

# Intelligence artificielle générative pour IT

Redha Moulla

Toulouse, 6 - 7 octobre 2025

# Plan de la formation

- Qu'est-ce que l'intelligence artificielle ?
- Éléments de machine learning et de deep learning
- IA générative
- Prompt engineering
- Les RAGs
- IAG agentique
- Intégration de l'IA générative
- Enjeux éthique, de sécurité et de conformité de l'IAG

# Qu'est-ce que l'intelligence artificielle ?

# Définition littérale de l'intelligence artificielle

## 1. Intelligence

Ensemble des fonctions mentales ayant pour objet la connaissance conceptuelle et relationnelle.

- Larousse

## 2. Artificielle

Qui est produit de l'activité humaine (opposé à la nature).

- Larousse

# Conférence de Dartmouth ?

Article

AI Magazine Volume 27 Number 4 (2006) (© AAAI)

## A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence

August 31, 1955

John McCarthy, Marvin L. Minsky,  
Nathaniel Rochester,  
and Claude A. Shannon

■ The 1956 Dartmouth summer research project on artificial intelligence was initiated by this August 31, 1955 proposal, submitted by John McCarthy, Marvin Minsky, Nathaniel Rochester, and Claude Shannon. The original typewritten document of 17 pages plus a title page. Copies of the typewritten are housed in the archives at Dartmouth College and Stanford University. The first 5 pages state the proposal, and the remaining pages give qualifications and interests of the four who proposed the study. In the interest of brevity, this article reproduces only the proposal itself, along with the short autobiographical statements of the proposers.

**W**e propose that a 2-month, 10-man study of artificial intelligence be carried out during the summer of 1956 at Dartmouth College in Hanover, New Hampshire. The study is to proceed on the basis of the conjecture that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it. An attempt will be made to find how to make machines use lan-

guage, form abstractions and concepts, solve kinds of problems now reserved for humans, and improve themselves. We think that a significant advance can be made in one or more of these problems if a carefully selected group of scientists work on it together for a summer. The following are some aspects of the artificial intelligence problems.

### 1. Automatic Computers

If a machine can do a job, then an automatic calculator can be programmed to simulate the machine. The speed and memory capacities of present computers may be insufficient to simulate many of the higher functions of the human brain, but the major obstacle is not lack of machine capacity, but our inability to write programs taking full advantage of what we have.

### 2. How Can a Computer be Programmed to Use a Language

It may be speculated that a large part of human thought consists of manipulating words according to the rules of meaning and rules of inference. From this point of view, forming a generalization consists of admitting a new

*"We propose that a 2-month, 10-man study of artificial intelligence be carried out during the summer of 1956 at Dartmouth College in Hanover, New Hampshire. The study is to proceed on the basis of the conjecture that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it. An attempt will be made to find how to make machines use language, form abstractions and concepts, solve kinds of problems now reserved for humans, and improve themselves. We think that a significant advance can be made in one or more of these problems if a carefully selected group of scientists work on it together for a summer."*

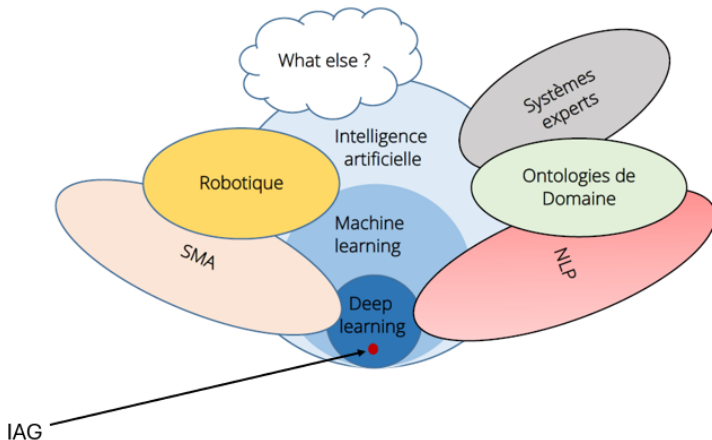
# L'intelligence artificielle selon John McCarthy

*"It is the science and engineering of making intelligent machines, especially intelligent computer programs. It is related to the similar task of using computers to understand human intelligence, but AI does not have to confine itself to methods that are biologically observable."*

— John McCarthy

# Définition pragmatique de l'intelligence artificielle

Il s'agit d'un ensemble de techniques qui permettent à la machine d'accomplir des tâches qui requièrent traditionnellement une intelligence humaine.



# Rappels sur le machine learning et le deep learning



# Qu'est-ce que la machine learning ?

L'apprentissage automatique est une branche de l'intelligence artificielle qui consiste à doter les machines de la capacité d'apprendre à partir de données sans que celles-ci ne soient explicitement programmées pour exécuter des tâches spécifiques.

Le machine learning englobe plusieurs types d'apprentissage :

- **Supervisé** : Les algorithmes apprennent à partir de données étiquetées pour faire des prédictions ou classifications.
- **Non supervisé** : L'apprentissage est effectué sur des données non étiquetées pour trouver des structures cachées.
- **Auto-supervisé** : La machine apprend à compléter des séquences de données (texte, images, etc.).
- **Par renforcement** : Les modèles apprennent à prendre des décisions en maximisant une récompense à travers des interactions.

# L'apprentissage supervisé

**L'apprentissage supervisé** consiste à apprendre un modèle qui associe une étiquette (*label*) à un ensemble de caractéristiques (*features*).

- **Inputs** : un jeu de données *annotées* pour entraîner le modèle.
  - Exemple : des textes (tweets, etc.) avec les *sentiment* associés, positifs ou négatifs.
- **Output** : une étiquette pour un point de donnée inconnu par le modèle.

L'apprentissage supervisé se décline lui-même en deux grandes familles :

- **La classification** : prédire une catégorie ou une classe.
  - Exemple : prédire l'étiquette d'une image (chat, chien, etc.), le sentiment associé à un texte, le centre d'intérêt d'un client à partir de ses commentaires, etc.
- **La régression** : prédire une valeur continue (un nombre réel typiquement).
  - Exemple : prédire le prix d'un appartement, la lifetime value d'un client, etc.

# Classification

## Exemple de classification : credit scoring

Âge	Revenu annuel (k€)	Historique de crédit	Nombre de cartes de crédit	Niveau d'éducation	Propriétaire immobilier (Oui/Non)	Label (y)
30	50	Bon	2	Licence	Oui	Accepté
45	80	Moyen	3	Master	Oui	Accepté
22	20	Mauvais	1	Bac	Non	Refusé
35	60	Bon	4	Licence	Oui	Accepté
40	70	Moyen	2	Bac+2	Non	Refusé

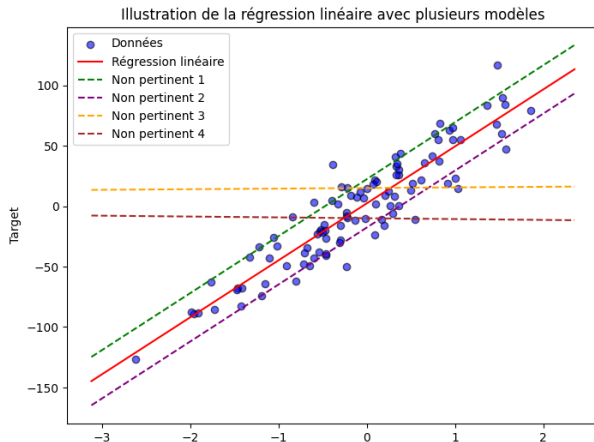
## Exemple de régression : prédiction des prix des logements

