Inventer n’est pas prédire

IA générative

Apprentissage supervisé

On des exemples :
Exemple (X1, Y1)
Exemple (X2, Y2)
Exemple (X3, Y3)
…

Entrée X → → Prédiction Y

Génération

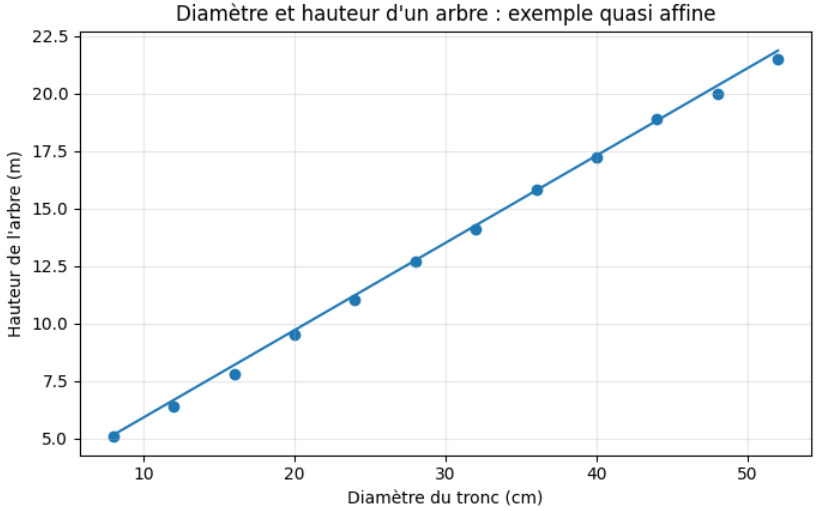
On des exemples :
Exemple X1,
Exemple X2,
Exemple X3,
…

 → Génération de Y

Promenons-nous dans les bois…

donnees_arbre_affine_realiste

Diamètre du tronc (cm)	Hauteur de l'arbre (m)
8	5.1
12	6.4
16	7.8
20	9.5
24	11.0
28	12.7
32	14.1
36	15.8
40	17.2
44	18.9
48	20.0
52	21.5



Pour voir ce qui n'y est pas…

- On peut résumer l'information du tableau par l'équation suivante:
  - $h(d) = 0{,}38\,d + 2{,}1$ où d est en cm, et $h(d)$ en m.

- Mais l'équation ne résume pas seulement les données : elle permet d'en trouver d'autres !
  - $h(30) = 0{,}38\,.\,30 + 2{,}1 = 13{,}5$
  - $h(22) = 0{,}38\,.\,22 + 2{,}1 = 10{,}46$

- Ces deux valeurs (30, 22) ne font pas partie du tableau :
  - l'équation représente l'information du tableau et permet de trouver des réponses à des questions que le tableau de données ne sait pas traiter.
  - L'équation permet d'oublier les données de départ elles sont abstraites dans cette équation qui les représentent tout en permettant de les extrapoler.

L'Iag et la recherche d'information

- Recherche d'information :
  - Le principe est de trouver une réponse à une question posée par un usager en lui fournissant un document apportant la réponse.
- Approche :
  - La question formulée en mots clés synthétisant l'information recherchée
    - C'est le requêtage
  - Les documents sont décrits en mots-clés synthétisant l'information contenue
    - C'est l'indexation
  - La recherche apparie les mots-clés de la requête avec ceux de l'indexation
    - Les documents indexés par les mots-clés de la requête sont proposés.
- Mise en œuvre :
  - On a un tableau mettant en correspondance un mot-clé avec tous les documents qui sont indexés par lui (tables inverses)

De la recherche à la génération d'information

- Les requêtes :
  - Un nombre de la colonne diameter / Mot-clé
- Les documents :
  - le contenu de la colonne hauteur / noms de document
- Question :
  - Que serait l'équivalent de l'équation pour représenter l'information documentaire ?
  - Cela permettrait de poser des questions différentes des index.

Mot clé	Noms de documents contenant ce mot clé
arbre	rapport_arbre_2024.pdf ; mesure_arbre_parc.docx ; inventaire_arbres.xlsx
climat	etude_climat_urbaine.pdf ; synthese_climat_2023.docx
budget	budget_previsionnel_2025.xlsx ; note_budget_service.docx
santé	rapport_sante_travail.pdf ; enquete_sante_agents.xlsx
formation	plan_formation_2026.docx ; bilan_formation_annuel.pdf
énergie	consommation_energie_batiment.xlsx ; audit_energie_ecole.pdf

Réponse : l'espace latent

- L'Iag représente l'information contenue dans les documents dans un espace abstrait, l'espace latent :
  - C'est l'équivalent de notre fonction affine
  - Les documents ne sont donc plus présents, on n'en a plus besoin, comme du tableau de données des diamètres d'arbre
  - On peut désormais poser une question à laquelle on répond avec les éléments calculés de l'espace latent, réponse qui n'appartient en propre à aucun document en particulier.
- Principe :
  - L'espace est un espace vectoriel de plusieurs centaines de dimensions (souvent 512)
  - Les « mots » des documents sont des points de cet espace ;
  - Les questions utilisant ces mots permettent de calculer des points voisins comme réponse, c'est-à-dire la génération d'autres mots que les documents auraient pu dire (mais n'ont pas dit).

Embedding (plongement)

- Principe :
  - Chaque token est associé à un vecteur qui donne sa composition en traits sémantiques. Ce vecteur est de taille variable, en général 512, voire 4096 dans les modèles plus récents.
  - Ces traits ne sont pas prédéfinis, mais obtenus par apprentissage : les tokens sont associés entre eux par des séquences issues des données d'apprentissage. Le vecteur est mis à jour à chaque fois qu'une association est constatée.
    - Intérêt :
      - Le nombre de dimension est bien plus petit que le nombre de mots du corpus. On n'a pas un espace avec une dimension par mot (chaque vecteur étant one-hot ou un parmi n).
      - Deux mots voisins auront des vecteurs proches, dont la similarité peut être calculée de manière vectorielle (cosinus par exemple). On ramène la comparaison à des projections entre vecteurs.
    - Fondement : la sémantique distributionnelle de Harris.

En gros…

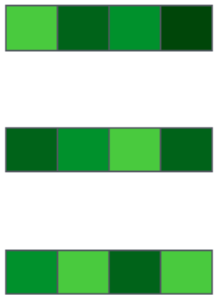
Input

Je

suis

étudiant

0.901	-0.651	-0.194	-0.822
-0.351	0.123	0.435	-0.200
0.081	0.458	-0.400	0.480



De l'ordre de
x0 000 tokens

En général, 512
ou 4096
dimensions

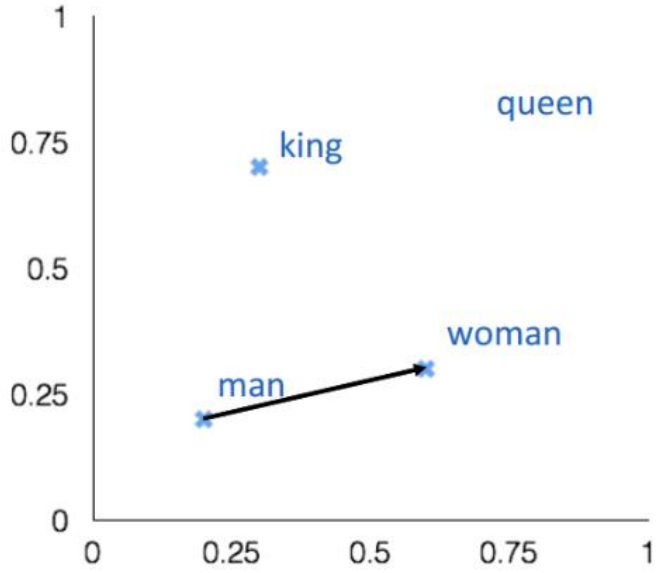
Un token = un vecteur
Généralisations :

- Une phrase, un vecteur
- Un document, un vecteur

L'intérêt : proximité vectorielle = proximité sémantique

man:woman :: king:?

+	king	[0.30 0.70]
-	man	[0.20 0.20]
+	woman	[0.60 0.30]
	queen	[0.70 0.80]



Lignée technique

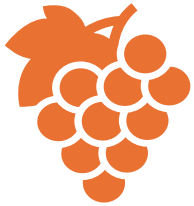
Comprise ainsi, l'IA générative :

N'est que l'étape suivante de l'instrumentation du paradigme de la recherche d'information ;

N'est pas une rupture;

Est plutôt l'aboutissement de ce paradigme :

- La recherche d'information s'intéresse à l'information à travers ses différentes expressions documentaires :
- L'IA générative permet de représenter l'information indépendamment des expressions documentaires ;



En gros :

Ce n'est pas vraiment surprenant qu'on utilise désormais chatGPT plutôt que Google…

Disparition des moteurs de recherche ?

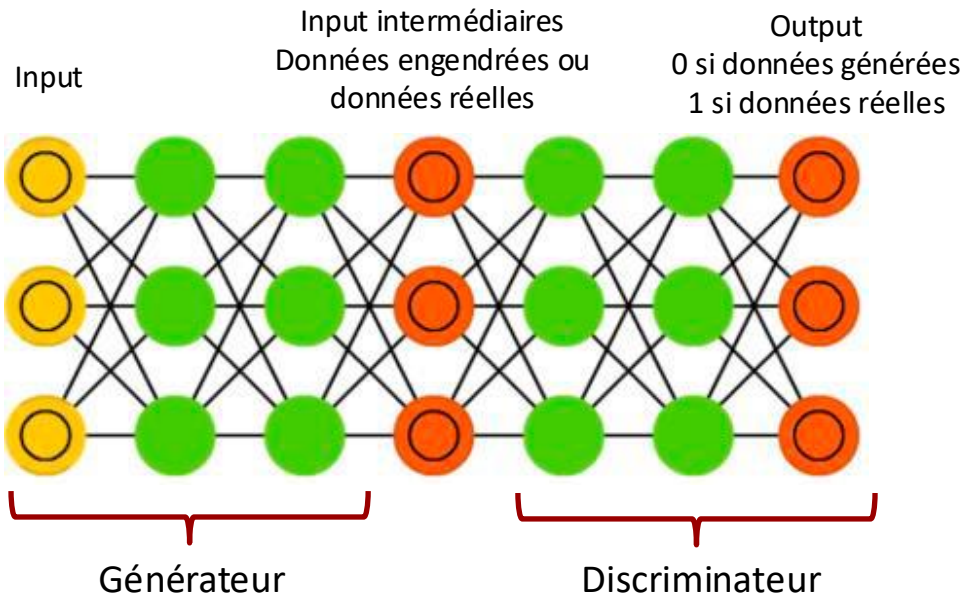
- Tendance :
  - Si on cherche de l'information et non de la documentation, les Iag sont destinées à remplacer les moteurs de recherche.
    - La future version d'Androïd serait livrée sans navigateur Web
- Résistance :
  - Sourcer l'information;
  - Faire la synthèse soi-même plutôt que la confier à un outil
- Hybridation :
  - Les Iag intégreraient les fonctionnalités des moteurs de recherche (sans les hallucinations actuelles)

Les generative adversarial networks

GAN : this person doesnot exist…

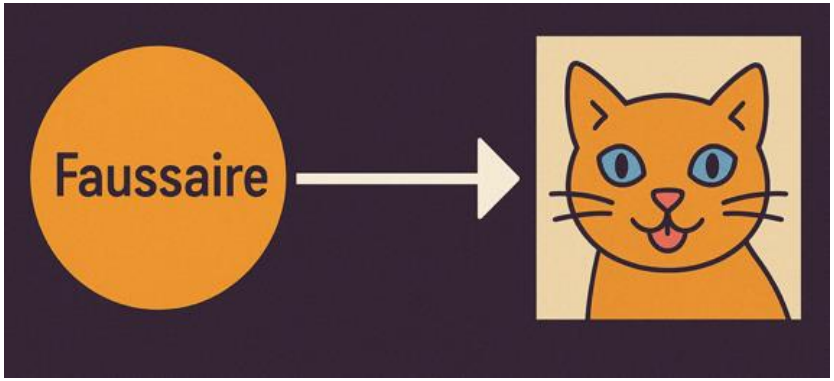
Generative Adversarial Networks (GAN) : principe

- Enjeu :
  - Créer des données qui ressemblent à des données de référence ou données « vraies »
- Exemple :
  - Créer des images de personnes qui n'existent pas mais qui semblent réelles
- Principe :
  - Un réseau générateur engendre des inputs qui ont l'air vrais
  - Un réseau discriminateur tente de distinguer les données engendrées des données réelles

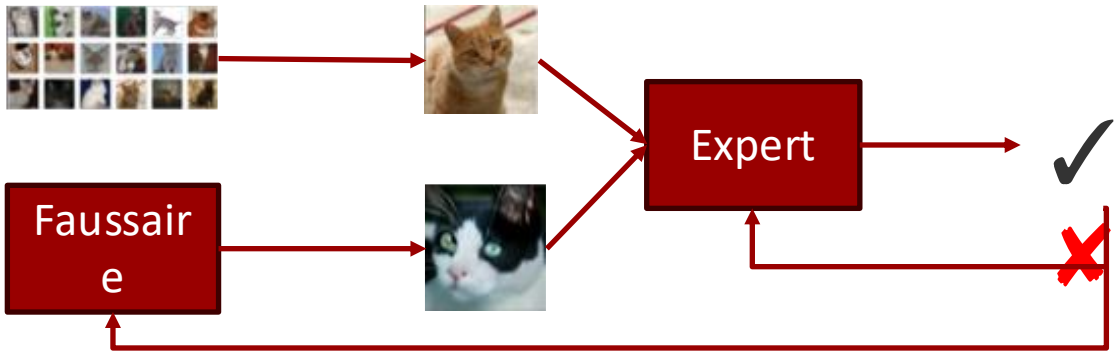


Deux réseaux adversaires : le faussaire / générateur pour tromper le discrimateur

