

# Machine Learning

EMSI - Université Côte d'Azur  
Richard Grin  
Version 1.4 - 7/12/24

1

## Objectif du cours

- Ce cours « Agents conversationnels en Java avec LangChain4j » fait partie du parcours MIAGE « Intelligence Artificielle Appliquée » (IA2)
- Il est essentiellement pratique et montre comment tirer profit de LLMs existants dans les applications d'entreprise, en utilisant l'API de ces modèles
- LLM : Large Language Model ; utilisé, par exemple, par ChatGPT

R. Grin

ML

2

2

## Bonus

- Pour la réactivité : réponse aux emails en respectant les formats demandés, installation des logiciels, TPs terminés,...
- Pour les réponses aux questions pendant le cours
- Pour les questions intéressantes
- Retenez votre numéro dans la liste ; vous me donnerez ce numéro si vous avez un bonus, ou dans vos emails
- Malus pour retards, manque de travail manifeste, pas pour mauvaises réponses ou questions

Richard Grin

Présentation Jakarta EE

page 3

3

## Examen

- Durée 2 heures, sans documents
- Les bonus seront ajoutés à la note de l'examen pour avoir la note finale
- Vous choisissez la date, avec l'administration de l'EMSI ; date préconisée : autour de la fin de l'année, fin décembre ou début janvier (le temps de finir les TPs)
- Le délégué me tient au courant assez rapidement

R. Grin

ML

4

4

## TPs

- Ce cours ne peut être assimilé qu'en faisant les TPs et il est donc **indispensable** de les faire
- Des questions de l'examen pourront porter sur les TPs
- Bonus pour les TPs seulement si projet GitHub avec tous les commits
- Vous pourrez terminer les TPs après mon départ et avant l'examen

Richard Grin

Présentation Jakarta EE

page 5

5

## Comment demander de l'aide

- Lisez attentivement cette page : <http://richard.grin.free.fr/emsi/casablanca-ia/tp/demandeaide.html>
- N'envoyez pas de copie d'écran, sauf exception
- Le minimum d'information à fournir :
  - environnement d'exécution (version de l'OS et des logiciels,...)
  - étapes qui ont conduit au problème
  - message d'erreur (et logs)

Richard Grin

Présentation Jakarta EE

page 6

6

## Supports du cours

- Premier support rappelle quelques généralités et concepts importants du machine learning
- Deuxième support sur les embeddings et les LLMs
- Troisième support sur LangChain4j, le framework standard de facto pour utiliser les LLMs avec du code Java, utilisé dans les TP
- Dernier support sur le RAG qui permet d'améliorer les réponses des LLMs

R. Grin

ML

7

7

## Plan de ce support

- Généralités sur l'IA
- Machine learning
- Réseaux de neurones
- Transformeurs
- Références

R. Grin

ML

8

8

## Intelligence artificielle

R. Grin

ML

9

9

## Définition

- Ensemble des théories et techniques visant à réaliser des machines capables de simuler l'intelligence humaine (Wikipedia)

R. Grin

ML

10

10

## Types d'IA

- IA « classique » : s'appuie sur des règles et des connaissances fournies par des experts ; fournit des systèmes experts
- Machine learning (ML) : développe des algorithmes et des modèles permettant à des ordinateurs de faire des prédictions, en s'appuyant sur des données d'entraînement fournies lors d'une phase d'apprentissage
- Deep learning (DL) : branche du ML basée sur les réseaux de neurones

R. Grin

ML

11

11

## IA générative

- Branche du DL qui se concentre sur la création de contenus nouveaux à partir de données existantes
- Peut générer du texte mais aussi des images, de la musique, des vidéos, etc.
- Exemples : ChatGPT, Gemini, LLama

R. Grin

ML

12

12

# Machine learning

R. Grin

ML

13

13

## Présentation

- Durant la phase d'apprentissage le logiciel apprend à partir d'exemples (*dataset*) qu'on lui fournit
- La **phase d'apprentissage** du ML fournit un **modèle** qui est utilisé ensuite pour prévoir de nouveaux résultats pendant l'**inférence** (utilisation du modèle)
- Le modèle peut servir pour
  - classer, par exemple décider si un email est un spam
  - estimer des valeurs (régression) par exemple estimer le coût d'un appartement
  - générer du contenu, par exemple générer la réponse à une question

R. Grin

ML

14

14

## Types de machine learning

- Dans le ML « classique », le data scientist (expert en science des données) choisit le type de modèle à utiliser (modèle linéaire, polynomial, arbre de décision, clustering par exemple)
- Le deep learning (DL, apprentissage profond) est un type de ML qui utilise des réseaux de neurones ; le data scientist peut paramétrer un réseau de neurones ou choisir des outils ou des LLMs, mais il n'a pas à choisir un type de modèle

R. Grin

ML

15

15

## Modèle

- Représentation mathématique d'un système ou d'un processus qui permet de faire des prédictions ou de prendre des décisions basées sur des données
- Fonction  $Y = f(x_1, \dots, x_n)$  ; la phase d'apprentissage cherche la fonction « prédictive »  $f$  qui sera la meilleure approximation de la fonction « parfaite » qui donnerait toujours le bon résultat
- Pour le type régression,  $Y$  est numérique (valeur simple, vecteur,...) ; pour le type catégoriel,  $Y$  est une catégorie (classification) ; pour le type génératif,  $Y$  peut être du texte, une image, une vidéo, du son,...

