité de ***cooccurrence*** *des mots associés*

$$F(w_i - w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}} .$$

Glove

Université Stanford

L'intuition principale qui sous-tend le modèle est la simple observation que les rapports de probabilités de cooccurrence mot-mot ont le potentiel d'encoder une certaine forme de sens.

Exemple, considérons les probabilités de cooccurrence des mots cibles glace et vapeur avec divers mots du vocabulaire.

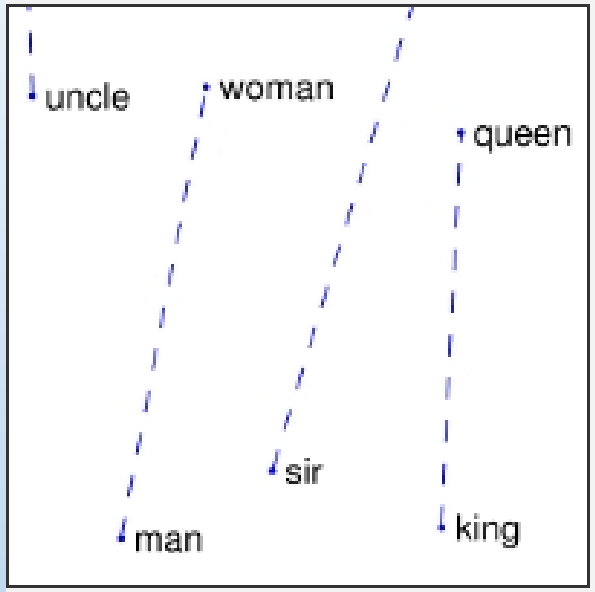
Probabilités réelles à partir d'un corpus de 6 milliards de mots :

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

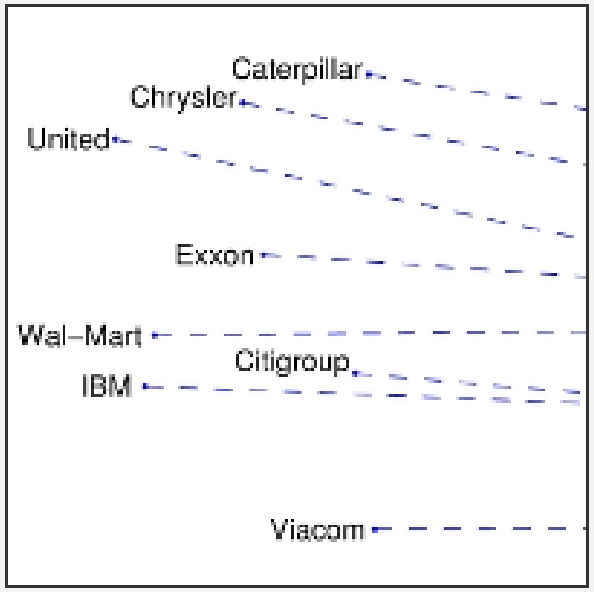
Ce n'est que dans le rapport des probabilités que le bruit des mots non discriminatoires comme *l'eau* et *la mode* s'annulent.

Glove

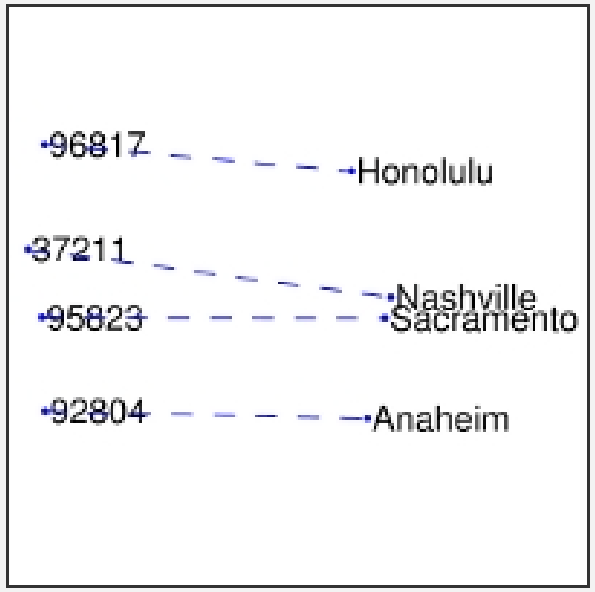
GloVe est conçu pour que ces différences vectorielles capturent autant que possible le sens spécifié par la juxtaposition de deux mots.



