Institut des Actuaires

11 janvier 2024

Les grands modèles de langage,

une expédition dans le moteur

olivier.sorba@randompulse.net

- Panélistes:
  - Florence Picard
  - Marc Juillard
  - Michael Donio
  - Nicolas Marescaux

Réseaux de neurones et apprentissage

Echauffement : bataille navale

- Faire une carte avec une caméra à un seul pixel,
- et une mémoire de moineau

Gradient Stochastique

- C’est le principe du gradient stochastique
- L’idée de la descente de gradient remonte à Augustin-Louis Cauchy, en 1847 puis Jacques Hadamard en 1907
- Permet de traiter une quantité illimitée de données sans les charger toutes en mémoire.
- La descente de gradient stochastique prend en compte la présence de bruit dans les données.
- Les gradients sont une représentation numérique de la direction de plus grande pente.

Une video d’Andrew Ng

Linear regression with one variable

Gradient descent

- https://youtu.be/yFPLyDwVifc?si=pf66zSqTd8Yga67y

En résumé

- Dans quel espace fait-on cette recherche ?
  - Dans l'espace des paramètres (réglages possibles) d'un modèle que nous entrainons.
- A quoi correspond l'altitude ?
  - Idéalement à une mesure erreur moyenne sur l'ensemble des données
  - Mais nous n'avons qu'un échantillon : on l'utilise comme représentant de la distribution inconnue, pour produire un taux d'erreur empirique
  - Mais nous ne pouvons pas traiter tout l'échantillon en une seule fois, et il est entaché de bruit: nous utilisons chaque élément de l'échantillon comme un représentant de l'ensemble, séquentiellement: gradient stochastique.

Risques:

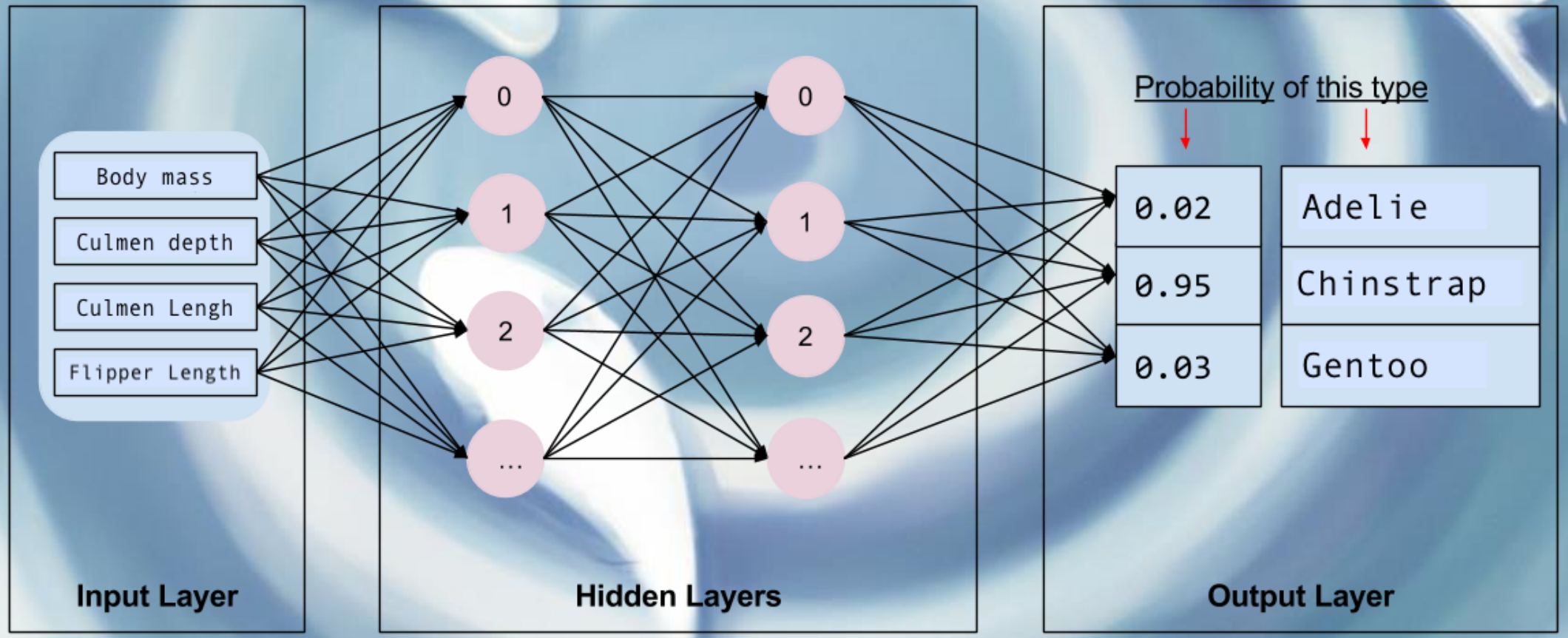
- Tourner en rond sans jamais converger
  - Solution: algorithmes adaptatifs
- Sur-apprendre l'échantillon avec son bruit
  - Régularisation, early-stopping basé sur un échantillon de test distinct.
- Mais aussi par la suite (pour mémoire)
  - La distribution des données change (j'utilise en Bretagne mon modèle de la côte basque)
  - La réalité a changé: les bancs de sable ont bougé
  - Quelqu'un a changé le format des coordonnées GPS sans prévenir le Chief Data Officer
  - Et cetera

De la régression linéaire aux réseaux de neurones (en sautant le machine learning)

- Les éléments d'un réseau de neurone:
- Prendre des inputs (ici les coordonnées GPS)
- Les combiner avec les paramètres du réseau (les poids)
- Pour obtenir un résultat, par exemple une prédiction (« mer à 60% ») ou une génération de contenu.
- Par un enchaînement de calculs
  - En général sans boucle (« réseaux feed forward »)
- Les règles de calcul et d'enchaînement forment la structure du réseau
  - Elle est en général fixe
- Chaque étape de calcul prend comme intrant tout ou partie des résultats des précédentes
  - Elle est l'analogue lointain d'un neurone biologique (sans les boucles de rétroaction)
  - Le chiffre produit est l'équivalent de l'activation neuronale.

- Une couche (layer) représente un ensemble de neurones partageant la même structure de calcul, souvent mis à jour en bloc.
  - L'input et l'output sont des ensembles de chiffres disposés en vecteur, matrice, matrice de matrices, etc… : tenseurs.

Un classifieur de pingouins



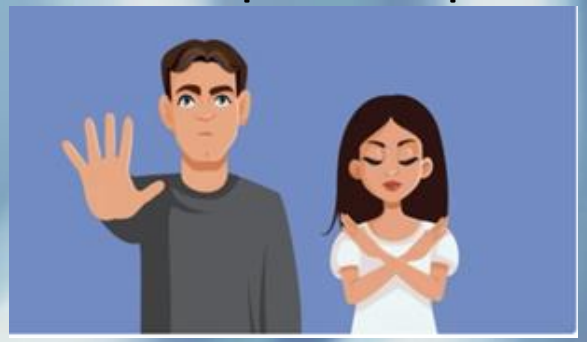
Source: Tensorflow

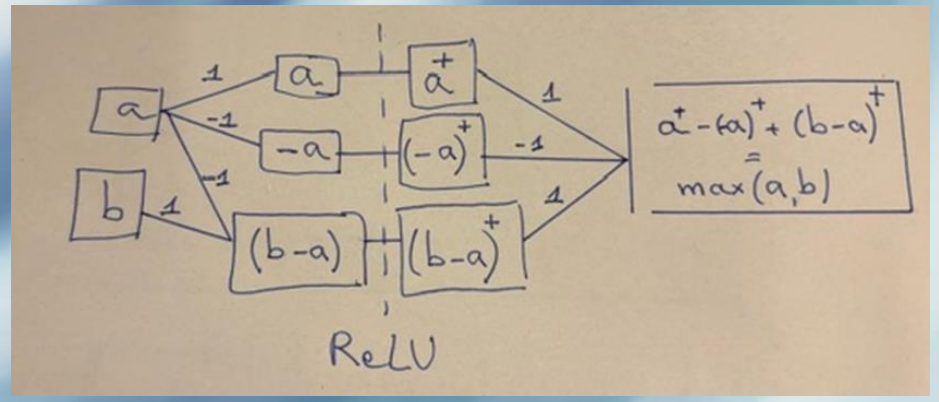
Un exemple animé

- retour à la régression linéaire
- importance des non-linéarités

Quiz:

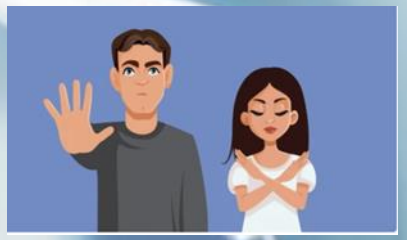
- Comment calculer le maximum de deux nombres
  - avec seulement des opérations linéaires: multiplications par des paramètres fixes et additions ?
  - Et avec une non-linéarité ?
    - $(a, b) \rightarrow a + (b - a)^+$
  - Et avec une non-linéarités à chaque étape ?
    - $(a, b) \rightarrow a^+ - (-a)^+ + (b - a)^+$
    - $(3,7) \rightarrow 3 - 0 + 4 = 7$
- Rapprocher du AND et du OR logique

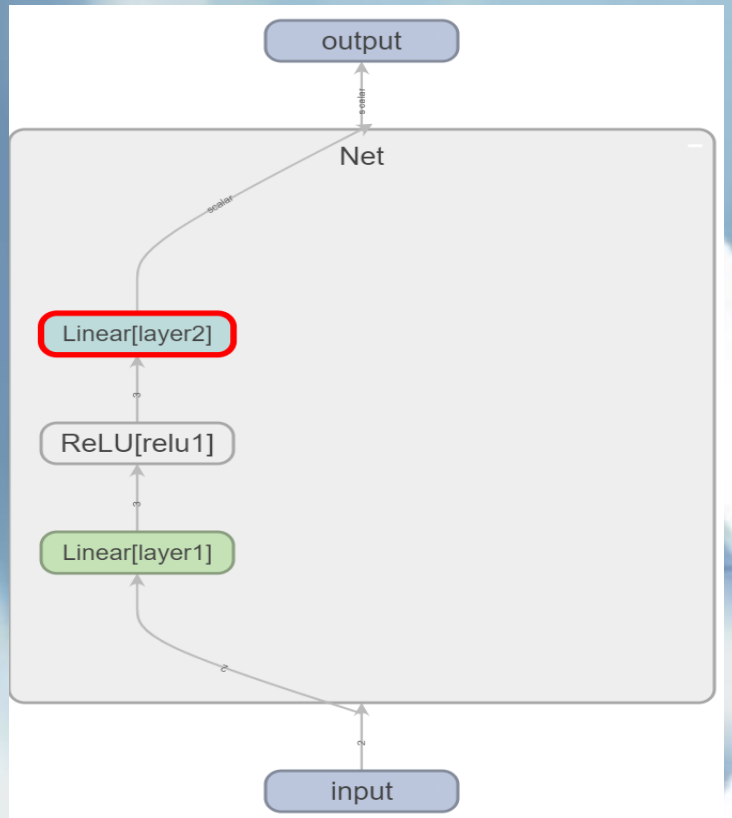


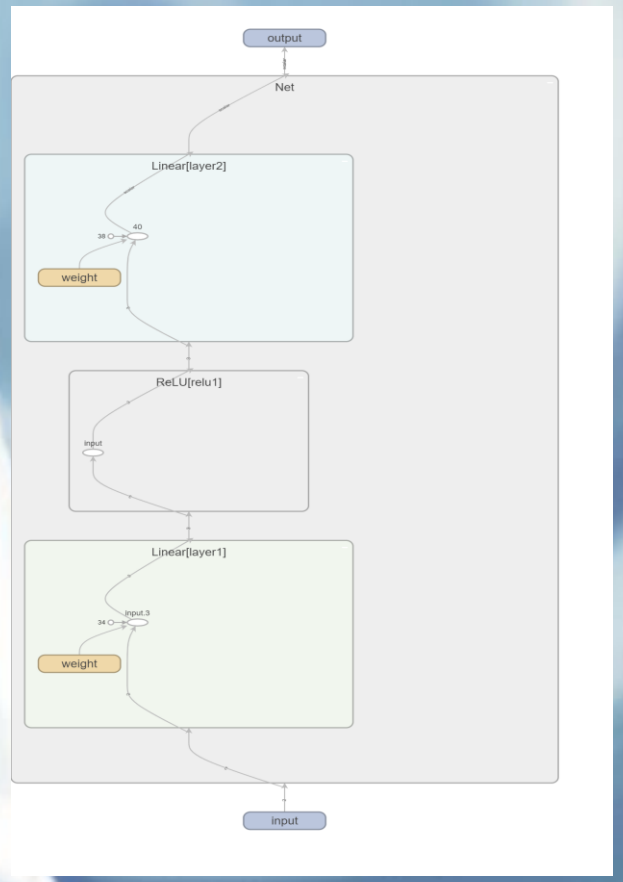


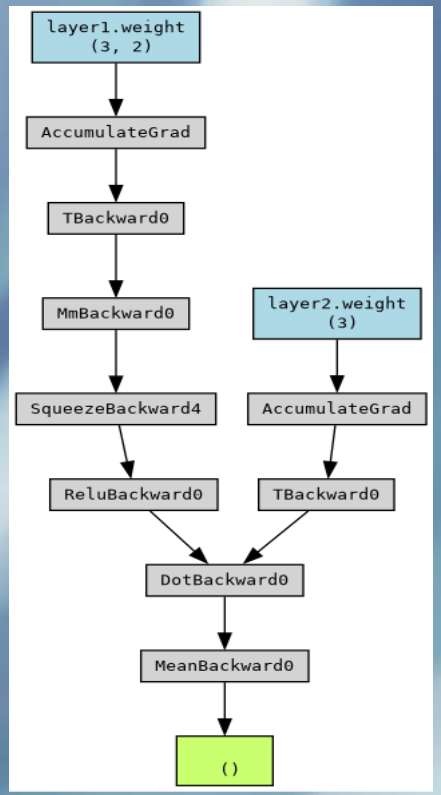
Quiz, vu par Pytorch

```
my_net=nn.Sequential()
my_net.add_module('layer1', nn.Linear(2, hidden_size, bias=False))
my_net.add_module('relu1', nn.ReLU())
my_net.add_module('layer2', nn.Linear(hidden_size, 1, bias=False))
```