- Objectif : engendrer des images de chats
- Principe : entraîné à partir d'une initialisation aléatoire
- Fonction de perte : il faut tromper l'expert / discriminateur

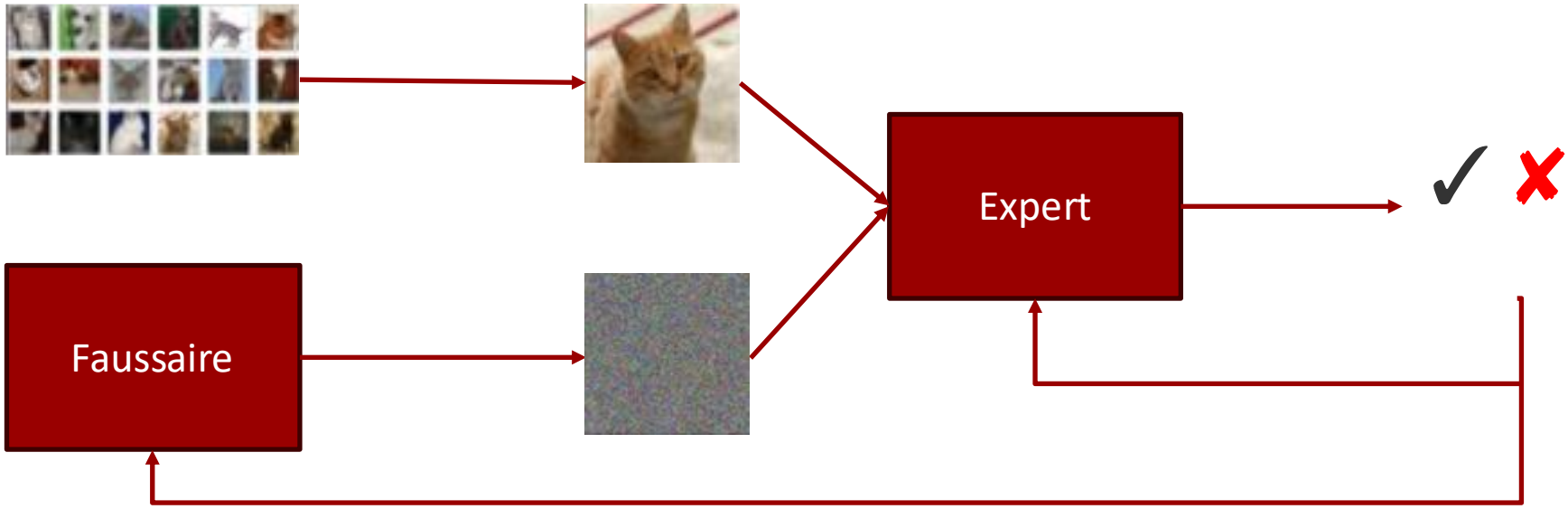


Deux réseaux adversaires : Le faussaire, l'expert

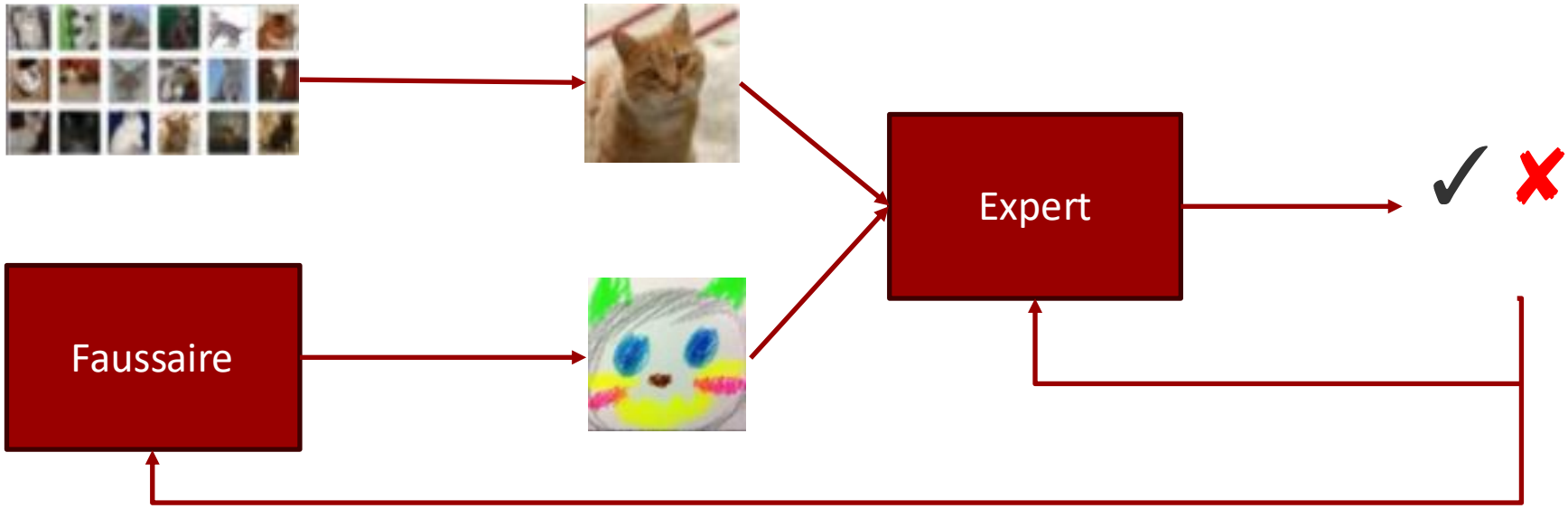


- Objectif : reconnaître une image réelle (dans la base d'apprentissage) d'une image inventée
- Principe : entraîné à partir d'une initialisation aléatoire
- Fonction de perte : s'il est trompé.

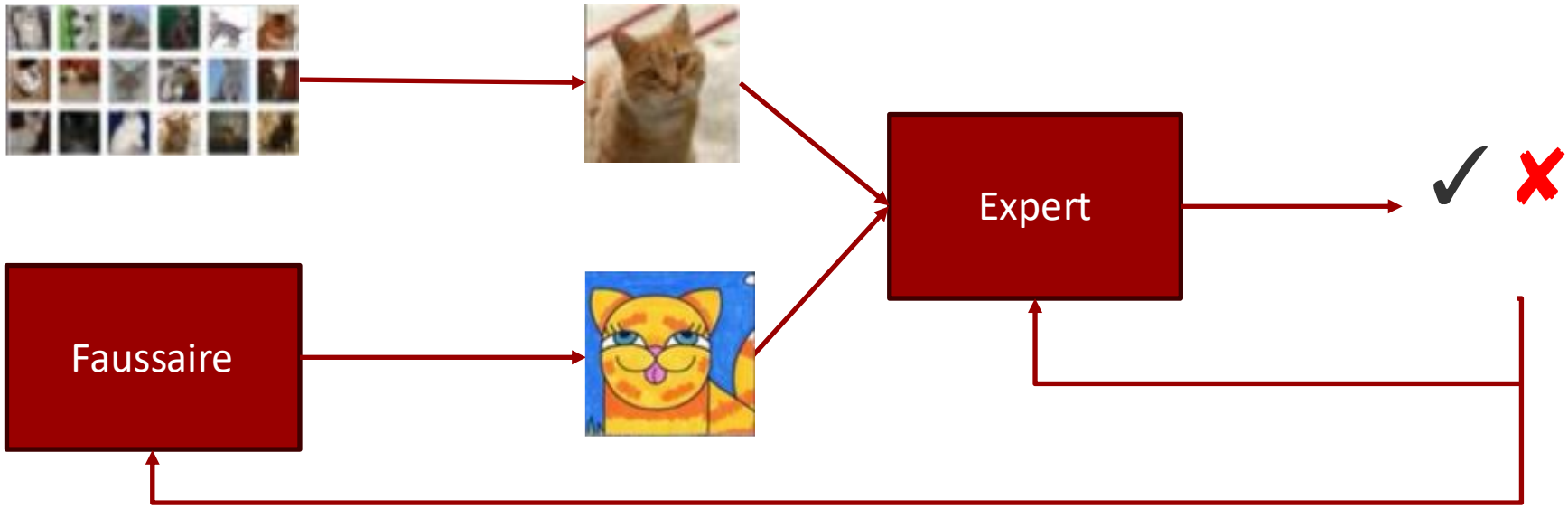
Ça commence mal…



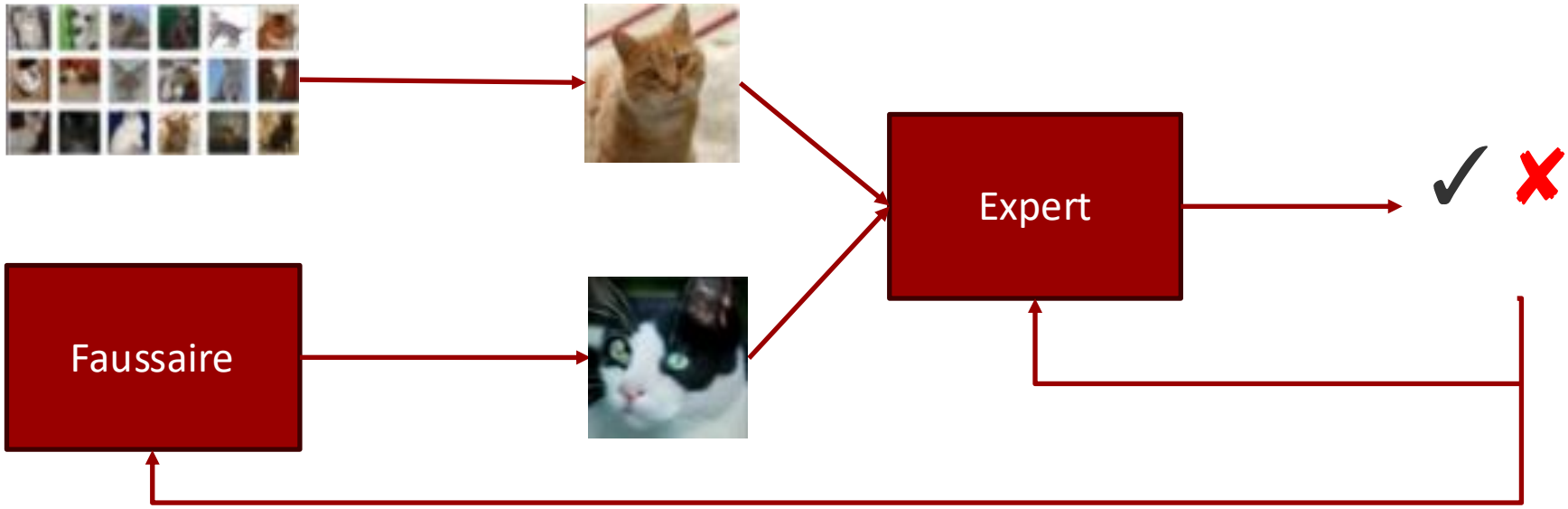
Puis ça s'améliore…



Encore un peu…



Ben c'est pas mal…

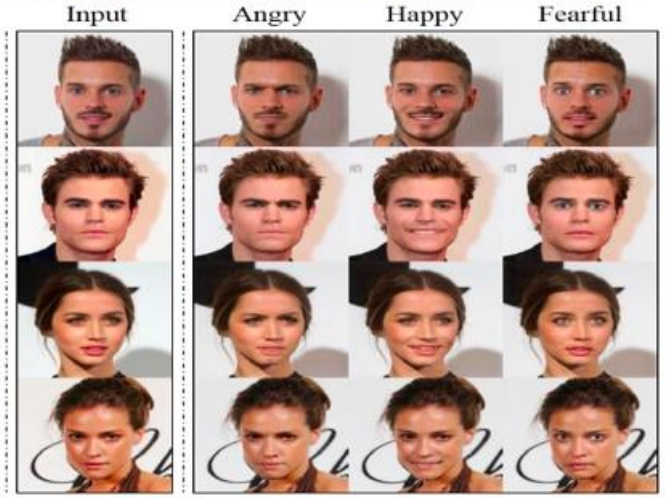


Comment ça marche…

- Méthode : on dispose de données réelles
- Première étape :
  - générateur initialisé aléatoirement, le discriminateur est entraîné (Inputs : données réelles ou générées, Outputs : 0 si généré, 1 si réel)
- Deuxième étape :
  - on fixe le discriminateur, et on entraîne le générateur à tromper le discriminateur (il faut que le discriminateur donne 1 au maximum de données générées)
- Ensuite :
  - on reprend l'étape 1 avec le nouveau générateur entraîné, puis l'étape 2 et ainsi de suite

