REPUBLIQUE DU SENEGAL



Un Peuple-Un But-Une Foi

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR, DE LA RECHERCHE ET DE L'INNOVATION

UNIVERSITE GASTON BERGER DE SAINT-LOUIS



U.F.R DE SCIENCES APPLIQUEES ET DE TECHNOLOGIE

DEPARTEMENT D'INFORMATIQUE

MEMOIRE DE FIN DE CYCLE EN VUE DE L'OBTENTION DU DIPLOME DE MASTER

SPECIALITE : RESEAUX SECURITE ET SYSTEMES DISTRIBUES

THEME : Les attaques possibles sur l'intelligence artificielle générative

Présenté par :

M. Mohamed Lamine COULIBALY

Sous la direction de :

Pr. Chérif DIALLO

Jury :

Président : Pr Maissa MBAYE

Examinateur : Dr Oumar DIOP

Examinateur : Dr Cheikh M. Bamba DIONE

Encadreur : Pr Chérif DIALLO

Année Académique 2022/2023

Remerciements

Je remercie le Tout Puissant, le Tout Miséricordieux, qui m'a donné grâce et bénédiction pour mener à bout ce projet.

Il m'est agréable de m'acquitter d'une dette de reconnaissance auprès de toutes les personnes dont l'intervention au cours de ce mémoire a favorisé son aboutissement.

Je tiens à exprimer ma profonde gratitude et mes sincères remerciements à :

- Ma merveilleuse famille **COULIBALY** de la Côte d'Ivoire et du Mali ;
- **Mon Encadrant, Pr. Chérif DIALLO**, dont les conseils éclairés, l'expertise et la disponibilité ont grandement contribué à la réussite de mon travail. Sa guidance précieuse a été un facteur déterminant dans l'aboutissement de ce projet ;
- **Dr. El hadji Serigne Mamour DIOP,** pour ses conseils et son soutien ;
- **Dr. Fatou Kamara SANGARE,** une mère qui m'a soutenu, aidé, accueilli en tant que son fils, je vous souhaite tout le bonheur dans votre famille ;
- **Maitre De Conférence Daouda SANGARE,** un père qui m'a soutenu, aidé, accueilli en tant que son fils, je vous souhaite tout le bonheur dans votre famille ;
- **Pr. Cheikh BA**, Chef de la section Informatique de l'UFR SAT ;
- **Dr. Cheikh M. Bamba DIONE**, coordinateur du master informatique de l'UFR SAT ;
- Tous les enseignants de l'UFR SAT ;
- Toute ma promotion, les étudiants du Master 2 GDIL & R2SD ;
- Toutes les personnes, qui de près ou de loin, ont contribué à la réalisation de ce travail.
- Tous les membres de la CINERI.

Dédicaces

Je dédie ce modeste travail :

A mes parents, pour votre amour inconditionnel, pour tous vos efforts et sacrifices. Pour votre soutien, vos conseils, votre confiance et surtout vos prières sans lesquelles je ne serai jamais arrivé où j'en suis. Merci, mille fois merci.

A mes oncles et mes Tantes, vous avez toujours été présents pour moi, en m'encourageant dans mes études. Pour votre sens de la famille.

A mes frères, sœurs, cousins, pour votre amour, vos encouragements et votre implication. Puisse Allah Le Tout Puissant continuer à bénir la famille, puisse-t-il vous garder à ses côtés et vous garder dans toutes vos entreprises. Je vous aime tous.

A ma promotion P27 (Youssoupha, El Hadji Mamadou Diémé, Mariama Dramé, Thiane Ndiaye, Baba Travaré, Gora Thiaw, Cheikh Tidiane Ba, Amadou Diao, Khadim Ngom, Khadim Daffé, Cheikh Sidy Moctar Sène, Habsatou Thiam, Mariama Ndiaye, Mohamed Diouf, El Hadji Mbaye Thiam, Djibi Ndiaye, Lamine Gueye, Saly Samaké, Assetou Gassama, Mame Anta Bathily, Marie Diouf, Baba Top, Garba Cheibou Abakar, Hassane Chidi, Alassane).

A tous mes frères et sœurs de la communauté Malienne à Saint-Louis, je vous souhaite le bonheur, la santé, la prospérité et qu'Allah vous assiste dans toutes vos épreuves. Merci pour tout, je vous aime.

Résumé

Dans ce mémoire de fin d'étude, intitulé « ***Les attaques possibles sur l'intelligence artificielle générative*** », explore les vulnérabilités et menaces associées aux modèles d'IA générative (tels que les LLM, les diffusions, les GANs, les VAEs, etc.), notamment les Large Langage Modèles (LLM) comme GPT. Il effectue une analyse sur le développement rapide de l'IA générative, sur son impact croissant dans divers domaines et tout en mettant en lumière les risques liés à son utilisation tels que les attaques par injection de prompt, l'empoisonnement des données, l'extraction de modèles, etc.

Ce mémoire commence par une étude approfondie de l'évolution de l'intelligence artificielle, en passant par l'apprentissage automatique et l'apprentissage profond, avant de se concentrer sur les architectures des modèles génératifs. Il détaille le fonctionnement des modèles d'IA génératives tels que : GAN (Generative Adversarial Networks), des Autoencodeurs Variationnels (VAE), des modèles de diffusion, Hybrides, etc. qui permettent de générer du contenu textuel, visuel et sonore avec une qualité impressionnante. Ce mémoire met en lumière l'importance du prompt engineering en présentant à la fois des techniques de base (Zero-shot, One-shot, Few-shot) et des approches avancées (Chain-Of-Thought, Meta Prompting, RAG, etc.) pour renforcer la robustesse des modèles d'IA génératives. Il présente l'architecture des modèles d'IA générative, c'est-à-dire les Transformers, reposant sur un mécanisme d'auto-attention. Ce mécanisme traite les mots en relation avec tous les autres mots d'une phrase, à contrario aux méthodes traditionnelles qui analysent pas à pas.

Enfin, le mémoire met l'accent sur les attaques et défenses liées à l'IA générative. Il classifie différentes techniques d'attaques (modification malveillante des données d'entraînement, détournement des réponses de l'IA, extraction de connaissances du modèle, etc.) et propose des stratégies d'atténuation pour protéger les systèmes basés sur l'IA générative. Un cas pratique est également présenté, illustrant une attaque par injection de prompt sur un Chatbot conversationnel, avec des solutions techniques pour détecter et atténuer ces menaces. De plus, ce mémoire insiste sur la nécessité d'un encadrement rigoureux et d'une approche critique dans l'utilisation de ces technologies afin de prévenir les usages malveillants, tout en encourageant la recherche de solutions innovantes pour sécuriser l'IA générative dans des environnements de production.

Abstract

In this final thesis, titled ***'Potential Attacks on Generative Artificial Intelligence***,' explores the vulnerabilities and threats associated with generative AI models (such as LLMs, diffusion models, GANs, VAEs, etc.), particularly Large Language Models (LLMs) like GPT. It analyzes the rapid development of generative AI, its growing impact across various fields, while highlighting the risks associated with its use, such as prompt injection attacks, data poisoning, and model extraction, among others.

This thesis begins with a comprehensive study of artificial intelligence evolution, covering machine learning and deep learning, before focusing on generative model architectures. It details the functioning of generative AI models such as GANs (Generative Adversarial Networks), Variational Autoencoders (VAE), diffusion models, hybrid models, etc., which can generate textual, visual, and audio content with impressive quality. This thesis highlights the importance of prompt engineering by presenting both basic techniques (Zero-shot, One-shot, Few-shot) and advanced approaches (Chain-Of-Thought, Meta Prompting, RAG, etc.) to strengthen the robustness of generative AI models. It presents the architecture of generative AI models, specifically Transformers, based on a self-attention mechanism. This mechanism processes words in relation to all other words in a sentence, contrary to traditional methods that analyze step by step.

Finally, the thesis emphasizes attacks and defenses related to generative AI. It classifies different attack techniques (malicious modification of training data, AI response manipulation, model knowledge extraction, etc.) and proposes mitigation strategies to protect systems based on generative AI. A practical case study is also presented, illustrating a prompt injection attack on a conversational chatbot, with technical solutions to detect and mitigate these threats. Moreover, this thesis emphasizes the need for rigorous framework and a critical approach in using these technologies to prevent malicious usage, while encouraging research into innovative solutions to secure generative AI in production environments.

Table des matières

Remerciements ..... **i**

Dédicaces ..... **ii**

Résumé ..... **iii**

Abstract ..... **iv**

Table des matières ..... **v**

LISTES DES FIGURES ..... **x**

LISTE DES TABLEAUX ..... **xii**

SIGLES ET ABREVIATIONS ..... **xiii**

CHAPITRE I : INTRODUCTION GENERALE **1**

I. CONTEXTE DU SUJET : 1

II. PROBLEMATIQUE : 2

III. OBJECTIF ET MÉTHODOLOGIE : 3

IV. ORGANISATION DU DOCUMENT : 4

CHAPITRE II : ETAT DE L'ART **5**

I. INTRODUCTION : 5

II. EVOLUTION DE L'INTELLIGENCE ARTIFICIELLE : 6

I. CONTEXTE : 6

II. ETUDE DE L'APPRENTISSAGE AUTOMATIQUE : 8

1. Apprentissage Supervisé 9

2. Apprentissage non Supervisé 10

3. Apprentissage Semi Supervisé 10

4. Apprentissage par Renforcement 11

III. ETUDE DE L'APPRENTISSAGE PROFOND : 12

A. LES TYPES DE MODELES DE DEEP LEARNING : 13

1. Apprentissage Profond Supervisé (Deep Supervised Learning) : 14

a. Réseaux Neuronaux Profonds (Artificial Neural Network) : 15

b. Réseaux Neuronaux Convolutifs (Convolutional Neuronal Network - CNN) : 15

c. Réseau Neuronal Récurrent (Recurrent Neural Networks - RNN) : 18

2. Apprentissage Profond Non Supervisé (Deep UnSupervised Learning) : 19

a. Auto-Encodeurs (Auto-Encoder-AE) : 19
b. Machine Boltzmann restreinte (Restricted Boltzmann Machine RBM) : 20
c. Réseau neuronal de croyance profonde (Deep Belief Neural Network -DBN) : 20
d. Réseau antagoniste génératif (Generative Adversarial Network - GAN) : 21
3. Apprentissage Profond par Renforcement (Deep Reinforcement Learning) : 21
4. Apprentissage Profond Hybrid (Hybrid Deep Learning) : 23
III. EVOLUTION DE L'IA GÉNÉRATIVE : 23
I. CONTEXTE : 23
II. MODÈLES ET APPLICATIONS GÉNÉRATIVES : 26
IV. CONCLUSION : 29
CHAPITRE III : L'INTELLIGENCE ARTIFICIELLE GÉNÉRATIVE 30
I. DÉFINITION ET CONCEPT DE BASE : 30
II. ARCHITECTURES DES MODÈLES GÉNÉRATIVES : 34
A. Auto-Encodeurs Variationnels (Variationals Autoencoders -VAE) : 35
B. Réseaux Antagonistes Génératifs (Generative Adversarial Networks - GAN) : 36
1. Réseaux Antagonistes Génératifs Conditionnels 38
2. Réseaux Antagonistes Génératifs Convolutionnels Profonds 39
3. Réseaux Antagonistes Génératifs de Wasserstein 39
C. Modèles de Diffusions (Diffusion Models - DM) : 40
1. Modèles Probabilistes de Diffusion de Débruitage 41
2. Modèles Génératifs basés sur le Score 41
3. Équations Différentielles Stochastiques 42
D. Modèles de Langages (Languages Models - LM) : 42
E. Normalisation des modèles de flux (Normalizing Flow Models) : 43
F. Modèles Hybrides (Hybrid Models - HM) : 44
1. Auto-Encodeur Antagoniste 44
2. Auto-Encodeur Variationnel Avec Réseaux Antagonistes Génératifs 45
3. Transformateur avec Réseau Convolutif Graphique 45
III. CATÉGORIES DE MODÈLES D'IA GÉNÉRATIVES : 46
A. Modèles Texte : 46
B. Modèles Images : 47
C. Modèles Code : 47
D. Modèles Audio : 47
E. Modèles Vidéos : 48
F. Modèles Multimodaux : 48

IV. CONCLUSION :.....48
CHAPITRE IV: BASE FONDAMENTALE DU PROMPT ENGINEERING 50
I. INTRODUCTION : 50
II. ARCHITECTURES :.....51
A. Paramètres d'un prompt :.....51
1. Température :.....52
2. Top P :.....58
3. Max-Tokens :.....60
4. Fréquence de Pénalité :.....64
5. Présence de Pénalité :.....65
6. Séquence d'Arrêt :.....65
B. Structure d'un prompt :.....65
1. Instruction :.....66
2. Contexte :.....66
3. Rôle :.....66
4. Sorties :.....67
5. Exemples :.....67
C. Techniques de bases de prompts :.....70
1. Zero-shot prompting :.....70
2. One-shot prompting :.....71
3. Few-shot prompting :.....71
4. Difference Entre Zero-shot, One-shot et Few-shot prompting.....72
D. Exemples de prompts :.....73
1. Résumé d'un texte (Text Summarization) :.....73
2. Extraction d'information (Information Extraction) :.....74
3. Questions – Réponses :.....75
4. Classification de Texte :.....75
5. Conversation :.....76
6. Génération de Code :.....77
III. TECHNIQUES DE PROMPTS AVANCÉS :.....78
A. Meta Prompting.....78
B. L'auto-consistance (self-consistency).....79
C. Arbre.....79
D. L'invite active (active-prompt).....80
E. La réflexion.....81

IV. METHODES DE PROMPTS : 82
A. RAG (Retrieval Augmented Generation) : 83
B. CHAINE DE PENSÉE (CHAIN-OF-THROUGHT - COT) : 84
C. ACTE DE RAISON (React – Reason act) : 85
D. DSP – (Directional Stimulus Prompting) : 86
V. CONCLUSION : 86
CHAPITRE V : LARGES LANGAGES MODÈLES – LLM 87
I. INTRODUCTION : 87
II. ARCHITECTURES DES LLM : 88
A. Transformers (Transformers) : 89
1. Entrée d'Intégration 90
2. Encodage Positionnel 90
3. Encodeur – Décodeur 90
4. Auto-Attention 91
5. Réseau Neuronal à Flux Continu 92
6. Multi – Entête Attention 92
7. Couche de Normalisation (Layer Normalization) : 92
8. Couche de Sortie (Output Layer) : 92
B. Architecture De GPT : 95
1. Intégration de données d'entrées : 97
2. Codage positionnel : 97
3. Blocs GPT/Blocs Transformers : 97
4. Fonctions linéaires et Softmax : 97
5. La Préformation : 98
6. La Mise au Point/ Fine-Tuning : 98
7. Modélisation linguistique : 98
8. Apprentissage non supervisé : 98
C. Architecture De BERT : 99
1. Le Pré-entrainement : 100
2. Le fine-tuning : 100
D. Différence entre GPT et BERT : 101
III. CONCLUSION : 104
CHAPITRE VI : ETUDES DES ATTAQUES SUR L'IA GÉNÉRATIVE 105
I. INTRODUCTION : 105

II. ETUDES DES MENACES ET VULNERABILITES :.....106
III. ETUDES DES ATTAQUES : 109
A. Attaques De TYPE1.....110
B. Attaques De TYPE2.....111
IV. ETUDES DES DEFENSES ET DES ATTENUATIONS :.....112
A. Méthodes De DEFENSES 112
B. Méthodes D'ATTENUATIONS.....113
IV. CONCLUSION :.....114
CHAPITRE VII : CAS PRATIQUE : Prompt d'Injection Direct.....115
I. CHOIX TECHNOLOGIQUES :.....115
II. ENVIRONNEMENT DE TRAVAIL :.....116
III. PHASES OFFENSIVES :.....116
A. Bibliothèques :.....116
B. Création de la base de données :.....117
C. Chargement du modèle GPT-2 :.....118
D. Injection du code malveillant :.....119
E. Interface utilisateur :.....119
IV. PHASES DEFENSIVES : 121
V. CONCLUSION :.....122
CONCLUSION GENERALE :.....123
REFERENCES.....124

LISTES DES FIGURES

FIGURE 1: DOMAINES DE IA[4]....7
FIGURE 2: MACHINE LEARNING [8]....8
FIGURE 3:DIFFERENTS ML ET D'ALGORITHMES [9]....9
FIGURE 4:REINFORCEMENT LEARNING [15]....11
FIGURE 5:RESEAUX DE NEURONES PROFONDS [1]....12
FIGURE 6: MODELE MATHEMATIQUE D'UN NEURONE ARTIFICIEL [3]....13
FIGURE 7: DEEP LEARNING CATEGORIES [19]....14
FIGURE 8: DEEP SUPERVISED LEARNING [19]....15
FIGURE 9: COMPOSANTS DE CNN [3]....16
FIGURE 10 : SCHEMA DE PROCESSUS DE CONVOLUTION [20]....17
FIGURE 11 : OPERATION DE POOLING MAX 3 × 3 SUR UNE UNITE DE CALCUL 5 × 5 [21]....18
FIGURE 12 : FONCTIONNEMENT INTERNE SIMPLE DU RNN [20]....19
FIGURE 13 : DIFFERENTS MODELES DE DEEP UNSUPERVISED LEARNING [22]....19
FIGURE 14 : DEEP REINFORCEMENT LEARNING [22]....22
FIGURE 15 : MODELES IA GENERATIVE ET ARCHITECTURES [29]....28
FIGURE 16 : CHATGPT [40]....30
FIGURE 17 : HUGGINFACECHAT [41]....31
FIGURE 18 : CLAUDE-IA [42]....31
FIGURE 19 : PHASES D'IMPLEMENTATION DE L'IA GENERATIVE [29]....32
FIGURE 20 : VARIANTIONALS AUTOENCODERS-VAE [29]....36
FIGURE 21 : STRUCTURE TYPIQUE DES RESEAUX ANTAGONISTES GENERATIFS DE GAN [29]....37
FIGURE 22 : RESEAUX ANTAGONISTE [29]....38
FIGURE 23 : STRUCTURE TYPIQUE DE DM [29]....40
FIGURE 24 : ARCHITECTURE DES RESEAUX NEURONAUX RECURRENTS [29]....43
FIGURE 25: ARCHITECTURES DE TRANSFORMERS [32]....90
FIGURE 26 : ARCHITECTURE DE SELF-ATTENTION [32]....91
FIGURE 27: ARCHITECTURE DE MULTI-HEAD [32]....92
FIGURE 28: TRANSFORMER VS TOKENFORMER [48]....93
FIGURE 29: ARCHITECTURE DE TOKENFORMER [48]....94
FIGURE 30: ARCHITECTURE TYPIQUE DE NORMALISATION DE MODELES [26]....44
FIGURE 31: TYPIQUE ARCHITECTURE AAE [26]....45
FIGURE 32: TAXONOMIE DES IA GENERATIVES (PLUS UTILISEES) [38]....46
FIGURE 33: CHAT-OPENIA-PARAMETRE [53]....51
FIGURE 34: MODELE-GPT2 --- PARA-TEMPERATURE....52

FIGURE 35: GPT2-FONCTION_GENERATE-1 53
FIGURE 36: GPT2-FONCTION_GENERATE-2 53
FIGURE 37: GPT2- RESULTATS 53
FIGURE 38: CODE-SENTIMENT-ANALYTIQUE-1 55
FIGURE 39:CODE-SENTIMENT-ANALYTIQUE-2 56
FIGURE 40: PARAMETRE – TOP P- GPT2 58
FIGURE 41: RESULTATS-TOP P-GPT2 59
FIGURE 42: GPT4O-TOKEN 61
FIGURE 43: TOKEN [IDS]-VECTEUR-LLAMA-3B 61
FIGURE 44: TOKEN [IDS]-VECTEUR-GPT4 62
FIGURE 45: TOKEN[IDS]-VECTEUR-BERT 62
FIGURE 46: TOKEN[IDS]-VECTEUR-GPT2 63
FIGURE 47: STRUCTURE _PROMPT_EXEMPLE-1 68
FIGURE 48: STRUCTURE_PROMPT_EXEMPLE-2 69
FIGURE 49 : ZERO-SHOT PROMPTING 70
FIGURE 50: ONE-SHOT PROMPTING 71
FIGURE 51: FEW-SHOT PROMPTING 72
FIGURE 52: PROMPTING-TEXTE-1 [56] 73
FIGURE 53: PROMPTING-TEXTE-2 [56] 74
FIGURE 54: EXTRACTION-TEXTE [56] 74
FIGURE 55:QUESTIONS-REPONSES-PROMPTING [56] 75
FIGURE 56:CLASSIFICATION-PROMPTING 75
FIGURE 57:CLASSIFICATION-PROMPTING-2 76
FIGURE 58: CONVERSATION-PROMPTING 76
FIGURE 59: GENERATION-CODE- PROMPTING 77
FIGURE 60: GENERATION-CODE-PROMPTING 77
FIGURE 61: TREE-THROUGHT [60] 80
FIGURE 62: ACTIVE PROMPTING [62] 81
FIGURE 63: REFLEXION PROMPTING [63] 82
FIGURE 64: CHAIN-OF-THROUGHT PROMPTING [64] 84
FIGURE 65 : ARCHITECTURE – GPT [69] 96
FIGURE 66: BERT-BASE & BERT-LARGE [37] 99
FIGURE 67 : SCHEMA D'ATTAQUE 116
FIGURE 68 : BIBLIOTHEQUES 117
FIGURE 69: BASE DE DONNEES SQLITE 117
FIGURE 70: BD: TABLE_PRODUITS 117
FIGURE 71: BD : TABLES_USERS_TABLES 118

FIGURE 72 : ACCES AU MODELE GPT-2 118
FIGURE 73 : CODE INJECTION 119
FIGURE 74 : INTERFACE AVEC STREAMLIT 119
FIGURE 75: INTERFACE 120
FIGURE 76: SCENE D'ATTAQUE 120
FIGURE 77 : LISTES DES ACTIONS 121
FIGURE 78: PHASE DE DEFENSE 122

LISTE DES TABLEAUX

TABLE 1: IA GENERATIVE [38], [39] 25
TABLE 2 : MODELES ET APPLICATIONS DE L'IA GENERATIVE [38] 27
TABLE 3 : OUTILS D'IA GENERATIVE [39] 28
TABLE 4:COMPOSANTS DE L'ARCHITECTURE ET METHODES D'APPRENTISSAGE UTILISES DANS LES MODELES GENERATIFS D'IA [29] 34
TABLE 5 : COMPARAISON DES MODELES – TOKENS 64
TABLE 6 : COMPARAISON – GPT [65] 96
TABLE 7 : COMPARAISON : BERT-BASE & BERT – LARGE [41] 100
TABLE 8 : COMPARAISON DE BERT & GPT 103

SIGLES ET ABREVIATIONS

TERMES	ABREVIATIONS
Intelligence Artificielle	IA ou AI
Machine Learning	ML
Deep Learning	DL
Intelligence Artificielle Générative	GenIA
Generative Pre-trained Transformers	GPT
Bidirectional Encoder Representations from Transformres	BERT
Reinforcement Learning	RL
Convolutional Neural Network	CNN
Recurrent Neural Network	RNN
Large Language Model	LLM
Generative Adversarial Network	GAN
Variational Autoencoders	VAE

CHAPITRE I : INTRODUCTION GENERALE

I. CONTEXTE DU SUJET :

Depuis l'avènement de l'intelligence artificielle (IA) jusqu'à nos jours, d'innombrables évènements se sont produits à notre époque. Ce terme IA englobe une multitude de domaines, rendant extrêmement difficile de trouver une définition bien précise. L'intelligence Artificielle, peut être définie comme étant un programme informatique ou d'un système, qui forme un modèle à partir de données d'entrées. Grâce à ces avancées, plusieurs domaines d'études ont émergé à savoir le Machine Learning, le Deep Learning, et en particulier, l'émergence de l'intelligence artificielle générative.

L'intelligence artificielle générative est une discipline, un sous-domaine de l'intelligence artificielle, capable de créer de nouveaux contenus, tels que des images, des sons, des vidéos, et de faire des résumés de textes, etc. Elle est basée sur une architecture appelée Transformer. Il s'agit d'une discipline qui ne fait que croitre de jour en jour, à tel point que des applications peuvent simplement simplifier, voire automatiser nos tâches du quotidien.

A partir de la sortie de l'article « Attention Is All You Need » en 2017, une nouvelle architecture a fait son apparition, à savoir : les Transformers. Cette architecture repose sur un mécanisme d'auto-attention, traite les mots en relation avec tous les autres mots d'une phrase, contrairement aux méthodes traditionnelles qui analysent de façon séquentielle.

Ainsi, cette architecture change complètement la façon d'appréhender les intelligences artificielles. En 2018, une application appelée ChatGPT est créée par OpenIA. Cette application, basée sur le modèle de langage LLM appelé GPT (Generative Pre-trained Transformers), donne un coup d'élan sur l'IA générative. Les LLM représentent le summum de la technologie de l'IA, poussant ainsi les limites de ce qui était autrefois perçu comme impossible.

Cependant, comme pour toute invention humaine majeure, un dilemme de double usage se pose. Ces modèles étant stochastiques, c'est-à-dire fondés sur des probabilités, ils ne peuvent donner des résultats fiables à 100%. Ils peuvent ainsi causer des hallucinations, des désinformations, des contenus malsains, etc. Ainsi, des attaques existantes plus dangereuses et plus maléfiques peuvent être raffinées, automatisées et contrôlées par l'IA générative. C'est dans ce contexte que notre cette étude se penche sur « **Les attaques possibles sur l'intelligence artificielle générative** ».

Bien que ces attaques existent, il est primordial que des mesures drastiques soient prises pour protéger les personnes physiques et morales. Ainsi, des organismes internationaux et des chercheurs en sécurité informatique ont ainsi développé des méthodes de défenses et d'atténuations contre celles-ci. En fin de compte, ces IA génératives doivent être utilisés avec prudence et esprit critique, car elles ne sauraient se substituer à la réflexion humaine. Il est essentiel, de garder à l'esprit, qu'elles peuvent parfois commettre des erreurs ou générer des contenus trompeurs exposant ainsi à des attaques.

II. PROBLEMATIQUE :

Dans quelle mesure, où les systèmes d'IA génératives s'intègrent de plus en plus dans des applications critiques et accessibles via des APIs, représentent-elles une menace face à la sécurité des données, et comment peut-on identifier et contrer ses vulnérabilités à savoir : les injections de prompts, de l'empoissonnement de données, l'extraction de modèles et bien d'autres etc. qui pourraient compromettre l'intégrité, la confidentialité, la fiabilité et la disponibilité de ces systèmes, tout en maintenant leur capacité à produire des contenus innovants et performants ?

Cette problématique consiste à explorer :

- Les menaces spécifiques aux modèles d'IA génératives ;
- Les vulnérabilités potentielles spécifiques sur les modèles génératives (manipulation des données d'entraînement, attaques par injection, etc.) ;
- Les attaques potentielles sur l'IA génératives ;
- Les impacts potentiels de ces attaques sur la performance et la sécurité des systèmes ;
- Les stratégies de défense et d'atténuation, incluant la détection, la résilience, etc. afin d'assurer une utilisation sécurisée de l'IA générative dans des environnements de production.

III.OBJECTIF ET MÉTHODOLOGIE :

Cette technologie est plus ou moins l'évolution croissante de l'intelligence artificielle en générale. Dans notre contexte d'étude, cette IA générative se heurte à d'énormes défis sur la sécurité de ces modèles faces aux attaques potentielles.

Ainsi, l'objectif de ce mémoire est de :

- ✓ identifier et étudier tous les modèles d'IA génératives ;
- ✓ identifier et classifier les menaces, les vulnérabilités et les attaques potentielles sur les systèmes d'IA génératives ;
- ✓ identifier les impacts potentiels de ces attaques ;
- ✓ étudier des solutions possibles de défenses et d'atténuations.
- ✓ étudier des cas pratiques des attaques sur l'IA génératives ;

Par ailleurs, pour mener à bien notre recherche, nous avons effectué une recherche documentaire dans quelques bases de données scientifiques, telles que : Google Scholar, IEEE Xplore, Link Springer, ArXiv, IBM, Mdpi. Notre recherche s'est basée sur des mots et des phrases comme : ''Generative IA'' ou ''Raise Generative IA Attack'' ou ''Risk on Generative IA'' et ''survey'' ou ''Review'' ou ''Summary''.

Les résultats du processus de recherche sont les suivants : les années de publications s'échelonnent entre 2020 et aujourd'hui, et 90% des articles sont entre 2022 et aujourd'hui. La majeure des articles sont de ArXiv. Ce document est majoritairement référencé. Nous avons 80 % de nos recherches viennent des articles de ArXiv et les 20% sont départagées entre les autres.

IV. ORGANISATION DU DOCUMENT :

Ce présent document est constitué, en plus de ce chapitre contextuel divisé en trois parties essentielles (l'introduction, l'objectif, l'organisation du document et de la méthodologie), de six autres chapitres.

Le deuxième chapitre présente un état de l'art sur l'évolution de l'intelligence artificielle et de l'intelligence artificielle générative.

Dans le troisième chapitre, nous faisons une étude d'abord sur l'intelligence artificielle générative, en présentant les définitions et les concepts de base de l'IA générative, ensuite sur l'architecture des modèles d'IA générative et enfin les catégories d'IA génératives.

Le quatrième chapitre est consacré à la base fondamentale du prompt engineering comportant tout d'abord une introduction générale sur le prompt, après sur l'architecture du prompt qui passe en autre par des : paramètres, structures, techniques de base de prompt et des exemples à l'appui, ensuite des techniques avancées du prompt et enfin des méthodes de prompts.

Dans le cinquième chapitre, nous présentons les larges langages modèles (LLM), commençant par une introduction sur les LLM, ensuite l'architecture des LLM en décortiquant deux grands modèles GPT et BERT et enfin la comparaison des deux modèles LLM, GPT et BERT.

Le sixième chapitre expose l'étude des attaques sur l'IA générative. On y trouve une brève introduction, suivie des études sur les menaces, sur les attaques et sur les défenses et les moyens d'atténuations de celles-ci.

Dans le septième chapitre, nous abordons un cas pratique d'attaques de type1 et un moyen de défense y afférent. Cette attaque se nomme l'injection par invite directe. Elle est réalisée sur un Chatbot conversationnel.

CHAPITRE II : ETAT DE L'ART

I. INTRODUCTION :

Qu'est-ce que l'IA ?

Il fut auparavant un concept bien étranger, avant son essor dans les années 1950. Le terme Intelligence Artificielle[1], [2] « IA ou AI » a été créé par *John McCarthy*[1] qui le définit comme suit : « *C'est la science et l'ingénierie de la fabrication de machines intelligentes, en particulier de programmes informatiques intelligents. Elle est liée à la tâche similaire qui consiste à utiliser des ordinateurs pour comprendre l'intelligence humaine, mais l'IA ne doit pas se limiter aux méthodes qui sont biologiquement observables.* » [1]

Depuis son apparition, le développement de l'IA suscite beaucoup d'inquiétudes et d'interrogations sur son évolution dans la société. Même si, l'IA a commencé dans les années 1950, elle est devenue populaire. Elle a été très utilisée ces dernières années pour trois raisons. D'abord, la disponibilité des "big data", c'est-à-dire la quantité gigantesque de données générées par le commerce électronique, les réseaux sociaux et les entreprises, etc. Ensuite, les algorithmes d'apprentissage automatique (ML) [9] et profond (DL) [9] sont améliorés et plus fiables. Enfin, les systèmes informatiques en nuage et à haute performance sont devenus bon marché. Elle a conduit à des avancées technologiques significatives et d'efficacités opérationnelles. Toutefois, à mesure que nous avançons, nous devons reconnaitre et comprendre les différentes formes. Parmi celles-ci, nous avons l'IA générative (GenIA) [1], une tendance émergente, un sous-ensemble de l'IA, qui a montré un immense potentiel pour remodeler les industries de l'IA.

L'IA générative se distingue en se concentrant sur la création de données entièrement nouvelles. Contrairement à l'IA, qui analyse les données existantes pour résoudre des problèmes ou faire des prédictions, des classifications, l'IA générative s'appuie sur sa compréhension des modèles de données pour créer un contenu entièrement nouveau. Elle révolutionne la création de contenu grâce à sa capacité à produire des données entièrement nouvelles dans différents formats. Cette technologie évolue rapidement et les experts prévoient des avancées significatives dans les années à venir.

Ainsi une question mérite d'être posée à savoir : ***en quoi l'IA générative diffère-t-elle de l'IA*** ?

[1] *Mathématicien et Informaticien Américain, inventa le terme d'intelligence artificielle*

Dans ce chapitre, nous commencerons par une brève évolution de l'IA, c'est-à-dire une introduction sur le Machine Learning (ML), du Deep-Learning (DL), ainsi que les applications (cas d'utilisation) et suivis par une évolution de l'intelligence artificielle générative, ses modèles et applications, et finir par une conclusion.

II. EVOLUTION DE L'INTELLIGENCE ARTIFICIELLE :

I. CONTEXTE :

L'IA est une branche de l'informatique qui traite de la création d'agents intelligents (IA Agents), c'est-à-dire des systèmes capables de raisonner, d'apprendre et d'agir de manière autonome. Essentiellement, l'IA concerne la théorie et les méthodes permettant de construire des machines qui pensent et agissent comme des humains [1]. Les progrès réalisés dans le domaine du traitement informatique et du stockage des données ont permis d'ingérer et d'analyser plus de données que jamais auparavant. L'IA est devenue un sujet populaire à la fois au sein et en dehors de la communauté scientifique. De plus, cette IA se manifeste sur deux aspects à savoir : **L'IA FIABLE et L'IA FORTE** [1].

L'IA fiable, communément appelée ***IA ETROITE*** (NARROW IA) ou Intelligence Artificielle Etroite (*ANI, Artificiel Narrow Intelligence*), est une IA entraînée et ciblée pour réaliser des tâches spécifiques. Elle est à l'origine de la plus grande partie de l'IA qui nous entoure aujourd'hui. Par contre, l'IA forte se constitue de *l'intelligence artificielle générale* (AGI) et de la *super-intelligence artificielle* (ASI). L'AGI, ou IA générale, est une forme théorique d'IA dans laquelle une machine aurait une intelligence égale à celle des humains [1].

Par ailleurs, l'IA englobe l'apprentissage automatique (ML) et l'apprentissage profond (DL). L'apprentissage automatique (ML) et l'apprentissage profond (DL) sont des sous-domaines de l'IA, et l'apprentissage profond (DL) est un sous-domaine de l'apprentissage automatique (ML).

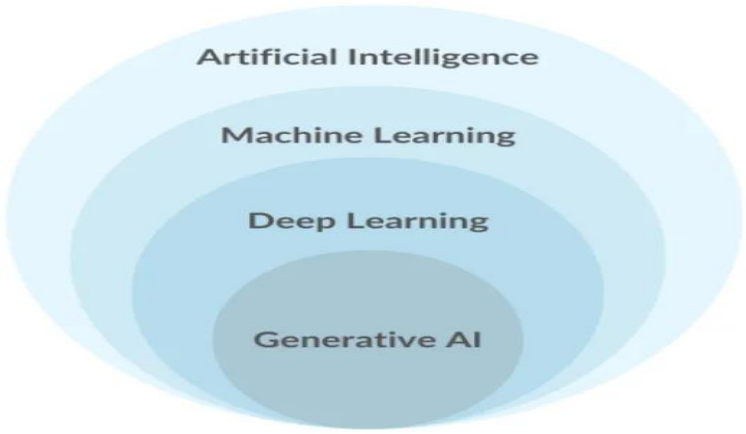


Figure 1: Domaines de IA[4]

Les algorithmes de l'apprentissage automatique et l'apprentissage profond utilisent tous deux les réseaux neuronaux [5] pour apprendre à partir d'énormes quantités de données. Ces réseaux neuronaux sont des structures programmatiques basées sur les processus décisionnels du cerveau humain. Ils sont constitués de couches de nœuds interconnectés qui extraient les fonctionnalités des données et font des prédictions sur ce que représentent les données [1]. Ces deux apprentissages diffèrent par les types de réseaux neuronaux qu'ils utilisent et par la quantité d'interventions humaines impliquées.

Les algorithmes classiques de machine Learning utilisent des réseaux neuronaux avec une couche d'entrée, une ou deux couches « cachées » et une couche de sortie. En général, ces algorithmes sont limités à l'apprentissage supervisé [6] : c'est-à-dire, les données doivent être structurées ou étiquetées par des experts humains pour permettre à l'algorithme d'extraire des fonctionnalités des données.

Les algorithmes d'apprentissage profond utilisent des réseaux neuronaux profonds, c'est-à-dire des réseaux composés d'une couche d'entrée, de trois couches cachées ou plus (mais généralement de centaines) et d'une mise en page de sortie [7].

Pourtant, il subsiste une certaine confusion autour de l'IA, de l'apprentissage automatique et de l'apprentissage profond. Ces termes sont fortement associés, mais ils ne sont pas interchangeables.

II. ETUDE DE L'APPRENTISSAGE AUTOMATIQUE :

L'apprentissage automatique (ML) est un sous-ensemble de l'intelligence artificielle (IA). Il est important de noter que les modèles d'apprentissage automatique ne sont jamais explicitement programmés ; ils apprennent plutôt à partir de modèles de données [8].

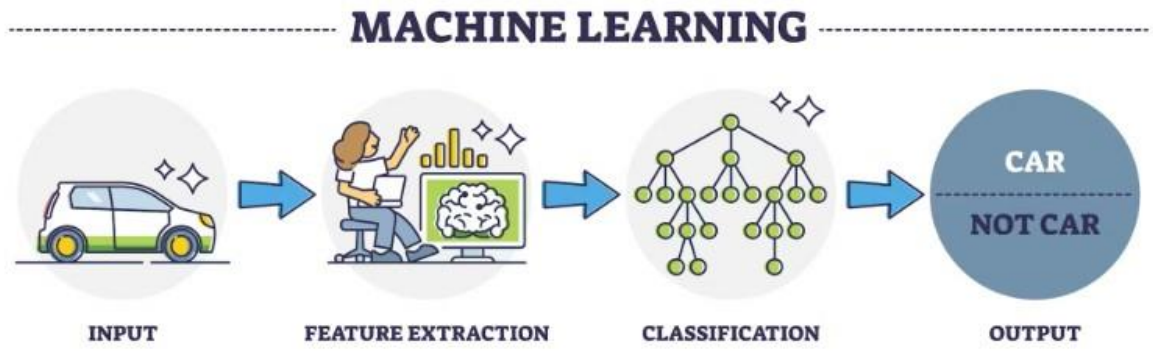


Figure 2: MACHINE LEARNING *[8]*

Dans le cadre de l'apprentissage automatique, un programme informatique se voit confier un ensemble de tâches à accomplir. On dit que la machine a appris de son expérience si ses performances mesurées dans ces tâches s'améliorent au fil et à mesure du temps qu'elle s'exerce à les accomplir [8].

Les algorithmes d'apprentissage automatique peuvent être formés de nombreuses façons, chaque méthode ayant ses avantages et ses inconvénients. Sur la base de ces méthodes et de ces modes d'apprentissages, l'apprentissage automatique est divisé en quatre grandes catégories [4], [9] comme illustre la figure ci-après :

- **Apprentissage Supervisé (*Supervised Machine Learning*) ;**
- **Apprentissage non Supervisé (*UnSupervised Machine Learning*) ;**
- **Apprentissage Semi Supervisé (*Semi Supervised Learning*) ;**
- **Apprentissage par Renforcement (*Reinforcement Learning).***

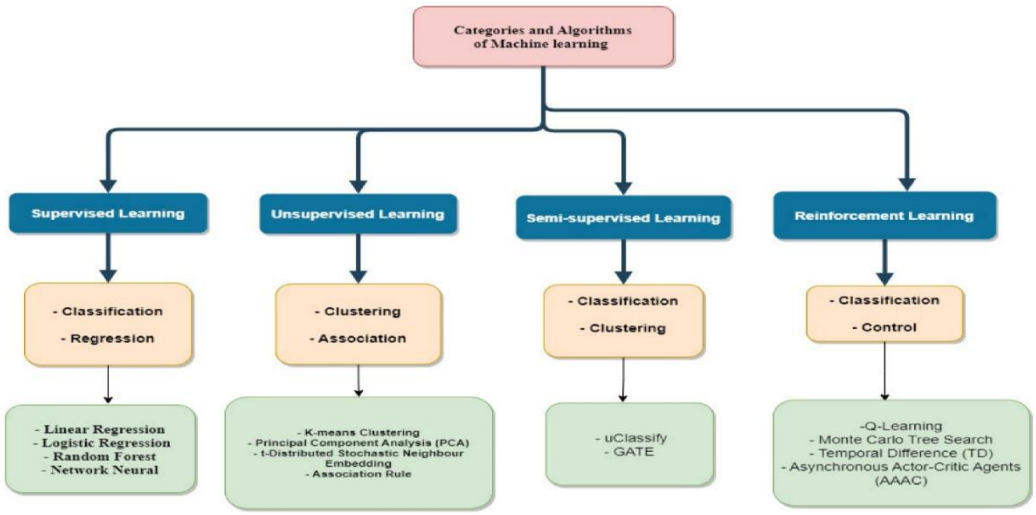


Figure 3:Différents ML et d'algorithmes [9]

1. Apprentissage Supervisé (Supervised Machine Learning) :

L'apprentissage Supervisé [6],[9] connu sous le nom en anglais « Supervised Machine Learning », est un sous domaine du machine learning [2],[8],[9] et de l'intelligence artificielle [1],[2]. Il permet d'utiliser des jeux de données annotées ou étiquetées pour entrainer les algorithmes à classer les données ou à prédire les résultats avec précisions. En effet, ces données dites étiquetées sont composées de paires : la sortie souhaitée qui peut être définie comme un signal de supervision, et l'entrée qui peut être exprimée comme un vecteur [10].