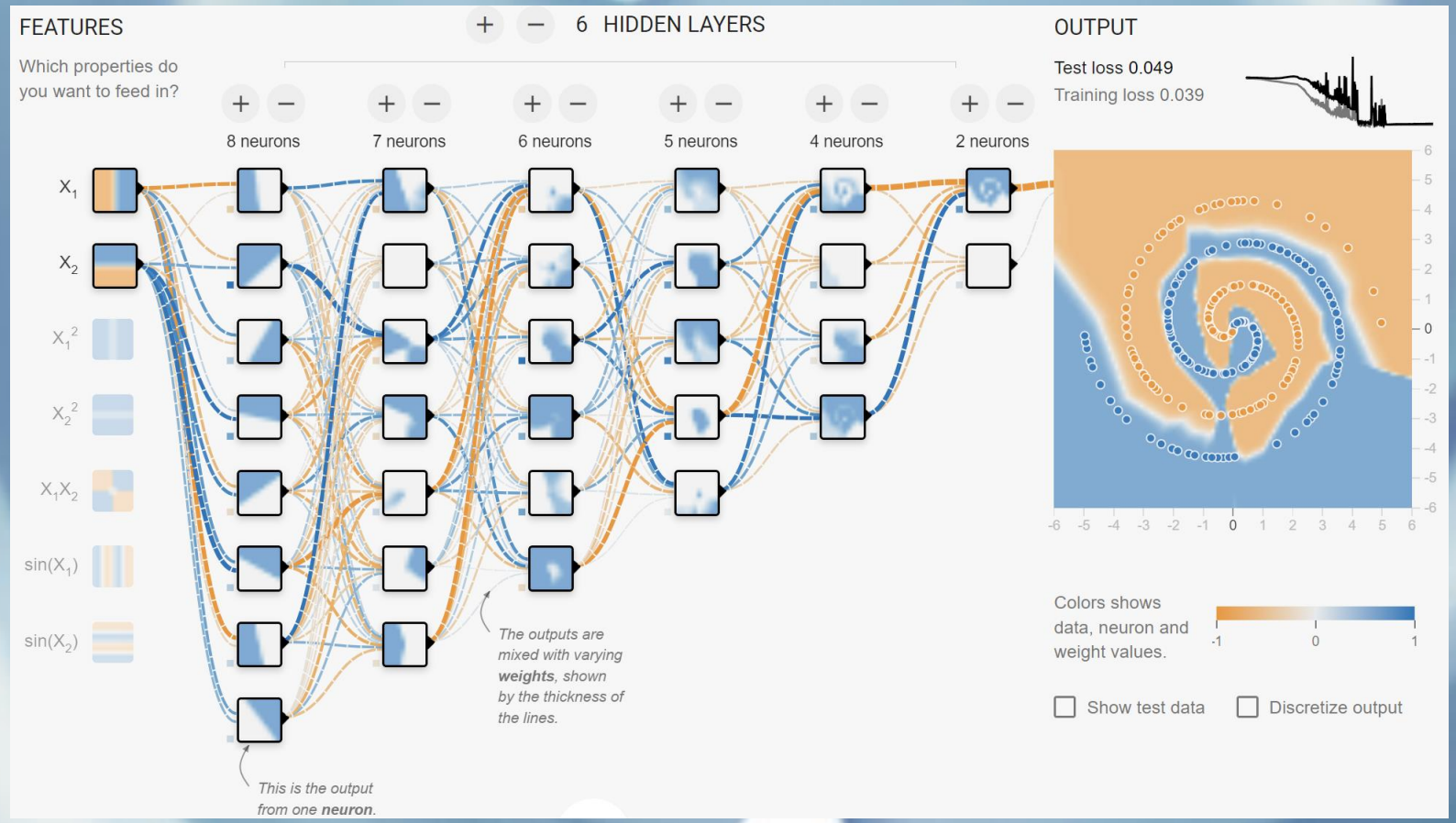






Non-linéarités

- [Un exemple plus complexe]



L’espace latent

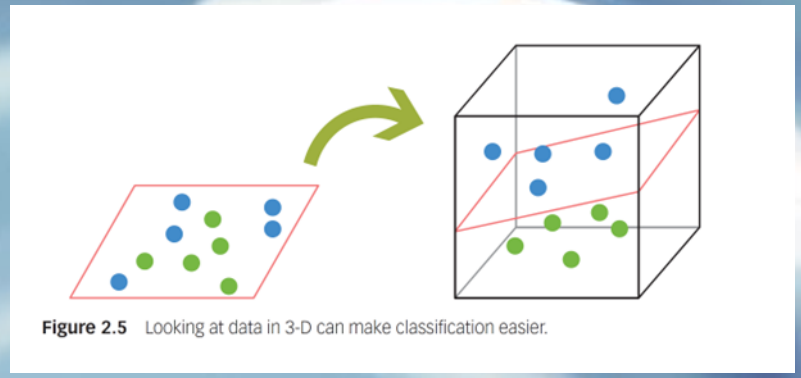
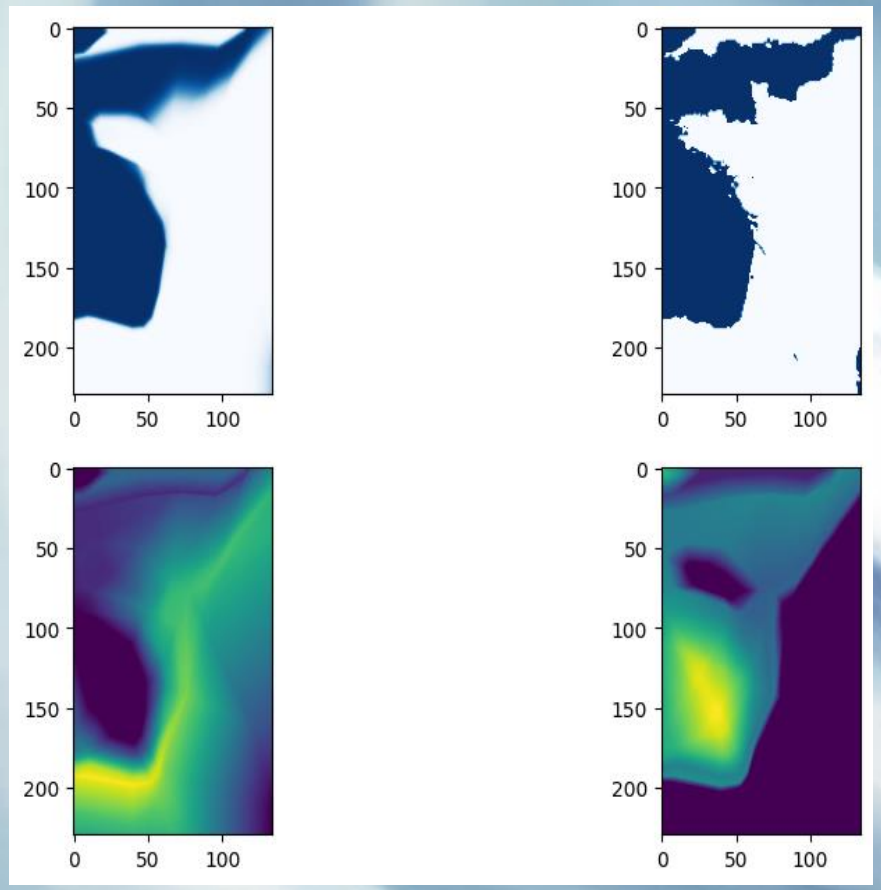


Figure 2.5 Looking at data in 3-D can make classification easier.



Traiter le texte

- Quelques problèmes :
  - Vocabulaire infini
  - Longueur des séquences d'entrée
  - Les réseaux de neurones ne peuvent traiter que des nombres
  - Disponibilité d'échantillons d'entraînement
    - textes utiles
    - non-toxiques

Vocabulaire infini : Tokenisation

- Inspiré des méthodes de compression (zip)

supermédiateur, doublevédoublevé-doublevé, auto-diagnostiqués, néo-célibataires, sur-monétisation
agroécologiste, multiactivité, auto-obscurcissant, neo-retraité, macrostabilité
e-détournements, partenadver-saires, hollandisme, retricoté, agnélise
pagano-satanisme, auto-diagnostiqués, neo-retraité, agroécologiste, e-détournements

Source inria.hal.science/hal-00959079

```
--------------------------------------------------------------------------------
sentence:       "Once upon a time there was a Horrific Gryphon. Not unnusual !"
token ids:      [7454, 2402, 257, 640, 612, 373, 257, 6075, 81, 811, 33958, 746, 261, 13, 1892, 41231, 37850, 5145]
tokens:         [Once, Ġupon, Ġa, Ġtime, Ġthere, Ġwas, Ġa, ĠHor, r, ific, ĠGry, ph, on, ., ĠNot, Ġunn, usual, Ġ!]
--------------------------------------------------------------------------------
sentence:       "Once upon a time there was a horrific gryphon. Not unusual !"
token ids:      [7454, 2402, 257, 640, 612, 373, 257, 19447, 308, 563, 746, 261, 13, 1892, 8468, 5145]
tokens:         [Once, Ġupon, Ġa, Ġtime, Ġthere, Ġwas, Ġa, Ġhorrific, Ġg, ry, ph, on, ., ĠNot, Ġunusual, Ġ!]
```

Des mots aux vecteurs : Embeddings

- représenter les mots par des vecteurs
- Exemple avec des vecteurs tirés au hasard:

```
--------------------
Angle with Lily+cat+play:
Lily  :                 cosine: 0.54    degres: 57.04
cat   :                 cosine: 0.50    degres: 60.24
play  :                 cosine: 0.53    degres: 57.71
sister:                 cosine: -0.04   degres: 92.39
ball  :                 cosine: -0.05   degres: 92.85
house :                 cosine: 0.02    degres: 88.89
Lily+cat+play:          cosine: 1.00    degres: 0.00
--------------------
Embeddings:
Lily            -1.23, -1.23, 0.77, -0.25, -0.90, -1.12, 0.45, -0.03, 1.31, 0.09, 2.14, -0.09, -1.68, 0.43, -0.30 ...
cat             1.17, -0.02, -0.94, -0.61, -1.10, -1.46, 1.31, -0.53, -0.70, -1.59, 0.01, 0.35, -1.62, -0.33, 0.53 ...
play            -1.36, 1.09, -0.85, -0.03, -1.20, 0.11, -0.33, -0.84, -0.19, 0.95, -1.89, 2.19, 2.11, -1.23, 0.89 ...
sister          0.88, 0.19, 0.48, -1.90, -1.74, 1.38, -0.71, -0.37, -0.19, -0.76, -2.03, -1.10, -1.89, -1.66, -1.35 ...
ball            -0.56, -1.57, 0.03, -0.23, -0.82, 0.08, -1.13, 0.42, -1.24, -0.50, -0.30, -0.54, 0.19, 1.32, -0.83 ...
house           0.66, 0.43, 0.22, -0.94, -0.15, -0.27, -0.04, -0.55, -1.33, 0.28, -1.18, 0.26, -2.09, 0.12, 0.96 ...
Lily+cat+play   -1.42, -0.17, -1.02, -0.89, -3.19, -2.47, 1.42, -1.39, 0.42, -0.56, 0.26, 2.45, -1.19, -1.13, 1.12 ...
--------------------
```

```python
import numpy as np
from numpy.linalg import norm

names = ['Lily  ', 'cat   ', 'play  ', 'sister', 'ball  ', 'house ', 'Lily+cat+play']
#
embeddings = np.random.normal(size=(len(names),252))
#
combined = embeddings[0]+embeddings[1]+embeddings[2]
embeddings[-1] = combined
for n, w in enumerate(names):
    cosine = np.dot(combined, embeddings[n])/norm(embeddings[n])/norm(combined)
    theta = np.arccos(cosine)/np.pi*180
    print(f"{w}: \t\tcosine: {cosine:.2f} \tdegres: {theta:.2f}")
print("-"*20)
print("Embeddings:")
for n, w in enumerate(names):
    print("{0:10}".format(w), ', '.join(f'{q:.2f}' for q in embeddings[n][:15])+' ...')
```

Embeddings

- L'information de plusieurs vecteurs se fusionne par addition
  - Pourquoi sans trop de perte ? Les vecteurs tendent à être très éloignés les uns des autres dans un espace de grande dimension. Peu de collisions.
  - *Si on a le temps : dénombrer les coins d'un hyper cube.*
    - *d=1 ->2*
    - *d=2 ->4*
    - *d=3 ->8*
    - *d=10 -> 1024*