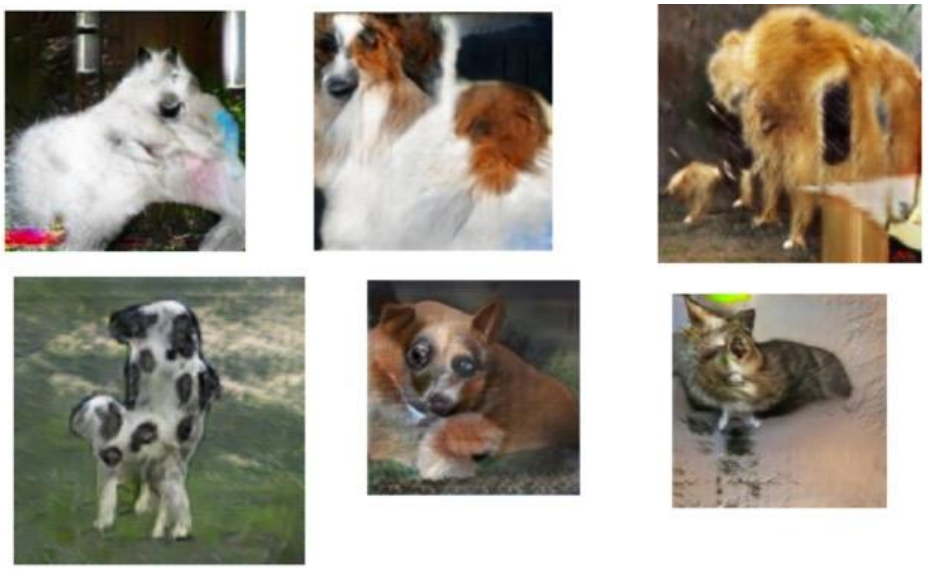
Gan : quand ça marche

StarGAN, *Choi et al.*, novembre 2017,
https://arxiv.org/abs/1711.09020



StarGAN, *Choi et al.*, novembre 2017,
https://arxiv.org/abs/1711.09020

Gan : quand ça ne marche pas…



Les modèles de diffusion

MidJourney, Dall-E

Engendrer des œuvres…

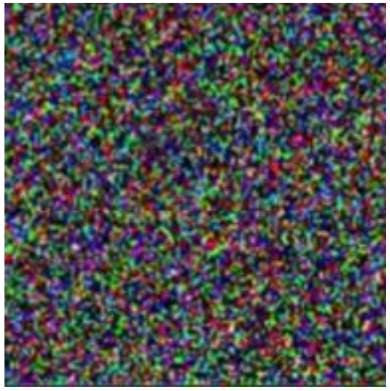
- Jason Allen et Midjourney 2022
- « Théâtre d'opéra spatial »

Bruiter...

95 %

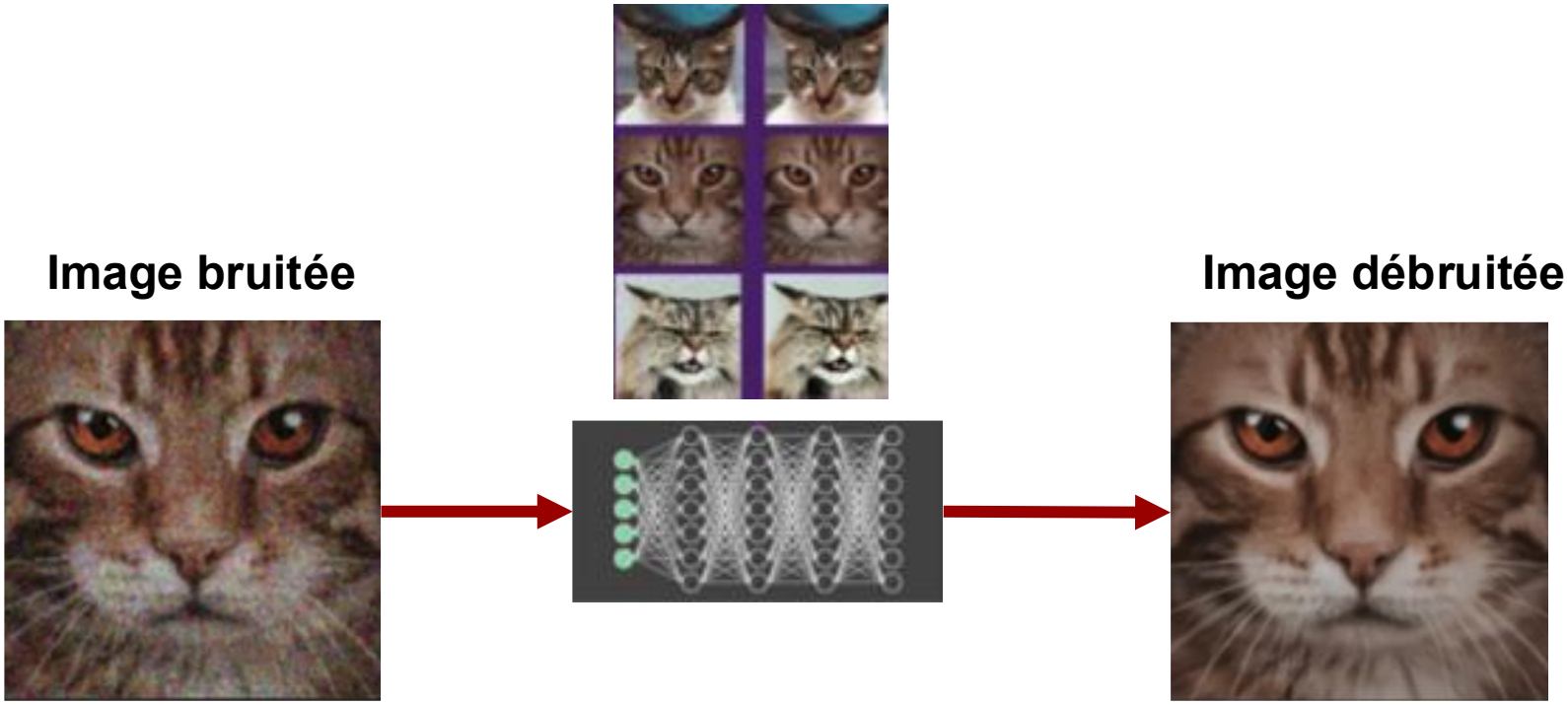
+

5 %

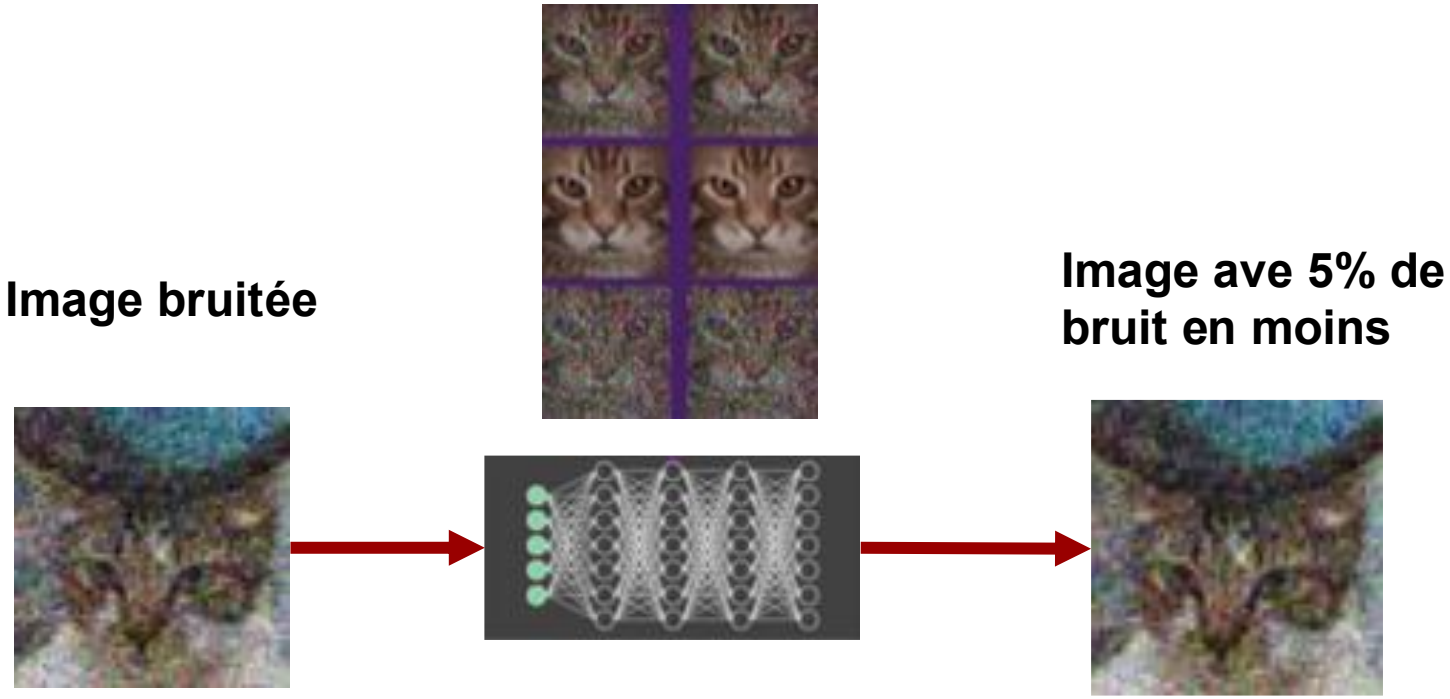


=

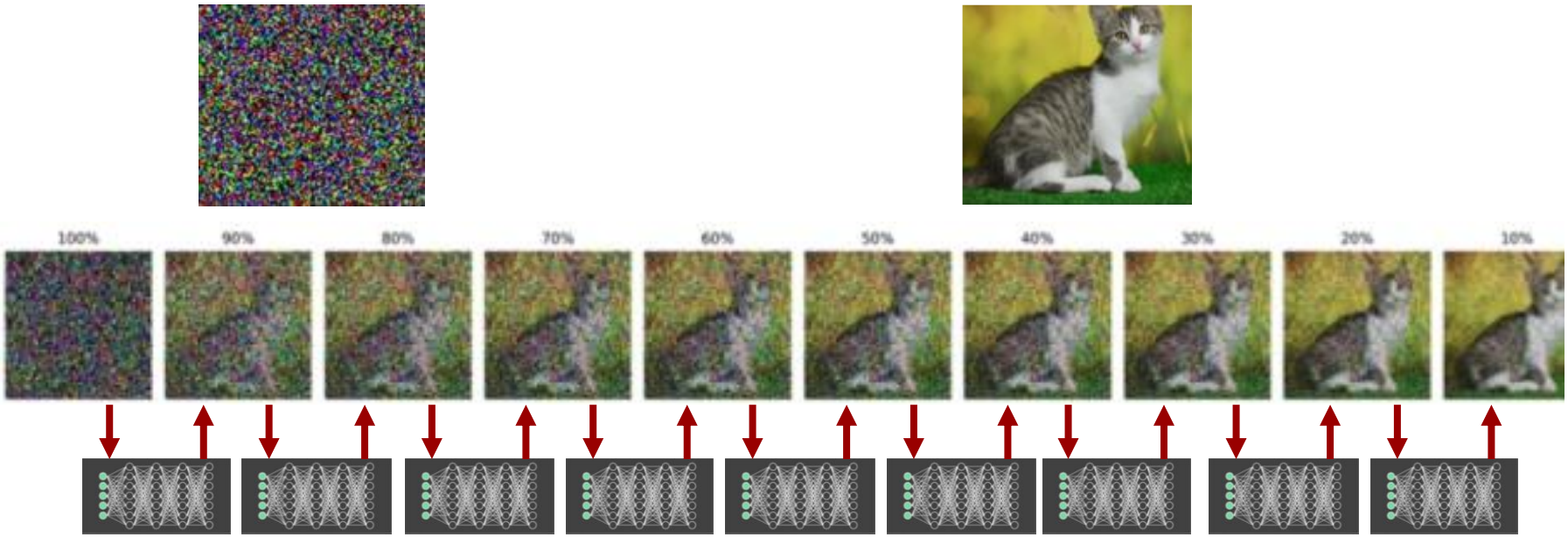
Pour apprendre à débruiter



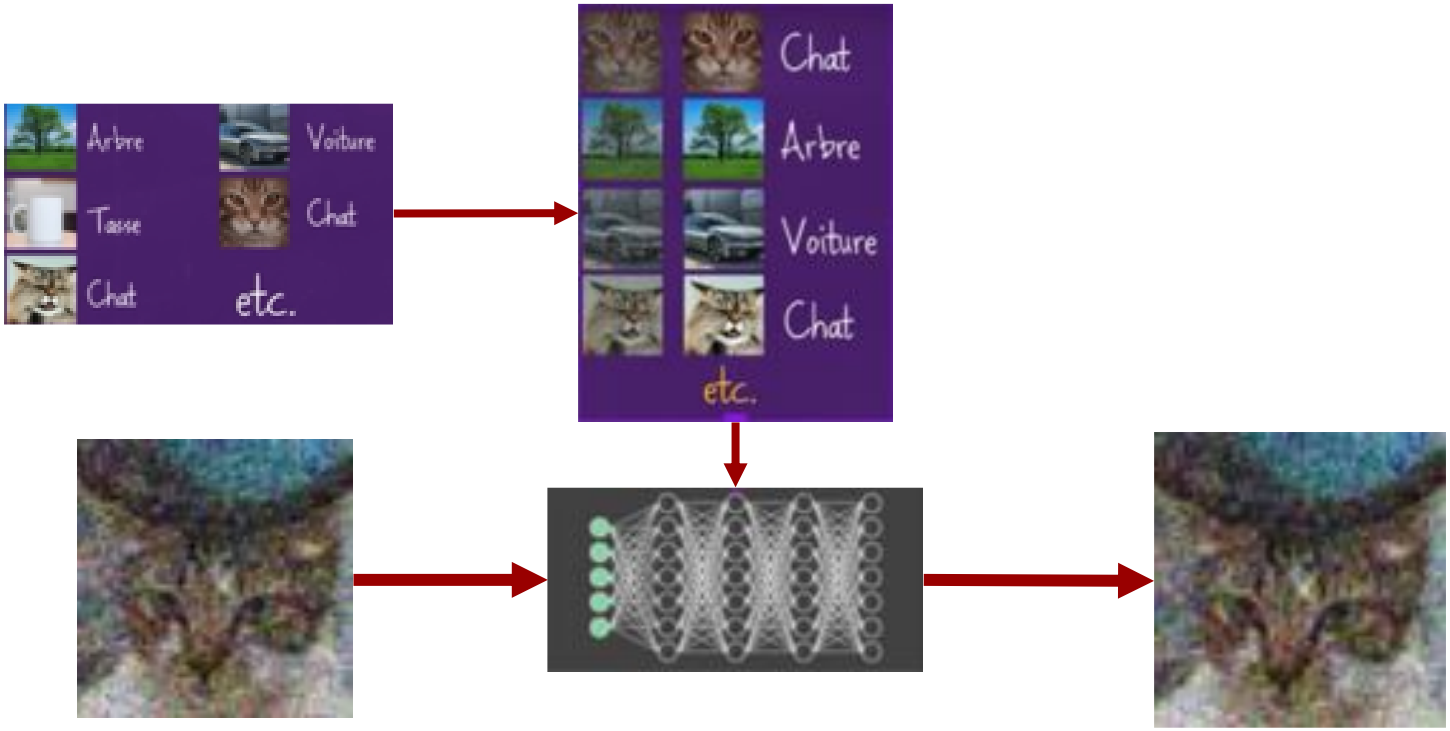
Apprendre à débruiter de manière progressive