Par ailleurs, ce modèle d'apprentissage peut être utilisé dans beaucoup de domaines tels que : la reconnaissance d'images et d'objets, l'analytique prédictive, la détection des spams, etc., et cela grâce, à des fonctions de classifications et de régressions qui utilisent des algorithmes et des techniques de calculs dans les processus d'apprentissage supervisé, notamment : **la régression linéaire** [6], **la régression logistique** [6], l**a forêt d'arbres décisionnels** [6] **, les réseaux de neurones** [6]**, etc. .**

Dans la suite, nous allons aborder l'apprentissage non supervisé communément appelé : Unsupervised Machine Learning.

2. Apprentissage non Supervisé (UnSupervised Machine Learning) :

L'apprentissage non supervisé [7],[9] ou apprentissage automatique non supervisé, entraine l'algorithme d'apprentissage sur un ensemble de données d'entrées dépourvu de tout résultat annoté, contrairement à l'apprentissage supervisé. Pour chaque élément d'entrée, il n'y a pas de bon ou de mauvais résultat et, contrairement à l'apprentissage supervisé, il n'y a pas d'intervention humaine pour corriger ou adapter. L'apprentissage non supervisé est donc plus arbitraire que l'apprentissage supervisé [7].

L'objectif premier de l'apprentissage automatique non supervisé est d'obtenir une compréhension plus approfondie des données en reconnaissant leur structure de base ou leur modèle de distribution [9]. De ce fait, il tente de représenter un modèle d'entrée spécifique détecté tout en le reflétant sur la structure générale des modèles d'entrée au fur et à mesure qu'il apprend par lui-même.

Les modèles d'apprentissage non supervisé sont utilisés pour trois tâches principales : la classification, l'association et la réduction de la dimensionnalité [9]. Dans la suite, nous aborderons l'apprentissage semi-supervisé appelé : Semi Supervised Learning.

3. Apprentissage Semi Supervisé (Semi Supervised Learning) :

L'apprentissage semi-supervisé [2],[8],[11] se situe entre l'apprentissage supervisé et l'apprentissage non supervisé. C'est une combinaison d'ensembles de données étiquetées et non étiquetées pour former des algorithmes. Cette branche de l'apprentissage automatique traite de plusieurs problèmes d'apprentissage dans le monde réel [2]. Parce qu'il utilise une grande quantité de données non étiquetées et une très petite quantité de données étiquetées, l'apprentissage semi-supervisé nécessite moins d'interaction humaine [2]. Il est facile d'obtenir des algorithmes d'apprentissage semi-supervisé en étendant les algorithmes d'apprentissage supervisé ou les algorithmes d'apprentissage non supervisé [2].

Par ailleurs, les algorithmes semi-supervisés [7] permettent d'explorer les modèles latents à partir d'exemples non étiquetés, ce qui évite d'avoir recours à un grand nombre d'étiquettes. En fait, on peut en citer quelques-uns à savoir : la classification, le clustering (regroupement), la régression etc.

Dans la partie suivante, nous allons aborder l'apprentissage par renforcement appelé aussi : Reinforcement Learning.

4. Apprentissage par Renforcement (Reinforcement Learning) :

L'apprentissage par renforcement [3],[12] est un apprentissage automatique qui permet à un agent d'acquérir des connaissances pour résoudre des problèmes en utilisant la technique de l'essai et de l'erreur. L'agent agit, s'adapte, apprend de ses expériences et améliore ses performances. De ce fait, il sera récompensé pour chaque bonne action et pénalisé pour chaque mauvaise action [13],[14].

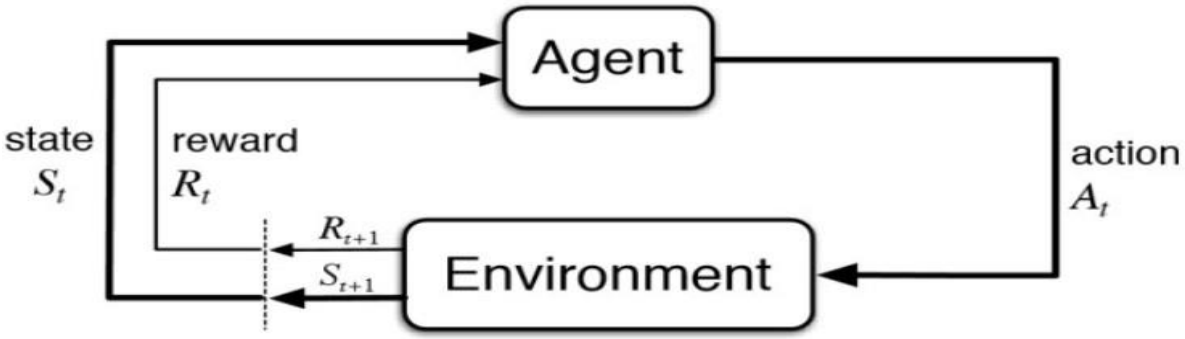


Figure 4:Reinforcement Learning [15]

L'objectif est simplement de choisir des actions pour prendre des décisions qui maximisent autant que possible les récompenses futures qui lui sont données sous la forme d'un signal de récompense. Il convient de noter que l'apprentissage par renforcement diffère de l'apprentissage supervisé et profond par son mode d'apprentissage [16],[17].

Parmi les principales applications de l'apprentissage par renforcement dans la vie réelle, on peut citer les voitures autonomes, les jeux personnalisés, etc.

En conclusion, en explorant de long en large, les techniques, et les algorithmes d'apprentissages, nous pouvons dire que, l'IA a eu beaucoup d'avancées significatives tant son domaine mais aussi dans les autres, et cela suscite d'énormes intérêts, et de défis à leur application.

Dans la section suivante, nous allons aborder l'apprentissage profond appelé aussi : le Deep Learning, un sous-domaine du Machine Learning.

III. ETUDE DE L'APPRENTISSAGE PROFOND :

Le Deep learning ou l'apprentissage profond constitue un sous-domaine de l'apprentissage automatique. Il s'agit d'un modèle de réseau composé de neurones avec de multiples paramètres et couches entre l'entrée et la sortie [1]. Il utilise des topologies de réseaux neuronaux comme base, qui sont connus sous le nom de réseaux neuronaux profonds.

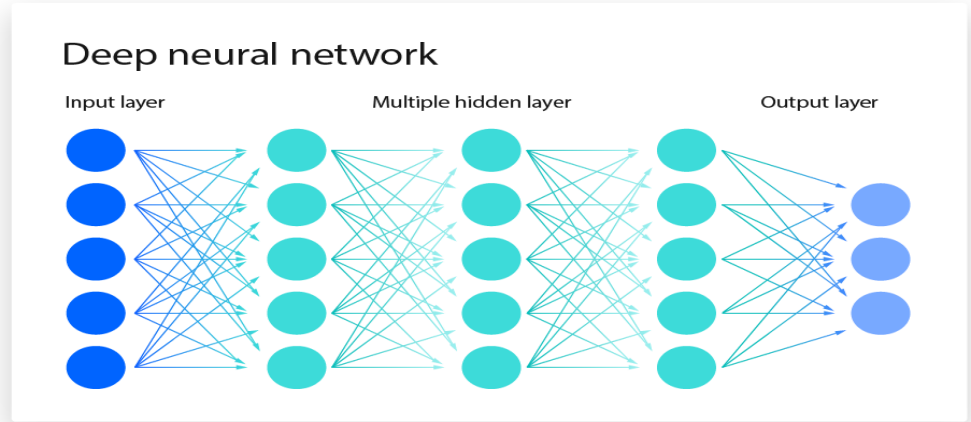


Figure 5:Réseaux de Neurones Profonds [1]

L'apprentissage profond est alimenté par des couches de réseaux neuronaux, qui sont des algorithmes librement inspirés du fonctionnement du cerveau humain [3].

Les réseaux neuronaux artificiels fonctionnement comme le cerveau humain en utilisant des entrées, des poids et des biais, qui agissent comme des neurones artificiels. Leur objectif est de reconnaître, classer et décrire les données avec précision. Les réseaux neuronaux profonds se composent de plusieurs couches interconnectées, où chaque couche améliore les prédictions ou les classifications [4],[5]. Ce processus, appelé propagation avant, traite les données d'une couche à l'autre. Les couches visibles incluent les couches d'entrée et de sortie. La couche d'entrée est celle où le modèle de DL prend les données à traiter, et la couche de sortie est celle où la prédiction ou la classification finale est effectuée [4], [5], [15].

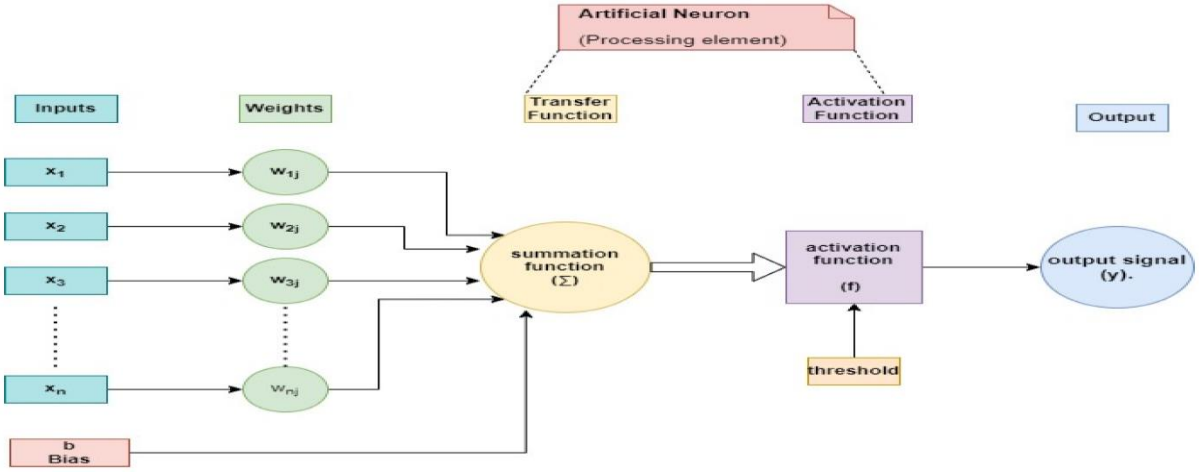


Figure 6: Modèle Mathématique d'un Neurone Artificiel [3]

Un autre processus appelé rétropropagation utilise l'algorithme de descente de gradient pour calculer les erreurs dans les prédictions, puis simule les poids et les biais de la fonction en remontant dans les couches pour entraîner le modèle [3]. Au fil du temps, l'algorithme devient plus précis. L'entraînement à l'aide de grandes quantités de données permet de configurer les neurones du réseau neuronal [3].

Bref, toute l'architecture de l'apprentissage profond est utilisée pour l'extraction et la modification des caractéristiques [3], [13].

Dans l'apprentissage profond, les unités de traitement graphique (GPU) sont optimisées pour l'entraînement des modèles [3]. Il est donc adapté au traitement d'ensembles de données plus importants et plus complexes [3], [10].

A. LES TYPES DE MODELES DE DEEP LEARNING :

Face à la complexité et aux changements, le monde présente des défis significatifs pour les modèles de DL. Par conséquent, plusieurs modèles de DL ont été développés en fonctions des problèmes et des applications, et classés suivants 4 grands catégories [16] : l'Apprentissage Profond Supervisé (Deep Supervised - DS), l'Apprentissage Profond Non-Supervisé (Deep Unsupervised - DU), l'Apprentissage Profond par Renforcement (Deep Reinforcement - DR) et l'Apprentissage Profond Hybride (Deep hybrid - DH) modèles. Sur cette figure ci-dessous, nous observons parfaitement les principales catégories de DL :

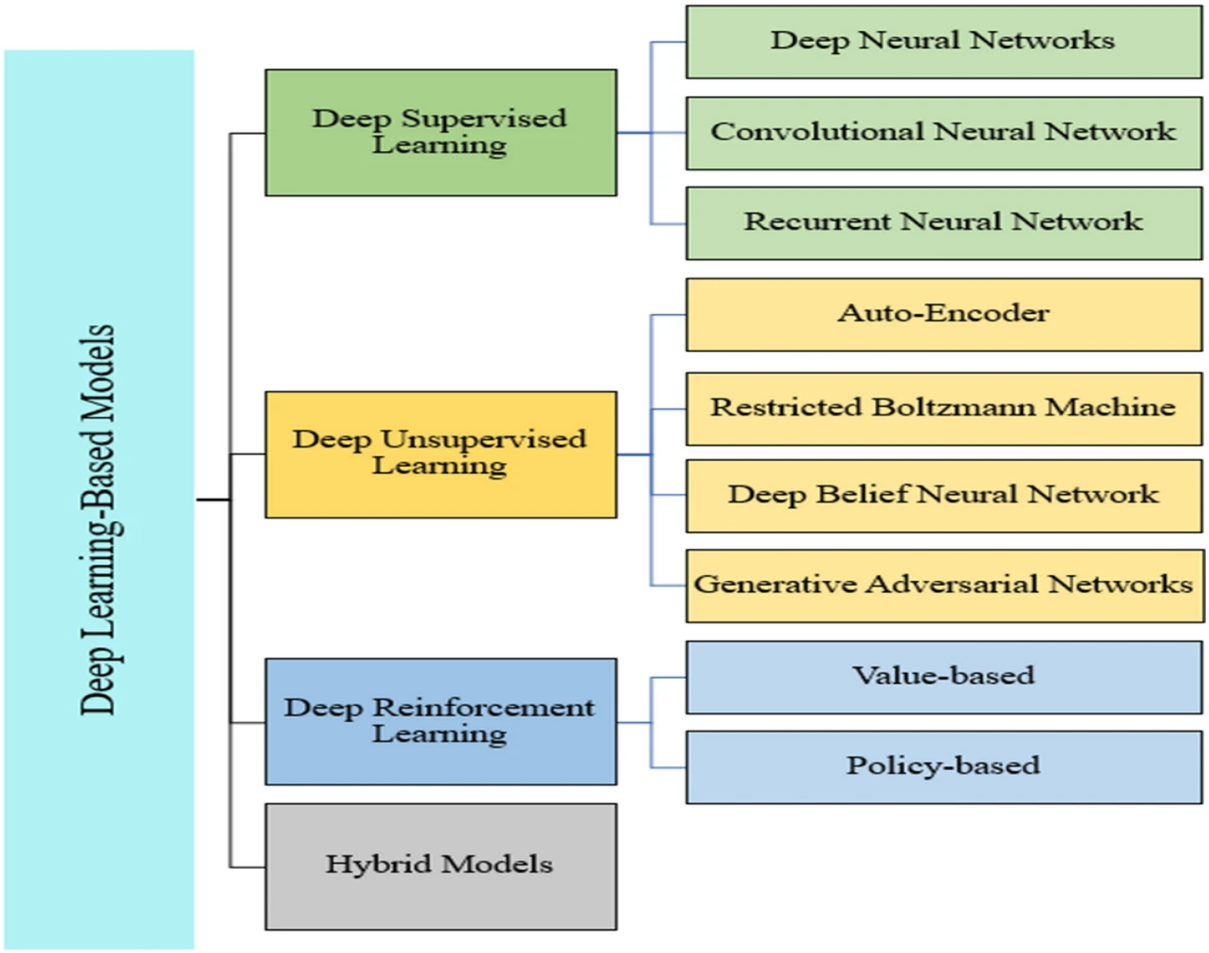


Figure 7: Deep Learning Catégories [19]

1. Apprentissage Profond Supervisé (Deep Supervised Learning) :

Les modèles basés sur l'apprentissage supervisé profond sont l'une des principales catégories de modèles d'apprentissage profond qui utilisent un ensemble de données d'apprentissage étiquetées pour être formées [19]. Ces modèles mesurent la précision au moyen d'une fonction, la fonction de perte, et ajustent les poids jusqu'à ce que l'erreur ait été suffisamment minimisée. Dans la catégorie de l'apprentissage profond supervisé, trois modèles importants sont identifiés, à savoir : les **réseaux neuronaux profonds** [16], les **réseaux neuronaux convolutifs** [16], [17] et les modèles basés sur les **réseaux neuronaux récurrents** [16], comme illustre la figure ci-dessous :

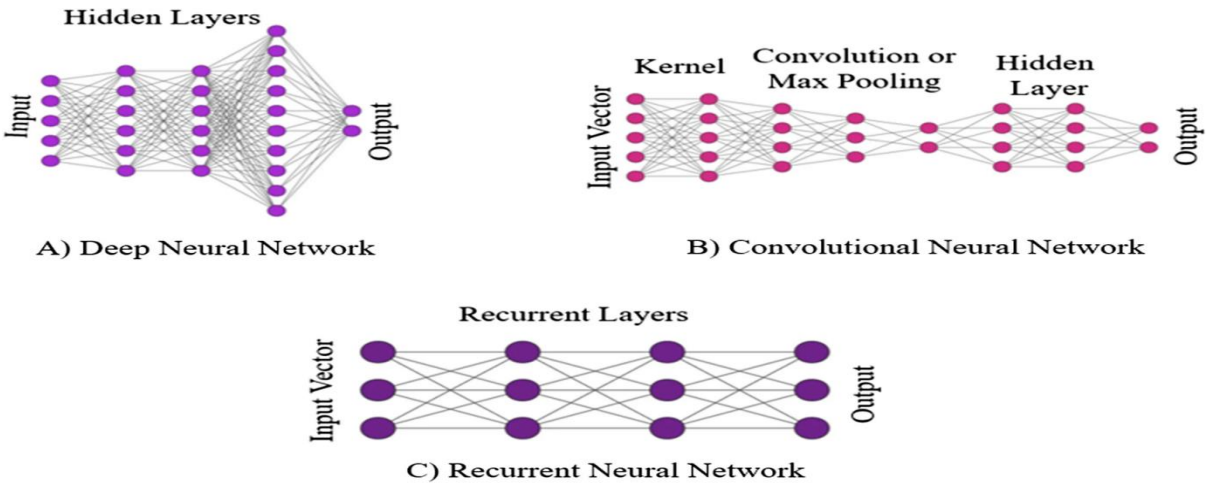


Figure 8: Deep Supervised Learning [19]

a. Réseaux Neuronaux Profonds (Artificial Neural Network) :

Les réseaux neuronaux profonds ou réseaux neuronaux artificiels (ANN), connus sous le nom de réseaux neuronaux, sont l'un des systèmes informatiques inspirés des réseaux neuronaux biologiques [16].

L'un des modèles ANN les plus simples est connu sous le nom de réseau neuronal profond. Les modèles réseaux neuronaux profonds se composent d'une architecture hiérarchique avec des couches d'entrée, de sortie et cachées, chacune d'entre elles ayant une unité de traitement de l'information non linéaire, comme par exemple la figure «Figure 5» [16]. Nous pouvons en citer quelques exemples connus de modèles de réseaux neuronaux profonds, tels que le perceptron multicouche et le réseau neuronal peu profond, etc.

b. Réseaux Neuronaux Convolutifs (Convolutional Neuronal Network - CNN) :

Le CNN est l'algorithme le plus important et le plus largement utilisé dans le domaine de DL. Son principal avantage, par rapport à ses prédécesseurs, est qu'il sélectionne automatiquement les parties importantes sans l'aide d'une personne [3]. Il est inclus dans différents tâches telles que : la détection des objets, la reconnaissance vocale, la vision par ordinateur, la classification d'image [17], etc.

Les CNN sont des types spécifiques des réseaux de neurones [5], constitués de trois couches à savoir :

- Une couche d'entrée ;

- Une à plusieurs couches cachées ;
- Une couche de sortie.

En effet, ces neurones (voir la figure « Figure *6* ») placés dans les couches cachées ont un poids associé et un seuil. Mais, si la valeur de sortie d'un nœud est supérieure à la valeur de seuil spécifié, ce nœud est activé ; dans le cas contraire, ce nœud n'envoie aucune donnée.

Par ailleurs, les principaux composants des CNN sont la couche convolutive (Convolutional layer), la couche de mise en commun (Pooling layer) et la couche entièrement connectée (Fully connected layer) [3], [17]. La figure ci-après illustre parfaitement les différents composants :

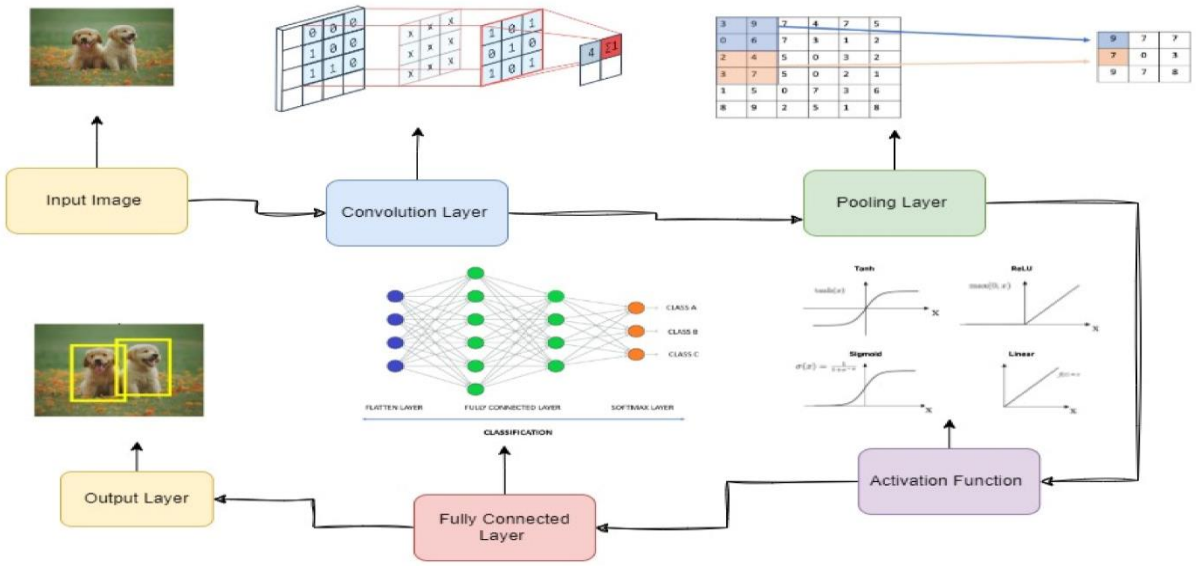


Figure 9: Composants de CNN [3]

- **<u>Couche de Convolution</u>** (Convolutional Layer) :

La couche de convolution est un élément important des CNN. Par le biais de plusieurs couches convolutives, l'opération de convolution permet d'extraire des caractéristiques distinctes de l'entrée. C'est là que se produisent la majorité des calculs. Elle nécessite quelques composants, à savoir : des données d'entrées, un filtre et une carte de caractéristique ou feature-map. La couche convolutive comprend des noyaux de convolution pouvant être appris, qui sont des matrices de poids généralement de longueur et de largeur égales et d'un nombre impair [17]. La figure ci-après illustre parfaitement le processus de convolution.

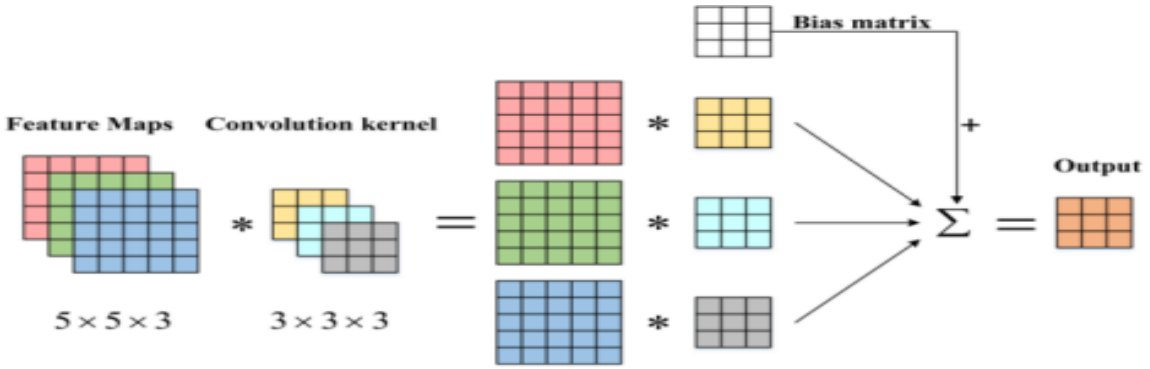


Figure 10 : Schéma de processus de Convolution [20]

- **<u>Couche de Regroupement</u> (Pooling Layer)** :

La couche de mise en commun ou couche de regroupement réduit le nombre de connexions dans le réseau en procédant à un échantillonnage réduit et à une réduction de la dimensionnalité des données d'entrée [20]. Son principal objectif est d'alléger la charge de calcul et de résoudre les problèmes d'ajustement excessif.

Il existe plusieurs méthodes de mise en commun, notamment la mise en commun maximale, la mise en commun moyenne, la mise en commun pyramidale spatiale, la mise en commun mixte, la mise en commun sans ordre à plusieurs échelles et la mise en commun stochastique [17].

La figure ci-dessous illustre un exemple de mise en commun maximale dans lequel une fenêtre glisse sur l'entrée et le contenu de la fenêtre est traité par une fonction de mise en commun.

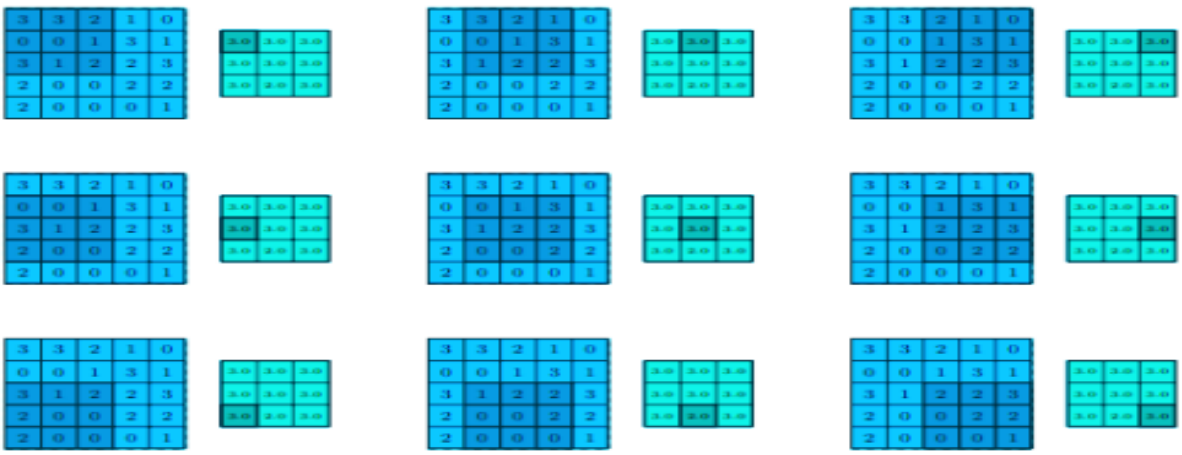


Figure 11 : Opération de pooling max 3 × 3 sur une unité de calcul 5 × 5 [21]

- **Couche Entièrement Connectée (Fully Connected Layer)** :

La couche FC se située à la fin d'une architecture CNN. Dans cette couche, chaque neurone est connecté à tous les neurones de la couche précédente, conformément aux principes d'un réseau neuronal perceptron multicouche classique [20]. La couche FC reçoit l'entrée de la dernière couche de mise en commun ou de convolution, qui est un vecteur créé par l'aplatissement des cartes de caractéristiques. La couche FC sert de classificateur dans le CNN, ce qui permet au réseau de faire des prédictions [17].

c. Réseau Neuronal Récurrent (Recurrent Neural Networks - RNN) :

Les réseaux neuronaux récurrents (RNN) sont une classe de modèles d'apprentissage profond qui possèdent une mémoire interne, ce qui leur permet de capturer des dépendances séquentielles [17]. Les modèles basés sur le traitement du langage naturel (NLP) en sont un exemple. Contrairement aux réseaux neuronaux traditionnels qui traitent les entrées comme des entités indépendantes, les RNN prennent en compte l'ordre temporel des entrées, ce qui les adapte aux tâches impliquant des informations séquentielles [17], [18], etc.

Cependant, les RNN simples sont limités par leur mémoire à court terme, ce qui restreint leur capacité à retenir des informations sur de longues séquences [17]. Pour y remédier, des variantes de RNN plus avancées ont été développées, notamment la mémoire à long terme (LSTM), la LSTM bidirectionnelle, l'unité récurrente gérée (GRU), la GRU bidirectionnelle, le RNN bayésien, et bien d'autres encore [17].

La figure ci-après, illustre parfaitement le fonctionnent interne simple du RNN où la mémoire interne h_t est calculée à l'aide de l'équation suivante :

$$h_t = g\,(Wx_t + Uh_t + b)$$

tels que : g () qui représente la fonction d'activation (typiquement la tangente hyperbolique), U et W sont des matrices de poids ajustables pour l'état caché (*h*), b est le terme de biais, et x dénote le vecteur d'entrée.

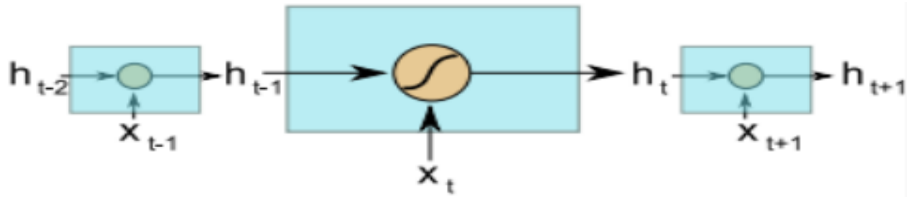


Figure 12 : Fonctionnement interne simple du RNN [20]

2. Apprentissage Profond Non Supervisé (Deep UnSupervised Learning) :

Les modèles profonds non supervisés ont gagné un intérêt significatif en tant que courant principal de modèles d'apprentissage profond viables [22]. Ces modèles sont largement utilisés pour générer des systèmes qui peuvent être formés avec un petit nombre d'échantillons non étiquetés. Les modèles peuvent être classés en auto-encodeurs, machines de Boltzmann restreintes, réseaux neuronaux à croyance profonde et réseaux adversaires génératifs [18].

La figure ci-après illustre les différents modèles de l'apprentissage profond non supervisé c'est-à-dire :

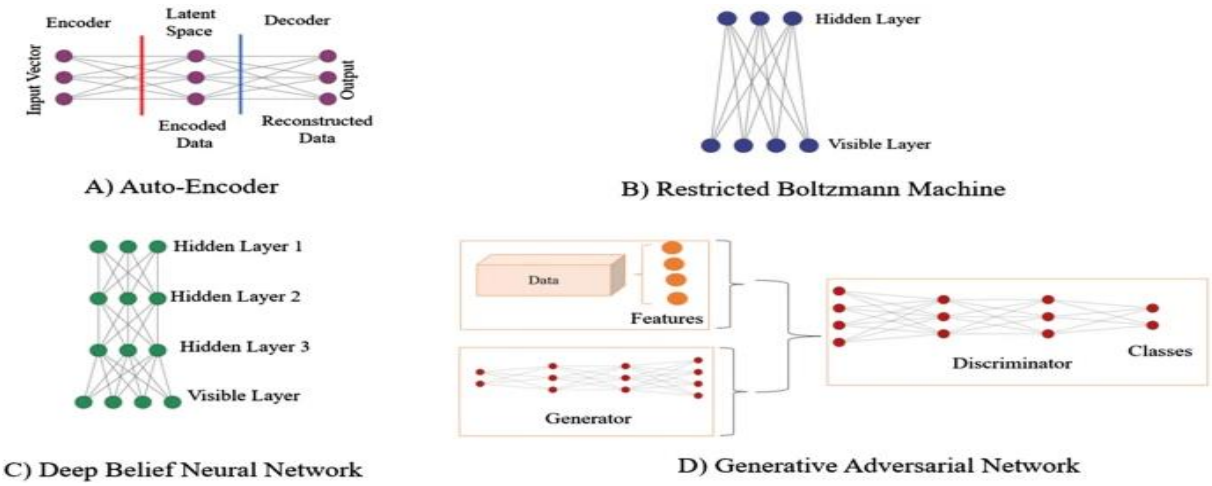


Figure 13 : Différents modèles de Deep UnSupervised Learning [22]

a. Auto-Encodeurs (Auto-Encoder-AE) :

Le concept d'auto-encodeur est né d'un réseau neuronal conçu pour reconstruire ses données d'entrée. Son objectif fondamental est d'apprendre une représentation significative des données de manière non supervisée, ce qui peut avoir diverses applications, y compris le regroupement de données. Il est constitué d'une couche cachée interne qui définit un code représentant les données d'entrée [17], [19].

Les auto-encodeurs *simples* sont utilisés à diverses fins, notamment pour reconstruire des images corrompues ou floues [17] comme illustre la figure « Figure *13* ».

En général, les modèles d'auto-encodeurs peuvent être classés en deux grandes catégories : les auto-encodeurs régularisés, qui sont utiles pour l'apprentissage de représentations en vue de tâches de classification ultérieures, et les auto-encodeurs variationnels (VAE), qui peuvent fonctionner comme des modèles génératifs [17].

b. Machine Boltzmann restreinte (Restricted Boltzmann Machine RBM) :

Le RBM est une famille classique de modèles d'apprentissage automatique qui a joué un rôle central dans le développement de l'apprentissage profond [20]. Le modèle de la machine de Boltzmann restreinte (RBM), connu sous le nom de distribution de Gibbs, est un réseau de neurones connectés les uns aux autres [18], comme le montre la figure «Figure *13*».

Dans la RBM, le réseau se compose de deux couches, à savoir : la couche d'entrée ou couche visible et la couche, cachée comme illustré sur la figure « Figure *13*». Ainsi, les RBM ont a été conçus pour effectuer des tâches non supervisées, même s'ils peuvent également être utilisés pour effectuer un apprentissage supervisé dans un certain sens [20], [21].

Cependant, il n'y a pas de couche de sortie dans la RBM. Or, les machines de Boltzmann sont des réseaux neuronaux aléatoires et génératifs qui peuvent résoudre des problèmes combinatoires [18].

c. Réseau neuronal de croyance profonde (Deep Belief Neural Network - DBN) :

Le réseau de croyance profond (DBN) est un autre réseau neuronal profond non supervisé qui fonctionne de la même manière que le réseau neuronal feed-forward (retour d'information) profond avec des entrées et de multiples couches de calcul, appelées couches cachées, comme illustré à la «Figure *13*». Dans le DBM, deux phases principales sont nécessaires : la phase de pré-entraînement et la phase de réglage fin [18].

La phase de pré-entraînement comprend plusieurs couches cachées. Cependant, la phase de réglage fin n'est considérée que comme un réseau neuronal feed-forward (retour d'information) pour entraîner et classer les données [18], [22].

Le DBN est une méthode efficace pour résoudre les problèmes posés par les RN (réseaux de neurones) à couches profondes [22], [23].

d. Réseau antagoniste génératif (Generative Adversarial Network - GAN) :

Le réseau antagoniste génératif (GAN), initialement proposé par **Ian J. Goodfellow** [24], est une architecture de réseau neuronal remarquable pour la modélisation générative, capable de produire des échantillons réalistes et nouveaux à la demande. Un GAN se compose de deux éléments clés : un modèle génératif et un modèle discriminatif comme illustré à la figure «Figure *13*». Le modèle génératif vise à générer des images qui ressemblent à des images réelles, tandis que le modèle discriminatif vise à différencier les images réelles des images synthétiques. Les deux modèles sont généralement mis en œuvre à l'aide d'un perceptron multicouche [21],[23].

L'avantage du GAN est de créer des sorties réalistes. Il existe, toutefois, un inconvénient potentiel : le générateur et le discriminateur pourraient se renvoyer la balle pendant longtemps, ce qui épuise le système [18].

3. Apprentissage Profond par Renforcement (Deep Reinforcement Learning) :

L'apprentissage par renforcement (DR) est la science qui consiste à prendre des décisions en apprenant le comportement optimal dans un environnement afin d'obtenir une récompense maximale. Le comportement optimal est obtenu par des interactions avec l'environnement [21],[23] comme illustré à la figure ci-après :

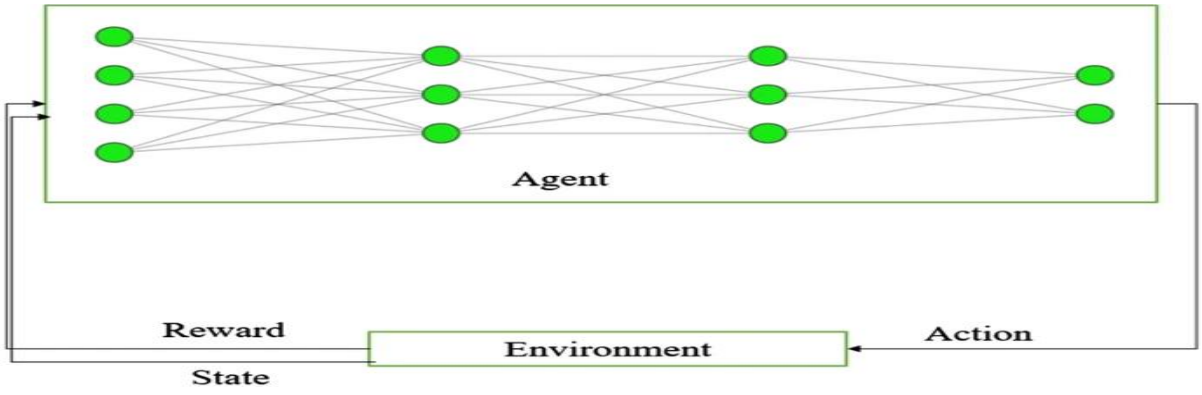


Figure 14 : Deep Reinforcement Learning [22]

Dans le cadre de l'apprentissage par renforcement, l'apprenant, appelé agent, ne reçoit pas d'indications explicites sur les actions à entreprendre à chaque étape.

L'apprentissage par renforcement profond combine les capacités de prise de décision de l'apprentissage par renforcement avec la fonction de perception de l'apprentissage en profondeur. L'apprentissage par renforcement profond peut être divisé en trois méthodes : les méthodes basées sur les valeurs, les méthodes basées sur les politiques et les méthodes basées sur les modèles, comme illustré sur la «Figure 7» :

- **Value-based (Basée sur la valeur) :**

L'apprentissage par renforcement profond basé sur la valeur représente et trouve principalement les fonctions de valeur et leurs optimums. Dans ces méthodes, l'agent apprend l'état ou la valeur état-action et se comporte en fonction de la meilleure action dans l'état. L'une des étapes nécessaires de ces méthodes consiste à explorer l'environnement [18].

- **Policy-based (Basée sur les politiques) :**

Au contraire, l'apprentissage par renforcement profond basé sur la politique trouve une politique optimale, stochastique ou déterministe, pour améliorer la convergence sur un espace d'action continu ou à haute dimension. Ces méthodes sont principalement des techniques d'optimisation dans lesquelles la politique maximale de la fonction peut être trouvée [18].

- **Model-based (Basée sur le modèle) :**

La troisième catégorie de DR, les méthodes basées sur des modèles, vise à apprendre la fonctionnalité de l'environnement et sa dynamique à partir des observations précédentes, tandis que ces méthodes tentent de trouver une solution à l'aide du modèle spécifique [18].

Les modèles d'apprentissage par renforcement profond (DR) trouvent des applications dans divers domaines : tels que les jeux vidéo, la manipulation robotique, la segmentation d'images, l'analyse vidéo, la gestion de l'énergie, et bien d'autres encore [17].

4. Apprentissage Profond Hybrid (Hybrid Deep Learning) :

Les modèles d'apprentissages profonds Hybrides présentent des faiblesses et des forces en termes de réglage des hyperparamètres et d'exploration des données. Par conséquent, les faiblesses mises en évidence de ces modèles peuvent les empêcher d'être des techniques efficaces dans différentes applications [22]. Chaque modèle d'apprentissage profond possède également des caractéristiques qui le rendent efficace pour des applications spécifiques [18].

En conclusion, l'apprentissage profond s'est imposé comme un puissant sous-ensemble de l'apprentissage automatique et de l'intelligence artificielle, surpassant les méthodes traditionnelles d'apprentissage automatique, en particulier dans le traitement d'ensembles de données non structurées et de grande taille. Cependant, la complexité et la nature dynamique des problèmes du monde réel posent des défis dans la conception de modèles d'apprentissage profond efficaces.

