Synthèse : Introduction aux LLM et à l'IA conversationnelle

---

1. Introduction

L'intelligence artificielle (IA) a connu une forte croissance, notamment grâce aux **Grands Modèles de Langage (LLM)**. Ces modèles permettent à des programmes de comprendre et de générer du texte de manière naturelle, ouvrant la voie à des **chatbots intelligents** comme ChatGPT ou Claude. Ce document présente les bases nécessaires pour comprendre et utiliser ces outils.

---

2. Qu'est-ce qu'un LLM ?

Un **LLM** (*Large Language Model*, ou **grand modèle de langage**) est un **programme d'intelligence artificielle** conçu pour **comprendre et produire du texte**.
Il fonctionne un peu comme un **assistant virtuel** capable de rédiger, expliquer, traduire ou répondre à des questions de manière naturelle.

Pour apprendre, un LLM est **entraîné sur des milliards de textes** provenant de livres, d'articles ou de sites web. Son objectif pendant cet apprentissage est simple : **deviner le mot suivant** dans une phrase. En répétant cet exercice des milliards de fois, il apprend progressivement les **règles du langage**, les **liens entre les mots** et la **logique du discours**.

Une fois entraîné, le modèle peut être intégré dans des **applications conversationnelles** comme **ChatGPT** ou **Claude**, qui servent d'interface entre le modèle et l'utilisateur.
Ces applications ajoutent des fonctions pratiques : historique des échanges, gestion de fichiers, outils de recherche, etc.

Les LLM sont aujourd'hui capables de **rédiger un texte**, **résumer un document**, **traduire**, **générer du code**, ou **expliquer un concept**. Cependant, ils ne "pensent" pas réellement : ils ne font qu'**estimer les mots les plus probables** selon le contexte.

👉 En résumé, un **LLM** est une IA qui **apprend à écrire comme un humain** en analysant d'énormes quantités de texte et en **générant, mot après mot**, des réponses cohérentes et pertinentes.

---

3. Les principaux LLM en 2025

Modèle	Entreprise / Organisation	Pays
GPT-4 / GPT-5	OpenAI	États-Unis
Gemini	Google DeepMind	États-Unis / Royaume-Uni
Claude	Anthropic	États-Unis
Grok	xAI (Elon Musk)	États-Unis
LLaMA	Meta (Facebook)	États-Unis
Mistral	Mistral AI	France
DeepSeek	DeepSeek AI	Chine
Qwen	Alibaba Cloud	Chine
GLM	Zhipu AI	Chine

À retenir :

- Les États-Unis dominent le domaine, mais la Chine et la France développent aussi des modèles puissants.
- Certains modèles sont **open source** (libres d'utilisation et de modification), comme LLaMA ou Mistral.

---

4. Qu'est-ce qu'un Chatbot ?

Un **chatbot** est un programme capable de **dialoguer avec un humain** en langage naturel. Aujourd'hui, la majorité des chatbots utilisent un LLM pour comprendre les questions et produire des réponses pertinentes.

Pourquoi passer par un chatbot ?

- Interface simple et intuitive (comme une messagerie)
- Conservation du contexte des échanges
- Format conversationnel naturel (questions/réponses)
- Facile à intégrer dans des sites, applis ou services clients

Exemples : ChatGPT, Claude.ai, Google Bard, bots Discord, Copilot, etc.

---

5. Qu’est-ce qu’un Prompt ?

Le **prompt** est la **consigne** que l’on donne à l’IA. C’est une instruction en langage naturel qui indique ce que l’on attend du modèle.

Exemples :

- Simple : "Explique la différence entre HTTP et HTTPS"
- Avancé : "Tu es enseignant en BTS SIO. Rédige une fiche de cours sur HTTP vs HTTPS."

➡️ Un bon prompt est **clair, précis et contextualisé**. Plus il est détaillé, plus la réponse est pertinente.

---

6. Tokens, Contexte et Mémoire

- **Token** : unité de texte (mot, syllabe ou symbole). Les modèles comptent et facturent souvent à la token.
  → 1000 tokens ≈ 750 mots environ.
- **Fenêtre de contexte** : quantité maximale de texte que le modèle peut analyser en une fois (ex : GPT-4 ≈ 128k tokens).
- **Mémoire conversationnelle** : le chatbot conserve l’historique du dialogue dans la limite de cette fenêtre. Au-delà, il “oublie” les premiers messages.