Inventer en débruitant

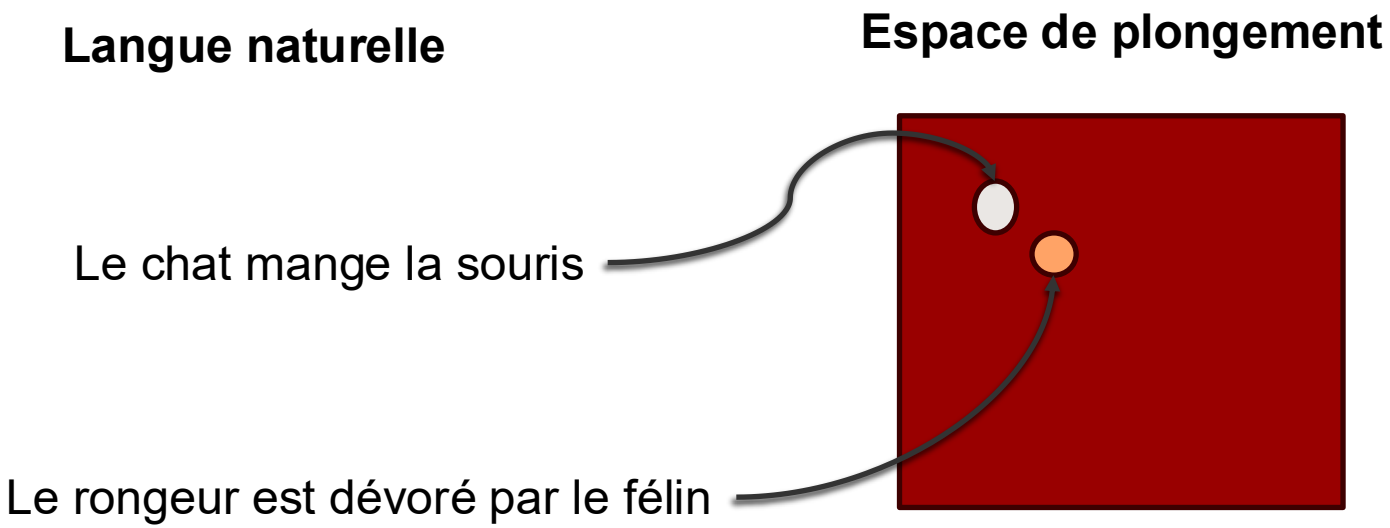


Conditionner l'invention par des catégories et les images associées



Conditionnement = chat 5 % de bruit en moins

Conditionner avec des phrases : codage vectoriel de phrases par leur plongement



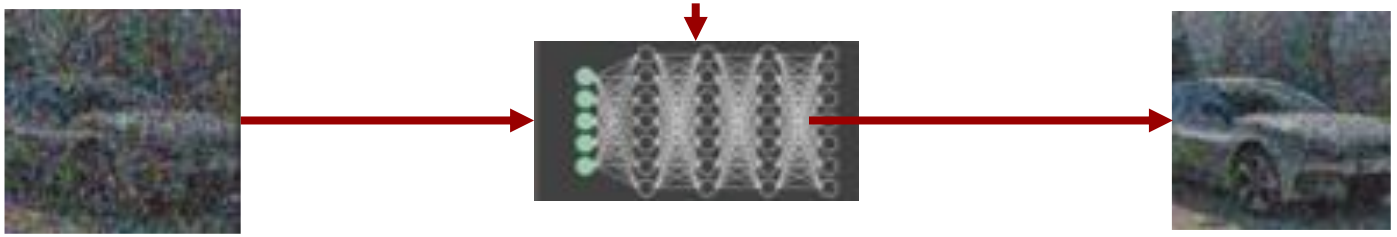
Conditionner par le plongement de phrases et des images associées

Une photo de la tête d'un chat

Un arbre isolé dans une prairie verte

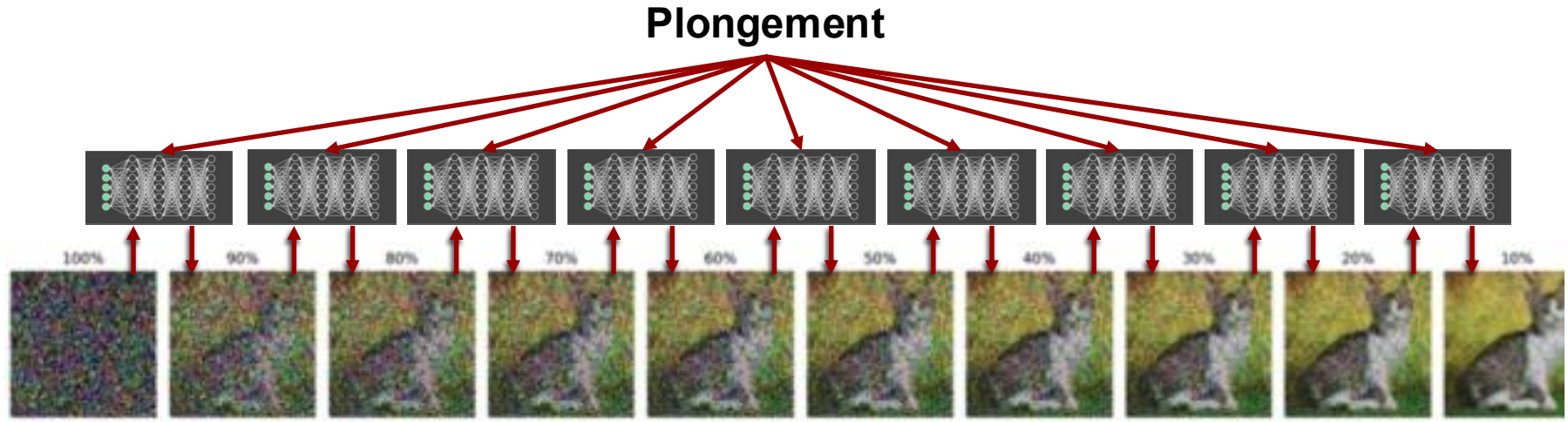
Une voiture grise en ville

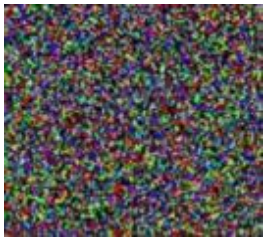
Tête d'un chat en colère



Conditionnement : plongement de
« une voiture grise en ville »

Au final…



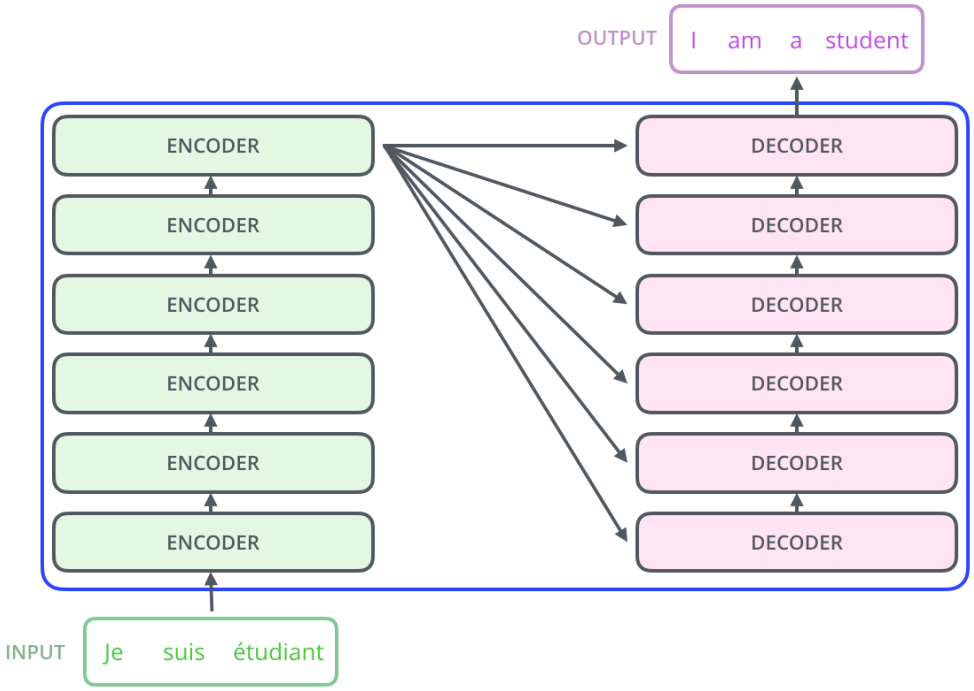


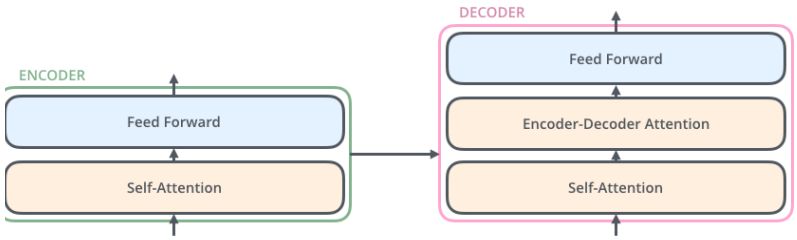
Transformers

L'idée en quelques mots

- On dispose d'une séquence en entrée et il faut produire une autre séquence : traduction, suite, réponse, catégorisation, etc.
- On s'appuie sur les réseaux de neurones. Mais ceux existants font mal le travail:
  - CNN : on reconnaît des motifs dans les données ; bien mais ne traitent pas les dépendances longues ni les données séquentielles ;
  - RNN : on analyse des séquences que l'on complète ; bien mais fonctionnent de manière itérative (et non parallèle), donc long à traiter.
- Donc, ce qu'on veut :
  - Considérer chaque terme de la séquence d'entrée de manière parallèle ;
  - Prendre en compte les dépendances entre les termes, même éloignés.
  - Traiter les séquences.
- Approche :
  - Représenter les termes par des codes numériques (tokenisation) ;
  - Chaque code correspond à un vecteur sémantique (embedding) ;
  - Calculer pour chaque position à compléter un vecteur ;
    - Coder en parallèle les termes en entrée, prenant en compte leur dépendance (l'attention dans les Transformers)
    - Calculer à partir de ce codage le code de chaque terme en sortie ;
  - Projeter ce vecteur sur le vocabulaire disponible et prendre celui qui a le meilleur score.

Principe


OUTPUT
I am a student
ENCODER
ENCODER
ENCODER
ENCODER
ENCODER
ENCODER
DECODER
DECODER
DECODER
DECODER
DECODER
DECODER
INPUT
Je suis étudiant


ENCODER
Feed Forward
Self-Attention
DECODER
Feed Forward
Encoder-Decoder Attention
Self-Attention

Les étapes

Tokenisation

- Traduire la phrase d'entrées en une liste de tokens (de l'ordre de 10 000).

Plongement (Embedding)

- Coder chaque token en un vecteur de dimension plus petite (512)

Couche d'encodage

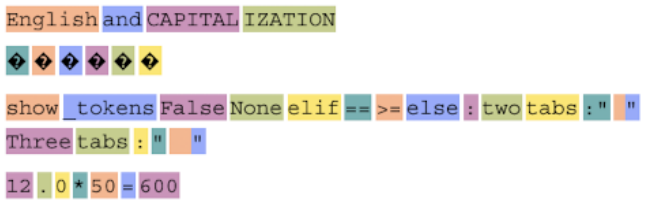
- Calcul de l'attention entre les mots, et propagation

Couche de décodage

- Prend en entrée une liste vide (inférence) qu'on va compléter mot à mot ;
- Prend en entrée une phrase cible (apprentissage) pour entraîner les réseaux ;
- A comme paramètre l'encodage produit par la première couche d'encodeurs ;
  - Ce paramètre conditionne l'attention calculée dans le décodage.