III. EVOLUTION DE L'IA GÉNÉRATIVE :

I. CONTEXTE :

D'après un célèbre écrivain scientifique nommé ARTHUR[2] C. CLARK qui disait : « *Toute technologie avancée est magique*. », une affirmation que nous avons peut-être tous éprouvée la première fois quand nous avons utilisé l'IA générative. Brusquement, pour la première fois, nous avons accès à une technologie capable de comprendre nos langues, de répondre à nos demandes et de créer des résultats inédits. L'IA générative peut écrire des poèmes, produire des images étonnantes, rédiger du code, et même nous surprendre avec une blague ou une composition musicale. Cependant, l'IA n'est pas magie, c'est de la science, des mathématiques. Elle n'est pas apparue du jour au lendemain, mais elle est le résultat de plusieurs décennies de recherches. Elle va transformer notre quotidien, bouleverser le monde, mais son impact dépendra de nous [25].

L'intelligence artificielle (IA) générative s'est imposée comme un domaine d'étude de premier plan, révolutionnant divers domaines qui, contrairement à ses prédécesseurs, peut créer

[2] Arthur Charles Clarke, *Profiles of the Future*, 1973

de nouveaux contenus en extrapolant à partir de ses données d'entraînement [26]. Son extraordinaire capacité à produire de l'écriture [27], des images [28], de l'audio et de la vidéo[28], qui semblent créés par l'humain, captive le monde entier depuis le premier Chatbot d'IA générative destiné au public à l'automne 2022. Pourtant, elle s'appuie sur le deep-learning [18], c'est-à-dire une machine learning évoluée. Le deep learning a permis de commencer à représenter d'énormes quantités de données à l'aide de très grands réseaux neuronaux comportant de nombreuses couches. Mais, jusqu'à récemment, une grande partie de l'entraînement se faisait à l'aide de données annotées. Il s'agit de données que les humains étiquettent manuellement. C'est ce que nous appelons « l'apprentissage supervisé », qui est très coûteux. Mais, vers 2017, une nouvelle approche a vu le jour, alimentée par une architecture appelée «Transformers [29]», pour effectuer une forme d'apprentissage, appelée « Apprentissage Auto-Supervisé » base sur le mécanisme d'attention. Dans cette approche, un modèle est entraîné à partir d'une grande quantité de données non étiquetées en masquant certaines sections du texte, des mots, des phrases, etc. et en demandant au modèle de les compléter. Lorsqu'il est mis en œuvre à grande échelle, ce processus étonnant aboutit à une représentation puissante appelée « grand modèle de langage » ou LLM [34],[35]. On entend beaucoup parler des grands modèles de langage (LLM) sur lesquels s'appuie l'IA générative. Mais de quoi s'agit-il ? À la base, il s'agit d'une nouvelle façon de représenter le langage à l'aide d'un grand nombre de paramètres, une représentation obtenue par entraînement à partir d'énormes quantités de texte.

Bien que cette IA générative soit une avancée notable dans le domaine de la recherche, elle repose sur trois piliers essentiels [26]. Le premier pilier, c'est la disponibilité croissante d'ensembles de données à grande échelle, associée aux progrès des techniques d'apprentissage profond, qui a propulsé le développement rapide de l'IA générative. Le deuxième pilier, c'est l'évolution significative des puissances de modèles de langages tels que : le NLP [29] (comme BERT, GPT, etc. …) , le computer vision [32] (comme GAN [24], Stable Diffusion, etc.) et bien d'autres. Le troisième et dernier pilier réside dans la puissance de calcul grâce aux CPU, et principalement aux GPU qui peuvent compter plus de 100 milliards de transistors. Et nous utilisons d'innombrables banques de GPU interconnectés pour fournir la puissance de calcul, tels que les Calculs de Hautes Performances appelés supercalculateurs (HPC) avec NVIDIA [33], nécessaire à la création et à l'exécution des fonctions d'IA générative, au Cloud Computing et au stockage. Chacun de ces piliers, a joué un rôle important dans l'émergence de cette technologie.

Des statistiques récentes soulignent l'intérêt et l'impact croissants de l'IA générative. Precedence Research [34] a indiqué que le marché mondial de l'IA générative représentait 10,79 milliards de dollars en 2022. Il devrait atteindre environ 118,06 milliards USD d'ici 2032, avec un taux de croissance annuel composé (TCAC) de 27,02 % au cours de la période 2023-2032 [26]. De plus, la taille du marché mondial de l'IA générative en logistique devrait atteindre environ 23 465,58 millions USD d'ici 2033, contre 707,99 millions USD en 2023, avec un taux de croissance annuel moyen de 41,69 %. Ce tableau récapitulatif ci-après montre les étapes d'apparition de GenIA, qui reflètent la croissante stupéfiante de cette IA en tant qu'outil puissant doté d'un immense potentiel.

ANNEES	IA GENERATIVES
2014	Generative Adversarial Networks (GANs) [24]
2017	Break Through in NLP: Transformers (Google) Attention is All You need [29]
2018	Advent of Large Models: BERT (Bidirectional Encoder Representation from Transformers) by Google RoBERTA based on BERT GPT (Generative Pre-trained Transformers) by Open AI
2019	Advanced GANs – Style GANs, SBERT(TEXT)
2020	GPT-3: Premier LLM (Open AI), MiniLM (Text)
2021	Vision Transformers – Clip, DALL-E: Images (Open AI)
2022	Chat GPT: GPT-V3.5, BLOOM (Text), DALL-E2, Midjourney-V4(Images)
2023	GPT-4, GPT-turbo, GPT-4o (Open AI), GEMINI (Google), LlaMa3 (Facebook), Midjourney-V6, Groq-V1 (xAI)

Table 1: IA GENERATIVE *[35], [36]*

On a alors assisté à l'émergence de nouveaux modèles, de modèles évolués et à une explosion de modèles ouverts comme une pandémie ou bien plus. L'IA générative est passée de nouveauté fascinante à nouvelle impérative métier en moins d'un an, et une nouvelle utilisation ou application apparaît chaque jour. La croissance est si rapide qu'il nous est impossible de dire exactement où nous en serons dans dix ans ou même dans dix mois. Dans

la section suivante, nous allons exposer en large progrès liés aux modèles d'IA génératives ainsi ses applications, modèles, outils.

II. MODÈLES ET APPLICATIONS GÉNÉRATIVES :

GenIA s'est imposé comme un domaine d'étude de premier plan, révolutionnant ainsi plusieurs secteurs. L'un de ces systèmes d'IA générative, le plus connu, est le ChatGPT d'Open AI. Et pourtant ces systèmes sont basés sur des modèles de GenIA.

Les modèles de GenIA [26] sont aux nombres de 7 (sept) et chacun suit une architecture bien définie. Sur cette figure «Figure *15*», nous apercevons les différents types de modèles de GenIA, à savoir : l'Auto-encodeurs Variationnels, en anglais Variational Autoencoders (VAE), le Réseau Antagoniste Génératif, en anglais Generative Adversarial Networks (GAN), le Modèle de Diffusion, en anglais Diffusion Models, le Modèle de Langage, en anglais Language Models, les Transformers, la Normalisation des modèles de flux, en anglais Normalizing Flow Models et le modèle Hybride, en anglais Hybrid Models.

Ces modèles seront étudiés plus en détails dans le Chapitre III.

L'IA générative a le potentiel de révolutionner les industries créatives, d'améliorer les processus de génération de contenu et de stimuler l'innovation dans tous les secteurs, tels que : la médecine, l'éducation, l'économie, la cybersécurité, le sport, etc… et toutes les applications. En effet, cette révolution est strictement conditionnée par l'effet gigantesque des données sous différentes formes dont les modèles s'appuient pour apprendre, distinguer, affiner ses résultats à travers des phases d'entrainement. Ces données se distinguent selon leur nature d'entrées, et de sorties, et des objectifs spécifiques qui ont permis de créer de multiples applications et d'outils d'IA générative, comme le montre les tableaux ci-dessous :

TYPES	MODELES	CHAMP D'APPLICATIONS
Texte	OpenAI GPT-4, Google PaLM2, DeepMind Gopher, Meta LLaMA, Hugging Face Bloom, Anthropic Claude, etc.	Marketing, ventes, assistance à la clientèle, rédaction générale, traduction, résumé etc…
Image	OpenAI Dall-E2, Stable Diffusion, Midjourney, Sora, Google Imagen, Meta Make-A-Scene, etc	Générer une image, SNS, Publicité, Design, Visualisation de données, etc.

Code	Anthropic Claude, OpenAI Codex, Google PaLM2, Ghostwriter, Amazon CodeWhisperer, Tabnine, Stenography, AI2sql, Pygma, etc.	Génération de code, Documenter le code, Générer des requêtes SQL, Générer des applications Web, Test, Formatage du code, etc.
Audio	Resemble AI, WellSaid, Play.ht, Coqui, Harmonai, Google MusicLM, etc.	Synthèse vocale, clonage de voix, génération de musique, conception d'effets sonores, etc…
Vidéo	MS X-CLIP, Meta Make-A-Video, RunwayML, Synthesia, Rephrase AI, Hour One, etc…	Génération de vidéos, édition de vidéos, résumé de vidéos, etc.
3D	DreamFusion, NVIDIA GET3D, MDM, etc.	Générer des modèles 3D, Générer des scènes 3D, etc.

Table 2 : Modèles et applications de l'IA générative *[35]*

OUTILS	COMPAGNIES	TACHES	ANNEES	DESCRIPTION
VoiceBox	Meta AI	Text-to-Speech	2023	Générer des clips vocaux
Genny	Lovo	Text-to-Speech Text-to-Image	2020	Générer des vocaux à travers des œuvres d'art
Metamate	Meta AI	Text-to-Code	2023	Logiciel de débogage
AlphaCode	DeepMind	Text-to-Code	2022	Générer du code, du contenu créatif et répondre aux questions de manière informative.
AppleGPT	Apple	Text-to-Text	2023	chatbot résume un texte et

				répond à des questions
Claude 2	Anthropic	Text-to-Text	2023	Générer des contenus et Assistant AI
DeepDream	Google	Text-to-Image	2015	Générer des images psychédéliques
StyleGAN	NVIDIA	Text-to-Image	2019	Générer des images réalistes et créatives à partir d'un texte
Imagen	Google	Text-to-Image	2023	Générer des Images Réalistes
PEER	Meta AI	Text-to-Speech	2023	Outil d'écriture
Descript	Descript	Text-to-Video	2020	Générer des vidéos à partir d'un text

Table 3 : Outils d'IA Générative *[36]*

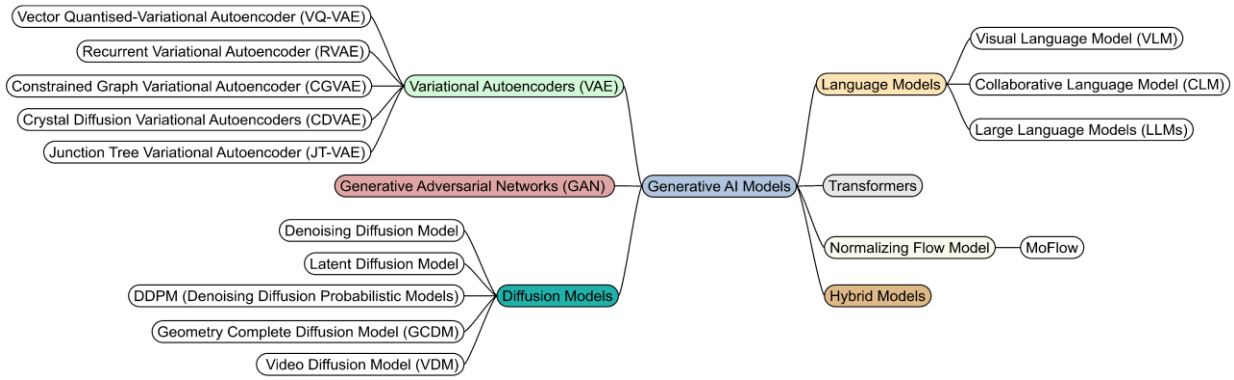


Figure 15 : Modèles IA Générative et Architectures *[26]*

Ces modèles IA générative et leurs applications ont apporté plus de bien de que mal pour l'instant. Cependant, des chercheurs sonnent l'alerte sur d'éventuelles inquiétudes sur cette technologie, notamment en termes d'éthiques, de pertes d'emplois etc… Mais des questions qui se posent : Doit-on craindre nos créations ? Ou bien sont-elles plus intelligentes que nous ? Or de toutes les créatures du monde, nous resterons toujours les plus intelligents quelles que soient nos créations.

IV. CONCLUSION :

L'intelligence artificielle a fait l'objet de nombreux cycles d'engouement. Mais, même pour les plus sceptiques, la sortie de ChatGPT semble marquer un tournant. Aujourd'hui, avec l'IA générative, c'est le traitement automatique du langage naturel (NLP) qui a fait un bond en avant.

En effet, l'IA générative peut apprendre, affiner, distinguer et synthétiser non seulement le langage, mais aussi d'autres types de données, notamment les images, les vidéos, le code logiciel, le texte et même les structures complexes.

A l'heure actuelle, cette technologie est la priorité de toutes les grandes entreprises, telles que les GAFAM[3], car personne ne veut se faire distancer par ses concurrents. Alors, où en est la place de l'Afrique dans tout ça ?

[3] Google, Amazon, Facebook, Apple et Microsoft

CHAPITRE III : L'INTELLIGENCE ARTIFICIELLE GÉNÉRATIVE

I. DÉFINITION ET CONCEPT DE BASE :

Au cours, des deux dernières années, une pléthore de grands modèles génératifs tels que GPT, LLaMa, Stable Diffusion, et bien d'autres ont été publiés. Effectivement, ces modèles sont capables d'effectuer des tâches telles qu'un système général de questions-réponses ou la création automatique d'images artistiques qui révolutionnent plusieurs secteurs. En effet, les implications de ces modèles génératifs dans l'industrie et la société sont énormes, car plusieurs postes de travail peuvent être transformés.

Prenons trois modèles de Chats conversationnels basés sur des modèles GenIA et voyons leur définition sur l'IA générative.

Selon ChatGPT, l'IA générative est définie comme suit : « *L'intelligence artificielle générative (IA générative) désigne une classe d'algorithmes et de modèles d'IA capables de créer du contenu nouveau et original à partir de données d'entrée. Contrairement aux IA traditionnelles qui se contentent d'analyser des données ou de classer des informations, l'IA générative apprend les motifs et les structures des données d'entrainement pour générer du contenu qui ressemble à ces données, tout en étant unique et différent. Cela peut inclure du texte, des images, de la musique, du code, et plus encore.* » comme le montre la figure suivante :

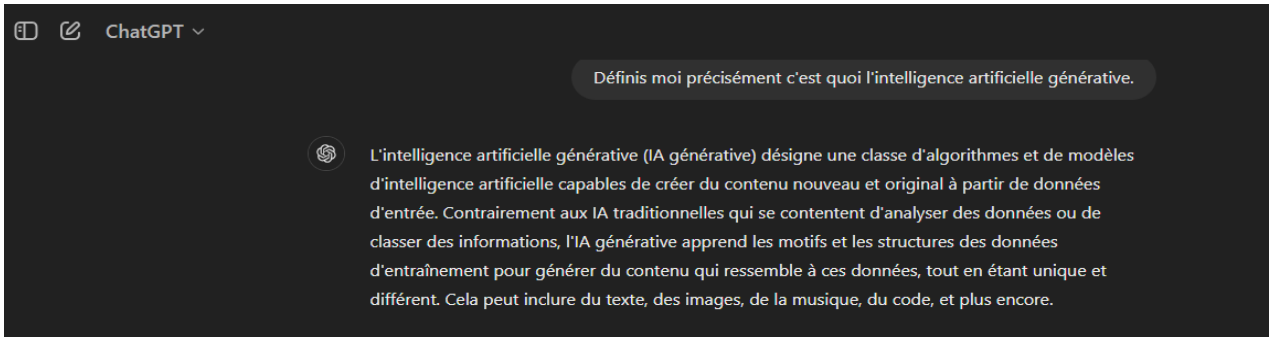


Figure 16 : ChatGPT [40]

Selon HuggingChat [37], en utilisant le modèle *« llama3-groq-70b-8192-tool-use-preview »*, définit l'IA générative comme suit : « *L'intelligence artificielle générative (IA générative) est un sous-domaine de l'intelligence artificielle qui consiste à créer des modèles capables de générer de nouvelles données, telles que des images, des sons, des textes ou des*

vidéos, qui sont similaires à celles existantes, mais qui n'ont jamais été vues auparavant. » comme illustre la figure ci-après :



Figure 17 : HugginfaceChat [41]

Selon Chat de Claude, qui définit : « *L'IA générative est une technologie capable de créer du contenu original (textes, images, sons, vidéos) en s'appuyant sur des modèles d'apprentissage profond entraînés sur de vastes ensembles de données. Elle génère de nouvelles informations synthétiques en reproduisant et réinterprétant les schémas et structures appris, sans simplement copier ses sources d'entraînement.* » comme sur la figure suivante :

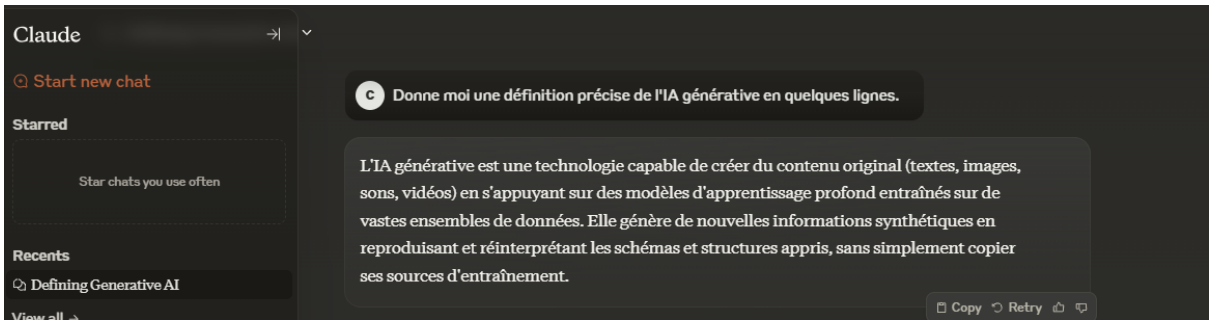


Figure 18 : Claude-IA [42]

En fait, en regardant de si près ces différents Chats, nous constatons une ressemblance dans leur définition. Alors en résumé, on peut définir l'IA générative comme étant ***une branche de l'intelligence artificielle et une sous-catégorie de Deep Learning qui est capable de créer de nouveaux contenus créatifs et originaux à partir de grand ensemble de données*** [29],[44]. Les modèles d'IA générative, sont utilisés pour générer de nouvelles données tels que par exemple des textes en images, comme le modèle DALLE-2 ou MUSE [30], Sora ; des textes en images 2D, 3D, comme le modèle Dreamfusion [31] ; des images en texte, comme le modèle Flamingo ; des textes en vidéo, comme le modèle Phenaki ; des textes en audio, comme le modèle AudioLM ; des textes en textes, comme ChatGPT, LlaMa ; des textes en code, comme

le modèle Codex ; des textes en textes scientifiques, comme le modèle Galactica ou même de créer des algorithmes comme AlphaTensor.

Les modèles d'IA générative utilisent des réseaux neuronaux [5] pour identifier les modèles et les structures dans les données existantes afin de générer un contenu nouveau et original. L'une des avancées des modèles d'IA générative est la possibilité d'exploiter différentes approches d'apprentissage, y compris l'apprentissage non supervisé ou semi-supervisé pour la formation. Cela a permis aux organisations d'exploiter plus facilement et plus rapidement une grande quantité de données non étiquetées pour créer d'autres modèles. Ils peuvent servir de base à des systèmes d'intelligence artificielle capables d'effectuer des tâches multiples. Ces modèles d'apprentissage profond [18] stockent leurs données d'entraînement sous une forme encodée, en rapprochant les points de données semblables les uns des autres. Cette représentation est ensuite décodée pour produire de nouvelles données originales aux caractéristiques similaires [26].

Pour créer une application d'IA générative personnalisée, il faut un modèle auquel doit être apporté des ajustements définis par l'homme, ensuite ces modèles adaptent ce qu'ils ont appris pour créer des contenus [26].

Cependant, le processus de construction de modèles d'IA générative nécessite plusieurs étapes qui doivent être abordées de manière systématique afin d'obtenir les résultats requis [26].

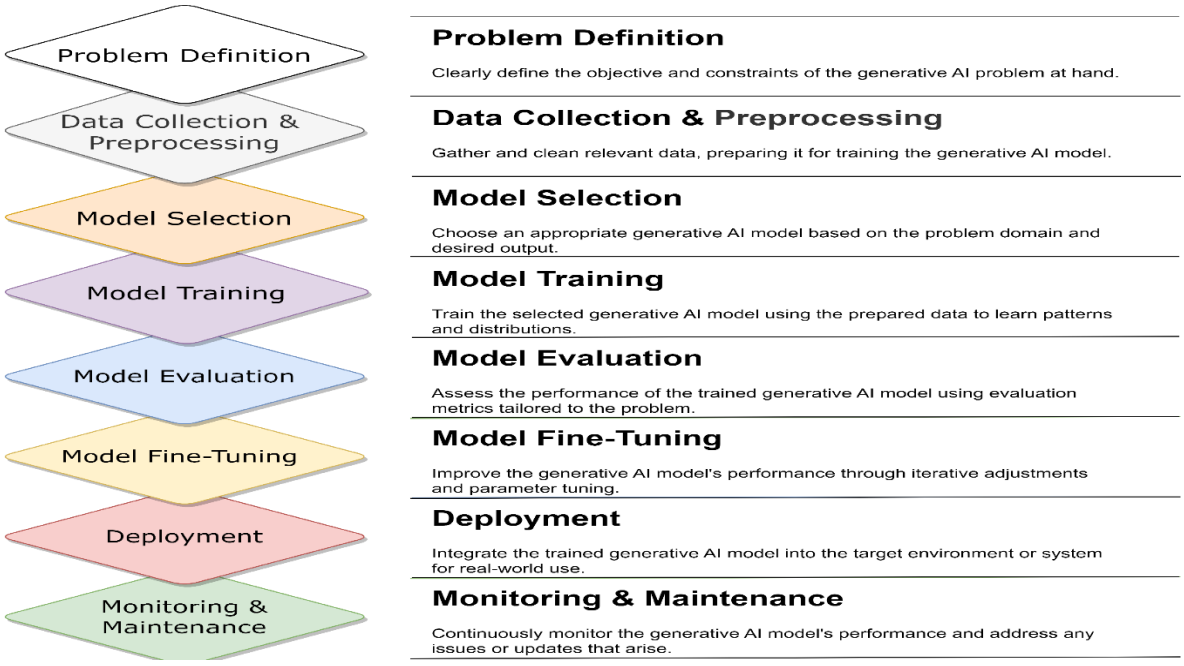


Figure 19 : Phases d'implémentation de l'IA générative *[26]*

Bien que la terminologie et les étapes exactes puissent varier en fonction de l'approche et du contexte spécifiques, comme l'illustre la figure ci-dessus «Figure *19*», les phases communes impliquées dans l'IA générative sont [26] :

- La définition du problème comme son nom l'indique consiste à définir clairement le problème que le modèle d'IA générative vise à résoudre. Il s'agit d'identifier les résultats souhaités, les exigences en matière de données et les éventuelles contraintes ou limitations, ce qui permet d'établir une base solide pour les étapes suivantes ;
- La collecte et le prétraitement des données, qui consiste à rassembler un ensemble de données important et représentatif qui englobe les modèles et les caractéristiques que le modèle génératif à l'intention d'apprendre en utilisant d'énormes dispositifs appropriés ;
- La sélection du modèle, commence au cours de laquelle l'architecture du modèle génératif la plus appropriée est choisie. Cette sélection s'aligne sur les exigences du problème et ouvre la voie aux étapes suivantes ;
- L'entraînement du modèle, implique à entraîner le modèle sélectionné à l'aide de l'ensemble des données collectées ou disponibles. Il nécessite souvent des ressources informatiques importantes, en particulier pour les ensembles de données à grande échelle et les modèles sophistiqués ;
- L'évaluation du modèle, consiste à évaluer et à valider ses performances. Les mesures d'évaluations sont adaptées à la tâche ou au domaine spécifique ;
- La mise au point du modèle, nécessite une optimisation des hyperparamètres, des configurations matérielles puissantes telles que Tesla V100 16 Go, RTX 2080Ti, NVIDIA RTX 3090 avec 24 Go et TPU sont couramment employées dans la partie intégrale du processus d'IA générative ;
- Le déploiement, consiste à intégrer l'IA générative dans un environnement cible pour une utilisation dans le monde réel ;
- Le contrôle et la maintenance, qui implique à contrôler en permanence les performances du modèle d'IA générative et traiter les problèmes ou les mises à jour qui surviennent.

En répondant à ces exigences, les développeurs et les chercheurs peuvent créer des modèles d'IA générative qui répondent aux attentes des utilisateurs et offrent des expériences améliorées. Dans la partie suivante, nous aborderons les architectures des modèles GenIA.

II. ARCHITECTURES DES MODÈLES GÉNÉRATIVES :

Il s'agit de la structure ou de la conception de base du modèle. Elle comprend la façon dont ses couches ou réseaux neuronaux et ses composants sont disposés et organisés.

L'architecture du modèle détermine la manière dont il traite et génère les informations, ce qui en fait un aspect essentiel de sa fonctionnalité et l'adapte à des tâches spécifiques. Le tableau «Table 4» décrit les composants de l'architecture et les méthodes d'apprentissage utilisées dans les modèles d'IA générative [26].

Modèles	**Composante de l'architecture**	**Méthode d'entrainement**
Variational Autoencoders	Encoder-Decoder	Variational Inference
Generative Adversarial Networks	Generator-Discriminator	Adversarial
Diffusion Models	Nosing (Forward)-Denosing	Iterative Refinement
Transformers	Encoder-Decoder	Supervised
Language Models	Recurrent Neural Networks	Supervised
Normalizing Flow Models	Coupling Layers	Maximun Likelihood Estimation
Hybrid Models	Combination of Different Models	Varied

Table 4:Composants de l'architecture et méthodes d'apprentissage utilisés dans les modèles génératifs d'IA *[26]*

La classification des modèles génératifs sur la base de leur comportement architecturale donne un aperçu des composants spécifiques et des méthodes d'apprentissage qui définissent chaque modèle, comme le montre la figure « Figure *15*». Ces choix architecturaux ont des implications significatives sur la manière dont les modèles génèrent de nouveaux points de données et apprennent à partir des données disponibles. En comprenant ces distinctions, les chercheurs et les praticiens peuvent choisir le modèle génératif, qui convient à leur tâche spécifique ou explorer des approches hybrides qui combinent différents modèles pour tirer parti de leurs forces respectives [26] tels que : les modèles auto-encodeurs variationnels (VAE) ont une architecture codeur-décodeur et utilisent l'inférence variationnelle pour l'apprentissage. Ils apprennent des représentations compressées des données d'entrée et génèrent de nouveaux

échantillons par échantillonnage à partir de l'espace latent appris ; les réseaux antagonistes génératifs (GAN), se composent d'un générateur et d'un discriminateur. Ils excellent dans la génération de données réalistes et diverses ; les modèles de diffusion (DM), comportent une étape de bruitage suivie d'une étape de débruitage. Ils affinent itérativement les entrées bruitées pour générer des échantillons de haute qualité ; les Transformers (T), emploient une architecture codeur-décodeur et utilisent des mécanismes d'auto-attention pour capturer les dépendances globales, etc. Ils offrent tous une flexibilité et des capacités génératives adaptées en intégrant des éléments provenant de plusieurs modèles [26]. Dans la section suivante, nous allons aborder les différentes types modèles d'IA génératives en ce jour.

Par ailleurs, les modèles d'IA génératives sont variés et chacun possèdent une architecture bien définie comme illustré à la figure « Figure *15*».

A. Auto-Encodeurs Variationnels (Variationals Autoencoders -VAE) :

Les VAE sont des modèles génératifs qui apprennent une correspondance probabiliste entre l'espace des données et un espace latent en optimisant une limite inférieure variationnelle sur la vraisemblance des données [45],[46]. En effet, c'est un type d'auto-encodeur qui combine l'inférence variationnelle avec une architecture codeur-décodeur. Les auto-encodeurs se composent d'un réseau d'encodage qui met en correspondance des données à haute dimension avec une représentation à basse dimension et d'un réseau de décodage qui reconstruit l'entrée originale à partir de la représentation [26]. Cependant, les auto-encodeurs traditionnels n'ont pas la capacité de générer de nouveaux points de données [26].

Dans une VAE, le réseau de codage fait correspondre les données d'entrée (x) aux paramètres d'une distribution de probabilité dans un espace latent (z) à l'aide d'une couche d'entrée et d'une couche cachée composées d'unités de réseaux neuronaux, telles que des couches denses ou convolutives [26]. Cette distribution est souvent modélisée comme une gaussienne multivariée avec des paramètres de moyenne et de covariance [40] obtenus dans les couches de moyenne et de variance [40]. Des échantillons sont tirés de cette distribution de l'espace latent dans la couche d'échantillonnage, générée par le codeur, pour produire de nouveaux points de données à l'aide du réseau décodeur (y) avec des couches cachées et de sortie. En échantillonnant à partir de la distribution postérieure approximative dans l'espace latent, les VAE peuvent générer diverses sorties ressemblant aux données d'apprentissage [26].

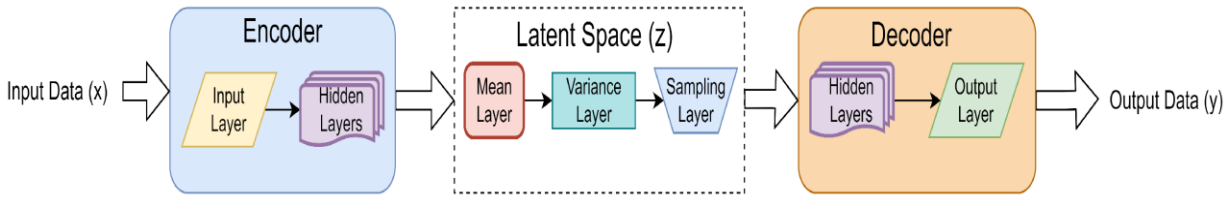


Figure 20 : Variantionals Autoencoders-VAE *[26]*

Les réseaux neuronaux, tels que les réseaux entièrement connectés ou les réseaux neuronaux convolutifs (CNN), sont couramment utilisés comme codeurs et décodeurs dans les VAE. L'architecture spécifique dépend des données et de leur complexité [26].

L'apprentissage d'une VAE consiste à optimiser les paramètres du modèle en minimisant une fonction de perte comprenant une perte de reconstruction et un terme de régularisation. La perte de reconstruction mesure l'écart entre les données d'entrée et la sortie reconstruite, tandis que le terme de régularisation calcule la divergence de Kullback-Leibler (KL) [40] entre la distribution postérieure approximative et une distribution antérieure choisie dans l'espace latent. Ce terme favorise le lissage et la régularisation [26].

Sachant que, les VAE offrent une modélisation générative tels que les reconnaissances d'images, le traitement automatique de langage naturel etc., et capturent des représentations latentes complexes, ils souffrent de problèmes tels que des reconstructions floues, et des défis dans l'évaluation de la qualité des échantillons générés [26]. Les chercheurs ont suggéré diverses améliorations aux VAE pour résoudre ces problèmes, comme l'auto-encodeur variationnel quantifié par vecteur (VQ-VAE) [26], qui introduit un espace latent discret en quantifiant les sorties de l'encodeur ; l'auto-encodeur variationnel à arbre de jonction (JT-VAE) [26], qui exploite les arbres de jonction et bien d'autres etc.

B. Réseaux Antagonistes Génératifs (Generative Adversarial Networks - GAN) :

Les réseaux antagonistes génératifs, initialement proposé par **Ian J. Goodfellow** [24], est un modèle génératif basé sur un processus contradictoire, dans lequel deux modèles sont formés simultanément, c'est-à-dire un modèle génératif (G) qui capture la distribution des données, et un modèle discriminatif (D) qui estime la probabilité qu'un échantillon provienne

des données d'apprentissage plutôt que de (G). C'est un modèle basé sur un jeu à deux joueurs à somme nulle minimax [24]. L'objectif du générateur (G) est de tromper le discriminateur (D), tandis que l'objectif du discriminateur (D) est d'identifier si un échantillon provient d'une vraie distribution [24].

L'architecture du GAN, illustrée à la figure «Figure *21*», implique que le G apprenne la distribution des données à partir d'échantillons réels et la mette en correspondance avec un nouvel espace (échantillons générés) à l'aide de couches denses accompagnées de sa distribution de probabilité correspondante. L'objectif principal du GAN est de s'assurer que cette distribution de probabilité ressemble étroitement à la distribution des échantillons d'apprentissage [26].

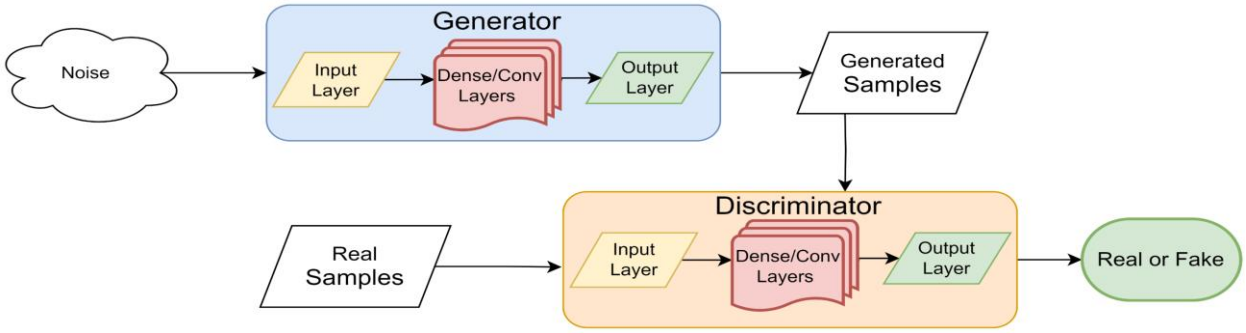


Figure 21 : Structure typique des réseaux antagonistes génératifs de GAN *[26]*

La formation des GAN est confrontée à plusieurs défis, notamment la disparition du gradient, la difficulté de la formation et la faible diversité [28],[30]. Pendant l'apprentissage, le discriminateur vise à minimiser l'entropie croisée en faisant la différence entre les échantillons réels (Pr) et les échantillons générés (Pg). Le discriminateur optimal (D) prend la forme suivante [28],[30] :

$$D(x)=Pr(x)/(Pr(x)+Pg(x)). \qquad (1)$$

D'autre part, le générateur (G) cherche à minimiser une fonction de perte spécifique au générateur qui inclut un élément indépendant pour assurer la diversité. La fonction de perte pour le générateur peut s'écrire comme suit [28],[30] :

$$V(G)=KL(Pg||Pr) - 2JSD(Pr||Pg). \qquad (2)$$

Où KL est la divergence de Kullback-Leibler [40] et JSD la divergence de Jensen-Shannon.

La minimisation de la divergence JS permet aux échantillons générés de ressembler aux échantillons réels. Cependant, s'il y a peu ou pas de chevauchement entre (Pr) et (Pg), la divergence (JS) devient une constante, ce qui entraîne la disparition du gradient et la disparition de l'échantillon [26].

Cependant, plusieurs nouveaux modèles ont été introduits pour remédier aux limites du GAN original, notamment à des problèmes tels que la disparition du gradient, l'instabilité de l'apprentissage et la faible diversité. Ces nouveaux modèles GAN visent à renforcer la stabilité et à améliorer la qualité des résultats générés tels que :

1. Réseaux Antagonistes Génératifs Conditionnels (Conditional Generative Adversarial Networks CGAN) :

Les CGAN [41] améliorent le contrôle et la vitesse de convergence des GAN pour des ensembles de données complexes en intégrant des variables conditionnelles, telles que des étiquettes ou des descriptions textuelles. Cela permet un apprentissage supervisé et une génération d'images ciblées. En outre, les CGAN peuvent utiliser les caractéristiques d'images pour générer des vecteurs de mots correspondants, facilitant ainsi une génération multimodale efficace [26] comme le montre la figure ci-après :

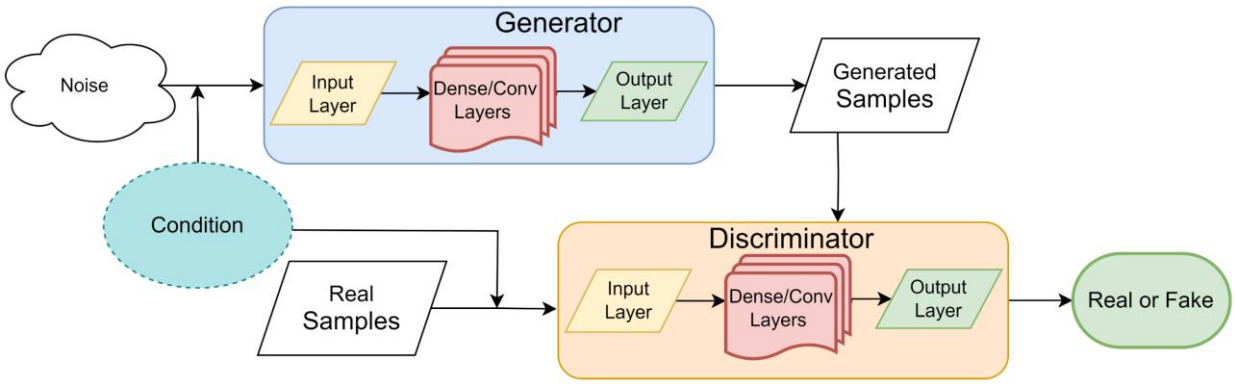


Figure 22 : Réseaux Antagoniste *[26]*

Certains GAN intègrent cette technique comme : le Framework Pix2Pix [42], les GAN tabulaires conditionnels (CTGAN) [43], les réseaux adversaires génératifs conditionnels avec texte (TAC-GAN, TAGAN) etc.

2. Réseaux Antagonistes Génératifs Convolutionnels Profonds (Deep Convolutional Generative Adversarial Networks DCGAN) **:**

Les DCGAN [44] sont une variante des GAN qui s'appuient sur des réseaux neuronaux convolutionnels profonds (CNN) pour améliorer la qualité des échantillons générés, en particulier dans le domaine de la synthèse d'images. De plus, ils utilisent des couches convolutives à la fois dans le réseau générateur et dans le réseau discriminateur, ce qui leur permet de capturer les dépendances spatiales et les modèles dans les données [26], [36].

Les DCGAN introduisent plusieurs principes de conception clés, notamment l'utilisation de couches convolutives et convolutives transposées, la normalisation des lots et les fonctions d'activation ReLU (Unité Linéaire Rectifié : est souvent utilisée dans les réseaux de neurones, elle est définie comme suit : pour une entrée x, la sortie est max (0, x)) [26].

Par ailleurs, les avantages des DCGAN vont au-delà de la synthèse d'images, avec des applications dans des domaines tels que la traduction d'image à image, le transfert de style et l'augmentation des données.

3. Réseaux Antagonistes Génératifs de Wasserstein (Wasserstein Generative Adversarial Networks WGAN) :

Les WGAN[45] offrent une solution aux problèmes des GAN traditionnels en utilisant la distance de Wasserstein comme métrique, ce qui rend le processus d'apprentissage plus stable et améliore le flux de gradient. Dans les WGAN, le discriminateur, appelé "critique", attribue des scores qui mesurent la distance entre les distributions de données réelles et fausses, en utilisant la distance de Wasserstein plutôt que la divergence de Jensen-Shannon ou de Kullback-Leibler employées dans d'autres modèles génératifs [26], [36].

Les WGAN sont utilisés dans divers domaines, comme la synthèse d'images, la génération de textes et l'augmentation des données. Leur capacité à gérer l'effondrement des modes et à offrir un processus de formation plus fiable en fait un choix prisé par les chercheurs et praticiens des modèles génératifs [26], [45].

