

Python pour le traitement du langage naturel

Redha Moulla

Paris, 1 - 3 juillet 2024

Plan

- Introduction au NLP
- Techniques statistiques pour le NLP
- Machine learning pour le NLP
- Deep learning pour le NLP
- Transformers et LLMs
- Enjeux éthiques

Introduction au NLP

1956 : conférence de Dartmouth

L'histoire du NLP est intimement liée à celle de l'intelligence artificielle. Il figure même dans le programme de la conférence de Dartmouth, qui a fondé l'IA.

Articles

AI Magazine Volume: 21 Number 4 (2006) (© AAAI)

A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence

August 31, 1956

John McCarthy, Marvin L. Minsky,
Nathaniel Rochester,
and Claude E. Shannon

The 1956 Dartmouth summer research project on artificial intelligence was initiated by this August 31, 1955 proposal, authored by John McCarthy, Marvin Minsky, Nathaniel Rochester, and Claude E. Shannon. The original typescript contained 17 pages of text, plus one page of references, and was typed in the offices of Dartmouth College and Massachusetts Institute of Technology. The first two pages state the purpose of the meeting, the names of the invited participants, and the time and location of the four proposed research projects. The remaining 15 pages describe the details of each project itself, along with the short descriptions of the other three projects.

The Dartmouth meeting was the first step in a series of projects that eventually led to the development of computer programs that could play checkers, chess, and other games, learn abstractions and concepts, and solve kinds of problems more普遍 than those that had been posed and impeded themselves. We think that such an achievement can be made only if a number of theoretical problems of a carefully selected group of scientists work on it together for a substantial period of time. This paper describes some aspects of the early history of the Dartmouth meeting and the early stages of the artificial intelligence problem.

1. Automatic Computers

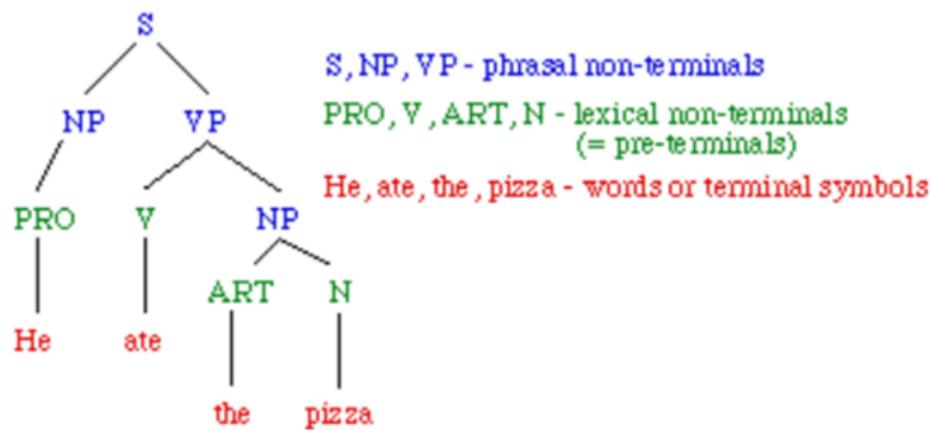
If a machine can do a job, then an automatic computer can do it.

We propose that a 2 month study of artificial intelligence be carried out during the summer of 1956 at Dartmouth College in Hanover, New Hampshire. The purpose of this study would be to test the conjecture that every aspect of intelligence can in principle be precisely described and that every aspect of intelligence can be made to work if it attains an optimum level. It will be made to find how to make machines use language.

"We propose that a 2-month, 10-man study of artificial intelligence be carried out during the summer of 1956 at Dartmouth College in Hanover, New Hampshire. The study is to proceed on the basis of the conjecture that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it. An attempt will be made to find how to make machines use language, form abstractions and concepts, solve kinds of problems now reserved for humans, and improve themselves. We think that a significant advance can be made in one or more of these problems if a carefully selected group of scientists work on it together for a summer."

1970: approches traditionnelles du NLP

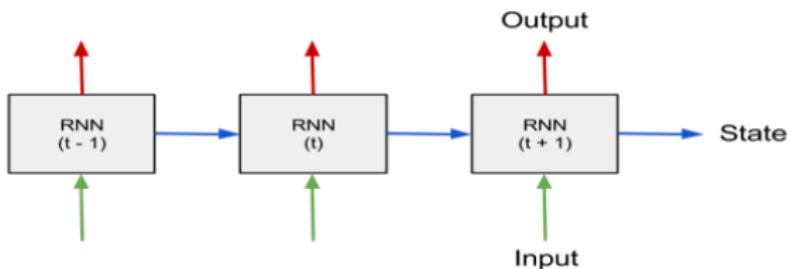
Les modèles traditionnels reposent majoritairement sur une approche linguistique. Ils consistent généralement à parser le langage naturel, même quand celui-ci possède une structure complexe.



1986: Les réseaux de neurons récurrents

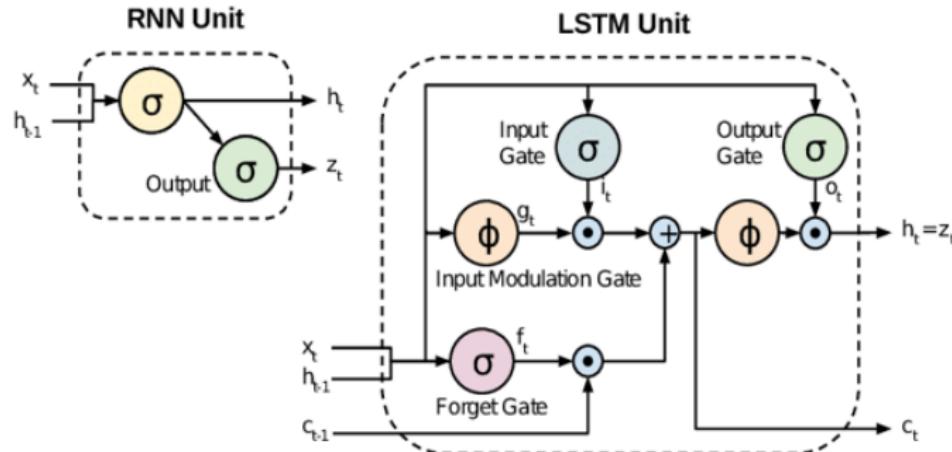
Les RNNs permettent de traiter des séquences ; ils sont ainsi naturellement adaptés au langage naturel. Ils ont cependant rapidement montré des limites.

We describe a new learning procedure, back-propagation, for networks of neurone-like units. The procedure repeatedly adjusts the weights of the connections in the network so as to minimize a measure of the difference between the actual output vector of the net and the desired output vector. As a result of the weight adjustments, internal 'hidden' units which are not part of the input or output come to represent important features of the task domain, and the regularities in the task are captured by the interactions of these units. The ability to create useful new features distinguishes back-propagation from earlier, simpler methods such as the perceptron-convergence procedure¹.



1997: Long Short-Term Memory

Les LSTMs sont nés pour dépasser les limites des RNNs, en particulier le problème de l'annulation du gradient.



2010s: Renaissance du deep learning

Plusieurs avancées ont été réalisées durant la dernière décennie en *deep learning*, notamment depuis 2012 (AlexNet).

- 2013 : représentations distribuées et Word2Vec
- 2014 : Émergence du mécanisme de self-attention
- 2017 : Émergence des modèles Transformers
- 2018 : ELMo, BERT, GPT.
- 2020 : GPT-3.
- 2022 : ChatGPT, etc.

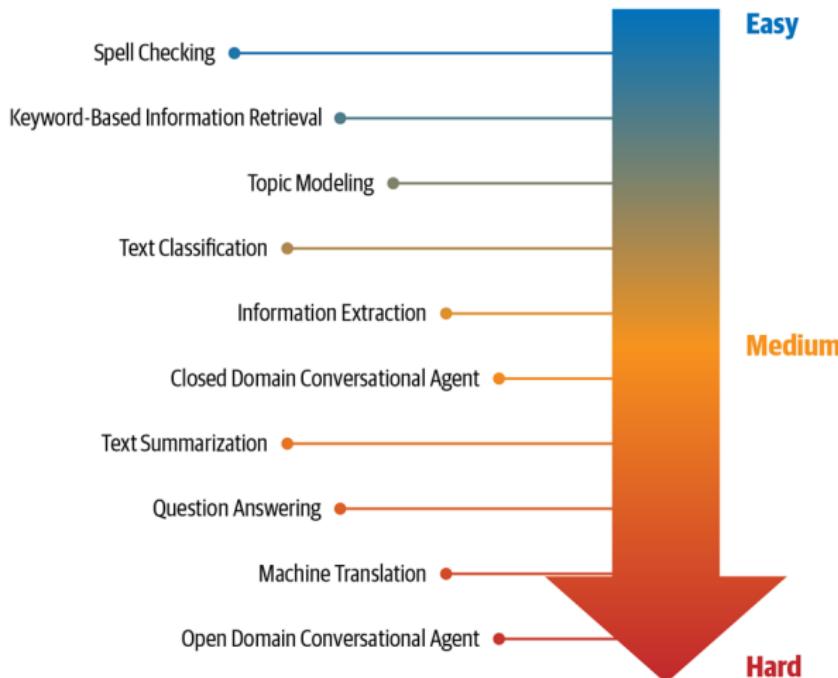
Tâches classiques en NLP

On retrouve dans le NLP un certain nombre de tâches classiques, qui servent à la fois dans les applications pratiques que dans l'évaluation de l'état de l'art.

- Modélisation du langage (*language modeling*)
- Classification de texte
- Extraction d'informations
- Détection de thématiques (*topic modeling*)
- Résumé de texte
- Question/réponse
- Traduction automatique
- Agent conversationnel (domaine ouvert ou fermé)

Classement des tâches en NLP par ordre de difficulté

Le classement ci-dessous est à titre indicatif et sujet à des changements. Certaines tâches, comme la traduction, paraissent aujourd'hui moins difficiles qu'il y a quelques années.



Pourquoi le NLP est si difficile ?

- Le langage naturel est **ambigu**
 - “Acheter un avocat”
- Le langage naturel repose sur le **sens commun**
 - Il fait froid *en hiver*
 - Le beurre fond dans le four
- Le langage naturel n'est pas basé sur des **règles**
 - *En bleu adorable fleurit
Le toit de métal du clocher. Alentour
Plane un cri d'hirondelles, autour
S'étend le bleu le plus touchant. Le soleil
Friedrich Hölderlin*

Pré-traitement des données textuelles

Les données textuelles sont non structurées. Les principales étapes de pré-traitement impliquent :

- ① Segmentation des phrases et **tokenisation** ;
- ② Suppression de **stop words**, de la ponctuation, etc.
- ③ **Stemming**, lemmatisation, conversion des majuscules, etc.

Et

- Détection du langage, POS tagging, etc.

Tokenisation

La tokenisation consiste à segmenter une phrase en unités plus petites (*tokens*), qui sont généralement des mots, mais qui peuvent également être des caractères ou ensembles de caractères.

Exemple:

- "Il fait beau aujourd'hui à Paris" → ["il", "fait", "beau", "aujourd'hui", "à", "Paris"]
- "Paris" → ["P", "a", "r", "i", "s"]

La tokenisation par subword est utilisée d'une manière assez fréquente en **deep learning**.

Stemming and lemmatisation

- Le **stemming** est l'opération qui consiste à supprimer les suffixes et réduire les mots à leur forme de base.
 - marcher → march
 - marchait → march
 - marcha → march
 - marchassiez → march
 - marché → march
- La **lemmatisation** est l'opération qui consiste à réduire un mot à sa racine.
 - nous → nous
 - sommes → être
 - venus → venir
 - faire → faire
 - les → le
 - marchés → marché

Vectorisation

- **One-Hot encoding:** Étant donné un vocabulaire V , chaque mot est représenté par un vecteur binaire (à 0 ou 1) de dimension V .
 - $V = \{Tom, mange, pomme, chien, ciel\}$, où $\dim(V) = 5$
 - Tom = [1, 0, 0, 0, 0]
 - mange = [0, 1, 0, 0, 0]
 - pomme = [0, 0, 1, 0, 0]
 - etc.
- **N-grams**
 - "machine learning est très utilisé en NLP."
 - 1-gram: {machine, learning, est, très, utilisé, en, NLP}
 - 2-gram: {machine learning, learning est, est utilisé, utilisé en, en NLP}

Extraction d'informations

L'information peut concerner différentes entités (événements clés, personnes, lieux, etc.). L'extraction d'information inclut :

- Détection d'entités nommées (NER).
- Extraction de mots clés (KPE).
- Extraction de relations.

Entités nommées (NER)

L'extraction d'entités nommées consiste à détecter des informations clés qui sont des noms propres (noms de personnes, de lieux, etc.)

Example:

Established in 1896, the site was taken over by **Askham Bryan** in 2011 and it has 536 students, including apprentices. A group set up to keep it open had tried to find another college to take it over, after two previous bids were deemed unsuitable. **Tim Whitaker**, chief executive officer and principal at **Askham Bryan College**, said he regretted "the upset" the closure and job losses will cause. "Whilst it was very disappointing that the strategic review didn't receive a sustainable option for **Newton Rigg** campus, we welcome the plans for the preservation of land-based provision in **Cumbria**. "We will support and work with those involved in these plans, to ensure that current students and future applicants interested in land-based courses have a smooth transition." He added that the college had "always been clear" educational provision would not continue at **Newton Rigg** from July 2021, and students, staff and the local community were told in 2020.

Extraction de relations

L'extraction de relations consiste à détecter des relations entre deux entités nommées.

Example:

Established in 1896, the site was taken over by **Askham Bryan** in 2011 and it has 536 students, including apprentices. A group set up to keep it open had tried to find another college to take it over, after two previous bids were deemed unsuitable. **Tim Whitaker**, chief executive officer and principal at **Askham Bryan College**, said he regretted "the upset" the closure and job losses will cause.

"Whilst it was very disappointing that the strategic review didn't receive a sustainable option for **Newton Rigg** campus, we welcome the plans for the preservation of land-based provision in **Cumbria**."

"We will support and work with those involved in these plans, to ensure that current students and future applicants interested in land-based courses have a smooth transition."

He added that the college had "always been clear" educational provision would not continue at **Newton Rigg** from July 2021, and students, staff and the local community were told in 2020

Extraction de mots clés

L'extraction de mots clés consiste à extraire les mots clés les plus importants, permettant de capturer le contenu d'un texte.

Example:

Established in 1896, the site was taken over by **Askham Bryan** in 2011 and it has 536 **students**, including apprentices. A group set up to keep it open had tried to find another college to take it over, after two previous bids were deemed unsuitable. **Tim Whitaker**, chief executive officer and principal at **Askham Bryan College**, said he regretted "the upset" the closure and job losses will cause.

"Whilst it was very disappointing that the strategic review didn't receive a sustainable option for **Newton Rigg** campus, we welcome the plans for the preservation of land-based provision in **Cumbria**."

"We will support and work with those involved in these plans, to ensure that current students and future applicants interested in land-based courses have a smooth transition."

He added that the college had "always been clear" educational provision would not continue at **Newton Rigg** from July 2021, and students, staff and the local community were told in 2020

Part of Speech tagging

Part of speech consiste à détecter la catégorie grammaticale d'un mot au sein d'un texte.

Exemple :

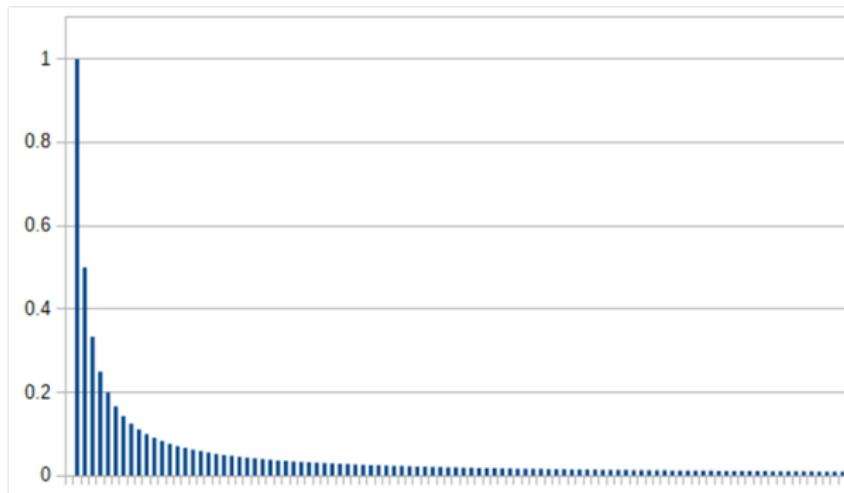
"this is a very simple example"

[('this', 'DT'), ('is', 'VBZ'), ('a', 'DT'), ('very', 'RB'), ('simple', 'JJ'), ('example', 'NN')]

Techniques statistiques pour le NLP

Principe des techniques statistiques pour le NLP

- Un document est caractérisé par la distribution des mots qui le composent.
- La distribution des mots obéit généralement à une loi de puissance (ou de Zipf). La majorité du langage que nous utilisons ordinairement est composé d'un nombre très limité de mots.



