RÉPUBLIQUE TOGOLAISE
Ministère de l'Économie Numérique et de la Transformation Digitale
Togo Data Lab

Masterclass organisée par :
Togo Data Lab
Fondements Mathématiques des Transformers et des LLMs

UNIVERSITE DE KARA

Module 3 : Des Transformers aux LLMs : fondements et pré-entraînement

Présentée par : Tiebekabe Pagdame
Enseignant-chercheur - Université de Kara

Dates : 15-16 juillet 2025

RÉPUBLIQUE TOGOLAISE
Ministère de l'Économie Numérique et de la Transformation Digitale
Togo Data Lab

RÉPUBLIQUE TOGOLAISE
Ministère de l'Économie Numérique et de la Transformation Digitale

atd AGENCE TOGO DIGITAL

Sommaire

Modèle de Langage Probabiliste

- Un **modèle de langage** assigne une probabilité $P(w_1, w_2, \ldots, w_T)$ à une séquence de mots.
- Objectif : capturer les *règles syntaxiques et sémantiques* à partir de données textuelles.

Formulation mathématique

$$P(w_1, w_2, \ldots, w_T) = \prod_{t=1}^{T} P(w_t \mid w_1, \ldots, w_{t-1})$$

- On approxime souvent par des modèles de Markov d'ordre n :

$$P(w_t \mid w_1, \ldots, w_{t-1}) \approx P(w_t \mid w_{t-n+1}, \ldots, w_{t-1})$$

Modèle de Langage Probabiliste

- Un **modèle de langage** assigne une probabilité $P(w_1, w_2, \ldots, w_T)$ à une séquence de mots.
- Objectif : capturer les *règles syntaxiques et sémantiques* à partir de données textuelles.

Formulation mathématique

$$P(w_1, w_2, \ldots, w_T) = \prod_{t=1}^{T} P(w_t \mid w_1, \ldots, w_{t-1})$$

- On approxime souvent par des modèles de Markov d'ordre n :

$$P(w_t \mid w_1, \ldots, w_{t-1}) \approx P(w_t \mid w_{t-n+1}, \ldots, w_{t-1})$$

Modèle de Langage Probabiliste

- Un **modèle de langage** assigne une probabilité $P(w_1, w_2, \ldots, w_T)$ à une séquence de mots.
- Objectif : capturer les *règles syntaxiques et sémantiques* à partir de données textuelles.

Formulation mathématique

$$P(w_1, w_2, \ldots, w_T) = \prod_{t=1}^{T} P(w_t \mid w_1, \ldots, w_{t-1})$$

- On approxime souvent par des modèles de Markov d'ordre n :

$$P(w_t \mid w_1, \ldots, w_{t-1}) \approx P(w_t \mid w_{t-n+1}, \ldots, w_{t-1})$$

Exemple simple de modèle de langage

- Vocabulaire : {je, suis, content}
- Corpus : “je suis content”, “je suis”
- Estimation de $P(suis \mid je)$:

$$P(suis \mid je) = \frac{\text{count}(\texttt{je suis})}{\text{count}(\texttt{je})} = \frac{2}{2} = 1$$

- Estimation de $P(content \mid suis)$:

$$P(content \mid suis) = \frac{1}{2}$$

Les modèles neuronaux remplacent les comptages par des prédictions paramétriques.

Exemple simple de modèle de langage

- Vocabulaire : {je, suis, content}
- Corpus : “je suis content”, “je suis”
- Estimation de $P(suis \mid je)$:

$$P(suis \mid je) = \frac{\text{count}(\texttt{je suis})}{\text{count}(\texttt{je})} = \frac{2}{2} = 1$$

- Estimation de $P(content \mid suis)$:

$$P(content \mid suis) = \frac{1}{2}$$

Les modèles neuronaux remplacent les comptages par des prédictions paramétriques.

Exemple simple de modèle de langage

- Vocabulaire : {je, suis, content}
- Corpus : “je suis content”, “je suis”
- Estimation de $P(suis \mid je)$:

$$P(suis \mid je) = \frac{\text{count}(\texttt{je suis})}{\text{count}(\texttt{je})} = \frac{2}{2} = 1$$

- Estimation de $P(content \mid suis)$:

$$P(content \mid suis) = \frac{1}{2}$$

Les modèles neuronaux remplacent les comptages par des prédictions paramétriques.