💡 **À retenir :** Le LLM ne se souvient pas d’une session à l’autre. La mémoire est gérée par le logiciel du chatbot, pas par le modèle lui-même.

---

7. Les différents moyens d’utiliser un LLM

1. **Via un chatbot en ligne** (ex : ChatGPT, Claude, Gemini...)
   → Interface simple pour dialoguer sans coder.
2. **Via une API**
   → Pour les développeurs souhaitant intégrer l’IA dans une application.
   Exemple : API OpenAI.
3. **En local / auto-hébergé**
   → Avec des modèles open source (LLaMA, Mistral, etc.) sur son propre serveur.
4. **Intégré dans des outils existants**
   → Word, Excel, VS Code, Copilot, etc.

---

8. Le Prompt Engineering

Le **prompt engineering** est l'art de **formuler les bons prompts** pour obtenir des réponses précises et cohérentes. C'est une compétence essentielle pour interagir efficacement avec les LLM.

---

9. Conclusion

Les **LLM** comme GPT, Gemini ou Mistral représentent une avancée majeure en intelligence artificielle. Ils sont capables de comprendre et générer du langage humain avec une grande précision. Pour les étudiants de **BTS SIO**, comprendre ces notions permet :

- d'utiliser efficacement les outils d'IA dans les projets,
- d'intégrer des assistants conversationnels via API,
- et de développer une culture numérique moderne.

💡 **Clé de réussite :** maîtriser les bases des LLM et savoir rédiger des prompts clairs et structurés.

🧠 1. « Un LLM ne comprend pas le langage naturel (sous forme textuelle). Est-ce vrai ? »

➡️ **Pas tout à fait.**

Un **LLM (Large Language Model)** ne "comprend" pas le texte **comme un humain**, mais il **traite** le langage naturel sous forme de nombres.
Voici comment cela fonctionne :

1. L'utilisateur écrit une phrase en langage naturel (ex. : *« Explique-moi ce qu'est un chatbot »*).
2. Cette phrase est **découpée en tokens** (morceaux de mots).
3. Chaque token est **converti en nombre** grâce à un système de codage appelé *tokenizer*.
4. Le modèle traite ces nombres via des calculs statistiques (réseau de neurones) pour **prédire le mot suivant**.

🧩 Donc le LLM **ne comprend pas le sens du texte** au sens humain du terme, mais il **reconnaît des structures linguistiques** et **génère une réponse cohérente** grâce à ce qu'il a appris sur d'immenses quantités de textes.

🟢 En résumé : le LLM ne "comprend" pas, mais il **simule la compréhension** en manipulant le langage naturel sous forme numérique.

💬 2. « Le prompt est exprimé en langage naturel par l'utilisateur dans le chatbot. Est-ce le chatbot qui traduit le prompt pour qu'il soit compréhensible par le LLM ? »

➡️ **Oui, exactement.**

Le **chatbot** joue le rôle d'**intermédiaire** entre toi (l'utilisateur humain) et le **LLM** (le modèle d'IA).
Voici ce qu'il fait :

1. Tu saisis ton **prompt** dans la fenêtre du chatbot.
2. Le chatbot **formate ta requête** : il ajoute parfois des informations supplémentaires (par ex. un *system prompt* qui définit le rôle de l'IA).
3. Puis il **envoie le texte au LLM**, qui le transforme en tokens et génère une réponse.

💡 Le chatbot ne traduit pas le texte en "langage humain → langage machine" comme une traduction complète.
Mais il **prépare et structure** la requête pour qu'elle soit correctement comprise par le modèle.

🟢 En résumé : le chatbot agit comme un **traducteur et un médiateur** entre ton langage naturel et le format attendu par le LLM.

💻 3. « Lors de la réponse du LLM au chatbot, qui transforme la réponse en langage naturel (sous forme textuelle) ? »

➡️ **C'est le chatbot, encore une fois.**

Voici le chemin du retour :

1. Le **LLM génère** une suite de tokens (nombres correspondant à des morceaux de texte).
2. Ces tokens sont **convertis en mots** par le **tokenizer inverse** du modèle.
3. Le **chatbot récupère le texte** généré, puis l'affiche joliment dans l'interface (fenêtre de discussion).