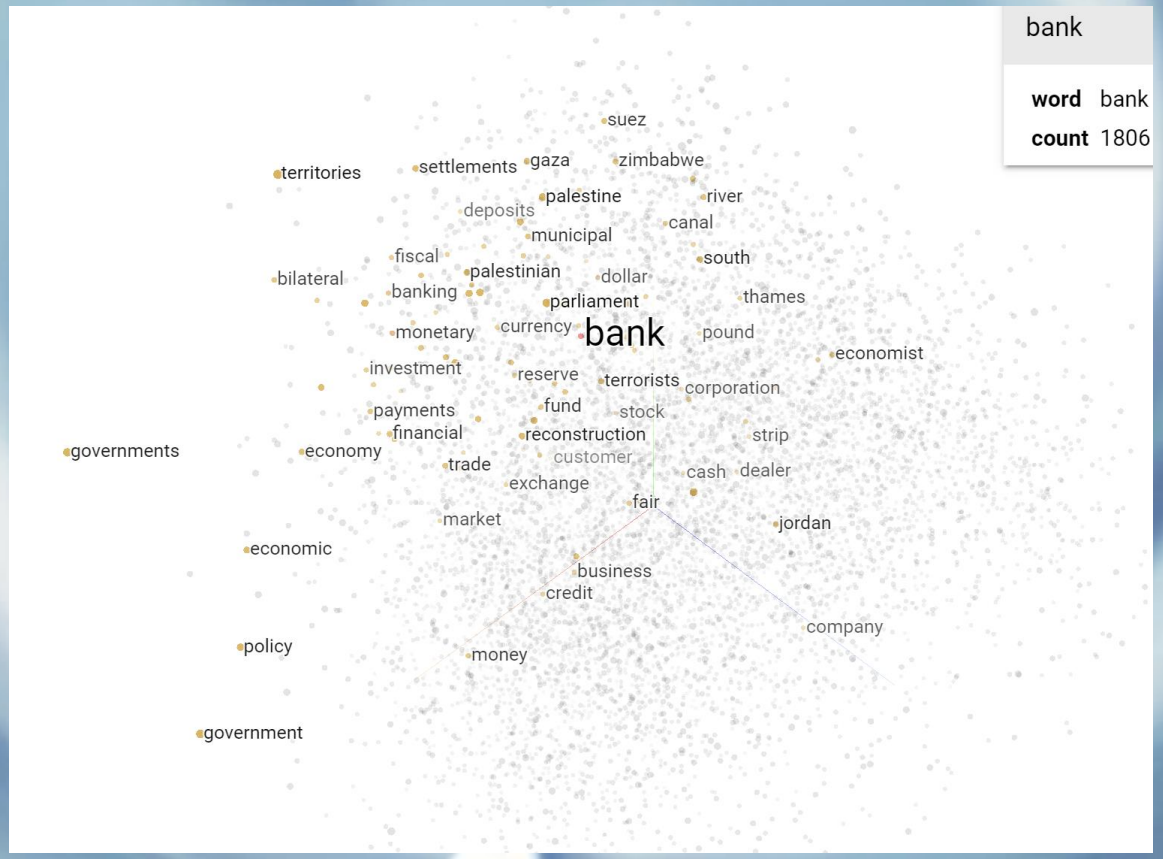
- La proximité se mesure par l'angle
  - Pourquoi l'angle et pas aussi la taille ? En grande dimension, la taille (la norme) n'est qu'un paramètre parmi beaucoup. On ne perd pas beaucoup d'information à l'ignorer

Embeddings

- En pratique, on construit un embedding en entrainant un réseau à effectuer une tache sur des mots, par exemple deviner un mot masqué à partir de ses voisins. L'embedding apparait souvent comme première couche d'un réseau:
- [word seq]->[word id seq]->{[embedding vector seq]->layers}->output

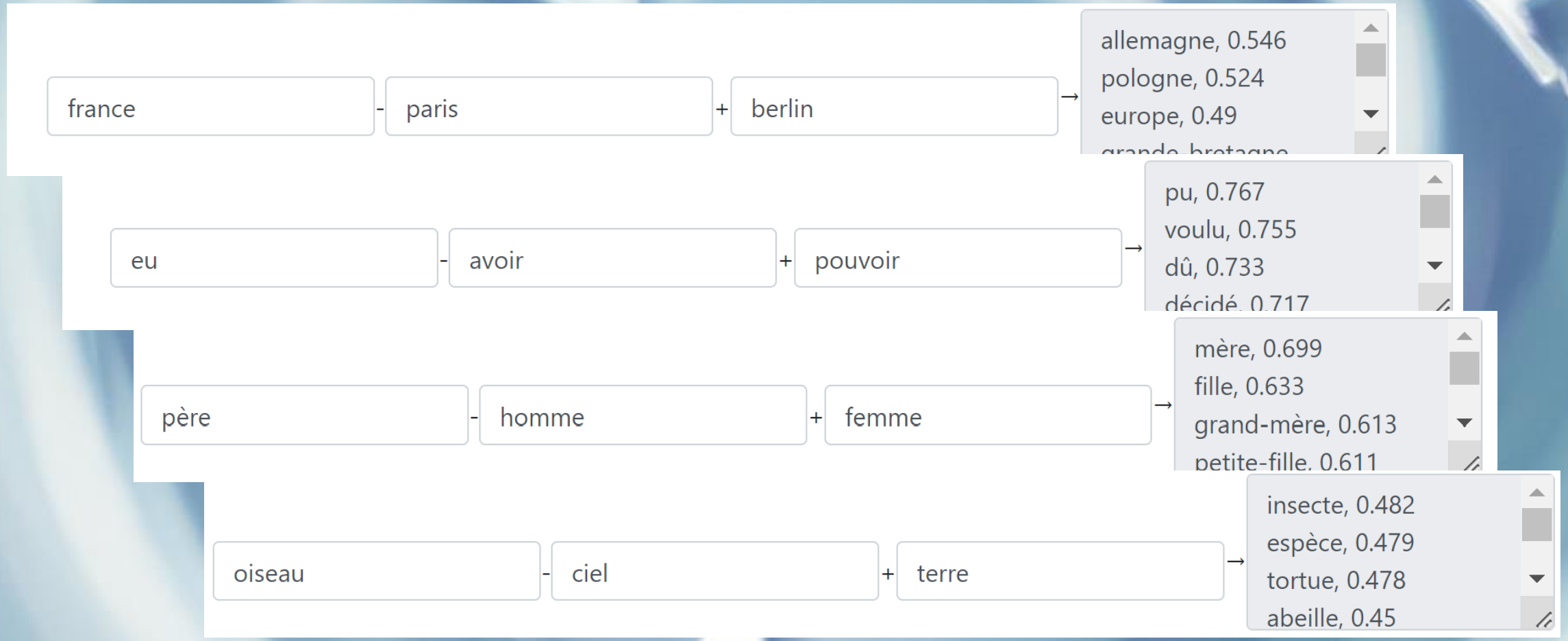
Embeddings

- captent la proximité entre les mots
  - https://projector.tensorflow.org/



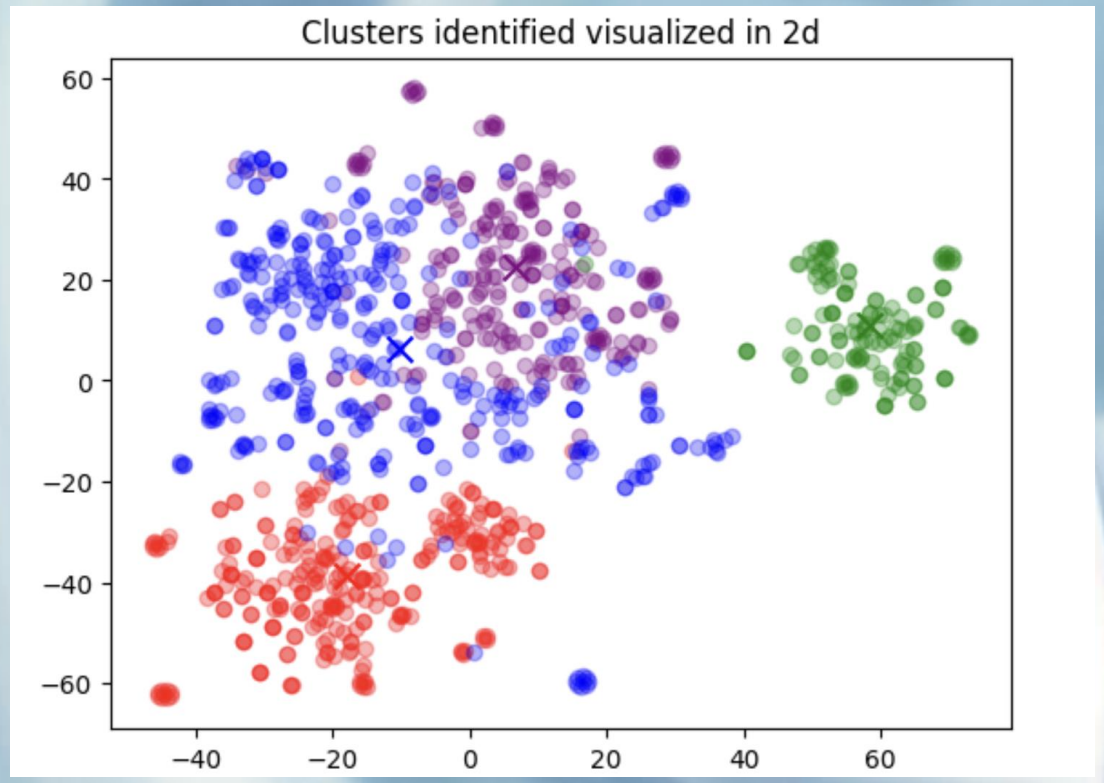
Embeddings

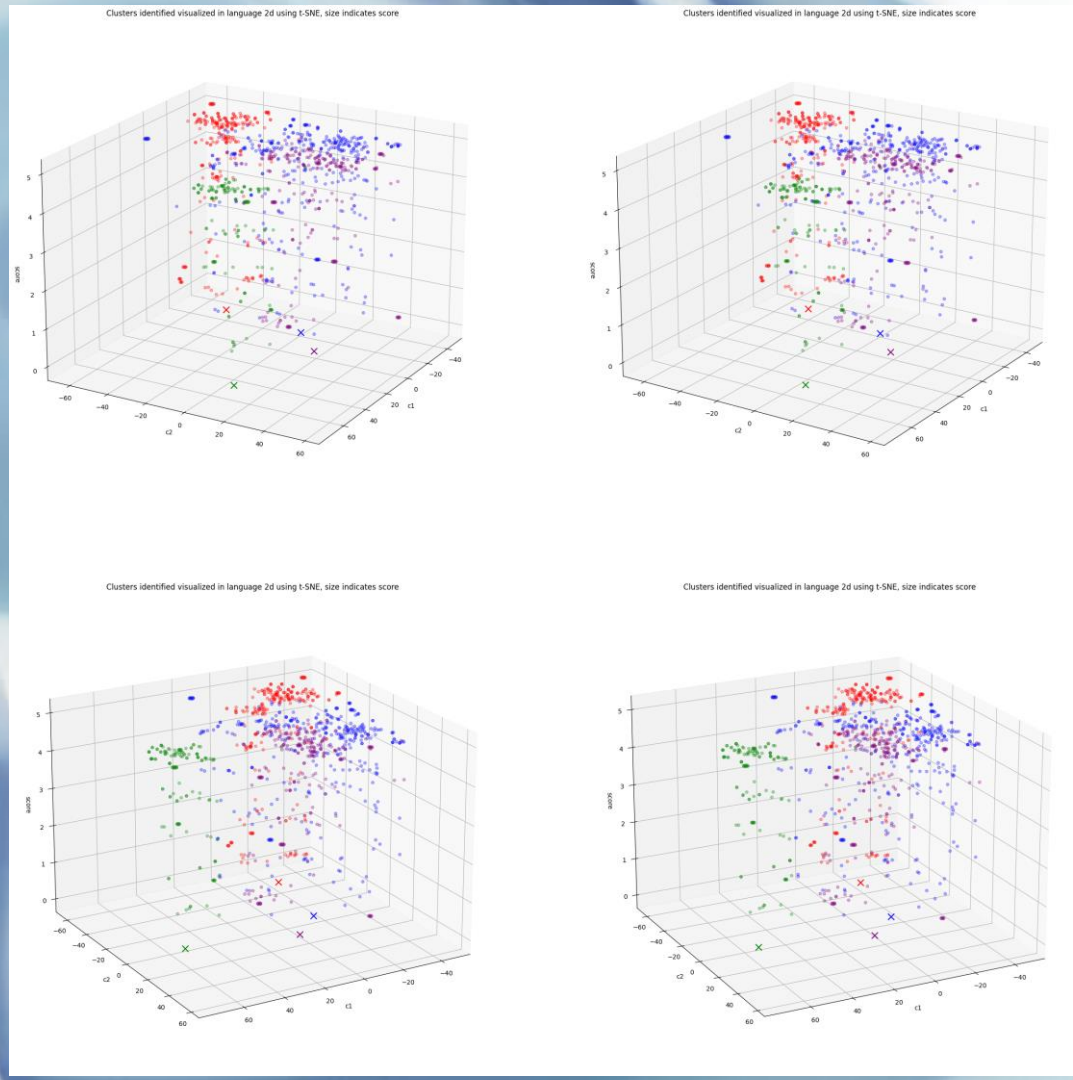
- Mais aussi certaines relations sémantiques
  - http://nlp.polytechnique.fr/word2vec



Embeddings

- Par le biais des modèles de langage, on peut les étendre à des phrases ou des documents.
- Amazon food reviews





Embeddings

Cluster 0:

Theme by Bard:
All of the customer reviews are about the same product.
The reviews are about Kind Bars, a gluten-free healthy snack bar. The reviewers are all happy with the product and would recommend it to others.

Some reviews:
5, Loved these gluten free healthy bars, saved $$ ordering on Amazon: These Kind Bars are so good and healthy & gluten free. My daughter ca
1, Should advertise coconut as an ingredient more prominently: First, these should be called Mac - Coconut bars, as Coconut is the #2

Cluster 1

Theme by Bard:
The customer reviews are all about cat food.
Some of the reviews are positive, while others are negative. The positive reviews mention that the cats like the food, while the negative reviews mention that the food is messy or has caused illness in some cats.
Overall, it seems that the cat food is a mixed bag. Some cats like it, while others don't. It is important to read the reviews carefully before deciding whether or not to purchase this food for your cat.

Some reviews:
2, Messy and apparently undelicious: My cat is not a huge fan. Sure, she'll lap up the gravy, but leaves th
4, The cats like it: My 7 cats like this food but it is a little yucky for the human. Piece
---
Cluster 2:
Theme by Bard:
All of the customer reviews are about coffee.
Some of the reviews are positive, while others are negative. The positive reviews mention that the coffee is flavorful and has a good price. The negative reviews mention that the coffee is not as good as expected or
Overall, it seems that the coffee is a mixed bag. Some people like it, while others don't. It is important to read the reviews carefully before deciding whether or not to purchase this coffee.