Exemple simple de modèle de langage

- Vocabulaire : {je, suis, content}
- Corpus : “je suis content”, “je suis”
- Estimation de $P(suis \mid je)$:

$$P(suis \mid je) = \frac{\text{count}(\texttt{je suis})}{\text{count}(\texttt{je})} = \frac{2}{2} = 1$$

- Estimation de $P(content \mid suis)$:

$$P(content \mid suis) = \frac{1}{2}$$

Les modèles neuronaux remplacent les comptages par des prédictions paramétriques.

Fonction de Vraisemblance et Entraînement

Objectif

Maximiser la **vraisemblance** de la séquence observée $(w_1, \ldots, w_T)$:

$$\mathcal{L}(\theta) = \prod_{t=1}^{T} P_\theta(w_t \mid w_1, \ldots, w_{t-1})$$

Log-vraisemblance (optimisation)

$$\log \mathcal{L}(\theta) = \sum_{t=1}^{T} \log P_\theta(w_t \mid w_1, \ldots, w_{t-1})$$

- En pratique, on minimise la **fonction de perte négative** :

$$\mathcal{J}(\theta) = -\sum_{t=1}^{T} \log P_\theta(w_t \mid w_1, \ldots, w_{t-1})$$

Fonction de Vraisemblance et Entraînement

Objectif

Maximiser la **vraisemblance** de la séquence observée $(w_1, \ldots, w_T)$:

$$\mathcal{L}(\theta) = \prod_{t=1}^{T} P_\theta(w_t \mid w_1, \ldots, w_{t-1})$$

Log-vraisemblance (optimisation)

$$\log \mathcal{L}(\theta) = \sum_{t=1}^{T} \log P_\theta(w_t \mid w_1, \ldots, w_{t-1})$$

- En pratique, on minimise la **fonction de perte négative** :

$$\mathcal{J}(\theta) = -\sum_{t=1}^{T} \log P_\theta(w_t \mid w_1, \ldots, w_{t-1})$$

Fonction de Vraisemblance et Entraînement

Objectif

Maximiser la **vraisemblance** de la séquence observée $(w_1, \ldots, w_T)$:

$$\mathcal{L}(\theta) = \prod_{t=1}^{T} P_\theta(w_t \mid w_1, \ldots, w_{t-1})$$

Log-vraisemblance (optimisation)

$$\log \mathcal{L}(\theta) = \sum_{t=1}^{T} \log P_\theta(w_t \mid w_1, \ldots, w_{t-1})$$

- En pratique, on minimise la **fonction de perte négative** :

$$\mathcal{J}(\theta) = -\sum_{t=1}^{T} \log P_\theta(w_t \mid w_1, \ldots, w_{t-1})$$

Résumé visuel

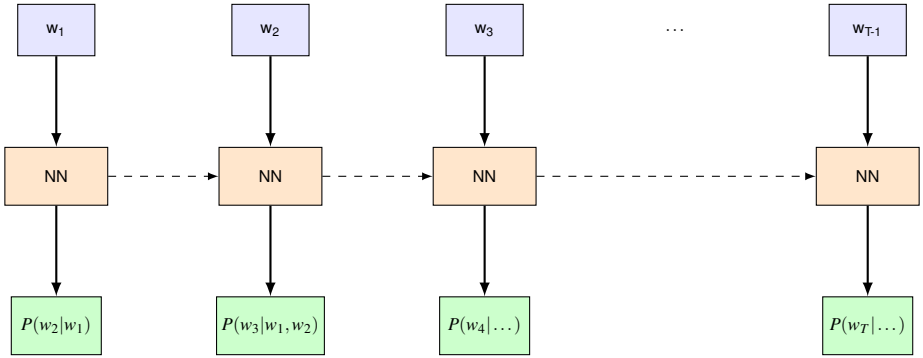


Figure – Architecture d'un modèle de langage probabiliste basé sur réseau neuronal.

- Les probabilités sont estimées par un réseau neuronal (RNN, Transformer, etc.)
- L'entraînement se fait par maximisation de la vraisemblance

Problème de la Tokenisation

- Dans les LLMs, le texte est converti en unités élémentaires : les **tokens**.
- Un token peut être un mot, une syllabe, un caractère ou une sous-unité morphologique.
- Objectifs :
  - Réduire la taille du vocabulaire.
  - Gérer les mots inconnus ou rares (*Out-of-Vocabulary*).
  - Maximiser la réutilisabilité statistique des morceaux.