Surface (m <sup>2</sup> )	Nombre de chambres	Distance du centre-ville (km)	Année de construction	Quartier (Score 1-10)	Prix (k€) (y)
80	3	5	2010	8	300
120	4	10	2005	7	450
60	2	2	2020	9	200
150	5	15	1995	6	600
100	3	8	2015	7	400

# Régression linéaire simple

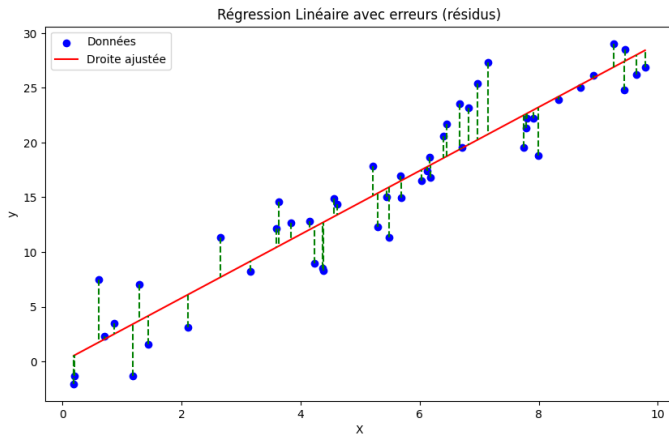
Soit un ensemble de  $n$  observations  $x_1, x_2, \dots, x_n$  avec les labels correspondants  $y_1, y_2, \dots, y_n$ , on cherche le modèle linéaire qui ajuste le mieux ces données.

$$\hat{y} = \beta_0 + \beta_1 x$$

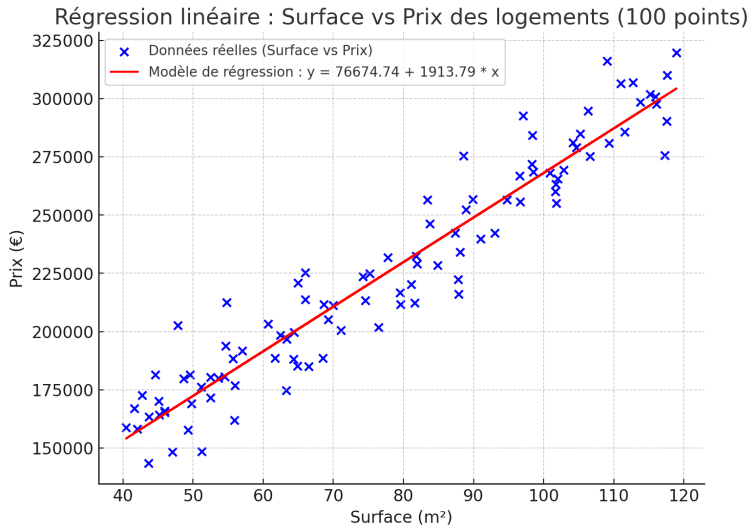


# Minimisation du risque empirique 1/2

L'erreur de prédiction pour la  $i$  ième observation est :  $e_i = y_i - \hat{y}_i$ . où  $\hat{y}_i = \beta_0 + \beta_1 x_i$ .



# Prédire le prix en fonction de la surface : inférence



# L'apprentissage non supervisé

**L'apprentissage non supervisé** se réfère à l'utilisation de modèles d'apprentissage automatique pour identifier des patterns et des structures dans des données qui ne sont pas étiquetées.

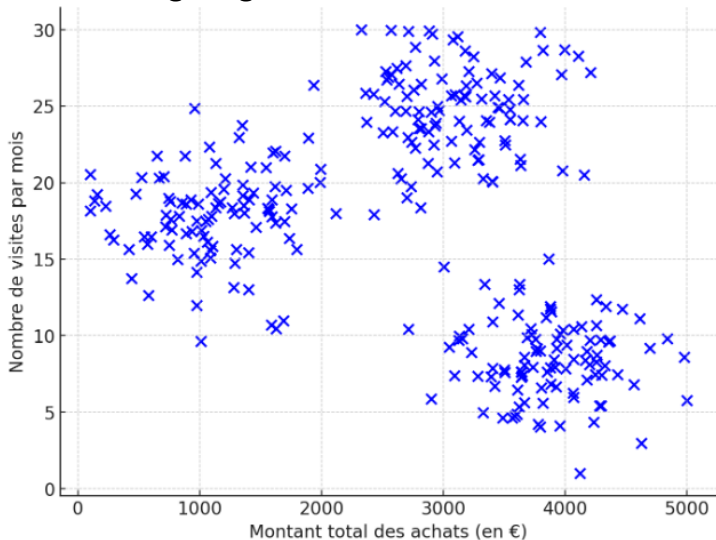
Principales typologies de l'apprentissage non supervisé :

- **Clustering** : Regroupement de points de données similaires ensemble.  
Exemple : segmentation de marché, regroupement social.
- **Détection d'anomalies** : Détecter des observations dont les caractéristiques sont inhabituelles par rapport à la majorité.



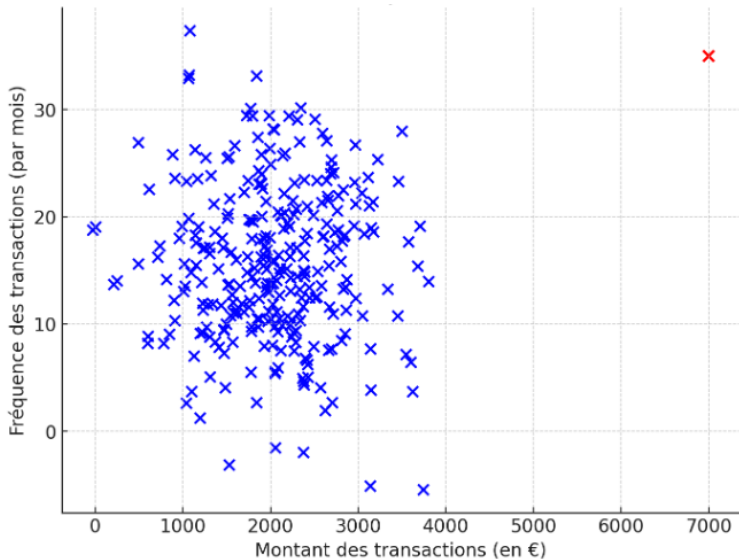
# Clustering

## Exemple de clustering : segmentation clients



# Détection d'anomalies

## Exemple de détection d'anomalies : fraude bancaire



# Eléments de deep learning

# Introduction aux réseaux de neurones

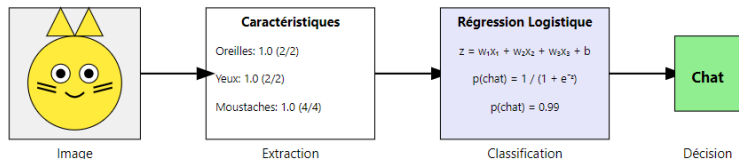
**Définition** : Les réseaux de neurones sont des modèles computationnels inspirés par le fonctionnement des neurones dans le cerveau humain. Ils sont capables d'apprendre des tâches complexes en modélisant des relations non linéaires entre les entrées et les sorties.