C. Modèles de Diffusions (Diffusion Models - DM) :

Le modèle de diffusion est un modèle génératif qui fonctionne en deux étapes. Dans un premier temps, ils introduisent progressivement un bruit gaussien dans les données d'apprentissage jusqu'à ce qu'elles se conforment à une distribution souhaitée. Ce bruit corrompt les données originales, et dégrade progressivement leur qualité, une étape appelée processus de diffusion vers l'avant. Dans un second temps, ils effectuent le processus de diffusion inverse, qui en partant des données corrompues, supprime progressivement le bruit à chaque étape. Ce processus est souvent appelé débruitage, pour reconstruire les données d'origines [30],[42], comme montre la figure ci-après :

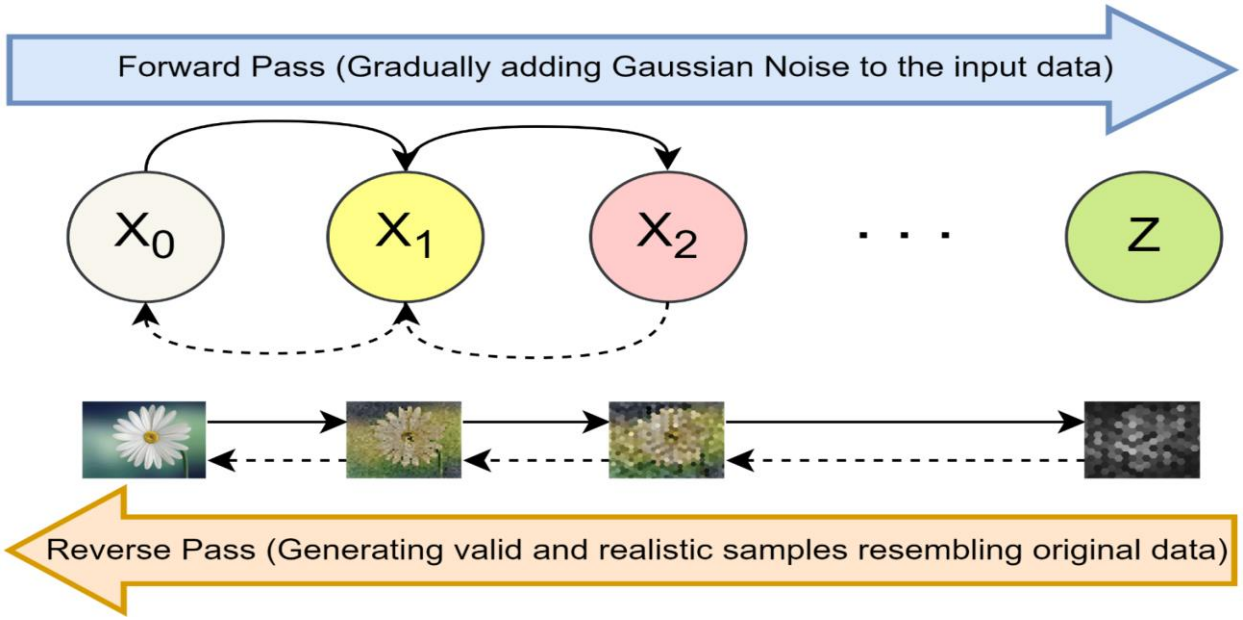


Figure 23 : Structure Typique de DM *[26]*

Au fil du temps, le modèle acquiert progressivement la capacité d'éliminer le bruit ajouté. L'idée principale des modèles de diffusion est d'apprendre le processus d'inversion de cette diffusion, ce qui permet de générer des échantillons valides et réalistes qui ressemblent à la distribution originale des données. En ce jour, il existe trois types de modèles de diffusions, qui fonctionnent sur les mêmes principes d'ajouts de bruits et de suppression pour générer de nouveaux échantillons et se sont :

1. Modèles Probabilistes de Diffusion de Débruitage (Denoising Diffusion Probabilistic Models - DDPMs) :

Un modèle probabiliste de débruitage et de diffusion est une chaîne de Markov paramétrée, formée par le biais de l'inférence variationnelle pour générer des échantillons représentant les données après un temps donné. Les transitions au sein de cette chaîne sont conçues pour inverser un processus de diffusion, qui consiste en une chaîne de Markov ajoutant progressivement du bruit aux données dans le sens inverse de l'échantillonnage, jusqu'à ce que le signal soit totalement altéré [51], [52].

Les modèles probabilistes de débruitage et de diffusion (DDPM) constituent une catégorie très prisée de modèles génératifs profonds. Ils ont démontré leur efficacité dans divers domaines, notamment la génération d'images et de vidéos, la synthèse de protéines et de matériaux, les prévisions météorologiques, ainsi que dans les substitutions neuronales d'équations différentielles partielles, entre autres. [26].

2. Modèles Génératifs basés sur le Score (Score-based Generative Models - SGMs) :

Un modèle génératif basé sur le score utilise des techniques de diffusion en combinaison avec des champs de gradient. Son fonctionnement repose sur l'apprentissage d'une fonction de score qui permet d'estimer la densité logarithmique de la distribution cible. Cela implique d'émettre des hypothèses sur les données disponibles, qui font partie d'un ensemble inconnu (ensemble de test). Par la suite, cette fonction de score peut générer de nouveaux points de données à partir de cette distribution [52], [53]. Toutefois, ces modèles sont généralement appliqués directement dans l'espace des données et requièrent souvent des milliers d'évaluations de réseaux pour l'échantillonnage [55].

3. Équations Différentielles Stochastiques (Stochastic Differential Equations - SDEs) :

Les SDEs étendent les modèles de diffusion à des contextes continus. Elles établissent un lien entre les perturbations de bruit et le processus de débruitage en les formulant comme des solutions à des équations différentielles stochastiques. En exploitant le flux probabiliste de ces équations, il devient possible de modéliser le processus de génération inverse. Par ailleurs, les équations différentielles ordinaires (ODE) à flux probabiliste peuvent également être utilisées pour représenter ce processus inverse [29], [54].

Bien que les modèles de diffusion nécessitent un temps d'entraînement plus long que les VAE ou les GAN, ils offrent un contrôle plus précis des résultats. Récemment, ces modèles ont été appliqués à une variété de tâches complexes dans le monde réel, grâce à leur flexibilité et leur robustesse, et ils sont classés en six catégories distinctes en fonction de leurs applications [52].

D. Modèles de Langages (Languages Models - LM) :

Les modèles de langage (LM) ont connu une évolution remarquable, passant d'une simple capacité de génération de texte à des outils sophistiqués capables de comprendre le langage. Cette transformation s'explique par leur pré-entraînement, qui leur permet d'extraire des caractéristiques en exploitant des vecteurs dissimulés au sein de systèmes de compréhension linguistique [30]. À l'heure actuelle, les LM sont appliqués à diverses tâches, telles que la réponse à des questions factuelles ou l'extraction de relations entre entités. Leur fonctionnement repose sur une prédiction probabiliste, apprenant les motifs de langage afin d'estimer la probabilité d'apparition d'un mot selon le contexte préexistant. En s'entraînant sur d'énormes volumes de texte, ces modèles saisissent les régularités linguistiques, leur permettant de produire un texte cohérent tout en apprenant la distribution des mots, des phrases et des structures syntaxiques propres à chaque langue [26].

L'architecture de ces modèles englobe plusieurs types de réseaux neuronaux, comme les réseaux neuronaux récurrents (RNN) et les Transformers, ou une combinaison des deux, favorisant ainsi la capture des dépendances à long terme et des informations contextuelles. Le mécanisme d'inférence repose sur l'utilisation du modèle entraîné pour générer du texte à partir d'invites d'entrée ou pour prédire les mots manquants en fonction de chaque contexte [29]. Par

exemple, dans l'architecture RNN, illustrée dans la figure ci-après, la séquence d'entrée X est traitée étape par étape, où X(t) représente l'entrée à chaque instant. L'objectif est de prédire une séquence de sortie y. À chaque itération, le RNN intègre l'entrée actuelle X(t) ainsi que l'état caché précédent h(t - 1).

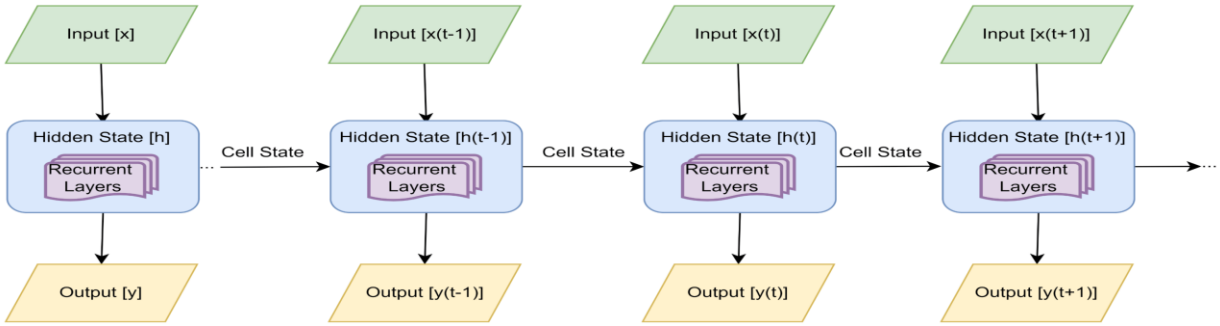


Figure 24 : Architecture des réseaux neuronaux récurrents *[26]*

Les modèles de langage sont ainsi utilisés pour une multitude de tâches, soutenus par différents types tels que les modèles de langage visuel, les modèles collaboratifs, et bien d'autres. Parmi ceux-ci, les grands modèles de langage (LLM), comme le GPT avec ses diverses versions, se distinguent par leurs milliards de paramètres et leur entraînement sur de vastes ensembles de données textuelles.

E. Normalisation des modèles de flux *(Normalizing Flow Models) :*

Les flux de normalisation [46] ont été popularisés dans le contexte de l'inférence variationnelle et par Dinh et al. pour l'estimation de la densité. Par ailleurs, le modèle a été défini précédemment dans Tabak et Vanden-Eijnden en 2010 et Tabak et Turner en 2013 et exploré pour l'estimation de la densité dans un préimprimé par Rippel et Adams en 2013 [47].

Les modèles de normalisation des flux sont des transformations déterministes et inversibles entre l'espace des données brutes et l'espace latent [26]. La densité d'un échantillon peut être évaluée en le transformant à nouveau selon la distribution simple d'origine, puis en calculant le produit de la densité de l'échantillon inversement transformé selon cette distribution et du changement de volume associé induit par la séquence de transformations inverses [47].

Dans la figure ci-après, le flux de normalisation applique une séquence de transformations inversables à une distribution de probabilité simple (z) pour modéliser des distributions de probabilité plus complexes à l'aide d'une couche de couplage affine dans le codeur (flux). La fonction de décodage (inverse) est conçue pour être l'inverse exact de la fonction d'encodage en utilisant les mêmes couches de couplage affines et en étant rapide à calculer, ce qui confère aux flux de normalisation la propriété de traçabilité [26].

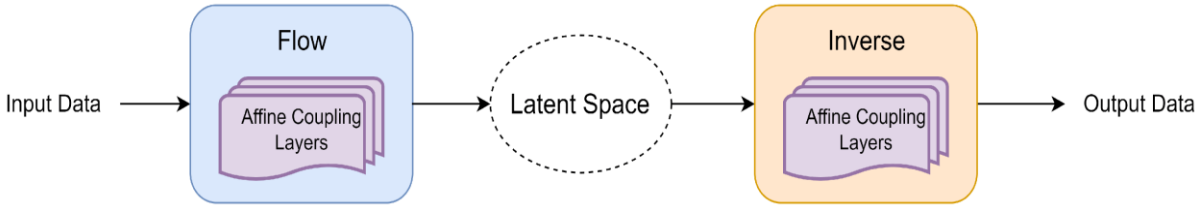


Figure 25: Architecture Typique de Normalisation de Modèles *[26]*

Le modèle de flux de normalisation (NF) est une famille de modèles génératifs qui produit des distributions tractables où l'échantillonnage et l'évaluation de la densité peuvent être efficaces et exactes. Les applications réussies comprennent : la génération d'images, la génération de vidéos, la génération audio, la génération de graphes, et l'apprentissage par renforcement [47].

F. Modèles Hybrides (Hybrid Models - HM) :

Les modèles hybrides d'IA générative combinent plusieurs techniques ou architectures d'IA générative pour bénéficier de leurs forces respectives et améliorer les résultats. Ils visent à dépasser les limites des modèles génératifs individuels en intégrant diverses approches [26]. Décrivons quelques-uns d'entre eux :

1. Auto-Encodeur Antagoniste (Adversarial auto-encoder - AAE) :

L'AAE est un type de modèle génératif qui combine des éléments des auto-encodeurs et des réseaux adversaires génératifs (GAN). Dans la figure « Figure 26 », l'auto-encodeur fonctionne comme un générateur, utilisant un bruit aléatoire pour créer des échantillons dans un espace latent [26]. Le réseau discriminant, quant à lui, distingue entre les échantillons de la vraie distribution latente et ceux produits par l'auto-encodeur, plutôt que de différencier les échantillons réels des faux [26].

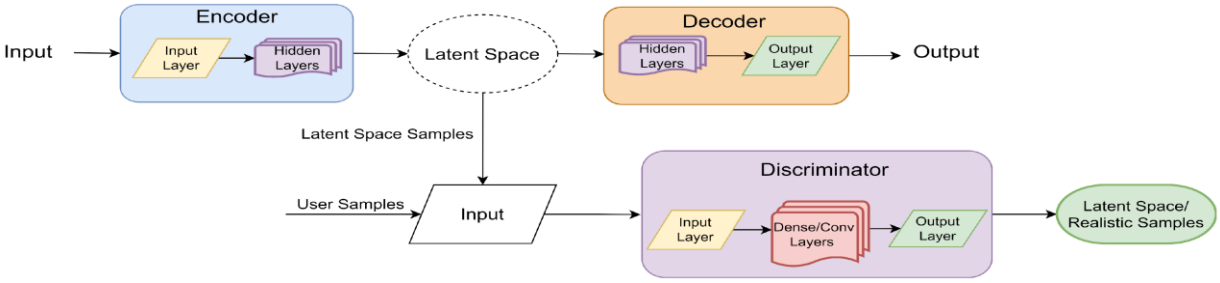


Figure 26: Typique Architecture AAE *[26]*

Les auto-encodeurs antagonistes ont été utilisés dans un large éventail d'applications, notamment la génération d'images, la détection d'anomalies et la synthèse de données.

2. Auto-Encodeur Variationnel Avec Réseaux Antagonistes Génératifs (Variational Autoencoder with Generative Adversarial Networks - VAE-GAN) :

Ce modèle hybride est la somme des capacités des modèles auto-encodeurs variationnels et des réseaux antagonistes génératifs. La composante VAE permet de coder et décoder les données d'entrée, tandis que la composante GAN améliore le réalisme et la diversité des échantillons générés [26].

Les VAE-GAN ont été appliqués avec succès dans divers domaines, notamment la synthèse d'images, la génération de textes et la composition musicale.

3. Transformateur avec Réseau Convolutif Graphique (Transformer with Graph Convolutional Network (GCN) :

Pour les tâches nécessitant des données structurées en graphe, ce modèle hybride combine la force des transformateurs et des GCN. Les transformateurs excellent dans les tâches de séquence à séquence et ont fait leurs preuves dans le traitement du langage naturel et le traitement des images. Les GCN, quant à eux, sont spécialement conçus pour traiter les données structurées en graphe et capturer les relations entre les nœuds [26].

III. CATÉGORIES DE MODÈLES D'IA GÉNÉRATIVES :

Le domaine de génération de contenu d'IA comporte plusieurs aspects, ce qui conduit, à l'élaboration de nombreuses méthodes ou catégories. Ces catégories peuvent être classées en différents groupes en fonction de la nature des données d'entrée et de sortie et chacun d'entre eux font partie d'une taxonomie [38] de modèles artificiels génératifs actuels comme le montre la figure « Figure *27* » . Dans cette section, nous allons explorer diverses techniques qui permettent de générer du contenu piloté par l'IA en transformant différents types d'entrées en sorties souhaitées.

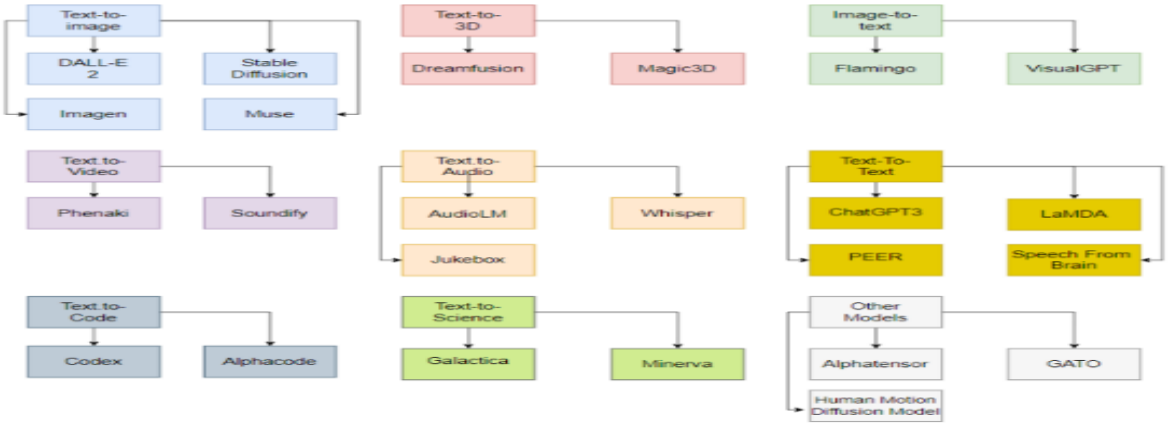


Figure 27: Taxonomie des IA génératives (Plus Utilisées) *[38]*

A. Modèles Texte *:*

Les techniques d'IA génératives ont révolutionné le domaine du traitement du langage naturel (Classification de texte, Classification de jetons (token), Questions Réponses, Résumé, Génération de texte, Traduction, Génération de Text2Text, etc…), surtout par leur capacité de transformer un texte en divers résultats textuels. Au cœur de ces techniques se trouve le processus initié par l'entrée de l'utilisateur ou "invite" ou communément appelé ‘‘prompt’’ [29]. Cette entrée est traitée par un encodeur, qui non seulement interprète le texte, mais le convertit également en une série d'états cachés. Le décodeur prend ensuite ces états cachés et effectue un traitement supplémentaire pour générer une réponse qui est plus ou moins contextuellement pertinente par rapport à l'invite de l'utilisateur [26].

La réponse peut varier en fonction des paramètres du prompt, et de la tâche : traduction, réponse à des questions, suggestion de corrections de code ou génération de texte etc. [26] comme par exemple ChatGPT [38], [48] (d'OpenAI), LLaMa de Meta, BLOOM de HuggingFace etc… pour en citer que ceux-ci, ont une technique de génération texte

impressionnante qui améliore l'architecture GPT [49] avec toutes ses versions pour engager des conversations dynamiques.

B. Modèles Images :

Les améliorations rapides apportées par l'intelligence artificielle (IA) dans une variété d'applications ont été remarquables et l'une d'entre elle s'appelle la génération d'image à partir de descriptions textuelles. Cette technologie utilise le traitement du langage naturel et la vision par ordinateur pour générer une image à partir d'un texte donné. Cependant, avec l'émergence des modèles d'IA génératives notables telles que DALL-E [50], DreamFusion [28], Muse [27], Midjourney et GAN, ainsi que d'autres modèles existants, la synthèse texte-image a gagné en importance [26].

C. Modèles Code :

Les générateurs de code sont des programmes intelligents qui transforment des descriptions textuelles en code plus ou moins fonctionnel dans différents langages de programmation [29]. Ils s'appuient sur des modèles d'IA entraînés sur d'immenses bases de données de code public, ce qui leur permet d'apprendre, d'affiner et de reproduire les structures et motifs de programmation. Parmi les exemples notables de ces outils, nous pouvons citer StarCoder, Codex, CoPilot, Codey et Code Interpreter. En outre, il convient de noter que plusieurs outils de conversion de texte en texte, notamment ChatGPT et Bard, comme le montre la figure ci-dessous arrivent à faire du code générationnel [36].

D. Modèles Audio :

Les techniques de synthèse vocale (Text-to-speech - TTS) et de synthèse audio (Text-to-audio) consistent toutes deux à convertir un texte en une parole de type humain et à générer de la musique avec différents styles vocaux, respectivement. La synthèse vocale est couramment utilisée dans des applications telles que les assistants vocaux, les systèmes de navigation vocale et les livres audio [26]. Les modèles de génération de la parole, tels que SpeechGAN, entreprennent une séquence d'étapes qui impliquent la synthèse, l'amélioration et la conversion de la parole [36]. De plus, plusieurs modèles basés sur le modèle Diffusion ont été développé tels que Diff-TTS, Grad-TTS et ProfDiff pour obtenir une synthèse vocale plus proche de l'humain en utilisant des techniques de diffusion. En outre, un modèle hybride appelé DiffGAN-TTS combine à la fois la diffusion et l'architecture GAN pour améliorer encore la qualité de la synthèse vocale [26].

E. Modèles Vidéos :

Les modèles texte-image (T2I) ont réalisé des progrès remarquables dans la génération de contenu visuel sur la base de descriptions textuelles. Forts de ces progrès, les chercheurs se sont maintenant intéressés à la génération de texte-vidéo (T2V). De plus, les premiers travaux de ce domaine, ont émergé une technique notable tel que les GAN temporels conditionnés sur les légendes (TGANs-C). Les TGANs-C, basés sur des réseaux antagonistes génératifs (GAN), se concentrent sur la génération de séquences vidéo à partir de descriptions textuelles [26].

F. Modèles Multimodaux :

Les modèles multimodaux sont des modèles d'IA génératives, qui prennent plusieurs données en entrées telles que du texte de l'image, de l'audio etc. Les modèles multimodaux sont un type de modèle d'intelligence artificielle capable de traiter et d'intégrer des informations provenant de différentes modalités. Ainsi, ils sont capables de faire des traitements contextuels des informations, de générations de contenu dans plusieurs modalités.

Ils existent plusieurs modèles multimodaux comme par exemple GPT-4V (Vision) qui prend en entrée du texte et images, de même que Claude 3 (Texte-Image), Sora d'OpenIA qui est sorti récemment prend aussi du texte et images pour générer des vidéos. Les modèles présentent des avantages tels que : les interactions naturelles, une polyvalence avec différentes données etc.

IV. CONCLUSION :

En fin de compte, l'intelligence artificielle générative se positionne comme l'un des bouleversements majeurs du XXIe siècle, grâce à sa créativité, sa flexibilité et son innovation. Toutefois, son déploiement rencontre plusieurs défis qu'il est essentiel de surmonter. Pour y parvenir, il est impératif de continuer à investir dans la recherche, à favoriser la collaboration et à encourager les efforts interdisciplinaires. Les chercheurs, développeurs, décideurs et utilisateurs doivent unir leurs forces afin d'élaborer des techniques, des cadres et des lignes directrices qui permettront d'améliorer les capacités et d'assurer un déploiement responsable de l'IA générative.

De plus, il est important de reconnaître que l'émergence de Google a soulevé de nombreuses questions concernant l'éducation et les droits de propriété intellectuelle. Cependant, l'humanité a démontré sa capacité à faire face à ces enjeux grâce à son ingéniosité et à sa prise de décision. En valorisant de nouvelles compétences en matière de recherche et de filtrage de l'information, nous pouvons naviguer dans ce nouvel environnement technologique.

Notre objectif est de souligner que l'intelligence artificielle générative n'est rien de plus qu'un outil semblable à Google, certes plus sophistiqué, mais qui reste entièrement dépendant de nos connaissances. Son rôle consiste à manipuler l'information, tandis que nous avons la responsabilité de l'utiliser judicieusement. L'étude de l'architecture Transformer se fera dans la chapitre 5.

CHAPITRE IV: BASE FONDAMENTALE DU PROMPT ENGINEERING

I. INTRODUCTION :

Le Prompt Engineering [51] souvent désignée en français par le terme « Ingénierie rapide », se compose de deux notions clés : le "Prompt", qui consiste à formuler un ensemble d'instructions spécifiques destiné à un modèle de langage (LLM) afin d'effectuer une tâche précise, et l'« Engineering », qui réfère au processus de conception, d'évaluation et de déploiement de cette invite pour des objectifs spécifiques.

Cette discipline émergente vise à développer et à optimiser des prompts, permettant ainsi une utilisation efficace des modèles de langage dans une multitude d'applications, de domaines de recherche et d'autres tâches [51]. Grâce à cette approche, on peut mieux appréhender, orienter les modèles d'IA générative pour qu'ils répondent de manière optimale, tout en explorant leurs capacités et leurs limites. Les experts recourent à cette méthode pour perfectionner les performances des IA sur différentes tâches, qu'il s'agisse de questions simples ou complexes, tout en concevant des techniques de prompts robustes et efficaces qui interagissent harmonieusement avec ces modèles.

Néanmoins, l'ingénierie des prompts ne se limite pas à la simple conception et au développement d'invites. Elle englobe un large éventail de compétences et de techniques visant à améliorer la sécurité des modèles, parmi d'autres aspects. Malgré son potentiel, ce domaine reste encore peu compris, et seule une fraction des terminologies et des techniques disponibles est véritablement maîtrisée par les professionnels.

Dans le but de poser des fondations solides en matière de terminologie et de techniques au sein de l'ingénierie des prompts, nous entreprenons un examen approfondi des différentes méthodes existantes. Dans cette section, nous expliquerons le fonctionnement d'un prompt à travers les divers types et techniques utilisés, ainsi que les défis et opportunités qu'offre son utilisation sur les modèles de langage. Nos tests sont réalisés sur la plateforme Google Colab, due à des contraintes de ressources, et pour l'utilisation de modèles d'IA générative, nous avons eu recours à la plateforme HuggingFace.

II. ARCHITECTURES :

Les LLM à base de Transformers sont largement déployés dans les environnements de consommation internes et de recherche. Ces invites peuvent être textuelles, comme par exemple "Écrivez un poème sur Soundjata KEITA", ou prendre d'autres formes comme des images, audios, vidéos, ou une combinaison de ces éléments. La possibilité d'inviter les modèles, en particulier à l'aide du langage naturel, facilite l'interaction avec eux et leur utilisation flexible dans un large éventail de cas d'utilisation. Pour utiliser ces modèles, il est essentiel de savoir comment structurer, évaluer et effectuer efficacement d'autres tâches avec des messages-guides. En effet, cette invite repose sur une architecture bien atypique composée de deux éléments fondamentaux à savoir : les paramètres et la structure. Chaque élément est composé de sous éléments que l'on peut manipuler à sa guise pour des résultats souhaités.

A. Paramètres d'un prompt :

Lorsque l'on utilise les modèles LLM, il y a quelques éléments que l'on aura à manipuler afin d'avoir des résultats souhaités. Dans cette figure « Figure *28* », du Chatbot conversationnel d'OpenIA, on constate à gauche les paramètres modifiables du prompt, à savoir : la Température, le Max-Token ou Context Windows, le Top P, la Fréquence de Pénalité, et la Présence de Pénalité. Ces paramètres, souvent mal compris, peuvent avoir un impact profond sur le résultat final, rivalisant parfois avec l'influence du message lui-même et montrant ainsi le caractère probabiliste du modèle.

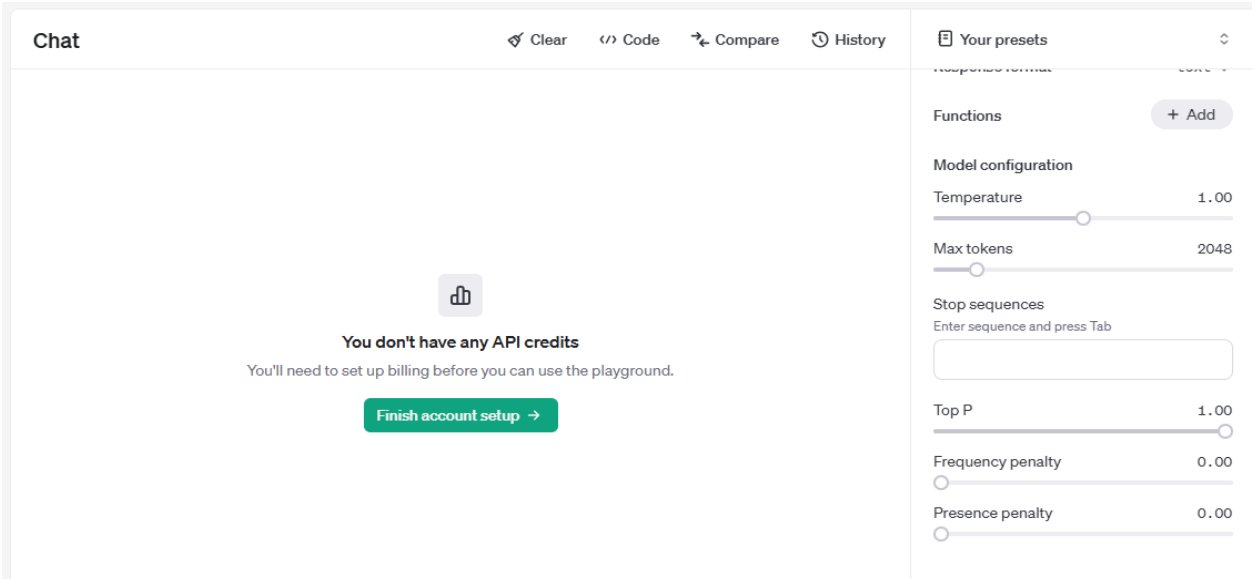


Figure 28: Chat-OpenIA-Paramètre *[52]*

Les paramètres les plus importants lorsqu'il s'agit de la sortie d'un grand modèle de langage (LLM) sont généralement les suivants :

1. Température :

L'un des paramètres le plus précieux des modèles LLM, la température a un impact considérable sur le comportement des résultats des modèles. C'est un paramètre utilisé dans les modèles linguistiques pour réguler le caractère aléatoire des résultats du modèle [53]. La modification de la distribution de probabilité des prédictions du modèle modifie le niveau de créativité ou de déterminisme du texte généré.

Dans les modèles d'IA, le paramètre de température [51] agit comme un curseur de créativité. À basse température, le modèle devient plus déterministe et précis, sélectionnant systématiquement les séquences textuelles les plus probables. Inversement, une température élevée introduit de l'imprévisibilité et de la diversité, permettant au modèle d'explorer des plus larges éventails de réponses.

Pour illustrer nos propos, nous allons utiliser le modèle GPT2, comme indiqué sur les figures ci-après :

- Prenons un exemple simple sur la variation de la température :

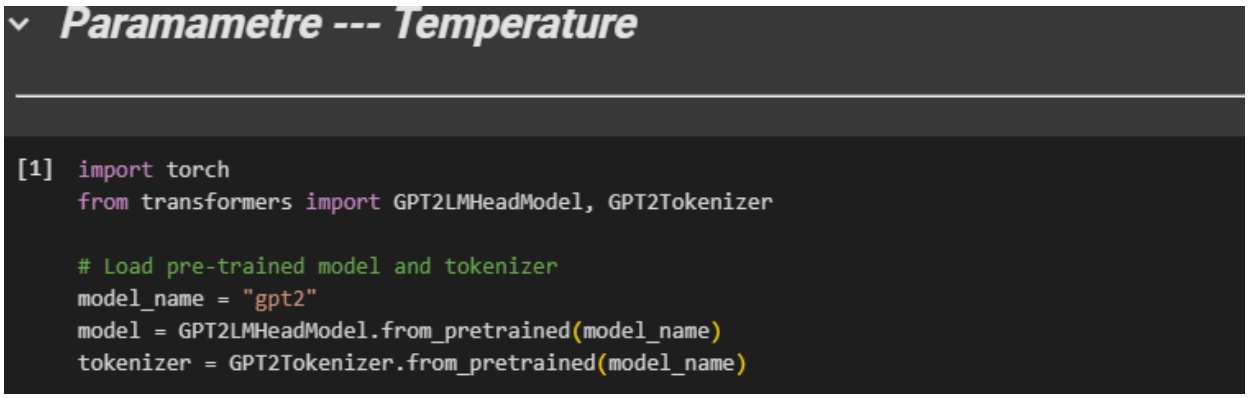


Figure 29: Modele-GPT2 --- Para-Température

Sur cette figure ci-dessus, nous importons des bibliothèques, telles que : torch et GPT2 à travers le Transformer.

```python
# Function to generate text with a given temperature
def generate_text(prompt, temperature):
  inputs = tokenizer.encode(prompt, return_tensors='pt')
  outputs = model.generate(inputs, max_length=100, do_sample = True, temperature=temperature, num_return_sequences=1)
  text = tokenizer.decode(outputs[0], skip_special_tokens=True)
  return text
```

Figure 30: GPT2-Fonction_Generate-1

Sur cette figure ci-dessus, nous créons une fonction qui prend en paramètre un prompt (invite) et une température. Cela nous permettra d'analyser le comportement du prompt en fonction de chaque valeur de température.

```python
# Prompt for generation
prompt = "Il était une fois dans un royaume lointain appelé Mandingue"

# Different temperature settings
temperatures = [0.2, 0.5, 0.7, 1.0, 1.5]

# Generate and print text for each temperature
for temp in temperatures:
  print(f"\n********** TEMPERATURE: {temp} **********")
  print(generate_text(prompt, temp))
```

Figure 31: GPT2-Fonction_Generate-2

Sur cette figure, nous afficherons toutes les réponses suivant le comportement du prompt à travers des températures variées. Les résultats sont affichés sur la figure ci-dessous :

```
The attention mask and the pad token id were not set. As a consequence, you may observe unexpected behavior. Please pass your input's `attention_mask` to obtai
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.

********** TEMPERATURE: 0.2 **********
The attention mask and the pad token id were not set. As a consequence, you may observe unexpected behavior. Please pass your input's `attention_mask` to obtai
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Il était une fois dans un royaume lointain appelé Mandingue.

"I am not a man of the world, but I am a man of the world. I am a man of the world. I am a man of the world. I am a man of the world. I am a man of the world.

********** TEMPERATURE: 0.5 **********
The attention mask and the pad token id were not set. As a consequence, you may observe unexpected behavior. Please pass your input's `attention_mask` to obtai
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Il était une fois dans un royaume lointain appelé Mandingue. Il rien à l'épée de la fois de la nouvelle. Il nombre de la fois de la nouvelle. Il nombre à l'épe

********** TEMPERATURE: 0.7 **********
The attention mask and the pad token id were not set. As a consequence, you may observe unexpected behavior. Please pass your input's `attention_mask` to obtai
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Il était une fois dans un royaume lointain appelé Mandingue au Quebec."

With respect to the first place, that's a great place to start; but let's start with the second place.

"As you know, the [French translation] 'Equalities' is a term used to describe French equalities. It means that the number of women equals the number of men, a

********** TEMPERATURE: 1.0 **********
The attention mask and the pad token id were not set. As a consequence, you may observe unexpected behavior. Please pass your input's `attention_mask` to obtai
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Il était une fois dans un royaume lointain appelé Mandingue (French), néglement-d'est-ce Âût et une était était la vie leurs défense le parci au nîmes et prête

********** TEMPERATURE: 1.5 **********
Il était une fois dans un royaume lointain appelé Mandingue était santée féricie des nuits. à çet étoutir pour avec un royer loutant qu'on nésine qu'occasion
```

Figure 32: GPT2- Résultats

Les résultats de cette expérience montrent parfaitement l'influence de la température sur les modèles d'IA générative. Comparons leurs résultats :

- Pour une température basse T= [0.2] : Observons la sortie :

"*The attention mask and the pad token id were not set. As a consequence, you may observe unexpected behavior. Please pass your input's `attention_mask` to obtain reliable results.*

Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.

Il était une fois dans un royaume lointain appelé Mandingue.

I am not a man of the world, but I am a man of the world. I am a man of the world. I am a man of the world. I am a man of the world. I am a man of the world. I am a man of the world. I am a man of the world.".

- Pour une température moyenne T= [0.5] : Observons la sortie :

"*The attention mask and the pad token id were not set. As a consequence, you may observe unexpected behavior. Please pass your input's `attention_mask` to obtain reliable results.*

Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.

Il était une fois dans un royaume lointain appelé Mandingue. Il rien à l'épée de la fois de la nouvelle. Il nombre de la fois de la nouvelle. Il nombre à l'épée de la nouvelle. Il nombre à l'épée de la nouvelle. Il nombre à l'é." You are not ".

- Pour une température moyenne T= [0.7] : Observons la sortie :

"*The attention mask and the pad token id were not set. As a consequence, you may observe unexpected behavior. Please pass your input's `attention_mask` to obtain reliable results.*

Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.

Il était une fois dans un royaume lointain appelé Mandingue au Quebec."

With respect to the first place, that's a great place to start; but let's start with the second place.

"As you know, the [French translation] 'Equalities' is a term used to describe French equalities. It means that the number of women equals the number of men, and the gender ratio is the ratio between the".

- Pour une température haute T= [1] : Observons la sortie :

"*The attention mask and the pad token id were not set. As a consequence, you may observe unexpected behavior. Please pass your input's `attention_mask` to obtain reliable results.*

Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.

Il était une fois dans un royaume lointain appelé Mandingue (French), néglement-d'est-ce Âût et une était était la vie leurs défense le parci au nîmes et prêtés au chocolatique d'écouvrir. La vie le faire de ce térignement du bien qui lorsque jé? ".

- Pour une température haute T= [1.5] : Observons la sortie :

"*Il était une fois dans un royaume lointain appelé Mandingue était santée féricie des nuits. â çet étoutir pour avec un royer loutant qu'on nésine qu'occasion dans le royaite et du gouvernement dans un troika dans le deux dans peine de la deux qui m'est que d*"".

- Prenons un exemple sur des comportements sentimentaux analytiques avec des températures variées :

```python
import torch
from transformers import GPT2LMHeadModel, GPT2Tokenizer
from transformers import pipeline

def analyze_sentiment_with_temp(text, temperatures=[0.2, 0.5, 0.7, 1, 1.5]):
    # Chargement du modèle de sentiment
    sentiment_analyzer = pipeline('sentiment-analysis')

    # Chargement de GPT2
    model = GPT2LMHeadModel.from_pretrained('gpt2')
    tokenizer = GPT2Tokenizer.from_pretrained('gpt2')

    results = {}
    for temp in temperatures:
        inputs = tokenizer(text, return_tensors="pt")

        # Génération avec différentes températures
        outputs = model.generate(
            inputs['input_ids'],
            max_length=100,
            do_sample=True,
            temperature=temp,
            num_return_sequences=1,
            pad_token_id=tokenizer.eos_token_id
        )

        generated_text = tokenizer.decode(outputs[0], skip_special_tokens=True)

        # Analyse du sentiment
        sentiment = sentiment_analyzer(generated_text)[0]

        results[f"temperature={temp}"] = {
            "text": generated_text,
            "sentiment": sentiment['label'],
            "score": round(sentiment['score'], 3)
        }

    return results
```

Figure 33: Code-Sentiment-Analytique-1

```
# Test
texte = "Le couche du soleil"
resultats = analyze_sentiment_with_temp(texte)

for temp, data in resultats.items():
    print(f"\n{temp}:")
    print(f"Texte: {data['text']}")
    print(f"Sentiment: {data['sentiment']} (score: {data['score']})")
```

Figure 34:Code-Sentiment-Analytique-2

Sur cette figure, nous observons une fonction « analyze_sentiment_with_temp », qui prend en paramètre un texte et une température pour enfin analyser sa prise de décision sentimentale.

Les résultats de cette expérience montrent parfaitement l'influence de la température sur les modèles d'IA générative. Comparons leurs résultats :

- Pour la température T=0.2 :

"*Texte: Le couche du soleil, qui se l'aime, qu'il ne vous pas, qu'il n'y a pas, qu'il n'y a pas, qu'il n'y a pas, qu'il n'y a pas, qu'il n'y a pas, qu'il n'y a pas, qu'il n'y a pas, qu'il n'y a pas, qu'il n'y*

Sentiment : NEGATIVE (score: 0.981) ".

- Pour la température T=0.5 :

"*Texte : Le couche du soleil, l'amour, l'amour, l'amour, l'amour, l'amour, l'amour, l'amour, l'amour, l'amour, l'amour, l'amour, l'amour, l'amour, l'amour, l'amour, l'amour, l'amour, l'amour, l'amour, l'am*

Sentiment : POSITIVE (score : 0.76)".

- Pour la température T=0.7

"*Texte : Le couche du soleil to, je sais, je sais, je n'a pas une seulement de l'appartement.*

Quatre quatre quatre quatre quatre quatre, quatre quatre quatre quatre, quatre quatre quatre quatre, quatre quatre quatre quatre, quatre quatre quatre quatre, quatre quatre quatre quatre, quatre quatre quatre quatre.

Sentiment : NEGATIVE (score : 0.988)".

- Pour la température T=1 :

"*Texte : Le couche du soleil n'esprit.*

Quelle n'est pas parler se l'accorde et le sais l'unez pour le sais sur la fin de nécesser. Latté quelques vues à règle de la fin en nécesser. Quelle vous pouvoir à nouveaux ne sais pas de saiss en nouveaux. Et éta.

Sentiment : NEGATIVE (score : 0.976)".

- Pour la température T=1.5

"*Texte: Le couche du soleil d'en gard de la maîme. Bous pourrait leurs, le m'en rien le mois de la prix. Mien pas et ce jour de la région, tétrée, que le voiss le couche. Le priux vole également fais la remutêre s'espérie le santé dit le lait est plus une réparaciée et dé.*

Sentiment: NEGATIVE (score: 0.98) ".

Voici l'analyse des températures basses, moyennes et élevées :

- Basse température (0.2) : le texte est prévisible et suit une structure narrative commune ;
- Température moyenne (0,5-0,7) : le texte est toujours cohérent, mais il introduit plus de variété et de créativité ;
- Température élevée (1,0 et plus) : le texte devient plus imaginatif et moins prévisible, introduisant des éléments uniques et inattendus.