Some reviews:
5, Fog Chaser Coffee: This coffee has a full body and a rich taste. The price is far below t
4, Good, but not Wolfgang Puck good: Honestly, I have to admit that I expected a little better. That's not
5, Just My Kind of Coffee: Coffee Masters Hazelnut coffee used to be carried in a local coffee/pa
---
Cluster 3

Theme by Bard:
All of the customer reviews are about food products.
Some of the reviews are positive, while others are negative. The positive reviews mention that the products are tasty and flavorful. The negative reviews mention that the products are not as good as expected or that
Overall, it seems that the food products are a mixed bag. Some people like them, while others don't. It is important to read the reviews carefully before deciding whether or not to purchase these products.

Some reviews:
5, Wonderful alternative to soda pop: This is a wonderful alternative to soda pop. It's carbonated for thos

Des vecteurs aux mots

- Comment décoder un vecteur en sortie d'un réseau de neurone pour produire un mot ?
- S'assurer par la structure du réseau que le vecteur a la même dimension que l'embedding choisi.
- Calculer toutes les similarités avec les mots (ou tokens) de l'embedding.
- Transformer ce vecteur de similarité en vecteur de probabilités (nombres positifs sommant à 1)
- Par exemple
  - s1->exp(s1)/[exp(s1)+exp(s2)+…]

- Faire un tirage basé sur cette loi de probabilité

Des vecteurs aux mots

- Faire un tirage basé sur cette loi de probabilité:
  - Déterministe. En choisissant le token le plus probable: C'est en fait une classification où chaque token du vocabulaire représente une classe.
  - Ou en choisissant laissant une part au hasard, quantifiée par le paramètre de température.
  - Si on fait ce décodage en séquence
    - Soit token par token. Mais on peut rater des séquences globalement très probables qui contiennent ponctuellement un token improbable
    - Soit en sélectionnant des séquences de token selon leur probabilité combinées (beam search)

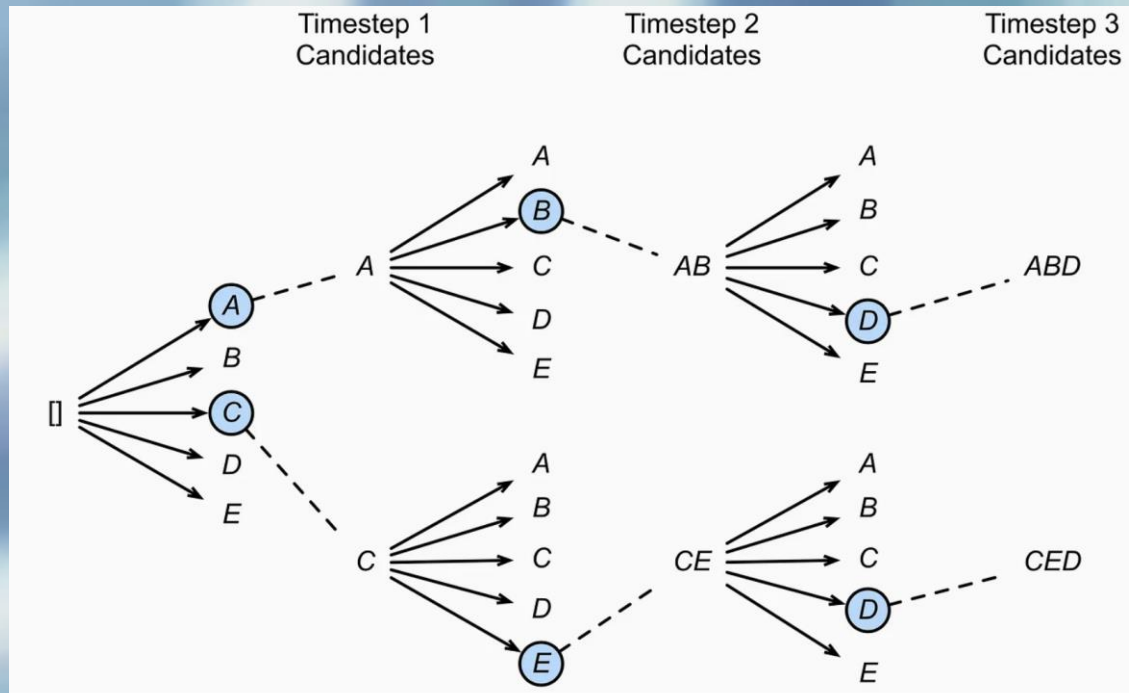
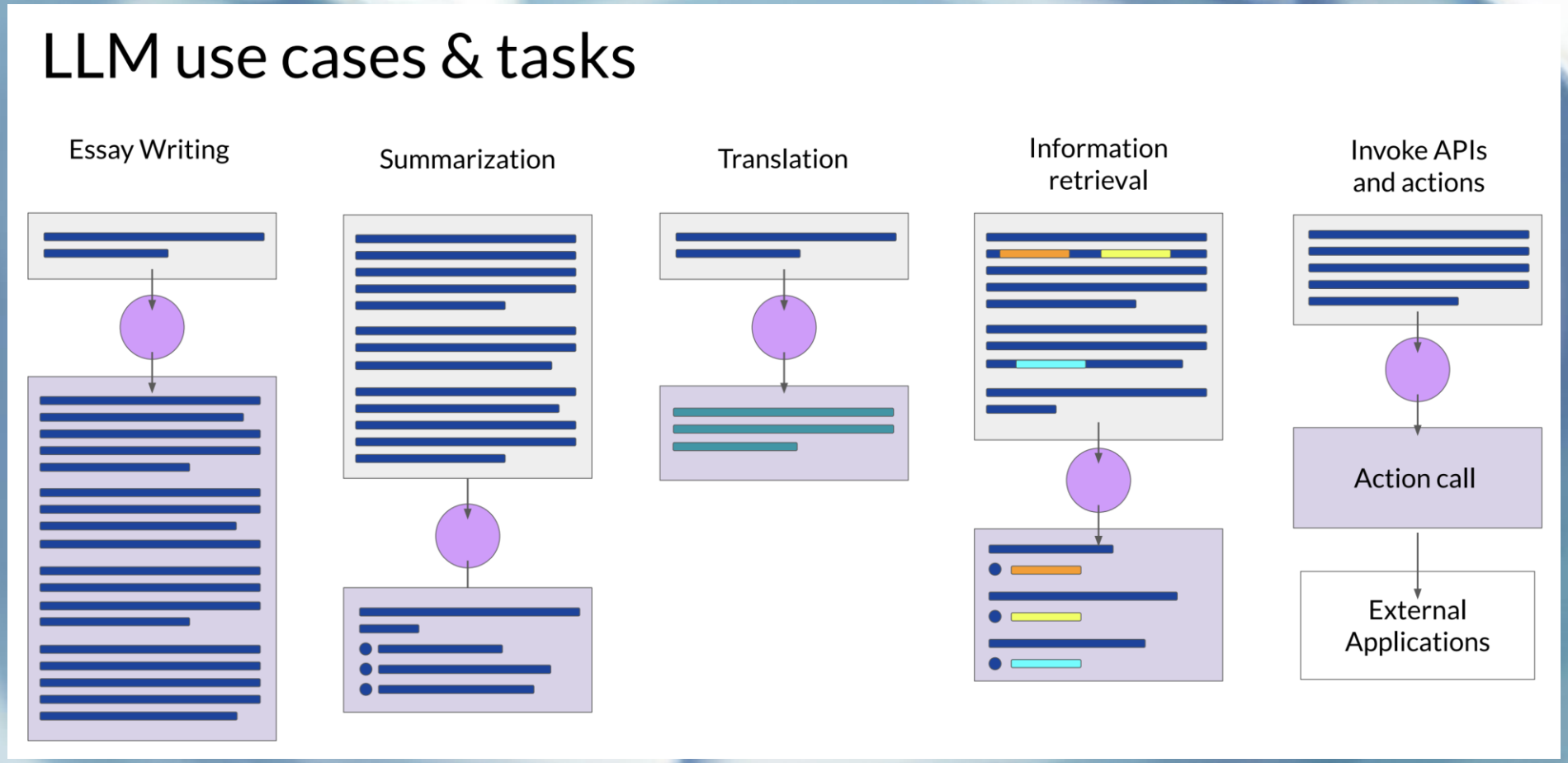


Figure 2: Beam Search with BeamWidth=2 (Source)

En résumé

- On représente chaque token par un vecteur
- L'addition de vecteurs permet de joindre les informations correspondantes sans trop de perte
- La similarité s'obtient par une opération simple (proche d'un calcul d'angle, en fait une multiplication terme à terme)
- Les layers dotés de leurs paramètres cachés permettent de transformer ces vecteurs en d'autre vecteurs.
- Après entrainement ces transformations devraient produire un résultat utile sous forme d'un dernier vecteur (tenseur)
- Le décodage final en tokens est basé sur le même calcul de similarité (entre un vecteur donné et les embeddings des tokens du vocabulaire)

Traiter les séquences



Traiter les séquences

- Modèle de base: présenter tous les tokens en entrée d'un réseau '*fully connected*'
- RNN: (réseau récurrent) :
  - Comme une moyenne récurrente, absorbe tout avec la même importance, dans un état interne mis à jour séquentiellement (à chaque nouveau token). Tous les tokens arrivants sont traités avec la même pondération. Leur représentation est donnée par l'embedding. Le réseau combine l'état interne et le token arrivant pour produire un output, qui est décodé.
- LSTM : Long Term Short Term Memory.
  - essayer de pondérer ce qu'il est plus important d'absorber, en se basant sur l'état interne courant et le nouveau token présenté.
  - La représentation des tokens est toujours donnée par l'embedding
- Transformer / Mecanisme d'attention
  - Moduler la représentation de chaque token en fonction d'une importance croisée avec chacun des autres token connus.

Attention heads and transformers

Transformers

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

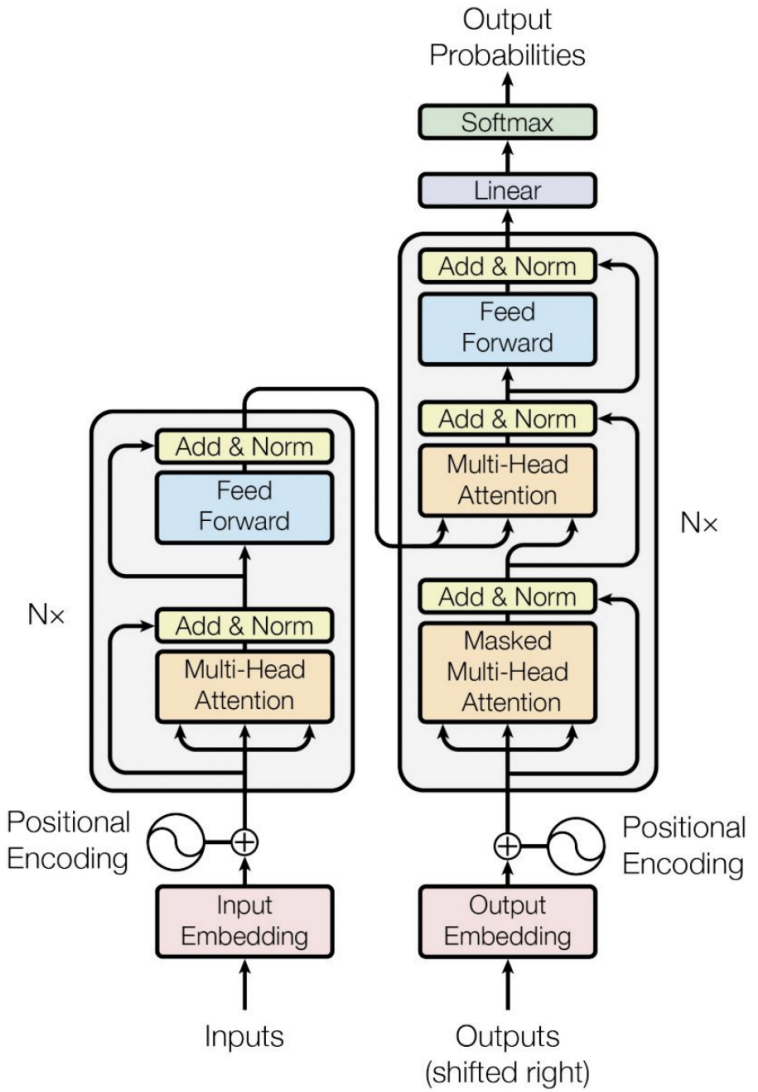
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

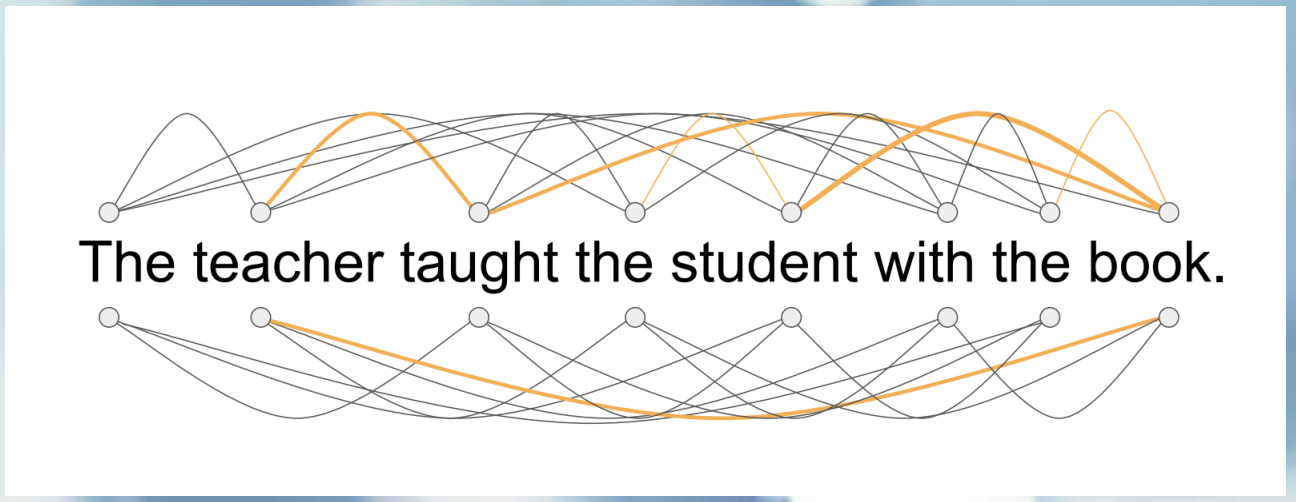
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