## Caractéristiques :

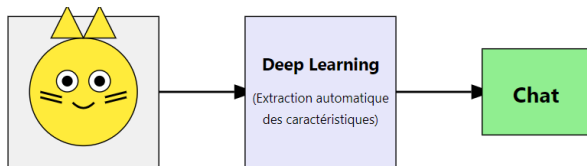
- **Extraction automatique des features** : Capacité d'adaptation et d'extraction des features à partir des données sans programmation explicite.
- **Modélisation non linéaire** : Aptitude à capturer des relations complexes dans les données.
- **Modélisation en grande dimension** : Les modèles de deep learning sont particulièrement adaptés pour les données en grande dimension (images, texte, etc.).
- **Flexibilité** : Applicables à un large éventail de tâches et de typologies de données (images, langage naturel, données graphiques, etc.).

# Extraction de features

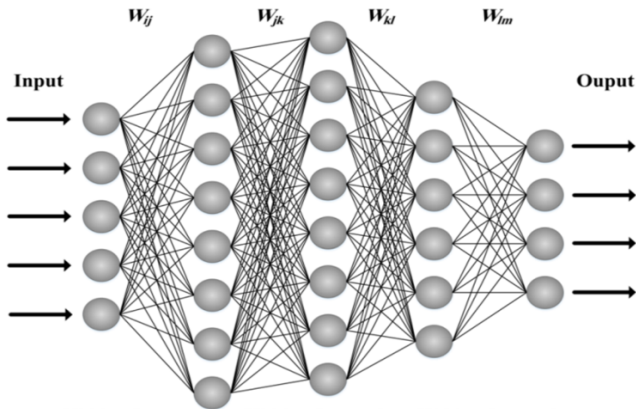
## Extraction de features en machine learning "classique"



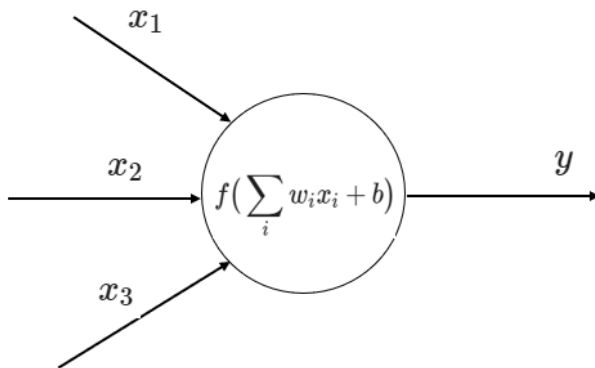
## Extraction de features en deep learning



# Illustration d'un réseau de neurones classique

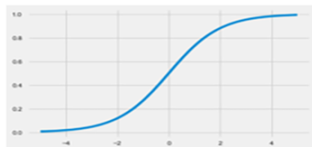


# Neurone artificiel

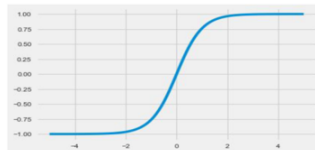


# Illustration des fonctions d'activation

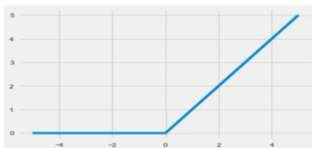
Sigmoïde



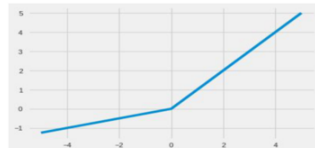
Tanh



ReLU

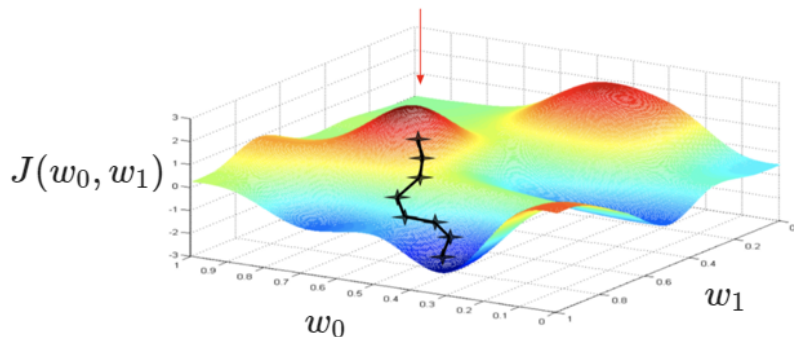


ReLU paramétrique





# Illustration graphique de la SGD



# Intelligence artificielle générative (IAG)

# Qu'est-ce que l'IA générative ?

**Définition** : L'intelligence artificielle générative (IAG) désigne les modèles capables de produire de nouvelles données (texte, image, son, code) qui imitent ou enrichissent des données existantes.

**Idée clé** : alors que le machine learning classique prédit, l'IAG crée.

## Caractéristiques :

- Modèles entraînés sur des volumes massifs de données hétérogènes.
- Génération conditionnée par un prompt (instruction utilisateur).
- Capacité à généraliser au-delà des données vues pendant l'entraînement.

# Deux grandes familles d'IA générative

## **IA générative pour le langage :**

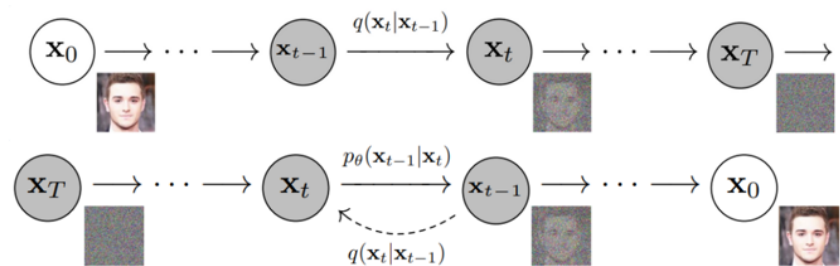
- Modèles de langage de grande taille (LLMs).
- Génération de texte cohérent et contextuel.
- Applications : chatbots, rédaction assistée, analyse de documents, génération de code.
- Exemples : GPT-4, Claude, LLaMA, Mistral.

## **IA générative pour les images :**

- Modèles de diffusion et GANs.
- Création d'images réalistes ou stylisées à partir de descriptions textuelles.
- Applications : design, marketing, prototypage, art numérique.
- Exemples : DALL-E, Stable Diffusion, MidJourney.

# Génération d'images

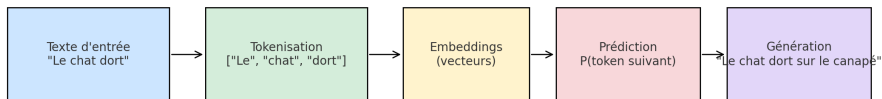
La génération d'images utilise généralement des techniques dites de diffusion.



# Principe de fonctionnement

## Étapes principales :

- Tokenisation : découper un texte en unités de base appelées tokens.
- Encodage : transformer les tokens en vecteurs numériques (embeddings).
- Prédiction : estimer la probabilité du token suivant à partir du contexte.
- Génération : produire un texte en générant un token après l'autre.