Problème : Comment découper un texte efficacement tout en gardant une expressivité linguistique et une efficacité computationnelle ?

Problème de la Tokenisation

- Dans les LLMs, le texte est converti en unités élémentaires : les **tokens**.
- Un token peut être un mot, une syllabe, un caractère ou une sous-unité morphologique.
- Objectifs :
  - Réduire la taille du vocabulaire.
  - Gérer les mots inconnus ou rares (*Out-of-Vocabulary*).
  - Maximiser la réutilisabilité statistique des morceaux.

Problème : Comment découper un texte efficacement tout en gardant une expressivité linguistique et une efficacité computationnelle ?

Tokenisation par mots vs. caractères vs. sous-mots

Tokenisation par mots

- Facile à interpréter.
- Taille de vocabulaire énorme.
- Problèmes avec les mots rares ou inconnus.

Tokenisation par caractères

- Très petit vocabulaire.
- Longues séquences.
- Perte de structure linguistique.

Tokenisation par sous-mots (subwords)

- Compromis entre expressivité et efficacité.
- Basée sur la fréquence des co-occurrences.
- Utilisée par les LLMs modernes : GPT, BERT, T5.

Les sous-mots permettent d'équilibrer couverture lexicale et généralisation.

Tokenisation par mots vs. caractères vs. sous-mots

Tokenisation par mots

- Facile à interpréter.
- Taille de vocabulaire énorme.
- Problèmes avec les mots rares ou inconnus.

Tokenisation par caractères

- Très petit vocabulaire.
- Longues séquences.
- Perte de structure linguistique.

Tokenisation par sous-mots (subwords)

- Compromis entre expressivité et efficacité.
- Basée sur la fréquence des co-occurrences.
- Utilisée par les LLMs modernes : GPT, BERT, T5.

Les sous-mots permettent d'équilibrer couverture lexicale et généralisation.

Byte Pair Encoding (BPE)

- Méthode introduite initialement pour la compression.
- Appliquée aux séquences de caractères pour apprendre les sous-mots.
- Algorithme :
  1. Compter les paires adjacentes les plus fréquentes dans le corpus.
  2. Fusionner cette paire en un nouveau symbole.
  3. Répéter jusqu'à une taille de vocabulaire fixée.

Exemple :

l o w l o w e r n e w e s t w i d e s t

$\Rightarrow$ fusion de l o $\rightarrow$ lo, etc.

Byte Pair Encoding (BPE)

- Méthode introduite initialement pour la compression.
- Appliquée aux séquences de caractères pour apprendre les sous-mots.
- Algorithme :
  1. Compter les paires adjacentes les plus fréquentes dans le corpus.
  2. Fusionner cette paire en un nouveau symbole.
  3. Répéter jusqu'à une taille de vocabulaire fixée.

Exemple :

`l o w  l o w e r  n e w e s t  w i d e s t`

$\Rightarrow$ fusion de `l o` $\rightarrow$ `lo`, etc.

WordPiece : Amélioration de BPE

- Utilisé dans BERT et autres modèles Google.
- Objectif : Maximiser la probabilité jointe des tokens.
- Algorithme :
  1. Commencer par un vocabulaire initial (caractères).
  2. Ajouter itérativement le token qui augmente le plus la vraisemblance du corpus.
- Plus lente que BPE mais statistiquement plus cohérente.

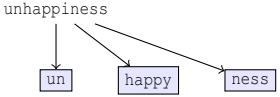
Fonction objective : maximiser $P(\text{corpus}|\text{vocabulaire})$

WordPiece : Amélioration de BPE

- Utilisé dans BERT et autres modèles Google.
- Objectif : Maximiser la probabilité jointe des tokens.
- Algorithme :
  1. Commencer par un vocabulaire initial (caractères).
  2. Ajouter itérativement le token qui augmente le plus la vraisemblance du corpus.
- Plus lente que BPE mais statistiquement plus cohérente.

Fonction objective : maximiser $P(\text{corpus}|\text{vocabulaire})$

Visualisation : Exemple de Tokenisation Subword



Découpe par BPE ou WordPiece

Impact sur les modèles LLM

- La tokenisation affecte :
  - La longueur des séquences d'entrée.
  - La couverture linguistique.
  - La capacité de généralisation du modèle.
- Choix du vocabulaire $\rightarrow$ compromis entre complexité et expressivité.
- Exemples :
  - GPT utilise un vocabulaire de $\sim$ 50k subwords (BPE).
  - BERT utilise WordPiece avec $\sim$ 30k tokens.

