# Module 4 – Les modèles de langage (petits et grands)

Ce module présente l'évolution des modèles de langage, des approches classiques aux modèles modernes fondés sur les transformers. Il montre comment ces systèmes ont permis de passer de la simple prévision statistique des mots à des conversations complexes avec les machines.

## 1. Introduction aux modèles de langage

Un **modèle de langage** est un système qui attribue une probabilité à une séquence de mots. Son objectif principal est de prévoir le mot suivant dans un texte, ce qui est essentiel pour des applications comme la reconnaissance vocale, la traduction automatique ou la recherche d'information.

Les approches classiques utilisaient surtout des modèles statistiques appelés **n-grammes**. Dans un modèle n-gramme, la probabilité d'un mot dépend des (n-1) mots précédents. Par exemple, un trigramme (n=3) prend en compte les deux mots qui précèdent chaque mot à prédire. Ces modèles avaient deux limites majeures :

- Ils nécessitaient des corpus très volumineux pour estimer les probabilités.
- Ils ne représentaient pas les liens de sens entre les mots (par exemple, «chien» et «chiens» étaient traités comme totalement distincts).

## 2. Premiers réseaux de neurones appliqués au langage et représentations vectorielles

Avec l'essor de l'apprentissage machine et des réseaux neuronaux, des chercheurs ont cherché à dépasser ces limites. Passer d'un modèle purement statistique à un modèle **neuronal** a permis de représenter les mots sous une forme plus compacte et plus riche en information.

C'est dans ce contexte qu'ont été introduites les **représentations vectorielles des mots**, connues sous le nom de **word embeddings**. Le modèle Word2Vec, proposé par Mikolov et ses collègues en 2013, a permis d'apprendre un vecteur dense (par exemple, de 300 dimensions) pour chaque mot. Ces vecteurs capturent les relations sémantiques et contextuelles. On pouvait ainsi observer des analogies célèbres :

roi - homme + femme  reine

Ces représentations ont permis de mieux généraliser aux mots rares ou nouveaux et ont constitué une avancée majeure.

## 3. Réseaux neuronaux récurrents et modèles séquence-à-séquence

Pour dépasser la rigidité des n-grammes, les **réseaux neuronaux récurrents** (RNN) ont été introduits. Un RNN traite la séquence mot par mot, en maintenant une mémoire interne qui encode le contexte précédent. Cela permet, en théorie, de prendre en compte des dépendances longues.

Cependant, les RNN simples présentaient des difficultés :

- Les gradients disparaissaient au fil des étapes (**problème du gradient qui s'annule**), rendant difficile l'apprentissage des relations à long terme.
- L'entraînement était lent, car les séquences devaient être traitées une étape après l'autre.

Pour résoudre ces problèmes, les architectures **LSTM** (Long Short-Term Memory) et **GRU** (Gated Recurrent Unit) ont été développées. Elles utilisent des mécanismes appelés **portes** qui régulent l'information qui circule dans la mémoire.

Ces modèles ont permis de créer les architectures **séquence-à-séquence** (Seq2Seq), capables de transformer une séquence en une autre (par exemple, traduire une phrase d'une langue à une autre).

## 4. L'avènement des Transformers

En 2017, Vaswani et ses collègues publient l'article «Attention Is All You Need», qui marque une rupture profonde avec les approches précédentes. Le **Transformer** abandonne la récurrence et repose uniquement sur un mécanisme d'*auto-attention*.

### 4.1 Le principe de l'auto-attention

L'auto-attention permet à chaque mot de la séquence de tenir compte de tous les autres mots en parallèle. Concrètement :

- Chaque mot est transformé en trois vecteurs : **requête** (Query), **clé** (Key) et **valeur** (Value).
- Pour un mot donné, on compare sa requête avec toutes les clés des autres mots, ce qui produit des scores d'attention.
- Ces scores sont normalisés avec une fonction softmax, puis servent à pondérer la combinaison des valeurs.

Ainsi, chaque mot obtient une représentation enrichie du contexte global, sans avoir besoin de parcourir la séquence dans l'ordre.

### 4.2 Les principaux composants d'un Transformer

Un bloc Transformer comprend plusieurs éléments :

- Un mécanisme d'*auto-attention multi-têtes* (chaque tête apprend à repérer différents types de relations).
- Un réseau de neurones **feedforward** appliqué indépendamment à chaque position.
- Des normalisations de couche (**Layer Normalization**) et des connexions résiduelles qui stabilisent et accélèrent l'apprentissage.

### 4.3 Les embeddings et la position

Contrairement aux RNN, le Transformer ne possède pas naturellement la notion d'ordre des mots. Il faut donc ajouter des **embeddings positionnels** qui donnent à chaque mot une information sur sa place dans la séquence. Ces embeddings peuvent être calculés avec des fonctions sinusoïdales ou appris directement pendant l'entraînement.

### 4.4 Traitement parallèle et performance

L'un des avantages majeurs du Transformer est que toutes les positions peuvent être traitées en même temps (**parallélisme**). Cette caractéristique rend l'entraînement beaucoup plus rapide et plus efficace que celui des réseaux récurrents.

Ces propriétés ont permis de concevoir des modèles très puissants, comme BERT, GPT, T5 et leurs successeurs.

## 5. Des modèles prédictifs aux assistants conversationnels

Les premiers Transformers servaient surtout à prédire le mot suivant ou à compléter un texte. Pour passer à des modèles capables de répondre à des instructions et de dialoguer avec les humains, il a fallu plusieurs étapes supplémentaires.

### 5.1 Les modèles InstructGPT et ChatGPT

Ces modèles sont obtenus par un **ajustement fin** (fine-tuning) des Transformers de base sur des exemples de conversations et des instructions annotées par des humains. Ce processus permet d'orienter le comportement du modèle vers des réponses plus pertinentes et utiles.

### 5.2 L'apprentissage par renforcement avec retour humain (RLHF)

Le **Reinforcement Learning from Human Feedback** (apprentissage par renforcement avec retour humain) se déroule en plusieurs étapes :

1. Le modèle produit différentes réponses à une même question.
2. Des évaluateurs humains les classent de la meilleure à la moins bonne.
3. Ces classements servent à entraîner un **modèle de récompense** qui estime la qualité des réponses.
4. Le modèle principal est ensuite optimisé par renforcement (souvent avec un algorithme appelé PPO, Proximal Policy Optimization) pour maximiser cette récompense.

Ce processus permet d'aligner les modèles avec des critères humains de politesse, de sécurité et de pertinence.

## 6. Conclusion

Les modèles de langage ont évolué d'outils statistiques simples à des systèmes capables de mener des conversations détaillées et nuancées. Comprendre leur histoire permet de mesurer à la fois leur puissance et les défis qu'ils posent.