# Entraînement des LLMs

L'entraînement des LLMs se déroule généralement en trois étapes principales :

- ❶ **Pré-entraînement sur un corpus volumineux** : phase initiale sur un ensemble massif de données textuelles issues de diverses sources (livres, articles, sites web, code) afin d'apprendre une compréhension générale du langage. L'entraînement consiste à prédire le mot suivant dans une séquence de texte.
- ❷ **Fine-tuning supervisé** : adaptation du modèle à des tâches ou domaines spécifiques (dialogues, traduction, analyse de sentiments, code). Cette étape utilise des jeux de données annotés et ciblés.
- ❸ **Reinforcement Learning from Human Feedback (RLHF)** : ajustement par apprentissage par renforcement, où des annotateurs humains évaluent et classent les réponses du modèle pour affiner leur pertinence, leur clarté et leur alignement avec les attentes.

# Modèles de raisonnement

**Objectif** : améliorer la capacité des LLMs à résoudre des problèmes complexes en rendant explicites les étapes de raisonnement.

## Principe des Chain-of-Thoughts (CoT) :

- Décomposer une question en **étapes intermédiaires logiques**.
- Forcer le modèle à **“penser à voix haute”** avant de donner la réponse finale.
- Exemple : résoudre une équation, planifier une suite d'actions, raisonner sur des données tabulaires.

## Méthodes d'entraînement :

- **Fine-tuning supervisé** sur des datasets annotés avec étapes de raisonnement.
- **Self-consistency** : générer plusieurs chaînes et agréger la réponse la plus fréquente/cohérente.
- **Distillation** : transférer les capacités de raisonnement d'un grand modèle vers un plus petit.



# Agents basés sur les LLMs

**Définition** : un agent est un système qui exploite un LLM comme moteur de raisonnement et de décision, capable d'interagir avec des outils et un environnement externe.

## Caractéristiques principales :

- **Boucle perception–action** : observe, raisonne, agit, et réévalue.
- **Accès à des outils** : API, bases de données, navigateurs, scripts.
- **Mémoire** : conserve l'historique et les connaissances pertinentes.
- **Planification** : décompose un objectif en sous-tâches exécutables.

## Exemples d'agents :

- **AutoGPT, BabyAGI** : agents autonomes exécutant des suites de tâches complexes.
- **MetaGPT** : coordination automatique de rôles dans un projet logiciel (PO, dev, QA).

# LLMs propriétaires vs Open Source

## ● Modèles propriétaires

- ChatGPT, Claude, Gemini, Grok...
- Poids non disponibles au public.
- Performances élevées, accès limité par API payante
- Protection intellectuelle stricte (code non accessible)
- Support technique assuré, documentation avancée
- Dépendance aux fournisseurs et coûts potentiellement élevés

## ● Modèles open source

- Llama, Mistral, Gemma, DeepSeek, Qwen...
- Poids disponibles au public
- Transparence du code, flexibilité d'adaptation
- Performances parfois inférieures, mais amélioration continue
- Gratuit, mais coût de l'infrastructure à gérer
- Exposition aux failles potentielles de sécurité, support communautaire

**Conclusion :** Choisir entre contrôle, flexibilité et performances optimales.

# Comparaison des performances des LLMs : MLE Arena

## Text

🕒 5 days ago

Rank (UB) ↑ Model ↑↓ Score ↑↓

1  gemini-2.5-pro 1456

1  gpt-5-high 1447

1  claude-opus-4-1-20250805-thi... 1447

2  o3-2025-04-16 1444

2  chatgpt-4o-latest-20250326 1443

2  gpt-4.5-preview-2025-02-27 1439


2  claude-opus-4-1-20250805 1436

7  gpt-5-chat 1426

## WebDev

🕒 12 days ago

Rank (UB) ↑ Model ↑↓ Score ↑↓

1  GPT-5 (high) 1481

1  Claude Opus 4.1 thinking-16k... 1474

3  Claude Opus 4.1 (20250805) 1436

4  Gemini-2.5-Pro 1405

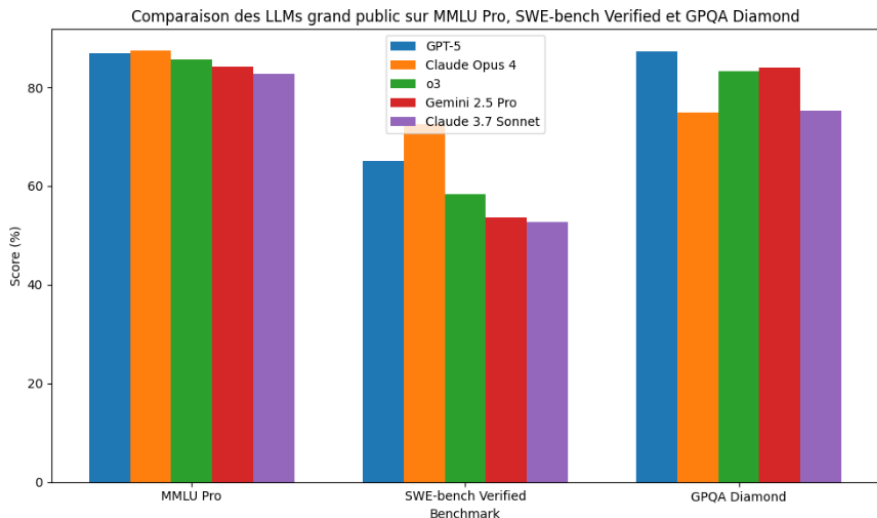
4  DeepSeek-R1-0528 1392

5  Claude Opus 4 (20250514) 1383

6  GLM-4.5 1368

6  GLM-4.5-Air 1364

# Comparaison des performances des LLMs : benchmarks classiques



# Introduction au prompting

**Définition** : Le prompting est l'art de formuler des instructions à un modèle de langage afin d'obtenir une réponse pertinente et de qualité.

**Idée clé** : La qualité de la sortie dépend fortement de la manière dont l'entrée est rédigée.

**Enjeux** :

- Obtenir des résultats précis et cohérents.
- Réduire les risques d'ambiguïtés et d'hallucinations.
- Adapter le style et le niveau de détail de la réponse.

# Techniques de base du prompting

- **Prompt clair et explicite** : éviter les formulations vagues.
- **Contexte** : fournir des informations supplémentaires pour orienter le modèle.
- **Format attendu** : indiquer la structure de sortie souhaitée (liste, tableau, texte court).
- **Exemples** : montrer un ou plusieurs cas pour guider la génération.

Exemple : Au lieu de "Explique le machine learning", écrire "Donne une définition simple du machine learning, suivie d'un exemple d'application en santé".

# Techniques avancées

- **Few-shot prompting** : fournir quelques exemples d'entrée-sortie pour calibrer les réponses.
- **Chain-of-thought** : inciter le modèle à détailler son raisonnement étape par étape.
- **Role prompting** : demander au modèle d'adopter un rôle précis (enseignant, expert technique...).
- **Prompt engineering** : expérimentation systématique pour optimiser les formulations.

# Bonnes pratiques de prompting

- Commencer simple, puis raffiner progressivement.
- Tester plusieurs formulations pour comparer les résultats.
- Toujours vérifier la fiabilité des réponses produites.
- Documenter les prompts efficaces pour réutilisation.