R. Grin

ML

16

16

## Quelques types de modèles

- Linéaire - fonction linéaire entre les variables d'entrée et de sortie (régression linéaire)
- Arbre de décision - utilise une structure d'arbre pour prendre des décisions
- Clustering - regroupe des données similaires dans étiquettes prédéfinies (k-means clustering)
- Réseaux de neurones - utilise des neurones artificiels

R. Grin

ML

17

17

## Exemple 1

- Modèle qui prédit si un client est susceptible d'acheter un certain produit
- Un client est décrit par des caractéristiques qui correspondent aux données fournies pendant l'apprentissage : âge, groupe social ou géographique, achats déjà effectués, etc.
- Le modèle est entraîné avec un grand nombre d'exemples **réels** de clients, avec leurs caractéristiques, en précisant si le client a acheté le produit
- Après l'entraînement, le modèle doit être capable de prédire (et avec quelle probabilité) si un client quelconque est susceptible d'acheter le produit

R. Grin

ML

18

18

## Exemple 2

- Modèle qui calcule le prix d'un appartement à afficher, compte tenu de son emplacement, de sa superficie, de son étage, de son environnement, ... pour qu'il trouve rapidement un acquéreur
- Pendant l'entraînement du modèle on lui fournit les caractéristiques de très nombreux appartements qui ont été vendus, et le temps qu'il a fallu pour trouver un acquéreur

R. Grin

ML

19

19

## Préparation des données

- Important
- Les données en entrée doivent souvent être préparées ; par exemple suppression des balises HTML, emojis non pertinents, corriger fautes d'orthographe, espaces superflus, suppression des mots « vides » (stop words)
- Eliminer le bruit et les biais
- Gestion des données sensibles (enlever les mots de passe, les clés secrètes)
- Ajout éventuel de contexte

R. Grin

ML

20

20

## Paramètres d'un modèle

- La plupart des types de modèle ont des paramètres définis à l'avance  $p_1, p_2, \dots$  qui déterminent le comportement du modèle
- Par exemple, si on choisit un modèle linéaire, le modèle est représenté par une fonction affine qui a 2 paramètres  $a$  et  $b$  :  $f(x) = ax + b$
- Les valeurs de ces paramètres sont déterminées automatiquement durant l'apprentissage pour minimiser la fonction de perte, afin que le modèle ait les meilleures prédictions possibles

R. Grin

ML

21

21

## Paramètres et hyperparamètres

- Les paramètres d'un modèle sont internes au modèle ; leurs valeurs sont ajustées pendant l'apprentissage, comme les valeurs  $a$  et  $b$  d'un modèle linéaire
- Les hyperparamètres sont des valeurs choisies par le data scientist pour paramétrer le modèle,
  - pendant l'apprentissage du modèle ; par exemple, taille des pas pendant la descente de gradient, nombre des couches cachées dans les réseaux de neurones
  - pendant l'inférence ; par exemple température

R. Grin

ML

22

22

## Apprentissage

- Nombreux types d'apprentissage mais ils utilisent tous plus ou moins les mêmes éléments :
  - **Exemples d'apprentissage** ; chaque exemple est composé d'une valeur qu'on donnera en entrée du modèle, et du résultat attendu pour cette entrée
  - **Modèle** qui apprend à partir des données d'apprentissage
  - **Mesure de performance** qu'il faut optimiser pendant l'apprentissage ; la fonction de perte (ou de coût) mesure l'erreur faite par le modèle à un moment de l'apprentissage, lorsqu'il est testé sur une donnée d'apprentissage
  - **Algorithme d'optimisation** pour obtenir les meilleurs résultats sur les exemples d'apprentissage

R. Grin

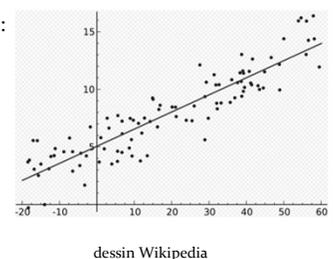
ML

23

23

## Exemple régression linéaire (1/2)

- Données d'apprentissage : points qui correspondent aux données relevées
- Modèle : fonction affine  $f(x) = ax + b$  représentée par une droite de régression



