Traitement Automatique du Langage Naturel (NLP)

Word embeddings et Transformers

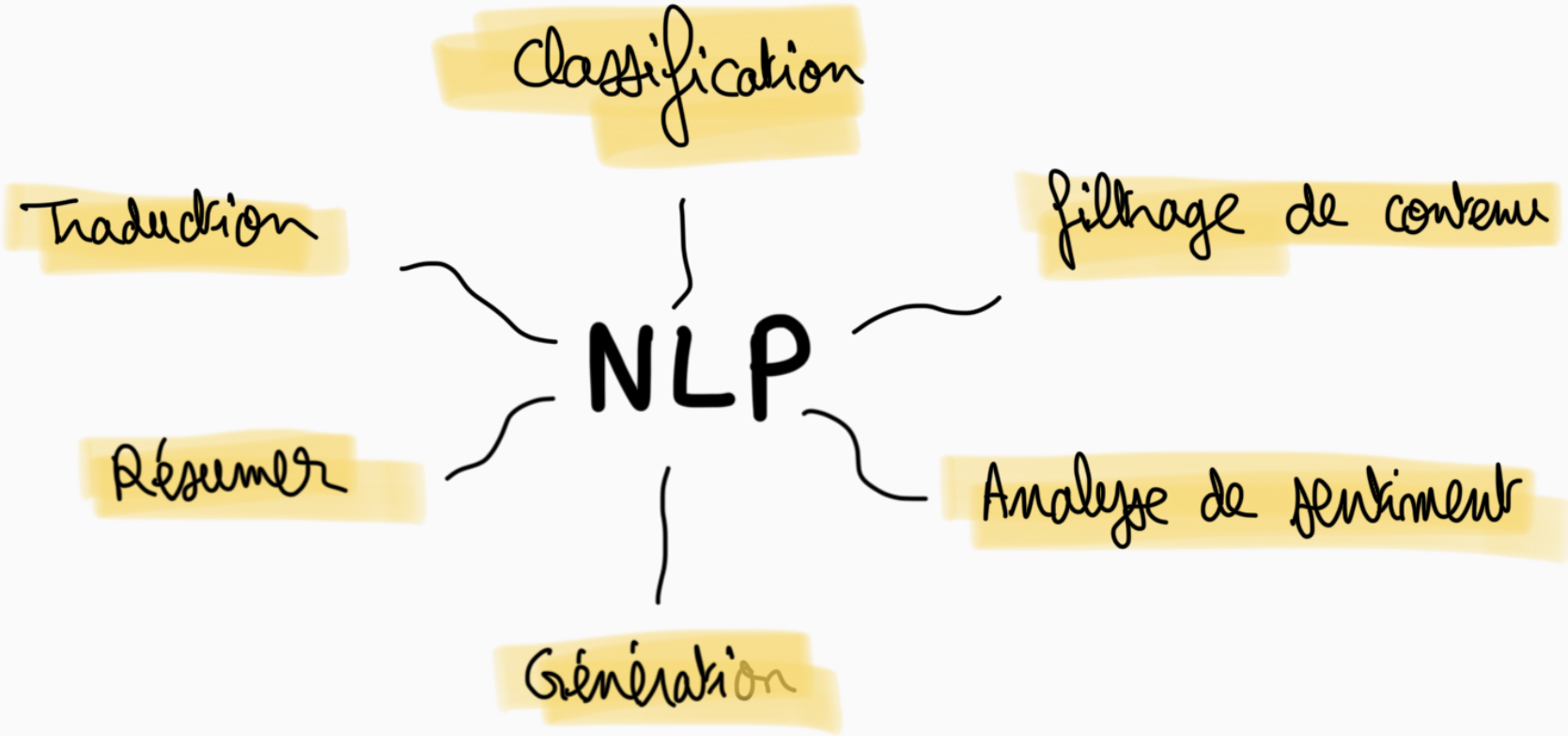
NLP

Qu'est-ce que le NLP ?