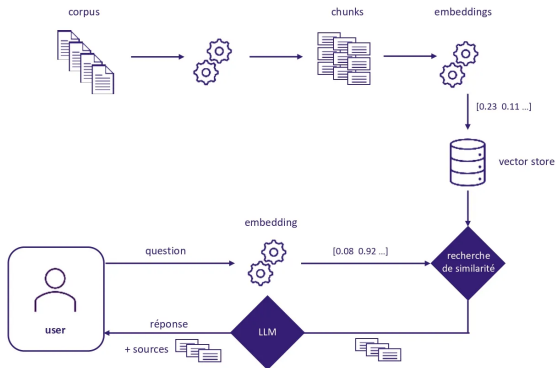
# Les RAGs

# Qu'est-ce qu'un Retrieval-Augmented Generation (RAG) ?

- **Définition** : Un modèle RAG combine la recherche d'information (*retrieval*) avec la génération de texte pour améliorer la précision et la pertinence des réponses.
- **Principe** : Le modèle récupère d'abord des documents pertinents à partir d'une base de données, puis les utilise pour générer une réponse enrichie.
- **Avantages** : Réduction des hallucinations, accès à des connaissances actualisées et contrôlées, amélioration de l'explicabilité des réponses.
- **Cas d'usage** : Chatbots spécialisés, assistants de recherche, génération de résumés, support client automatisé.

# Principes du RAG

## RAG basique



# Stratégies de chunking

- **Définition** : Le chunking consiste à segmenter un document en morceaux (chunks) pour optimiser la récupération et la génération d'informations dans un système RAG.
- **Principales stratégies** :
  - **Fixed-size chunking** : découpage en segments de longueur fixe (ex : 512 tokens), simple mais peut fragmenter le contexte.
  - **Overlapping chunks** : chevauchement entre les segments pour conserver le contexte et éviter la perte d'informations critiques.
  - **Semantic chunking** : segmentation basée sur la structure du texte (paragraphe, titres, sections) ou l'analyse sémantique.
  - **Recursive chunking** : division progressive en fonction de la granularité des informations pertinentes.
- **Exemples d'outils** : LangChain, NLTK, SpaCy, tiktoken.
- **Avantages** : Améliore la récupération d'informations, réduit le bruit et optimise l'usage des modèles génératifs.

# Bases de données vectorielles

- **Définition** : Stockent et indexent des représentations numériques (embeddings) pour effectuer des recherches sémantiques efficaces.
- **Principe** : Chaque document est transformé en un vecteur dense dans un espace de grande dimension. La recherche repose sur la similarité (cosinus, ANN, etc.) plutôt que sur des mots-clés.

The FAISS logo is a rectangular box with a gradient background transitioning from dark blue on the left to light green on the right. The word "FAISS" is written in white, bold, uppercase letters in the center.

FAISS

# Approches avancées du RAG

- **Reranking** : Amélioration de la sélection des documents récupérés en utilisant un modèle plus puissant (ex : cross-encoder, Cohere Rerank, ColBERT).
- **Fine-tuning du retrieveur** : Optimisation de l'indexation et du scoring des documents avec un modèle entraîné spécifiquement sur le domaine d'application.
- **Graph RAG** : Utilisation d'un graphe de connaissances pour structurer les relations entre documents et améliorer la récupération d'information.
- **Agent RAG** : Intégration d'agents conversationnels capables d'exécuter des actions (recherches multiples, vérification des réponses) avant de générer une réponse finale.

# Reranking dans le RAG

- **Définition** : Le reranking consiste à réordonner les documents récupérés pour améliorer la pertinence des résultats avant de les passer au modèle génératif.
- **Méthodes principales** :
  - **BM25 + Reranker** : recherche lexicale suivie d'un modèle neuronal qui affine le classement des documents.
  - **Cross-encoder** : modèle BERT-like qui évalue chaque document en prenant en compte à la fois la requête et le contenu.
  - **Dense Reranking** : ajuste les scores des documents en utilisant des embeddings appris sur des données spécifiques.
- **Exemples d'outils** : Cohere Rerank, ColBERT, MonoT5, Rank-BERT.
- **Avantages** : Meilleure précision des résultats, réduction du bruit, amélioration des performances du modèle génératif.

# Fine-tuning dans le RAG

- **Définition** : Le fine-tuning consiste à ajuster un modèle pré-entraîné sur un ensemble de données spécifique pour améliorer ses performances dans un domaine précis.
- **Types de fine-tuning** :
  - **Fine-tuning du retrieveur** : amélioration de la récupération des documents en entraînant un modèle dense (ex : Contriever, ColBERT) sur un corpus spécialisé.
  - **Fine-tuning du modèle génératif** : adaptation d'un LLM pour mieux exploiter les documents récupérés et générer des réponses plus pertinentes.
  - **Fine-tuning hybride** : entraînement simultané du retrieveur et du modèle génératif pour une synergie optimale.
- **Avantages** : L'avantage principal est une meilleure spécialisation du modèle.



# Graph RAG

- **Définition** : Le Graph RAG utilise une structure de graphe pour organiser et récupérer l'information de manière plus contextuelle et structurée.
- **Principes clés** :
  - **Relations entre documents** : les nœuds représentent des documents, concepts ou entités, et les arêtes définissent leurs connexions.
  - **Recherche basée sur la connectivité** : la récupération d'informations se fait en explorant les relations entre les nœuds pour trouver les réponses les plus pertinentes.
  - **Augmentation du contexte** : le modèle peut utiliser des chemins sémantiques dans le graphe pour enrichir le prompt avec des données plus cohérentes.
- **Exemples d'outils** : Neo4j, NetworkX, Weaviate.
- **Avantages** : Meilleure structuration des connaissances, récupération d'information plus précise, réduction des erreurs contextuelles.

# Agent RAG

- **Définition** : L'Agent RAG intègre un agent autonome qui orchestre la récupération et l'exploitation des documents pour affiner la génération de réponses.
- **Principes clés** :
  - **Multi-recherches adaptatives** : l'agent effectue plusieurs requêtes en fonction du contexte et ajuste dynamiquement le retrieveur.
  - **Vérification et correction** : l'agent peut analyser la réponse générée, détecter d'éventuelles erreurs et reformuler la requête si nécessaire.
  - **Actions spécifiques** : l'agent peut interagir avec des bases de données, des API externes ou exécuter du code pour enrichir la réponse finale.
- **Exemples d'outils** : LangChain Agents, CrewAI, etc.
- **Avantages** : Plus grande autonomie, récupération d'informations plus dynamique, réduction des erreurs génératives.

# IA générative agentique

# Définition d'un agent IA

- **Autonomie** : Capacité à exécuter des actions de bout en bout sans supervision constante, sous contraintes (budgets, rôles, règles).
- **Objectifs** : Définis explicitement (tâche métier) ou dérivés d'un contexte (logs, tickets, KPI).
- **Outils** : APIs, bases de données, systèmes de fichiers, services cloud, commandes internes.
- **Cadre de gouvernance** : Autorisations, limitations, journalisation, relecture humaine ciblée.
- **Exemple concret** : Un agent d'exploitation qui diagnostique une alerte serveur et propose un correctif.