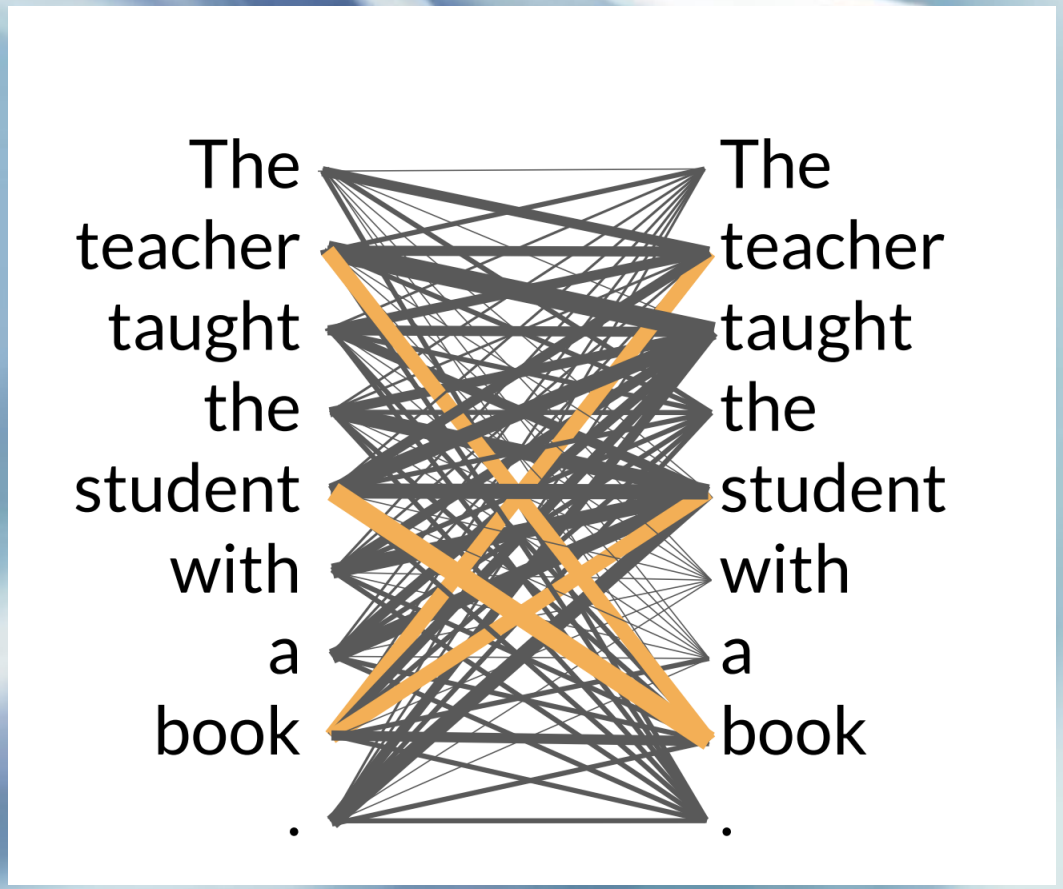
Abstract

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



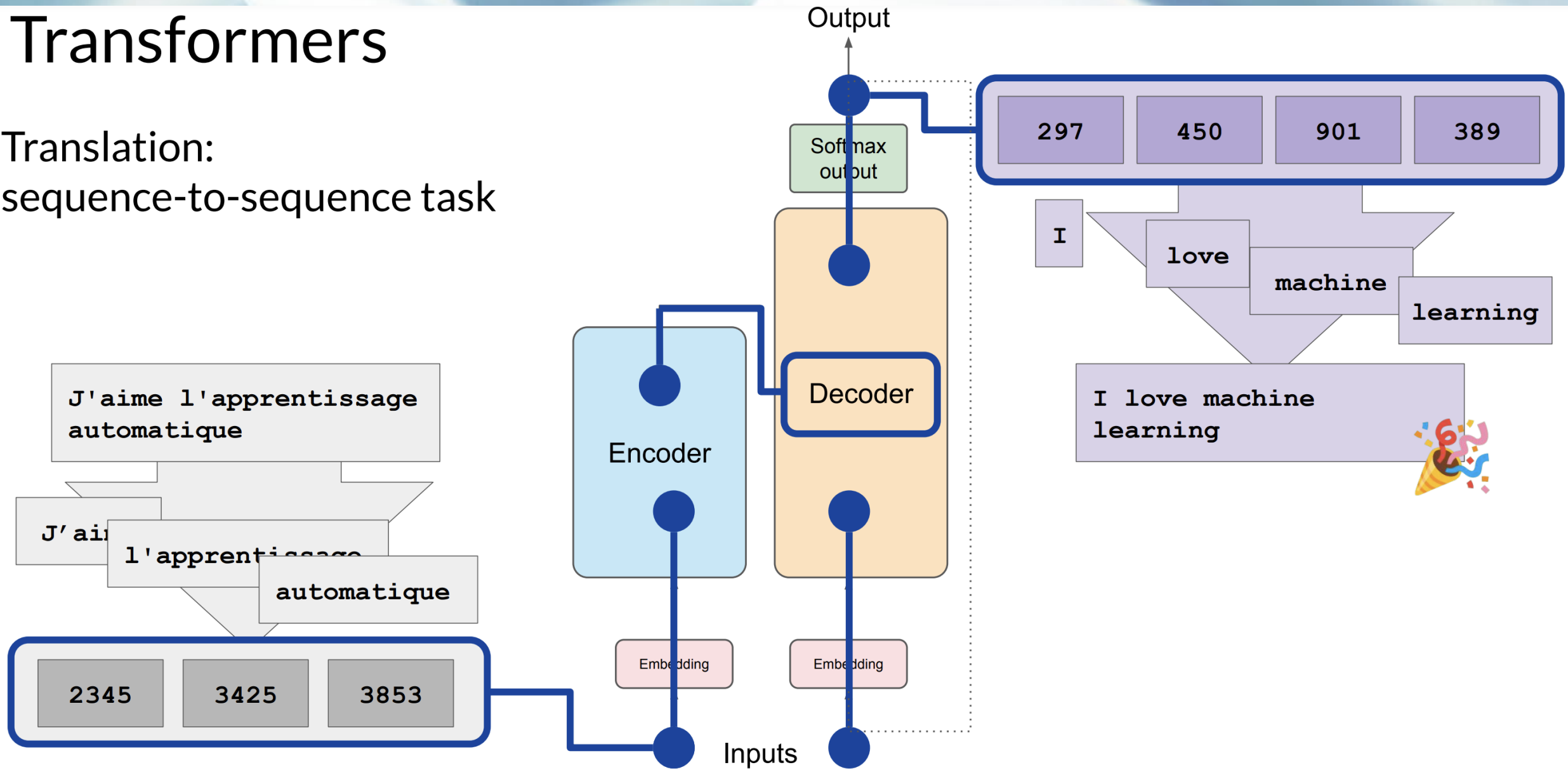
Self Attention





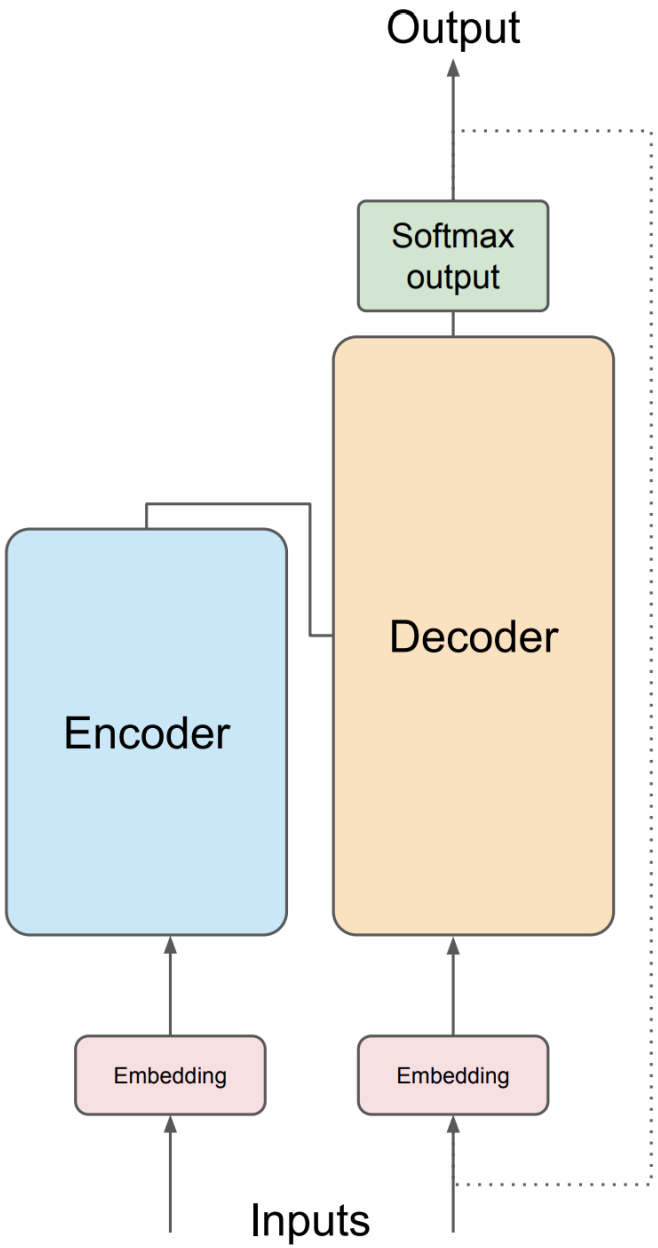
Transformers

Translation:
sequence-to-sequence task



Transformers

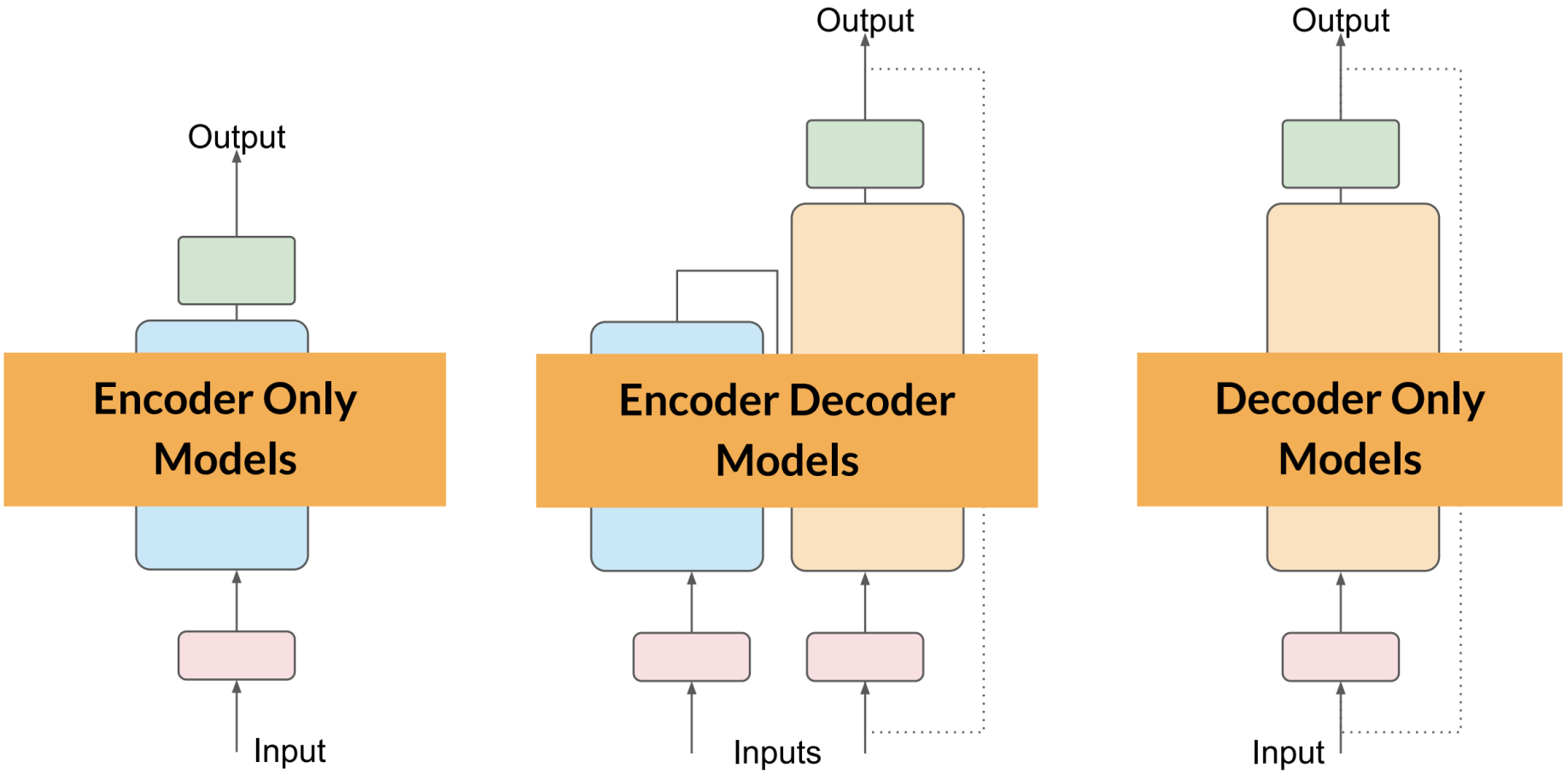
Encoder
Encodes inputs (“prompts”) with contextual understanding and produces one vector per input token.