Conséquence : Le succès des LLMs dépend grandement de la stratégie de tokenisation.

Impact sur les modèles LLM

- La tokenisation affecte :
  - La longueur des séquences d'entrée.
  - La couverture linguistique.
  - La capacité de généralisation du modèle.
- Choix du vocabulaire $\rightarrow$ compromis entre complexité et expressivité.
- Exemples :
  - GPT utilise un vocabulaire de $\sim$ 50k subwords (BPE).
  - BERT utilise WordPiece avec $\sim$ 30k tokens.

Conséquence : Le succès des LLMs dépend grandement de la stratégie de tokenisation.

Apprentissage auto-supervisé

- L'auto-supervision consiste à créer automatiquement des labels à partir des données elles-mêmes.
- Très utile dans les modèles de langage : aucun besoin d'annotations humaines coûteuses.
- Deux paradigmes majeurs :
  - **Prédiction causale** : prédire le prochain token à partir du contexte passé.
  - **Prédiction masquée** : prédire les tokens manquants dans une séquence.

Ces stratégies d'entraînement sont à la base des grands modèles comme GPT et BERT.

Apprentissage auto-supervisé

- L'auto-supervision consiste à créer automatiquement des labels à partir des données elles-mêmes.
- Très utile dans les modèles de langage : aucun besoin d'annotations humaines coûteuses.
- Deux paradigmes majeurs :
  - **Prédiction causale** : prédire le prochain token à partir du contexte passé.
  - **Prédiction masquée** : prédire les tokens manquants dans une séquence.

Ces stratégies d'entraînement sont à la base des grands modèles comme GPT et BERT.

Apprentissage Auto-Régressif (Causal LM)

- On modélise la probabilité jointe d'une séquence :

$$P(x_1, \ldots, x_T) = \prod_{t=1}^{T} P(x_t \mid x_1, \ldots, x_{t-1})$$

- L'entraînement consiste à prédire x_t à chaque pas à partir de $x_{<t}$.
- C'est le paradigme utilisé dans :
  - GPT-1, GPT-2, GPT-3, GPT-4.
  - Les modèles de génération de texte.

Avantage : génération fluide.
Inconvénient : pas de contexte futur, entraînement unidirectionnel.

Apprentissage Auto-Régressif (Causal LM)

- On modélise la probabilité jointe d'une séquence :

$$P(x_1, \ldots, x_T) = \prod_{t=1}^{T} P(x_t \mid x_1, \ldots, x_{t-1})$$

- L'entraînement consiste à prédire x_t à chaque pas à partir de $x_{<t}$.
- C'est le paradigme utilisé dans :
  - GPT-1, GPT-2, GPT-3, GPT-4.
  - Les modèles de génération de texte.

Avantage : génération fluide.
Inconvénient : pas de contexte futur, entraînement unidirectionnel.

Apprentissage Masqué (Masked LM)

- Paradigme introduit par BERT :

  Remplacer des tokens par un symbole spécial `[MASK]`.

- Le modèle doit prédire les tokens manquants à partir du contexte gauche et droit :

$$P(x_i \mid x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_T)$$

- En pratique :
  - On masque aléatoirement 15% des tokens.
  - 80% sont remplacés par `[MASK]`, 10% par un autre mot, 10% inchangés.

Avantage : contexte bidirectionnel.
Inconvénient : incohérence entre entraînement (avec `[MASK]`) et inférence (sans `[MASK]`).

Apprentissage Masqué (Masked LM)

- Paradigme introduit par BERT :

  Remplacer des tokens par un symbole spécial `[MASK]`.

- Le modèle doit prédire les tokens manquants à partir du contexte gauche et droit :

$$P(x_i \mid x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_T)$$

- En pratique :
  - On masque aléatoirement 15% des tokens.
  - 80% sont remplacés par `[MASK]`, 10% par un autre mot, 10% inchangés.

Avantage : contexte bidirectionnel.
Inconvénient : incohérence entre entraînement (avec `[MASK]`) et inférence (sans `[MASK]`).