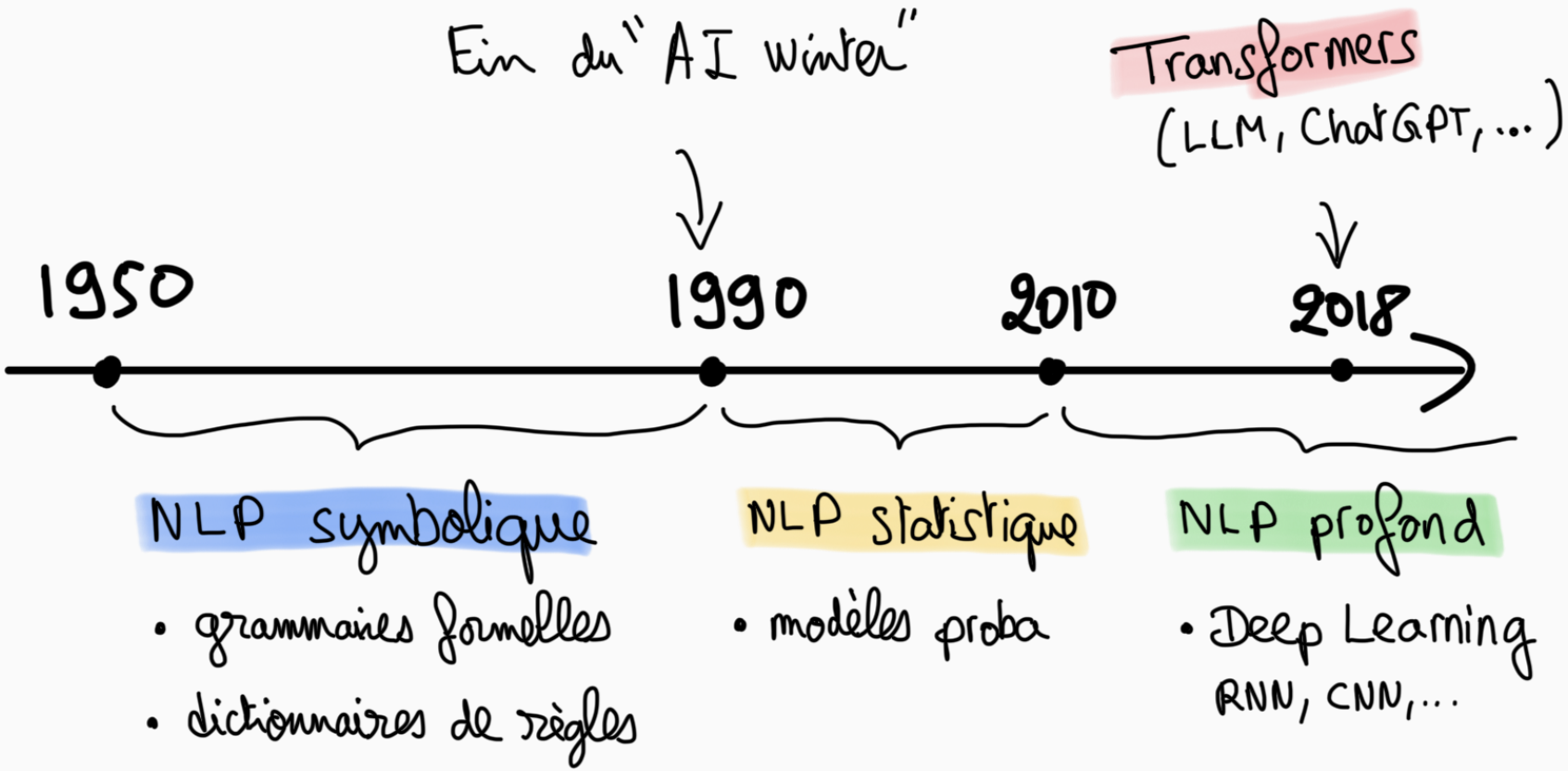
- **Natural Language Processing (NLP)**, ou **Traitement Automatique du Langage Naturel (TALN)** en français
  - But : développer des outils permettant aux machines de traiter le langage humain.
- **Langues naturelles** : (français, anglais, arabe, chinois, etc.)
  - Langues utilisées par les humains pour communiquer.
  - Imprécises, ambiguës, et fortement dépendantes du contexte.
- **Langages machines** : (Python, Assembleur, LISP, etc.)
  - Langages conçus pour être compris et exécutés par les machines.
  - Précis, souvent non ambigus et régis par des règles déterministes.

Pourquoi le NLP ?

L'objectif n'est pas de "comprendre" le langage humain, mais de l'utiliser pour accomplir des tâches spécifiques.