R. Grin

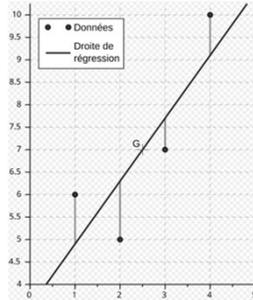
ML

24

24

## Exemple régression linéaire (2/2)

- Fonction de perte : « somme des distances » des points à la droite de régression
- Algorithme d'optimisation : minimisation de la somme, par exemple avec une descente de gradient



R. Grin

ML

25

25

## Phase d'apprentissage (1/2)

- But : trouver les meilleurs paramètres pour le modèle
- On commence par donner des valeurs aléatoires aux paramètres du modèle
- La fonction de perte dépend des paramètres du modèle et des exemples d'apprentissage (valeurs en entrée et attendues en sortie)
- Pour chaque exemple (ou groupe d'exemples) d'apprentissage, l'algorithme d'optimisation calcule comment améliorer les paramètres du modèle, c'est-à-dire comment minimiser la fonction de perte

R. Grin

ML

26

26

## Phase d'apprentissage (2/2)

- Avec les nouveaux paramètres mis à jour, on refait le même processus avec l'exemple d'apprentissage suivant, et ainsi de suite, jusqu'au dernier exemple d'apprentissage (fin d'une « époque » ; on peut commencer une nouvelle époque avec des exemples différents, ou identiques mais pas dans le même ordre)

R. Grin

ML

27

27

## Algorithme descente de gradient

- Algorithme d'optimisation qui permet de trouver des paramètres du modèle qui minimisent la fonction de perte
- S'il y a  $n$  paramètres, on se place dans un espace à  $n + 1$  dimensions ; cet algorithme revient à se déplacer dans cet espace (donc à modifier les paramètres) en prenant la direction de la plus grande pente descendante pour atteindre un minimum

En dimension 2 ( $n = 1$ )

Valeur fonction de perte en fonction valeur paramètre

R. Grin

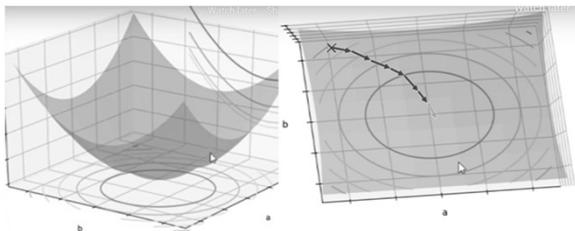
ML

28

28

## Algorithme de descente de gradient

En dimension 3 (2 paramètres) En projetant :



R. Grin

ML

29

29

## Données de test

- Il est nécessaire de garder une partie des exemples d'apprentissage pour tester le modèle après l'apprentissage
- Un modèle peut être surentraîné sur les données d'entraînement et ne pas donner des bons résultats avec d'autres données (overfitting, sur-apprentissage) ; il faut savoir s'arrêter à temps lors de l'apprentissage

R. Grin

ML

30

30

## Types d'apprentissages

- Supervisé (supervised learning)
- Non supervisé (unsupervised learning)
- Auto-supervisé (self-supervised learning)
- Par renforcement (reinforcement learning)

R. Grin ML 31

31

## Apprentissage supervisé

- Un être humain fournit des exemples de ce qu'on attend : des données structurées et étiquetées, avec des entrées (caractéristiques, features) et des sorties (variables cibles)
- Par exemple,
  - estimer le prix d'un appartement (variable cible) à partir de ses caractéristiques (étage, emplacement, superficie,...) en partant des caractéristiques et prix d'appartements récemment vendus
  - classer des photos entre des photos de chiens et de chats en partant d'exemples d'images étiquetées chien ou chat

R. Grin ML 32

32

## Apprentissage non supervisé

- Le logiciel doit trouver des patterns ou des relations entre des données, sans recevoir d'aide
- Les données d'entraînement ne sont pas étiquetées
- L'objectif peut être
  - d'extraire des classes d'individus présentant des caractéristiques communes (clustering) ; les définitions des classes ne sont pas données a priori
  - de trouver des associations entre les données ; par exemple indiquer que les clients qui achètent du pain achètent aussi souvent du beurre, en analysant un historique des achats
- Moins performant que l'apprentissage supervisé mais peut être intéressant si l'étiquetage est complexe ou coûteux

R. Grin ML 33

33

## Apprentissage auto-supervisé

- Forme intermédiaire entre l'apprentissage supervisé et non supervisé
- Le modèle apprend à partir d'échantillons de données non annotées et les pseudo-étiquettes sont générées automatiquement à partir des données d'entraînement
- Par exemple, on cache une partie des informations et le modèle doit retrouver cette partie cachée
- L'entraînement d'une IA générative peut se faire en tronquant des phrases récupérées dans un livre ou sur Internet ; l'IA doit retrouver la partie manquante