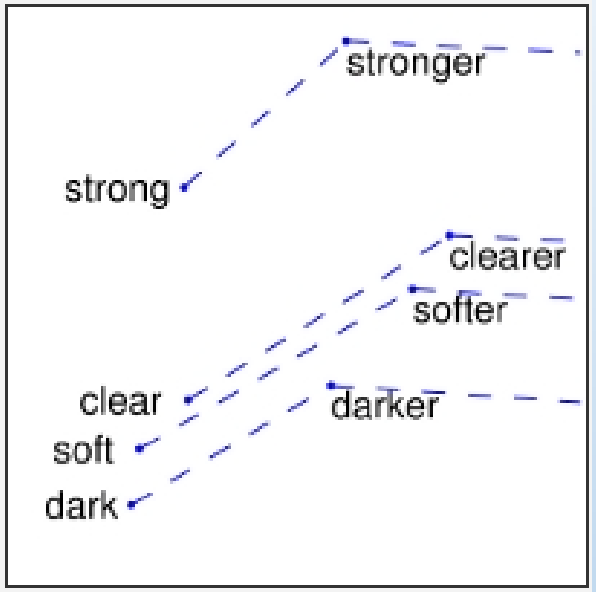
homme - femme



Entreprise - PDG



Ville - Code postal

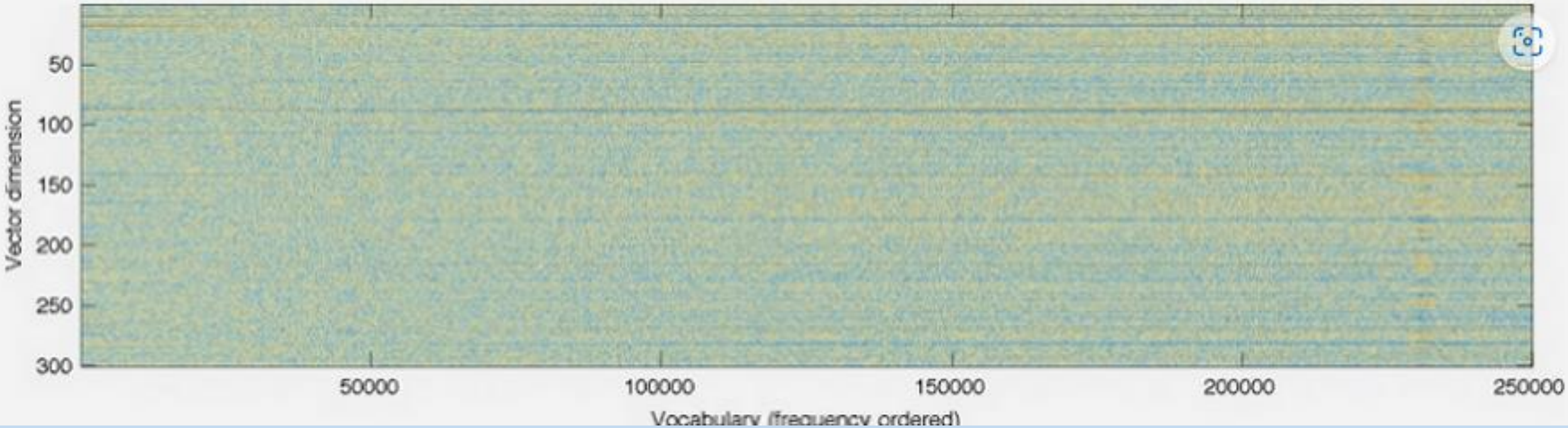


Comparatif - Superlatif

Glove

Université Stanford

GloVe produit des vecteurs de mots avec une structure à bandes marquées qui peut être visualisée



Les bandes horizontales deviennent plus prononcées à mesure que la fréquence des mots augmente.
Les bandes verticales, comme celle autour des mots 230k-233k, sont dues à des densités locales de mots apparentés (généralement des nombres) qui ont des fréquences similaires