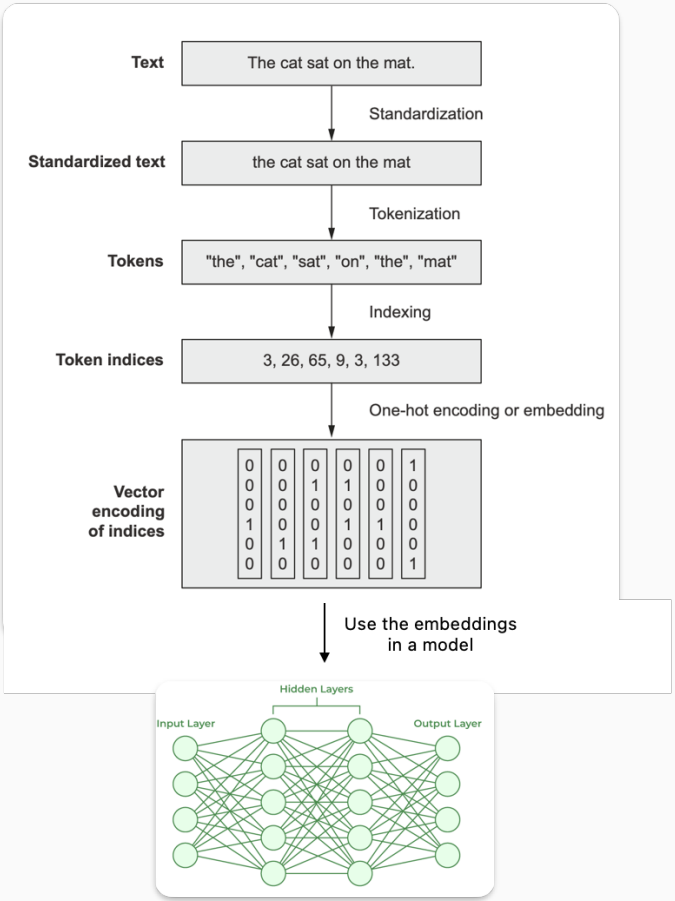


Évolution du NLP



Pipeline NLP

- **Standardisation** : Normalisation du texte (minuscules, accents, enlever la ponctuation, etc.)
- **Tokenisation** : Découpage du texte en caractères, mots, groupes de mots, etc.
- **Indexation** : Création d'un index pour accéder rapidement aux tokens.
- **Vectorisation** : Transformation des tokens en vecteurs numériques.



Credit: Livre de Francois Chollet "Deep Learning with Python", et le site GeeksforGeeks.

Tokenisation

- **Tokenisation** :
  - Découpage en mots (ou 1-grammes)
    - Chaque mot correspond à un token. “Le chat a son zoomie” -> [“Le”, “chat”, “a”, “son”, “zoomie”].

Tokenisation

- **Tokenisation** :
  - Découpage en mots (ou 1-grammes)
    - Chaque mot correspond à un token. “Le chat a son zoomie” -> [“Le”, “chat”, “a”, “son”, “zoomie”].
  - Découpage en N-grammes
    - Un token est un ensemble de N mots consécutifs. Par exemple, un bigramme ($N = 2$) est un ensemble de deux mots consécutifs. “Le chat a son zoomie” -> [“Le chat”, “chat a”, “a son”, “son zoomie”].
    - Permet d’injecter un peu d’ordre entre les mots.

On parle de vocabulaire pour désigner l’ensemble des tokens dans un corpus.

Indexation

- Chaque token est associé à un identifiant unique (un entier).

Exemple 1.

- Vocabulaire : [“chat”, “chien”, “oiseau”, “poisson”, “a”, “le”, “son”, “zoomie”]
- “Le”, “chat”, “a”, “son”, “zoomie” -> [5, 0, 4, 6, 7]

Indexation

- Chaque token est associé à un identifiant unique (un entier).

Exemple 1.

- Vocabulaire : [“chat”, “chien”, “oiseau”, “poisson”, “a”, “le”, “son”, “zoomie”]
- “Le”, “chat”, “a”, “son”, “zoomie” -> [5, 0, 4, 6, 7]

Exemple 2.