R. Grin ML 34

34

## Apprentissage par renforcement

- Le logiciel apprend en interagissant avec un environnement
- Il reçoit un feedback (retour d'information) en recevant des récompenses ou des pénalités en fonction des actions qu'il entreprend dans l'environnement
- Souvent utilisé quand les données d'entraînement sont rares ; l'IA apprend par essais et erreurs

R. Grin ML 35

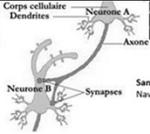
35

## Réseaux de neurones

R. Grin ML 36

36

## Cerveau humain

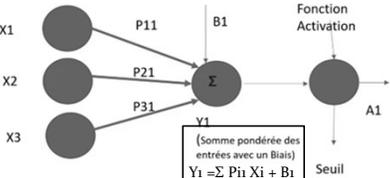


- 86 milliards de neurones
- Environ 2000 synapses (points d'entrée) par neurones ; le plus souvent un seul axone mais avec plusieurs ramifications (point de sortie)
- Chaque synapse peut être utilisé plusieurs centaines de fois pas seconde
- Très efficace : consommation de 25 watts (25 kwatts pour Big Blue)
- Apprendre c'est créer, supprimer des synapses, modifier leur sensibilité

R. Grin ML 37

37

## Au début : le perceptron



- Créé par Frank Rosenblatt en 1957
- Un perceptron reçoit des valeurs en entrées  $X_1, X_2, X_3$  et retourne une valeur  $Y_1$  qui est passée à une fonction d'activation (qui active ou pas la sortie) pour donner le résultat final  $A_1$
- $P_i$  : poids de l'entrée ;  $B_1$  : biais du neurone

R. Grin ML 38

38

## Neurones artificiels

- La fonction d'activation du perceptron renvoie 0 si la somme pondérée des entrées dépasse un certain seuil ; elle renvoie 1 si le seuil est dépassé
- La sortie finale dépend donc linéairement des entrées, ce qui est rigide ; par exemple, en dimension 2, une fonction affine ne peut représenter qu'une droite comme figure géométrique
- Les neurones artificiels modernes sont des évolutions du perceptron avec, en particulier, des fonctions d'activation qui permettent d'introduire du non linéaire

R. Grin ML 39

39

## Réseau de neurones artificiels

- Les neurones artificiels sont regroupés en réseaux
- Ces réseaux sont structurés en couches de neurones

R. Grin ML 40

40

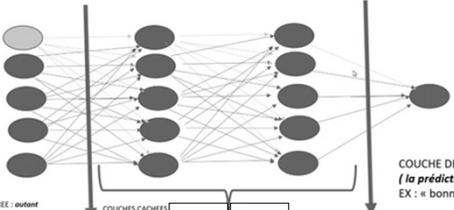
## Couches réseau de neurones

- La couche d'entrée est composée de neurones qui reçoivent des valeurs de l'extérieur du réseau
- La couche de sortie fournit à l'extérieur un résultat calculé par le réseau de neurones
- Entre ces 2 couches on trouve les couches intermédiaires qui sont cachées et qui participent au résultat calculé par le réseau
- Souvent, les neurones d'une couche sont connectés à tous les neurones de la couche précédente et de la couche suivante

R. Grin ML 41

41

## Exemple



COUCHE D'ENTRÉE : autant de neurones que de variables (ici 5)

COUCHES CACHÉES : Généralement deux couches suffisent avec à minima, pour chacune, le même nombre de neurones que la couche d'entrée

COUCHE DE SORTIE (la prédiction)  
EX : « bonne affaire »

- Exemple pour estimation d'un bien immobilier avec 5 variables : maison, adresse, surface, prix, année

R. Grin ML 42

42

## Liaison entre neurones

- Chaque liaison a un poids qui traduit le degré d'influence du neurone en amont : plus le poids est grand, plus le neurone en amont exerce une grande influence sur l'activation du neurone
- Un neurone reçoit en entrée la somme des valeurs venant des autres neurones auquel il est connecté en sortie, pondérée par les poids des liaisons, à laquelle on ajoute le biais associé au neurone :

$$y_j = \sum_{i=1,n} p_{ij}x_i + b_j$$

R. Grin

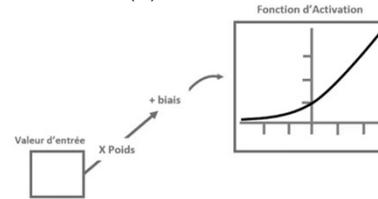
ML

43

43

## Fonction d'activation

- Pour introduire du non-linéaire, donc plus de souplesse et de complexité, la valeur reçue par un neurone est donnée en entrée d'une fonction d'activation  $f$  pour donner la valeur finale émise par un neurone  $V = f(A)$