Fast Text

Facebook 2016

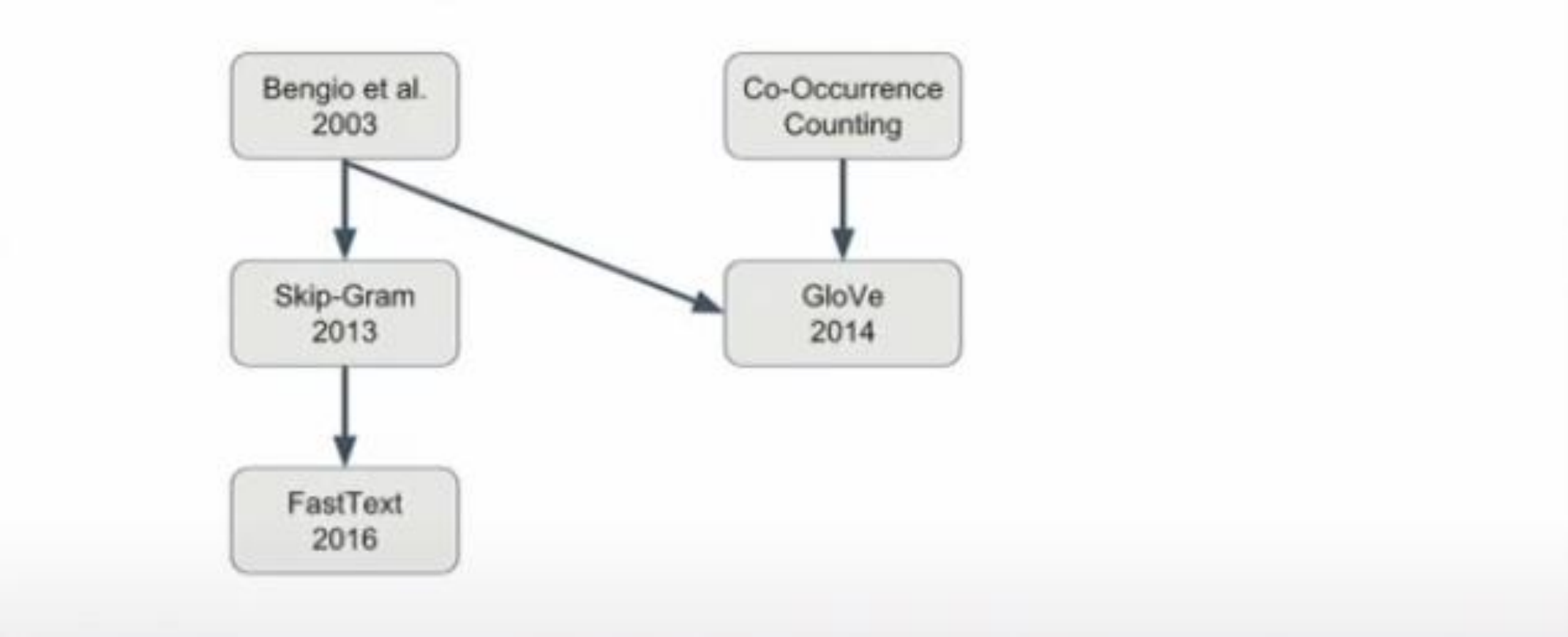
FastText

- **FastText**: Variante du skip-gram où chaque mot est représenté par un ensemble de n-grams
  - Ex: "**where**" - <wh, whe, her, ere, re>
  - Un mot = somme des embeddings de ses N-grams
  - Possibilité d'embeddings pour des mots **hors-vocabulaire**.
  - Fonction de score:

$$s(w,c) = \sum_{g \in G_w} z_g^T v_c \quad \text{Avec} \quad G_w = \text{Ens. des N-grams du mot } w \quad z_g = \text{Embeddings du N-gram } g$$

Bojanowski et al., "Enriching word vectors with subword information," arXiv preprint arXiv:1607.04606

Embeddings existants



Tous ces embeddings (pré formés ou entrainés… sont disponibles en ligne

Evaluation d'un Embedding

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

Source: https://opensource.googleblog.com/2013/08/learning-meaning-behind-words.html