Term Frequency

- Term Frequency (TF): la fréquence avec laquelle un terme t apparaît dans un document d .

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

Exemple : étant donné 1000 mots.

Terme	Fréquence	TF score
chat	15	0.015
lait	4	0.004
mange	6	0.006
feu	3	0.003
...

Inverse Document Frequency

- Inverse Document Frequency (IDF): permet de pénaliser les termes les plus fréquents et de donner plus de poids aux termes les moins fréquents (plus parlants).

$$\text{IDF}(t) = \log_e \left(\frac{\text{Total number of documents in the corpus}}{\text{Number of documents with term } t \text{ in them}} \right)$$

Example: given 100 documents

Terme	Nombre de documents	IDF score
chat	70	0.36
lait	9	2.41
manger	80	0.22
feu	5	3.00
...

Term Frequency – Inverse Document Frequency

- TF-IDF score est le produit de Tf et IDF.

$$\text{TF-IDF Score} = \text{TF Score} \times \text{IDF score}$$

Terme	TF score	IDF score	TF-IDF score
chat	0.015	0.36	0.0054
lait	0.004	2.41	0.0096
manger	0.006	0.22	0.0013
feu	0.003	3.00	0.0090
...

Topic Modeling

- Topic modeling is a technique for:
"automatically organizing, understanding, searching and summarizing large electronic archives." - David Blei
- Découvrir des thématiques sous-jacentes à un document ou un ensemble de documents (Google News, etc.)
- Annoter ou caractériser les documents selon les thématiques.

Machine learning

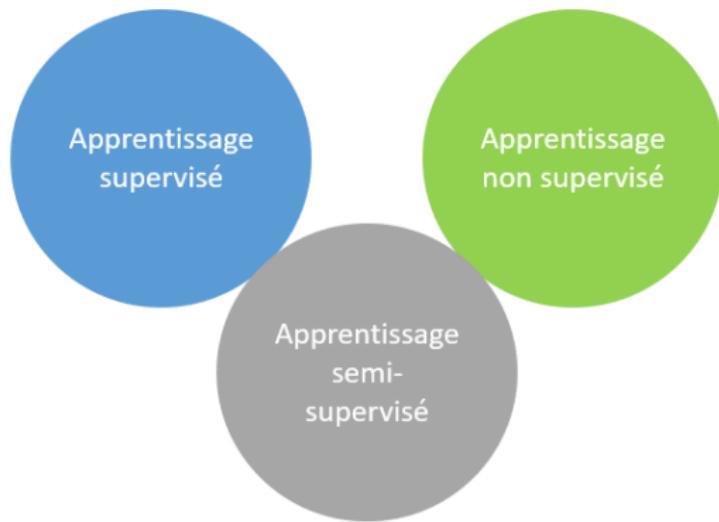
Définition de l'apprentissage automatique

L'apprentissage automatique est une branche de l'intelligence artificielle qui consiste à doter les machines de la capacité d'apprendre à partir de données sans que celles-ci ne soient explicitement programmées pour exécuter des tâches spécifiques.

Le machine learning englobe plusieurs types d'apprentissage :

- **Supervisé** : Les algorithmes apprennent à partir de données étiquetées pour faire des prédictions ou classifications.
- **Non supervisé** : L'apprentissage est effectué sur des données non étiquetées pour trouver des structures cachées.
- **Semi-supervisé** : Combine des éléments des deux premiers types en utilisant une petite quantité de données étiquetées et une grande quantité de données non étiquetées.
- **Par renforcement** : Les modèles apprennent à prendre des décisions en maximisant une récompense à travers des interactions.

Typologies d'apprentissage automatique



L'apprentissage supervisé

L'apprentissage supervisé consiste à apprendre un modèle qui associe une étiquette (*label*) à un ensemble de caractéristiques (*features*).

- **Inputs** : un jeu de données *annotées* pour entraîner le modèle.
 - Exemple : des textes (tweets, etc.) avec les *sentiment* associés, positifs ou négatifs.
- **Output** : une étiquette pour un point de donnée inconnu par le modèle.

L'apprentissage supervisé se décline lui-même en deux grandes familles :

- **La classification** : prédire une catégorie ou une classe.
 - Exemple : prédire l'étiquette d'une image (chat, chien, etc.), le sentiment associé à un texte, le centre d'intérêt d'un client à partir de ses commentaires, etc.
- **La régression** : prédire une valeur continue (un nombre réel typiquement).
 - Exemple : prédire le prix d'un appartement, la lifetime value d'un client, etc.

L'apprentissage non supervisé

L'apprentissage non supervisé se réfère à l'utilisation de modèles d'apprentissage automatique pour identifier des patterns et des structures dans des données qui ne sont pas étiquetées.

Principales typologies de l'apprentissage non supervisé :

- **Clustering** : Regroupement de points de données similaires ensemble.
Exemple : segmentation de marché, regroupement social.
- **Réduction de dimensionnalité** : Techniques pour réduire la dimension des données tout en préservant leur structure. Exemple : visualisation de données en grande dimension.
- **Détection d'anomalies** : Déetecter des observations dont les caractéristiques sont inhabituelles par rapport à la majorité.

L'apprentissage semi-supervisé

L'apprentissage semi-supervisé combine des éléments des approches supervisées et non supervisées. Il utilise un petit ensemble de données étiquetées et un plus grand ensemble de données non étiquetées pour former des modèles.

Cette méthode est particulièrement utile quand :

- Les données étiquetées nécessitent des ressources coûteuses pour les obtenir, mais les données non étiquetées sont abondantes.
- L'ajout d'un peu d'information étiquetée peut améliorer significativement la performance de modèles entraînés avec des données non étiquetées.

Les applications typiques incluent :

- Développement de systèmes de recommandation plus performants.
- Traitement de langage naturel et analyse de sentiment lorsque les annotations complètes ne sont pas disponibles.

L'apprentissage tente d'exploiter "le meilleur des deux mondes" de l'étiquetage et de la découverte de structure.

L'apprentissage par renforcement

L'apprentissage par renforcement est une approche de l'apprentissage automatique où un agent apprend à prendre des décisions en interagissant avec un environnement. L'objectif est de maximiser une récompense cumulative.

Principes clés de l'apprentissage par renforcement :

- **Exploration vs Exploitation** : L'agent doit explorer l'environnement pour découvrir des actions qui maximisent la récompense, tout en exploitant ses connaissances actuelles pour prendre des décisions avantageuses.
- **Politique** : Une stratégie qui guide l'agent à choisir une action à partir d'un état donné.
- **Récompense** : Un signal immédiat reçu après chaque action, qui aide à évaluer la performance.
- **Valeur** : Une estimation de la récompense future attendue, tenant compte des récompenses passées et actuelles.

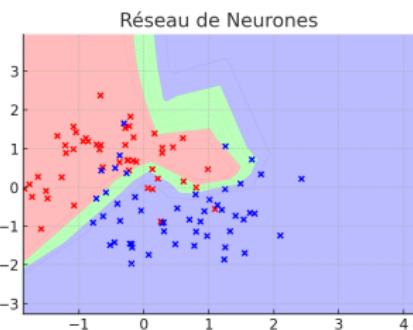
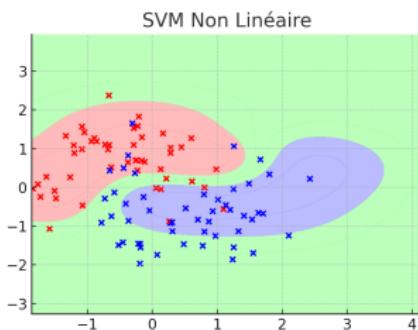
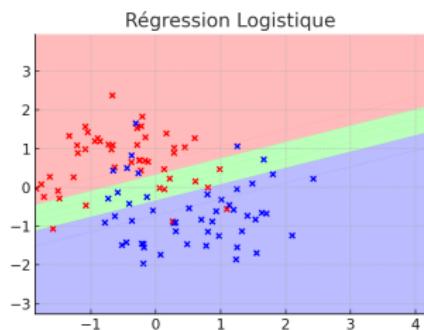
L'apprentissage par renforcement est largement utilisé dans les domaines tels que la robotique, les jeux vidéos, les enchères, la recommandation, etc.

Apprentissage supervisé

Formalisation du problème de l'apprentissage supervisé

Comment choisir un modèle parmi tous les modèles possibles ?

- A priori sur la classe du modèle (biais inductif).
- Minimisation du risque empirique.



Principes de l'apprentissage supervisé

La construction d'un modèle en apprentissage supervisé repose sur trois piliers essentiels qui guident le processus d'apprentissage et déterminent sa réussite :

- **Données** : L'ensemble d'exemples étiquetés utilisés pour l'entraînement du modèle. La qualité, la quantité et la représentativité de ces données sont cruciales pour la capacité du modèle à apprendre et à généraliser à de nouvelles observations.
- **Fonction objective** : Aussi connue sous le nom de fonction de perte ou de coût, elle quantifie l'erreur entre les prédictions du modèle et les valeurs réelles. L'objectif de l'apprentissage est de minimiser cette fonction, guidant ainsi le modèle vers une meilleure performance.
- **Hypothèses (biais inductif)** : Les hypothèses préalables sur la forme du modèle et les relations dans les données. Ces hypothèses concernent le choix de l'algorithme d'apprentissage, et toute préconception sur la distribution des données. Ces hypothèses dirigent l'espace de recherche des solutions possibles et influencent directement la capacité du modèle à apprendre et à généraliser.

Biais inductif

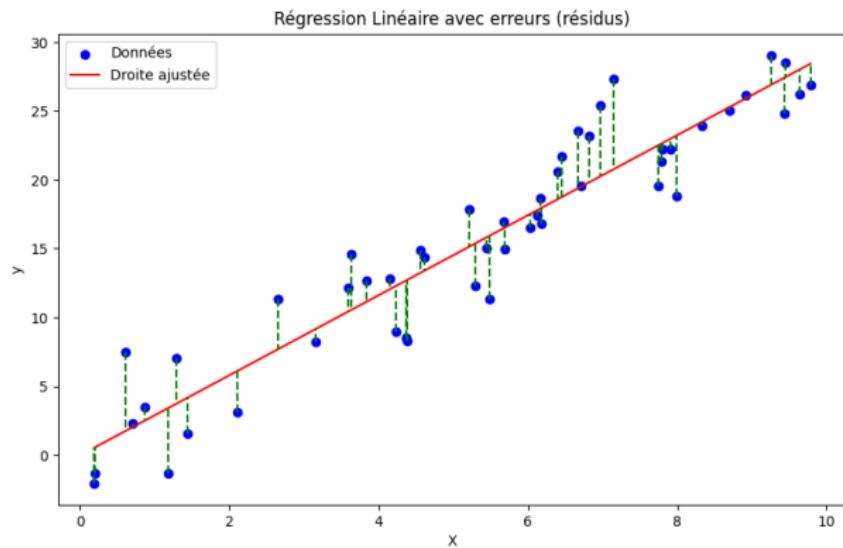
Le biais inductif est l'ensemble des a priori (hypothèses) qui déterminent la classe de modèles à laquelle appartient la fonction f . Ces hypothèses traduisent les connaissances ou les biais de la personne qui construit le modèle.

Exemple de biais inductif :

- Hypothèse de linéarité.
- Hypothèse de proches voisins.
- Hypothèse de maximum de marge.
- Hypothèse d'équivariance par translation.
- Etc.

Minimisation du risque empirique

Une fois la famille de modèles est choisie, un modèle peut être déterminé en minimisant l'erreur de prédiction sur le jeu de données d'entraînement.



Fonctions de coût

Le choix de la fonction de coût dépend de la nature du problème et des données.
On distingue deux types de fonctions de coût.

Régression

- L'erreur quadratique : $L(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$
- L'erreur absolue : $L(\hat{y}, y) = |\hat{y} - y|$

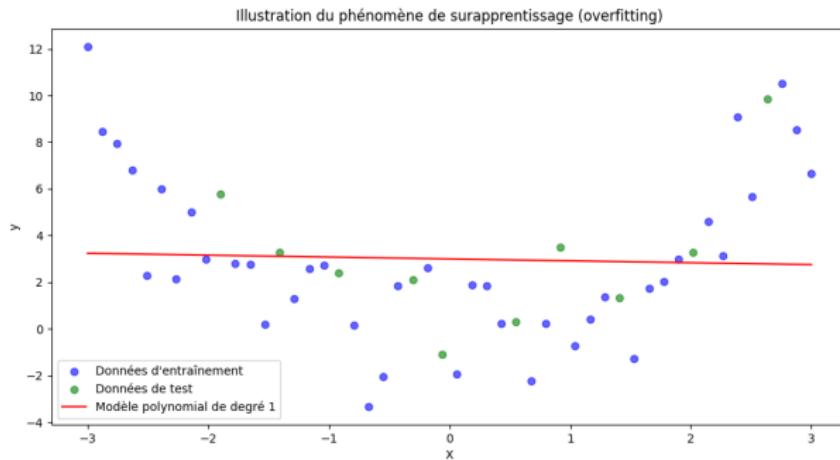
Classification

- Entropie croisée : $L(\hat{y}, y) = -y\log(\hat{y}) - (1 - y)\log(1 - \hat{y})$
- L'erreur Hinge : $L(\hat{y}, y) = \max(0, 1 - \hat{y}y)$

Sous-apprentissage

Definition

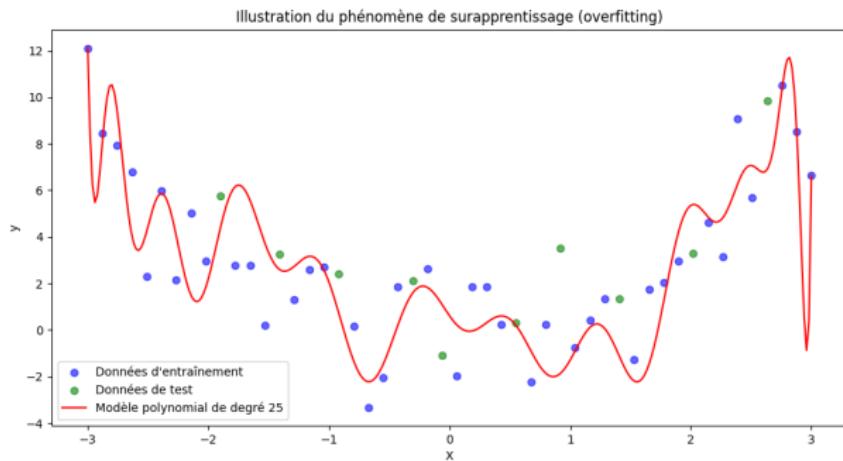
On dit qu'un modèle de machine learning est en régime de sous-apprentissage (underfitting) lorsqu'il n'arrive pas à capturer la complexité (l'information) présente dans le jeu de données d'entraînement.



Sur-apprentissage

Definition

On dit qu'un modèle de machine learning est en régime de sur-apprentissage (overfitting) lorsqu'il n'arrive pas à généraliser à des données non encore observées, i.e. lorsqu'il est trop adapté aux données d'entraînement.



Sélection de modèle

Pour sélectionner le modèle le plus pertinent par rapport à une métrique donnée, on applique la méthodologie suivante :

- On partitionne le jeu de données disponible en trois parties : un jeu d'entraînement, un jeu de validation et un jeu de test.
- On entraîne M modèles sur le jeu d'entraînement.
- On évalue les performances respectives des M modèles sur le jeu de validation et on sélectionne le meilleur.
- Le modèle sélectionné est ensuite évalué sur le jeu de test. Idéalement, le jeu de test est ainsi utilisé une seule fois.