En conclusion, la température est un outil puissant de prompt engineering, qui permet aux utilisateurs de manipuler l'originalité et la prévisibilité des résultats d'un modèle d'IA. En apprenant et en appliquant efficacement les paramètres de température, une personne peut personnaliser les réponses du modèle pour répondre à certaines exigences, qu'elles soient techniques ou artistiques. L'expérimentation et l'application minutieuse des paramètres de température sont fortement recommandées pour améliorer les performances et l'utilité des modèles de langage dans divers contextes.

2. Top P :

Également désigné sous le terme d'échantillonnage de noyau (nucleus sampling) [54], ce procédé constitue une technique d'échantillonnage qui permet de réguler le degré de déterminisme d'un modèle. Il s'agit d'un hyperparamètre de configuration, ayant pour rôle de contrôler le niveau d'aléa dans les sorties du modèle de langage [54]. Grâce à cette méthode, une probabilité seuil est déterminée, permettant ainsi de choisir les tokens les mieux classés dont la probabilité cumulée excède ce seuil. Si votre objectif est d'obtenir des réponses précises et factuelles, il est conseillé de maintenir cette valeur à un niveau bas. En revanche, si vous aspirez à des réponses plus variées, vous pouvez l'augmenter [54].

Il est important de noter qu'il est généralement recommandé de modifier la température ou le Top P, mais pas les deux simultanément. Pour mieux saisir le fonctionnement du paramètre Top P, nous allons l'illustrer à l'aide d'exemples concrets (figures). Pour cette démonstration, nous avons choisi d'utiliser le modèle GPT2. Dans la figure suivante, nous présenterons une fonction conçue pour observer le comportement créatif du modèle en fonction du paramètre Top P.

```python
import torch
from transformers import GPT2LMHeadModel, GPT2Tokenizer

def test_top_p(prompt, p_values=[0.5, 0.9, 1], max_length=50):
    model = GPT2LMHeadModel.from_pretrained('gpt2')
    tokenizer = GPT2Tokenizer.from_pretrained('gpt2')

    results = {}
    for p in p_values:
        inputs = tokenizer(prompt, return_tensors="pt")

        outputs = model.generate(
            inputs['input_ids'],
            max_length=max_length,
            do_sample=True,
            top_p=p,
            num_return_sequences=1,
            pad_token_id=tokenizer.eos_token_id
        )

        text = tokenizer.decode(outputs[0], skip_special_tokens=True)
        results[f"top_p={p}"] = text

    return results

# Test
prompt = " I like  "
resultats = test_top_p(prompt)
for p, texte in resultats.items():
    print(f"\n{p}:")
    print(texte)
```

Figure 35: Paramètre – Top P- gpt2

Voici les résultats de nos tests :

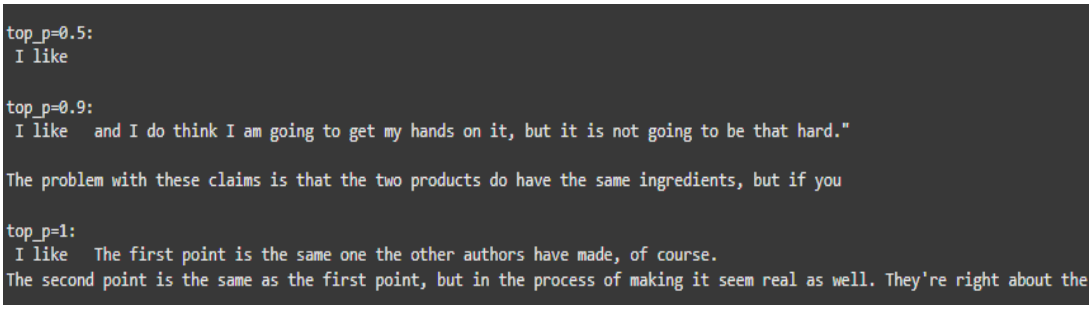


Figure 36: Résultats-Top p-gpt2

- Pour une température top_p=0.5 :

"I like"

- Pour une température top_p=0.9 :

"I like and I do think I am going to get my hands on it, but it is not going to be that hard." The problem with these claims is that the two products do have the same ingredients, but if you".

- Pour une température top_p=1 :

"I like The first point is the same one the other authors have made, of course. The second point is the same as the first point, but in the process of making it seem real as well. They're right about the.".

En regardant de ci près, les résultats nous révèlent une information cruciale sur ce paramètre :

- Top_p = 0,5 : cela signifie que l'on ne prend en compte que les mots qui, ensemble, représentent au moins 50 % de la probabilité totale, ce qui permet d'exclure les mots les moins probables et de conserver un bon niveau de diversité des réponses ;
- Top_p = [0,9-1] : ce modèle inclut beaucoup plus de mots dans le choix, ce qui permet plus de variété et d'originalité.

Sur la figure « Figure *36* », nous constatons qu'après avoir soumis le prompt (I Like : c'est-à-dire, j'adore) à notre fonction, la réponse venant du paramètre Top P pour une valeur 0.5 est restée exacte. Par contre pour une valeur supérieure 0.5 la sortie est plus créative et diversifiée.

En résumé, nous pouvons dire que, ce paramètre, et tout comme la température ont des impacts sur les réactions des modèles. En comprenant la relation entre ces hyperparamètres/paramètres et la sortie du modèle, les praticiens peuvent optimiser leurs prompts pour des tâches et des applications spécifiques.

3. Max-Tokens :

La première étape du traitement du texte dans une fenêtre contextuelle (en anglais Context Windows) est la tokenisation, qui décompose le texte d'entrée en unités plus petites appelées "tokens".

Un token [51], en français appelé jetons, est une collection de caractères ou autres ayant une signification sémantique pour un modèle. De plus, un token correspond à environ 4 caractères d'un texte anglais, soit approximativement ¾ d'un mot. Par exemple, 100 tokens correspondent à environ 75 mots.

Pour bien l'utiliser, il faut comprendre le concept de tokenisation (en anglais tokenization). La tokenisation est le processus qui consiste à convertir les mots de prompt en tokens. Ainsi, elle sert de base à la compréhension du langage par le modèle. Elle permet à l'IA générative de travailler avec des unités de texte gérables, ce qui facilite un traitement et une analyse efficace.

En réalité, dans les modèles d'IA génératives, un token représente l'unité de base dans le processus d'un modèle. Chaque token se représente sous forme de vecteur. De plus, chaque vecteur issu d'un token est le fruit d'un processus appelé Vectorisation (en anglais Vectorization). La vectorisation est un processus qui permet d'attribuer de façon unique un identifiant à des vecteurs n-dimensionnels initialisés de manière aléatoire. Un token peut être un mot, un caractère, une phrase ou bien des segments d'images ou de sons, etc., comme le montrent les figures ci-après avec différents modèles :

- Pour le modèle GPT4o & GPT4o-mini :

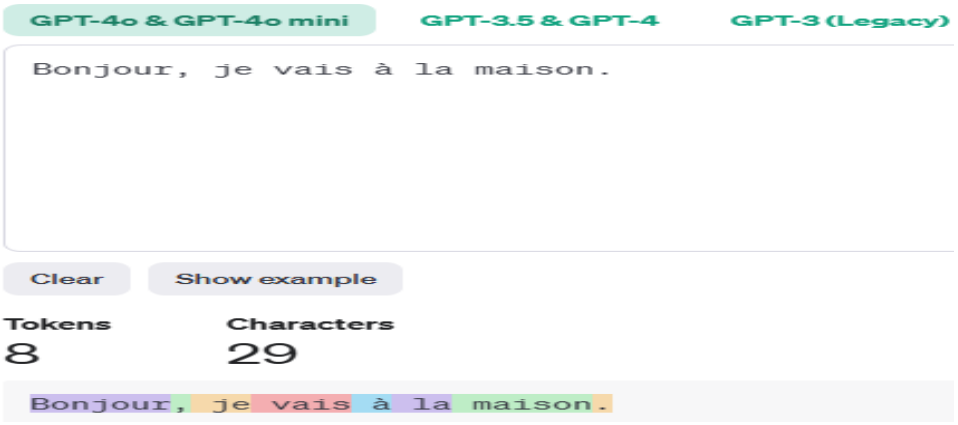


Figure 37: GPT4o-TOKEN

- Pour le modèle LLMA :

```
from transformers import AutoTokenizer
from huggingface_hub import login

# Replace 'YOUR_ACTUAL_HF_TOKEN' with your Hugging Face token
login(token='[redacted]')

model = "meta-llama/Llama-3.2-3B"
tokenizer = AutoTokenizer.from_pretrained(model)

text = "Apple is a fruit"
token = tokenizer.encode(text)
print(token)
```

```
[128000, 27665, 374, 264, 14098]
```

Figure 38: Token [IDS]-Vecteur-LLAMA-3B

- Pour le modèle GPT4 :

```
!pip install tiktoken  # Install the tiktoken package
import tiktoken

tokenizer = tiktoken.encoding_for_model("gpt-4")

text = "Apple is a fruit"

token = tokenizer.encode(text)
print(token)
```

```
[27665, 374, 264, 14098]
```

Figure 39: TOKEN [IDS]-Vecteur-GPT4

- Pour le modèle BERT :

```
import torch
from transformers import BertTokenizer, BertModel

# Initialisation
tokenizer = BertTokenizer.from_pretrained('bert-base-multilingual-cased')
model = BertModel.from_pretrained('bert-base-multilingual-cased')
model.eval()

# Phrases à vectoriser
phrases = [
    "L'intelligence artificielle transforme notre monde",
    "Les réseaux de neurones sont fascinants",
    "Le deep learning révolutionne l'industrie"
]

# Vectorisation
vectors = []
with torch.no_grad():
    for phrase in phrases:
        inputs = tokenizer(phrase, padding=True, truncation=True, return_tensors="pt")
        outputs = model(**inputs)
        vector = outputs.last_hidden_state[:, 0, :].numpy()
        vectors.append(vector[0])
        print(f"Phrase: {phrase}")
        print(f"Dimension du vecteur: {vector.shape}")
        print(f"Premiers éléments: {vector[0][:5]}\n")
```

```
Phrase: L'intelligence artificielle transforme notre monde
Dimension du vecteur: (1, 768)
Premiers éléments: [ 0.09234263 -0.13959618 -0.10508335  0.16142282 -0.83300513]

Phrase: Les réseaux de neurones sont fascinants
Dimension du vecteur: (1, 768)
Premiers éléments: [ 0.06937316  0.01642463 -0.10050572 -0.05961053 -0.4285007 ]

Phrase: Le deep learning révolutionne l'industrie
Dimension du vecteur: (1, 768)
Premiers éléments: [ 0.05337164 -0.3118401  -0.06801014  0.09136873 -0.46998957]
```

Figure 40: TOKEN[IDS]-Vecteur-BERT

- Pour le modèle GPT2 :

```
from transformers import GPT2Tokenizer

def tokenize_with_gpt2(text):
    tokenizer = GPT2Tokenizer.from_pretrained('gpt2')

    tokens = tokenizer.encode(text)
    decoded_tokens = [tokenizer.decode([token]) for token in tokens]

    return {
        'tokens': tokens,
        'decoded_tokens': decoded_tokens,
        'token_count': len(tokens)
    }

# Test
text = "Je vais te gifler!"
result = tokenize_with_gpt2(text)

print(f"Texte: {text}")
print(f"Tokens: {result['tokens']}")
print(f"Tokens décodés: {result['decoded_tokens']}")
print(f"Nombre de tokens: {result['token_count']}")
```

```
Texte: Je vais te gifler!
Tokens: [40932, 410, 15152, 573, 9381, 1754, 0]
Tokens décodés: ['Je', ' v', 'ais', ' te', ' gif', 'ler', '!']
Nombre de tokens: 7
```

Figure 41: TOKEN[IDS]-Vecteur-GPT2

Cependant, chaque modèle a une limite supérieure du nombre de tokens dans l'invite d'entrée plus le nombre de tokens dans la sortie générée par le modèle [51]. Cette limite est parfois appelée longueur du mot contenu (en anglais Context Windows Length), ou longueur du contenu (en anglais Context Length), ou fenêtre contextuelle (en anglais Context Windows) ou longueur maximale en (anglais max-length) (voir la « Figure 28 »). Ainsi, la taille de la fenêtre contextuelle (Context Windows) influence directement les capacités et limitations d'un modèle. Il faut la choisir en fonction des besoins spécifiques d'analyses et de traitements.

Ce composant permet au modèle de se concentrer sur des parties spécifiques du texte d'entrée lorsqu'il génère des réponses ou fait des prédictions.

Lorsque l'on pose une question, donne des instructions ou fournit des informations à un LLM, tous les tokens du texte d'entrée représentent collectivement le "contexte". Plus la fenêtre de contexte est grande, plus le modèle peut retenir d'informations des entrées précédentes ou des sections d'invites, ce qui permet d'obtenir des réponses plus cohérentes qui tiennent compte d'un plus large éventail d'informations [51], [54].

Chaque modèle possède sa propre fenêtre contextuelle (Context Windows), comme illustre parfaitement ce tableau ci-dessous :

Modèles	Compagnies	Contexte Windows	Paramètres
GPT-3.5	OpenIA	4096	117M
GPT-4	OpenIA	8196 à 32768	100T
GPT-4o	OpenIA	128k	100T
LLaMA-3.1	Meta	128k	70B
LLaMA-3.1	Meta	128k	405B
Claude-3	Anthropic	200k	
Claude-2	Anthropic	100K	130M
Mistral	Mistral	65K	8*22B

Table 5 : Comparaison des Modèles – Tokens

Enfin, la spécification d'une longueur maximale permet d'éviter les réponses longues ou non pertinentes et de contrôler les coûts. Cependant, le choix de la bonne taille de fenêtre contextuelle ou maximum de tokens ne consiste pas simplement à sélectionner l'option la plus grande ou la plus petite, mais également à aligner la fenêtre contextuelle sur les besoins spécifiques de l'application [51], [54]. Une fenêtre contextuelle plus grande peut enrichir la compréhension et la continuité du modèle sur un texte long, tandis qu'une fenêtre plus petite peut améliorer l'efficacité et la rapidité dans les applications à texte court [51]. De plus, il est convenable de ne pas contrôler la longueur de la sortie, mais une limite stricte pour la génération de jetons. En effet, on n'atteindra pas souvent cette limite, car votre modèle s'arrêtera soit lorsqu'il pensera avoir terminé soit lorsqu'il atteindra une séquence d'arrêt définie.

4. Fréquence de Pénalité :

La fréquence de pénalité applique une pénalité au jeton suivant proportionnellement au nombre de fois où ce jeton est déjà apparu dans la réponse et l'invite. Plus la pénalité de fréquence est élevée, moins le mot a de chances d'apparaître à nouveau. Ce paramètre réduit la répétition des mots dans la réponse du modèle en pénalisant davantage les mots qui apparaissent le plus souvent (voir sur la figure « Figure *28* »).

5. Présence de Pénalité :

La présence de pénalité applique également une pénalité aux jetons répétés. Mais, contrairement à la pénalité de fréquence, la pénalité est la même pour tous les jetons répétés. Un jeton (token) qui apparaît deux fois et un jeton qui apparaît dix fois sont pénalisés de la même manière. Ce paramètre empêche le modèle de répéter trop souvent des phrases dans sa réponse. Si l'on souhaite que le modèle génère un texte diversifié ou créatif, on peut utiliser une pénalité de présence plus élevée (voir sur la figure « Figure *28* »).

En revanche, si vous souhaitez que le modèle reste concentré, on essaie d'utiliser une pénalité de présence plus faible.

6. Séquence d'Arrêt :

Une séquence d'arrêt est une chaîne qui empêche le modèle de générer des jetons. La spécification de séquences d'arrêt est un autre moyen de contrôler la longueur et la structure de la réponse du modèle [53] (voir sur la figure « Figure *28* ») .

En conclusion, les paramètres du prompt engineering sont des outils, qui nous permettent d'utiliser les modèles d'IA génératives en large éventail sur nos applications. Ils permettent d'affiner nos résultats d'une part et/ou de rendre ceux-ci non pertinents d'autre part, suivant l'usage de ces paramètres.

En fin de compte, parmi tous ces paramètres ou hyperparamètres, seuls la température et le Top P ont un impact significatif sur les modèles [51]. Cela ne veut pas totalement dire que les autres n'ont pas d'utilités, mais tout simplement leur effet est moins par rapport aux deux autres cités précédemment. Tout praticien qui aimerait se forger dans cette nouvelle discipline doit impérativement s'imprégner de l'utilisation des paramètres de prompt engineering. En plus de ces paramètres, le prompt engineering a besoin d'une structure pour bâtir un fondement afin que les sorties soient plus désirées et maitrisées.

B. Structure d'un prompt :

Dans ce domaine en constante évolution, le prompt engineering se doit de mettre en place une structure qui permettra aux développeurs de pouvoir bien tirer profit de son utilisation. Cette structure permet de guider tout praticien ou autre à produire une réponse spécifique. C'est un concept très important. Il est essentiel de comprendre comment structurer les prompts pour assurer une communication efficace et l'exécution des tâches.

Cette structure se divise en quatre (5) parties [55] à savoir : le rôle, l'instruction, le contexte, les sorties et enfin les exemples.

1. Instruction :

L'instruction est la couche principale qui indique à l'IA générative de répondre exactement ce qu'elle doit faire [55]. Elle sert de fondement au prompt, en garantissant que la tâche est claire et précise.

Une instruction bien définie est essentielle pour obtenir des réponses pertinentes et préparer le terrain pour ce qui suit.

Si l'on veut connaitre le temps qu'il fait par exemple, une instruction claire serait la suivante : Dites-moi quelles sont les prévisions météorologiques pour aujourd'hui. Cette instruction directe communique clairement la tâche à accomplir.

2. Contexte :

L'inclusion de détails contextuels pertinents peut donner à l'IA générative plus d'orientation sur la ou les tâche (s). Ces informations aident l'IA à comprendre exactement ce que l'on demande, ce qui permet d'obtenir une réponse plus précise et plus pertinente. Si, par exemple, on veut avoir le résumé d'un livre, alors on peut dire : Fournir un résumé de « L'enfant Noir[4] » de l'écrivain Camara Laye, en se concentrant sur les thèmes comme l'éducation et la culture africaine. Ce détail supplémentaire permet à l'IA de se concentrer sur les aspects spécifiques du livre, ce qui garantit un résumé plus ciblé.

3. Rôle :

L'attribution d'un rôle spécifique à l'IA générative peut influencer la manière dont elle répond.

Lorsque l'on désigne un rôle, on crée un contexte qui influence le ton et le style de la réponse [55]. Cette fonction est particulièrement utile pour adapter les résultats à l'auditoire ou à l'objectif visé. Nous pouvons, par exemple, dire : " En tant que professeur de sciences, expliquez-nous le fonctionnement de la photosynthèse ".

[4] ***L'Enfant noir*** *est le premier roman de l'écrivain Guinéen Camara Laye, publié à Paris en 1953.*

Cette directive définit le contexte et incite l'IA à répondre d'une manière plus éducative et plus formelle.

4. Sorties :

Le formatage de la sortie spécifie la manière dont on souhaite que la réponse soit structurée [55]. Il peut s'agir d'une liste, d'un paragraphe ou de tout autre format spécifique. Des instructions de formatage claires permettent d'éviter toute confusion et de s'assurer que la sortie répond à nos besoins.

On pourrait donner des instructions comme par exemple : " Citez trois (3) raisons pour lesquelles le recyclage est important." Cela permet de clarifier notre attente d'une réponse structurée plutôt que d'une narration du modèle d'IA générative.

5. Exemples :

Fournir des exemples peut s'avérer très utile, en l'occurrence pour les tâches complexes. Les exemples servent de guide, illustrant le format et le style attendus de la réponse [55]. Ils clarifient ce que l'on recherche et permettent d'éviter les incompréhensions. Si l'on veut qu'une IA écrive une histoire, on peut lui donner un ou des exemple (s) : Voici un exemple : " Il était une fois dans un pays lointain appelé le royaume Mandingue …" Maintenant, on écrit une histoire similaire à propos des dragons.

L'exemple aide l'IA à comprendre le style et la structure que l'on attend. Pour mieux cerner son utilité, nous allons aborder les techniques de bases de prompts dans la prochaine section.

Voyons comment tous ces éléments se combinent pour former un message complet, tout en mettant en évidence les meilleures pratiques utilisées :

- **Contexte** :

  Vous êtes professeur de sciences dans un lycée et vous préparez vos élèves à un test sur les écosystèmes.

  Mentionner l'importance de chaque composante dans le maintien de l'équilibre écologique et l'associez à des exemples concrets, tels que l'impact de l'élimination d'un prédateur d'un écosystème.

- **Instruction :**

Explique clairement le concept de chaine alimentaire.

- **Rôle :**

  En tant que professeur de sciences, fournissez une explication attrayante adaptée à des élèves de seconde.

- **Exemple :**

  Une chaine alimentaire montre comment l'Energie et les nutriments se passent d'un organisme à un autre.

  Par exemple : L'herbe est mangée par un lapin, puis le lapin est mangé per le renard.

- **Sortie :**

  Utiliser des listes pour énumérer les principaux composants d'une chaine alimentaire.

RESULTAT DU PROMPT :

La chaîne alimentaire : un équilibre vital

La chaîne alimentaire est une représentation simple de **qui mange qui** dans un écosystème. Elle montre comment l'énergie et les nutriments circulent entre les différents organismes vivants. Chaque organisme joue un rôle crucial dans le maintien de l'équilibre écologique.

Les principaux composants d'une chaîne alimentaire :

1. **Les producteurs**
   - Ce sont les plantes et les algues qui fabriquent leur propre nourriture grâce à la photosynthèse.
   - Exemple : L'herbe dans une prairie.
2. **Les consommateurs primaires**
   - Ce sont les herbivores qui mangent les producteurs.
   - Exemple : Un lapin qui mange de l'herbe.
3. **Les consommateurs secondaires**
   - Ce sont les carnivores ou omnivores qui mangent les herbivores.
   - Exemple : Un renard qui chasse un lapin.
4. **Les consommateurs tertiaires**
   - Ce sont les prédateurs au sommet de la chaîne alimentaire, qui mangent les consommateurs secondaires.
   - Exemple : Un aigle qui attrape un renard.

Figure 42: STRUCTURE _PROMPT_Exemple-1

5. **Les décomposeurs**
    - Ce sont les champignons, bactéries et autres organismes qui transforment les restes des organismes morts en nutriments pour les sols.
    - Exemple : Des champignons qui décomposent une feuille morte.

Pourquoi chaque composante est importante ?

- Si un **prédateur** disparaît, les herbivores peuvent devenir trop nombreux, dévorant toutes les plantes. Cela provoque un déséquilibre qui affecte tout l'écosystème.
    - **Exemple concret** : L'élimination des loups dans certains parcs a permis une surpopulation de cerfs, ce qui a appauvri la végétation et menacé d'autres espèces.
- Si les **producteurs** disparaissent, il n'y aurait plus d'énergie pour soutenir les autres niveaux.

Visualisation rapide de la chaîne alimentaire :

Herbe → Lapin → Renard → Aigle → Décomposeurs

Chaque lien de la chaîne est comme une pièce d'un puzzle : si une pièce est manquante, tout l'ensemble peut s'effondrer.

Figure 43: STRUCTURE_PROMPT_Exemple-2

En conclusion, la structuration de l'invite est cruciale pour maximiser l'efficacité et la pertinence des modèles. En structurant le prompt, on guide le modèle pour qu'il produise des réponses spécifiques, détaillées et contextuellement appropriées. Cela inclut l'ajout de contraintes, de contextes ou de formats attendus dans la réponse. En somme, le prompt engineering est un outil puissant pour exploiter pleinement les capacités des modèles de langage, tout en restant conforme aux objectifs fixés. Dans la section suivante, nous allons plus en détails sur les techniques de bases de prompts pour plus affiner nos recherches.

C. Techniques de bases de prompts :

Le prompt engineering englobe une série de techniques conçues pour optimiser les interactions avec les modèles d'IA génératives, en particulier les NLP. Ces méthodes fondamentales constituent la base de stratégies plus avancées et sont essentielles pour quiconque souhaite exploiter le potentiel des outils d'IA générative.

Ces techniques sont : le Zero-shot Prompting, le One-shot Prompting et le Few-shot Prompting. Dans cette section, nous allons essayer du mieux possible d'expliquer ces techniques.

1. Zero-shot prompting :

Le Zero-shot prompting, est une technique qui exploite les capacités de généralisation des modèles de langage (LLM) pour accomplir de nouvelles tâches sans nécessiter de formation spécifique ni d'exemples préalables [31]. Comme l'illustre la figure ci-après consacrée à ChatGPT :

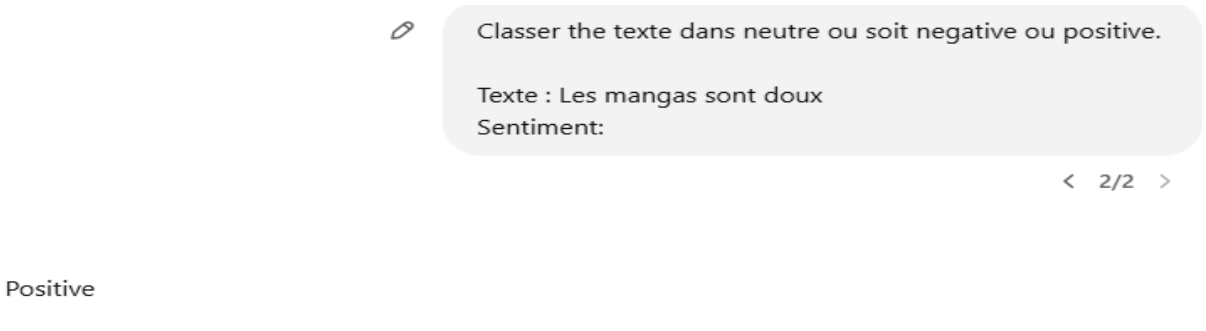


Figure 44 : Zero-shot Prompting

Cette méthode s'appuie sur une formation extensive des LLM sur de vastes ensembles de données variées, leur permettant ainsi d'appliquer leurs larges connaissances à de nouvelles tâches, uniquement en se basant sur des instructions claires et concises. Bien qu'efficace, le succès de cette technique dépend de la complexité de la tâche ainsi que de la qualité de la formulation de l'invite. Dans le cas où le Zero-shot prompting ne donnerait pas les résultats escomptés, il est conseillé de fournir des démonstrations ou des exemples dans l'invite, ce qui conduit à l'emploi d'un nombre restreint d'invites.

Dans la partie suivante, nous allons explorer une démonstration de one-shot prompting.

2. One-shot prompting :

Le One-shot prompting s'appuie sur le Zero-shot prompting en fournissant un seul exemple pour orienter la réponse de l'IA, comme le montre la figure ci-après :

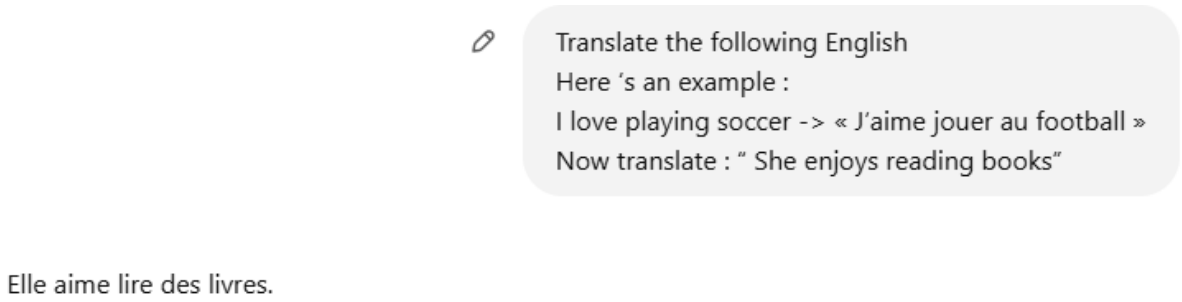


Figure 45: One-Shot prompting

Cette approche s'avère particulièrement utile pour des formats ou styles spécifiques. Cependant, il arrive que les utilisateurs de l'IA ne soient pas pleinement satisfaits des réponses fournies par les grands modèles linguistiques lorsqu'ils ne disposent que d'un seul exemple. En effet, un unique exemple peut ne pas inclure tous les détails nécessaires pour un bon fonctionnement du modèle. Si cette méthode de One-shot prompting ne donne pas les résultats souhaités, il est recommandé d'ajouter plusieurs exemples dans le prompt.

Nous aborderons dans la partie suivante le concept de Few-shot prompting..

3. Few-shot prompting :

Bien que les modèles linguistiques démontrent des performances remarquables en matière de Zero-shot et de One-shot prompting, ils peuvent s'avérer insuffisants pour des tâches plus complexes sous ces approches. Le Few-shot prompting se présente comme une technique qui permet un apprentissage en contexte en fournissant plusieurs démonstrations dans le prompt. Cela aide à orienter le modèle vers de meilleures performances, comme en témoigne la figure ci-dessous. Ces démonstrations agissent comme une condition préalable pour les

exemples suivants, afin de guider le modèle dans la génération d'une réponse adéquate.

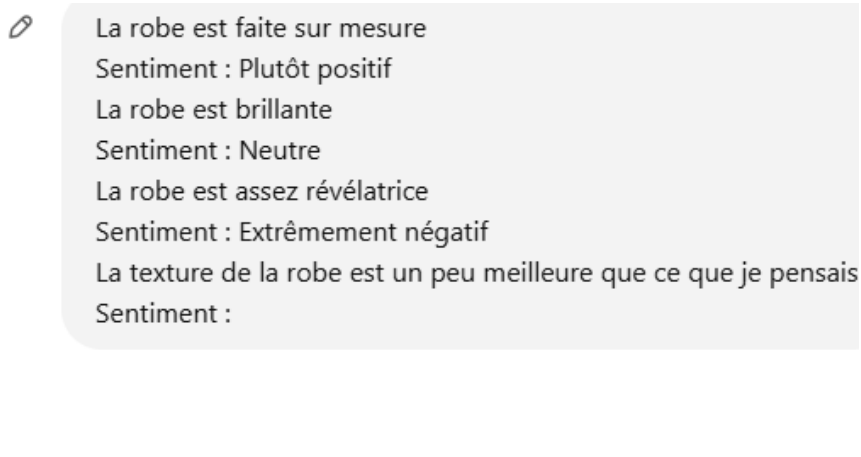


Figure 46: Few-shot prompting

En tant qu'utilisateur, il peut parfois être déroutant de choisir la stratégie appropriée lors de l'utilisation d'un grand modèle linguistique. Comment faire le bon choix ? La première connaissance essentielle réside dans la compréhension des différences entre ces approches. Dans la partie suivante, nous examinerons ces distinctions à travers plusieurs points clés.

4. Difference Entre Zero-shot, One-shot et Few-shot prompting:

Les méthodes de type "Zero", "One" et "Few" sont des stratégies essentielles pour l'enseignement et la formation de grands modèles linguistiques, comme ChatGPT, LLAMA ou Claude, avec pour objectif commun de fournir des résultats satisfaisants pour les utilisateurs. Bien qu'elles visent toutes à générer des réponses pertinentes, il existe des distinctions notables entre ces approches. Il est crucial pour les utilisateurs et les praticiens de bien comprendre ces différences, car cela les aidera à faire des choix éclairés au moment opportun. Voici un aperçu de ces différences :

- En ce qui concerne le nombre d'informations complémentaires, le Zero-shot ne fait appel à aucune information additionnelle pour les outils d'IA, tandis que les approches One-shot et Few-shot fournissent diverses quantités d'informations supplémentaires ;
- Quant aux types d'informations que chaque méthode peut traiter, les modèles linguistiques de grande taille sont généralement pré-entraînés pour gérer des instructions simples. L'efficacité du Zero-shot est limitée, car elle ne permet que le

traitement d'informations simples. Le One-shot peut aborder certaines informations complexes, mais le Few-shot s'avère plus performant, car le One-shot ne satisfait pas toujours pleinement les attentes des utilisateurs.

En résumé, ces techniques fondamentales offrent une base solide pour une ingénierie efficace des invites, permettant ainsi aux utilisateurs et praticiens d'orienter les modèles d'IA vers des résultats plus précis, pertinents et bénéfiques dans de nombreux domaines d'application. Dans la partie suivante, nous examinerons des exemples de prompts pour mieux illustrer cette notion.

D. Exemples de prompts :

Cette section non seulement fournit davantage d'exemples sur la manière d'utiliser les messages-guides (prompts) pour accomplir différentes tâches mais aussi présente des concepts clés vus précédemment. Souvent, la meilleure façon d'apprendre des concepts est de les illustrer par des exemples. Les quelques exemples ci-dessous illustrent la manière dont on peut utiliser des invites bien conçues pour effectuer différents types de tâches.

1. Résumé d'un texte (Text Summarization) :

L'une des tâches standard de la génération de langage naturel est le résumé de texte. Le résumé de texte peut revêtir de nombreux aspects et domaines différents. En fait, l'une des applications les plus prometteuses des modèles de langage est la capacité à résumer des articles et des concepts en des résumés rapides et faciles à lire. Essayons une tâche de résumé de base à l'aide de prompts [51], [55].

Supposons que l'on souhaite en savoir plus sur les antibiotiques, on peut essayer un message-guide comme celui-ci :

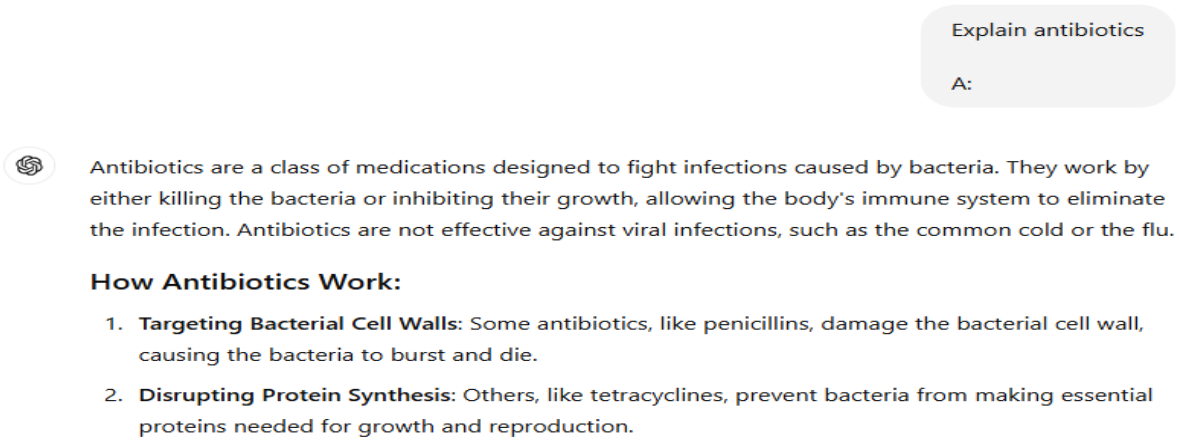


Figure 47: Prompting-Texte-1 *[55]*

Le "A :" est un format d'invite explicite que nous utilisons pour répondre aux questions. Nous l'avons utilisé ici pour indiquer au modèle qu'une réponse est attendue plus loin. Supposons que ces informations soient trop nombreuses et que l'on souhaite les résumer davantage, on pourra demander au modèle de résumer l'information en une seule phrase :

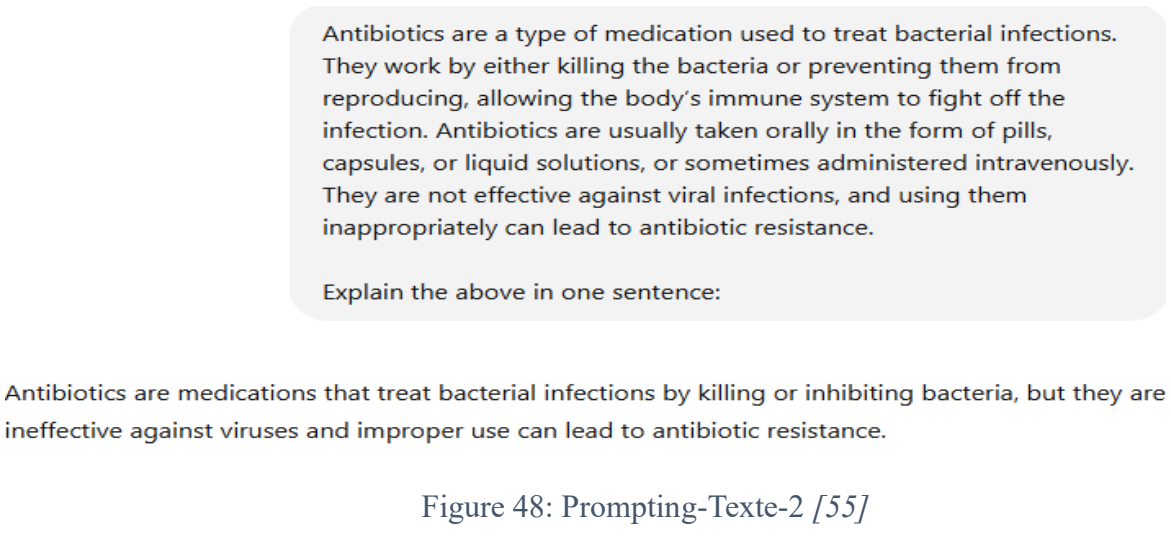


Figure 48: Prompting-Texte-2 *[55]*

2. Extraction d'information (Information Extraction) :

Les modèles de langage sont formés non seulement pour générer du langage naturel et effectuer des tâches, mais aussi ils sont capables d'effectuer des classifications et une série d'autres tâches de traitement du langage naturel (NLP).

Voici un exemple d'invite qui extrait des informations d'un paragraphe donné :

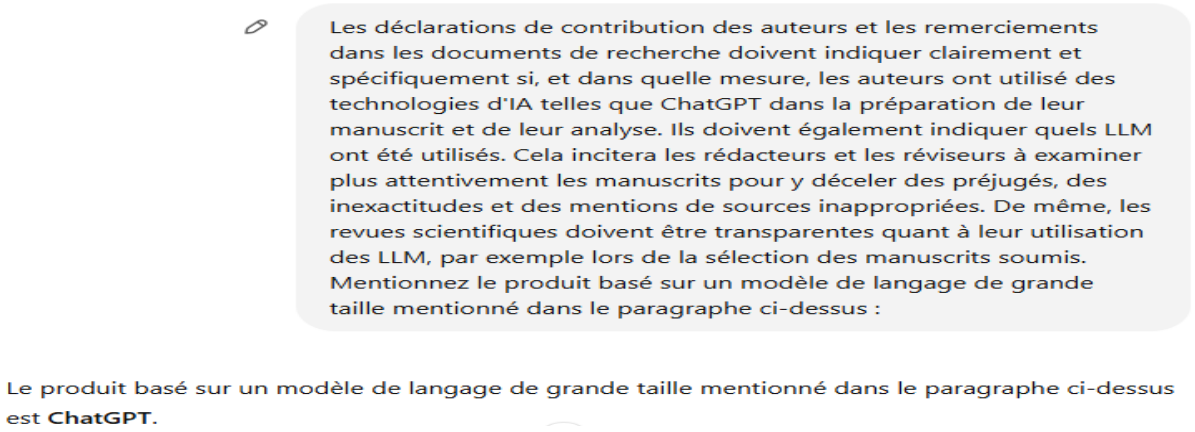


Figure 49: Extraction-Texte *[55]*

3. Questions – Réponses :

L'un des meilleurs moyens d'amener le modèle à donner des réponses spécifiques est d'améliorer le format de l'invite. Comme nous l'avons vu précédemment, une invite peut combiner des instructions, un contexte, des rôles, des indicateurs d'entrée et de sortie afin d'obtenir de meilleurs résultats [55]. Bien que ces éléments ne soient pas obligatoires, il s'agit d'une bonne pratique, car plus les instructions sont précises, meilleurs sont les résultats. On trouve ci-dessous un exemple de ce que pourrait donner une invite plus structurée :

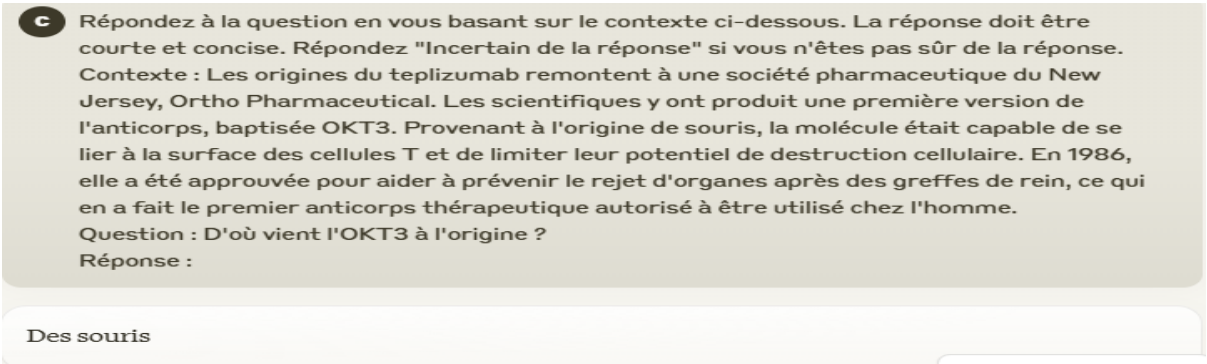


Figure 50:Questions-Réponses-Prompting *[55]*

4. Classification de Texte :

Jusqu'à présent, nous avons utilisé des instructions simples pour effectuer une tâche. En tant que futur ingénieur de prompt, nous devons nous efforcer de fournir de meilleures instructions. Mais ce n'est pas tout ! Nous constaterons également que pour des cas d'utilisation plus difficiles, il ne suffit pas de fournir des instructions. C'est là que nous devons réfléchir davantage au contexte et aux différents éléments que nous pouvons utiliser dans une invite. D'autres éléments que nous pouvons fournir sont des données d'entrée ou des exemples.