R. Grin

ML

44

44

## Types de fonctions d'activation

1. **Sigmoïde** :  $\sigma(x) = 1 / (1 + e^{-x})$  - Produit des sorties dans la plage de 0 à 1 ; couramment utilisée dans les couches de sortie pour les problèmes de classification binaire.
2. **Tangente hyperbolique (tanh)** :  $\tanh(x) = (e^{-x} - e^x) / (e^{-x} + e^x)$  - Produit des sorties dans la plage de -1 à 1, permettant de gérer les valeurs négatives.
3. **ReLU (Rectified Linear Unit)** :  $\text{ReLU}(x) = \max(0, x)$  - Renvoie simplement zéro pour les valeurs négatives et laisse les valeurs positives inchangées. Une des fonctions d'activation les plus populaires pour les couches cachées.
4. **Leaky ReLU** :  $\text{Leaky ReLU}(x) = \max(\alpha x, x)$  - Variation de ReLU qui permet un faible taux de fuite ( $\alpha$  est une petite valeur constante pour les entrées négatives) pour éviter certains problèmes liés à ReLU.
5. **Softmax** : Souvent utilisée dans la couche de sortie pour les problèmes de classification multi-classes. Elle convertit un vecteur de nombres réels en une distribution de probabilité.

R. Grin

ML

45

45

## Paramètres du réseau

- L'ensemble des poids  $p_{ij}$  de toutes les liaisons entre neurones et les biais  $b_j$  de chaque neurone constituent l'ensemble des paramètres du réseau ; ils déterminent le comportement du réseau
- Ce sont ces paramètres qui sont calculés lors de la phase d'apprentissage du réseau

R. Grin

ML

46

46

## Back propagation (propagation arrière)

- Processus clé dans l'entraînement des réseaux neuronaux artificiels
1. Durant cet entraînement, les données d'apprentissage sont introduites dans le réseau
  2. Les erreurs dans les prédictions du modèle sur ces données sont calculées à l'aide de la fonction de perte
  3. Pendant la « back propagation », les poids des liaisons entre neurones sont ajustés pour minimiser la valeur de la fonction de perte (le plus souvent en utilisant la descente de gradient)
- Ces 3 étapes peuvent être répétées pour minimiser l'erreur des prédictions

R. Grin

ML

47

47

## ONNX

- Open Neural Network Exchange
- Format ouvert et open source pour représenter les modèles
- Permet d'échanger des modèles entre différents frameworks et environnements d'exécution
- Facilite l'interopérabilité entre les outils de DL comme PyTorch, TensorFlow, etc.

R. Grin

ML

48

48



## Vocabulaire

- Ensemble des tokens utilisés dans une langue (ou dans plusieurs langues)

R. Grin

ML

55

55

## Réseaux de neurones récurrents

- Les RNNs ont été introduits pour traiter les données ordonnées en séquences : texte, image, son,..
- Le traitement utilise une chaîne de réseaux de neurones (le même réseau est utilisé dans toute la chaîne) ; à chaque étape la sortie d'un réseau de la chaîne est donnée en entrée au réseau suivant
- Exemple : 1<sup>er</sup> mot d'une phrase est donné en entrée au réseau puis on donne en entrée au 2<sup>ème</sup> réseau la sortie du 1<sup>er</sup> réseau et le 2<sup>ème</sup> mot de la phrase, etc.

R. Grin

ML

56

56

## Problèmes RNNs

- Traitement séquentiel des tokens, donc apprentissage long et moins performant pendant leur utilisation
- Difficulté à capturer les relations complexes entre les éléments de la séquence ou les relations entre éléments éloignés
- Biais temporel : les premiers éléments d'une séquence sont traités différemment des derniers
- Les transformeurs réduisent fortement ces problèmes

R. Grin

ML

57

57

## Transformeurs

R. Grin

ML

58

58

## Présentation

- Architecture de réseau de neurones utilisée pour traiter des séquences de données : texte, image, vidéo, série temporelle,...
- Exemples d'utilisations : traduction, génération ou résumé de textes, chat
- Toutes les IA génératives le plus connues s'appuient sur des transformeurs : OpenAI (ChatGPT), Gemini, Llama, ...