Validation croisée K-fold

La validation croisée est une méthode plus robuste pour évaluer les performances des modèles. Il y a deux manières d'appliquer une validation croisée : K-fold et Leave-One-Out (LOO).

La validation croisée K-fold s'effectue selon la méthodologie suivante :

- On partitionne le jeu de données \mathcal{D} en K parties ayant approximativement la même taille.
- Pour chaque partie \mathcal{D}_k , on entraîne le modèle sur l'ensemble des données restantes. Ce modèle est ensuite évalué sur \mathcal{D}_k .
- On considère la moyennes des performances du modèles sur les différents \mathcal{D}_k .

La validation croisée K-fold permet ainsi de prendre en compte la variabilité qu'il peut y avoir dans les données. Par ailleurs, on peut également avoir une estimation de la variance du modèle.

Validation croisée Leave-One-Out

La validation croisée Leave-One-Out est un cas particulier de la validation croisée k-fold où le nombre de parties (folds) est également au nombre d'observations ($K = n$).

La validation croisée LOO s'effectue selon la méthodologie suivante :

- On partitionne le jeu de données \mathcal{D} en n parties ayant chacune $n - 1$ observations.
- On entraîne le modèle sur chacune des parties \mathcal{D} (ayant $n-1$ observations) et on l'évalue sur l'observation restante.
- On considère la moyennes des performances du modèles sur les différents \mathcal{D}_k .

La validation croisée LOO présente l'avantage que les modèles sont entraînés sur des jeux de données d'une taille plus grande. Par ailleurs, le nombre de modèles obtenus est plus grand également, ce qui peut plaider pour davantage de robustesse. Cependant, ces modèles sont entraînés sur des jeux de données plus similaires les uns par rapport aux autres. D'autre part, les jeux de test étant composés d'une seule observation, les performances risquent de présenter variabilité plus grande.

Bootstrap

Le Bootstrap est une technique d'évaluation de modèle qui consiste à partitionner le jeu de données \mathcal{D} en K échantillons $\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_K$. Chaque jeu de données est obtenu en tirant n observations avec remise.

La performance du modèle est alors obtenue en moyennant sur ses différentes performances sur les K jeux de données.

Métriques de performance : régression

On dispose d'un certain nombre de métriques pour évaluer les performances des modèles de machine learning. Celles-ci peuvent être divisées en deux catégories.

Régression

- L'erreur quadratique moyenne (MSE) : elle est définie comme la moyenne des carrés des écarts entre les prédictions et les valeurs observées.

$$MSE = \frac{1}{n} \sum_{i=1}^n (f(x_i) - y_i)^2$$

- La racine carrée de l'erreur quadratique moyenne (RMSE) :

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (f(x_i) - y_i)^2}$$

Métriques de performance : classification 1/2

Accuracy : L'accuracy est la métrique de base qui permet d'évaluer les performances d'un modèle de classification. Elle est définie comme :

$$\text{Accuracy} = \frac{\text{Nombre de prédictions correctes}}{\text{Nombre total de prédictions}}$$

Matrice de confusion : La matrice de confusion est une représentation permettant d'offrir plus de finesse par rapport à l'accuracy, notamment quand le jeu de données est déséquilibré (présence de classes majoritaires). Elle compare les prédictions du modèle avec les valeurs réelles et est structurée comme suit :

		Valeur Prédite	
		Positif	Négatif
Valeur Réelle	Positif	Vrai Positif (VP)	Faux Négatif (FN)
	Négatif	Faux Positif (FP)	Vrai Négatif (VN)

Métriques de performance : classification 1/2

A partir de la matrice de confusion, on peut dériver d'autres métriques :

- Précision : elle est définie comme la proportion des prédictions correctes parmi toutes les prédictions positives :

$$\text{Précision} = \frac{VP}{VP + FP}$$

- Rappel (recall) : il représente la proportion des vrais positifs correctement prédits par le modèle.

$$\text{Rappel} = \frac{VP}{VP + FN}$$

- Score F1 (F1-score) : Le score F1 est défini comme la moyenne harmonique de la précision et du rappel.

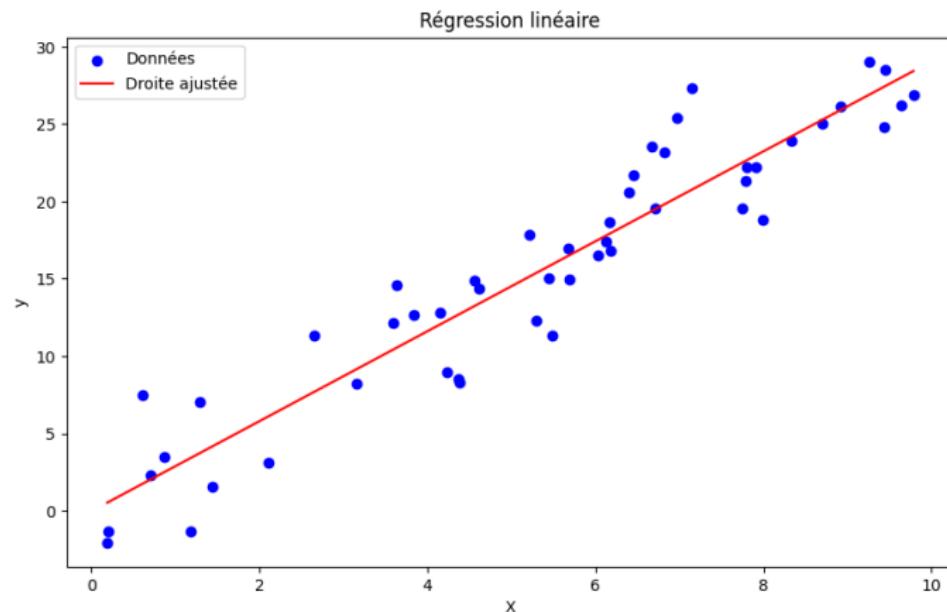
$$\text{Score F1} = 2 \frac{\text{Précision} \times \text{Rappel}}{\text{Précision} + \text{Rappel}}$$

Modèles classiques de machine learning

Régression linéaire simple

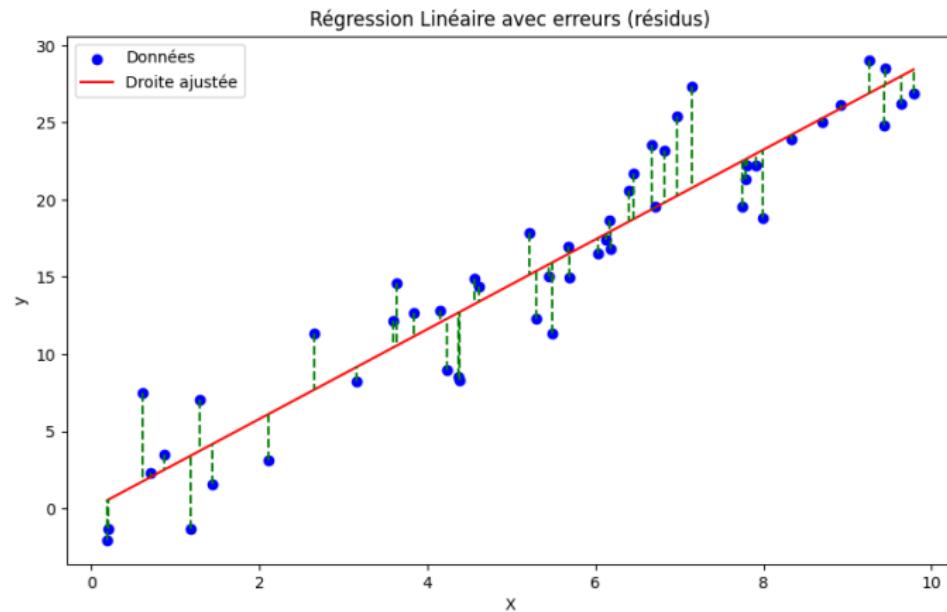
Soit un ensemble de n observations x_1, x_2, \dots, x_n avec les labels correspondants y_1, y_2, \dots, y_n , on cherche le modèle linéaire qui ajuste le mieux ces données.

$$\hat{y} = \beta_0 + \beta_1 x$$



Minimisation du risque empirique 1/2

L'erreur de prédiction pour la i ième observation est : $e_i = y_i - \hat{y}_i$. où $\hat{y}_i = \beta_0 + \beta_1 x_i$.



Minimisation du risque empirique 2/2

Déterminer un modèle de régression linéaire simple revient à minimiser les erreurs de prédiction (risque empirique) :

$$\sum_{i=1}^n (y_i - \hat{y}_i)^2$$

Autrement dit, chercher les coefficients β_j qui minimisent le risque empirique :

$$\arg \min_{\beta_0, \beta_1} \left(\sum_{i=1}^n (y_i - \beta_0 - \beta_1 x_i)^2 \right)$$

Régression linéaire multiple

On considère n observations X^1, X^2, \dots, X^n où chaque observation X^i est désormais un vecteur ayant p composantes (p variables explicatives).

$$X^i = \begin{pmatrix} x_1^i \\ x_2^i \\ \vdots \\ x_p^i \end{pmatrix}$$

La régression linéaire s'écrit alors :

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_p x_p$$

Les coefficients $\beta_0, \beta_1, \dots, \beta_p$ sont déterminés par la méthode des moindres carrés :

$$\arg \min_{\beta_0, \beta_1, \dots, \beta_p} \sum_{i=1}^n \left(y^i - (\beta_0 + \sum_{j=1}^p \beta_j x_j^i) \right)^2$$

Régression logistique : introduction

La régression logistique est une technique d'analyse statistique utilisée pour modéliser la probabilité d'une variable dépendante binaire. C'est un cas particulier de modèle linéaire généralisé qui est utilisé pour des problèmes de classification.

Principes de la régression logistique :

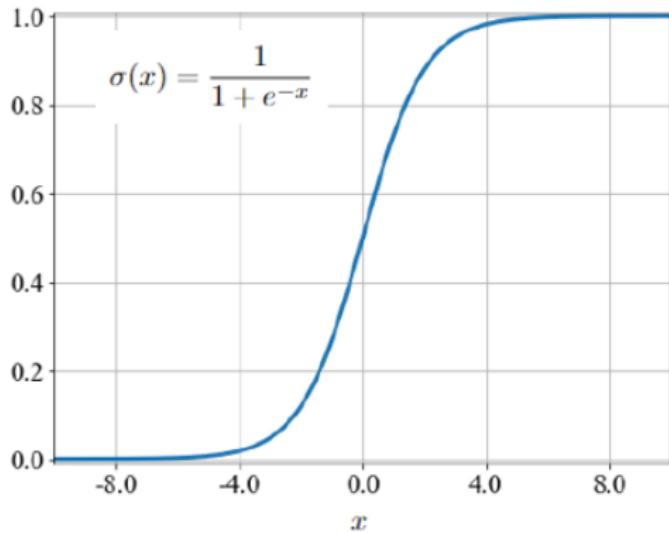
- **Variable dépendante** : On cherche la probabilité que la variable dépendante (y) appartienne à une classe (0 ou 1, vrai ou faux, succès ou échec). Autrement dit, on cherche à modéliser $P(y = 1)$ en fonction des variables dépendantes (explicatives) x .
- **Odds ratio** : Plus concrètement, on cherche à exprimer la côte anglaise (odd ratio) en fonction des variables dépendantes (x).

$$\ln \frac{p(\mathbf{x})}{1 - p(\mathbf{x})} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_p x_p$$

Régression logistique : fonction sigmoïde

Après quelques simplifications, on peut écrire la probabilité $p(x)$ (la probabilité pour que y soit un succès par exemple) :

$$p(\mathbf{x}) = \frac{1}{1 + e^{-(\beta^T \mathbf{x})}}$$



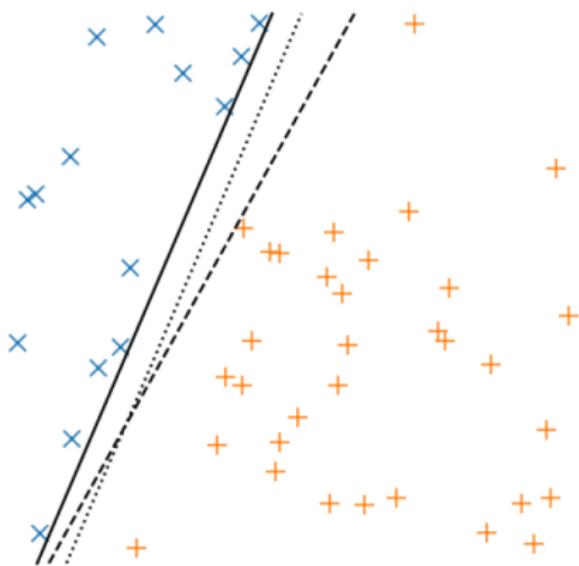
Calcul des coefficients de la régression logistique

Les coefficients de la régression logistique peuvent être calculés en minimisant le risque empirique par rapport à une fonction de coût sous forme d'entropie croisée :

$$\arg \min_{\beta_0, \beta_1, \dots, \beta_p} \left(- \sum_{i=1}^n y_i \ln(\hat{y}_i) + (1 - y_i) \ln(1 - \hat{y}_i) \right)$$

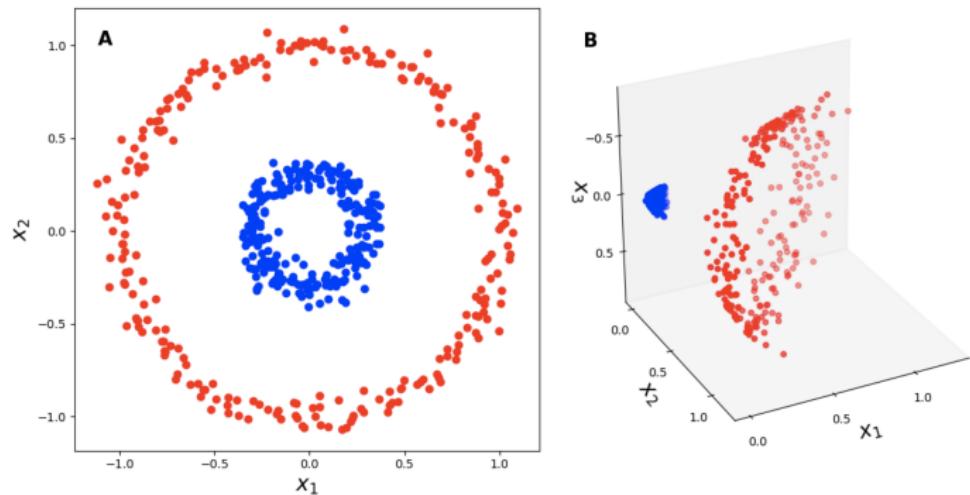
Machines à vecteurs de support (SVM)

Considérons un problème de classification binaire. On recherche l'hyperplan séparateur qui maximise la marge γ entre les deux classes. La marge γ étant définie comme la distance entre cet hyperplan et les observations les plus proches.



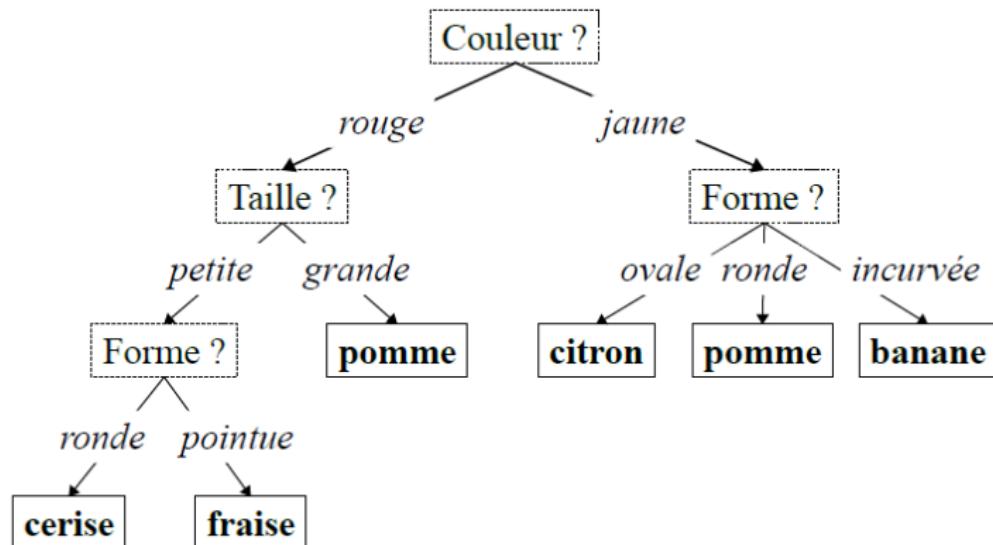
SVM à noyau

On considère un problème de classification non linéaire. L'astuce du noyau consiste à augmenter la dimension du problème, pour le résoudre ensuite avec un séparateur linéaire dans le nouvel espace.