- Vocabulaire : [“chien a”, “son os”, “le chat”, “chat a”, “a son”, “son zoomie”, “le chien”]
- “Le chat”, “chat a”, “a son”, “son zoomie” -> [2, 3, 4, 5]

Vectorisation : deux exemples simples

- **One-Hot Encoding** :
  - Un token = un vecteur binaire de taille k, où k est la taille du vocabulaire.
  - On met à 1 l'indice correspondant au token, et 0 partout ailleurs.
  - Exemple :
    - Vocabulaire : [“chat”, “chien”, “oiseau”, “poisson”, “a”, “le”, “son”, “zoomie”]
    - “chat”, index 0 : [1, 0, 0, 0, 0, 0, 0, 0]
    - “chien”, index 1 : [0, 1, 0, 0, 0, 0, 0, 0]
    - “oiseau”, index 2 : [0, 0, 1, 0, 0, 0, 0, 0]
    - etc.

La phrase “Le chat a son zoomie” peut être représentée

- une matrice de taille 5×8 (5 tokens dans la phrase, 8 tokens dans le vocabulaire) modèle séquentiel, où chaque ligne correspond à un token de la phrase.
- un vecteur de taille 8 modèle bag-of-words, où chaque indice correspond à un token du vocabulaire.

Modèle séquentiel et one-hot

- Chaque document est représenté par une matrice de taille $n \times k$, où n est le nombre de tokens dans le document et k est le nombre total de tokens dans le vocabulaire.
- La matrice est composée de 0 partout, sauf aux indices correspondant aux mots présents dans le document, où elle est égale à 1.
- Exemple :
  - Vocabulaire : [“chat”, “chien”, “oiseau”, “poisson”, “a”, “le”, “son”, “zoomie”]
  - “Le chat a son zoomie” :

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

  - etc.

Modèle bag-of-words

- Un document = un vecteur de taille k, où k est le nombre total de tokens dans le vocabulaire.
- On met 0 partout, et 1 aux indices correspondant aux tokens présents.
- On perd l'ordre exact des mots.
- On ajoute souvent un peu d'ordre entre les mots en utilisant des N-grammes plutôt que de faire un découpage en mots.
- Exemple :
  - Vocabulaire (tokenisation $\leq$ 2-grammes) : ["le chat", "chat a", "a son", "son zoomie", "le chien", "chien a", "a son", "son os", "le", "chat", "chien", "os", "poisson", "a", "son", "zoomie"]
  - "Le chat a son zoomie" : [1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1]
  - "Le chien a son os" : [0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0]
  - etc.

OOV (Out-Of-Vocabulary) et le Padding

- L'index 1 est réservé aux tokens qui ne sont pas dans le vocabulaire connu. Ces tokens sont remplacés par un token spécial, souvent appelé <UNK> (pour "unknown").
- L'index 0 est utilisé pour le **padding**, qui permet de normaliser la longueur des séquences. Cela remplit les vecteurs de taille variable avec des zéros pour obtenir des matrices de taille fixe (nécessaire pour les modèles séquentiels comme les réseaux de neurones récurrents).

- **Exemple** :
  - **Vocabulaire** :
    [*mask token*, <UNK>, “chat”, “chien”, “oiseau”, “poisson”, “a”, “le”, “son”, “zoomie”]
  - **Phrase d’entrée** :
    “Le chat a eu son zoomie” (encodée comme une séquence de longueur 7) : $[7, 2, 5, 1, 8, 9, 0, 0]$.
  - **Matrice résultante** (représentation des tokens en one-hot encoding) :

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Session de coding

nlp.ipynb

- **Dataset IMDB** :
  - 50 000 critiques de films.
  - **Données** :
    - **X** = “A rating of”1” does not begin to express how dull, depressing and relentlessly bad this movie is.”
    - **y** = 0 (négatif)
- **Objectif** :

  Créer un modèle de base capable de classer les critiques de films comme étant positives ou négatives.

Limites des Représentations Classiques

- **Taille du vocabulaire** :
  - Le vocabulaire peut être extrêmement large, surtout dans des corpus volumineux, rendant les méthodes classiques comme le **one-hot encoding** et le **bag-of-words** très inefficaces en termes de mémoire et de calcul.
- **Absence de sémantique** :
  - Les méthodes classiques ne capturent pas la **sémantique** des mots. Chaque mot est représenté de manière indépendante, sans aucune information sur les relations ou similarités entre eux.
  - Idéalement, on souhaite que “chat” et “félin” soient représentés par des vecteurs similaires. C’est-à-dire que ces mots soient **proches** pour la **distance cosinus** ou **euclidienne**.

Tesgüino (exemple de Lin, 1998)

Tesgüino, kézako ?

Tesgüino (exemple de Lin, 1998)

Tesgüino, kézako ?