R. Grin

ML

59

59

## Génération de texte

- Grandes étapes pour générer du texte :
  1. Le texte en entrée (par exemple question utilisateur) est enrichi dans des structures qui capturent des informations sur le sens, la grammaire, la syntaxe, les tokens les plus importants,...
  2. A partir de la séquence d'entrée enrichie, le prochain token est généré (souvent le plus probable, ou l'un des plus probables, d'après la phase d'apprentissage)
  3. Le nouveau texte (on ajoute le dernier token généré) est envoyé à l'étape 1, et ainsi de suite, jusqu'à obtenir la réponse complète

R. Grin

ML

60

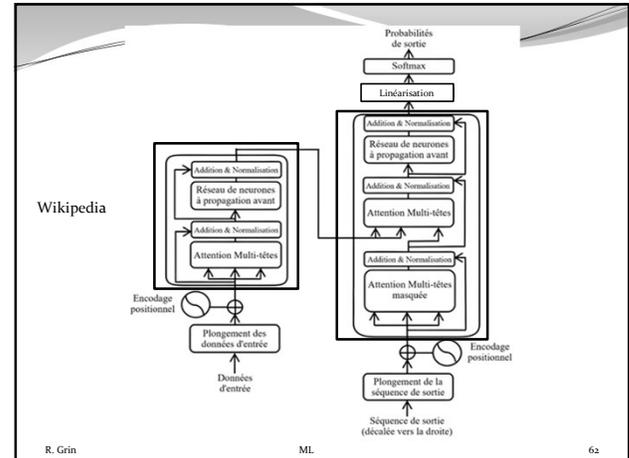
60

## Structure d'un transformeur

- Essentiellement 2 parties : encodeur et décodeur
- En fait, une pile d'encodeurs et une pile de décodeurs identiques ; taille de la pile 6 dans article fondateur des transformeurs

R. Grin ML 61

61



62

## Embeddings

- Représentation sous la forme d'un vecteur de nombres réels, d'un mot, d'un texte ou, plus généralement, d'un objet (image, audio, graphe,...)
- Ils permettent d'associer du sens au texte ; ainsi des textes ayant un sens proche ont des embeddings proches
- Les textes sont transformés en embeddings pour être traités par un transformeur
- Etudiés en détails dans le prochain support

R. Grin ML 63

63

## Encodeur

- Transforme la séquence d'entrée (par exemple un texte qui représente une question, ou un texte à traduire) en une séquence d'embeddings
- Les embeddings sont traités en parallèle
- Utilise un mécanisme d'**attention** qui permet de s'intéresser plus particulièrement à certaines parties de la séquence d'entrée et de détecter les relations entre ces parties, même si elles sont éloignées dans la séquence
- Le résultat est une représentation enrichie de la séquence d'entrée

R. Grin ML 64

64

## Décodeur

- De base, prend une séquence de tokens et sort le token suivant (itérativement une séquence de tokens)
- Si traitement préalable par encodeur, utilise la représentation enrichie de la séquence d'entrée fournie par l'encodeur pour générer une séquence de sortie
- Génère la séquence de sortie token par token, en se basant sur les informations fournies par l'encodeur, et sur les tokens qu'il a déjà générés
- Pour chaque itération, les probabilités de tous les tokens du vocabulaire sont calculées et le transformeur en choisit un (le choix dépend des hyperparamètres)

R. Grin ML 65

65

## Encodeur ou/et décodeur

- Peuvent être utilisés indépendamment
- Modèles uniquement encodeurs : pour tâches qui nécessitent une compréhension de l'entrée, comme classification de phrases, analyse de sentiments, recherche de mots masqués
- Modèles uniquement décodeurs : pour tâches génératives telles que la génération de texte
- Modèles encodeurs-décodeurs : pour tâches génératives qui nécessitent une entrée, telles que la traduction ou le résumé de texte

R. Grin ML 66

66

- La suite détaille les processus qui s'exécutent dans les transformeurs

R. Grin

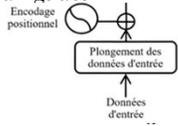
ML

67

67

## Travail préparatoire pour encodeur et décodeur

1. Séquence d'entrée est découpée en tokens
2. Chaque token est transformé en embedding, vecteur numérique de grande dimension
3. Tous les embeddings sont traités en parallèle (gros avantage sur les RNNs)
4. Positional encoding : ajoute aux embeddings des informations sur la position des tokens



R. Grin

ML

68

68

## Codage de position

- Permet de traiter les tokens de la séquence d'entrée en parallèle, sans perdre leur position dans la séquence
- L'information de position est une valeur sinusoïdale, pour des raisons techniques
- Cette information est un vecteur de même longueur que les embeddings, ajoutée à l'embedding de chaque token

R. Grin

ML

69

69

## Addition et normalisation

- Chaque étape importante des encodeurs et décodeurs est suivie d'une étape d'addition et de normalisation
- Addition : ajoute aux embeddings des informations fournies par l'étape précédente (par exemple informations d'attention) pour que cette information ne soit pas « oubliée » car déformée par la dernière étape
- Normalisation : il peut y avoir des variations importantes d'amplitude dans les valeurs fournies par les différentes étapes, ce qui peut fausser le travail effectué dans les étapes suivantes ; la normalisation contrôle ces variations