Arbres de décision

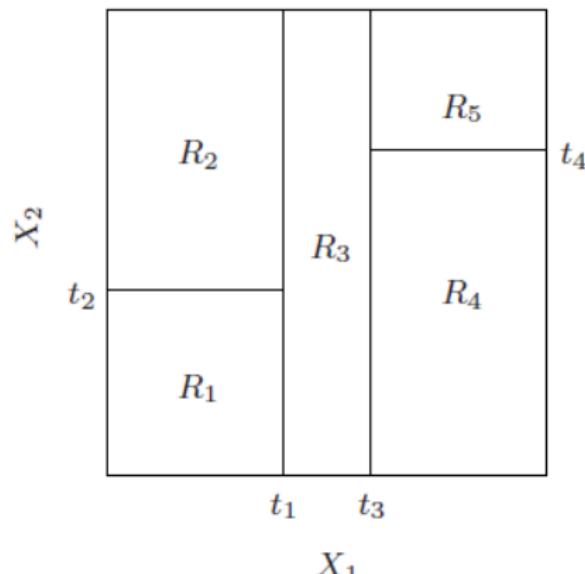
Les arbres de décisions sont des modèles dont le processus de décision est hiérarchique et prend la forme d'un arbre.



Entraînement des arbres de décision

Les arbres de décisions sont généralement entraînés à l'aide de la technique CART (Classification And Regression Trees).

Etant donné un ensemble d'observations $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$, les arbres de décision partitionnent cet espace en plusieurs régions R_1, R_2, \dots, R_m .



Critères d'optimisation

Le critère d'optimisation dépend de la tâche en question.

- **Classification**

- Indice de Gini simplifié : $\sum_{k=1}^K p_{mk}(1 - p_{mk})$
- Entropie croisée : $-\sum_{k=1}^K p_{mk} \ln(p_{mk})$

- **Régression**

$$\sum_{i=n}^n (y_i - f(x_i))^2$$

Forêts aléatoires (random forests)

La technique des forêts aléatoires (random forests) consiste à appliquer une approche de type bagging sur les arbres de décision.

L'algorithme random forests suit cette procédure :

- Tirer par bootstrap B échantillons de tailles n à partir de l'ensemble D .
- Pour chaque échantillon tiré, construire un arbre en répétant les étapes suivantes jusqu'à atteindre n_{min} .
 - Tirer d'une manière aléatoire m variables parmi les p variables.
 - Sélectionner la meilleure variable avec le meilleur point de partitionnement.
 - Partitionner le noeud en deux sous-branches.
- Agréger les arbres construits.

Les prédictions sont agrégées selon qu'il s'agisse de régression ou de classification :

- Régression : moyenne $f^B(x) = \frac{1}{B} \sum_{b=1}^B T_b(x)$
- Classification : vote majoritaire.

Remarques

- L'algorithme de random forests intègre nativement une forme de validation croisée. Les performances mesurées sur $\bigcup_{b_i \neq b_k}$ (out of bag ou OOB) sont souvent proches de celles que l'on pourrait mesurer avec une validation croisée.
- Le nombre de variable tirées pour chaque noeud est généralement donné par \sqrt{p} pour la classification et $\frac{p}{3}$ pour la régression. Cet hyperparamètre dépend cependant du problème considéré.
- Lorsque le nombre de variable est élevé alors que le nombre de variables réellement partinente est faible, la probabilité que les p variables sélectionnées pour chaque partitionnement incluent des variables pertinentes devient faible, et les performances du modèle en termes de généralisation peuvent se détériorer considérablement.
- L'algorithme random forests permet de restituer des informations sur l'importance des variable (feature importance).

Apprentissage non supervisé

Apprentissage non supervisé

Dans l'apprentissage non supervisé, on considère n observations sans labels. On s'intéresse fondamentalement à la probabilité jointe de ces observations.

On peut distinguer deux grandes catégories d'apprentissage non supervisé :

- **Clustering (partitionnement)** : cela consiste à partitionner les n observations en K groupes pertinents (généralement le critère de pertinence a une signification d'un point de vue métier).
- **Réduction de dimension** : il s'agit de trouver une représentation des données originelles dans un nouvel espace de plus petite dimension. Cela peut être effectué à différentes fins : visualisation des données, compression des données, amélioration des performances du modèles (modèles plus robuste, plus explicables, etc.).
- **Détection d'anomalie** : il s'agit de détecter des observations qui présentent un profil (des features) inhabituel par rapport au profil moyen de la majorité des observations.

K-means

L'algorithme de Lloyd tente de regrouper les données en clusters en minimisant les distances entre les points d'un même cluster tout en maximisant les distances entre points appartenant à différents clusters.

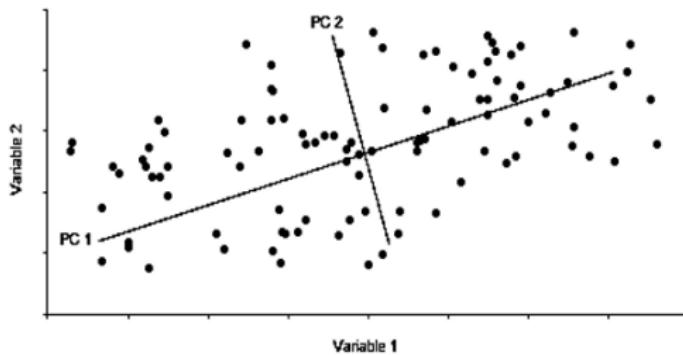


Remarques

- L'algorithme des k-means étant basé sur une distance euclidienne, il est nécessaire de normaliser les données avant de l'exécuter.
- L'algorithme des k-means est très sensible aux données aberrantes (outliers). Il faut donc considérer les données d'une manière attentive. Cependant, cela permet également d'utiliser l'algorithme des k-means pour la détection automatique des outliers.
- Les centroïdes étant initialisés d'une manière aléatoire, les clusters obtenus ne sont pas stables ; les clusters peuvent changer d'une exécution à l'autre. Il existe cependant une variante plus stable, appelée k-means++, qui permet de sélectionner les centroïdes d'une manière semi-aléatoire.
- Il est possible de partitionner les données avec une métrique plus générale que la distance euclidienne. On peut définir un algorithme k-means à noyau sur un espace de Hilbert pour aller au-delà de la métrique euclidienne.
- **K-means n'est pas adapté aux données en grande dimension.**

Analyse en Composantes Principales (ACP)

L'Analyse en Composantes Principales (ACP) est une technique statistique de réduction de dimensionnalité. Elle transforme les données en un nouveau système de coordonnées où la plus grande variance est capturée sur les premiers axes, appelés composantes principales.



Formulation de l'ACP

Soit X une matrice de données de dimension $n \times p$ (n observations, p variables), centrée (moyenne nulle). L'ACP cherche à trouver les vecteurs propres et les valeurs propres de la matrice de covariance $C = \frac{1}{n-1} X^T X$.

La matrice de covariance C peut être décomposée comme suit :

$$C = VLV^T$$

où $V = [u_1, u_2, \dots, u_p]$ est la matrice des vecteurs propres et L est une matrice diagonale des valeurs propres λ_k .

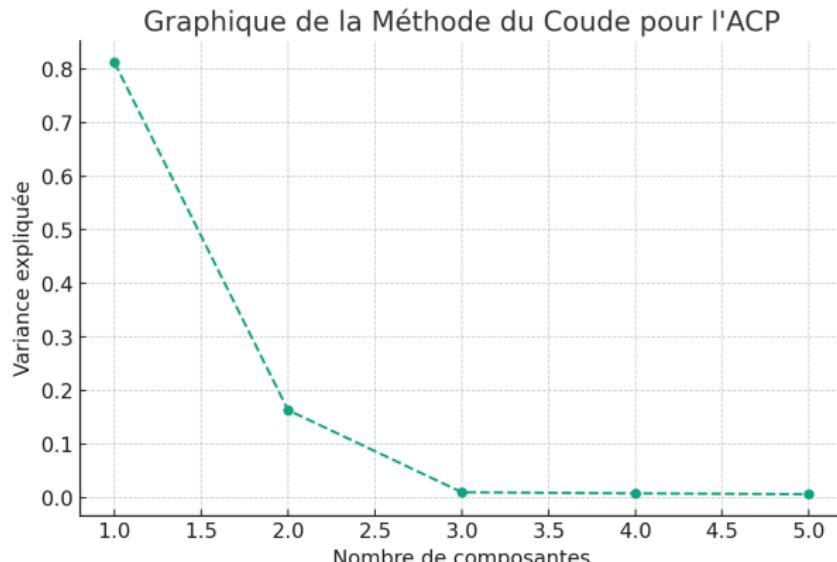
La contribution de chaque composante principale à la variance totale est donnée par :

$$\frac{\lambda_k}{\sum_{i=1}^p \lambda_i}$$

Choix du nombre de composantes principales

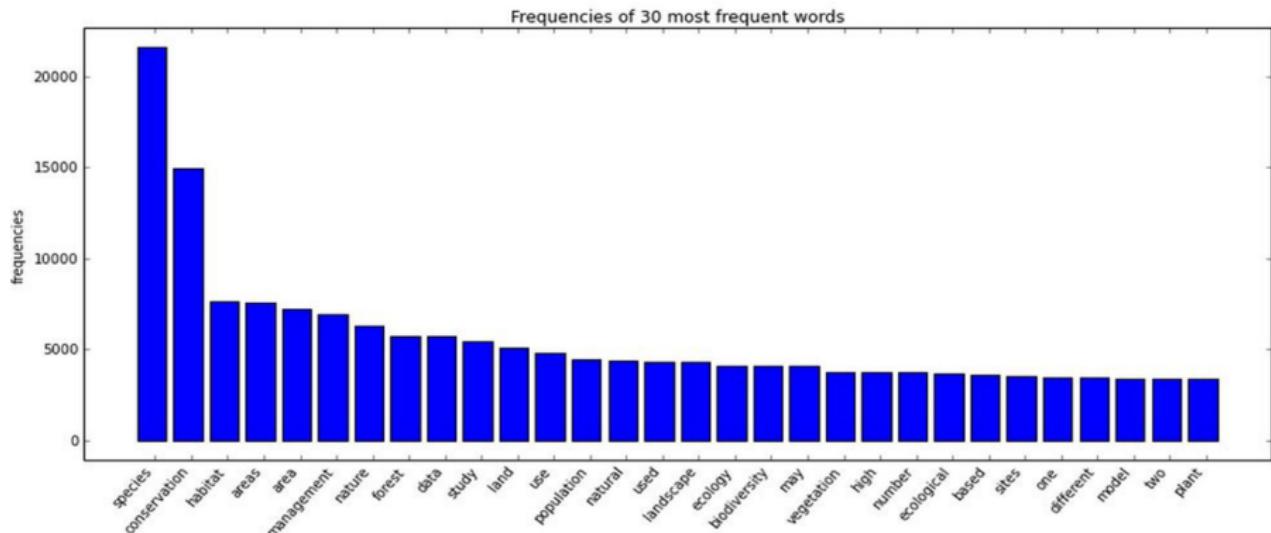
Le nombre de composantes à retenir est déterminé en fonction du pourcentage de variance totale que l'on souhaite expliquer.

On utilise généralement la méthode du coude (Scree plot). Il s'agit d'un graphique montrant la proportion de la variance expliquée en fonction du nombre de composantes.



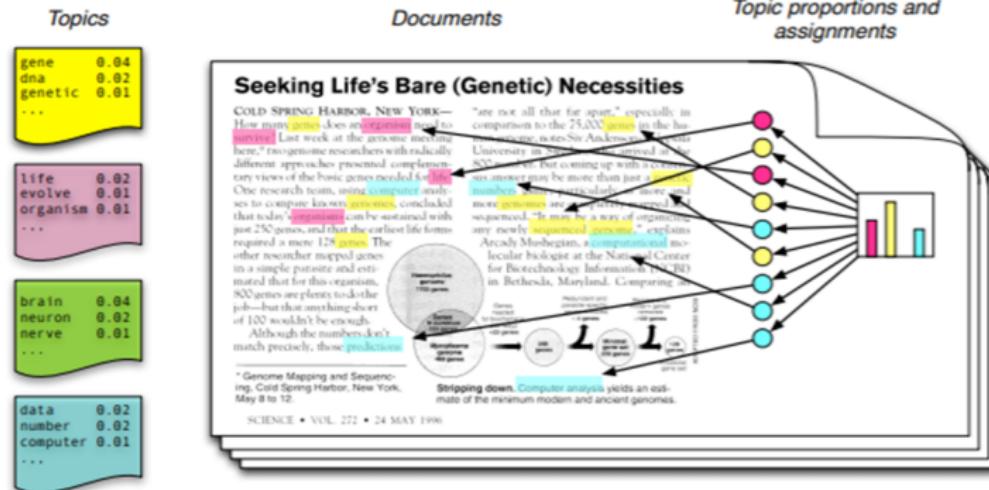
Topic Modeling : thématiques

Une thématique est caractérisée par une distribution de mots (la fréquence de certains mots).



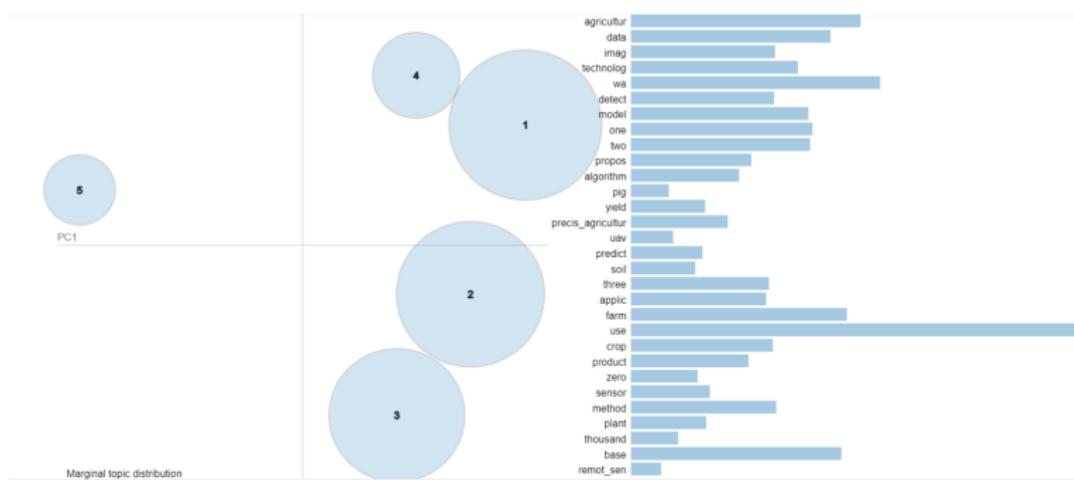
Topic Modeling : documents

Un document est d'une manière générale un mélange de thématiques.