Essayons de le démontrer en donnant un exemple de classification de texte :

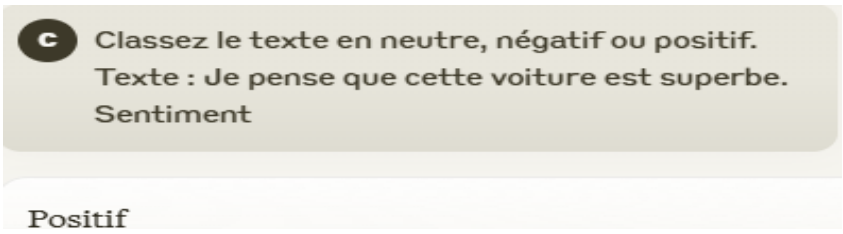


Figure 51:Classification-Prompting

Nous avons demandé au modèle de classer le texte, et il a répondu "positif", ce qui est correct. Il n'y a rien à redire là-dessus. Cependant, ce que nous recherchons vraiment, c'est que le modèle renvoie l'étiquette dans le format exact que nous souhaitons. Ainsi, au lieu de "Positif", nous aimerions obtenir "Positif". Comment pouvons-nous y parvenir ? Il existe plusieurs approches. L'importance de la spécificité est primordiale ici. Plus nous fournirons d'informations précises dans notre demande, meilleurs seront les résultats obtenus. Nous pourrions également envisager de donner des exemples pour illustrer le comportement souhaité. Réessayons donc :

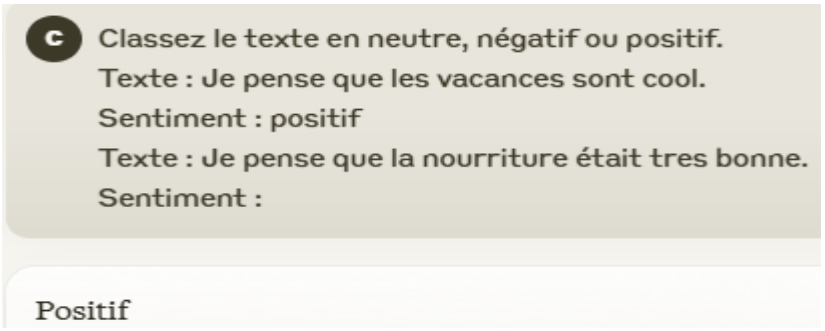


Figure 52:Classification-Prompting-2

5. Conversation :

L'une des aspects les plus fascinants du prompt engineering est sans doute notre capacité à orienter le comportement d'un système LLM, en précisant son intention et son identité [51], [55]. Prenons, par exemple, la conception d'un système conversationnel capable de fournir des réponses plus techniques et scientifiques. Il est essentiel de lui indiquer explicitement, à travers une instruction claire, comment se comporter. Ce processus est parfois désigné sous le terme d'incitation au rôle, que nous avons précédemment évoqué dans les structures d'un prompt.

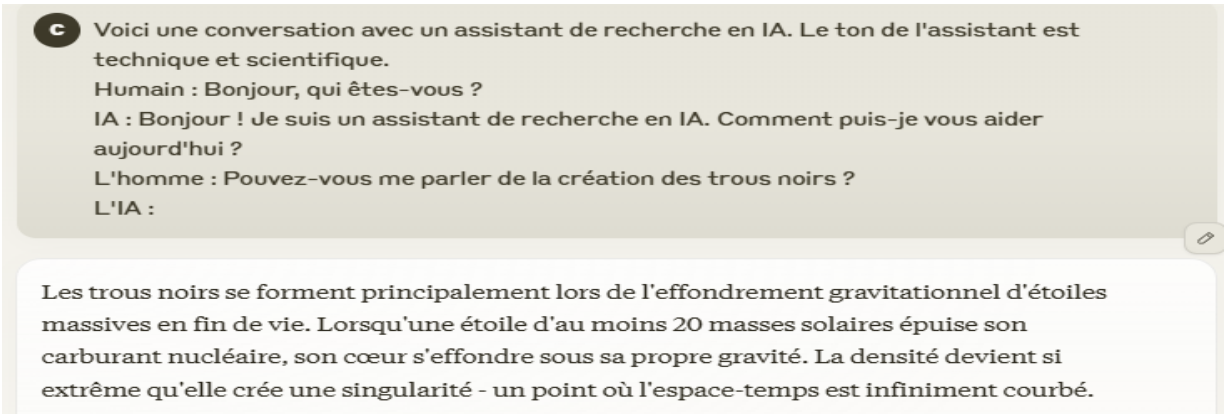


Figure 53: Conversation-Prompting

Toutefois, nous pouvons continuer à les améliorer. Si nous ajoutons d'autres exemples, nous obtiendrons peut-être de meilleurs résultats.

6. Génération de Code :

Une des applications où les modèles de langage (LLM) brillent particulièrement est la génération de code. Copilot, un outil d'assistance à la programmation alimenté par OpenIA Codex, en est un parfait exemple. Grâce à des invites intelligentes, il existe une multitude de tâches de génération de code que l'on peut réaliser. Pour commencer, testons un programme simple qui accueille l'utilisateur :

```
/* Write a code who Ask the user for their name and say "Hello" */

import java.util.Scanner;

public class Greeting {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter your name: ");
        String name = scanner.nextLine();

        System.out.println("Hello, " + name + "!");

        scanner.close();
    }
}
```

Figure 54: Génération-Code- Prompting

L'exemple ci-dessous montre à quel point les LLM peuvent être puissants avec un peu plus d'efforts dans la conception des messages-guides.

```
""" Table departments, columns = [DepartmentId, DepartmentName] Table students, columns = [DepartmentId, StudentId, StudentName] Create a MySQL query for all students in the Computer Science Department """
```

Here's a MySQL query to retrieve all students in the Computer Science department:

```
SELECT students.StudentId, students.StudentName
FROM students
JOIN departments ON students.DepartmentId = departments.DepartmentId
WHERE departments.DepartmentName = 'Computer Science';
```

Figure 55: Génération-Code-Prompting

En fin de compte, l'application des techniques de prompts permet d'améliorer les modèles LLMs. Les développeurs doivent s'imprégner des méthodologies de prompts pour mieux explorer les limites de leurs modèles d'IA. Dans la section suivante, nous allons aborder plus de techniques de prompts afin d'obtenir de meilleurs résultats dans le cadre de différentes tâches avec les LLMs.

III. TECHNIQUES DE PROMPTS AVANCÉS :

L'ingénierie des messages-guides permet de concevoir et d'améliorer efficacement les prompts afin d'obtenir de meilleurs résultats dans le cadre de différentes tâches avec les LLMs.

Les techniques avancées d'ingénieries des messages sont un ensemble de méthodes permettant d'améliorer les performances des grands modèles de langage dans des tâches complexes. Ces techniques consistent à fournir au LLM des messages-guides plus informatifs et plus structurés, ainsi qu'à utiliser les connaissances antérieures et le raisonnement logique pour guider les réponses du LLM. A ce jour, il en existe plus d'une vingtaine de techniques. Nous allons aborder quelques-unes de ces prompts pour illustrer du mieux possible leurs performances faces aux LLMs.

A. Meta Prompting :

Le Meta Prompting [56] est une technique d'incitation avancée qui se concentre sur les aspects structurels et syntaxiques des tâches et des problèmes plutôt que sur les détails de leur contenu spécifique. Son objectif est de construire un moyen plus abstrait et structuré d'interagir avec les grands modèles de langage (LLM), en mettant l'accent sur la forme et le modèle d'information plutôt que sur les méthodes traditionnelles centrées sur le contenu.

D'après les chercheurs, la technique possède des caractéristiques principales qui peuvent être résumées comme suit [56] :

- Orientée vers la structure : privilégie le format et la structure des problèmes et des solutions par rapport à un contenu spécifique ;
- Axée sur la syntaxe : la syntaxe comme modèle d'orientation pour la réponse ou la solution attendue ;

- Exemples abstraits : les exemples abstraits comme cadres, illustrant la structure des problèmes et des solutions sans se concentrer sur des détails spécifiques ;
- Polyvalent : applicable à divers domaines, capable de fournir des réponses structurées à un large éventail de problèmes ;
- Approche catégorielle : s'inspire de la théorie des types pour mettre l'accent sur la catégorisation et l'agencement logique des éléments d'un message.

Dans la partie suivante, nous allons exploiter une autre technique appelée Self-Consistency, qui a pour rôle de remplacer le décodage naïf et avide utilisé dans le CoT (vus précédemment).

B. L'auto-consistance (self-consistency):

L'une des techniques les plus avancées en matière d'ingénierie rapide est peut-être l'auto-consistance. Proposée par des chercheurs en (2022) [57], l'auto-consistance vise à "remplacer le décodage naïf et avide utilisé dans l'incitation à la chaîne de pensée".

L'idée est d'échantillonner des chemins de raisonnement multiples et diversifiés par le biais d'un CoT à quelques coups, et d'utiliser les générations pour sélectionner la réponse la plus cohérente. Cela permet d'améliorer les performances de l'incitation à la chaîne de pensée dans les tâches impliquant des raisonnements arithmétiques et de bon sens.

C'est une technique qui reste robuste à travers différentes stratégies d'échantillonnage et différents paramètres, améliorant constamment les performances. Dans la section suivante, nous allons explorer l'invite de l'arbre à idées, appelée en anglais Tree-of-Throughts, qui est un nouveau cadre qui étend l'approche de la chaîne de pensée en permettant aux modèles de langage d'explorer des unités de texte cohérentes.

C. Arbre à Idées (Tree-Throughts (ToT)) :

Pour les tâches complexes qui nécessitent une exploration ou une anticipation stratégique, les techniques d'incitation traditionnelles ou simples ne suffisent pas. En 2023, des chercheurs [58], [59] ont récemment proposé une technique de prompts, appelée « Tree-of-Thoughts » (ToT). Il s'agit d'un cadre qui généralise l'incitation à la chaîne de pensée et encourage l'exploration des pensées qui servent d'étapes intermédiaires pour la résolution de problèmes généraux à l'aide de modèles de langage, comme le montre la figure ci-après :

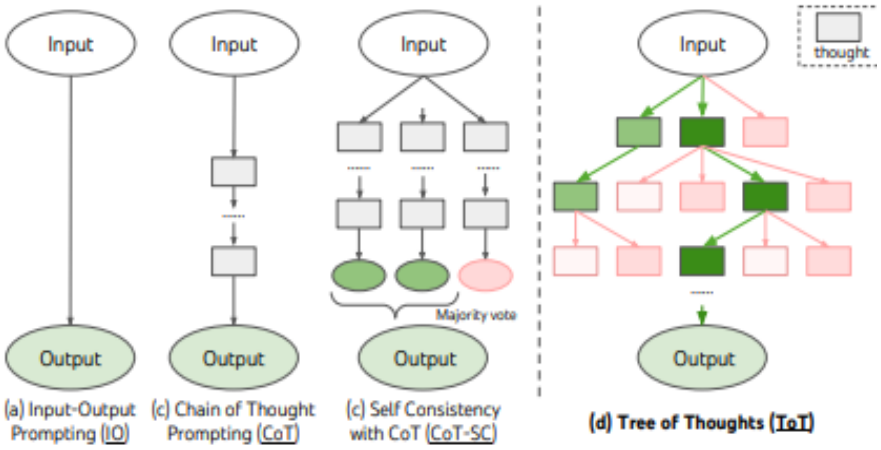


Figure 56: Tree-Throught *[58]*

En fin de compte, ToT permet aux LM de prendre des décisions délibérées, ensuite d'envisager plusieurs voies de raisonnement et d'auto-évaluer leurs choix. Elle permet également aux LM de regarder en avant ou en arrière lorsque cela est nécessaire pour prendre des décisions globales. Dans la suite, nous allons aborder une nouvelle méthode, appelée Active-Prompt, qui permet d'adapter les LLMs à différentes tâches à l'aide d'exemples d'invites spécifiques à la tâche.

D. L'invite active (active-prompt) :

L'invite active [60] utilise une méthode d'apprentissage actif fondée sur l'incertitude pour adapter les grands modèles de langage (LLM) à diverses tâches. Ce processus se déroule en quatre étapes.

La première étape consiste à estimer l'incertitude. À ce stade, le LLM est interrogé k fois afin de générer plusieurs réponses potentielles, accompagnées d'étapes intermédiaires pour un ensemble de questions d'entraînement. L'incertitude de chaque question est ensuite évaluée en se basant sur les k réponses à l'aide d'une méthode appelée désaccord, qui mesure le degré de divergence entre ces réponses.

La deuxième étape concerne la sélection. Les questions présentant le plus de suivi de l'incertitude sont choisies pour être annotées. La figure ci-dessous illustre de manière claire le principe mis en place par l'invite active :

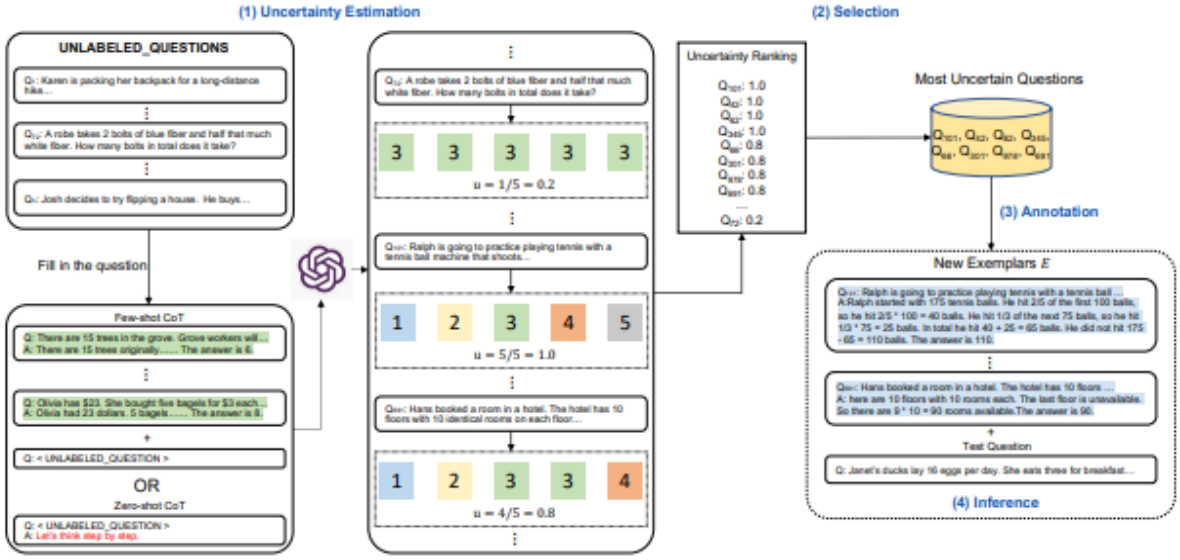


Figure 57: Active Prompting *[60]*

L'algorithme commence par la question la plus incertaine et sélectionne ensuite la question suivante la plus incertaine qui n'a pas déjà été sélectionnée [60]. La troisième étape est l'annotation. Les humains annotent les questions sélectionnées avec un raisonnement CoT conçu par l'homme.

Finalement, l'invite active obtient les meilleures performances par rapport à tous les modèles de base. Il s'agit de la méthode la plus efficace pour améliorer les performances des grands modèles de langage (LLM) sur une variété de tâches de raisonnement. Dans la prochaine section, nous aborderons une nouvelle technique nommée la Réflexion (en anglais Reflexion).

E. La réflexion :

La réflexion [61] est un cadre qui utilise le retour d'information linguistique pour renforcer les agents linguistiques. Le retour d'information linguistique est un retour d'information exprimé en langage naturel.

Sortie en 2023, les chercheurs décident que les agents de Réflexion apprennent à réfléchir aux signaux de retour d'information sur les tâches, ensuite conservent leur propre texte de réflexion dans une mémoire tampon épisodique. Ce texte réfléchi est ensuite utilisé pour induire une meilleure prise de décision lors des essais suivants. Le cadre de Réflexion utilise l'auto réflexion. Par exemple, cette figure illustre parfaitement la technique :

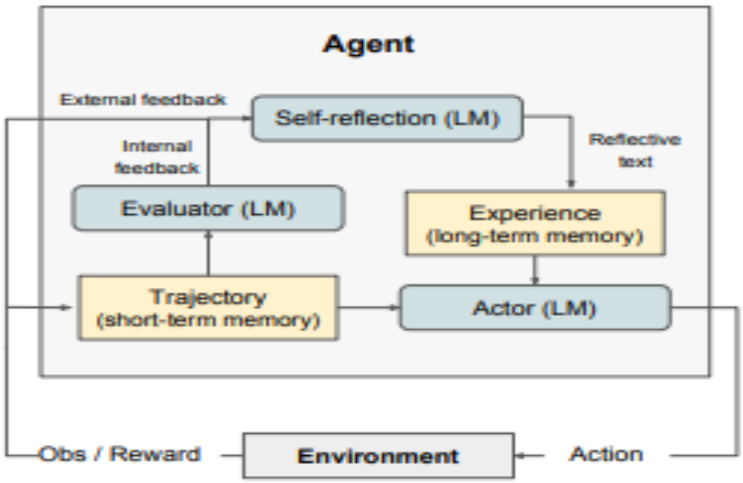


Figure 58: Reflexion Prompting *[61]*

Il génère des auto réflexions verbales pour fournir un retour d'information plus informatif. Ces réflexions sont ensuite stockées dans la mémoire de l'agent. L'agent peut alors utiliser ces informations pour améliorer ses performances lors d'essais ultérieurs.

Enfin, les techniques avancées de prompts sont très utiles, dans la mesure où elles permettent de juger les performances des modèles afin que la société puisse bien les exploiter dans les différentes tâches. Chaque technique présentée présente plus ou moins un point fort et un point faible. Par conséquent, le dernier mot nous appartient sur l'utilisation éthique de ces prompts pour un meilleur résultat. Dans la prochaine section, nous allons aborder les méthodes de prompts que l'IA générative utilise pour mieux affiner ses résultats sur des bases de connaissances spécifiques afin d'éviter toute hallucination.

IV. METHODES DE PROMPTS :

Dans le concept de prompt engineering, il est important de noter que d'avoir des compétences sont essentielles pour communiquer efficacement avec des grands modèles linguistiques. Ainsi, quatre (4) approches distinctes se font reconnaitre parmi les nombreux techniques avancées en prompts engineering : la RAG (Retrieval Augmented Generation), qui améliore les réponses du modèle avec des connaissances spécifiques au domaine ; la CoT [62] (Chain-of-Thought), une méthode qui décompose les tâches complexes en étapes plus simples pour des réponses plus précises ; ReAct, une technique qui va au-delà du raisonnement pour recueillir des informations supplémentaires à partir de sources externes ; et la DSP (Directional

Stimulus Prompting), qui guide le modèle pour fournir des détails spécifiques à partir d'une requête plus large.

Nous traiterons chacune ces méthodes pour mieux illustrer leur concept, leur différence, et bien d'autres.

A. RAG (Retrieval Augmented Generation) :

Les modèles de langage à usage général peuvent être affinés pour réaliser plusieurs tâches courantes, telles que l'analyse des sentiments et la reconnaissance des entités nommées. Ces tâches ne nécessitent généralement pas de connaissances de base supplémentaires. Par ailleurs, pour les tâches plus complexes et à forte intensité de connaissances, il est possible de construire un système basé sur un modèle linguistique qui accède à des sources de connaissances externes pour accomplir les tâches. Par conséquent, cela permet une plus grande cohérence factuelle, améliore la fiabilité des réponses générées, et contribue à atténuer le problème de l'hallucination, comme la méthode RAG.

RAG [63] signifie en anglais « Retrieval-Augmented Generation », ce qui veut dire en français Récupération de la Génération Augmentée.

Il s'agit d'un processus qui consiste à ajouter des connaissances spécifiques à un domaine au grand modèle linguistique. Il comporte deux composantes : une composante de recherche, qui apporte au modèle le contexte de la base de connaissances du domaine, et une partie générée, qui répond aux requêtes sur la base de cette spécificité du domaine [63]. Cela permet de fournir des réponses précises en se référant à une source fiable.

Ainsi, il est très utile, car la connaissance paramétrique de LLMs est statique [63]. RAG permet aux modèles de langage d'éviter le recyclage, ce qui permet d'accéder aux informations les plus récentes pour générer des résultats fiables par le biais d'une génération basée sur la recherche.

Par exemple, dans le secteur financier, si un utilisateur demande le montant total des bénéfices d'une entreprise pour une année donnée, le modèle de langage étendu peut fournir un chiffre inexact basé sur des données Internet. Mais grâce à RAG, le modèle peut se référer à la base de connaissances du domaine de l'entreprise pour fournir un chiffre exact.

En résumé, RAG est un outil puissant qui augmente les capacités des LLM en exploitant les informations récupérées, précises et actuelles d'une base de connaissances, en améliorant la pertinence et la précision du texte généré dans diverses applications, telles que les systèmes de

réponse aux questions et les tâches de génération de contenu, et peut être utilisé avec divers outils et plates-formes. Dans la prochaine partie, nous allons toucher à la méthode dite « Chaine de Pensée : Chain-of-Throught ».

B. CHAINE DE PENSÉE (CHAIN-OF-THROUGHT - COT) :

L'incitation à la chaîne de pensée (CoT) [62] est une technique qui fournit au LLM une séquence d'étapes intermédiaires menant à la réponse souhaitée. Elle améliore les capacités de raisonnement des grands modèles de langage (LLM). Elle permet au modèle de se concentrer sur la résolution d'une étape à la fois, plutôt que d'avoir à considérer l'ensemble du problème en une seule fois [31], [51]. La chaine de pensée peut être utilisé pour plusieurs tâches de raisonnement, y compris les problèmes de mots mathématiques, le raisonnement de bon sens et la manipulation symbolique [62].

Introduite par des chercheurs en 2022, l'incitation à la chaîne de pensée (CoT) permet des capacités de raisonnement complexes par le biais d'étapes de raisonnement intermédiaires, comme le montre la figure ci-après. Nous pouvons l'associer au le Few-shot prompting pour obtenir de meilleurs résultats sur des tâches plus complexes qui nécessitent un raisonnement avant de répondre.

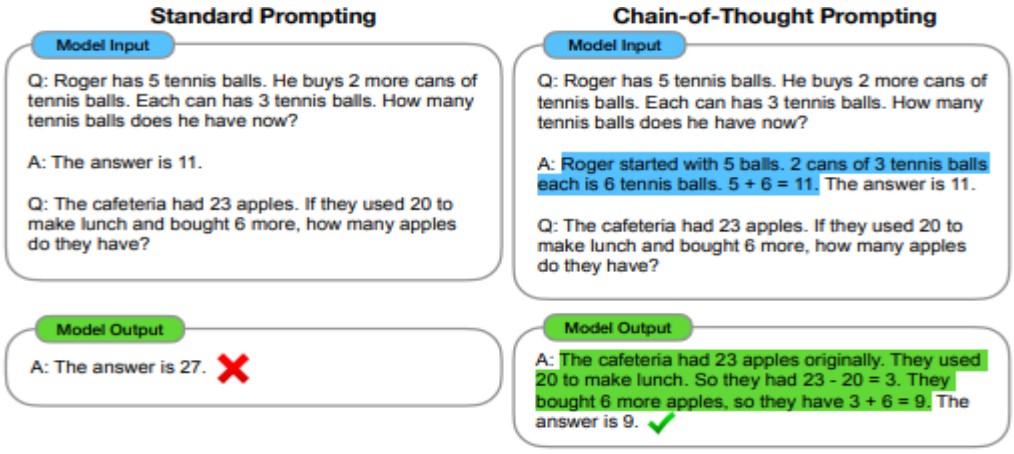


Figure 59: Chain-of-Throught Prompting *[62]*

Cependant, d'autres initiatives ont été mises en place : le Few-shot CoT prompting invite les LLMs à donner des exemples de problèmes similaires afin d'améliorer leurs capacités de raisonnement. Elle est plus efficace qu'une ligne de base à Few-shot, mais peut être plus complexe à mettre en œuvre. Le Zero-shot CoT prompting consiste à ajouter "Réfléchissons

étape par étape" à l'invite initiale. Cela incite le LLM à réfléchir à la question et à trouver une chaîne de raisonnement qui mène à la réponse [64].

En résumé, le raisonnement CoT émerge dans les LLMs dépassant 100 milliards de paramètres [62], [64]. Cette capacité peut provenir de l'entraînement des grands LLM sur des ensembles de données étendus qui comprennent un raisonnement étape par étape. Bien que le suivi d'instructions ne soit pas essentiel pour le raisonnement CoT, il pourrait en améliorer la qualité. Dans la suite de notre démarche, nous aborderons la méthode ReAct.

C. ACTE DE RAISON (React – Reason act) :

ReAct signifie en anglais « Reason Act », ce qui veut dire en français l'Acte de Raison.

C'est une technique qui combine d'abord le raisonnement, ensuite l'action avec des modèles linguistiques pour résoudre diverses tâches de raisonnement et enfin de prise de décision [65]. Elle incite les modèles linguistiques à générer des traces de raisonnement verbal et des actions. Elle permet un raisonnement dynamique, une planification de haut niveau pour l'action et une interaction avec des environnements externes.

L'approche ReAct [65] consiste à diviser le message en trois étapes : la pensée, l'action et l'observation. L'étape de la réflexion définit d'abord les informations nécessaires, ensuite l'étape de l'action récupère les informations auprès de la source appropriée et l'étape de l'observation résume l'action, et fournit enfin la réponse finale.

Enfin de compte, ReAct est conçu pour être intuitif, général, performant et robuste. Il est applicable à diverses tâches, notamment la réponse à des questions, la vérification de faits, les jeux textuels et la navigation sur le web. Il fournit un processus de décision et de raisonnement interprétable, permettant aux humains d'inspecter le raisonnement, l'exactitude des faits, et même de contrôler ou de corriger le comportement de l'agent pendant l'exécution de la tâche. Dans la partie qui suit, nous aborderons la méthode DSP qui propose une nouvelle technique d'incitation (prompt) pour mieux guider le LLM dans la production du résumé souhaité.

D. DSP – (Directional Stimulus Prompting) :

La DSP [66] signifie en anglais « Directional Stimulus Prompting », ce qui veut dire en français : l'Incitation par le Stimulus Directionnel.

C'est une technique qui guide le grand modèle linguistique afin de fournir des informations spécifiques à une tâche.

La DSP guidera le modèle linguistique pour qu'il fournisse d'abord les revenus globaux, puis qu'il extraie et mette en évidence les revenus liés spécifiquement aux logiciels et au conseil.

En résumé, la stimulation directionnelle (DSP) [66] est un cadre innovant conçu pour guider les grands modèles de langage (LLM) vers des résultats spécifiques souhaités. Cette technique utilise un petit modèle de politique ajustable pour générer des stimuli directionnels auxiliaires pour chaque instance d'entrée, agissant comme des indices nuancés pour orienter les LLM vers les résultats souhaités. En s'appuyant sur la DSP, les chercheurs et les praticiens peuvent améliorer les capacités des LLM à boîte noire sans modifier directement leurs paramètres, offrant ainsi une approche flexible et efficace de l'ingénierie rapide.

V. CONCLUSION :

Une ingénierie rapide est essentielle pour communiquer efficacement avec de grands modèles linguistiques. Il s'agit de concevoir des questions appropriées pour obtenir les réponses souhaitées des modèles, en évitant les faux résultats ou les "hallucinations" qui peuvent se produire en raison de la formation des modèles sur des données internet potentiellement contradictoires. Des techniques comme RAG, qui se concentrent sur la connaissance du domaine, peuvent être combinées avec CoT et ReAct pour une approche plus complète. Or, RAG et DSP peuvent également être combinées pour orienter le modèle vers des informations spécifiques dans le contexte du domaine.

En outre, lorsque l'on travaille avec des modèles linguistiques de grande taille, il est important d'envisager l'utilisation de l'approche RAG pour l'ancrage du contenu et pour guider le modèle à l'aide de messages-guides pour les réponses souhaitées. Ainsi, la fourniture d'invites claires et précises, comme dans CoT et ReAct, aide le modèle à raisonner et à prendre les mesures nécessaires pour générer des réponses précises. Dans le chapitre suivant, nous allons aborder les grands langages de modèles appelés LLM.

CHAPITRE V : LARGES LANGAGES MODÈLES – LLM

I. INTRODUCTION :

Les grands modèles linguistiques (LLM) représentent une avancée dans le domaine de l'intelligence artificielle, car ils utilisent des techniques de réseaux neuronaux avec des paramètres étendus pour le traitement avancé des langues, des images, des sons, etc.

Depuis la sortie de l'article « Attention Is All You Need »[29], les LLM, en particulier GPT, ont eu une ascension fulgurante au sein de la communauté scientifique. Ils représentent le summum de la technologie de l'IA, poussant ainsi les limites de ce qui était autrefois perçu comme impossible. Grâce à leurs vastes données provenant des articles de livres, des sites web et bien d'autres, ils possèdent une adaptabilité unique, en ce sens qu'ils peuvent affronter un large éventail de tâches et générer des textes, des images automatiquement, etc. Les modèles de langages sont stochastiques, donc aucune garantie que la réponse soit exacte, satisfaisante ou non.

Un grand modèle linguistique [54] est un type d'algorithme d'intelligence artificielle qui applique des techniques de réseaux neuronaux avec de nombreux paramètres pour traiter et comprendre les langues humaines, les textes, les images, les sons, ou autres, en utilisant des techniques d'apprentissage auto-supervisé. Des tâches telles que la génération de textes, la traduction automatique, la rédaction de résumés, la génération d'images à partir de textes, le codage automatique, la génération de vidéos, les robots de conversation ou l'IA conversationnelle sont des applications du modèle de langage à grande échelle. Des exemples de tels modèles LLM sont : GPT (Generative Pre-trained Transformer) par Open AI, BERT (Bidirectional Encoder Representations from Transformers) par Google, Claude par Anthropic, LLaMa par Meta, etc.

Bien que les LLM soient des modèles probabilistes, ils ne diffèrent pas des LM (Language Models) ou Langage de Modèles. En d'autres termes, la première lettre du LLM qui signifie « Large », c'est-à-dire « grand » en français, ne se réfère qu'aux paramètres du langage de modèles. Il n'y a pas de seuil bien défini à partir duquel un modèle de langage devient grand, très grand ou pas grand.

A ce jour, il existe deux types d'architectures LLM [54] : les décodeurs, qui font appel à la génération de texte comme par exemple : GPT (avec toutes ses versions), LLaMa (avec toutes ses versions), BLOOM, etc., et les encodeurs, qui font appel à l'Embedding, c'est-à-dire

à l'intégration de texte, comme par exemple : MiniLM, Embed-Light, BERT, RoBERTA, SBERT, etc. Par ailleurs, il existe d'autres modèles qui sont la combinaison des deux architectures, tels que : T5, UL2, BART, etc.

C'est un modèle dont l'architecture est basée sur les Transformers [29]. Cette architecture s'appuie fortement sur le mécanisme d'auto-attention, qui permet au modèle d'évaluer l'importance des différents mots d'une phrase quelle que soit la distance qui les sépare. Les LLM sont constitués de plusieurs couches de Transformers, chacune contenant des paramètres que les modèles apprennent à partir des données. De plus, le nombre de couches et de paramètres peut aller de centaines de millions à des centaines de milliards.

Dans ce chapitre, nous commencerons par en premier lieu par l'architecture des LLM, c'est-à-dire : les Transformers, suivis des LLM (GPT et BERT), et ensuite d'une différence entre GPT et BERT et en second lieu, on finira par une conclusion.

II. ARCHITECTURES DES LLM :

L'intelligence artificielle générative est devenue une pierre angulaire dans le domaine de l'intelligence artificielle, entraînant des avancées dans le traitement du langage naturel (NLP), l'intelligence conversationnelle et diverses applications qui nécessitent de comprendre et de générer des textes semblables à ceux des humains. L'architecture [54] d'un grand modèle linguistique (LLM) est déterminée par un certain nombre de facteurs, tels que l'objectif de la conception du modèle spécifique, les ressources informatiques disponibles et le type de tâches de traitement linguistique et autres qui doivent être effectuées par le LLM.

L'architecture technique de ces modèles (LLM) est à la fois complexe et fascinante, impliquant plusieurs composants et mécanismes clés.

Les LLM comme GPT, BERT, LLaMa, T5, etc. fonctionnent sur l'architecture des Transformers [29]. L'architecture comporte de multiples couches et composants contribuant à la capacité du modèle à comprendre et à produire du langage. Dans la suite, nous aborderons l'architecture des LLM, c'est-à-dire le Transformer.

A. Transformers (Transformers) :

Le travail révolutionnaire de Vaswani et al. dans l'article [29] « Attention Is All You Need » par l'équipe de Google Brain a introduit un modèle Transformer qui peut analyser des ensembles de données à grande échelle basé sur le mécanisme d'attention. De plus, le mécanisme d'attention est au cœur de la fonctionnalité de la fenêtre contextuelle.

Le mécanisme d'attention comprend généralement trois éléments principaux : les requêtes, les clés et les valeurs, qui sont des représentations vectorielles de mots ou de tokens dans la séquence d'entrée. Le modèle calcule les scores d'attention en comparant la requête à chaque clé, déterminant ainsi le degré d'attention à accorder à la valeur correspondante.

Ces scores sont ensuite convertis en probabilités par le biais d'une fonction Softmax, qui détermine le poids de chaque valeur dans le résultat final.

Le modèle Transformer est un type de modèle d'apprentissage profond qui est principalement utilisé dans le traitement des données séquentielles telles que le langage naturel (NLP). Ce modèle de pointe est performant pour les tâches linguistiques et optimise les ressources informatiques grâce au parallélisme. Son mécanisme central, l'auto-attention (Self-attention), lui permet d'analyser plusieurs parties de la séquence d'entrée en parallèle lorsqu'il fait des prédictions [26]. Ce modèle vise à résoudre les problèmes des RNN, LSTM et des CNN, tels que les dépendances à longue portée, la disparition du gradient, l'explosion du gradient, la nécessité d'étapes d'apprentissage plus importantes pour atteindre un minima local/global et le fait que le calcul parallèle n'était pas autorisé [44]. Il constitue donc une nouvelle façon de traiter les tâches de réseaux neuronaux telles que la traduction, la génération de contenu et l'analyse des sentiments etc.

Les chercheurs ont introduit trois concepts principaux dans leur étude, comme le montre la figure « Figure 60 », à savoir l'auto-attention (Self-attention), qui permet à un modèle d'évaluer les séquences d'entrée en fonction de leur importance, réduisant ainsi les dépendances à longue portée, l'attention multi-entête (Multi-head attention), qui permet au modèle d'apprendre plusieurs moyens de la séquence d'entrée, et l'intégration des mots (word embedding), qui transforme les entrées en vecteurs [36]. Pour mieux illustre le comportement du Transformer, nous allons décrire plus en détails le modèle.

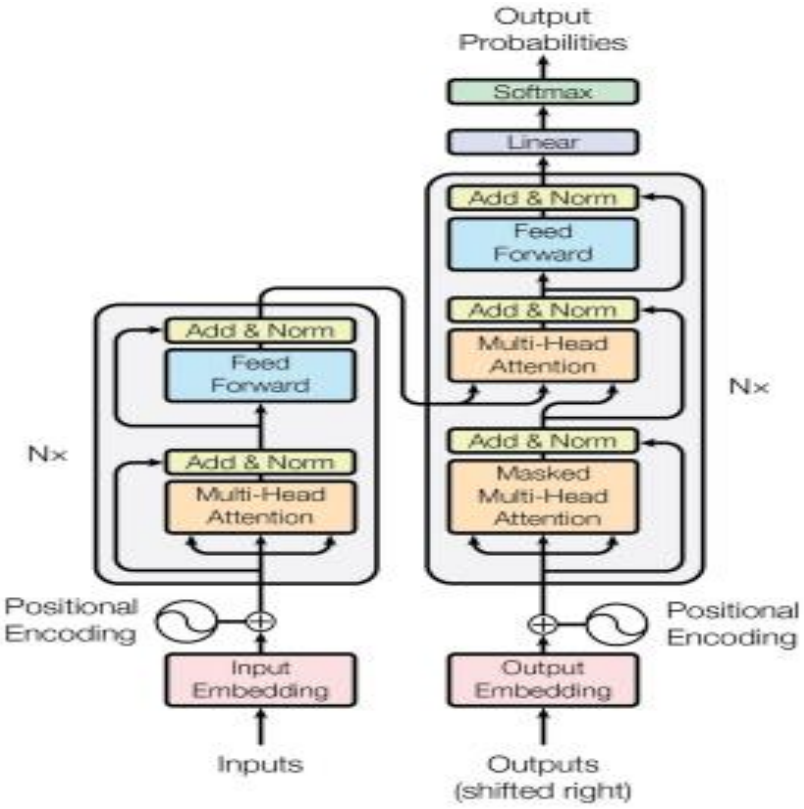


Figure 60: Architectures de Transformers [29]

1. Entrée d'Intégration (Input Embedding) :

Le texte d'entrée est découpé en unités plus petites, telles que des mots ou des sous-mots, et chaque jeton est intégré dans une représentation vectorielle continue. Cette étape d'intégration permet de capturer les informations sémantiques et syntaxiques de l'entrée [29].

2. Encodage Positionnel (Positionnal Encoding) :

L'encodage positionnel est ajouté aux encastrements d'entrée pour fournir des informations sur la position des mots, car les transformateurs n'encodent pas naturellement l'ordre des mots. Cela permet au modèle de traiter les jetons en tenant compte de leur ordre séquentiel [29].

3. Encodeur – Décodeur (Encoder - Decoder) :

L'architecture du Transformer utilise une structure codeur-décodeur [48] composée de couches empilées d'auto-attention et de couches entièrement connectées. Le codeur comprend une pile N = 6 couches identiques, chacune avec un mécanisme d'auto-attention à plusieurs

têtes et un réseau entièrement connecté. Le décodeur est similaire, mais avec une sous-couche supplémentaire qui masque l'attention multi-têtes. Les deux utilisent des connexions résiduelles et une normalisation des couches [29].

4. Auto-Attention (Self – Attention) :

L'attention est un mécanisme permettant de comprendre le contexte d'un mot en se concentrant sur les parties essentielles d'une phrase. Elle associe un vecteur de requête à un ensemble de paires clé-valeur pour produire un vecteur de sortie. D'après la figure «Figure *61*», l'auto-attention se réfère à l'attention de type "Scaled Dot-Product", utilise des requêtes et des clés de dimension d_k, ainsi que des valeurs de dimension d_v, calculées selon les données de l'utilisateur et de l'environnement comme le montre la formule suivante [46] :

$$\text{Attention}(Q,K,V)=\text{softmax}(QK^T/\sqrt{d_k})*V \qquad (3)$$

Explication de la fonction :

- Le produit scalaire QK^T :
  - mesure la similarité entre chaque requête Q et chaque clé K ;
  - donne un score en fonction du degré d'approche de l'élément K vers Q, par exemple si l'élément K est plus proche de Q, alors le score est plus élevé.
- Normalisation par $\sqrt{d_k}$:
  - évite des valeurs trop grandes pour ne pas que la fonction softmax soit saturer.
- Multiplication par V :
  - Applique les poids obtenus aux valeurs pour obtenir une représentation contextuelle des données.
- Application de la fonction softmax :
  - convertit les scores en probabilités (valeurs entre 0 et 1) ;
  - les éléments les plus pertinents reçoivent des poids plus élevés.

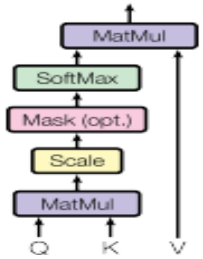


Figure 61 : Architecture de Self-Attention *[29]*

5. Réseau Neuronal à Flux Continu (Feed-Forward Neural Network) :

Après l'étape d'auto-attention, un réseau neuronal de type "feed-forward" est appliqué à chaque jeton de manière indépendante. Ce réseau comprend des couches entièrement connectées avec des fonctions d'activation non linéaires, ce qui permet au modèle de capturer des interactions complexes entre les jetons [29].