# Fonctionnement d'un agent basé sur LLM

- **Perception** : Interprétation de l'entrée (requête utilisateur, logs systèmes, contexte IT).
- **Planification** : Décomposition en sous-tâches, choix des outils, séquence d'exécution.
- **Action** : Appels aux APIs/outils, exécution de commandes, génération de code ou requêtes SQL.
- **Boucle itérative** : Observation des résultats → ajustement du plan → nouvelle action.

## Schéma mental

*Percevoir* → *Planifier* → *Agir* → *Vérifier* (cycle autonome).

# Chaînes de tâches et mémoire

- **Chaînes de raisonnement** : Structuration des étapes intermédiaires (ex. CoT, ReAct).
- **Chaînes de tâches IT** : Surveillance → Diagnostic → Correctif → Rapport.
- **Mémoire à court terme** : Contexte de la session, variables temporaires.
- **Mémoire à long terme** : Historique, préférences, données indexées via embeddings.
- **Valeur en production** : Traçabilité, continuité, cohérence des décisions dans le temps.

# Risques et limites actuelles

- **Hallucinations en chaîne** : Une erreur initiale peut se propager et contaminer toutes les étapes suivantes.
- **Robustesse limitée** : Sensibilité aux cas hors distribution, ambiguïtés, ou contextes adverses.
- **Hacking / Prompt injection** : Détournement d'instructions, exécution non souhaitée de commandes.
- **Coûts et latence** : Boucles trop longues, appels multiples à des outils → latence et budget imprévisibles.
- **Enjeu IT** : Trouver l'équilibre entre autonomie et garde-fous.

# Nouvelle génération d'agents

- **Nature** : Modèles dont l'objectif même est d'agir comme agents, et non seulement de générer du texte.
- **Entraînement** :
  - Supervision sur des jeux de données de **planification multi-étapes**.
  - Apprentissage de l'**utilisation d'outils** (API, DB, code) à partir de traces réelles.
  - **Reinforcement Learning** sur des tâches de long horizon (récompense retardée).
- **Caractéristiques** :
  - Planification et usage des outils intégrés nativement.
  - Moins dépendants de prompts ad hoc.
  - Plus robustes face aux contextes complexes.
- **Vision** : Passer du LLM déguisé en agent à un **agent entraîné pour agir**.



# Agents de génération de code

- **Cursor** : IDE augmenté, génération et refactorisation de code, intégration avec les tests.
- **Replit Ghostwriter** : Génération + exécution dans l'environnement cloud.
- **Cas d'usage IT** :
  - Génération de scaffolding de projets.
  - Refactorisation et documentation automatique.
  - Automatisation DevOps (scripts de déploiement, correctifs).
- **Bonnes pratiques** : Valider via tests, limiter les permissions, relire les diffs systématiquement.

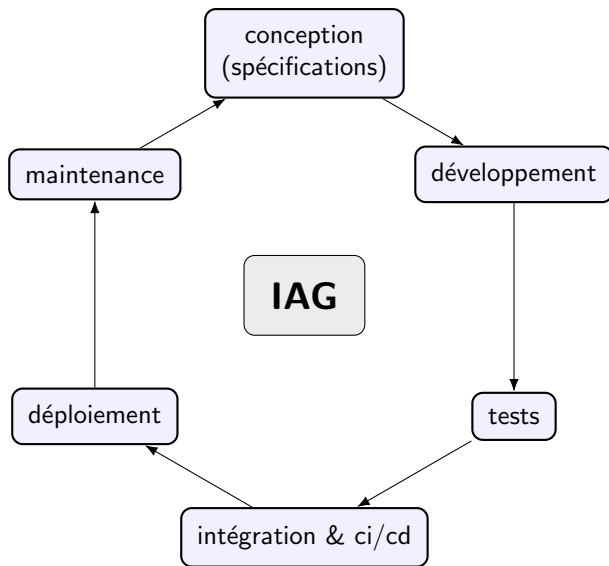
# Intégration de l'IA générative

# L'IAG dans le cycle de vie logiciel

L'intelligence artificielle générative ne se limite pas à l'autocomplétion de code : elle peut s'intégrer à chaque étape du cycle de vie logiciel pour assister les équipes.

- **Conception** : clarification des besoins, spécifications, ébauches d'architecture.
- **Développement** : génération et refactor de code, assistance aux patterns.
- **Tests** : génération de tests unitaires et fonctionnels, détection de cas limites.
- **Intégration & CI/CD** : review automatique de PR, génération de changelog et documentation.
- **Déploiement** : notes de version, assistance à l'infrastructure-as-code.
- **Maintenance** : classification des tickets, suggestions de correctifs, documentation à partir du code.

# IAG dans le cycle de vie logiciel



# Conception et spécifications

**Objectif** : réduire l'ambiguïté et accélérer le passage du besoin métier à la solution technique.

- Transformer une *user story* en endpoints d'API (routes, verbes, schémas JSON).
- Générer une première ébauche d'architecture (services, flux, dépendances).
- Produire un glossaire ou des critères d'acceptation testables.

**Exemple de prompt** : *À partir de cette user story, propose les endpoints REST, les codes d'erreur et un schéma JSON pour POST /orders.*

**Objectif** : accélérer l'implémentation tout en améliorant la lisibilité et la maintenabilité.

- Génération de fonctions à partir d'une description en langage naturel.
- Refactor : extraction de sous-fonctions, renommage de variables, simplification de structures.
- Suggestions de design patterns adaptés au contexte.

**Exemple** : Avant : fonction monolithique de 80 lignes. Après : découpage en `validate()`, `transform()`, `persist()` avec docstrings et exceptions typées.

**Objectif** : améliorer la couverture et détecter les cas limites dès la phase de développement.

- Génération de tests unitaires directement à partir du code.
- Proposition de cas extrêmes : valeurs limites, erreurs réseau, formats inattendus.
- Création de scénarios fonctionnels dérivés des user stories.

**Exemple de prompt** : *Voici la fonction `price(order)`. Génère 8 tests unitaires `pytest` couvrant remise, TVA, devise inconnue, quantité négative, seuils, valeur nulle.*

# Intégration et CI/CD

**Objectif** : enrichir les pipelines d'intégration continue avec des capacités d'analyse et de génération.

- Review automatique de pull requests : détection de failles de sécurité, de duplications, de style incohérent.
- Génération automatique de tests complémentaires pour valider les commits.
- Production de documentation et de changelogs à partir des messages de commits.

**Exemple** : Un workflow GitHub Actions envoie le *diff* à un LLM, qui commente directement la PR avec ses suggestions.



**Objectif** : automatiser et simplifier les tâches de mise en production.

- Assistance à la rédaction de scripts *infrastructure-as-code* (Terraform, Ansible).
- Génération automatique des notes de version lisibles à partir des commits.
- Support aux équipes DevOps pour documenter et communiquer les changements.

**Exemple** : À partir de commits techniques, l'IAG rédige un changelog clair pour les utilisateurs finaux.

**Objectif** : accélérer la résolution de problèmes et réduire la dette technique.

- Classification automatique des tickets d'incidents par priorité et catégorie.
- Proposition de correctifs initiaux pour certains bugs.
- Génération de documentation à partir de code existant ou de logs.