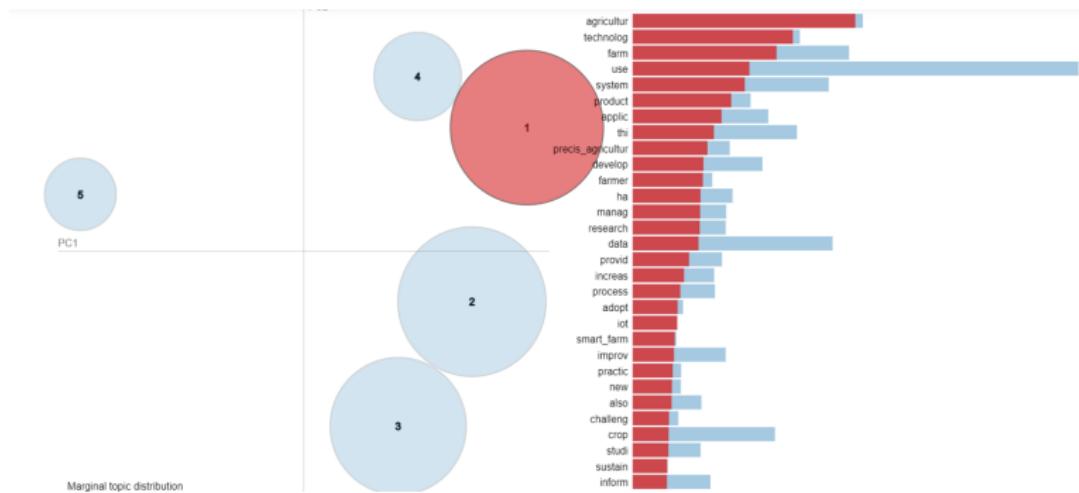
Topic Modeling : LDA

Extraction de thématiques d'un ensemble de documents traitant de l'agriculture
l'aide d'une technique LDA (Latent Dirichlet Allocation).



Topic modeling : exemple d'une thématique

La thématique extraite est définie par un ensemble de mots clés.



Deep learning

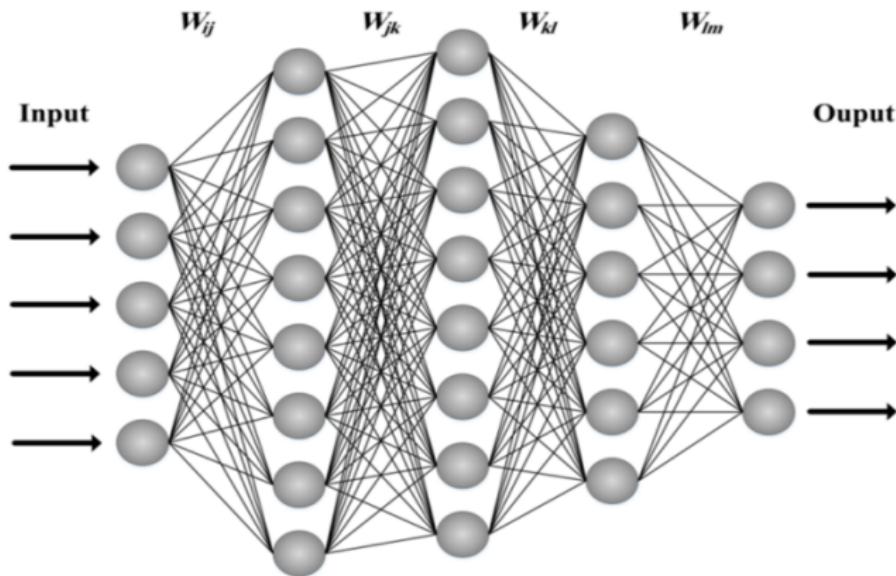
Introduction aux réseaux de neurones

Définition : Les réseaux de neurones sont des modèles computationnels inspirés par le fonctionnement des neurones dans le cerveau humain. Ils sont capables d'apprendre des tâches complexes en modélisant des relations non linéaires entre les entrées et les sorties.

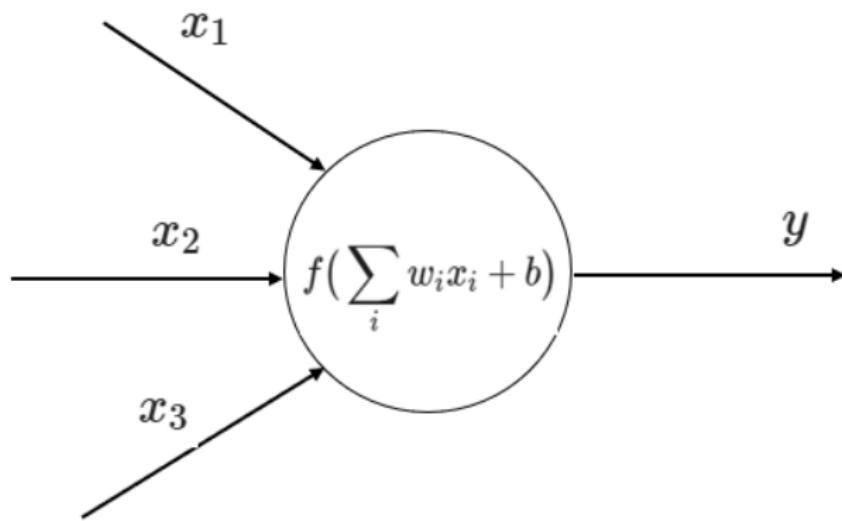
Caractéristiques :

- **Extraction automatique des features** : Capacité d'adaptation et d'extraction des features à partir des données sans programmation explicite.
- **Modélisation non linéaire** : Aptitude à capturer des relations complexes dans les données.
- **Modélisation en grande dimension** : Les modèles de deep learning sont particulièrement adaptés pour les données en grande dimension (images, texte, etc.).
- **Flexibilité** : Applicables à un large éventail de tâches et de typologies de données (images, langage naturel, données graphiques, etc.).

Illustration d'un réseau de neurones classique



Neurone aritificial



Fonctions d'activation

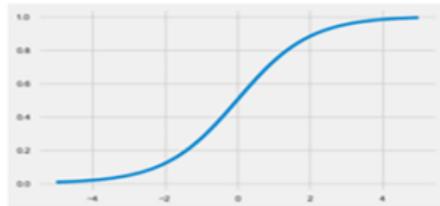
Les fonctions d'activation permettent aux modèles d'apprendre des relations plus complexes en capturant les non-linéarités dans les données.

- **Sigmoïde** : $\sigma(z) = \frac{1}{1+e^{-z}}$, plage de sortie (0, 1), utilisée pour la probabilité dans la classification binaire.
- **Tangente Hyperbolique (tanh)** : $\tanh(z)$, plage de sortie (-1, 1), version centrée et normalisée de la sigmoïde.
- **ReLU (Unité Linéaire Rectifiée)** : $f(z) = \max(0, z)$, non saturante, favorise la convergence rapide et permet d'éviter le problème de disparition des gradients.
- **Leaky ReLU** : $f(z) = \max(\alpha z, z)$, variante de ReLU qui permet un petit gradient lorsque $z < 0$.
- **Softmax** : Utilisée pour la couche de sortie des problèmes de classification multi-classes.

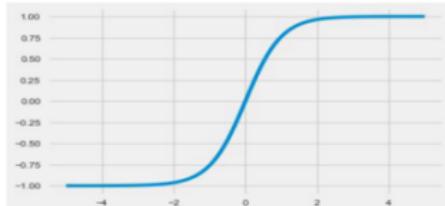
$$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}}$$

Illustration des fonctions d'activation

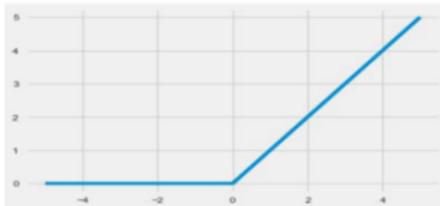
Sigmoïde



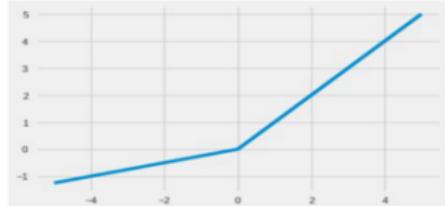
Tanh



ReLU



ReLU paramétrique



Entraînement d'un réseau de neurones artificiel

Principe d'Entraînement : L'entraînement d'un réseau de neurones consiste à ajuster ses poids pour minimiser une fonction de coût qui mesure l'erreur entre les prédictions et les vraies valeurs.

Descente de gradient stochastique (SGD) :

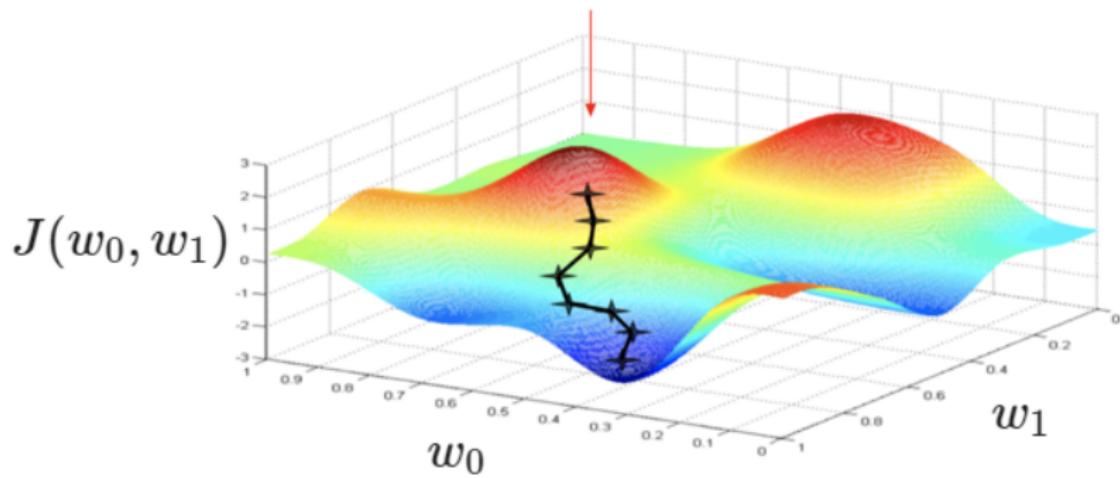
- Méthode d'optimisation utilisée pour mettre à jour les poids du réseau de manière itérative.
- À chaque itération, un sous-ensemble (batch) de données est utilisé pour calculer le gradient de la fonction de coût.
- Les poids sont mis à jour dans la direction opposée du gradient pour réduire l'erreur.

$$w_{new} = w_{old} - \eta \cdot \nabla_w J(w)$$

où :

- w_{old} et w_{new} sont les valeurs des poids avant et après la mise à jour,
- η est le taux d'apprentissage,
- $\nabla_w J(w)$ est le gradient de la fonction de coût par rapport aux poids.

Illustration graphique de la SGD



Rétropropagation du gradient

Il est facile de calculer les gradients correspondant à la dernière couche $\frac{\partial J}{\partial W^{(n)}}$ car l'erreur J dépend immédiatement de $W^{(n)}$.



Examinons maintenant le cas de la couche $n - 1$:

$$\frac{\partial J}{\partial W^{(n-1)}} = \frac{\partial J}{\partial W^{(n)}} \frac{\partial W^{(n)}}{\partial O_n} \frac{\partial O_n}{\partial W^{(n-1)}}$$

Or

$$\frac{\partial O_n}{\partial W^{(n-1)}} = \frac{\partial O_n}{\partial O_{n-1}} \frac{\partial O_{n-1}}{\partial W^{(n-1)}}$$

Surapprentissage dans les réseaux de neurones

Le surapprentissage (overfitting) se produit lorsqu'un réseau de neurones apprend trop bien les détails et le bruit des données d'entraînement, au détriment de sa capacité à généraliser sur de nouvelles données.

Le surapprentissage dans les réseaux de neurones peut se produire pour différentes raisons :

- Complexité excessive du modèle
- Manque de diversité dans les données d'entraînement
- Entraînement prolongé

Stratégies pour atténuer le surapprentissage :

- **Régularisation** : Techniques de régularisation telles que L1/L2, Dropout, pour limiter la complexité du modèle.
- **Dropout** : "Eteindre" un ensemble de neurones choisis aléatoirement pendant l'entraînement.
- **Early Stopping** : Arrêt de l'entraînement lorsque la performance sur un ensemble de validation cesse de s'améliorer.
- **Augmentation de données** : Augmenter la variété des données d'entraînement pour améliorer la robustesse du modèle.

Représentations distribuées

Introduction aux Embeddings

• Qu'est-ce qu'un Embedding ?

- Un embedding est une représentation vectorielle d'un mot qui capture le contexte du mot dans un document, des relations sémantiques et syntaxiques avec d'autres mots.
- Ils transforment des mots en vecteurs de nombres pour que les algorithmes de machine learning puissent les traiter efficacement.

• Pourquoi sont-ils importants ?

- Les embeddings permettent de capturer non seulement l'identité d'un mot mais aussi ses aspects sémantiques et contextuels.
- Ils facilitent des tâches telles que la classification de texte, la traduction automatique, et la détection des sentiments.

• Évolution des embeddings

- Historiquement, les mots étaient représentés comme des indices ou des vecteurs one-hot, où chaque mot est indépendant des autres.
- Les embeddings modernes, tels que Word2Vec et GloVe, représentent les mots dans des espaces vectoriels continus où les mots de sens similaires sont proches les uns des autres.

Différence entre One-hot Encoding et Embeddings

• One-hot Encoding

- Chaque mot est représenté par un vecteur avec un '1' dans la position qui lui est propre et des '0' partout ailleurs. Ce type de représentation est simple mais très inefficace en termes d'espace et ne capture pas les relations entre les mots.
- Exemple : pour un vocabulaire de dimension 100 000 :

Véhicule = [0, 0, 1, 0, 0, 0, 0, 0, ..., 0, 0]

Voiture = [0, 0, 0, 0, 0, 1, 0, 0, 0, ..., 0, 0]

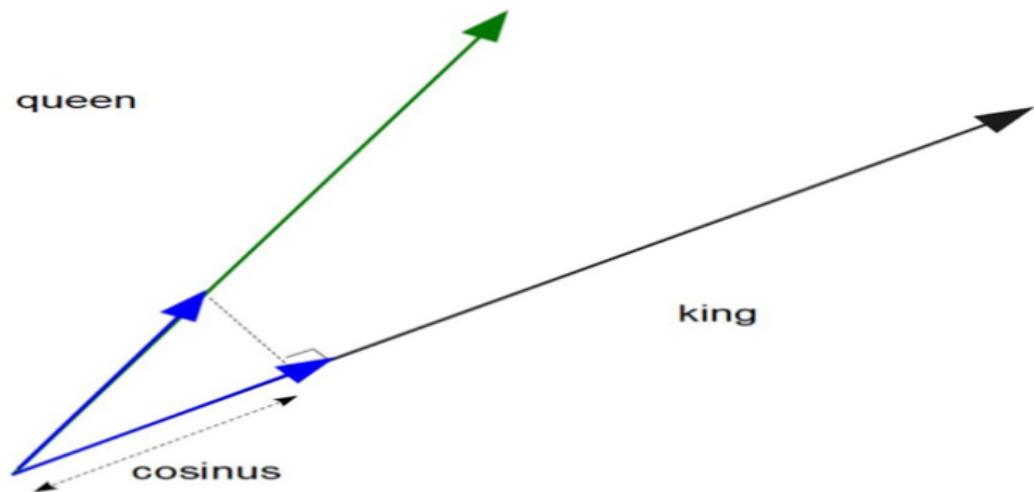
Cette représentation ne permet pas de prendre en compte la dimension sémantique

• Embeddings

- Les embeddings représentent les mots comme des vecteurs denses de nombres flottants (généralement entre 50 et 300 dimensions).
- Cette représentation est beaucoup plus riche et peut capturer des relations complexes entre les mots, comme la similarité sémantique.

Similarité sémantique

Nous sommes intéressés par des représentations de mots qui capturent la distance sémantique, sous forme de produit scalaire par exemple (ou distance cosiné).



Représentations distribuées : Principes

“You shall know a word by the company it keeps”

— J.R. Firth (1957)

Les mots sont similaires s'ils apparaissent fréquemment dans le même contexte.

- Il conduit son *véhicule* pour rentrer à la maison.
- Il conduit sa *voiture* pour rentrer chez lui.

Construction d'une représentation distribuée

Considérons le corpus suivant à titre d'exemple :

cnn in crop analysis

cnn and svm are widely used.

linear_regression performed along with svm

linear_regression for crop and farm

svm being used for farm monitoring

Do cnn, svm and linear_regression appear in the same context?