- Une bouteille de tesgüino est sur la table.

Tesgüino (exemple de Lin, 1998)

Tesgüino, kézako ?

- Une bouteille de tesgüino est sur la table.
- Tout le monde aime le tesgüino.

Tesgüino (exemple de Lin, 1998)

Tesgüino, kézako ?

- Une bouteille de tesgüino est sur la table.
- Tout le monde aime le tesgüino.
- Il est ivre mort après avoir bu du tesgüino.

Tesgüino (exemple de Lin, 1998)

Tesgüino, kézako ?

- Une bouteille de tesgüino est sur la table.
- Tout le monde aime le tesgüino.
- Il est ivre mort après avoir bu du tesgüino.
- Tesgüino est fait à partir de maïs fermenté.

Tesgüino (exemple de Lin, 1998)

Tesgüino, kézako ?

- Une bouteille de tesgüino est sur la table.
- Tout le monde aime le tesgüino.
- Il est ivre mort après avoir bu du tesgüino.
- Tesgüino est fait à partir de maïs fermenté.

–> Tesgüino est une boisson alcoolisée fait à partir de maïs fermenté.

Comment avez-vous compris le sens de tesgüino ?

Vous avez utilisé le contexte pour déduire la signification de tesgüino.

- (1) Une bouteille de . . . est sur la table.
- (2) Tout le monde aime le . . . .
- (3) Il est ivre mort après avoir bu trop de . . . .
- (4) . . . est fait à partir de maïs fermenté.

Comment avez-vous compris le sens de tesgüino ?

Vous avez utilisé le contexte pour déduire la signification de tesgüino.

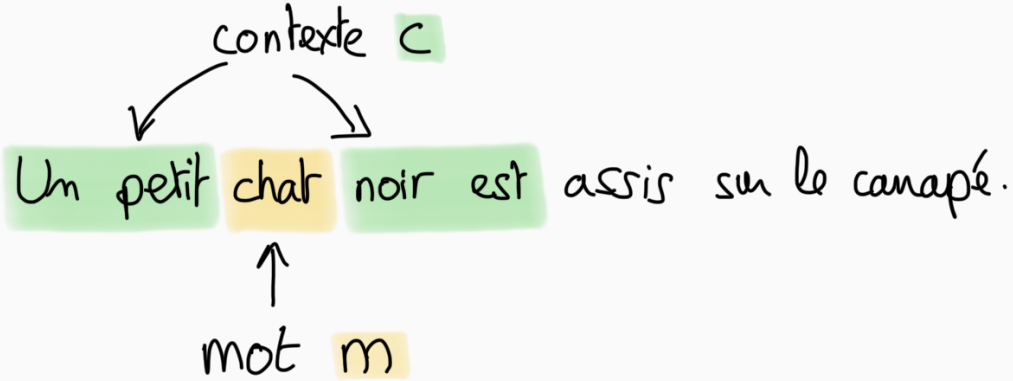
- (1) Une bouteille de . . . est sur la table.
- (2) Tout le monde aime le . . . .
- (3) Il est ivre mort après avoir bu trop de . . . .
- (4) . . . est fait à partir de maïs fermenté.

	(1)	(2)	(3)	(4)
tesgüino	1	1	1	1
bruit	0	0	0	0
huile moteur	1	0	0	1
tortilla	0	1	0	1
vin	1	1	1	0

Hypothèse Distributionnelle

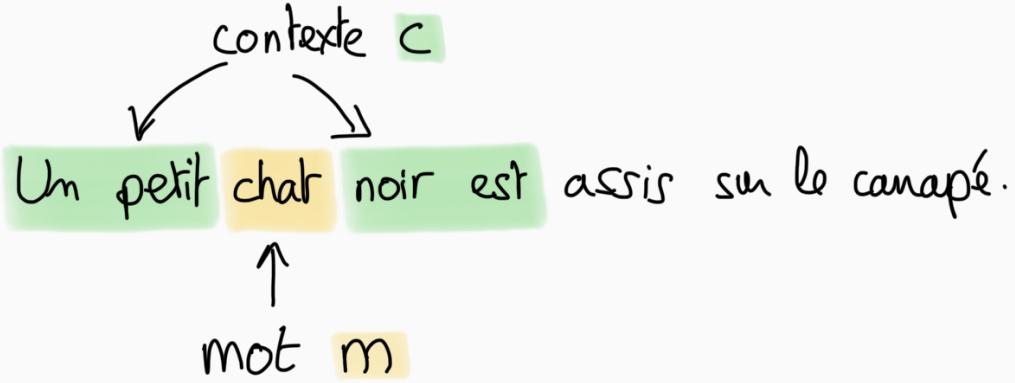
- **Hypothèse Distributionnelle** :
  - Les items linguistiques ayant des distributions similaires partagent des significations similaires.
  - "You shall know a word by the company it keeps." — **John Rupert Firth**
- Cela signifie que deux mots qui apparaissent dans des contextes similaires auront probablement des significations proches.