**Exemple** : Un log d'erreur est soumis à l'IAG, qui propose un diagnostic et une piste de correction.

# Limites et précautions

## Points de vigilance lors de l'intégration de l'IAG :

- **Qualité variable** : sorties parfois incomplètes ou erronées, besoin de supervision humaine.
- **Sécurité et confidentialité** : attention à l'envoi de code ou de données sensibles.
- **Coût et performance** : appels API facturés au token, temps de latence selon les modèles.
- **Dépendance technologique** : choix entre solutions cloud, open source locales ou hybrides.

**Message clé** : l'IAG est un assistant puissant, mais elle ne remplace pas le rôle critique du développeur et de l'architecte.

# Intelligence artificielle et organisation

# Pourquoi l'IA crée-t-elle de la valeur ?

- Optimisation (rationalisation des navettes, matching entre offre et demande, etc.)
- Automatisation (moteur de recommandation, moteur de recherche, etc.)
- Aide à la décision (ciblage clients, assortiment, etc.)
- Innovation (nouveaux produits, nouveaux services, etc.)

# Maturité des entreprises en IA

## Maturité en intelligence artificielle

1

### Aide à la décision

Il s'agit typiquement des anciennes pratiques en analyse de données, recherche opérationnelle et statistiques, remises au goût du jour.

2

### Automatisation optimisation

Cela concerne l'automatisation de certains processus ou tâches traditionnellement effectués d'une manière manuelle ou avec des techniques peu avancées (ciblage, etc.)

3

### Innovation

Création de nouveaux produits ou services en utilisant la donnée. Il s'agit du degré le plus avancé dans la transformation, induisant souvent un changement dans le business model.

# Cas d'usage pertinents du ML

- Validité du besoin de recourir à des algorithmes.
- Disponibilité des données en quantité et qualité suffisantes.
- Coût faible d'une mauvaise prédiction (risque financier, humain, etc. limité si l'algorithme commet une erreur de prédiction).
- Inefficacité avérée d'approches plus simples (moteur de règles, etc.) pour traiter la problématique.

# Automatisation vs Augmentation

Certains cas d'usage tombent naturellement dans l'une ou l'autre catégorie. La recommandation de produit par exemple sur une plateforme nécessite de l'automatisation car le nombre de décisions à prendre par unité de temps est très élevé. Inversement, les décisions stratégiques orientées données sont nécessairement validées et exécutées (ou non) par des humains.

D'autres cas, en revanche, peuvent tomber dans les deux catégories à la fois. C'est par exemple le cas de l'extraction d'informations à partir de documents. En fonction des performances des algorithmes, du coût des erreurs de prédiction, l'automatisation ou l'augmentation peuvent être privilégiés.



# Avantages de l'augmentation

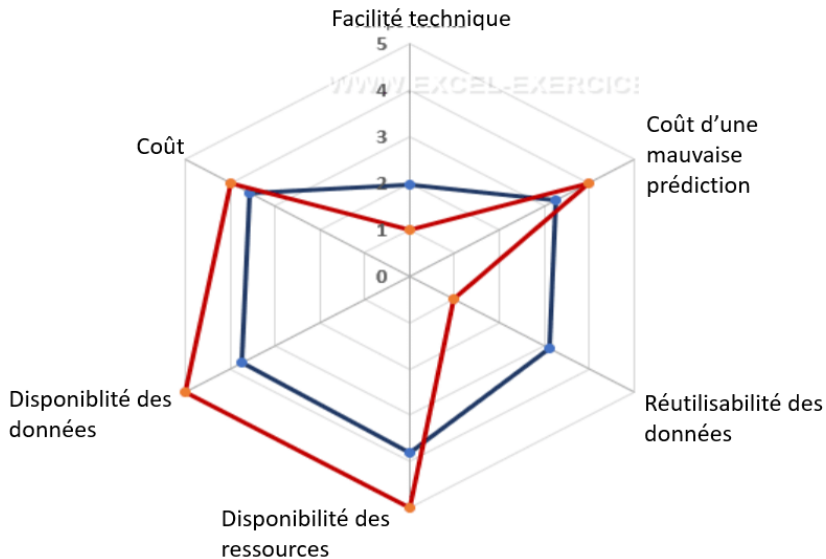
- Les cas problématiques pour l'algorithme peuvent être traités par les humains, induisant un gain d'efficacité dans l'ensemble.
- Compréhension plus accrue des décisions de l'algorithme, facilitant les améliorations qui peuvent en être apportées.
- Acquisition de potentielles connaissances métiers supplémentaires à même de tirer l'organisation tout entière vers le haut (voire inspirer des décisions stratégiques).
- Fédération des équipes plus importante.

# Sélection des cas d'usage

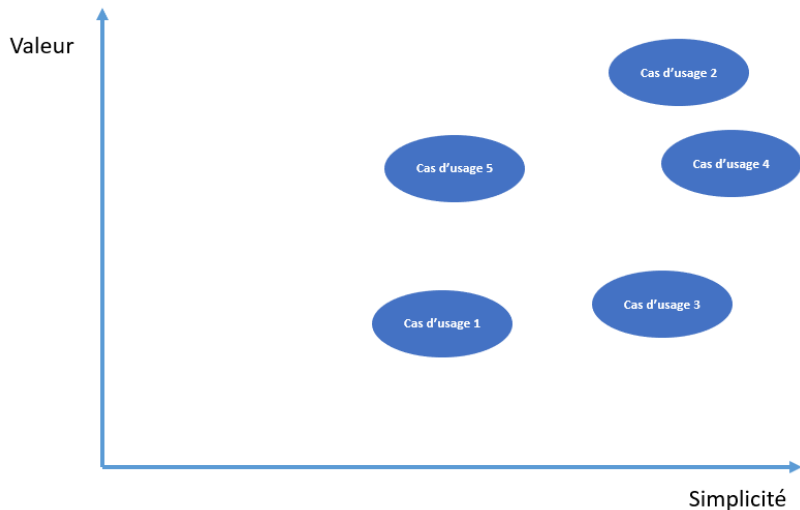
Plusieurs aspects peuvent être considérés, en fonction de l'organisation, pour sélectionner et prioriser les cas d'usage :

- Valeur générée par le projet.
- Coût du projet.
- Difficulté technique de la solution.
- Coût d'une mauvaise prédiction.
- Disponibilité des données.
- Disponibilité des ressources (compétences, infrastructures, etc.).
- Externalités positives du projets (les données peuvent-elles être réutilisées pour un autre projet, le projet fédère-t-il les équipes, etc.).

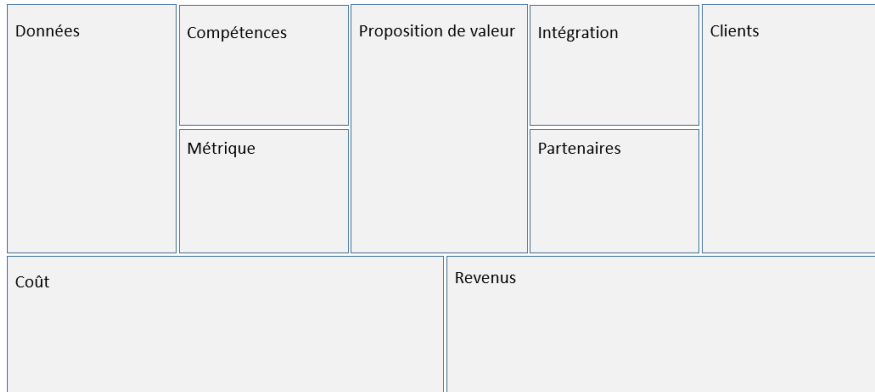
# Diagramme en toile d'araignée



# Diagramme simplifié



# AI canvas



# Budgétisation d'un projet IA

Les différents coûts peuvent se situer dans différentes catégories, mais la part la plus importante et la plus difficile à chiffrer est souvent celle qui est liée au personnel.