Tokenisation

- Soit le texte suivant :
  - text = """
  - English and CAPITALIZATION
  - 🎵鸟
  - show_tokens False None elif == >= else: two tabs:" " Three tabs: " "
  - 12.0*50=600
  - """
- Sa tokenisation (GPT 4) :

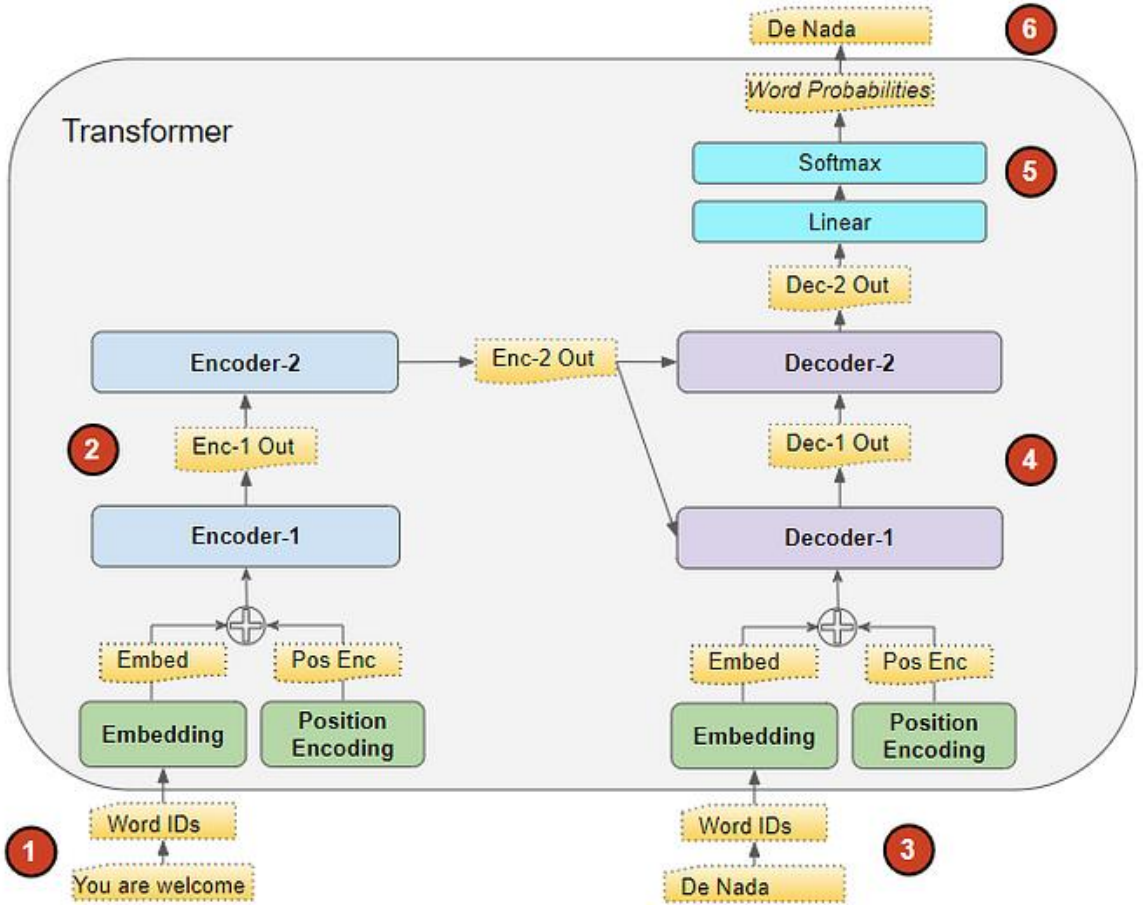


- Intérêts:
  - Traiter tous les types de caractères / symboles
  - Réduire la variabilité des formes en les réduisant à des formes élémentaires (token) que l'on peut combiner.
  - 100 000 (GPT4), 50000 (GPT2)
- Fonctionnement :
  - Apprentissage
  - Dépend des langues
  - Paramétrages variés

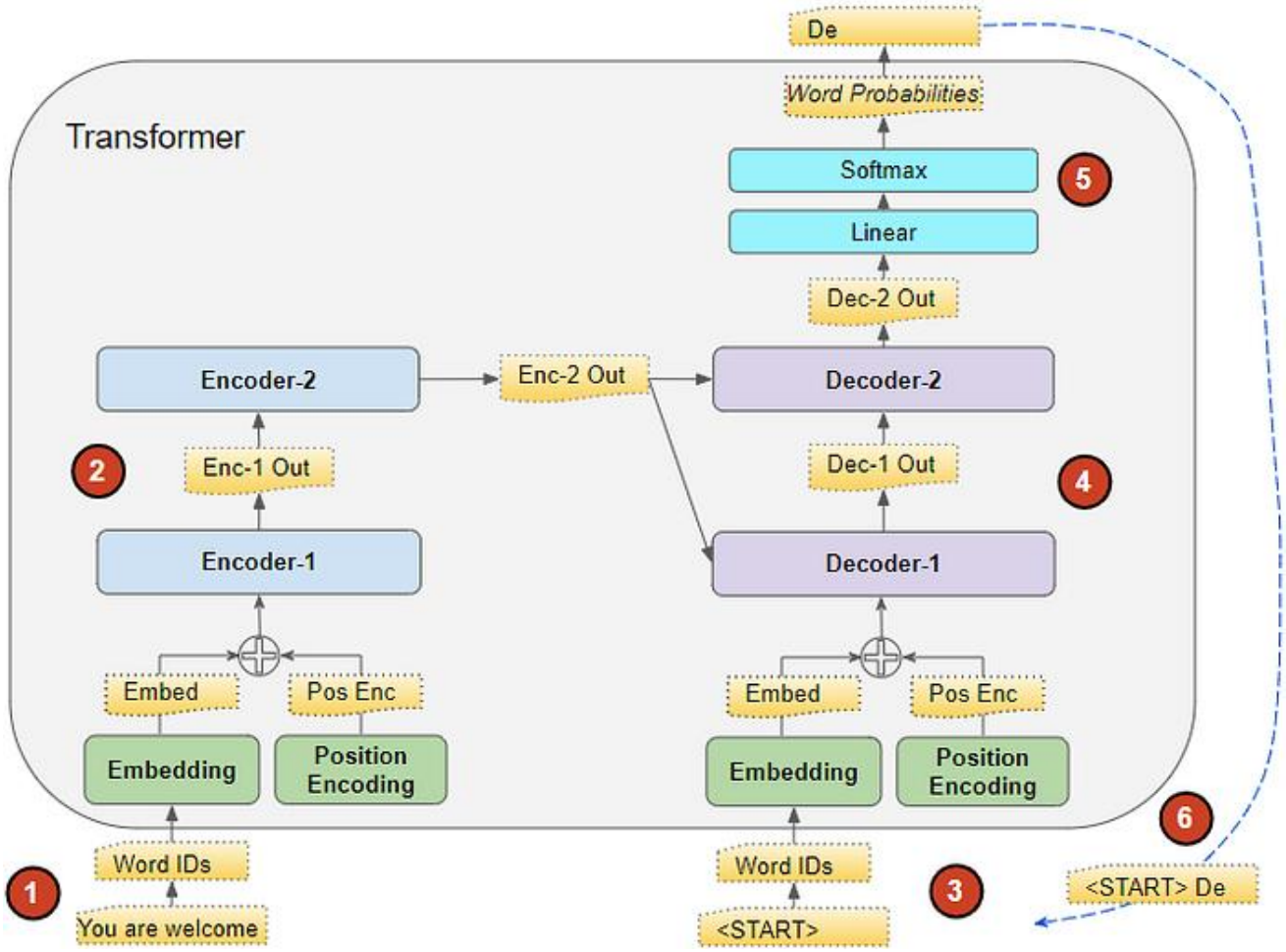
Embedding (plongement)

- Principe :
  - Chaque token est associé à un vecteur qui donne sa composition en traits sémantiques. Ce vecteur est de taille variable, en général 512.
  - Ces traits ne sont pas prédéfinis, mais obtenus par apprentissage : les tokens sont associés entre eux par des séquences issues des données d'apprentissage. Le vecteur est mis à jour à chaque fois qu'une association est constatée.
    - Intérêt :
      - Le nombre de dimension est bien plus petit que le nombre de mots du corpus. On n'a pas un espace avec une dimension par mot (chaque vecteur étant one-hot ou un parmi n).
      - Deux mots voisins auront des vecteurs proches, dont la similarité peut être calculée de manière vectorielle (cosinus par exemple). On ramène la comparaison à des projections entre vecteurs.
    - Fondement : la sémantique distributionnelle de Harris.

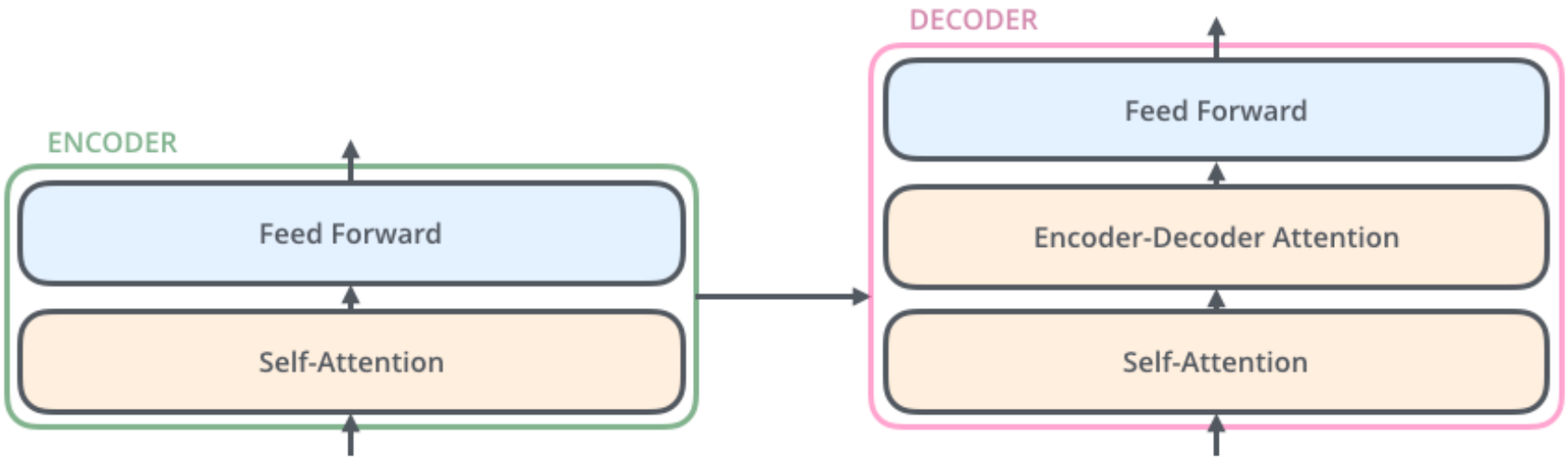
L'apprentissage



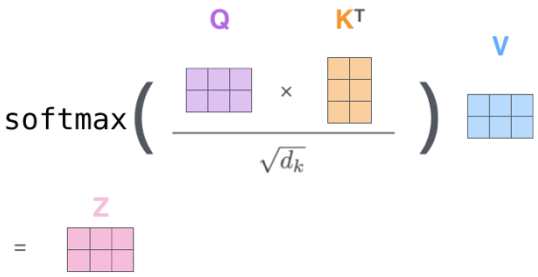
L'inférence

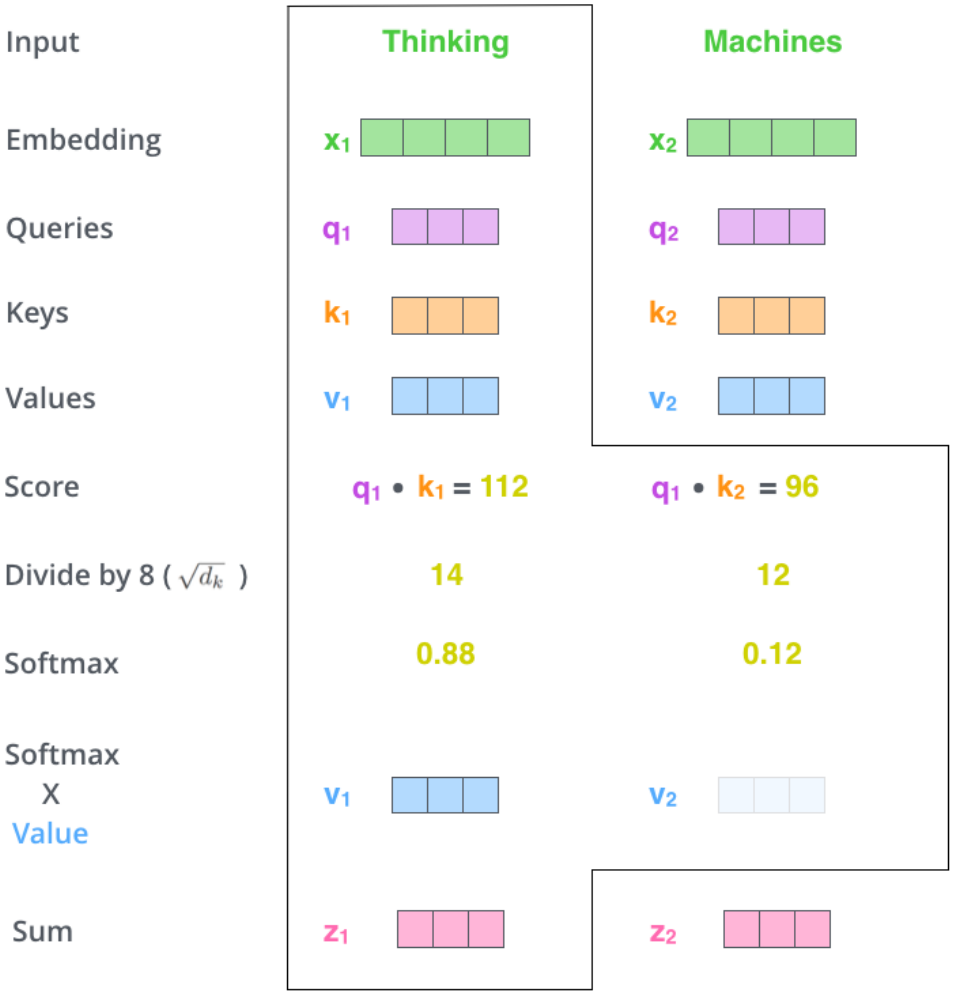

De
Word Probabilities
Transformer
Softmax
5
Linear
Dec-2 Out
Encoder-2
Enc-2 Out
Decoder-2
2
Enc-1 Out
Dec-1 Out
4
Encoder-1
Decoder-1
Embed
Pos Enc
Embed
Pos Enc
Embedding
Position Encoding
Embedding
Position Encoding
6
Word IDs
Word IDs
1
3
<START> De
You are welcome
<START>