Comparaison : Causal vs Masked LM

Auto-régressif (Causal LM)

- Unidirectionnel
- Génération naturelle
- Pas de `[MASK]` en entrée

Masked LM

- Bidirectionnel
- Bonne représentation du contexte
- Pas adapté à la génération

Synthèse : *Causal LM* pour la génération, *Masked LM* pour l'encodage profond.

Comparaison : Causal vs Masked LM

Auto-régressif (Causal LM)

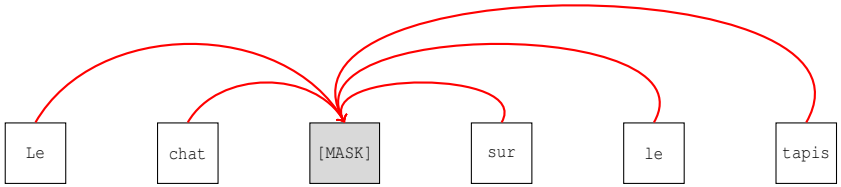
- Unidirectionnel
- Génération naturelle
- Pas de `[MASK]` en entrée

Masked LM

- Bidirectionnel
- Bonne représentation du contexte
- Pas adapté à la génération

Synthèse : *Causal LM* pour la génération, *Masked LM* pour l'encodage profond.

Visualisation : Masquage de tokens



Prédire `dort` à partir du contexte

Optimisation à grande échelle

- Les modèles de type Transformer comptent souvent des milliards de paramètres.
- L'optimisation efficace est donc critique :
  - pour la vitesse de convergence,
  - pour la stabilité de l'entraînement,
  - pour la généralisation du modèle.
- Trois leviers principaux :
  - Algorithmes d'optimisation (ex : Adam)
  - Programmation du taux d'apprentissage (warm-up, decay)
  - Régularisation (Dropout, Weight Decay, Label smoothing)

Un entraînement réussi repose sur la synergie entre ces techniques.

Optimisation à grande échelle

- Les modèles de type Transformer comptent souvent des milliards de paramètres.
- L'optimisation efficace est donc critique :
  - pour la vitesse de convergence,
  - pour la stabilité de l'entraînement,
  - pour la généralisation du modèle.
- Trois leviers principaux :
  - Algorithmes d'optimisation (ex : Adam)
  - Programmation du taux d'apprentissage (warm-up, decay)
  - Régularisation (Dropout, Weight Decay, Label smoothing)

Un entraînement réussi repose sur la synergie entre ces techniques.

Algorithme Adam (Adaptive Moment Estimation)

- Combinaison de :
  - **Momentum** : moyenne mobile du gradient
  - **RMSProp** : moyenne mobile du carré des gradients
- À chaque pas t :

$$g_t = \nabla_\theta \mathcal{L}_t(\theta_{t-1})$$
$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$

- Correction des biais :

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

- Mise à jour :

$$\theta_t = \theta_{t-1} - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon}$$

Algorithme Adam (Adaptive Moment Estimation)

- Combinaison de :
  - **Momentum** : moyenne mobile du gradient
  - **RMSProp** : moyenne mobile du carré des gradients
- À chaque pas t :

$$
\begin{aligned}
g_t &= \nabla_\theta \mathcal{L}_t(\theta_{t-1}) \\
m_t &= \beta_1 m_{t-1} + (1-\beta_1) g_t \\
v_t &= \beta_2 v_{t-1} + (1-\beta_2) g_t^2
\end{aligned}
$$

- Correction des biais :

$$
\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1-\beta_2^t}
$$

- Mise à jour :

$$
\theta_t = \theta_{t-1} - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon}
$$

Warm-up et décroissance du taux d'apprentissage

- Idée : démarrer avec un petit taux d'apprentissage η pour éviter les instabilités.
- **Warm-up** : augmentation linéaire du taux d'apprentissage pendant les N_{warm} premières étapes :

$$\eta_t = \eta_{\max} \cdot \frac{t}{N_{\text{warm}}}, \quad t \leq N_{\text{warm}}$$

- Ensuite, décroissance :
  - *Inverse Square Root Decay (Transformers)* :

$$\eta_t = \frac{\eta_{\max}}{\sqrt{t}}, \quad t > N_{\text{warm}}$$

  - *Cosine Annealing*, *Exponential Decay* : selon le cas.
- Pourquoi ? Trop grand η au départ $\Rightarrow$ explosion du gradient.