Factorisation de la Matrice de Co-occurrence

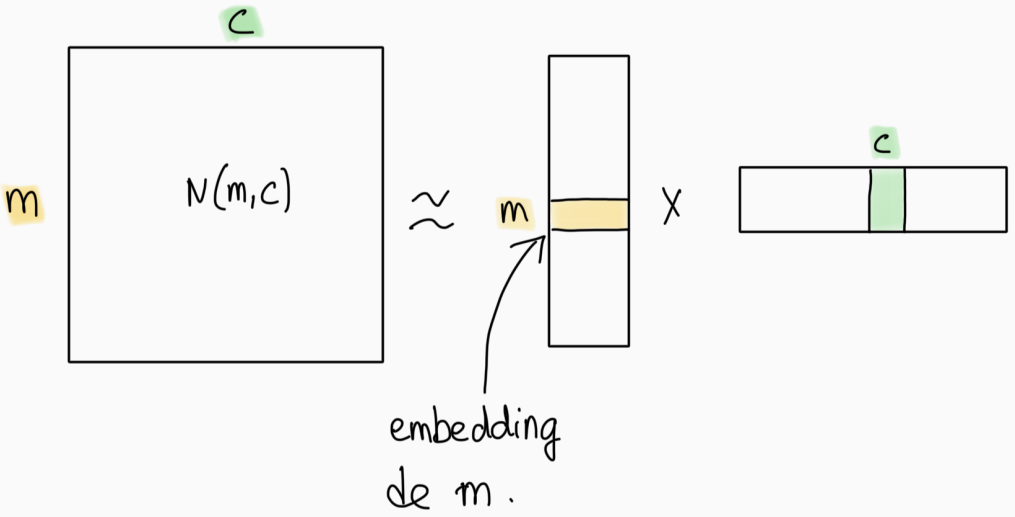


- $N(m, c)$ est le nombre d'occurrences du mot m dans le contexte c.

Factorisation de la Matrice de Co-occurrence



- $N(m, c)$ est le nombre d'occurrences du mot m dans le contexte c.



- on a alors $N(m, c) \approx Embedding(m) \cdot Embedding(c)$

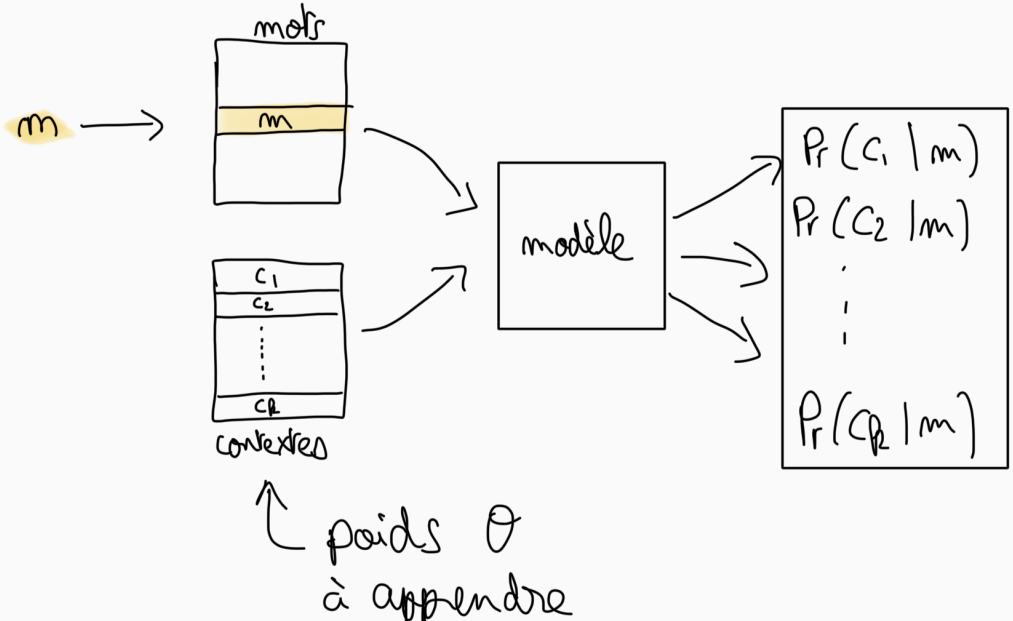
Autre méthode : à la Word2Vec (Mikolov et al., 2013, Google).