- Salaires chargés des personnels.
- Sous-traitance.
- Infrastructure (installation de capteurs, ordinateurs, services Cloud, etc.).
- Coûts liés aux données (achat de bases de données, labellisation des données, etc.).
- Licences logicielles.
- Etc.

# Proposition de valeur d'un projet IA

Traditionnellement, la proposition de valeur s'articule essentiellement autour du ROI du projet. Mais celle-ci peut aller au-delà pour tenter de lister toutes les externalités positives liées au projet qui peuvent être très importantes.

Par ailleurs, il faut distinguer les projets dont le ROI est relativement facile à calculer (souvent immédiatement tributaire des performances prédictives du modèle), comme le Credit Scoring par exemple, des projets dont le ROI est difficile à chiffrer (moteur de recherche).

Exemples de facteurs à prendre en compte dans ce cas-ci :

- Estimation du temps passé par des agents humains les tâches destinées à être automatisées.
- Pénibilité des tâches destinées à être automatisées.
- Qualité de l'expérience client.
- Etc.

# Organisation : principaux métiers de l'IA

Les métiers de l'IA sont en constante évolution et ont subi une évolution assez importantes ces dernières années. On peut distinguer cependant les métiers suivants :

- **Data scientist** : il est au cœur du projet IA. Son principal rôle consiste à traduire le besoin métier en une problématique technique pour élaborer un modèle de ML ou autre, qu'il doit tester et valider avec les métiers. Il est donc normalement amené à travailler étroitement avec les métiers.
- **Machine Learning Engineer** : son rôle consiste à adapter le code développé par le data scientist pour le déployer, souvent sur une infrastructure Cloud (AWS, GCP, Azure, etc.). Il doit ensuite veiller au bon fonctionnement de l'application.
- **Data Engineer** : il est responsable de l'infrastructure liée au projet (pipelines de données, etc.).
- **Data analyst** : son rôle est d'analyser les données pour répondre à des questions métier.
- **Data architecte** : il conçoit l'architecture de la solution qui intègre éventuellement de l'IA (bases de données, back-end, front-end, services Cloud, etc.), notamment quand celle-ci est complexe.



# Organisation : schéma

Les organisations autour d'un projet IA sont également en constante évolution et dépendent de plusieurs facteurs : legacy de l'entreprise, sa maturité IA, nature du projet IA (importance de la composante métier, de l'infrastructure, etc.), etc.

Mais il on peut distinguer deux principales manières de s'organiser :

- **Organisation centralisée** : cela consiste en une équipe technique (data scientists, ML Engineer, etc.) qui travaille d'une manière centralisée et transverse aux autres équipes/départements de l'entreprise, souvent dans un Data Lab . La coordination entre cette équipe et les équipes métiers est assurée par un chef de projet.
- **Organisation décentralisée** : les rôles techniques (data scientists, ML engineer, etc.) sont directement intégrés dans chaque équipe métier et travaillent directement sous la responsabilité du responsable métier (responsable marketing, finances, etc.) par l'intermédiaire ou non d'un chef de projet.

# Enjeux éthiques, sécurité et conformité de l'IAG

# Pourquoi parler d'éthique et de sécurité ?

L'intelligence artificielle générative ne pose pas seulement des défis techniques : elle soulève des enjeux critiques pour l'entreprise.

- **Sécurité** : risques de fuite de secrets ou d'attaques via les prompts.
- **Confidentialité** : conformité au RGPD, respect des données sensibles.
- **Propriété intellectuelle** : **propriété des données d'entraînement et du contenu généré par l'IA.**
- **Éthique** : biais, équité, responsabilité dans l'usage.

**Message clé** : l'adoption de l'IA doit être accompagnée d'une gouvernance adaptée.

## Exemples de menaces :

- **Fuite de secrets** : code ou identifiants envoyés à une API externe.
- **Prompt injection** : manipulation du modèle par un utilisateur malveillant.
- **Dépendance externe** : indisponibilité ou vulnérabilité du fournisseur de LLM.

**Bonne pratique** : utiliser des proxys, filtrer les prompts et éviter d'envoyer des données sensibles.

**Problématique centrale** : l'IAG traite souvent des données sensibles ou personnelles.

- **RGPD** : droit à l'oubli, consentement, minimisation des données.
- **Localisation des données** : datacenters UE vs hors UE.
- **Solutions** :
  - anonymisation avant envoi,
  - hébergement de modèles open source en interne,
  - passage par un proxy d'entreprise.

## Un enjeu majeur souvent sous-estimé dans l'usage de l'IAG.

- **Origine des données d'entraînement** : risque que le modèle régénère des contenus protégés (texte, code, images).
- **Droits sur les contenus générés** :
  - Dans l'UE, pas de droit d'auteur automatique sur une production 100% IA.
  - L'entreprise doit définir une politique claire (contrats, licences internes).
- **Licences et open source** : vérifier la licence des modèles (Apache 2.0, MIT, restrictions) et des datasets utilisés.
- **Bonnes pratiques** :
  - tracer les sources de données et les outputs,
  - sensibiliser les équipes aux risques de réutilisation non conforme,
  - éviter d'intégrer sans vérification un contenu généré dans des livrables contractuels.

# Biais et équité

**Constat** : les modèles reflètent les biais de leurs données d'entraînement.

- Risque de stéréotypes et de discriminations dans les réponses.
- Impact fort dans les domaines sensibles (recrutement, médical, juridique).
- Besoin d'auditer les modèles dans leur contexte métier.

**Message clé** : l'évaluation humaine reste indispensable pour corriger les biais.

## Questions clés :

- Qui est responsable si une IA génère du code erroné ou trompeur ?
- Comment tracer et auditer les décisions de l'IA ?

## Bonnes pratiques :

- Supervision humaine obligatoire pour les tâches critiques.
- Mise en place d'une charte d'usage de l'IAG.
- Définition de rôles et responsabilités clairs.



# Durabilité et coûts cachés

## Impact énergétique :

- Fine-tuning complet = forte consommation GPU et empreinte carbone.
- LoRA/PEFT = alternatives plus sobres, adaptées aux entreprises.

## Coûts cachés :

- Facturation à l'usage (tokens, appels API).
- Dépendance à un fournisseur unique.

**Message clé** : arbitrer entre innovation, budget et responsabilité environnementale.

# Conclusion et bonnes pratiques

## Synthèse :

- Les enjeux de sécurité, confidentialité et éthique sont incontournables.
- La gouvernance doit accompagner toute intégration de l'IA.
- Commencer petit, superviser, documenter et auditer.

**Bonne pratique** : intégrer les aspects éthiques dès la conception des projets IA.

# Merci de votre attention

[redha.moulla@axia-conseil.com](mailto:redha.moulla@axia-conseil.com)