Warm-up et décroissance du taux d'apprentissage

- Idée : démarrer avec un petit taux d'apprentissage η pour éviter les instabilités.
- **Warm-up** : augmentation linéaire du taux d'apprentissage pendant les N_{warm} premières étapes :

$$\eta_t = \eta_{\max} \cdot \frac{t}{N_{\text{warm}}}, \quad t \leq N_{\text{warm}}$$

- Ensuite, décroissance :
  - *Inverse Square Root Decay (Transformers)* :

$$\eta_t = \frac{\eta_{\max}}{\sqrt{t}}, \quad t > N_{\text{warm}}$$

  - *Cosine Annealing*, *Exponential Decay* : selon le cas.
- Pourquoi ? Trop grand η au départ $\Rightarrow$ explosion du gradient.

Régularisation dans les Transformers

- **Dropout** : désactive aléatoirement une fraction des neurones à l'entraînement.

$$y = x \odot \text{Bernoulli}(p)$$

- **Weight Decay** : pénalise les grands poids dans la fonction de perte.

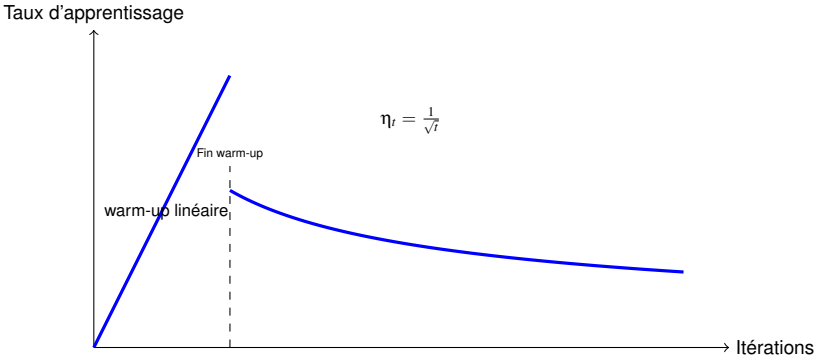
$$\mathcal{L}_{\text{totale}} = \mathcal{L}_{\text{cross-entropy}} + \lambda \|\theta\|_2^2$$

- **Label Smoothing** : évite les prédictions trop confiantes.

$$q_{\text{smoothed}}(y) = (1 - \varepsilon) \cdot \delta(y = y^*) + \frac{\varepsilon}{K}$$

- Ces techniques améliorent la robustesse et la généralisation.

Courbe typique de taux d'apprentissage



Alignement des modèles de langage par RLHF

- Les modèles préentraînés (GPT, T5, etc.) apprennent à prédire du texte plausible, mais pas nécessairement souhaitable.
- L'**alignement** vise à adapter les comportements du modèle aux préférences humaines :
  - **Éthique, utilité, sécurité, pertinence**.
- Méthodologie standard : **Reinforcement Learning from Human Feedback (RLHF)** :
  1. Préentraînement sur données massives (objectif de langage)
  2. Apprentissage d'un **modèle de récompense** à partir de notations humaines
  3. **Optimisation par PPO** (Proximal Policy Optimization)
- Référence : InstructGPT (Ouyang et al., 2022)

Alignement des modèles de langage par RLHF

- Les modèles préentraînés (GPT, T5, etc.) apprennent à prédire du texte plausible, mais pas nécessairement souhaitable.
- L'**alignement** vise à adapter les comportements du modèle aux préférences humaines :
  - **Éthique, utilité, sécurité, pertinence**.
- Méthodologie standard : **Reinforcement Learning from Human Feedback (RLHF)** :
  1. Préentraînement sur données massives (objectif de langage)
  2. Apprentissage d'un **modèle de récompense** à partir de notations humaines
  3. **Optimisation par PPO** (Proximal Policy Optimization)
- Référence : InstructGPT (Ouyang et al., 2022)

Formulation du RLHF

- Soit x une requête, et y une réponse générée par le modèle $f_\theta(y|x)$.
- On entraîne un modèle de récompense $r_\phi(x,y)$ à approximer les préférences humaines :

$$\max_\phi \sum_i \log \sigma \left(r_\phi(x_i, y_i^+) - r_\phi(x_i, y_i^-) \right)$$

- On définit ensuite une politique $f_\theta(y|x)$ optimisée par PPO pour maximiser la récompense :

$$\max_\theta \mathbb{E}_{y \sim f_\theta(\cdot|x)} [r_\phi(x,y)] - \beta \cdot \mathrm{KL}(f_\theta \| f_{\text{base}})$$