But : prédire le contexte à partir du mot.

$P(\text{contexte} \mid \text{mot})$

On veut maximiser la probabilité d'observer le contexte c sachant le mot m, i.e.,

$$\max L(\theta) = \max \prod_{(c,m) \in Data} P(c|m)$$

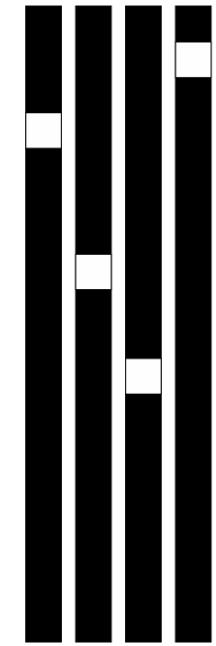


Avec Keras

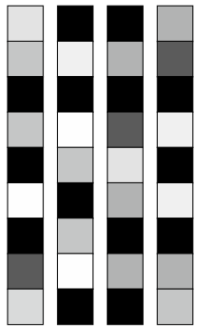
On utilise le layer `Embedding` de Keras pour apprendre des embeddings de mots.

Différences entre one-hot et embeddings

- On a maintenant des vecteurs avec des flottants.
  - Exemple : “chat” = [0.2, 0.5, 0.1, 0.3, 0.7, 0.8, 0.9, 0.4].
- En pratique, la taille des embeddings est de l’ordre de la centaine.



One-hot word vectors:
- Sparse
- High-dimensional
- Hardcoded



Word embeddings:
- Dense
- Lower-dimensional
- Learned from data

Figure 11.2 Word representations obtained from one-hot encoding or hashing are sparse, high-dimensional, and hardcoded. Word embeddings are dense, relatively low-dimensional, and learned from data.

Crédit : livre de François Chollet “Deep Learning with Python”.

La relation géométrique entre les embeddings reflète la relation sémantique entre les mots.

De nombreuses relations sémantiques et syntaxiques entre les mots sont (presque) linéaires dans l'espace vectoriel des mots. Par exemple "wolf" = "dog" + ("tiger" - "cat").

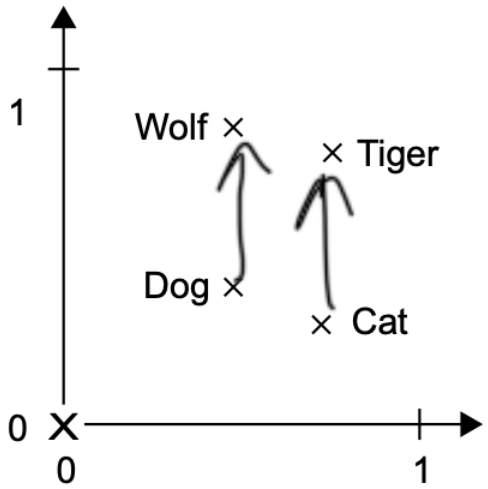


Figure 11.3 A toy example of a word-embedding space

Crédit : livre de François Chollet "Deep Learning with Python".

Il y a deux manières d'utiliser des embeddings sémantiques dans un modèle

On utilise le layer `Embedding` de Keras pour:

- récupérer des embeddings pré-entraînés (ex: Word2Vec, GloVe, FastText, etc.). Les poids du layer `Embedding` sont initialisés avec les embeddings pré-entraînés et sont freezés (non entraînables).
- entraîner des embeddings à partir de zéro sur un corpus spécifique, avec un layer d'embedding dans un modèle de deep learning. Les poids du layer `Embedding` sont alors initialisés aléatoirement et sont appris lors de l'entraînement du modèle.

Session de coding

Utiliser des embeddings de mots pré-entraînés couplé avec un modèle séquentiel LSTM pour classer des critiques de films IMDB.