Le vocabulaire est alors :

[cnn, in, crop, analysis, and, svm, are, widely, used,
linear_regression, performed, along, with, for, farm, being,
monitoring, do, appear, the, same, context]

$$\dim(\text{vocabulaire}) = 22$$

Similarité sémantique

La matrice de co-occurrence représente la fréquence à laquelle les mots apparaissent ensemble deux à deux.

cnn in crop ...

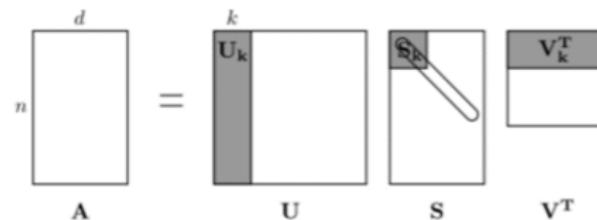
```
cnn [[0, 2, 1, 1, 1, 2, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1],  
in [2, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1],  
crop [1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],  
[1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],  
... [1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],  
[2, 1, 0, 0, 1, 0, 1, 1, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],  
[1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],  
[1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],  
[1, 0, 0, 0, 1, 2, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0],  
[1, 1, 1, 0, 1, 2, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1],  
[0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],  
[0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],  
[0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],  
[0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],  
[0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 2, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0],  
[0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 2, 0, 1, 1, 1, 0, 0, 0, 0, 0],  
[0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0],  
[0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0],  
[1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1],  
[1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1],  
[1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1],  
[1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1],  
[1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1],  
[1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0]]]
```

Réduction de la dimension

La décomposition en valeurs singulières est une technique permettant de réduire la dimension des données (similaire à l'ACP).

Étant donné une matrice $A \in \mathbb{R}^{n \times d}$

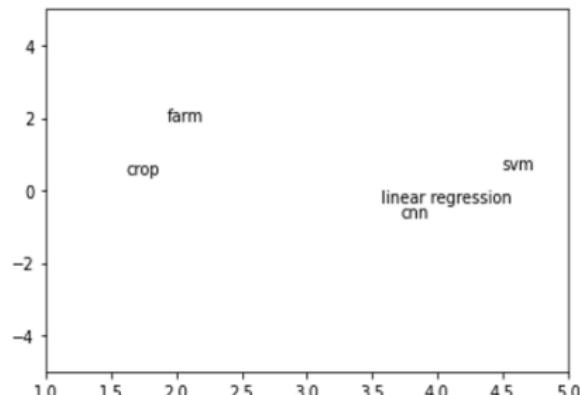
$$A = UDV^T \quad \text{où} \quad U \in \mathbb{R}^{n \times r}, D \in \mathbb{R}^{r \times r}, V \in \mathbb{R}^{d \times r}.$$



U est la matrice contenant les représentations (vecteurs de mots)

Visualisation des représentations distribuées

```
cnn :[ [ 3.72113518, -0.73233585],  
ln [ 2.7940387 , -1.40864784],  
crop [ 1.61178082, 0.44786021],  
... [ 0.77784994, -0.31230389],  
[ 2.12245048, 1.29571556],  
[ 4.49293777, 0.58090417],  
[ 1.33312748, 0.66758743],  
[ 1.33312748, 0.66758743],  
[ 2.25882809, 1.80738544],  
[ 3.56593605, -0.31006976],  
[ 0.95394179, 0.079159 ],  
[ 0.95394179, 0.079159 ],  
[ 0.95394179, 0.079159 ],  
[ 0.95394179, 0.079159 ],  
[ 1.92921553, 1.94783497],  
[ 1.92921553, 1.94783497],  
[ 1.12301312, 1.42126096],  
[ 1.12301312, 1.42126096],  
[ 2.26025282, -1.31571175],  
[ 2.26025282, -1.31571175],  
[ 2.26025282, -1.31571175],  
[ 2.26025282, -1.31571175],  
[ 2.26025282, -1.31571175]]
```



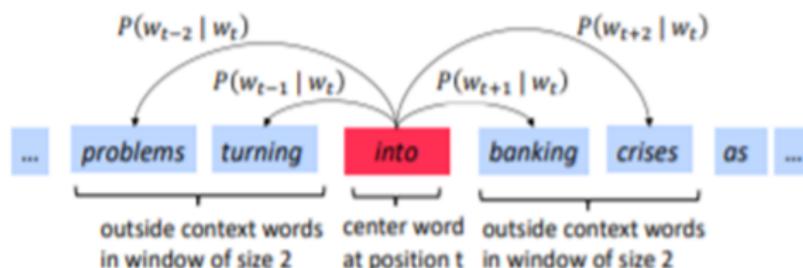
Word2Vec : Principles

- Principes de base :

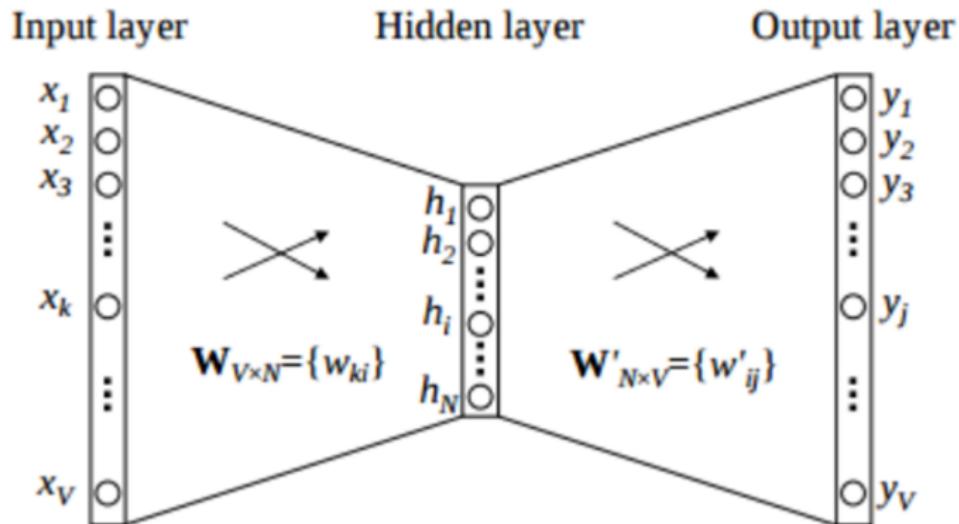
- Word2Vec est basé sur l'hypothèse distributionnelle : les mots qui apparaissent dans des contextes similaires ont des significations similaires.
 - Utilise un réseau de neurones peu profond pour apprendre des embeddings de mots à partir de grands corpus.

- Deux architectures principales :

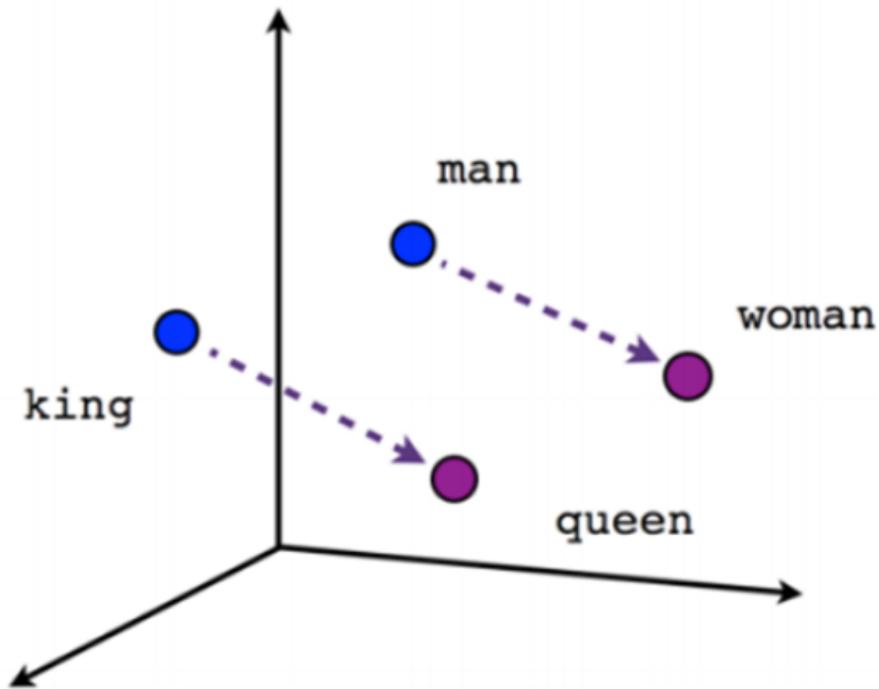
- ① *Continuous Bag of Words (CBOW)* : Prédit le mot cible à partir du contexte.
 - ② *Skip-Gram* : Prédit le contexte à partir du mot cible.



Architecture du modèle CBOW



Représentations Word2Vec



Applications pratiques de Word2Vec

- **Analogie de mots :**

- Word2Vec est célèbre pour capturer des relations complexes, comme "homme est à femme ce que roi est à reine".
- Permet de résoudre des analogies en utilisant des opérations arithmétiques simples sur les vecteurs de mots.

- **Clustering sémantique :**

- Les embeddings peuvent être utilisés pour regrouper des mots sémantiquement similaires, facilitant l'analyse de textes volumineux.

- **Amélioration des systèmes de recommandation :**

- Les vecteurs de mots de Word2Vec peuvent être utilisés pour améliorer la précision des recommandations en comprenant mieux les préférences des utilisateurs.

Autres représentations distribuées

Il existe d'autres représentations distribuées :

- Glove (2014) : proposé par une équipe de Stanford, il combine à la fois les techniques modernes de deep learning et les techniques statistiques (co-occurrences entre les mots). Il permet d'avoir des représentations des mots plus globales que Word2Vec.
- Fasttext (à partir de 2016) : la librairie a été créée par Facebook. Le modèle est entraîné sur des subwords (n-grams de caractères). Il est ainsi plus efficace pour traiter les mots inconnus (out of vocabulary).
- Représentations contextuelles (Transformers), etc.

Réseaux de neurones récurrents

Perspective historique des RNN

Les Réseaux de Neurones Récurrents (RNN) ont évolué au fil du temps, marquant des étapes importantes dans le domaine du machine learning et du NLP.

Développements clés :

- **1980s : Naissance des RNN** - Introduction des concepts de base des RNN par John Hopfield et David Rumelhart, posant les fondations des réseaux à mémoire.
- **1990s : Popularisation des RNN** - Jordan et Elman développent des architectures permettant une meilleure gestion des séquences temporelles et des dépendances.
- **Début des années 2000** - Identification des problèmes de disparition et d'explosion du gradient, limitant l'efficacité des RNN sur de longues séquences.
- **1997 : Avènement des LSTM** - Introduction des LSTM par Hochreiter et Schmidhuber, offrant une solution aux problèmes de mémoire à long terme.
- **2010s : GRU** - Les GRU simplifient la structure des LSTM. Les RNN sont de plus en plus utilisés dans des applications complexes comme la traduction automatique et la reconnaissance vocale.

Introduction aux RNN

Les Réseaux de Neurones Récurrents (RNN) sont une classe de réseaux de neurones artificiels spécialement conçus pour traiter des séquences de données, tels que des séries temporelles ou des séquences textuelles.

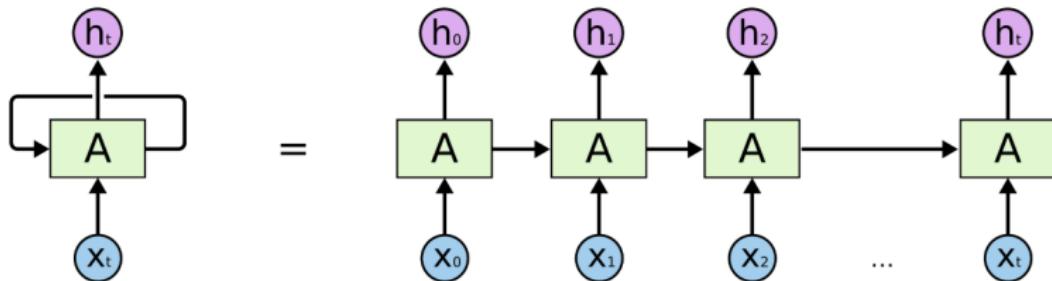
Caractéristiques:

- **Mémoire à court terme** : Les RNN ont la capacité de se "souvenir" d'informations passées grâce à leurs connexions récurrentes.
- **Traitement de séquences** : Ils sont particulièrement adaptés pour des tâches où les données sont séquentielles et où le contexte est important (ex: langage naturel, musique).
- **Modélisation de dépendances temporelles** : Les RNN peuvent capturer des dépendances temporelles et contextuelles dans les données.

Architecture des RNN

Cette architecture représente un RNN à la fois dans sa forme condensée et dépliée à travers le temps.

- À gauche, le RNN est présenté de manière condensée avec:
 - Une entrée x_t .
 - Une sortie d'état caché h_t .
 - Un bloc A représentant la cellule RNN qui effectue les calculs à partir de l'état caché précédent et de l'entrée courante pour produire le nouvel état caché.
- À droite, l'architecture est déroulée pour montrer la séquence complète:
 - Chaque bloc A est le même RNN à différents instants temporels.
 - Le bloc A prend une entrée x_t et produit un état caché h_t , qui est alors passé à la même cellule au pas de temps suivant.
 - L'état initial h_0 est souvent initialisé à zéro ou une petite valeur aléatoire.



Problème de disparition du gradient

Le problème de disparition du gradient survient lors de la rétropropagation dans les RNN traditionnels. Les gradients des paramètres peuvent devenir très petits, rendant l'apprentissage des dépendances à long terme difficile.

Mathématiquement, pendant la rétropropagation, si les valeurs de $\frac{\partial h_t}{\partial W}$ sont petites, le gradient total peut tendre vers zéro à mesure que T augmente.

Solutions au problème de disparition du gradient :

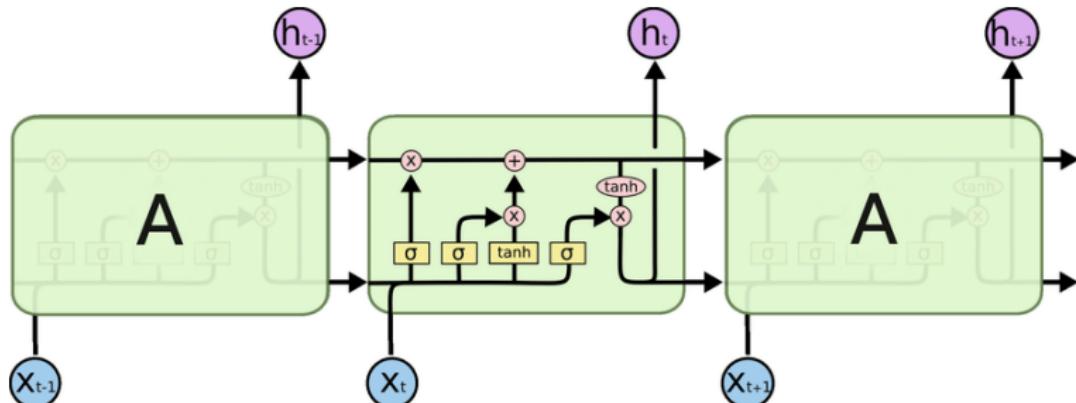
Pour atténuer le problème de disparition du gradient, différentes solutions ont été proposées :

- Utilisation de fonctions d'activation comme ReLU au lieu de sigmoïde ou tanh.
- Introduction d'architectures RNN avancées comme LSTM et GRU.
- Techniques de clipping de gradient pour éviter l'explosion des gradients.

RNN avec Long Short-Term Memory (LSTM)

Les LSTM sont une variante des RNN conçus pour mieux capturer les dépendances à long terme. Ils introduisent des 'cellules mémoire' avec des portes de régulation :

- Porte d'oubli : contrôle la quantité d'informations à retenir de l'état précédent.
- Porte d'entrée : contrôle la quantité d'informations à ajouter de l'entrée actuelle.
- Porte de sortie : détermine la quantité d'informations à transmettre à l'état suivant.



Applications des RNN

Les RNN sont largement utilisés dans divers domaines tels que :

- Traitement du langage naturel : traduction automatique, génération de texte.
- Reconnaissance vocale : conversion de la parole en texte.
- Prévision de séries temporelles : prévision météorologique, analyse boursière.