Encoder / décoder

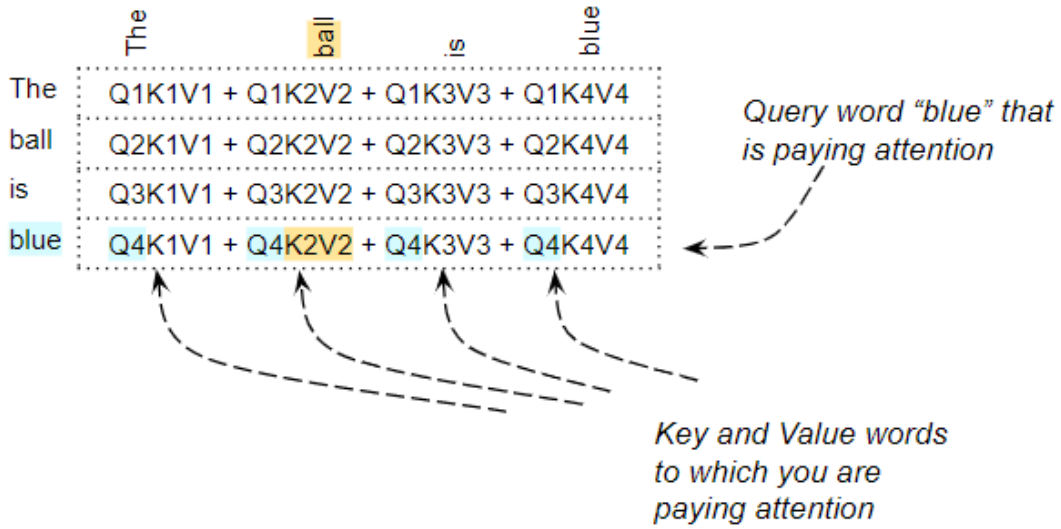


Le processus d'attention





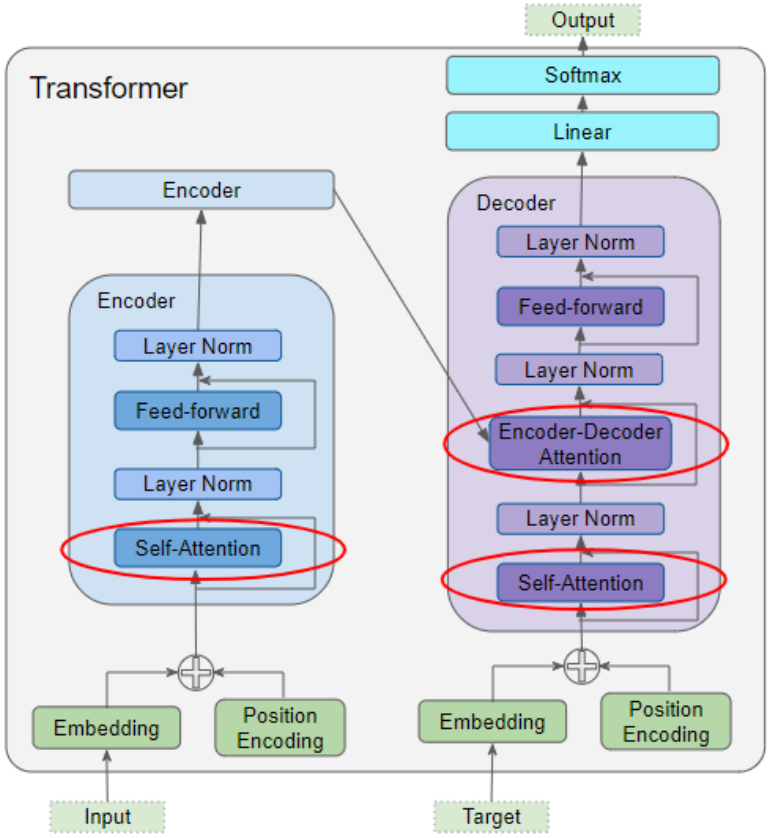
Query, Key, Value



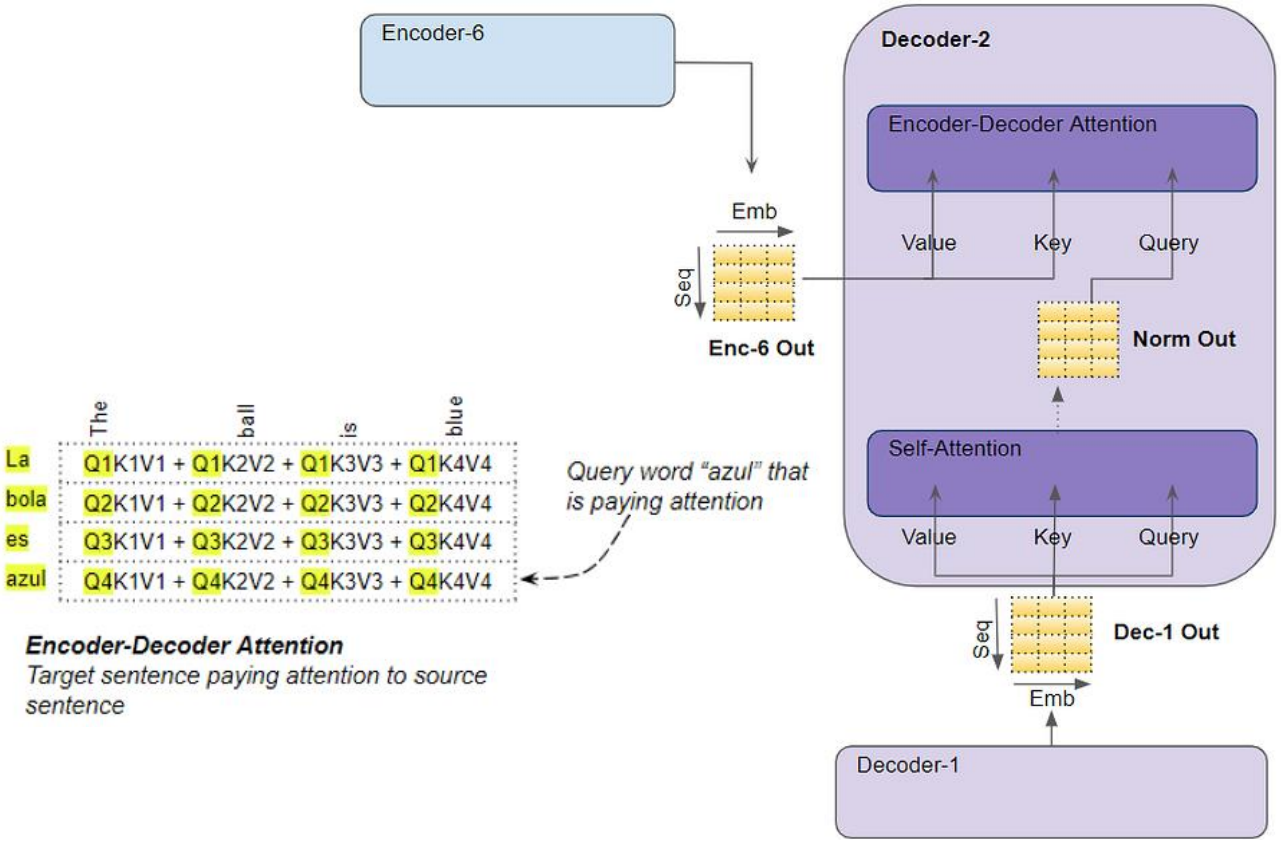
- Query :
  - De quelles informations ai-je besoin à la position où je suis ?
- Key :
  - En quoi suis-je pertinent pour les autres tokens ?
- Value :
  - la valeur d'information que le token peut apporter à ses voisins qui lui prêtent attention.