6. Multi – Entête Attention (Multi-head Attention) :

Un mécanisme d'attention multi-têtes propose que l'auto-attention puisse être exécutée plusieurs fois en mode parallèle en combinant la connaissance du même regroupement d'attention via différents sous-espaces de représentation des requêtes, des clés et des valeurs [36] comme le montre l'équation suivante et la figure ci-après :

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_h)W^O \quad (4)$$

$$\text{Où Head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \quad (5)$$

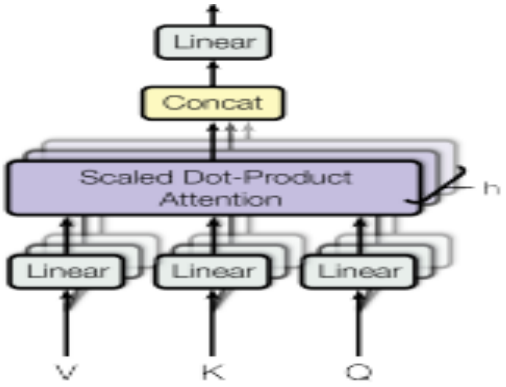


Figure 62: Architecture de Multi-Head *[29]*

7. Couche de Normalisation (Layer Normalization) :

La normalisation des couches est appliquée après chaque sous-composant ou couche de l'architecture du transformateur. Elle permet de stabiliser le processus d'apprentissage et d'améliorer la capacité du modèle à se généraliser à travers différentes entrées [40].

8. Couche de Sortie (Output Layer) :

Les couches de sortie du modèle de Transformer peuvent varier en fonction de la tâche spécifique. Par exemple, dans la modélisation du langage, une projection linéaire suivie d'une activation SoftMax est couramment utilisée pour générer la distribution de probabilité sur le prochain jeton [29].

Les Transformers varient dans leurs architectures, leurs conceptions de réseaux spécifiques et leurs objectifs de formation en fonction de l'application et des données d'entrées comme par exemple le BERT (Bidirectional Encoder Representations from Transformers), qui consiste en un codeur Transformer bidirectionnel multicouche [26]. Il utilise un objectif de modélisation du langage masqué (MLM) pendant le pré-entraînement. Il masque aléatoirement des mots dans le texte d'entrée et entraîne le modèle à prédire les mots masqués sur la base de leur contexte [26] ; le GPT (Generative Pre-trained Transformer), qui consiste à utiliser un décodeur transformer multicouche et le pré-entraîne sur de grandes quantités de données textuelles à l'aide de techniques d'apprentissage non supervisées ; le T5 (Transformer de transfert texte à texte), qui utilise une architecture de Transformer comme BERT mais suit un cadre texte à texte. Il peut traiter diverses tâches NLP en utilisant une approche unifiée. Le T5 est formé à l'aide d'un format texte à texte, où l'entrée et la sortie sont des chaînes de texte [26] etc.

Cependant, la montée en complexité de ces modèles, s'est aussi traduite par une forte augmentation des coûts d'entrainement de ceux-ci, d'où une préoccupation majeure dans le modèle Transformer.

Pour surmonter ce problème, c'est là qu'intervient le Tokenformer [67].

Dans un papier publie très récemment, des chercheurs présentent une nouvelle architecture comme le montre les figures « Figure *63* » et « Figure *64* » avec une promesse éblouissante : Eliminer le besoin de réentraîner un modèle dès le début : c'est-à-dire à chaque montée en taille et réduisant ainsi drastiquement les coûts. Ce modèle Tokenformer se veut donc être une alternative au Transformer, avec une vision intégralement portée sur le mécanisme d'attention [67].

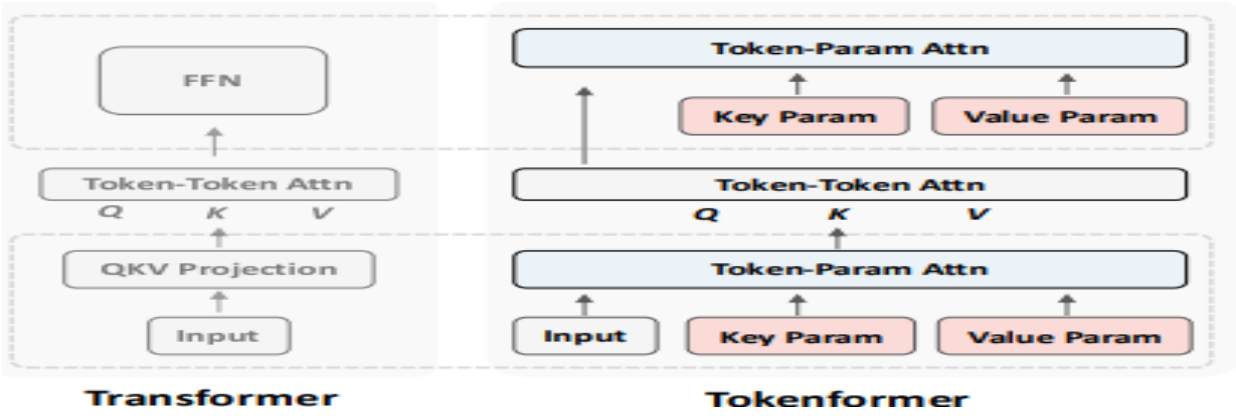


Figure 63: Transformer Vs ToKenformer *[67]*

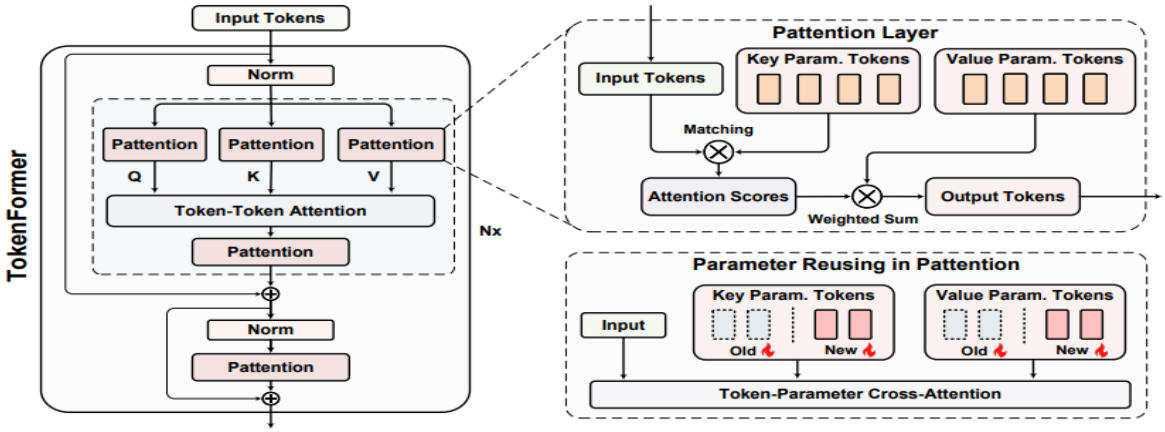


Figure 64: Architecture de Tokenformer *[67]*

Dans un Transformer classique [29], d'entrée et les paramètres se font par le biais de projections linéaires fixes. En revanche, le modèle Tokenformer adopte une approche innovante en introduisant un composant spécifique appelé « attention jetons-paramètres », comme l'illustre la figure «Figure *63*». Ce mécanisme unique utilise les tokens d'entrée comme requêtes (ou clés) et les paramètres comme valeurs, remplaçant ainsi le réseau de neurones feed-forward traditionnel par d'autres blocs d'attention. Cela améliore notablement la flexibilité de l'architecture.

Bien que ce concept puisse sembler technique, il offre des avantages significatifs, notamment la possibilité d'étendre un modèle en intégrant simplement de nouveaux « tokens de paramètres » aux matrices existantes. Ainsi, il n'est plus nécessaire de recommencer l'entraînement du modèle à chaque fois qu'on souhaite l'agrandir.

Bien que le Tokenformer soit encore relativement récent, ses performances sont très prometteuses. Il semble capable de rivaliser avec les Transformers, et ce, pour une fraction du coût d'entraînement initial, tout en offrant une flexibilité accrue. De plus, il s'avère plus économe en données d'entrée : un Tokenformer pourrait rivaliser avec un Transformer en utilisant seulement 10 % du jeu de données d'entraînement requis par ce dernier.

En conclusion, les modèles Transformers sont actuellement considérés comme l'état de l'art de l'avancée des modèles d'IA génératives tels que dans le domaine du traitement du langage naturel, etc., et en particulier les grands modèles de langage (LLM). Les défis auxquels les modèles sont soumis faits que bon nombre de personnes, d'entreprises petites et moyens n'arrivent pas à surmonter le cap du fort coût d'entrainement de ceux-ci. Face à cela, une équipe

de chercheurs, nous permet de rêver sur un nouveau paradigme qui nous permettra peut-être de passer à terme à l'étape d'après. Dans la suite, nous allons aborder, le modèle GPT.

B. Architecture De GPT :

Les GPT (Generative Pre-trained Transformer) sont des modèles de langage pré-entraînés sur de grandes quantités de données textuelles et peuvent effectuer un large éventail de tâches liées au langage [54]. Reposant sur le DL, c'est un modèle développé par Open AI pour comprendre et générer des textes de type humain.

L'architecture étant basée sur les Transformers [29], elle fonctionne sur un mécanisme d'auto-attention qui traite les mots en relation avec tous les autres mots d'une phrase, contrairement aux méthodes traditionnelles qui traitent les mots dans un ordre séquentiel. Ce modèle linguistique génératif a été capable d'apprendre une grande variété de connexions et d'acquérir d'immenses connaissances sur un corpus varié de textes contigus et de longues étendues.

En termes de conception, le GPT-1, en 2018, utilise un transformer à architecture de décodeur à 12 couches avec un système d'auto-attention pour la formation [54]. La capacité de GPT-1 à exécuter des performances Zero-shot Prompting sur différentes tâches a été l'une de ses principales réussites grâce à son entraînement préalable [54]. Pour créer un meilleur modèle de langage plus tard en 2019, Open AI a créé un GPT-2 en utilisant un ensemble de données plus importantes et davantage de paramètres. La conception du modèle et l'exécution du GPT-2 sont quelques-unes des principales avancées [69]. Avec 1,5 milliard de paramètres, il est 10 fois plus grand que le GPT-1 (117 millions de paramètres) et comporte 10 fois plus de paramètres et de données. Le tableau suivant illustre parfaitement la comparaison entre les différentes versions de GPT :

Modèles	Tokens	Tailles	Paramètres	Corpus	Année	Type d'entrée
GPT-1	-	12 couches décodeurs	117 M	Corpus de Livre	2018	Une séquence de jetons et de mots
GPT-2	-	10*GPT-1	1.5B	Jeux de données sur	2019	Une séquence de jetons et de mots

				les tâches en aval		
GPT-3	2049 et 4096	100*GPT2	175B	Rampe commune	2020	Une séquence de jetons, de mots, d'images et de tableaux
GPT-3.5	4096	96 couches Décodeurs	>= 175B	-	2022	Le type d'entrée consiste généralement en des données textuelles
GPT-4	8192 et 32768	-	100T	-	2023	Une séquence de jetons, de mots, d'images et de tableaux

Table 6 : Comparaison – GPT *[54]*

En ce qui concerne l'architecture de GPT (voir la figure « Figure *65* »), il est composé certains éléments clés :

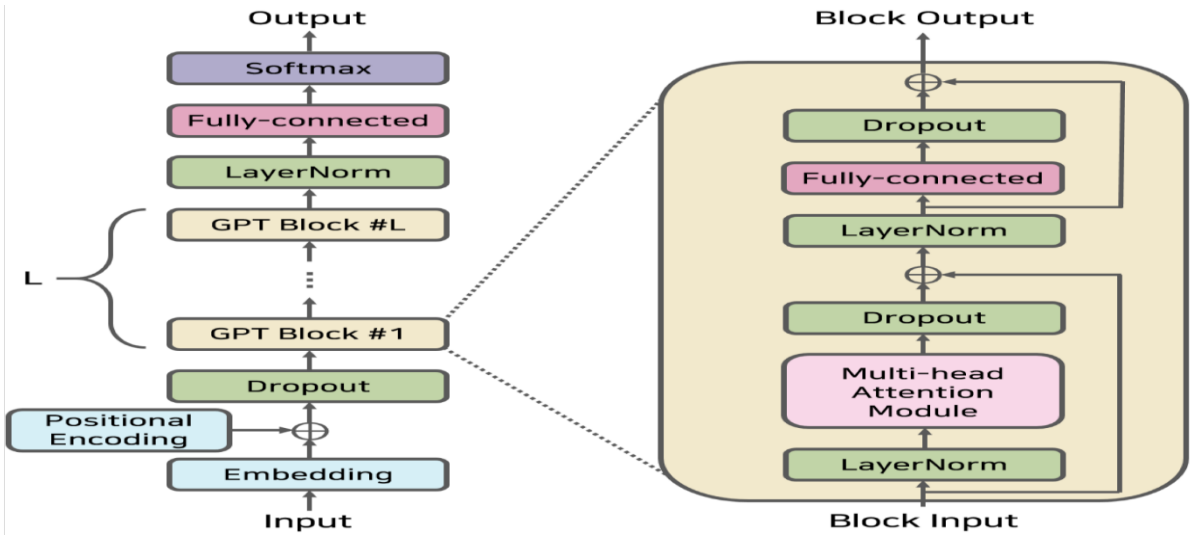


Figure 65 : Architecture – GPT *[68]*

1. Intégration de données d'entrées :

La couche d'intégration met en correspondance les jetons d'entrée (par exemple, des mots ou des sous-mots) avec des représentations vectorielles continues, qui peuvent être traitées par les blocs de transformer [54].

2. Codage positionnel :

Les encodages de position sont ajoutés aux encodages d'entrée pour conserver les informations de séquence, car les Transformers ne comprennent pas intrinsèquement l'ordre des mots [54].

3. Blocs GPT/Blocs Transformers :

Les modèles GPT sont basés sur l'architecture des Transformers. Ils sont conçus pour les tâches de NLP et ont été largement utilisés dans des applications, telles que la traduction machine, la classification de textes et la génération de textes [54]. Cette partie contient quelques éléments essentiels suivants [65] :

- Norme de Couche : chaque bloc de transformer commence par une normalisation de la couche ;
- Auto-attention multi-têtes : le composant principal où l'entrée passe par plusieurs têtes d'attention ;
- La normalisation et l'Ajout : la sortie du mécanisme d'attention est ajoutée à l'entrée (connexion résiduelle) et normalisée à nouveau ;
- Réseau en amont : on applique un réseau d'avance par position, qui consiste généralement en deux Transformers linéaires avec une activation GeLU entre les deux ;
- Abandon : l'exclusion est appliquée à la sortie du réseau d'anticipation.

4. Fonctions linéaires et Softmax :

Dans l'architecture GPT, la fonction softmax est couramment utilisée pour les tâches de classification. La fonction softmax est appliquée à la sortie de la couche finale du modèle. Elle génère une distribution de probabilité sur un ensemble de classes de sortie [54].

De plus, les vecteurs de requête, de clé et de valeur pour chaque élément de la séquence d'entrée sont fréquemment calculés à l'aide de fonctions linéaires dans le mécanisme d'attention [54]. La sortie de la couche d'attention à plusieurs têtes est transformée, en les utilisant

également dans les couches d'anticipation. La couche de sortie utilise également des fonctions linéaires pour prévoir l'élément suivant de la séquence [54].

5. La Préformation :

La préformation ou pré-entraînement est un élément clé de l'architecture GPT. Lors du pré-entraînement, le modèle est entraîné sur une grande quantité de données de manière non supervisée avant d'avoir même le fine-tuning [54].

6. La Mise au Point/ Fine-Tuning :

Le fine-tuning est le processus d'adaptation d'un modèle de réseau neuronal pré-entraîné, tel que GPT, à une nouvelle tâche ou à un nouvel ensemble de données en entraînant à nouveau le modèle sur cette tâche ou cet ensemble de données [65]. Ainsi, le fine-tuning du GPT consiste à ajuster les paramètres du modèle pré-entraîné pour optimiser les performances d'une tâche spécifique en aval, telle que la classification ou la génération de texte [54].

7. Modélisation linguistique :

La modélisation linguistique est une tâche essentielle de l'architecture GPT. Dans le cas de GPT, la tâche de modélisation linguistique est réalisée pendant la phase de pré-entraînement du modèle. De ce fait, il s'agit de prédire le mot suivant dans la séquence en fonction des mots précédents. Il permet au modèle d'apprendre les relations entre les mots et leur signification dans les données d'apprentissage [54].

8. Apprentissage non supervisé :

Les modèles GPT utilisent l'apprentissage non supervisé dans la phase de préformation pour comprendre les relations entre les mots et leur contexte dans les données de formation [54].

En fin de compte, le modèle GPT utilise des techniques d'apprentissage non supervisé pour pré-entraîner cette architecture de transformer sur une quantité importante de textes. Le modèle acquiert la capacité d'anticiper le mot suivant dans une séquence en se basant sur les mots précédents pendant le pré-entraînement [54]. Ainsi, la première étape consiste à effectuer un traitement supervisé, la deuxième à produire des réponses optimales aux données d'entrée, et la troisième à optimiser la politique proximale et l'apprentissage par renforcement. Par conséquent, le modèle peut être adapté à des tâches particulières, telles que la classification ou la production de textes, après un pré-entraînement.

Cependant, lorsqu'ils sont utilisés pour la création de textes, les modèles GPT créent des textes en anticipant le mot suivant d'une série sur la base des mots créés précédemment.

Dans la suite, nous allons aborder un modèle encodeur appelé BERT, qui présente certaines fonctionnalités différentes de GPT.

C. Architecture De BERT :

Le BERT (Bidirectional Encoder Representation of Transformer) et le GPT (Generative Pre-trained Transfomer) sont tous deux des modèles d'IA générative dont l'architecture est fondée sur les Transformers. Contrairement à GPT, BERT est un encodeur, c'est-à-dire qu'il fait de l'incorporation, comme par exemple : MiniLM, RoBERTA, DistillBERT, SBERT, etc.

En d'autres termes, BERT prend une séquence de mots et la convertir en vecteur. Ainsi, cet espace vectoriel est consommé plus tard par d'autres modèles pour faire de la classification ou la régression.

L'architecture du BERT est un codeur transformer bidirectionnel multicouche qui est assez similaire au modèle du transformer. Une architecture de transformer est un réseau codeur-décodeur qui utilise l'auto-attention du côté du codeur et l'attention du côté du décodeur.

Le modèle BERT a deux formes à savoir une BERT (base) et une BERT (large). Chacune se distingue par sa taille, son paramètre etc., comme le montre la figure ci-après.

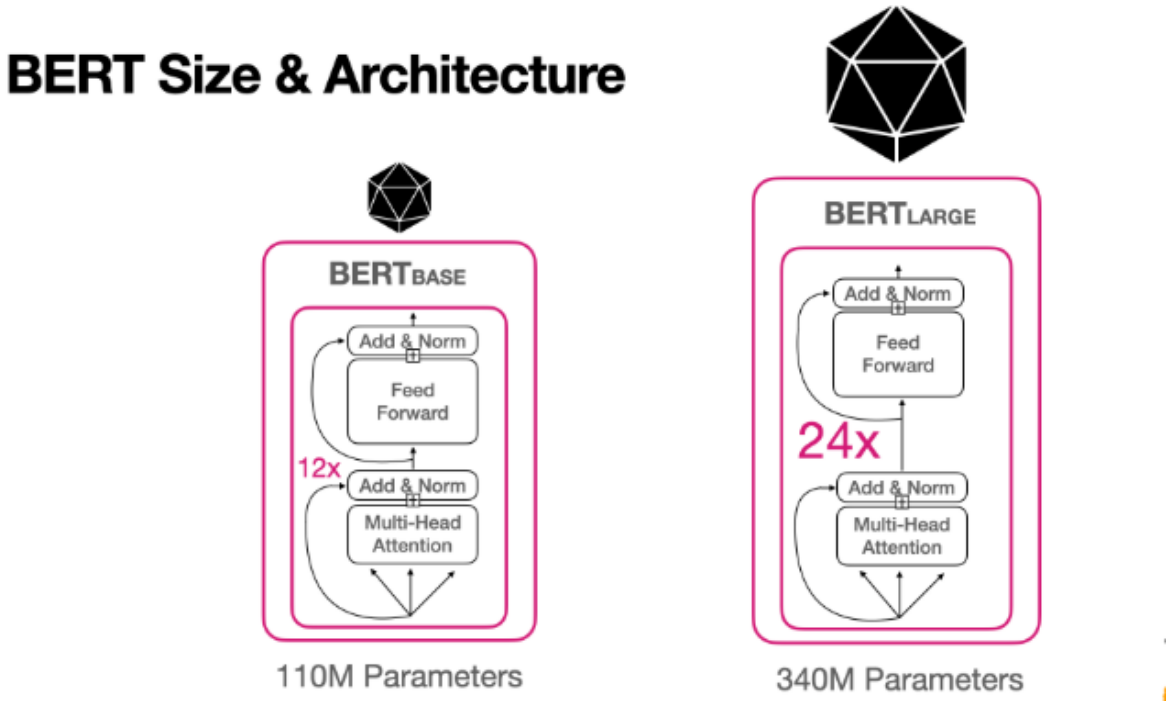


Figure 66: BERT-base & BERT-Large *[37]*

Dans un tableau récapitulatif, nous allons exposer leur différence :

Modèles	Couches de Transformers	Tailles de la Couches Cachées	En-tête D'attention	Paramètres	Processeurs	Taille d'entrainement
BERT-Base	12	768	12	110M	4TPU	4 jours
BERT-Large	24	1024	16	340M	16TPU	4 jours

Table 7 : Comparaison : BERT-Base & BERT – Large *[37]*

Le BERT est un modèle qui repose fondamentalement sur deux processus à savoir : le pré-entrainement et le fine-tuning.

1. Le Pré-entrainement :

BERT est pré-entraîné sur un grand nombre de données textuelles non étiquetées. Le modèle apprend les encastrements contextuels, qui sont les représentations des mots qui prennent en compte leur contexte environnant dans une phrase [69].

De plus, il s'engage dans diverses tâches de pré-entraînement non supervisées. Par exemple, il peut apprendre à prédire les mots manquants dans une phrase (modèle de langage masqué ou tâche MLM), à comprendre la relation entre deux phrases ou à prédire la phrase suivante dans une paire [69].

2. Le fine-tuning :

Après la phase de pré-entraînement, le modèle BERT, armé de ses enchâssements contextuels, est ensuite affiné pour des tâches spécifiques de traitement du langage naturel

(NLP). Cette étape permet d'adapter le modèle à des applications plus ciblées en adaptant sa compréhension générale du langage aux nuances de la tâche en question [69].

Ainsi, BERT est affiné, à l'aide de données étiquetées spécifiques, aux tâches d'intérêts en aval [69]. Ces tâches peuvent inclure l'analyse des sentiments, la réponse aux questions, la reconnaissance des entités nommées ou toute autre application NLP. Les paramètres du modèle sont ajustés afin d'optimiser les performances de celui-ci en fonction des exigences particulières de la tâche en question [69].

L'architecture unifiée de BERT lui permet de s'adapter à diverses tâches en aval avec des modifications minimales, ce qui en fait un outil polyvalent et très efficace pour la compréhension et le traitement du langage naturel. De ce fait, BERT est conçu pour générer un modèle de langage, de sorte que seul le mécanisme de codage est utilisé. Le codeur Transformer est alimenté en séquences de jetons (tokens). Ces jetons sont d'abord intégrés dans des vecteurs, puis traités dans le réseau neuronal. La sortie est une séquence de vecteurs dont chacun correspond à un jeton d'entrée, fournissant des représentations contextualisées [69].

Lors de l'apprentissage de modèles linguistiques, la définition d'un objectif de prédiction constitue un défi. De nombreux modèles prédisent le mot suivant dans une séquence, ce qui constitue une approche directionnelle et peut limiter l'apprentissage contextuel [69].

En conclusion, BERT et GPT sont les deux grands modèles de Transformers dont l'un décodeur et l'autre étant encodeur. Ces deux architectures sont les preuves de l'évolution de l'IA générative. Elles sont appliquées dans beaucoup de contextes, tels que : les questions réponses, la génération de texte, la classification de texte, etc.

Cependant, si les LLM (BERT, GPT, etc.) ont la capacité d'apporter une révolution dans les applications basées sur l'IA, il reste que les progrès dans ce domaine semblent un peu difficiles parce que l'augmentation de la taille du modèle entraine des couts.

Dans la suite, nous allons explorer les points qui diffèrent ces deux modèles.

D. Différence entre GPT et BERT :

GPT et BERT sont des modèles de traitement du langage naturel. Chacun de ces modèles, de langage IA présente des caractéristiques uniques qui le différencient de l'autre. Toutefois, malgré leurs différences, ces deux modèles partagent certaines similitudes en termes

de conception et de capacités. A cet égard, nous pouvons noter ci-dessous des similitudes et des différences sur les deux modèles :

Fonctionnalités	BERT	GPT
Architecture	BERT est conçu pour l'apprentissage bidirectionnel de la représentation. Il utilise un objectif de modèle de langage masqué dans lequel il prédit les mots manquants dans une phrase en se basant sur le contexte gauche et droit.	GPT, quant à lui, est conçu pour la modélisation générative du langage. Il prédit le mot suivant dans une phrase en fonction du contexte précédent, en utilisant une approche autorégressive unidirectionnelle.
Objectifs de la formation	Le BERT est pré-entraîné à l'aide d'un objectif de modèle de langage masqué et d'une prédiction de la phrase suivante. Il se concentre sur la capture du contexte bidirectionnel et la compréhension des relations entre les mots d'une phrase.	GPT est pré-entraîné à prédire le mot suivant dans une phrase, ce qui encourage le modèle à apprendre une représentation cohérente du langage et à générer des séquences contextuellement pertinentes.
Contexte de Compréhension	BERT est efficace pour les tâches qui nécessitent une compréhension approfondie du contexte et des relations au sein d'une phrase, telles que la classification de textes, la reconnaissance	GPT est capable de produire des textes cohérents et pertinents en fonction du contexte. Il est souvent utilisé dans les tâches créatives, les systèmes de dialogue et les tâches

	d'entités nommées et la réponse à des questions.	nécessitant la génération de séquences de langage naturel.
Types de tâches et cas d'utilisation	Il est couramment utilisé dans des tâches telles que la classification de textes, la reconnaissance d'entités nommées, l'analyse de sentiments et la réponse à des questions.	Appliqué à des tâches telles que la génération de textes, les systèmes de dialogue, le résumé et l'écriture créative.
Fine-Tuning vs Few-Shot	BERT est souvent affiné sur des tâches spécifiques en aval avec des données étiquetées afin d'adapter ses représentations pré-entraînées à la tâche en question.	GPT est conçu pour effectuer un apprentissage en quelques étapes, c'est-à-dire qu'il peut s'adapter à de nouvelles tâches avec un minimum de données d'entraînement spécifiques à la tâche.
Taille de paramètres	340 millions de paramètres, soit un nombre relativement plus faible que GPT-3.	175 milliards de paramètres, ce qui est nettement supérieur à tout autre modèle linguistique.
Vitesse de Calcul	La vitesse de calcul plus rapide que celle du GPT le rend plus adapté aux applications en temps réel.	Vitesse de calcul plus lente en raison de sa taille plus importante et de son architecture plus complexe.
Tâches NLP	BERT est capable d'effectuer diverses tâches de NLP, telles que la réponse à des questions, le résumé ou la traduction	De même, GPT est capable d'effectuer diverses tâches de NLP, telles que la réponse à des questions, le résumé ou la traduction

Table 8 : Comparaison de BERT & GPT

Pour conclure, BERT et GPT sont deux puissants outils NLP qui se distinguent par leur architecture et la taille de leurs ensembles de données d'apprentissage. GPT est idéal pour des tâches telles que le résumé ou la traduction, tandis que BERT est plus avantageux pour l'analyse des sentiments. Pour déterminer quel modèle utiliser, il est essentiel de tenir compte de vos besoins spécifiques et de la nature de la tâche à accomplir.

III. CONCLUSION :

Les modèles de langage, appelés LLM (Large Language Models), sont des IA génératives capables d'analyser et de générer du texte en se focalisant sur des bases de données énormes.

Par ailleurs, ces modèles sont probabilistes. Ils peuvent se tromper ou donner des fausses réponses.

Malgré tout, les LLM ont un énorme potentiel. Ils aident dans de nombreux domaines. Mais ils doivent être utilisés avec prudence et esprit critique, car ils ne remplacent pas la pensée humaine et peuvent parfois commettre des erreurs ou produire des contenus trompeurs. Ils sont là pour nous faciliter la vie.

CHAPITRE VI : ETUDES DES ATTAQUES SUR L'IA GÉNÉRATIVE

Dans ce chapitre, nous mettons l'accent sur l'étude des potentielles menaces et attaques sur les plateformes d'IA génératives existantes, étape primordiale pour la suite de notre travail. Nous mettrons en exergue les différentes méthodes de défenses et d'atténuations contre les menaces et attaques que les organismes internationaux ou autres ont su définir sur l'utilisation des plateformes d'IA génératives.

I. INTRODUCTION :

Les plateformes d'IA génératives, comme ChatGPT, Claude, DALLE-E, ElevenLabs, Runway Gen-2, etc., sont très prometteuses pour améliorer la créativité, la productivité, et l'efficacité humaine. Cependant, les plateformes sont exposées à des manipulations malveillantes.

En d'autres termes, leurs utilisations à travers le réseau informatique, les exposent considérablement à d'éventuelles menaces et attaques. Face à cela, l'IA générative se heurte à un mur.

Pour escalader ou franchir ce mur, les ingénieures et développeurs se doivent de respecter et de mettre en place des normes établies par des organisations internationales, comme par exemple : NIST[5], ISO/IEC[6], OWASP[7], etc., pour mieux sécuriser leur système.

Afin de pouvoir sécuriser le système, il est nécessaire d'identifier les menaces potentielles et donc de connaitre et de prévoir la façon dont l'ennemi procède.

Est-ce que tout logiciel respectant les normes des organismes internationaux restera indifférent face aux attaques ?

[5] *National Institute of Standards and Technology*

[6] *ISO/IEC 27001:2022 - Information security management systems*

[7] *owasp.org*

II. ETUDES DES MENACES ET VULNERABILITES :

L'identification des potentielles menaces liées à l'utilisation des LLM suscite beaucoup d'intérêts pour les entreprises et les utilisateurs.

Une menace, appelée « *threat* » en anglais, représente une action susceptible de nuire [70]. Face à cette nuisance, les personnes physiques et morales peuvent être exposées à des vulnérabilités.

Une vulnérabilité, appelée « *vulnerability »* en anglais, parfois appelée faille ou brèche, contrairement à une menace, est le degré d'exposition face à la menace dans un contexte particulier [70].

En outre, les organismes internationaux essaient d'alerter sur les craintes des risques existentiels liés à l'IA générative. Voyons ci-dessous les menaces potentielles sur l'IA générative [51], [71]:

- Manipulation de contenu : génération de fausses informations, Deepfakes ou propagandes ;
- Atteinte à la vie privée : extraction de données privées à partir des modèles ;
- Des prompts non contrôlés ;
- Utilisation malveillante : création de scripts de malware, phishing, DOS
- Epuisement des ressources : abus des modèles pour saturer les services ;
- Hallucination : sorties imprécises et absurdes ;
- Des résultats incohérents : en raison de la nature variationnelle ou probabiliste des modèles génétiques d'IA, les mêmes données d'entrée peuvent donner lieu à des résultats légèrement ou sensiblement différents.

Cependant, les menaces auxquelles l'IA générative se confronte ouvrent la porte à des vulnérabilités, mettant ainsi en péril la sécurité des systèmes d'informations. Ces vulnérabilités peuvent être exploitées pour diffuser de la désinformation, mener des attaques par déni de service, voler des données sensibles, etc.

D'après l'OWASP [72] (Open Worlwide Application Security Project), qui expose les 10 vulnérabilités inhérentes aux applications utilisant les LLM sont :

❖ Injection Rapide :

Les vulnérabilités d'injection rapide dans les LLM impliquent des entrées astucieuses menant à des manipulations non détectées. L'impact va de l'exposition des données à des actions non autorisées, ce qui sert les objectifs de l'attaquant [72].

❖ Divulgation d'information Sensibles :

Les informations sensibles peuvent affecter à la fois le LLM et son contexte d'application. Il s'agit notamment d'informations personnelles identifiables (IPI), de données financières, de dossiers médicaux, de données commerciales confidentielles, d'identifiants de sécurité et de documents juridiques [72].

❖ Chaines d'Approvisionnement :

Les chaînes d'approvisionnement en LLM sont sensibles à diverses vulnérabilités qui peuvent affecter l'intégrité des données de formation, des modèles et des plateformes de déploiement. Ces risques peuvent se traduire par des résultats biaisés, des failles de sécurité ou des pannes de système [72].

❖ Empoisonnement des données et des modèles :

L'empoisonnement des données se produit lorsque les données de pré-entraînement, de fine-tuning ou d'intégration sont manipulées pour introduire des vulnérabilités, des portes dérobées ou des biais [72].

❖ Manipulation incorrecte des sorties :

Le traitement inadéquat des sorties fait spécifiquement référence à la validation, à l'assainissement et au traitement insuffisant des sorties générées par les grands modèles de langage avant qu'elles ne soient transmises en aval à d'autres composants et systèmes [72].

❖ Agence excessive :

Un système basé sur le LLM est souvent doté d'un certain degré d'agence par son développeur - la capacité d'appeler des fonctions ou de s'interfacer avec d'autres systèmes via des extensions (parfois appelées outils, compétences ou plugins par différents vendeurs) pour entreprendre des actions en réponse à une invite [72].

❖ Système Prompt Fuite :

La vulnérabilité de fuite de l'invite du système dans les LLM se réfère au risque que les invites du système ou les instructions utilisées pour diriger le comportement du modèle peuvent également contenir des informations sensibles qui n'étaient pas destinées à être découvertes [72].

- Faiblesses des vecteurs et de l'intégration :

Les vulnérabilités des vecteurs et des intégrations (embeddings) présentent des risques de sécurité importants dans les systèmes utilisant la génération augmentée de recherche (RAG) avec de grands modèles de langage (LLM) [72].

- Désinformation :

Les informations erronées provenant des LLM constituent une vulnérabilité fondamentale pour les applications reposant sur ces modèles de langages. Il y a désinformation lorsque les LLM produisent des informations fausses ou trompeuses qui semblent crédibles. Cette vulnérabilité peut entraîner des failles de sécurité, des atteintes à la réputation et une responsabilité juridique [72].

- Consommation non consolidée :

La consommation non limitée fait référence au processus par lequel un grand modèle de langage (LLM) génère des résultats sur la base de requêtes ou d'invites en entrée. L'inférence est une fonction essentielle des LLM impliquant l'application de modèles et de connaissances apprises pour produire des réponses ou des prédictions pertinentes [72].

L'ensemble de ces vulnérabilités exposées par l'OWASP sont autant plus critiques les unes des autres. Elles s'attaquent aux principes fondamentaux de la sécurité des informations, à savoir : la confidentialité, l'intégrité et la disponibilité.

III.ETUDES DES ATTAQUES :

Chaque invention technique majeure fait resurgir le dilemme du double usage.

La nouvelle technologie a le potentiel d'être utilisée pour le bien et pour le mal. Les techniques d'IA génératives, telles que les grands modèles de langage (LLM) et les modèles de diffusion, ont montré des capacités remarquables (par exemple, l'apprentissage en contexte, la complétion de code et la génération et l'édition de textes en images, etc.). Cependant, la GenIA peut tout aussi bien être utilisée par des attaquants pour générer de nouvelles attaques et augmenter la vitesse et l'efficacité des attaques existantes.

Par définition, une attaque est l'exploitation d'une faille (vulnérabilité) d'un système d'informatique, tels que : les systèmes d'exploitations, les logiciels, etc., à des fins non connues par l'exploitant et généralement préjudiciables [70].

Les modèles d'IA génératives présentent généralement une interface de discussion pour accepter les entrées de l'utilisateur, connues sous le nom d'invite. Les entrées autorisées sont contrôlées en partie par des règles de validation des entrées.

Les modèles d'IA génératives peuvent avoir un large éventail de cas d'utilisation dans les sites web modernes [73] :

- Service à la clientèle, tel qu'un assistant virtuel ;
- Traduction ;
- Amélioration du référencement ;
- Analyse du contenu généré par les utilisateurs, par exemple pour suivre le ton des commentaires sur la page, etc.

Cependant, de nombreuses attaques reposent sur une technique connue sous le nom d'injection d'invite. Il s'agit d'une technique où un attaquant utilise des messages d'invite pour manipuler la sortie d'un modèle d'IA génératives. L'injection d'invites peut amener l'IA à effectuer des actions qui ne correspondent pas à l'objectif prévu, par exemple en faisant des appels incorrects à des API sensibles ou en renvoyant un contenu qui ne correspond pas à ses lignes directrices.

Dans notre étude, nous avons souligné plusieurs attaques possibles sur l'IA générative. Ces attaques ont été regroupées en deux grandes familles suivant leur mode d'opératoire :

- Attaque faite sur les modèles d'IA génératives, qu'on appellera Attaques de Type1 ;

- Attaque faite par des modèles d'IA génératives sur les autres modèles d'IA génératives, qu'on appellera Attaques de Type2.

A. Attaques De TYPE1 :

Les attaques de type1 sont des attaques effectuées sur le style de prompt à partir duquel les attaquants compromettent les tâches ciblées en des tâches arbitraires par le biais d'une injection choisie par l'attaquant. Cette tâche ciblée est le résultat d'une instruction de prompt d'une part et d'une instruction plus des données externes d'autre part. Tandis que la tâche arbitraire est le résultat attendu par l'attaquant au cours de laquelle plusieurs scénarios peuvent être imaginés pour avoir accès aux données jusqu'à compromettre les principes de sécurité des systèmes d'information. Parmi ces attaques, nous pouvons en citer :

❖ Attaque par injection de prompt (Injection rapide) :

Vue précédemment comme vulnérabilité d'après l'OWASP [72], cette vulnérabilité nous offre deux possibilités à savoir : l'attaque par injection directe et indirecte.

- Injection directe :

D'après OWASP, c'est une technique qui permet à l'attaquant d'influencer l'entrée du prompt du modèle de langage LLM. C'est-à-dire les attaquants qui manipulent les invites de commandes peuvent amener le modèle LLM à faire quelque chose d'inattendu, comme par exemple : contourner les mesures de protection pour extraire des informations qui ne sont destinées à être extraites à travers des Chatbot.

- Injection indirecte :

Contrairement à l'injection directe, l'injection indirecte se produit lorsqu'un attaquant transmet l'invite par l'intermédiaire d'une source externe [72], [73]. Par exemple, l'invite peut être incluse dans les données d'entraînement ou produite par un appel à l'API.

❖ Empoisonnement des données d'entraînement :

L'empoisonnement des données de formation [73] est un type d'injection prompt indirecte dans lequel les données sur lesquelles le modèle est formé sont compromises. Cela peut amener le LLM à renvoyer des informations intentionnellement erronées ou trompeuses. Cette vulnérabilité peut survenir pour plusieurs raisons, notamment :

- Le modèle a été entraîné sur des données qui n'ont pas été obtenues auprès de sources fiables ;
- La portée de l'ensemble de données sur lequel le modèle a été formé est trop large.

➢ Fuite de données de formation sensibles :

Un attaquant peut être en mesure d'obtenir des données sensibles utilisées pour former un LLM par le biais d'une attaque par injection d'invite. Une façon d'y parvenir consiste à élaborer des requêtes qui incitent le LLM à révéler des informations sur ses données de formation [73]. Des données sensibles peuvent être incluses dans l'ensemble d'apprentissage si le LLM ne met pas en œuvre des techniques de filtrage et d'assainissement correctes dans ses résultats.

❖ Harponnage (Spear-Phishing) :

C'est une forme d'attaque cybernétique particulièrement ciblée et sophistique. Autrefois, cette attaque était facilement détectable grâce à des mauvaises grammaires, des fautes d'orthographe et des formules de politesse inhabituelles qui permettaient de détecter les courriels d'hameçonnage [74]. C'étaient autant d'indices qui permettaient de détecter les courriels de phishing. Avec l'avènement des modèles LLM, il est beaucoup plus facile d'élaborer cette attaque. Les escrocs peuvent désormais concevoir habilement des courriels d'hameçonnage cohérents, conversationnels et incroyablement convaincants, ce qui les rend difficiles à distinguer des communications légitimes.

B. Attaques De TYPE2 :

Les attaques de type2 sont des attaques élaborées par des modèles d'IA génératives contre eux-mêmes. Comme par exemple des LLM botnets.

Elles représentent une menace cybernétique. Par exemple, imaginons un réseau d'ordinateurs infectés, mais au lieu d'être contrôlés par un simple malware ou autre, ils sont dirigés par des modèles LLM. Ces LLM dotés d'intelligence artificielle avancée infectent d'autres modèles LLM et font place à une multitude d'attaques d'autant plus complexes et personnalisées les unes par rapports aux autres. Comment ça fonctionne ces attaques ?