R. Grin

ML

70

70

## Partie 1 : encodeur

Enrichit la séquence d'entrée

R. Grin

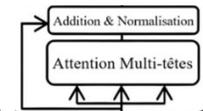
ML

71

71

## Multi-head (ou self-) attention

5. Etape très importante qui permet à l'encodeur de se concentrer sur les tokens (embeddings, en fait) importants de la séquence d'entrée et de capturer à quel point chaque token est relié à chaque autre token ; à la sortie les embeddings sont enrichis avec l'attention
6. Embeddings avec position du départ ajoutés aux embeddings avec attention calculés, pour éviter une éventuelle perte d'information après le traitement d'attention



R. Grin

ML

72

72

## Utilité de l'étape d'attention

- Sens des mots :
  - « Je n'ai pas pu aller à mon cours de batterie car la batterie de ma voiture est tombée en panne »
  - La seule façon de distinguer les 2 sens du mot « batterie » est d'analyser le contexte qui l'entoure ; l'attention le fait (relation de « batterie » avec « cours » ou bien avec « voiture »)
- Relation entre les mots :
  - Jean a montré à Paul où il devait signer.
  - L'attention va permettre d'associer « il » à Paul

R. Grin

ML

73

73

## Pourquoi « plusieurs têtes » ?

- On peut prêter attention à plusieurs choses dans une phrase
- De plus, une phrase peut être ambiguë : « Robert a vu Bernard avec le télescope » peut avoir 2 sens
- Une tête d'attention va associer « Robert », « a vu » et « Bernard », alors qu'une autre tête va associer « Bernard » et « avec le télescope »
- Les différentes têtes peuvent identifier différentes relations entre les mots ; les embeddings captureront ces différentes interprétations qui pourront être résolues par la suite avec le contexte

R. Grin

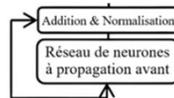
ML

74

74

## Réseau de neurones à propagation avant

7. Feed-forward layer : les embeddings contextuels (enrichis) sont transmis à un réseau de neurones
8. Le réseau de neurones permet d'extraire les caractéristiques de plus haut niveau, grâce à ses capacités linguistiques et à ses connaissances générales



R. Grin

ML

75

75

## Partie 2 : décodeur

Génère la séquence de sortie, token par token, à partir de la séquence d'entrée enrichie par le travail de l'encodeur

R. Grin

ML

76

76

## Globalement

- Au début du traitement, la séquence de sortie du décodeur est vide (en fait elle contient un token spécial de début de séquence) ; elle se remplit au fur et à mesure du traitement
- Celui-ci génère alors le 1<sup>er</sup> token qui est ajouté à la séquence de sortie du décodeur et on revient au début du traitement du décodeur
- Le décodeur génère ensuite le 2<sup>ème</sup> token, ajouté lui aussi à la séquence de sortie, et ainsi de suite, jusqu'à ce que le décodeur génère un token d'arrêt

R. Grin

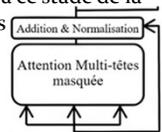
ML

77

77

## Masked multi-head attention

- « Self-attention »
- Comme pour l'encodeur, ajoute des informations d'attention sur la séquence de sortie déjà générée pour comprendre les relations entre ses tokens
- « Masked » car, pendant l'entraînement, on masque les tokens qui suivent le dernier token généré pour simuler ce qui va se passer pendant l'inférence : à ce stade de la génération, on ne connaît pas les tokens à droite du dernier mot généré



R. Grin

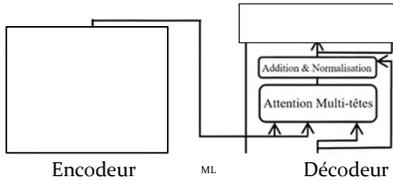
ML

78

78

## Multi-head attention

- « Cross-attention » ; différente de « self-attention » car utilise la séquence de sortie déjà générée **et** la séquence d'entrée enrichie par l'encodeur
- Enrichit les embeddings de la sortie en intégrant ceux de l'entrée ; particulièrement utile pour les tâches de traduction

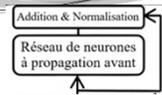


R. Grin ML 79

79

## Réseau de neurones

- Etape qui va affecter des scores à chaque token du vocabulaire, pour le choix du prochain token à générer
- Ces scores vont être convertis en probabilités par l'étape suivante



R. Grin ML 80

80

## Linéarisation

- Passage des embeddings aux tokens
- Prend en entrée la séquence enrichie par les étapes précédente
- Utilise tout ce qui a été appris dans la phase d'apprentissage pour attribuer des scores à chaque token du vocabulaire de la langue utilisée

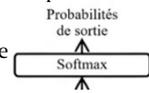


R. Grin ML 81

81

## Softmax, probabilités de sortie

- Softmax : Fonction qui génère des probabilités ; elle indique les chances que le modèle choisisse chaque token comme prochain dans la séquence
- Prédiction prochain token : peut être faite de plusieurs manières :
  - Choisir le token le plus probable
  - Choisir le token parmi les plus probables
  - Choix au hasard en tenant compte des probabilités