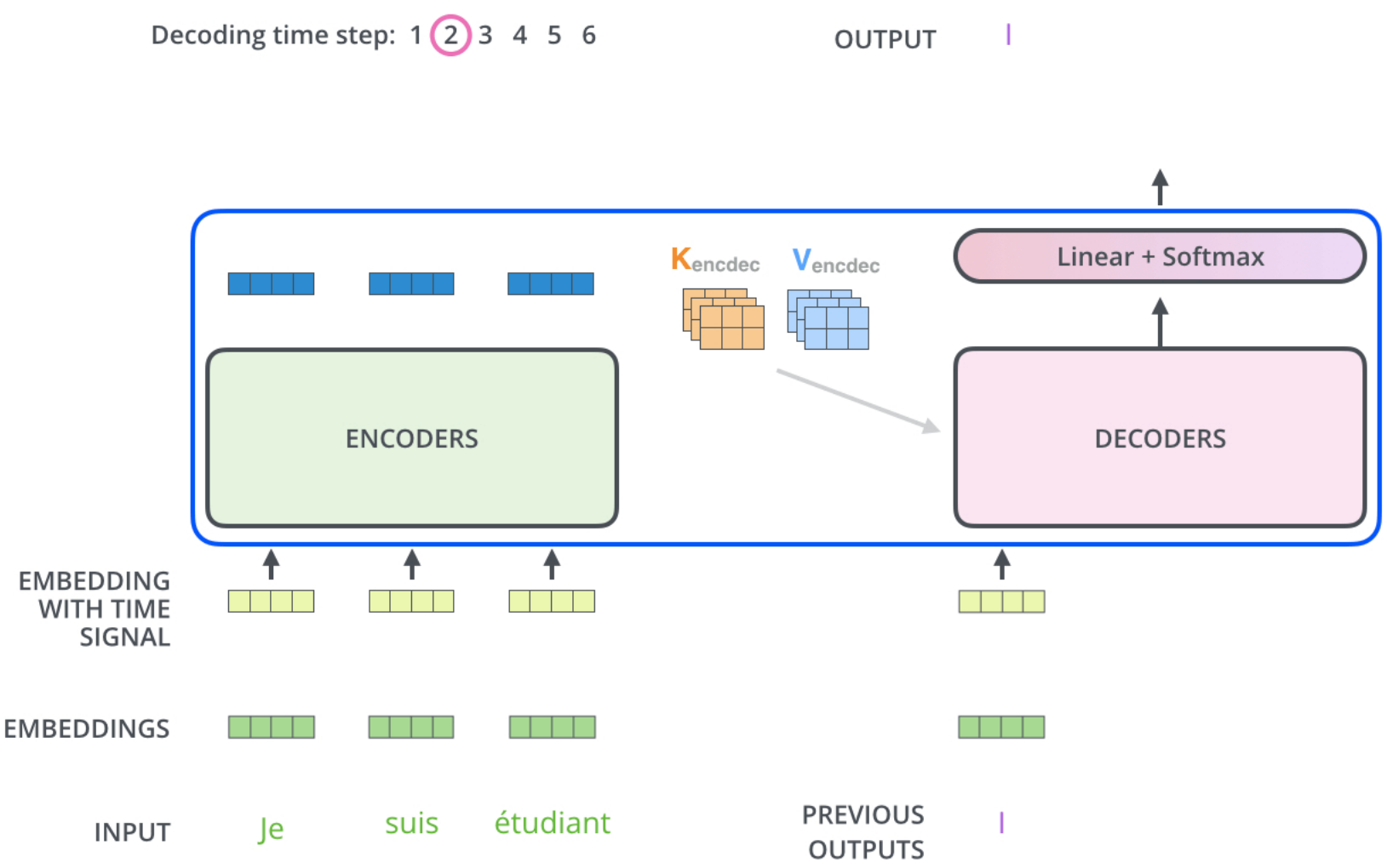
Faire attention aux autres…

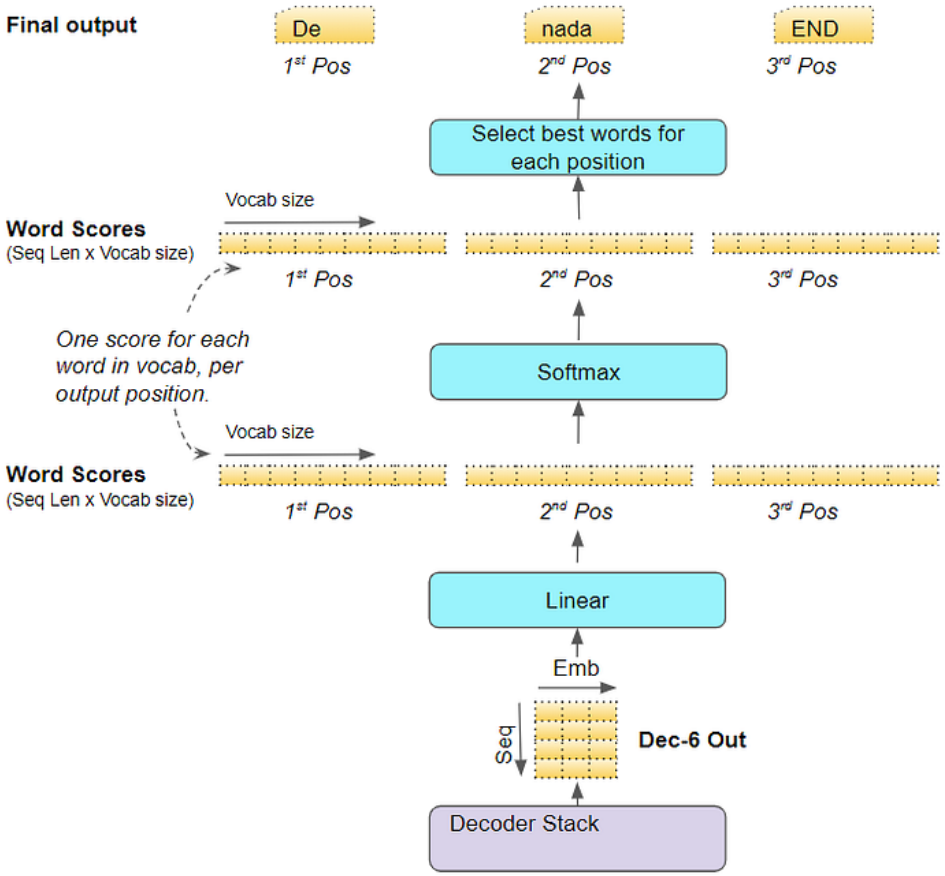
- Self-attention dans l'Encoder :
  - La séquence source fait attention à elle-même ;
- Self-attention dans le Decoder :
  - La séquence cible fait attention à elle-même ;
- Encoder-Decoder-attention dans le Decoder:
  - La séquence cible fait attention à la séquence source.



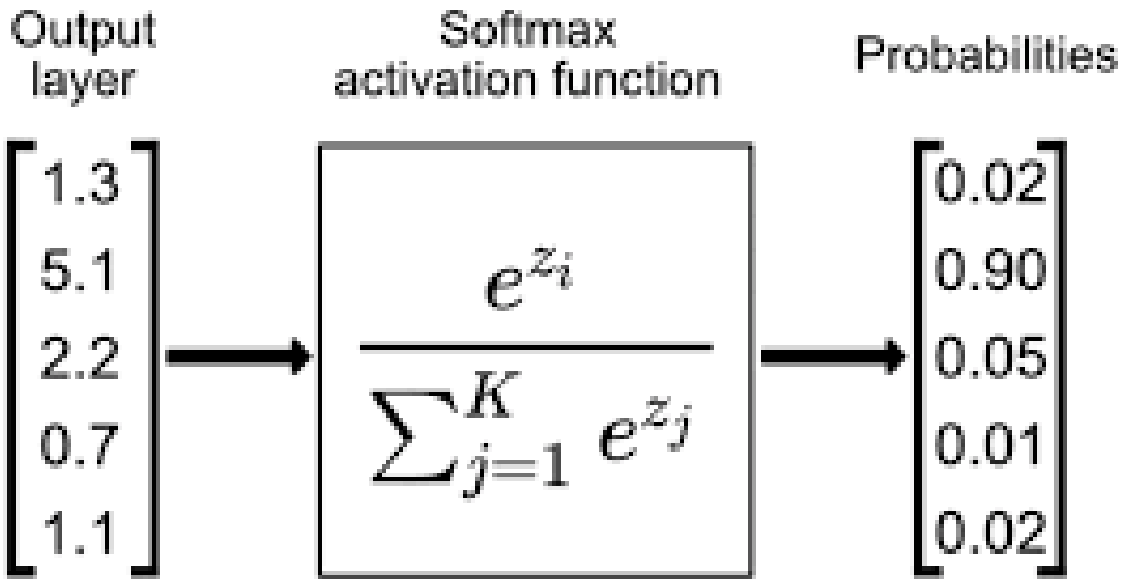
En particulier




Decoding time step: 1 2 3 4 5 6
OUTPUT
I
Linear + Softmax
Kencdec
Vencdec
ENCODERS
DECODERS
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
Je
suis
étudiant
PREVIOUS OUTPUTS
I


Final output
De
nada
END
1st Pos
2nd Pos
3rd Pos
Select best words for each position
Vocab size
Word Scores
(Seq Len x Vocab size)
1st Pos
2nd Pos
3rd Pos
One score for each word in vocab, per output position.
Softmax
Vocab size
Word Scores
(Seq Len x Vocab size)
1st Pos
2nd Pos
3rd Pos
Linear
Emb
Seq
Dec-6 Out
Decoder Stack

Le couteau suisse : softmax



Zoom sur l'apprentissage

- Soit la phrase « le chat mange la souris ».
- On tokenise : t1 t2 t3 t4 t5
- On plonge avec l'information de position : on obtient une matrice X

$$
\begin{matrix}
t11 & t12 & t13 & t14 \\
t21 & t22 & t23 & t24 \\
t31 & t32 & t33 & t34 \\
t41 & t42 & t43 & t44 \\
t51 & t52 & t53 & t54
\end{matrix}
$$

Cette matrice est de dimension T (le nombre de token considérés, c'est le nombre de lignes) fois D (le nombre de dimensions dans l'espace de plongement, c'est le nombre de colonnes).

Que veut-on apprendre avec ce vecteur X ?

- L'objectif est de savoir comment, à partir de mots connus, trouver le mot suivant.
- A partir de la phrase exemple (le chat mange la souris), on peut construire 4 exemples étiquetés :
  - « le » doit être suivi de « chat »
  - « le chat » doit être suivi de « mange »
  - « le chat mange » doit être suivi de « la »
  - « le chat mange la » doit être suivi de « souris »
- De manière générale, si on a un contexte de n mots, on a n-1 exemples que l'on peut utiliser pour entraîner le LLM.
- L'astuce des LLM qu'on va apprendre à partir de ces exemples en une fois, en traitant la matrice X de manière globale grâce à des transformations de matrices.
  - Inutile de prendre le premier exemple, puis le second, etc. ce qui peut être très long, on fait tout en une fois.
  - Comment ? C'est le mécanisme suivant.

On applique l'attention

- On prend trois matrices, W_Q, W_K, W_V, qu'on va appliquer à X pour construire trois vecteurs, Q, K, V qui vont dégager des informations à propos de X.
- La formule suivante permet d'exploiter ces matrices : Attention = SoftMax $(QK^T)V$;
  - $Q = X W_Q$, W_Q de dimension D x D, donc Q de dimension TxD
  - $K = X W_K$, W_K de dimension D x D, donc Q de dimension TxD
  - $V = X W_V$, W_V de dimension D x D, donc Q de dimension TxD
  - Q,K,V sont donc 3 vecteurs qui résultent d'une transformation de X, et sont aussi la liste des tokens avec des valeurs sur chaque dimension.
- QK^T:
  - dimension TxT, c'est une matrice qui croise chaque token avec tous les autres ;
  - Elle représente pour chaque token (ligne) l'attention qu'il a pour tout les autres.
  - Pour la ligne i qui représente le token i : les valeurs sont i1,i2,i3,i4,i5 (autant qu'il y a de tokens). C'est la matrice d'attention.
  - Pour que le token ne considère pas ce qui vient après (les tokens suivants), on met à $-\infty$ toutes les valeurs au dessus de la diagonale
  - on applique Softmax à chaque ligne : toutes les valeurs au dessus de la position diagonale valent 0.

On part de cette matrice X :

$$\begin{pmatrix} t_{11} & t_{12} & t_{13} & t_{14} \\ t_{21} & t_{22} & t_{23} & t_{24} \\ t_{31} & t_{32} & t_{33} & t_{34} \\ t_{41} & t_{42} & t_{43} & t_{44} \\ t_{51} & t_{52} & t_{53} & t_{53} \end{pmatrix}$$

Après le calcul de Q,K,V, on calcule QK^T:

$$\begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \end{array} \begin{array}{c} \begin{array}{ccccc} t_1 & t_2 & t_3 & t_4 & t_5 \end{array} \\ \begin{pmatrix} c_{11} & c_{12} & c_{13} & c_{14} & c_{15} \\ c_{21} & c_{22} & c_{23} & c_{24} & c_{25} \\ c_{31} & c_{32} & c_{33} & c_{34} & c_{35} \\ c_{41} & c_{42} & c_{43} & c_{44} & c_{45} \\ c_{51} & c_{52} & c_{53} & c_{54} & c_{55} \end{pmatrix} \end{array}$$

C’est une corrélation entre les tokens: chacun indique la valeur sa corrélation avec les autres.

Le masque causal

L'étape suivante est de mettre à $-\infty$ toutes les valeurs au dessus de la diagonale : c'est le masque causal.

$$\begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \end{array} \begin{array}{c} \begin{array}{ccccc} t_1 & t_2 & t_3 & t_4 & t_5 \end{array} \\ \begin{pmatrix} c_{11} & -\infty & -\infty & -\infty & -\infty \\ c_{21} & c_{22} & -\infty & -\infty & -\infty \\ c_{31} & c_{32} & c_{33} & -\infty & -\infty \\ c_{41} & c_{42} & c_{43} & c_{44} & -\infty \\ c_{51} & c_{52} & c_{53} & c_{54} & c_{55} \end{pmatrix} \end{array}$$

Le softmax met à 0 les cellules à $-\infty$

$$\begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \end{array} \begin{array}{c} \begin{array}{ccccc} t_1 & t_2 & t_3 & t_4 & t_5 \end{array} \\ \begin{pmatrix} p_{11} & 0 & 0 & 0 & 0 \\ p_{21} & p_{22} & 0 & 0 & 0 \\ p_{31} & p_{32} & p_{33} & 0 & 0 \\ p_{41} & p_{42} & p_{43} & p_{44} & 0 \\ p_{51} & p_{52} & p_{53} & p_{54} & p_{55} \end{pmatrix} \end{array}$$

Les p_{ij} sont des probabilités. Cela veut dire qu'un token t_i n'a aucune corrélation avec les tokens suivants. Cela implique que, quand on multiple cette matrice par V, les lignes obtenues ne prennent en compte que les lignes précédentes, jamais les suivantes.

On a donc au final $(QK^T)V$

$$\begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \end{array} \begin{array}{c} \begin{array}{ccccc} t_1 & t_2 & t_3 & t_4 & t_5 \end{array} \\ \begin{pmatrix} p_{11} & 0 & 0 & 0 & 0 \\ p_{21} & p_{22} & 0 & 0 & 0 \\ p_{31} & p_{32} & p_{33} & 0 & 0 \\ p_{41} & p_{42} & p_{43} & p_{44} & 0 \\ p_{51} & p_{52} & p_{53} & p_{54} & p_{55} \end{pmatrix} \end{array} \times \begin{pmatrix} v_{11} & v_{12} & v_{13} & v_{14} \\ v_{21} & v_{22} & v_{23} & v_{24} \\ v_{31} & v_{32} & v_{33} & v_{34} \\ v_{41} & v_{42} & v_{43} & v_{44} \\ v_{51} & v_{52} & v_{53} & v_{54} \end{pmatrix}$$

$$=$$

$$\begin{pmatrix} p_{11}v_{11} & p_{11}v_{12} & p_{11}v_{13} & p_{11}pv_{14} \\ p_{21}v_{11} + p_{21}v_{21} & p_{21}v_{12} + p_{21}v_{22} & p_{21}v_{13} + p_{21}v_{23} & p_{21}v_{1p_{21}v_{11}+p_{21}v_{21}4} + p_{21}v_{24} \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \end{pmatrix}$$

Chaque ligne ne dépend que des précédentes : la première ligne ne parle que de valeurs correspondant à t_1 (c'est le premier indice), la deuxième ligne de valeurs correspondant à t_1 et t_2, etc. On voit que l'information associée à une ligne (un token) ne prend en compte que les tokens qui précèdent.