- PPO garantit une mise à jour conservatrice pour éviter les dérives :

$$L^{\text{PPO}}(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta) \hat{A}_t, \text{clip}(r_t(\theta), 1-\varepsilon, 1+\varepsilon) \hat{A}_t \right) \right]$$

Formulation du RLHF

- Soit x une requête, et y une réponse générée par le modèle $f_\theta(y|x)$.
- On entraîne un modèle de récompense $r_\phi(x,y)$ à approximer les préférences humaines :

$$\max_\phi \sum_i \log \sigma\left(r_\phi(x_i, y_i^+) - r_\phi(x_i, y_i^-)\right)$$

- On définit ensuite une politique $f_\theta(y|x)$ optimisée par PPO pour maximiser la récompense :

$$\max_\theta \mathbb{E}_{y\sim f_\theta(\cdot|x)}[r_\phi(x,y)] - \beta \cdot \mathrm{KL}(f_\theta \| f_{\text{base}})$$

- PPO garantit une mise à jour conservatrice pour éviter les dérives :

$$L^{\text{PPO}}(\theta) = \mathbb{E}_t\left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t\right)\right]$$

Architectures : BERT, GPT, T5

BERT

- Encodeur uniquement
- Apprentissage bi-directionnel
- Masquage de tokens (Masked LM)
- Applications : classification, QA

GPT

- Décodeur uniquement
- Apprentissage causale unidirectionnel
- Génération autoregressive

T5 (Text-to-Text Transfer Transformer)

- Encodeur-décodeur complet
- Tous les problèmes = transformation texte texte
- Préentraînement par Corrupted Span Prediction

Architectures : BERT, GPT, T5

BERT

- Encodeur uniquement
- Apprentissage bi-directionnel
- Masquage de tokens (Masked LM)
- Applications : classification, QA

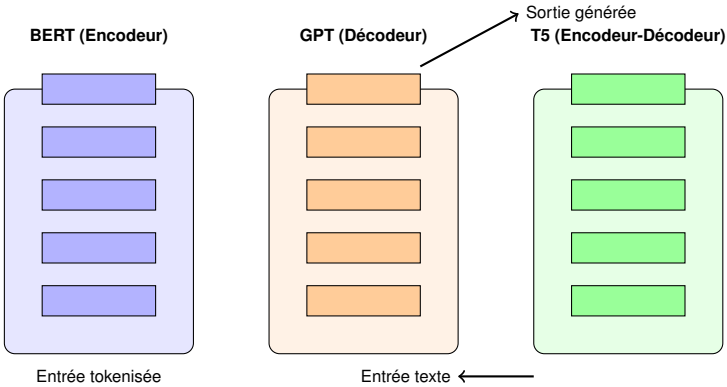
GPT

- Décodeur uniquement
- Apprentissage causale unidirectionnel
- Génération autoregressive

T5 (Text-to-Text Transfer Transformer)

- Encodeur-décodeur complet
- Tous les problèmes = transformation texte texte
- Préentraînement par Corrupted Span Prediction

Comparaison architecturale des modèles Transformer



Légende

- **BERT** : encodeur bidirectionnel optimisé pour la compréhension (classification, QA).
- **GPT** : décodeur unidirectionnel pour la génération de texte.
- **T5** : architecture encodeur-décodeur unifiée pour toutes les tâches sous forme textetexte.

Conclusion du Module 3

- Modèles de langage modernes combinent :
  - Apprentissage auto-supervisé massif
  - Architectures Transformer spécialisées
  - Optimisation à très grande échelle
  - Alignement via feedback humain (RLHF)
- GPT, BERT, T5 : chacun avec une philosophie propre
- RLHF est aujourd'hui essentiel pour des LLM sûrs et utiles.

Prochaine étape

Module 4 : Préentraînement à grande échelle et fine-tuning spécialisé.

Conclusion du Module 3

- Modèles de langage modernes combinent :
  - Apprentissage auto-supervisé massif
  - Architectures Transformer spécialisées
  - Optimisation à très grande échelle
  - Alignement via feedback humain (RLHF)
- GPT, BERT, T5 : chacun avec une philosophie propre
- RLHF est aujourd'hui essentiel pour des LLM sûrs et utiles.

Prochaine étape

Module 4 : Préentraînement à grande échelle et fine-tuning spécialisé.