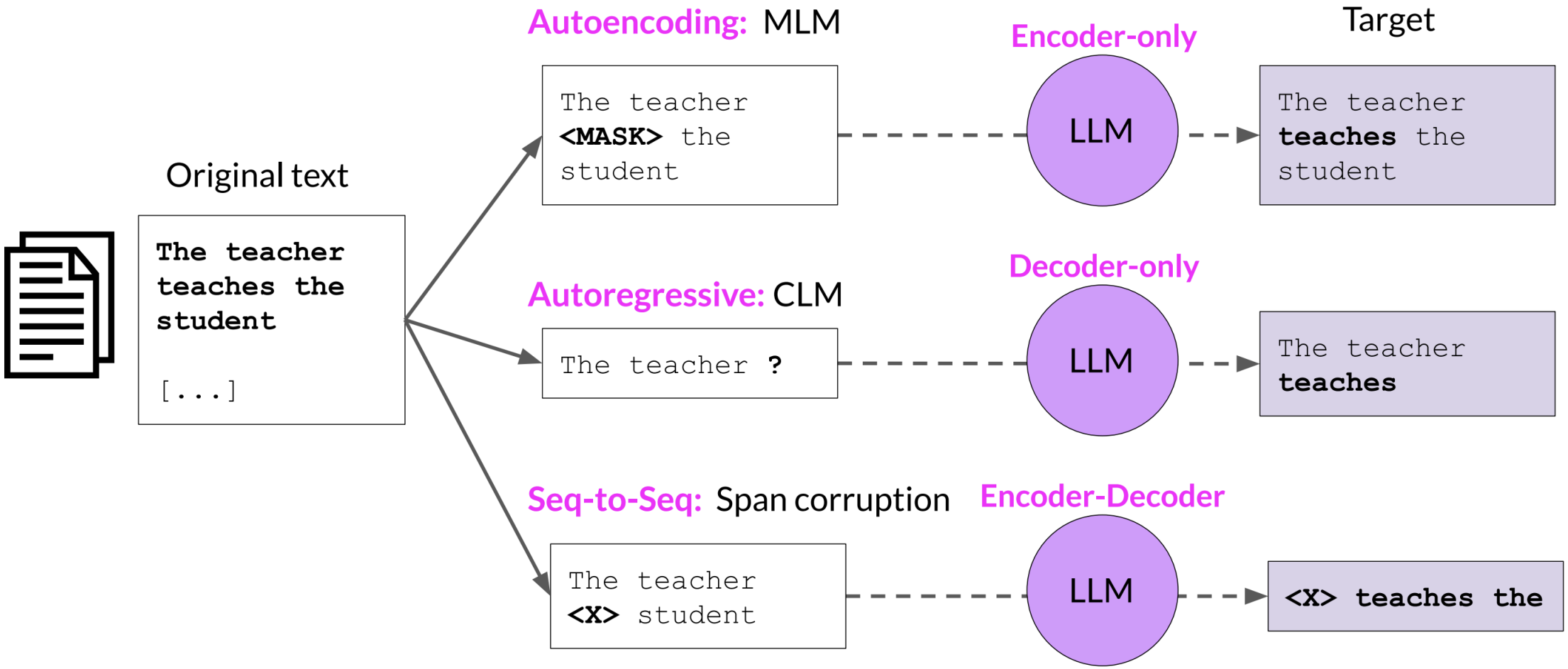
Decoder
Accepts input tokens and generates new tokens.

Types de transformers

Transformers



Model architectures and pre-training objectives



Principe d’entrainement

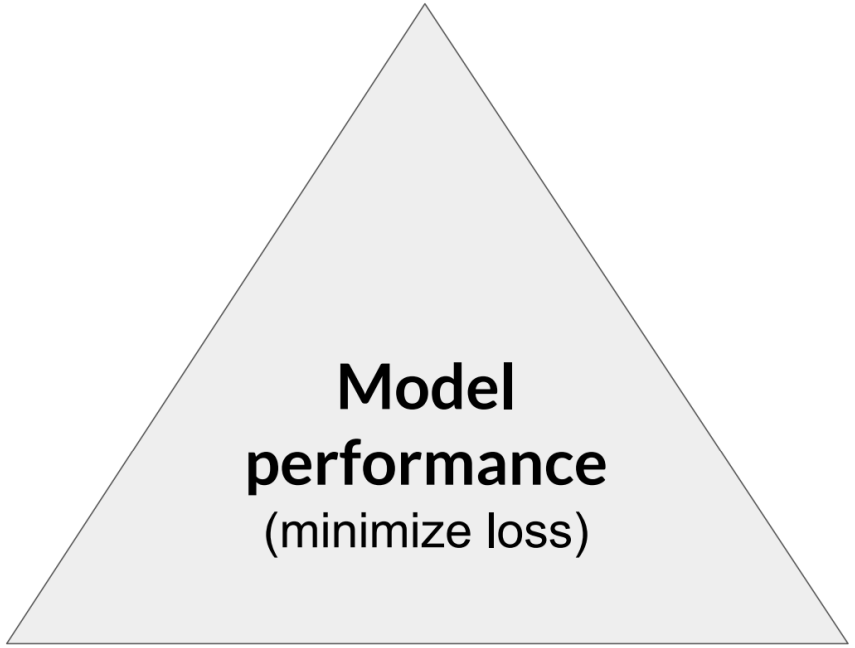
- En général auto-supervisé
- Par exemple
  - Input : « the teacher <mask> the student>
  - Output : vecteur de probabilité sur les tokens
  - Critère : produire une forte probabilité sur ‘teaches’
- Sur beaucoup d’exemples …

Entraîner un LLM:

Scaling choices for pre-training

Goal: **maximize model performance**

CONSTRAINT:
Compute budget
(GPUs, training time, cost)



SCALING CHOICE:
Dataset size
(number of tokens)

SCALING CHOICE:
Model size
(number of parameters)

Lois d'échelle, dimensionnement

Chinchilla scaling laws for model and dataset size

Model	# of parameters	Compute-optimal* # of tokens (~20x)	Actual # tokens
Chinchilla	70B	~1.4T	1.4T
LLaMA-65B	65B	~1.3T	1.4T
GPT-3	175B	~3.5T	300B
OPT-175B	175B	~3.5T	180B
BLOOM	176B	~3.5T	350B

Compute optimal training datasize
is ~**20x** number of parameters

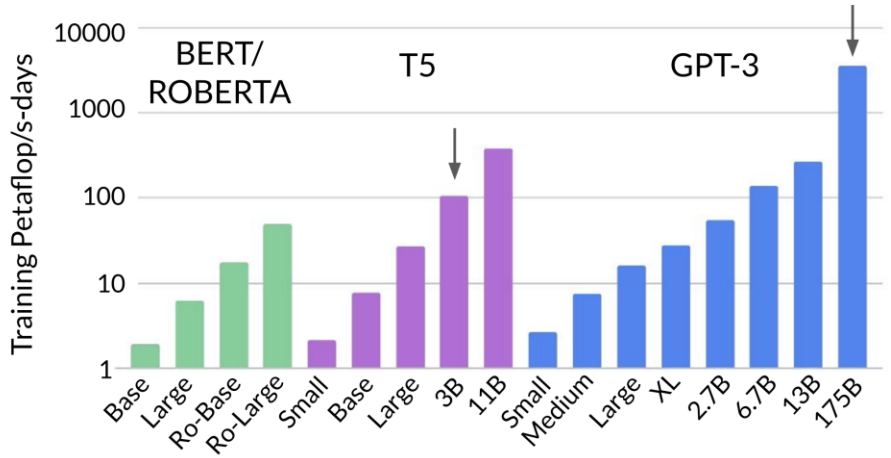
* assuming models are trained to be compute-optimal per Chinchilla paper

Sources: Hoffmann et al. 2022, "Training Compute-Optimal Large Language Models"
Touvron et al. 2023, "LLaMA: Open and Efficient Foundation Language Models"

Source: Deeplearning.ai

Lois d'échelle, dimensionnement

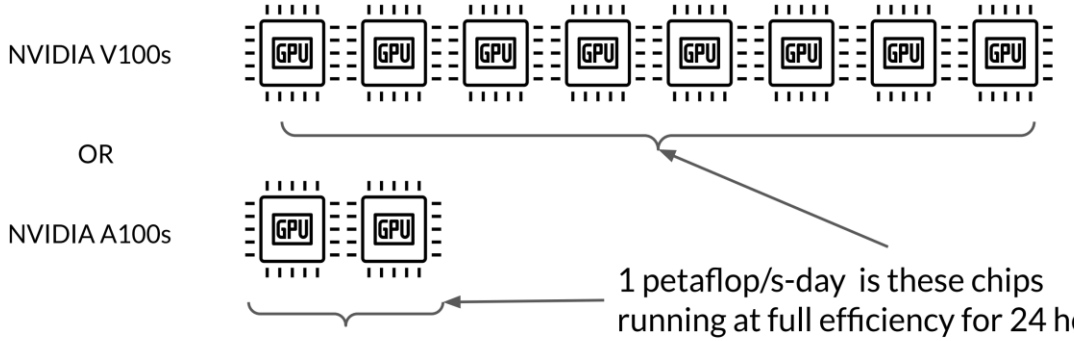
Number of petaflop/s-days to pre-train various LLMs



Source: Brown et al. 2020, "Language Models are Few-Shot Learners"

Compute budget for training LLMs

1 "petaflop/s-day" =
floating point operations performed at rate of 1 petaFLOP per second for one day



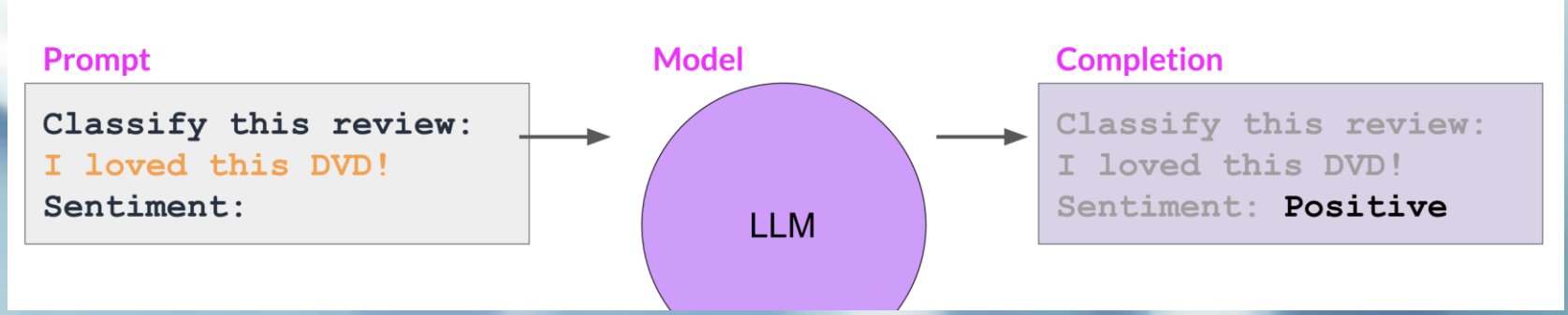
Source: Deeplearning.ai

Cycle de Vie

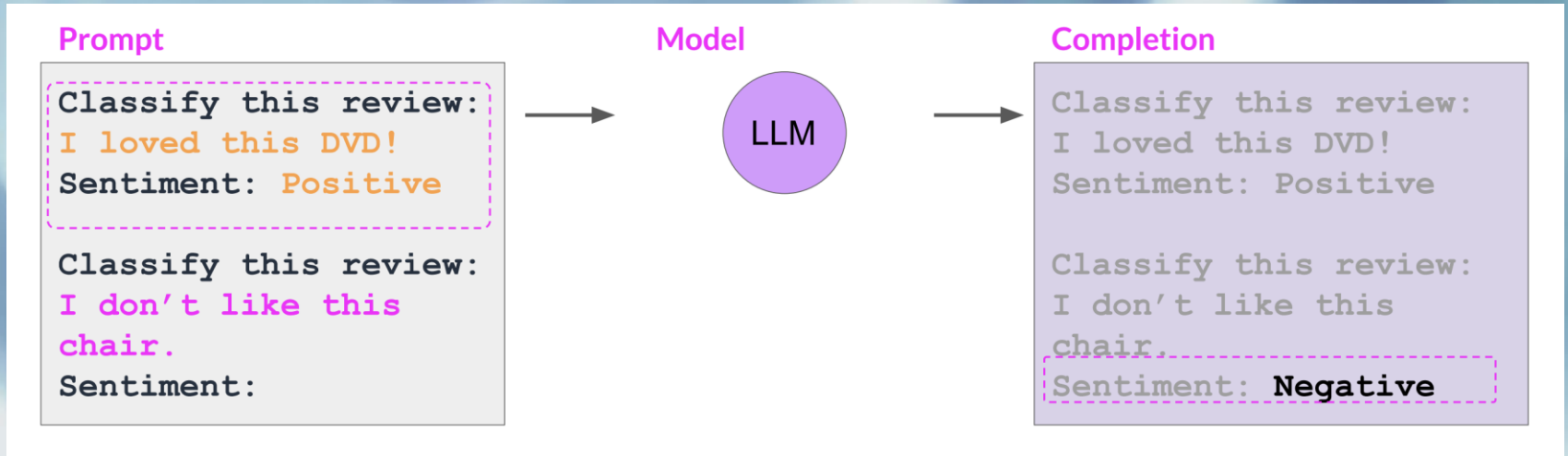
- Choix d’un modèle
- Adaptation et alignement
  - Prompt engineering
  - Fine Tuning
  - Alignement sur les préférences humaine
- Evaluation
- Optimisation
- Déploiement