L'architecture des transformers

Attention is All You Need (Vaswani et al., 2017)

Les **transformers** ont révolutionné le domaine du NLP en proposant une nouvelle architecture basée sur un mécanisme simple : **l'attention**.

Attention : une idée naturelle

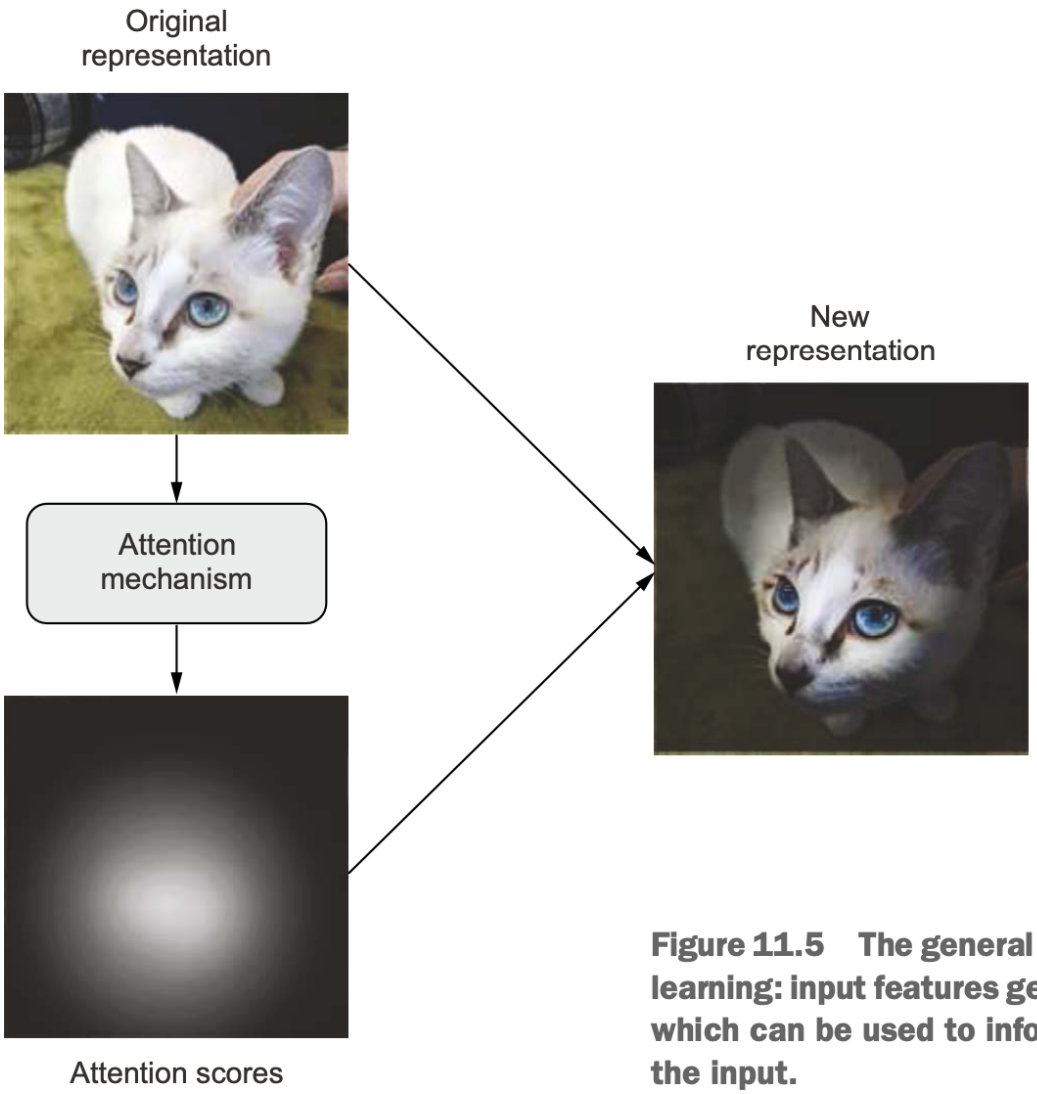


Figure 11.5 The general concept of "attention" in deep learning: input features get assigned "attention scores," which can be used to inform the next representation of the input.

Lisons le papier

- **Attention is All You Need** (Vaswani et al., 2017) :
  - https://arxiv.org/abs/1706.03762

Session de coding

Utiliser un modèle pré-entraîné BERT pour classer des critiques de films IMDB avec `sentence_transformers` de Hugging Face.