R. Grin ML 82

82

## Références

## IA pour les décideurs

- Présentation de l'IA pour les décideurs (2021 ; UNESCO):  
<https://unesdoc.unesco.org/ark:/48223/pf0000380006>

R. Grin ML 83

83

R. Grin ML 84

84

## MOOCs gratuits

- L'Intelligence Artificielle... avec intelligence ! : <https://www.fun-mooc.fr/fr/cours/lintelligence-artificielle-avec-intelligence/>
- Intelligence artificielle pour et par les enseignants : <https://www.fun-mooc.fr/fr/cours/intelligence-artificielle-pour-et-par-les-enseignants-ai4t/>

R. Grin

ML

85

85

- Lexique IA : <https://www.editions-eni.fr/lexique-intelligence-artificielle-chatgpt>

R. Grin

ML

86

86

## Vidéo sur IA

- Vidéo de 2 h 45 très intéressante de Yann Lecun sur l'IA en général et les limites des LLMs, en particulier des modèles non open source : <https://www.youtube.com/watch?v=5tivTLU7s4o>

R. Grin

ML

87

87

## Vidéos pour machine learning

- <https://www.youtube.com/channel/UCtYLUtG3k1Fg4y5tAhLbw> ; StatQuest essaie de vulgariser les statistiques et le machine learning
- <https://www.youtube.com/watch?v=trWrEWfhTVg> de David Louapre, sur le deep learning
- Tour d'horizon des modèles et algorithmes en 2 vidéos, avec des descriptions rapides de chaque modèle et algorithme ; Machine Learnia, Guillaume Saint-Cirgue : <https://www.youtube.com/watch?v=mT6NnslbNLM>

R. Grin

ML

88

88

## Vidéos pour réseaux de neurones

- <https://www.youtube.com/watch?v=7ell8KEbhJo>, en français ; à voir pour commencer ; David Louapre
- <https://www.youtube.com/watch?v=XUFLq6dKQok> (Machine learnia), en français, une série pour étudier les algorithmes et mathématiques utilisés par IA

R. Grin

ML

89

89

- Cours coursera sur deep learning : <https://www.coursera.org/specializations/deep-learning> (accès gratuit pendant 7 jours seulement)
- Hugging Face, site Web pour ceux qui travaille avec le deep learning : <https://huggingface.co/>

R. Grin

ML

90

90

- <https://www.youtube.com/watch?v=rniEE6M7fA> : courte vidéo qui explique rapidement le deep learning et qui utilise TensorFlow (bibliothèque qui permet de réaliser le réseau de neurones) et Keras (bibliothèque pour faire du deep learning en entraînant l'IA) ; fait partie de la série « Informatique sans complexe » qui contient d'autres vidéos courtes sur l'IA et sur divers autres domaines de l'informatique : <https://www.youtube.com/@InformatiqueSansComplexe>

R. Grin ML 91

91

- Site Web Kaggle pour une communauté d'utilisateur de IA et de ML (Machine Learning). Pour apprendre (cours, guides), pour les développeurs, pour les chercheurs, et des projets sous la forme de compétitions ou autres formes. <https://www.kaggle.com/>
- Article de vulgarisation très intéressant sur le phénomène de « grokking » : <https://scienceetonnante.substack.com/p/grokking-les-modeles-dia-sont-ils>

R. Grin ML 92

92

## Transformeurs (1/2)

- Cours Hugging Face : <https://huggingface.co/learn/nlp-course/chapter1/1>
- Visual Guide to Transformer Neural Networks : [https://www.youtube.com/playlist?list=PL86uXYUJ7999zE8u2-97i4KG\\_2Zpufkfb](https://www.youtube.com/playlist?list=PL86uXYUJ7999zE8u2-97i4KG_2Zpufkfb)
- Le transformer illustré : <https://a-coles.github.io/2020/11/15/transformer-illustre.html> (en français), <https://jalammar.github.io/illustrated-transformer/>, en vidéo : <https://www.youtube.com/watch?v=-QH8fRhgFHM>
- <https://lbourdois.github.io/blog/nlp/Transformer/>

R. Grin ML 93

93

## Transformeurs (2/2)

- Article fondateur des transformeurs « Attention is all you need », <https://arxiv.org/abs/1706.03762>
- Vidéo sur transformers, par Batool Haider : [https://www.youtube.com/playlist?list=PL86uXYUJ7999zE8u2-97i4KG\\_2Zpufkfb](https://www.youtube.com/playlist?list=PL86uXYUJ7999zE8u2-97i4KG_2Zpufkfb)

R. Grin ML 94

94