On a multiplié par V

- on calcule $A=QK^T$ par V :
  - Dimension : TxT par TxD donne TxD,
  - on a une matrice qui a une ligne par token et qui calcule une valeur qui ne tient compte que ces valeurs qui la précédent dans la séquence.
    - N.B. $\begin{pmatrix} a_{11} & 0 \\ a_{21} & a_{22} \end{pmatrix} \text{x} \begin{pmatrix} v_{11} & v_{12} \\ v_{21} & v_{22} \end{pmatrix} = \begin{pmatrix} a_{11}v_{11} & a_{11}v_{12} \\ a_{12}v_{11} + a_{22}v_{11} & a_{21}v_{12} + a_{22}v_{22} \end{pmatrix}$
  - Ces valeurs intègre l'apport de chaque token considéré (l'apport est codé par V) en fonction de l'attention de la position du token considéré pour les autres positions (ce que calcule QK^T).
- On obtient la matrice Y.
- Y passe ensuite dans un réseau multicouche complètement connecté.
- On recommence cette séquence (Attention + couche) 80 fois. On obtient H.

On retourne aux tokens

- Une fois qu'on a H, chaque ligne est un token décomposé sur les dimensions du modèle, qui encode l'information liée à sa position.
- On veut à partir de cette ligne trouver le token correspondant qui sera celui prédit par cette ligne.
- On utilise la matrice W_{out}, de dimension DxV : le modèle de représentation (D) par le nombre de token possible (V).
- H W_{out} calcule donc pour chaque ligne les scores attribués à chacun des tokens du vocabulaire.
- Or, on sait lequel il faut trouver : sur la première ligne, il faut que le score soit nul partout sauf dans la deuxième position (t2), et de manière générale, pour la ligne i, il faut que le score de la colonne du token t_j soit à un, les autres à 0.
- On calcule les erreurs propres (entropie croisée) à chaque ligne, on fait la somme et on applique la rétropropagation du gradient pour corriger toutes les matrices.

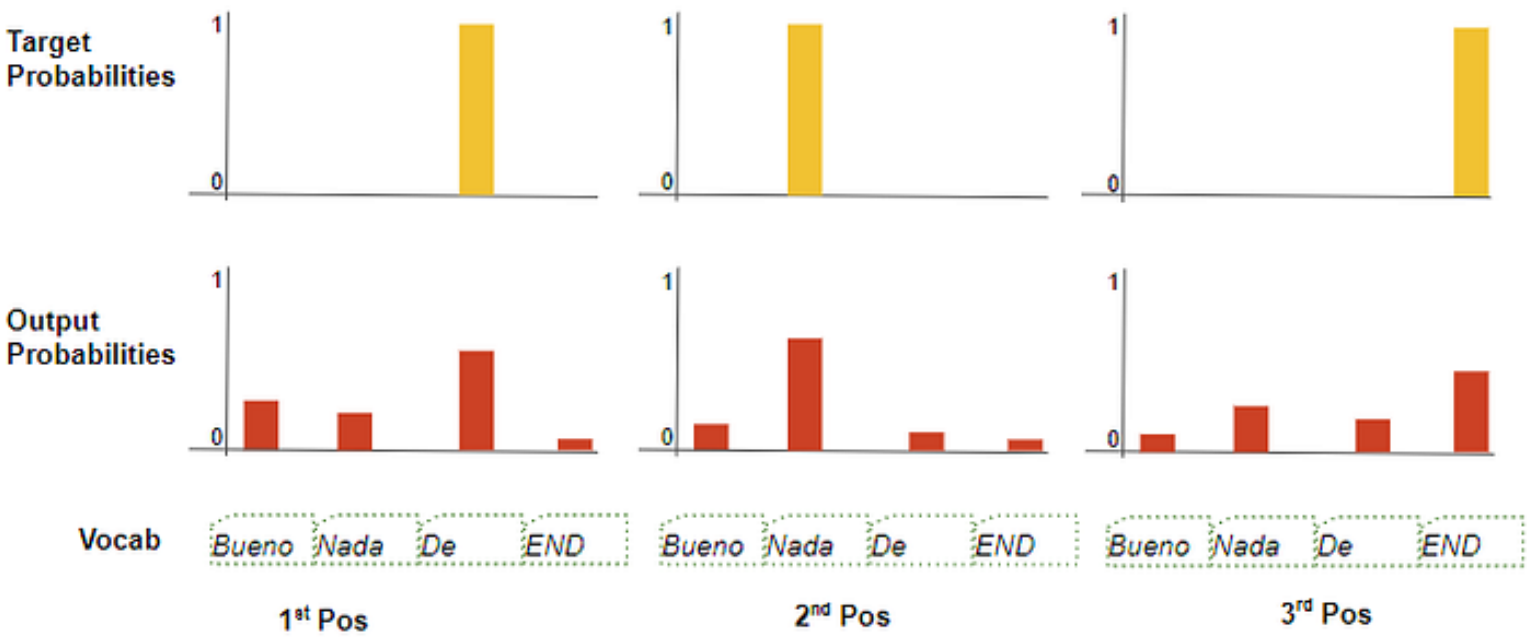
En génération / exploitaiton

- C’est la même chose, sauf que :
  - On ne calcule par les erreurs à la fin ;
  - on ne lit que la dernière ligne pour prédire le token suivant.
- Une fois ce mot généré, on recommence sur toute la séquence pour engendrer le mot suivant :
  - Le LLM met à jour à chaque mot engendré tous les contextes (attention) de chaque token du prompt + symboles générés.
    - N.B. comme les tokens ne considèrent que ceux qui les précédents, on peut stocker les calculs déjà faits : l’adjonction d’un nouveau token généré ne change pas l’attention des tokens qui précèdent car ils ne le considèrent pas.

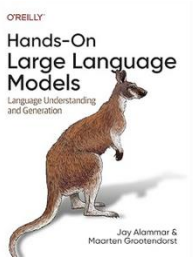
L'apprentissage

Entropie croisée entre les deux distributions de probabilités

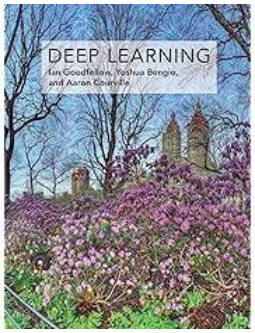
Rétropropagation du gradient pour modifier les matrices



Crédits



- Chaîne Youtube : ScienceEtonnante, « comment l'IA invente-t-elle des images ? », David Louapre
- Toward data Science :
  - Ketan Doshi : transformers explained visually ;
- Blog de Jay Alammar (https://jalammar.github.io/illustrated-transformer/)
- La bible :



IAG : quelques tensions

Trois niveaux : les 3D

Le dire

Le chat est sur le tapis

Le dit

Le désigné

Remarques

Le dire est un faire

Le dit est une chose

Le désigné appartient au monde des choses et du faire.

Des tensions

Dit et désigné

- Quel rapport entre les choses qui sont des mots/dits et les choses qui n'en sont pas ?
- Les dits sont des choses qui possèdent des relations entre elles de succession (syntagmatique) et de juxtaposition (paradigmatique):
  - On comprend comment un mot en entraîne un autre, un dit un autre dit.
  - On peut même considérer que ces successions sont la seule réalité du dit, en une sémiose illimitée ou infinie (Peirce), qui ne s'épuise pas dans un objet du monde mais une action (le faire). La référence au désigné est rapporté à l'inférence vers un objet qui devient un signe (tout est signe).

Dit et dire

- Tout dit émane d'un dire ;
  - Le dit renvoie à une intention appelant une interprétation : il ne se réduit pas à une simple chose que l'on peut décontextualiser du faire qui l'a produite / énoncée et du faire qui l'interprète / reçoit ?
- mais un dit est aussi une chose, donc productible par des faire.
  - Quel rapport entre le faire qu'est un dire et les faire dont le monde du désigné se constitue ?
  - Est-ce que tout dit résultant d'un faire quelconque serait analogue à un dit relevant d'un dire ?
  - Est-ce que tout faire produisant un dit serait analogue à un dire ?

Les glissements du dit

L'IAG travaille sur les dits :

- Elle porte sur des choses (les dits) ;
- Elle procède d'un faire (calcul sur les dits)

De ce fait :

- L'IA apparaît émaner d'un dire : le faire du calcul est assimilé au faire qu'est le dire ;
  - **Le problème de l'énonciation :** ça parle !
- L'IA apparaît porter sur les choses : la chose qu'est le dit est assimilée à la chose que le dit désigne.
  - **Le problème de la modélisation :** c'est porteur de connaissance

Modélisation

- Un modèle :
  - Artefact que l'on peut expérimenter en lieu et place du monde réel ;
  - Le modèle est un modèle du réel dans la mesure où il répond aussi bien que lui ; un modèle utilisé hors de son champ de validité n'est plus un modèle mais une fiction.
    - Cf. Modèle réduit, modèle numérique, modèle de langage, modèle de données, etc.
- Le modèle n'est pas une théorie, mais en résulte souvent pour substituer au monde un artefact reproduisant les comportements prévus par la théorie.
  - Tension entre le modèle comme objet réel et le modèle comme substitution à un objet réel.

IA/G et modèle

- En quoi l'IA se comporte-t-elle comme un modèle ou en produit-elle ?
  - On utilise les outils entraînés de l'IA comme des modèles : on les interroge à la place du réel lui-même, pour le contrôler, l'anticiper, le deviner.
    - L'IA, et l'IAG sont donc bien utilisés comme des modèles.
- Mais se comportent-ils comme des modèles ?
- Les IA/G sont des synthèses de dits :
  - Utilisés comme s'ils pensaient à ce qu'ils disent (cf Platon dans Phèdre)
  - Mais difficiles de contrôler leur domaine de validité car il faut pour cela :
    - Contrôler les données : savoir déterminer le périmètre correspondant aux corpus / données d'apprentissage ;
      - Le problème est que le corpus des IA/G est indéfini, voire infini ; l'apprentissage que l'on fait n'est pas meilleur que le pire des algorithmes (théorème no free lunch) di on ne spécialise pas les tâches.
    - Contrôler l'apprentissage : contrôler la véracité de la réponse vis-à-vis de celles des données ;
    - Contrôler le périmètre d'usage : s'assurer que la question posée possède une réponse dans le périmètre défini par les données.

Énonciation

- Point de vue « phénoménologique »
  - Acte de dire, de proférer un énoncé engageant donc un faire dans le monde, une action.
  - L'énonciation n'est pas le fait d'une structure idéale formelle s'imposant à des objets et leur conférant une sémiotisation, mais un faire s'engageant dans un monde qu'il modifie (performativité).
- Point de vue « structuraliste »
  - Énonciation comme adresse, comme médiation langue/ parole, comme praxis (©Dondero)
  - Positions dans un espace de valeur, chaque position ayant sa valeur par ses différences vis-à-vis des autres positions (à la Saussure).
  - L'énonciation est donc l'agencement de positions, l'articulation structurelle d'entités qui n'existent ou ne sont qu'à travers ces positions et valeurs positionnelles.
    - Les positions calculatoires déclenchent les instructions (adresse), engendrent leur propre temps (langue / parole), sont issues de pratiques sédimentées (praxis).

IAG et énonciation

- Point de vue « phénoménologique » :
  - Ce ne sont pas des personnes, des « qui », mais désormais des « quoi » qui nous répondent ;
  - Suites de caractères, de phonèmes ou de sons qui sont reconnus comme porteurs de sens mais non produits en vue du sens reconnus, mais seulement en fonction de leurs corrélations.
  - Par construction, un énoncé, pour être reçu comme tel, doit pouvoir être imputé à un qui (fait phénoménologique ou illusion grammaticale)
- Énonciation supposée avec l'IAG alors qu'il n'y en a pas.

- Point de vue « structuraliste » :
  - La machine agence des tokens :
    - Ces tokens sont issus de contenus analysés et corrélés (embedding) dans un espace de positions, ces dernières reflétant la proximité distributionnelle / sémantique ;
    - Reflète la praxis des énoncés
  - La machine exploite ces positions comme commande de production de nouveaux contenus
    - Reflète l'adresse et la commande dans laquelle les utilisateurs sont intégrés et mobilisés ;
  - La machine crée son propre temps d'exécution, l'espace de calcul entraînant les « pas » de calcul :
    - Reflète comment le système engendre la performance.
- Énonciation effectuée par l'IAG

Contexte et machine

L'outil, la machine, l'instrument sont des clôtures opérationnelles permettant la répétition et la production anticipée du résultat.

La machine

- N'a pas d'ici et d'ailleurs ;
- N'a qu'un ici (les données) et un néant : ce qui n'est pas donné, une donnée, n'existe pas :
  - Données de capteur (le là-bas comme un ici)
  - Données de situation (ailleurs spatial – pas ici - et/ou ailleurs temporel – passé/futur)
  - Données sociales (autrui représenté comme modalité ou données)

Conclusion

L'IA est un outil de synthèse de contenus,

- Comme on a des diamants synthétiques ou des molécules de synthèse ;
- Comme outil permettant de synthétiser des contenus divers fournis en données ;

L'IA produit donc des contenus sans lien avec un dire ni avec un désigné, qui restent à la charge interprétative de l'utilisateur qui doit, s'il le peut :

- Le rapporter à un dire possible ;
- Le vérifier vis-à-vis d'un désigné, soit par une confrontation au réel, soit par une dépendance à des données elles-mêmes déjà confrontées et validées
  - (pb de l'explication et de la justification).

Conclusion

L'IAG pose alors le problème de la décision et de la responsabilité :

- Décider quant au désigné, sachant que l'IAG n'en dit rien ;
- Assumer quant au dire, sachant que l'IAG ne dit rien ;

L'IAG ne peut donc fonctionner que comme suppléance :

- dans des domaines où l'on peut avoir d'autres moyens qu'elle pour décider et assumer ses résultats.