Modèles Seq2Seq

Modèles Seq-to-Seq

- Les modèles de séquence à séquence (seq-to-seq) sont une classe de modèles en apprentissage profond conçus pour transformer une séquence d'entrée en une séquence de sortie.
- Ils sont essentiels dans le domaine du traitement automatique du langage naturel (NLP), avec des applications allant de la traduction automatique au résumé de texte et à la génération de dialogue.

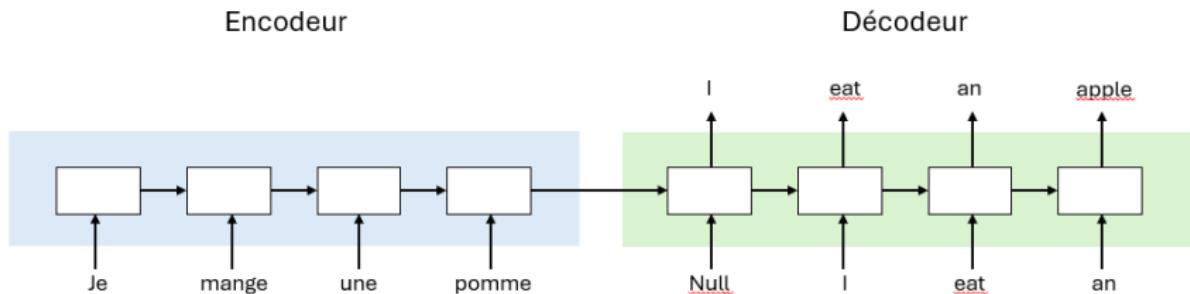
Pourquoi les modèles Seq-to-Seq?

- **Flexibilité** : Peuvent gérer des séquences d'entrée et de sortie de longueurs variables.
- **Puissance** : Capables de capturer des dépendances complexes entre les éléments de la séquence.
- **Universalité** : Applicables à une large gamme de tâches en NLP, démontrant une capacité exceptionnelle à modéliser le langage et d'autres séquences.

Architecture de base

L'architecture seq-to-seq comprend deux composants principaux :

- **Encodeur** : Un réseau de neurones qui lit et encode la séquence d'entrée en un vecteur de contexte (une représentation dense de la séquence).
 - **Décodeur** : Un autre réseau de neurones qui lit le vecteur de contexte pour générer la séquence de sortie, un élément à la fois.

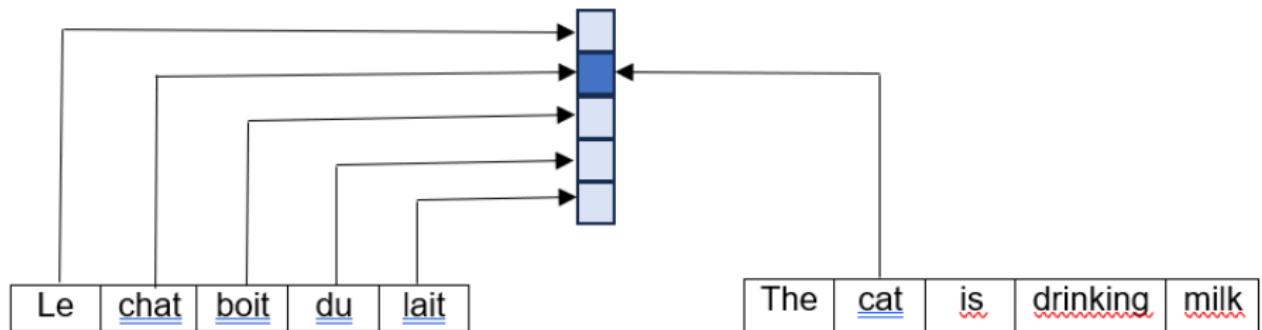


Introduction au mécanisme d'Attention

Le mécanisme d'attention est une amélioration clé apportée aux modèles seq-to-seq traditionnels, permettant au modèle de se "concentrer" sur différentes parties de la séquence d'entrée lors de la génération de la séquence de sortie.

- **Objectif** : Surmonter les limitations des encodeurs seq-to-seq qui compressent toute l'information d'une séquence d'entrée dans un vecteur de contexte fixe.
- **Avantage** : Améliore la capacité du modèle à gérer de longues séquences d'entrée, en rendant le processus de génération de la séquence de sortie plus dynamique et contextuellement informé.

Illustration du mécanisme d'Attention



Types d'Attention

Il existe plusieurs variantes du mécanisme d'attention, chacune avec ses propres caractéristiques et applications :

- **Attention globale** : Le décodeur considère tous les états cachés de l'encodeur simultanément pour générer le vecteur de contexte.
- **Attention locale** : Le décodeur ne considère qu'une sous-séquence des états cachés de l'encodeur autour d'une position centrale pour chaque étape de décodage.
- **Self Attention** : Utilisée dans les transformateurs, cette forme d'attention permet à chaque position dans une séquence d'entrée de considérer toutes les positions et de s'auto-pondérer.

Chaque type d'attention est adapté à différentes tâches et configurations de modèle.

Importance de l'Attention

Le mécanisme d'attention a révolutionné la manière dont les modèles de NLP traitent et génèrent des séquences :

- **Amélioration des Performances** : Permet une meilleure gestion des dépendances à longue distance dans les données.
- **Interprétabilité** : Les poids d'attention fournissent un aperçu de la manière dont le modèle prend ses décisions, en montrant quelles parties de l'entrée influencent le plus la sortie. item **Flexibilité et Adaptabilité** : Facilite l'adaptation des modèles à différentes longueurs de séquence et types de tâches, améliorant ainsi leur applicabilité à un large éventail de domaines.

En somme, le mécanisme d'attention a non seulement amélioré la capacité des modèles à traiter des informations séquentielles complexes, mais a également ouvert la voie à des avancées significatives dans la compréhension et la génération du langage naturel par les machines.

Transformers et LLMs

Attention Is All You Need

Ashish Vaswani*
Google Brain
avaswani@google.com

Noam Shazeer*
Google Brain
noam@google.com

Niki Parmar*
Google Research
nikip@google.com

Jakob Uszkoreit*
Google Research
usz@google.com

Llion Jones*
Google Research
llion@google.com

Aidan N. Gomez* †
University of Toronto
aidan@cs.toronto.edu

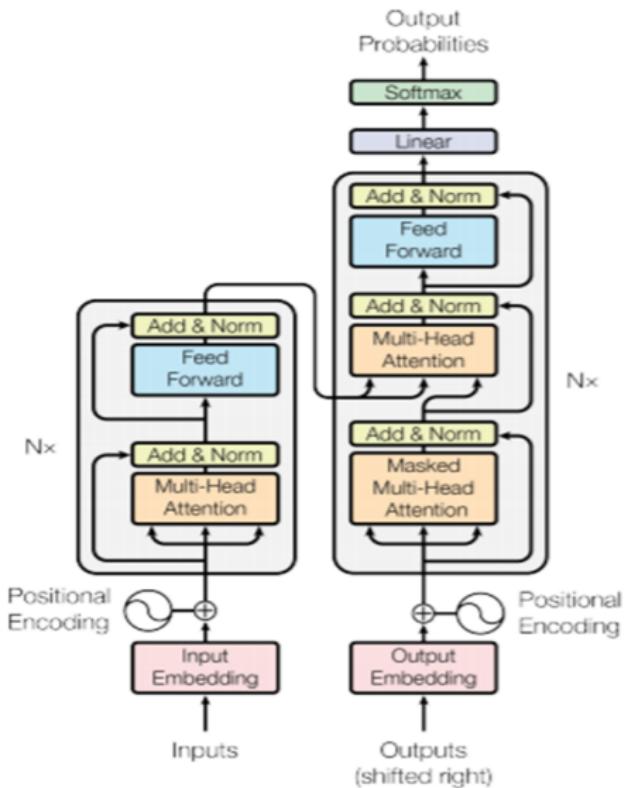
Lukasz Kaiser*
Google Brain
lukasz.kaiser@google.com

Illia Polosukhin* ‡
illia.polosukhin@gmail.com

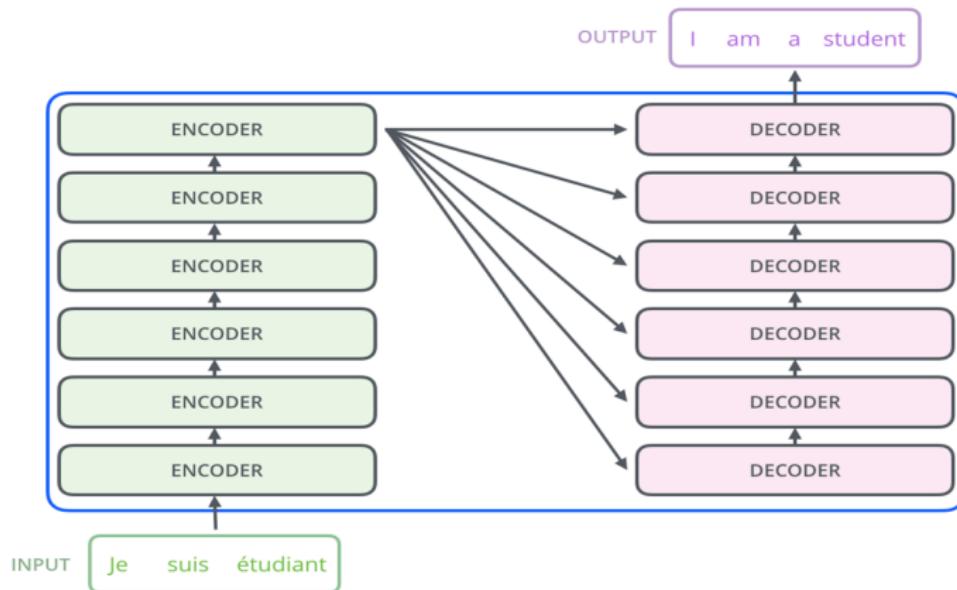
Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to

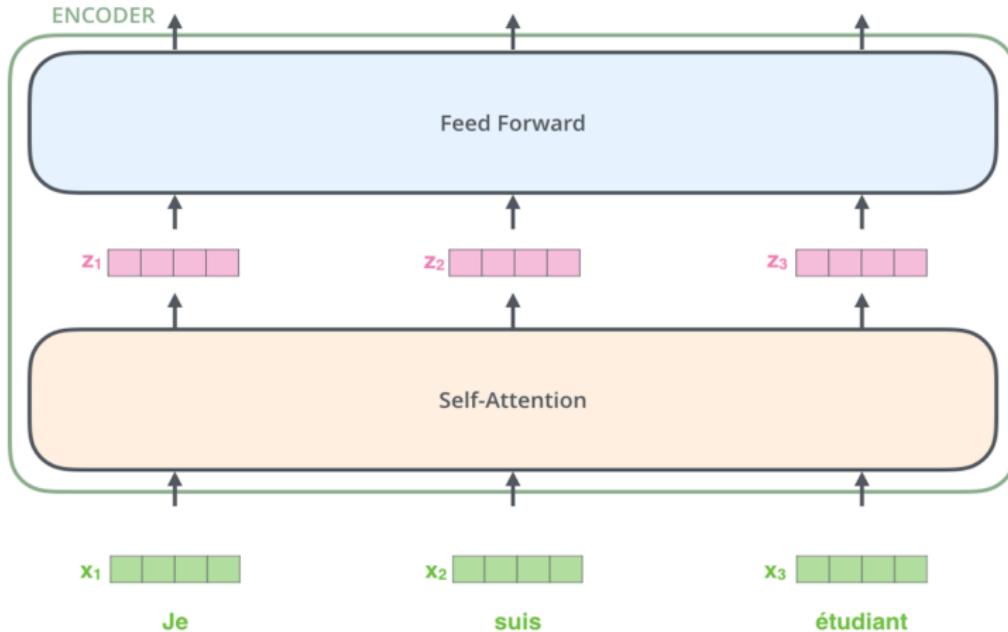
Transformer originel



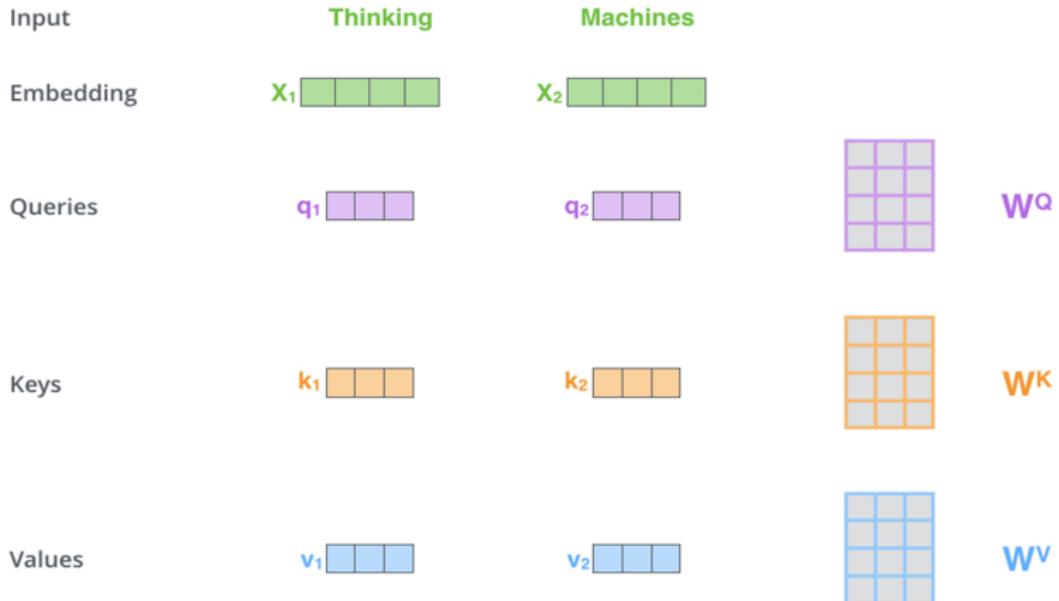
Mécanisme de cross-attention



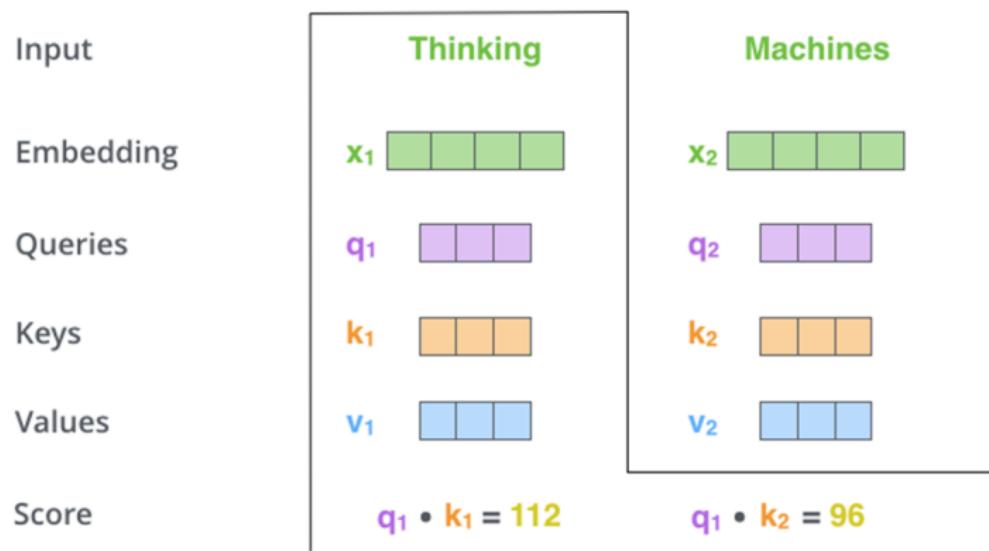
Mécanisme de self-attention



Fonctionnement du mécanisme de self-attention



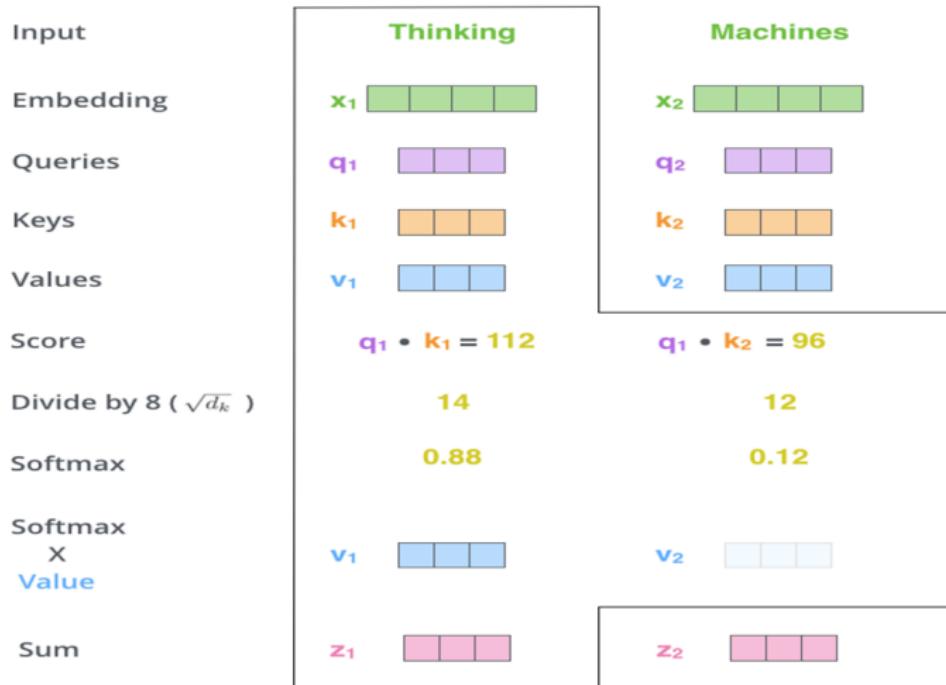
Calcul des scores de self-attention



Calcul des scores de self-attention

Input	Thinking		Machines	
Embedding	x_1	[4 green boxes]	x_2	[4 green boxes]
Queries	q_1	[3 purple boxes]	q_2	[3 purple boxes]
Keys	k_1	[3 orange boxes]	k_2	[3 orange boxes]
Values	v_1	[3 blue boxes]	v_2	[3 blue boxes]
Score	$q_1 \bullet k_1 = 112$		$q_1 \bullet k_2 = 96$	
Divide by 8 ($\sqrt{d_k}$)	14		12	
Softmax	0.88		0.12	

Calcul de la nouvelle représentation



Performances du tranformer originel

Entraîné sur WMT 2014 English-German dataset, comprenant près de 4.5 millions de phrases sentence, le WMT 2014 English-French dataset, comprenant près de 36 millions de phrases.