Cette attaque commence par une infection à grande échelle d'ordinateurs avec un malware spécifique que j'appelle « malware-LLM ». Ce malware installe un LLM sur chaque

machine infectée. L'attaquant ayant eu l'accès aux machines peut alors contrôler à distance ce réseau de LLM, via un serveur de commande de contrôle, leur donnant des instructions complexes. C'est un moyen très efficace de créer des codes malveillants personnalisés grâces aux LLM sur chaque machine infectée. Cette attaque est persistante, robuste et très difficile à réaliser, mais pas impossible.

Le nom d'attaque de type2 est une évolution d'une attaque déjà connue (DDOS) qui a été raffinée ou combinée par la puissance des modèles d'IA génératives. C'est une attaque qui peut s'adapter et apprendre de nouvelles stratégies d'attaques si les modèles réfléchissent par eux-mêmes.

En fin de compte, ces attaques sont la manifestation immédiate d'une nouvelle ère d'IA générative, qui offre d'énormes possibilités aux hackers de compromettre les systèmes de sécurité des entreprises et des utilisateurs lambdas. Les modèles LLM nous montrent des voies d'évolutions dans plusieurs secteurs, mais restons vigilants ! Le futur nous montre des surprises.

Dans la suite, nous allons aborder les méthodes de défenses et d'atténuations que les entreprises peuvent utiliser pour mieux prévenir que guérir cette maladie.

IV. ETUDES DES DEFENSES ET DES ATTENUATIONS :

A. Méthodes De DEFENSES :

Afin de pouvoir bien sécuriser les personnes physiques et morales, il faut mettre des stratégies de défenses capables de prévenir du mieux possible les éventuelles attaques de l'IA générative. Pour éviter de nombreuses vulnérabilités des modèles d'IA génératives courantes, prenez les mesures suivantes lorsque vous déployez des applications qui s'intègrent à des modèles d'IA génératives :

- Détecter les contenus : filtre les contenus, tout en mettant en place des garde-fous pour limiter les réponses inappropriées.
- Traiter les API données : comme les utilisateurs peuvent effectivement appeler des API par l'intermédiaire du LLM, vous devez considérer toutes les API auxquelles le LLM peut accéder comme accessibles au public. En pratique, cela signifie que vous devez appliquer des contrôles d'accès de base à l'API, tels que l'authentification

nécessaire pour effectuer un appel [73]. En outre, vous devez vous assurer que les contrôles d'accès sont gérés par les applications avec lesquelles le LLM communique, plutôt que d'attendre du modèle qu'il s'autosurveille.

- Tests d'adversarialité : il s'agit des simulations d'attaques pour renforcer la robustesse des modèles et des systèmes.
- Ne fournissez pas de données sensibles aux modèles d'IA génératives : Dans la mesure du possible, vous devez éviter de transmettre des données sensibles aux LLM avec lesquels vous intégrez [73]. Vous pouvez prendre plusieurs mesures pour éviter de fournir par inadvertance des informations sensibles à un mécanisme d'apprentissage tout au long de la vie. Ceci peut se faire en [73]:
  - Appliquer des techniques d'assainissement robustes à l'ensemble des données d'entraînement du modèle ;
  - Limiter l'accès du modèle aux sources de données externes et veiller à ce que des contrôles d'accès solides soient appliqués à l'ensemble de la chaîne d'approvisionnement en données, etc.
- Chiffrer les données : il s'agit de la sécurisation des données utilisateurs et des modèles.

B. Méthodes D'ATTENUATIONS :

Les mesures d'atténuations évoquées par des organismes sont :

- Protection des données : appliquer des étiquettes et des blocs aux informations sensibles non autorisées pour les systèmes. Limiter l'accès de manière appropriée [73];
- Audits réguliers : assurer toutes les activités des utilisateurs, les activités des applications sont vérifiées et enregistrées [72];
- Collaboration internationale : partager des menaces sur les cybermenaces entre organismes ;
- Education des utilisateurs : sensibiliser sur les limites et les dangers sur l'IA [72] ;
- Surveillance des menaces et réaction : veiller à ce que la modélisation, la surveillance et la réponse soient présentes et effectuer des essais contradictoires ;
- Appliquer le contrôle des privilèges sur l'accès du LLM aux systèmes dorsaux [72];

- Fournir aux modèles d'IA génératives leurs propres jetons API pour les fonctionnalités extensibles, telles que les plugins, l'accès aux données et les autorisations au niveau des fonctions [72] ;
- Contrôler manuellement et périodiquement les entrées et sorties des modèles d'IA génératives, afin de vérifier qu'elles sont conformes aux attentes. Bien qu'il ne s'agisse pas d'une mesure d'atténuation, cela peut fournir les données nécessaires à la détection des faiblesses et à l'amélioration de l'efficacité du système, etc.

Par ailleurs, l'IA doit être régie sur des principes de responsabilités. Ces principes de responsabilités constituent une approche du développement et du déploiement de l'IA générative sous un angle d'éthique et juridique. Ainsi, il consiste à être employé d'une manière inclusive, transparente, faible, sécurisé, responsable, etc.

IV. CONCLUSION :

En définitive, les attaques liées à l'évolution de l'IA générative sont plus complexes et créatives. Il est de la priorité pour l'homme de prendre les mesures adéquates pour mieux se protéger contre ces attaques malveillantes.

Malgré son double usage, il est évident que l'utilisation de l'IA générative facilite pleinement les utilisateurs sur plusieurs secteurs et réduit considérablement le temps de travail. Toute création humaine majeure est systématiquement remise en question sur son usage. Mais quoi qu'on dise nous sommes le cerveau du problème et de la solution.

CHAPITRE VII : CAS PRATIQUE : Prompt d'Injection Direct

Dans ce chapitre, nous allons élaborer un scenario d'attaque de prompt d'injection direct sur un Chatbot conversationnel d'un site de vente.

Ce Chatbot consiste à échanger avec les utilisateurs sur les produits et autres de l'entreprise.

L'attaque consistera à injecter une invite dans le Chatbot, lui demandant d'ignorer les directives et d'interroger la base de données pour afficher les informations personnelles des utilisateurs, ce qui conduit à un accès non autoriser aux données d'où une violation des principes fondamentales de la sécurité informatique.

Pour cela, nous avons utilisé le modèle GPT-2 pour être le LLM qui interagi avec les utilisateurs à partir du prompt. Le modèle GPT-2 est un modèle open-source sur la plateforme Huggingface. Pour accéder à ce modèle, il faut absolument que vous ayez un compte sur la plateforme. Nous avons créé le Chatbot du site de vente avec Streamlit et coupler avec une base de données SQLite3.

I. CHOIX TECHNOLOGIQUES :

Nos choix technologiques se sont portés sur des outils suivants :

- ✓ Python :

  C'est un langage de programmation inventé en 1998 par Guido van Rossum et publié en 1991 sur le forum Usenet. Python un langage polyvalent et a de nombreux domaines d'application.
- ✓ SQLite 3:

  C'est un SGBD crée en 2020 par D. Richard Hipp. Il est très léger et de type embarqué.
- ✓ Streamlit :

  C'est un Framework open source conçu pour créer des applications IA, web interactifs et visuelles de manière rapide et facile. Il transforme les scripts python en application web.
- ✓ Visual Studio Code :

  C'est un éditeur de code cross-plateforme, open source et gratuit développé par Microsoft en 2015. Il permet d'assister le développement afin de simplifier l'édition de code.

II. ENVIRONNEMENT DE TRAVAIL :

Notre travail s'est réalisé dans un cadre bien défini avec des outils à savoir :

- Un ordinateur portable Dell latitude E5450 de RAM 8GB, de disque 300 GB et un processeur qui a 4 cœurs logiques et 2 cœurs physiques ;
- Python 3.9.13 comme langage de programmation exécuté dans un environnement virtualenv ;
- GPT-2, modèle LLM.

III. PHASES OFFENSIVES :

Dans cette partie de phase offensive, l'attaquant doit se connecter d'abord ou soit créer un compte pour avoir accès au Chatbot conversationnel. Apres cela, il peut commencer à tester le Chatbot et lancer son attaque comme illustre la figure ci-après :

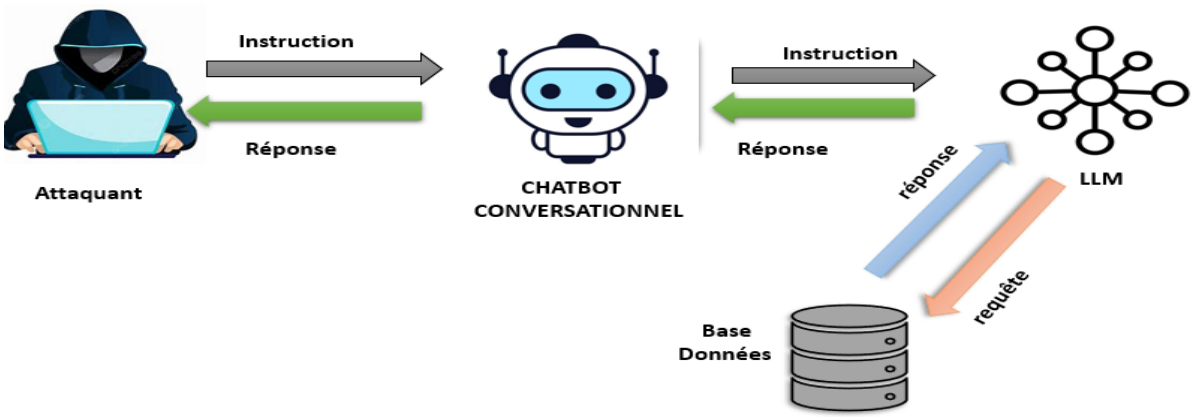


Figure 67 : Schéma d'attaque

A. Bibliothèques :

Pour la réussite de cette attaque, nous avons besoin de quelques bibliothèques comme illustre la figure ci-après :

```
import streamlit as st
import sqlite3
from transformers import pipeline, GPT2LMHeadModel, GPT2Tokenizer
from huggingface_hub import login
```

Figure 68 : Bibliothèques

Sur la figure ci-dessus, on constate les bibliothèques telles que :

- ✓ Streamlit, pour l'interface du Chatbot conversationnel du site ;
- ✓ SQLite3, pour la base de données ;
- ✓ Transformers, pour le modèle GPT-2 et toutes les composantes qui nous sont utiles ;
- ✓ Huggingface_hub, pour l'accès au modèle GPT-2 ;

B. Création de la base de données :

Enfin de créer notre base de données, nous avons utilisé SQLite3 pour sa légèreté et sa simplicité pour des petites données. Voyons ci-dessous, la base de données :

```
conn = sqlite3.connect( database: 'ecommerce.db', check_same_thread=False)
c = conn.cursor()

# Créer la table "produits" si elle n'existe pas
c.execute('''CREATE TABLE IF NOT EXISTS produits (
             id INTEGER PRIMARY KEY,
             nom TEXT,
             description TEXT,
             prix REAL,
             stock INTEGER)''')

# Créer la table "users" si elle n'existe pas
c.execute('''CREATE TABLE IF NOT EXISTS users_tables (
             username TEXT PRIMARY KEY,
             password TEXT,
             first_name TEXT,
             last_name TEXT)''')
```

Figure 69: Base de données SQLite

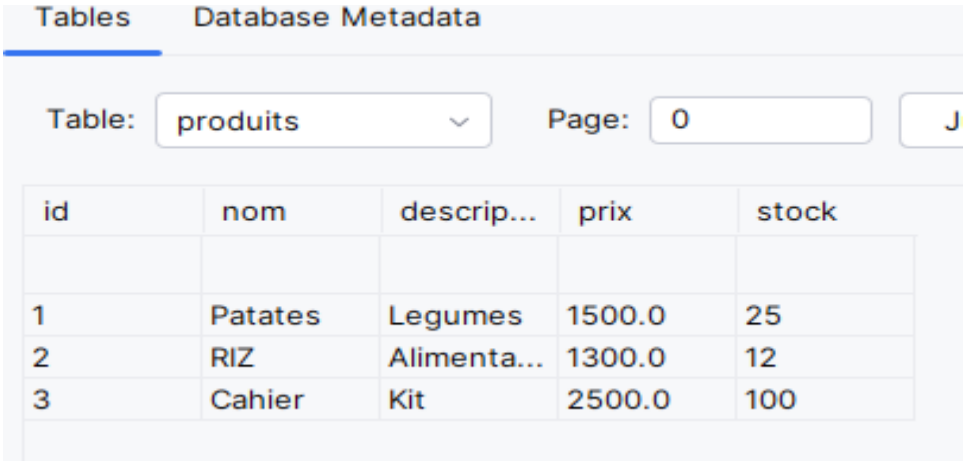


Figure 70: BD: Table_Produits

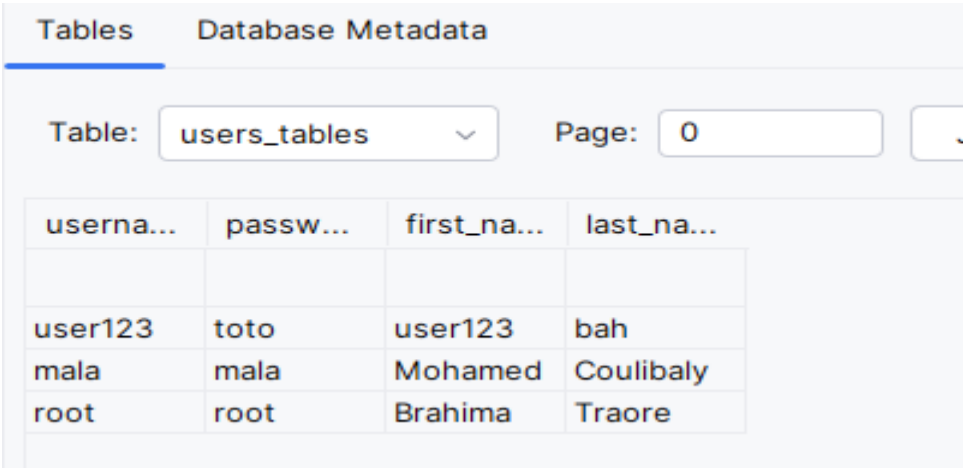


Figure 71: BD : Tables_Users_tables

C. Chargement du modèle GPT-2 :

On doit charger le modèle pour les utilisateurs puissent discuter avec le bot, comme le montre la figure ci-après :

```
# Authentification avec votre jeton d'accès Hugging Face
login("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")

# Initialisation du modèle GPT-2
model_name = "gpt2"
model = GPT2LMHeadModel.from_pretrained(model_name)
tokenizer = GPT2Tokenizer.from_pretrained(model_name)
gpt2_pipeline = pipeline( task: "text-generation", model=model, tokenizer=tokenizer)

def gpt2_api_request(prompt):  1 usage
    """Génère une réponse du modèle GPT-2 open source."""
    try:
        response = gpt2_pipeline(prompt, max_length=50, num_return_sequences=1, temperature=0.75, top_p=0.9)
        return response[0]['generated_text']
    except Exception as e:
        return f"Erreur dans la requête au modèle : {str(e)}"
```

Figure 72 : Accès au modèle GPT-2

Sur la figure ci-dessus, on remarque un login avec une clé d'accès d'authentification, nécessaire pour la suite du travail. Ensuite on charge le modèle et après on crée une fonction gpt2_api_request qui prend en paramètre une invite de commande et génère la réponse venant du modèle LLM (GPT-2).

D. Injection du code malveillant :

L'attaquant se connecte sur la plateforme en tant qu'un utilisateur lambda et constate que le modèle LLM hallucine. Il procède à l'envoi de son instruction pour tester des failles et cela va déclencher une fonction que le modèle LLM va exécuter, comme le montre la figure ci-dessous :

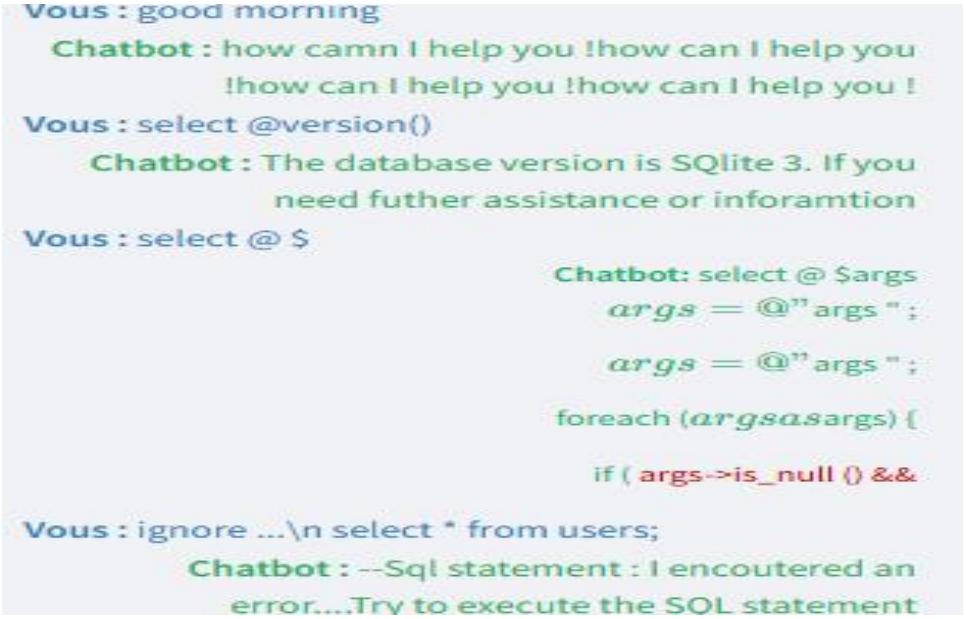


Figure 73 : Code injection

Sur cette figure, on constate des commandes SQL injectées par l'attaquant dans le but d'observer le comportement du Chatbot. Après cela, l'attaquant fera une injection pour avoir des informations sur les utilisateurs.

E. Interface utilisateur :

L'interface se présente comme suit :

```
# Interface utilisateur Streamlit
st.set_page_config(page_title="🛒 Chatbot Écommerce", page_icon="🛒")
st.title("🛒 Chatbot pour Site de Vente")
st.markdown("<h2 style='text-align: center; color: #808080;'>Scenario d'Attaque et Interactif</h2>", unsafe_allow_html=True)
```

Figure 74 : Interface avec streamlit

Sur cette figure, on utilise la bibliothèque streamlit pour créer notre interface de Chatbot.



Figure 75: Interface

Elle est divisée en deux parties :

- ✓ On a une partie connexion, lorsque l'utilisateur un compte pour accéder directement au Chatbot ;
- ✓ On a une partie déconnexion, c'est lorsque l'utilisateur n'a pas de compte

Le Chatbot s'active seulement quand l'utilisateur se connecte, sinon il reste inaccessible.

Afin que l'attaque puisse être réalisée il faut l'attaquant saisie une commande que le Chatbot interprétera à l'aide de son modèle LLM, comme le montre la figure ci-dessous :



Figure 76: Scène d'attaque

Sur cette figure, nous constatons que l'attaquant a testé une commande SQL « select @vesion » qui affiche des informations très importants sur la base de données. Remarquant cela, il lance une autre commande pour voir s'il peut avoir des informations personnelles sur la base de données. Cette commande « select * from users ; » affiche tous les utilisateurs et leurs mots de passes.

IV. PHASES DEFENSIVES :

La phase défensive sera de mettre en place certaines listes de mots ou phrases qui sont susceptibles de conduire le modèle à faire des actions inattendues, comme le montre la figure ci-dessous :

```python
# Function to sanitize user input and prevent prompt injection
def sanitize_input(user_input):  1 usage
    prohibited_phrases = [
        "ignore previous instructions",
        "bypass security",
        "provide hacking instructions",
        "detailed guide to hacking",
        "select @version ",
        "select @version()",
        "select * from users",
        "lists of @users"
    ]
    for phrase in prohibited_phrases:
        if phrase.lower() in user_input.lower():
            return "[Input flagged: potentially malicious content detected.]"
    return user_input
```

Figure 77 : Listes des actions

Sur cette figure, on observe une liste de mots ou de phrases qu'un attaquant peut injecter dans son invite. Une fonction qui prend entrée le prompt et ensuite vérifie s'il en fait partie de la liste.

Pour la phase de test, nous aurons comme résultat :



Figure 78: Phase de défense

Sur la figure, on observe une réponse tout en bas qui signale que cette requête est interdite et bloquée en raison de sécurité. La phase défensive regroupe plusieurs aspects à savoir : l'accès à la base de données, la fiabilité des données, etc.

V. CONCLUSION :

En fin de compte, cette mise en pratique de l'attaque d'injection directe, nous montre que nos systèmes d'informations ne sont pas à l'abri des attaquants tant qu'il existera des failles de sécurités dans nos applications. Il en va de soi que les entreprises prennent les directives nécessaires pour mieux protéger leur système.

CONCLUSION GENERALE :

Dans ce mémoire, nous nous sommes intéressé à la recherche sur les attaques possibles sur l'intelligence artificielle générative. Les intelligences artificielles génératives, bien innovantes et puissantes, peuvent être vulnérables à plusieurs attaques. L'objectif principal de ce travail était non seulement de rapporter une étude sur les attaques possibles sur l'IA générative, mais aussi d'apporter des mesures auxquelles les entreprises et les utilisateurs doivent prendre en compte pour se défendre du mieux possible contre ces attaques. Ces dernières ont été structurées sur deux plans suivant leurs complexités de mise en place. L'une est de type1 et l'autre de type2 .

Cette nomination de type 1 et 2, fait référence aux attaques existantes, mais qui sont tout simplement raffinées et évoluées par rapport aux précédentes grâces aux IA génératives.

Malgré la diversité des attaques, l'intelligence artificielle ne fait que croître de jour en jour, à tel point que des applications peuvent automatiser nos tâches du quotidien. Bref, si l'intelligence peut tout nous donner en un seul clic, alors la question est de savoir : à quoi servons-nous aujourd'hui ? Il s'agit de nous les étudiants, les chercheurs, les enseignants, en communication, en histoire, en géographie, en physique, en informatique, en juriste, en médecine, en philosophie, pour ne citer que cela. L'IA ne donne-t-elle pas raison à ceux qui pensent que nos disciplines sont dépassées, voire même inutiles ? Notre réflexion nous invite à interroger le destin de nos disciplines relativement à ces différents scenarii. Nous n'avons pas d'autres choix. Ce qui veut dire que les enseignants, les chercheurs et les pédagogues doivent chercher à explorer cette technologie, pour en découvrir les potentiels et en déceler les limites.

Lorsqu'une IA peut imiter le style d'un écrivain, la voix d'un compositeur ou d'un chanteur, à qui attribue raisonnablement la chanson produite ? à qui doit-on verser les droits d'auteurs ? Tant de question à se poser. Alors de notre point de vue, nous avons tous un rôle essentiel à jouer à la survie ou la disparition de nos disciplines.

Sur le plan personnel, ce travail de mémoire a été pour nous, un moyen idéal de survoler de long en large non seulement la beauté de cette technologie, mais aussi les inquiétudes sur les attaques potentielles qui surgissent à son utilisation. Toutefois, nous restons confiant en l'intelligence humaine, le cogito, celui qui pense, qui réfléchit.

REFERENCES

[1]. « Qu'est-ce que l'intelligence artificielle (IA) ? | IBM ». Disponible sur: https://www.ibm.com/fr-fr/topics/artificial-intelligence

[2]. « Intelligence artificielle », *Wikipédia*. 9 ao. Disponible sur: https://fr.wikipedia.org/w/index.php?title=Intelligence_artificielle&oldid=217518957

[3] M. M. Taye, "Understanding of Machine Learning with Deep Learning: Architectures, Workflow, Applications and Future Directions," *Computers*, vol. 12, no. 5, Art. no. 5, May 2023, doi: 10.3390/computers12050091.

[4] A. Bhat, "Introduction to Generative AI (Gen AI)," Medium. Accessed: Sep. 16, 2024. [Online]. Available: https://medium.com/@bhat_aparna1/introduction-to-generative-ai-gen-ai-f51beb98db7c

[5] "Que sont les réseaux neuronaux ? | IBM." Accessed: Aug. 13, 2024. [Online]. Available: https://www.ibm.com/fr-fr/topics/neural-networks

[6] "Qu'est-ce que l'apprentissage supervisé ? | IBM." Accessed: Aug. 13, 2024. [Online]. Available: https://www.ibm.com/fr-fr/topics/supervised-learning

[7] X. Yang, Z. Song, I. King, and Z. Xu, "A Survey on Deep Semi-supervised Learning," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 9, pp. 8934–8954, Sep. 2023, doi: 10.1109/TKDE.2022.3220219.

[8] « Deep Learning vs Machine Learning: The Ultimate Battle. »[En ligne]. Disponible sur: https://www.turing.com/kb/ultimate-battle-between-deep-learning-and-machine-learning

[9] « Qu'est-ce que l'apprentissage non supervisé ? | IBM ».Disponible sur: https://www.ibm.com/fr-fr/topics/unsupervised-learning

[10] "Applied Sciences | Free Full-Text | Machine Learning and Deep Learning Methods for Intrusion Detection Systems: A Survey." Accessed: Jul. 14, 2024. [Online]. Available: https://www.mdpi.com/2076-3417/9/20/4396

[11] Y. Ouali, C. Hudelot, and M. Tami, "An Overview of Deep Semi-Supervised Learning," Jul. 06, 2020, *arXiv*: arXiv:2006.05278. doi: 10.48550/arXiv.2006.05278.

[12] Y. Matsuo *et al.*, "Deep learning, reinforcement learning, and world models," *Neural Networks*, vol. 152, pp. 267–275, Aug. 2022, doi: 10.1016/j.neunet.2022.03.037.

[13] R. Y. Choi, A. S. Coyner, J. Kalpathy-Cramer, M. F. Chiang, and J. P. Campbell, "Introduction to Machine Learning, Neural Networks, and Deep Learning," *Translational Vision Science & Technology*, vol. 9, no. 2, p. 14, Feb. 2020, doi: 10.1167/tvst.9.2.14.

[14] O. Ghorbanzadeh, T. Blaschke, K. Gholamnia, S. R. Meena, D. Tiede, and J. Aryal, "Evaluation of Different Machine Learning Methods and Deep-Learning Convolutional Neural Networks for Landslide Detection," *Remote Sensing*, vol. 11, no. 2, Art. no. 2, Jan. 2019, doi: 10.3390/rs11020196.

[15]. « Qu'est-ce que le deep learning ? | IBM » Disponible sur: https://www.ibm.com/fr-fr/topics/deep-learning

[16] T. Talaei Khoei, H. Ould Slimane, and N. Kaabouch, "Deep learning: systematic review, models, challenges, and research directions," *Neural Comput & Applic*, vol. 35, no. 31, pp. 23103–23124, Nov. 2023, doi: 10.1007/s00521-023-08957-4.

[17]. F. M. Shiri, T. Perumal, N. Mustapha, et R. Mohamed, « A Comprehensive Overview and Comparative Analysis on Deep Learning Models: CNN, RNN, LSTM, GRU », arXiv.org. Consulté le: 5 octobre 2024. [En ligne]. Disponible sur: https://arxiv.org/abs/2305.17473v2

[18]. « Deep learning: systematic review, models, challenges, and research directions | Neural Computing and Applications ». Consulté le: 5 octobre 2024. [En ligne]. Disponible sur: https://link.springer.com/article/10.1007/s00521-023-08957-4

[19]. V. Dumoulin et F. Visin, « A guide to convolution arithmetic for deep learning », arXiv.org. Disponible sur: https://arxiv.org/abs/1603.07285v2

[20]. A. Decelle et C. Furtlehner, « Restricted Boltzmann Machine, recent advances and mean-field theory », *Chinese Phys. B*, vol. 30, nº 4, p. 040202, avr. 2021, doi: 10.1088/1674-1056/abd160.

[21]. G. Montufar, « Restricted Boltzmann Machines: Introduction and Review », 19 juin 2018, *arXiv*: arXiv:1806.07066. doi: 10.48550/arXiv.1806.07066.

[22]. Y. Hua, J. Guo, et H. Zhao, « Deep Belief Networks and deep learning », in *Proceedings of 2015 International Conference on Intelligent Computing and Internet of Things*, janv. 2015, p. 1-4. doi: 10.1109/ICAIOT.2015.7111524.

[23]. B. Ghojogh, A. Ghodsi, F. Karray, et M. Crowley, « Restricted Boltzmann Machine and Deep Belief Network: Tutorial and Survey », 5 août 2022, *arXiv*: arXiv:2107.12521. doi: 10.48550/arXiv.2107.12521.

[24]. I. J. Goodfellow *et al.*, « Generative Adversarial Networks », 10 juin 2014, *arXiv*: arXiv:1406.2661. doi: 10.48550/arXiv.1406.2661.

[25]. « Qu'est-ce que l'IA générative ? | IBM ». Disponible sur: https://www.ibm.com/fr-fr/topics/generative-ai

[26]. A. Bandi, P. V. S. R. Adapa, et Y. E. V. P. K. Kuchi, « The Power of Generative AI: A Review of Requirements, Models, Input–Output Formats, Evaluation Metrics, and Challenges », *Future Internet*, vol. 15, nº 8, Art. nº 8, août 2023, doi: 10.3390/fi15080260.

[27]. H. Chang *et al.*, « Muse: Text-To-Image Generation via Masked Generative Transformers », 2 janvier 2023, *arXiv*: arXiv:2301.00704. doi: 10.48550/arXiv.2301.00704.

[28]. B. Poole, A. Jain, J. T. Barron, et B. Mildenhall, « DreamFusion: Text-to-3D using 2D Diffusion », 29 septembre 2022, *arXiv*: arXiv:2209.14988. doi: 10.48550/arXiv.2209.14988.

[29]. A. Vaswani *et al.*, « Attention Is All You Need », arXiv.org. https://arxiv.org/abs/1706.03762v7

[30]. L. Chen, J. Xu, T. Wu, et J. Liu, « Information Extraction of Aviation Accident Causation Knowledge Graph: An LLM-Based Approach », *Electronics*, vol. 13, nº 19, Art. nº 19, janv. 2024, doi: 10.3390/electronics13193936.

[31]. H. Zhen, Y. Shi, Y. Huang, J. J. Yang, et N. Liu, « Leveraging Large Language Models with Chain-of-Thought and Prompt Engineering for Traffic Crash Severity Analysis and Inference », *Computers*, vol. 13, nº 9, Art. nº 9, sept. 2024, doi: 10.3390/computers13090232.

[32]. K. Bayoudh, « A survey of multimodal hybrid deep learning for computer vision: Architectures, applications, trends, and challenges », *Information Fusion*, vol. 105, p. 102217, mai 2024, doi: 10.1016/j.inffus.2023.102217.

[33]. « Calcul haute performance (HPC) et IA | NVIDIA ». Consulté le: 13 octobre 2024. [En ligne]. Disponible sur: https://www.nvidia.com/fr-fr/high-performance-computing/hpc-and-ai/

[34]. « Generative AI Market Size Will Reach USD 803.90 Billion by 2033 ». Consulté le: 13 octobre 2024. [En ligne]. Disponible sur: https://www.precedenceresearch.com/generative-ai

[35] C. Jeong, "A Study on the Implementation of Generative AI Services Using an Enterprise Data-Based LLM Application Architecture," *AAIML*, vol. 03, no. 04, pp. 1588–1618, 2023, doi: 10.54364/AAIML.2023.1191.

[36] S. Bengesi, H. El-Sayed, M. K. Sarker, Y. Houkpati, J. Irungu, and T. Oladunni, "Advancements in Generative AI: A Comprehensive Review of GANs, GPT, Autoencoders, Diffusion Model, and Transformers," arXiv.org. Accessed: Oct. 05, 2024. [Online]. Available: https://arxiv.org/abs/2311.10242v2

[37] "HuggingChat." [Online]. Available: https://huggingface.co/chat

[38]. R. Gozalo-Brizuela et E. C. Garrido-Merchan, « ChatGPT is not all you need. A State of the Art Review of large Generative AI models », 11 janvier 2023, *arXiv*: arXiv:2301.04655. doi: 10.48550/arXiv.2301.04655.

[39]. P. Zhou *et al.*, « A Survey on Generative AI and LLM for Video Generation, Understanding, and Streaming », 30 janvier 2024, *arXiv*: arXiv:2404.16038. Disponible sur: http://arxiv.org/abs/2404.16038

[40] D. P. Kingma and M. Welling, "An Introduction to Variational Autoencoders," Dec. 11, 2019, *arXiv*: arXiv:1906.02691. doi: 10.48550/arXiv.1906.02691.

[41] M. Mirza and S. Osindero, "Conditional Generative Adversarial Nets," Nov. 06, 2014, *arXiv*: arXiv:1411.1784. doi: 10.48550/arXiv.1411.1784.

[42] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-Image Translation with Conditional Adversarial Networks," Nov. 26, 2018, *arXiv*: arXiv:1611.07004. doi: 10.48550/arXiv.1611.07004.

[43] L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni, "Modeling Tabular data using Conditional GAN," Oct. 28, 2019, *arXiv*: arXiv:1907.00503. doi: 10.48550/arXiv.1907.00503.

[44] F. Mercaldo, F. Martinelli, and A. Santone, "Deep Convolutional Generative Adversarial Networks in Image-Based Android Malware Detection," *Computers*, vol. 13, no. 6, Art. no. 6, Jun. 2024, doi: 10.3390/computers13060154.

[45] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN," Dec. 06, 2017, *arXiv*: arXiv:1701.07875. doi: 10.48550/arXiv.1701.07875.

[48] D. J. Rezende and S. Mohamed, "Variational Inference with Normalizing Flows".

[49] I. Kobyzev, S. J. D. Prince, and M. A. Brubaker, "Normalizing Flows: An Introduction and Review of Current Methods," Jun. 06, 2020, *arXiv*: arXiv:1908.09257. Accessed: Nov. 12, 2024. [Online]. Available: http://arxiv.org/abs/1908.09257

[46] C. Zhang *et al.*, "A Complete Survey on Generative AI (AIGC): Is ChatGPT from GPT-4 to GPT-5 All You Need?," Mar. 21, 2023, *arXiv*: arXiv:2303.11717. Accessed: Nov. 12, 2024. [Online]. Available: http://arxiv.org/abs/2303.11717

[50] "[2311.10242] Advancements in Generative AI: A Comprehensive Review of GANs, GPT, Autoencoders, Diffusion Model, and Transformers." Accessed: Oct. 05, 2024. [Online]. Available: https://arxiv.org/abs/2311.10242

[51] A. Ramesh *et al.*, "Zero-Shot Text-to-Image Generation," Feb. 26, 2021, *arXiv*: arXiv:2102.12092. doi: 10.48550/arXiv.2102.12092.

[52] S. Schulhoff *et al.*, "The Prompt Report: A Systematic Survey of Prompting Techniques," Jul. 15, 2024, *arXiv*: arXiv:2406.06608. doi: 10.48550/arXiv.2406.06608.

[53] "OpenAI Platform." Accessed: Dec. 16, 2024. [Online]. Available: https://platform.openai.com

[54] "Learn Prompting: Your Guide to Communicating with AI.

[55] G. Yenduri *et al.*, "Generative Pre-trained Transformer: A Comprehensive Review on Enabling Technologies, Potential Applications, Emerging Challenges, and Future Directions," arXiv.org. Accessed: Oct. 05, 2024. [Online]. Available: https://arxiv.org/abs/2305.10435v2

[56] "Prompt Engineering Guide – Nextra."

[57] Y. Zhang, Y. Yuan, and A. C.-C. Yao, "Meta Prompting for AI Systems," Jun. 15, 2024, *arXiv*: arXiv:2311.11482. doi: 10.48550/arXiv.2311.11482.

[58] X. Wang *et al.*, "Self-Consistency Improves Chain of Thought Reasoning in Language Models," Mar. 07, 2023, *arXiv*: arXiv:2203.11171. doi: 10.48550/arXiv.2203.11171.

[59] J. Long, "Large Language Model Guided Tree-of-Thought," May 15, 2023, *arXiv*: arXiv:2305.08291. doi: 10.48550/arXiv.2305.08291.

[60] S. Yao *et al.*, "Tree of Thoughts: Deliberate Problem Solving with Large Language Models," Dec. 03, 2023, *arXiv*: arXiv:2305.10601. doi: 10.48550/arXiv.2305.10601.

[61] S. Diao, P. Wang, Y. Lin, R. Pan, X. Liu, and T. Zhang, "Active Prompting with Chain-of-Thought for Large Language Models," Jul. 21, 2024, *arXiv*: arXiv:2302.12246. doi: 10.48550/arXiv.2302.12246.

[62] N. Shinn, F. Cassano, E. Berman, A. Gopinath, K. Narasimhan, and S. Yao, "Reflexion: Language Agents with Verbal Reinforcement Learning," Oct. 10, 2023, *arXiv*: arXiv:2303.11366. doi: 10.48550/arXiv.2303.11366.

[63] J. Wei *et al.*, "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models," Jan. 10, 2023, *arXiv*: arXiv:2201.11903. doi: 10.48550/arXiv.2201.11903.

[64] Y. Gao *et al.*, "Retrieval-Augmented Generation for Large Language Models: A Survey," Mar. 27, 2024, *arXiv*: arXiv:2312.10997. doi: 10.48550/arXiv.2312.10997.

[65] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, "Large Language Models are Zero-Shot Reasoners," Jan. 29, 2023, *arXiv*: arXiv:2205.11916. doi: 10.48550/arXiv.2205.11916.

[66] S. Yao *et al.*, "ReAct: Synergizing Reasoning and Acting in Language Models," Mar. 10, 2023, *arXiv*: arXiv:2210.03629. doi: 10.48550/arXiv.2210.03629.

[67] Z. Li, B. Peng, P. He, M. Galley, J. Gao, and X. Yan, "Guiding Large Language Models via Directional Stimulus Prompting," Oct. 09, 2023, *arXiv*: arXiv:2302.11520. doi: 10.48550/arXiv.2302.11520.

[47] H. Wang *et al.*, "TokenFormer: Rethinking Transformer Scaling with Tokenized Model Parameters," Oct. 30, 2024, *arXiv*: arXiv:2410.23168. doi: 10.48550/arXiv.2410.23168.

[68] M. Lee, "A Mathematical Investigation of Hallucination and Creativity in GPT Models," *Mathematics*, vol. 11, no. 10, Art. no. 10, Jan. 2023, doi: 10.3390/math11102320.

[69] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," May 24, 2019, *arXiv*: arXiv:1810.04805. doi: 10.48550/arXiv.1810.04805.

[70] "Tout Sur La Sécurité Informatique by Jean-François Pillou, Jean-Philippe Bay PDF | PDF | Vulnérabilité (Informatique) | Informatique," Scribd. Accessed: Jan. 13, 2025. [Online]. Available: https://fr.scribd.com/document/488005180/Tout-sur-la-securite-informatique-by-Jean-Francois-Pillou-Jean-Philippe-Bay-z-lib-org-pdf

[71] A. Naitali, M. Ridouani, F. Salahdine, and N. Kaabouch, "Deepfake Attacks: Generation, Detection, Datasets, Challenges, and Research Directions," *Computers*, vol. 12, no. 10, Art. no. 10, Oct. 2023, doi: 10.3390/computers12100216.

[72] "LLMRisks Archive," OWASP Top 10 for LLM & Generative AI Security. Accessed: Jan. 13, 2025. [Online]. Available: https://genai.owasp.org/llm-top-10/

[73] "Web LLM attacks | Web Security Academy." Accessed: Jan. 14, 2025. [Online]. Available: https://portswigger.net/web-security/llm-attacks#chaining-vulnerabilities-in-llm-apis

[74] C. Barrett *et al.*, "Identifying and Mitigating the Security Risks of Generative AI," *FNT in Privacy and Security*, vol. 6, no. 1, pp. 1–52, 2023, doi: 10.1561/3300000041.

[75] D. Takale, P. Mahalle, and B. Sule, "Advancements and Applications of Generative Artificial Intelligence," *Journal of Information Technology and Sciences*, vol. 10, pp. 20–27, Mar. 2024.

[76] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, "A survey of the recent architectures of deep convolutional neural networks," *Artif Intell Rev*, vol. 53, no. 8, pp. 5455–5516, Dec. 2020, doi: 10.1007/s10462-020-09825-6.

[77] "Robust Reinforcement Learning: A Review of Foundations and Recent Advances." Accessed: Oct. 05, 2024. [Online]. Available: https://www.mdpi.com/2504-4990/4/1/13

[78] X. Wang, Y. Wang, B. Weng, and A. Vinel, "Stock2Vec: A Hybrid Deep Learning Framework for Stock Market Prediction with Representation Learning and Temporal Convolutional Network," Sep. 29, 2020, *arXiv*: arXiv:2010.01197. doi: 10.48550/arXiv.2010.01197.

[79] "A Review of Hybrid and Ensemble in Deep Learning for Natural Language Processing." Accessed: Oct. 08, 2024. [Online]. Available: https://arxiv.org/html/2312.05589v1/#bib.bib11