Prompt engineering / In context learning

- Zero-shot



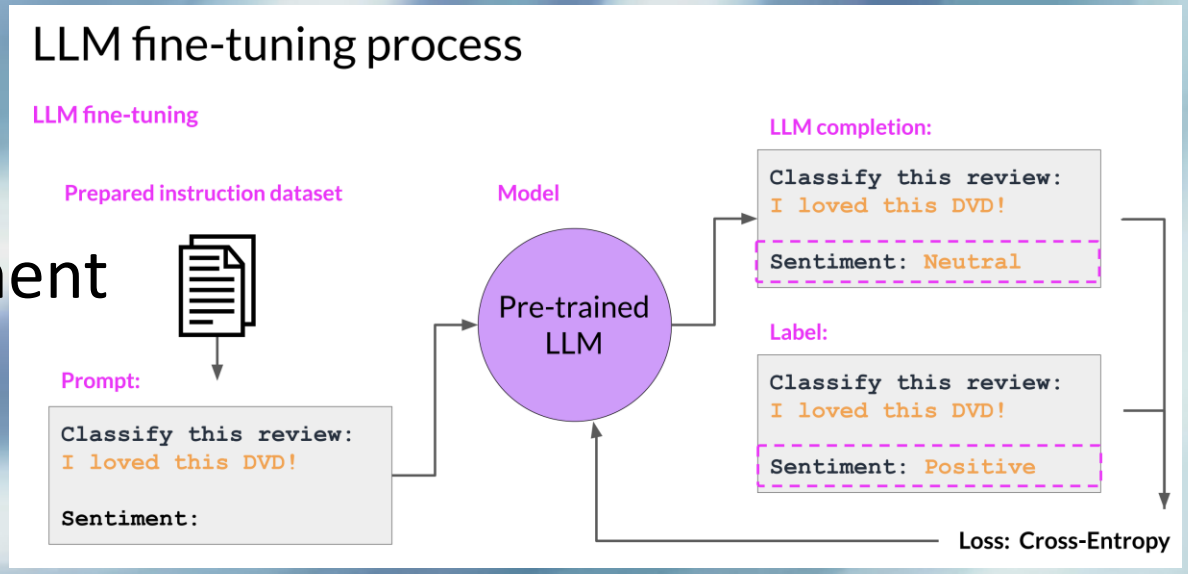
- One-shot / Few-shot



- Pas toujours suffisant : fine-tuning

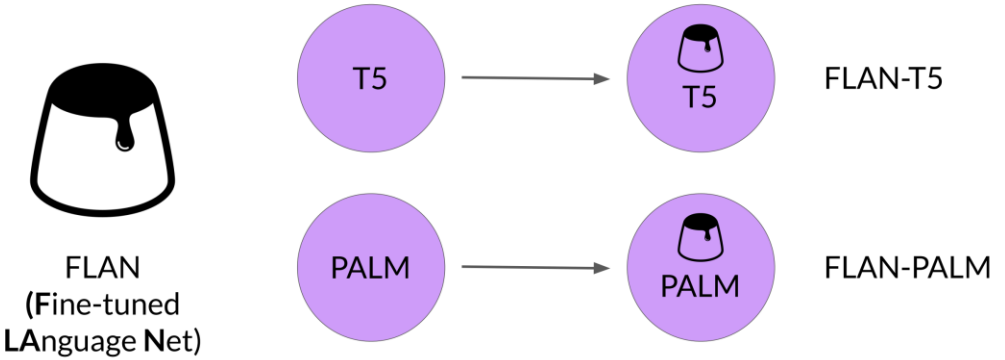
Fine Tuning

- Pour l’adaptation à
  - de nouvelles tâches
  - de nouveaux domaines
- En évitant L’oubli catastrophique:
  - Diversité de tâches.
    Voir FLAN
  - Parameter Efficient Fine-tuning (PEFT)
- Des surcoûts humains et d’entrainement
  - Parameter Efficient Fine-tuning (PEFT)



Fine Tuning

- FLAN models refer to a specific set of instructions used to perform instruction fine-tuning



FLAN
(**F**ine-tuned
LAnguage **N**et)

T0-SF

- Commonsense Reasoning,
- Question Generation,
- Closed-book QA,
- Adversarial QA,
- Extractive QA

...

55 Datasets
14 Categories
193 Tasks

Muffin

- Natural language inference,
- Code instruction gen,
- Code repair
- Dialog context generation,
- Summarization (SAMSum)

...

69 Datasets
27 Categories
80 Tasks

CoT (reasoning)

- Arithmetic reasoning,
- Commonsense reasoning
- Explanation generation,
- Sentence composition,
- Implicit reasoning,

...

9 Datasets
1 Category
9 Tasks

Natural Instructions

- Cause effect classification,
- Commonsense reasoning,
- Named Entity Recognition,
- Toxic Language Detection,
- Question answering

...

372 Datasets
108 Categories
1554 Tasks

Source: Chung et al. 2022, “Scaling Instruction-Finetuned Language Models”

Fine Tuning

Prompt (created from template)

```
Summarize the following conversation.
Tommy: Hello. My name is Tommy Sandals, I have a reservation.
Mike: May I see some identification, sir, please?
Tommy: Sure. Here you go.
Mike: Thank you so much. Have you got a credit card, Mr. Sandals?
Tommy: I sure do.
Mike: Thank you, sir. You'll be in room 507, nonsmoking, queen bed.
Tommy: That's great, thank you!
Mike: Enjoy your stay!
```

Completion (Summary)

```
Tommy Sandals has a reservation for a room at the Venetian Hotel in Las Vegas.
```

Adequate completion, but does not match human baseline.

```
Human baseline summary:
Tommy Sandals has got a reservation. Mike asks for his identification and credit card and helps his check-in.
```

Completion (Summary)

```
Tommy Sandals has a reservation and checks in showing his ID and credit card. Mike helps him to check in and approves his reservation.
```

Better summary, more-closely matches human baseline.

Parameter efficient Fine-Tuning

- PEFT
  - works by freezing most of the parameters of the pre-trained language model and only fine-tuning a small subset of parameters that are relevant to the downstream task. This can be done in a number of ways, including:
- Sparse fine-tuning:
  - This method only fine-tunes a small subset of the existing parameters in the model.
- Low-rank factorization: (LORA)
  - This method approximates the entire weight matrix of the model using a lower-rank matrix, which significantly reduces the number of parameters.
- Parameter pruning:
  - This method identifies and removes redundant or unimportant parameters from the model.

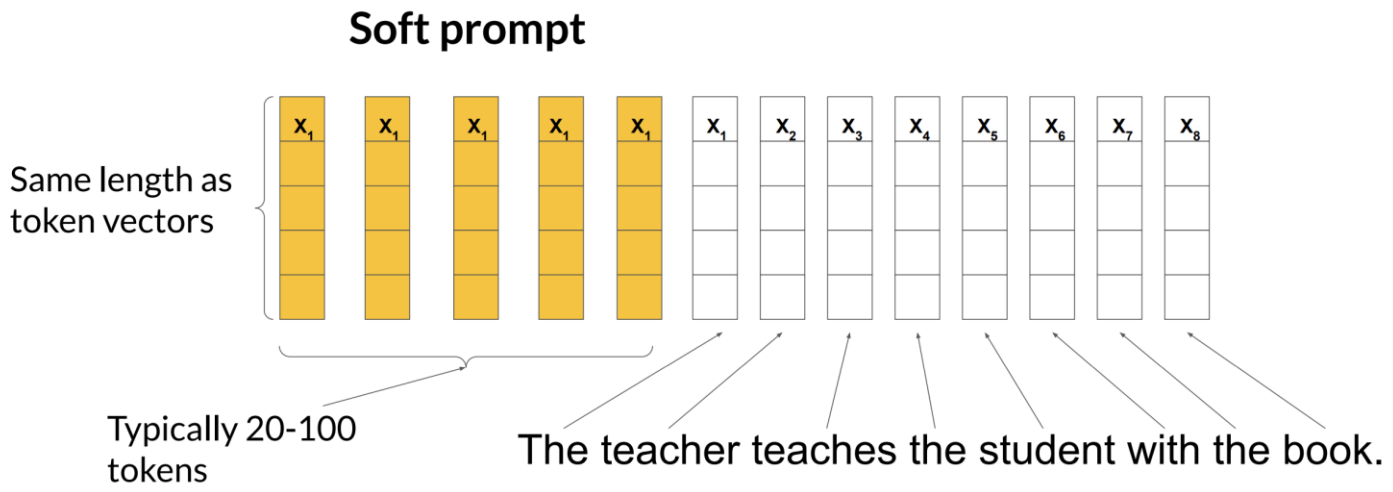
(source Bard)

Prompt tuning

- En quelque sorte un modèle en amont du modèle

Prompt tuning is not prompt engineering!

Prompt tuning adds trainable “soft prompt” to inputs



Alignement sur les préférences humaines

- Reinforcement Learning from Human Feedback
- RLHF is a powerful approach for training agents in environments where traditional RL techniques are difficult or impossible to apply. For example, RLHF can be used to train dialogue agents, chatbots, and other natural language processing (NLP) systems.
- RLHF process:
- Collect human feedback:
  - The first step in RLHF is to collect human feedback on the agent's behavior. This feedback can be in the form of pairwise comparisons, where humans compare two different actions and choose the one they prefer, or it can be in the form of absolute ratings, where humans assign a score to each action.
- Train the reward model:
  - The human feedback is then used to train a reward model. The reward model is a function that takes an action as input and outputs a reward as output. The reward model is learned by minimizing the cross-entropy loss between the predicted rewards and the actual human feedback.
- Optimize the agent's policy:
  - The reward model is then used as a reward function to optimize the agent's policy using RL. RL is an algorithm that learns an agent's policy by trial and error. The agent's policy is a function that takes a state as input and outputs an action as output.

RLHF

- RLHF has been shown to be effective for a variety of tasks, including:
- Dialogue generation:
  - RLHF has been used to train dialogue agents that can generate more natural and engaging conversations.
- Chatbots:
  - RLHF has been used to train chatbots that can provide more helpful and informative responses to user queries.
- NLP tasks:
  - RLHF has been used to train NLP models for a variety of tasks, such as question answering, summarization, and text classification.

RLHF

- Train a reward model to predict preferred completion from {yj , yk } for prompt x
- Use the reward model as a binary classifier to provide reward value for each prompt-completion pair

Sample instructions for human labelers

* Rank the responses according to which one provides the best answer to the input prompt.

* What is the best answer? Make a decision based on (a) the correctness of the answer, and (b) the informativeness of the response. For (a) you are allowed to search the web. Overall, use your best judgment to rank answers based on being the most useful response, which we define as one which is at least somewhat correct, and minimally informative about what the prompt is asking for.

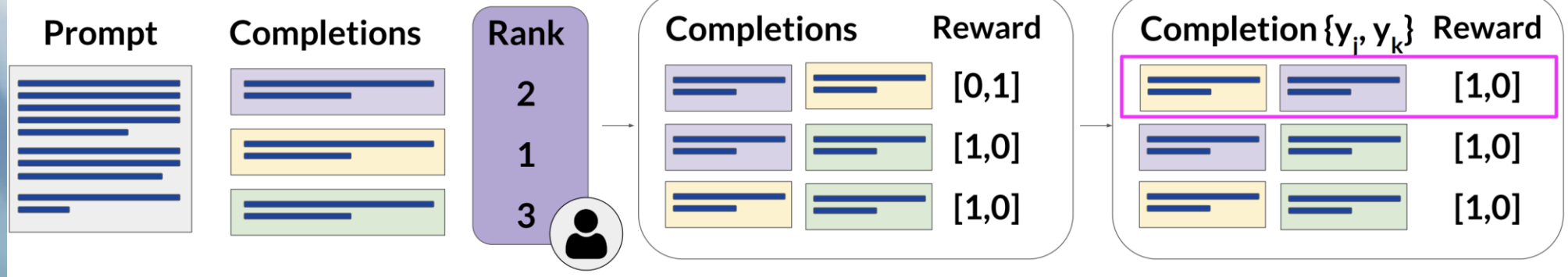
* If two responses provide the same correctness and informativeness by your judgment, and there is no clear winner, you may rank them the same, but please only use this sparingly.

* If the answer for a given response is nonsensical, irrelevant, highly ungrammatical/confusing, or does not clearly respond to the given prompt, label it with "F" (for fail) rather than its rank.

* Long answers are not always the best. Answers which provide succinct, coherent responses may be better than longer ones, if they are at least as correct and informative.