- 8 NVIDIA P10 GPUs
- **Base** : 12 heures
- **Large** : 3.5 jours

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	41.29	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	$3.3 \cdot 10^{18}$	
Transformer (big)	28.4	41.8	$2.3 \cdot 10^{19}$	

BERT

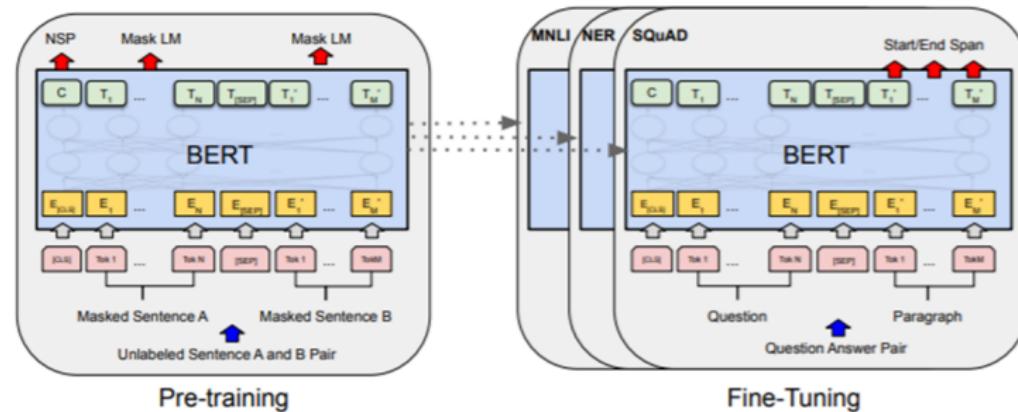
BERT (Bidirectional Encoder Representations from Transformers) représente une avancée majeure dans le NLP, introduisant une approche novatrice pour la modélisation de langage.

- **Contextualisation profonde** : Première architecture à utiliser pleinement la bidirectionnalité pour comprendre le contexte des mots, permettant une compréhension plus fine du langage.
- **Pré-entraînement et fine-tuning** : Méthodologie en deux étapes offrant une flexibilité pour l'adaptation à diverses tâches de TALN sans architecture spécifique à la tâche.
- **Impact sur le NLP** : Avant BERT, les approches de modélisation de langage étaient limitées par la compréhension unidirectionnelle ou des représentations statiques des mots. BERT a changé la donne en permettant des représentations contextuelles dynamiques.
- **Performances révolutionnaires** : À son introduction, BERT a établi de nouveaux standards de performance sur une gamme de benchmarks de NLP, y compris GLUE, SQuAD, etc.

Fine-tuning de BERT pour des tâches spécifiques

Après pré-entraînement, BERT est affiné pour des tâches spécifiques:

- Ajout d'une couche de sortie spécifique à la tâche (classification, NER, QA).
 - Fine-tuning de tous les paramètres du modèle pré-entraîné sur le corpus de la tâche.



Performances de BERT

BERT a été pré-entraîné sur le BookCorpus (800 millions de mots) et English Wikipedia (2,500 millions de mots).

Deux modèles :

- **BERT Base** : 12 couches avec 110 millions de paramètres.
- **BERT Large** : 24 couches avec 340 millions de paramètres.

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	Average
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
BERT _{BASE}	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
BERT _{LARGE}	86.7/85.9	72.1	92.7	94.9	60.5	86.5	89.3	70.1	82.1

Introduction aux modèles autorégressifs - GPT-3

GPT-3 (Generative Pre-trained Transformer 3) est le modèle de langue développé par OpenAI, représentant la troisième génération de la série GPT. Avec 175 milliards de paramètres, GPT-3 pousse les limites de la génération de texte et de la compréhension du langage naturel.

- **Capacités générales** : GPT-3 excelle dans une variété de tâches de TALN sans fine-tuning spécifique, grâce à sa puissance de modélisation du langage.
- **Architecture autorégressive** : Utilise un modèle Transformer pour prédire le mot suivant dans un texte, apprenant des patterns complexes sur des données massives.
- **Approche few-shot learning** : Capable d'adapter ses réponses à des tâches spécifiques avec peu ou pas d'exemples d'entraînement.
- **Impact sur IA et le NLP** : GPT-3 a significativement avancé les capacités des systèmes d'IA dans la compréhension et la génération de langage naturel, ouvrant de nouvelles voies pour l'application de l'intelligence artificielle.

Architecture de GPT-3

GPT-3 repose sur une architecture Transformer améliorée, optimisée pour traiter efficacement de grandes quantités d'informations et générer des réponses cohérentes et contextuellement pertinentes.

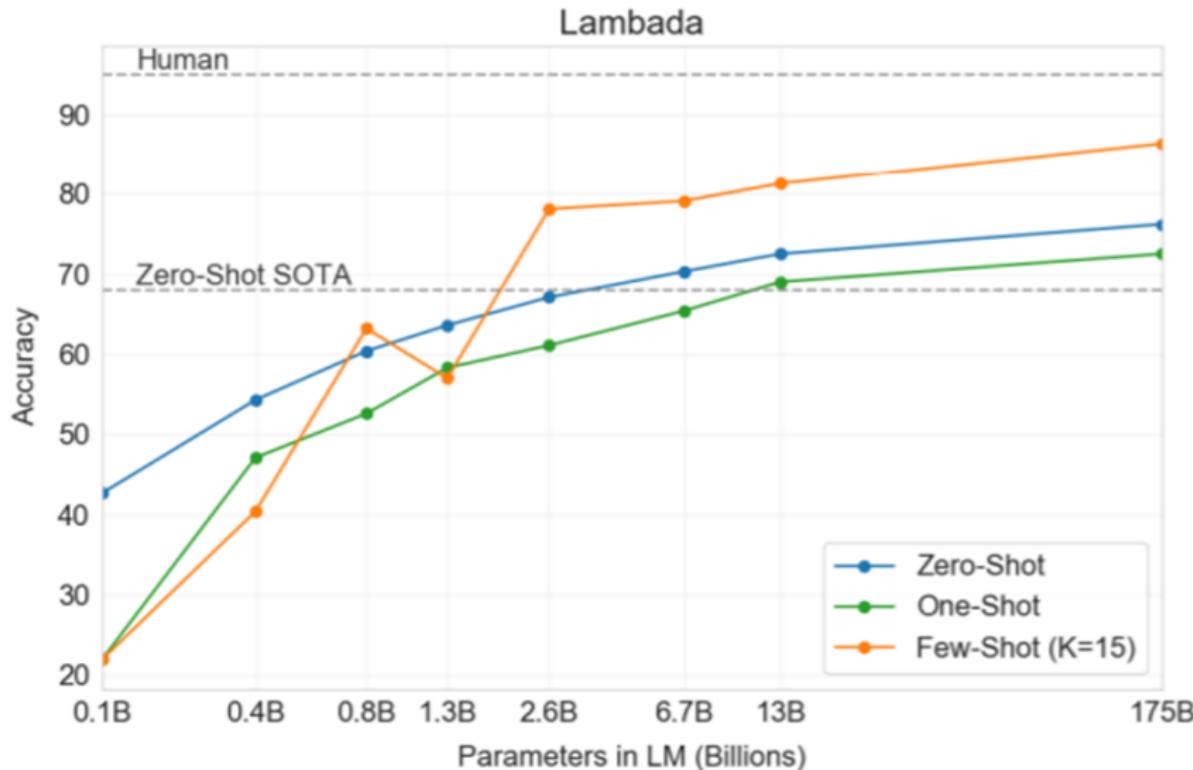
- **Taille du modèle :** Avec 175 milliards de paramètres, GPT-3 est l'un des modèles de langue les plus grands et les plus complexes à ce jour.
- **Mécanisme d'attention :** L'attention multi-têtes permet au modèle de pondérer différemment les parties d'un input, améliorant la compréhension du contexte.
- **Optimisation et entraînement :** Techniques d'optimisation avancées pour gérer la taille du modèle et l'efficacité de l'entraînement.

Few-Shot Learning avec GPT-3

L'une des innovations les plus remarquables de GPT-3 réside dans sa capacité à effectuer du few-shot learning, permettant au modèle de comprendre et d'exécuter des tâches spécifiques avec très peu d'exemples.

- **Définition :** Le few-shot learning désigne la capacité d'un modèle à apprendre une nouvelle tâche à partir d'un très petit nombre d'exemples d'entraînement, souvent seulement quelques-uns.
- **Mécanisme dans GPT-3 :**
 - GPT-3 utilise des prompts contenant quelques exemples de la tâche désirée pour guider le modèle sur ce qui est attendu, avant de présenter la question ou la tâche à résoudre.
 - Le modèle généralise ensuite à partir de ces exemples pour générer des réponses ou des solutions aux problèmes posés, montrant une compréhension étonnante de la tâche avec un minimum de guidance.
- **Exemples d'application :**
 - Classification de texte, génération de résumés, réponse à des questions spécifiques, et plus, avec seulement quelques exemples pour chaque tâche.
- **Impact :** Cette capacité émergente réduit considérablement le besoin de vastes ensembles de données d'entraînement spécifiques à la tâche, ouvrant la voie à des applications plus flexibles et accessibles de modèles de langue.

Performances de GPT-3 en few-shot learning



ChatGPT

ChatGPT, développé par OpenAI, est un modèle de langage basé sur l'architecture GPT (Generative Pre-trained Transformer) optimisé pour comprendre et générer des dialogues naturels. L'entraînement de ChatGPT se décline selon les étapes suivantes :

- ① Pré-entraînement sur un corpus volumineux :** Comme GPT-3, ChatGPT est d'abord pré-entraîné sur un vaste ensemble de données textuelles, englobant un large éventail de la littérature disponible sur Internet, pour apprendre une compréhension générale du langage.
- ② Fine-tuning supervisé :** Ensuite, ChatGPT est affiné sur des dialogues spécifiques pour améliorer ses compétences conversationnelles. Cette étape utilise des paires question-réponse et des conversations pour enseigner au modèle des structures de dialogue et des réponses contextuellement appropriées.
- ③ Reinforcement Learning from Human Feedback (RLHF):** Utilisation de techniques de renforcement pour ajuster les réponses du modèle basées sur les préférences et les corrections fournies par des évaluateurs humains, raffinant davantage la pertinence et la naturalité des réponses.

Fine-Tuning supervisé (FST)

Le fine-tuning supervisé joue un rôle essentiel dans l'adaptation de ChatGPT à des tâches conversationnelles spécifiques, améliorant sa capacité à fournir des réponses pertinentes et contextuellement adaptées.

Méthodologie:

- ① Collecte d'un ensemble de données de dialogues de haute qualité, comprenant des échanges humains et des interactions annotées pour capturer divers contextes conversationnels.
- ② Transformation de ces dialogues en paires de prompts et réponses pour créer des exemples d'entraînement qui guident le modèle dans l'apprentissage de structures conversationnelles et de réponses appropriées.
- ③ Utilisation d'un processus d'entraînement supervisé où le modèle apprend à prédire la réponse la plus probable à un prompt donné, en ajustant ses paramètres internes pour minimiser la divergence entre les réponses générées et les réponses attendues (annotations).

Alignement des LLM avec RLHF

- La génération de texte par les LLM est essentielle dans de nombreux domaines tels que la traduction automatique, la résumé automatique, et la création de contenu.
- Cependant, la qualité du texte généré peut être variable et dépend souvent des données d'entraînement disponibles.
- L'ajout du feedback humain permet d'améliorer la qualité du texte généré en fournissant une supervision supplémentaire.
- RLHF combine l'apprentissage par renforcement avec le feedback humain pour guider l'apprentissage des LLM de manière plus précise et efficace.
- L'objectif est d'entraîner les LLM à générer du texte de meilleure qualité en s'appuyant sur le savoir-faire humain pour corriger les erreurs et améliorer les performances.

Approche RLHF pour l'entraînement des LLM

- Collecte des feedbacks :
 - Les LLM génèrent du texte qui est évalué par des humains.
 - Les humains fournissent des annotations telles que des corrections, des scores de qualité ou des préférences.
- Calcul de la récompense :
 - Le feedback humain est utilisé pour calculer une récompense.
 - Différentes métriques peuvent être utilisées pour quantifier la qualité du texte généré.
 - La récompense peut être une fonction de ces métriques et des préférences humaines.
- Entraînement des LLM :
 - L'objectif est d'entraîner les LLM à maximiser la récompense définie par le feedback humain.
 - Les LLM sont entraînés à générer du texte de meilleure qualité en fonction de cette récompense.

Construction des feedbacks humains

- Les feedbacks humains sont construits sur la base des réponses générées par les LLM.
- Ces réponses sont ensuite évaluées par les humains, qui fournissent des annotations telles que des corrections, des scores de qualité, ou des préférences.
- Les feedbacks humains sont utilisés comme données d'entraînement pour apprendre au modèle de récompense à évaluer la qualité du texte généré.
- Les caractéristiques du texte généré, telles que la similarité avec des textes de référence ou la pertinence du contenu, peuvent être utilisées comme des caractéristiques pour l'entraînement du modèle.
- L'objectif est de créer un modèle de récompense capable d'évaluer de manière précise et fiable la qualité du texte généré par les LLM, selon les critères humains.

Enjeux éthiques de l'IA

- **Enjeux sociétaux** : production massive de deepfakes, environnement peuplé par des artefacts, etc.
- **Alignement** : les valeurs encodées dans les modèles génératifs sont aujourd'hui déterminées par une poignée de personnes dans la Silicon Valley.
- **Transhumanisme** : à mesure que les modèles deviennent très puissants — voire plus puissants que l'intelligence humaine, la tentation de l'augmentation se fera plus pressante.
- **Enjeux ontologiques** : si toutes les tâches et facultés dont on pensait jusque-là qu'elles étaient le propre de l'Homme sont accessibles à l'intelligence artificielle, alors quelle est finalement l'essence de l'humain ?

Merci de votre attention

redha_moulla@yahoo.fr