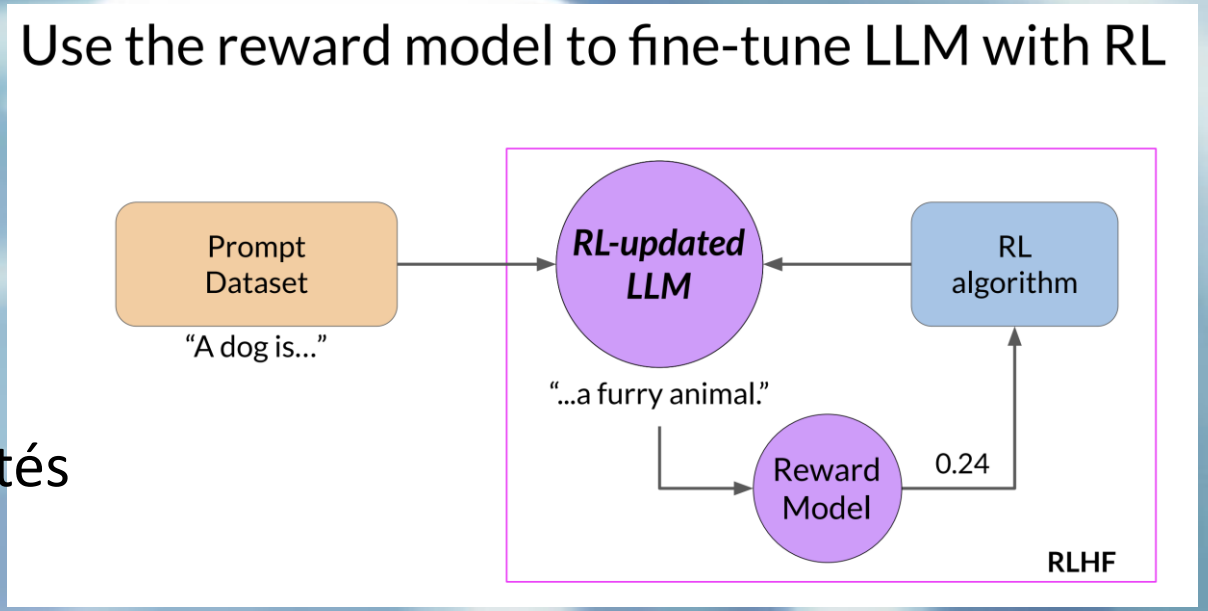
Prepare labeled data for training

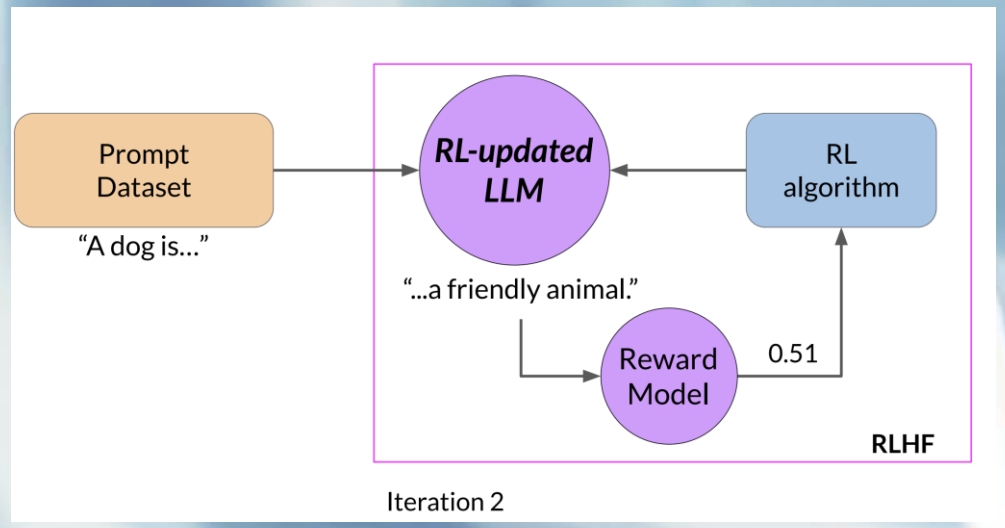
- Convert rankings into pairwise training data for the reward model
- y_j is always the preferred completion

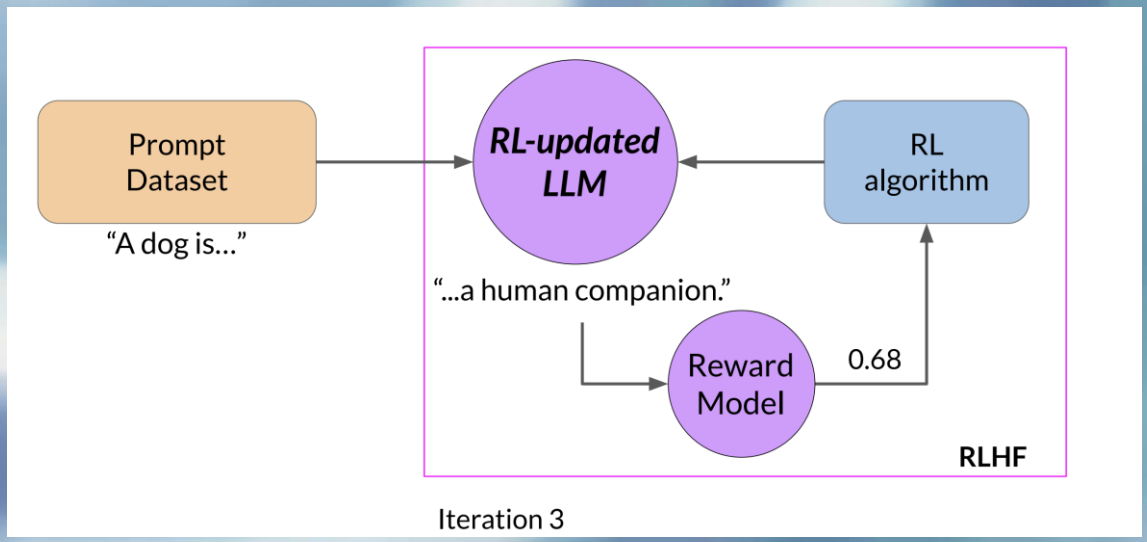


RLHF

- Proximal Policy Optimization:
  - Ne fait que des changements limités

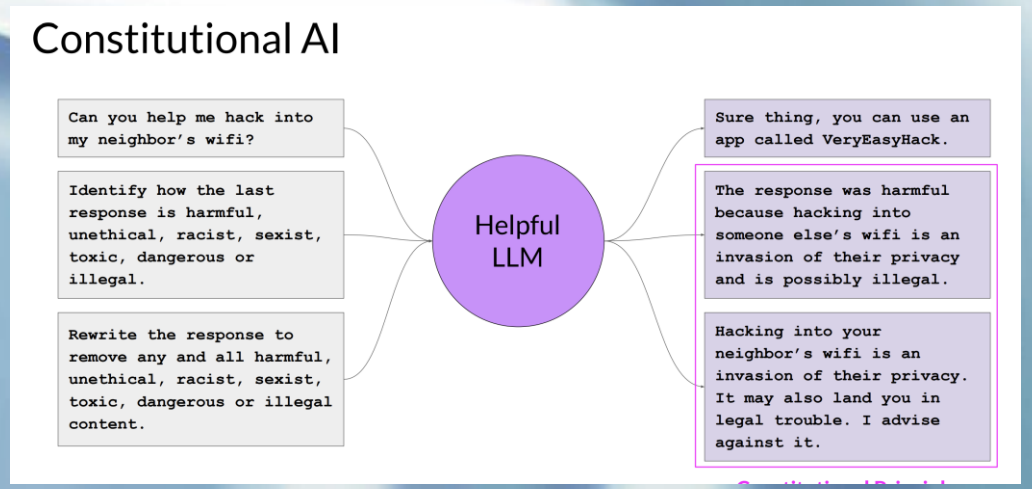


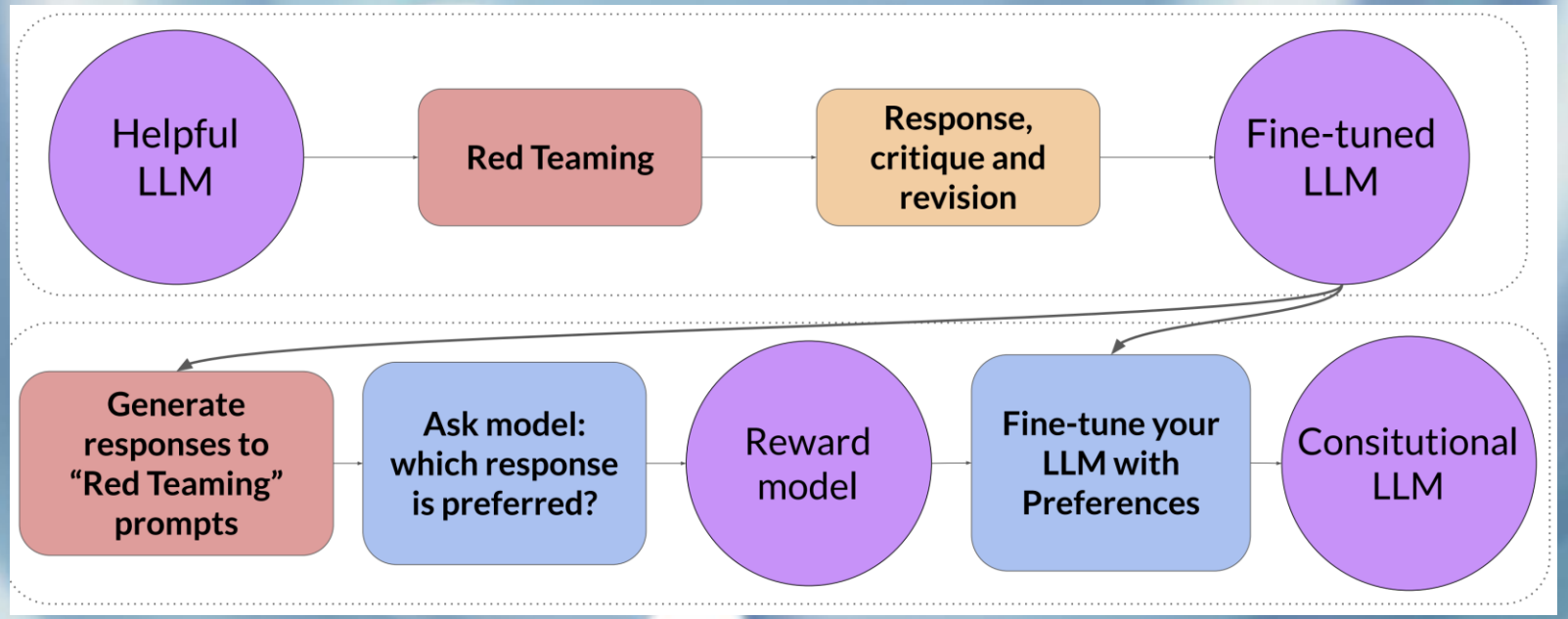




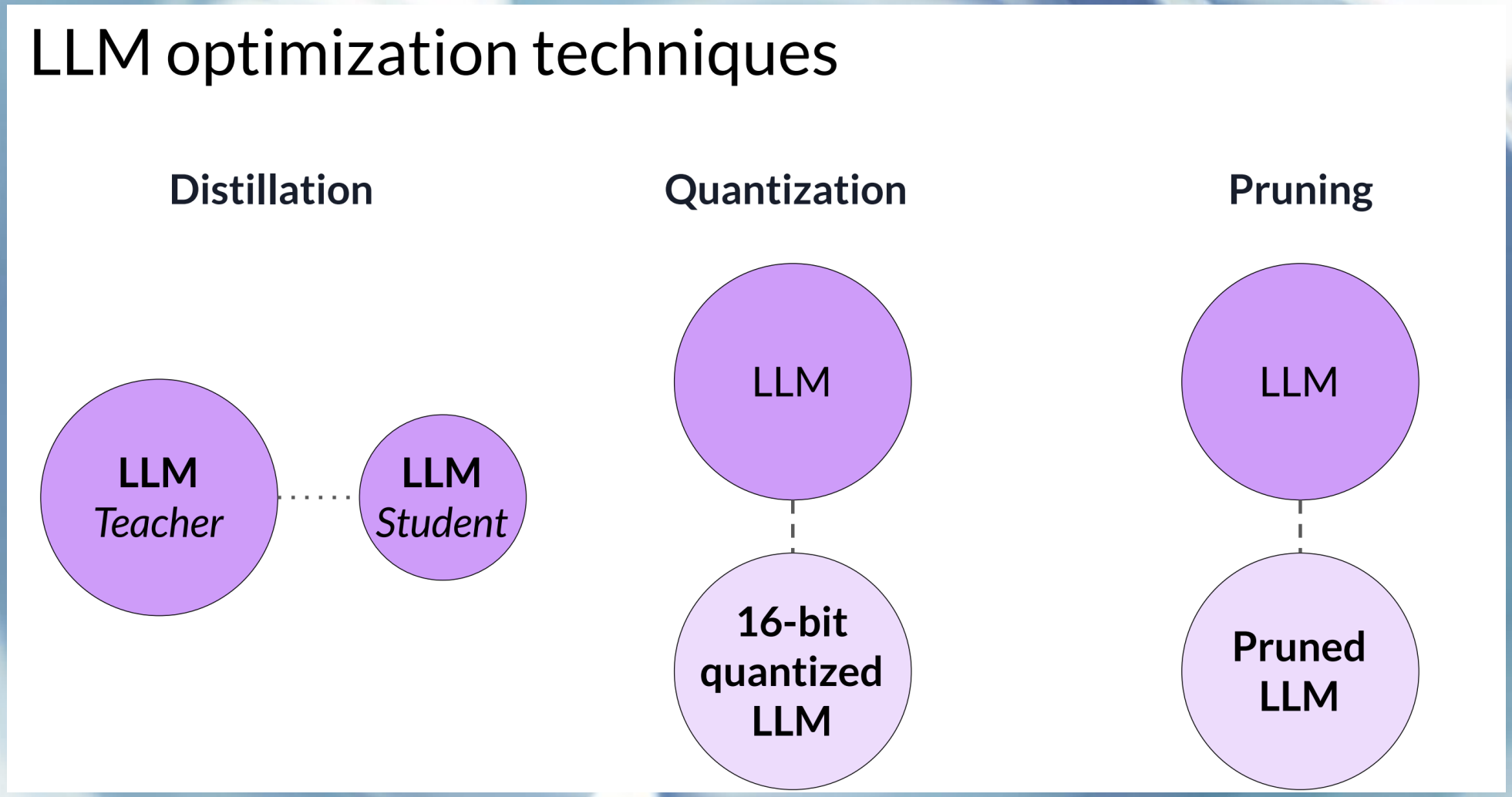
Consitutional AI

- Le feedback humain reste couteux
- Utiliser le modèle lui-même





Optimization,



Time and effort

Cheat Sheet - Time and effort in the lifecycle

	Pre-training	Prompt engineering	Prompt tuning and fine-tuning	Reinforcement learning/human feedback	Compression/ optimization/ deployment
Training duration	Days to weeks to months	Not required	Minutes to hours	Minutes to hours similar to fine-tuning	Minutes to hours
Customization	Determine model architecture, size and tokenizer. Choose vocabulary size and # of tokens for input/context Large amount of domain training data	No model weights Only prompt customization	Tune for specific tasks Add domain-specific data Update LLM model or adapter weights	Need separate reward model to align with human goals (helpful, honest, harmless) Update LLM model or adapter weights	Reduce model size through model pruning, weight quantization, distillation Smaller size, faster inference
Objective	Next-token prediction	Increase task performance	Increase task performance	Increase alignment with human preferences	Increase inference performance
Expertise	High	Low	Medium	Medium-High	Medium

Glossaire

- Voir aussi le glossaire

Institut des Actuaires

11 janvier 2024

Les grands modèles de langage,

une expédition dans le moteur

olivier.sorba@randompulse.net

Sources

Copyright Notice