Parfois, le chatbot applique aussi :

- des **filtres de sécurité** (pour éviter du contenu inapproprié),
- ou un **formatage** (gras, listes, tableaux, etc.) avant de montrer le message à l'utilisateur.

🟢 En résumé : le **LLM génère la réponse**, mais c'est le **chatbot** qui la met en forme et la **présente en langage naturel**.

🎯 Synthèse visuelle

Étape	Acteur principal	Rôle
🧍 **L'utilisateur écrit un prompt**	**Humain**	Utilise le langage naturel
🔄 **Le chatbot prépare la requête**	**Chatbot**	Formate le texte pour le LLM
⚙️ **Le LLM traite la requête**	**Modèle d'IA**	Analyse les tokens et génère une réponse
💬 **Le chatbot affiche la réponse**	**Chatbot**	Convertit le résultat en texte lisible

👍 Voici un **exemple concret et simple** du chemin parcouru par un prompt, depuis l'utilisateur jusqu'au traitement par le LLM :

🎯 Exemple de départ

Prompt de l'utilisateur (dans le chatbot) :

« Explique-moi simplement ce qu'est un pare-feu en informatique. »

⌨️ Étape 1 — L'utilisateur envoie le message

Tu écris ton texte dans l'interface du chatbot (par exemple ChatGPT, Claude ou Gemini).
➡️ À ce stade, ton message est du **langage naturel** (du texte brut compréhensible par un humain).

Exemple de prompt :
« Explique-moi simplement ce qu'est un pare-feu en informatique. »

- L'utilisateur tape sa question dans l'interface du **chatbot** (ChatGPT, Claude, Gemini, etc.).
- Ce texte est en **langage naturel** (une phrase "humaine").
- Le chatbot **n'interprète pas encore** le sens du texte, il se contente de le **récupérer** tel quel pour le préparer.

💡 À ce stade, il n'y a encore **aucun traitement par le LLM**. Le chatbot agit comme une "boîte aux lettres".

⚙️ Étape 2 — Le chatbot prépare la requête

Avant d'envoyer ta phrase au modèle, le **chatbot** ajoute des informations invisibles pour toi :

- Un **system prompt** (ex. : *« Tu es un assistant pédagogique spécialisé en informatique. »*)
- Ton **message utilisateur**
- L'**historique de la conversation récente** (pour que le modèle garde le contexte)

💡 Le tout est mis dans une structure standard, souvent un **fichier JSON**, du type :

```
{
  "messages": [
    {"role": "system", "content": "Tu es un assistant pédagogique."},
    {"role": "user", "content": "Explique-moi simplement ce qu'est un pare-feu en informatique."}
  ]
}
```

Le **chatbot** a pour rôle d'**emballer** ton message dans un format standard que le LLM comprend.
Il construit une requête contenant :

- un **system prompt** : décrit le rôle du modèle, son ton ou ses limites (ex : *« Tu es un assistant pédagogique qui répond de manière claire et bienveillante. »*)
- un **user prompt** : ton message réel.
- éventuellement l'**historique de la conversation** (les messages précédents utiles pour garder le contexte).

👉 Comment le chatbot sait quel *system prompt* ajouter ?

- Il est **défini par le concepteur du chatbot** (par exemple OpenAI, Anthropic, etc.).
- Chaque plateforme a son propre "rôle par défaut".
  - ChatGPT : *assistant généraliste, poli, précis, utile.*
  - Claude : *assistant réfléchi, prudent, éthique.*
  - Copilot : *assistant de programmation.*
- Le chatbot peut aussi **adapter le system prompt dynamiquement** (par exemple : *"mode enseignant", "mode créatif", "mode concis"*).

Exemple du format envoyé au LLM :

```
{
  "messages": [
    {"role": "system", "content": "Tu es un assistant pédagogique spécialisé en informatique."},
    {"role": "user", "content": "Explique-moi simplement ce qu'est un pare-feu en informatique."}
  ]
}
```

💡 Ce JSON est ce que reçoit **l'API du LLM**.
Le modèle ne lit pas directement du "texte libre" : il lit des **structures** avec des rôles bien définis.

🧩 Étape 3 — Le LLM transforme le texte en tokens

Le texte est **tokenisé** :

- “Explique-moi” → devient quelques nombres
- “pare-feu” → un ou deux nombres selon le découpage
- “informatique” → un autre nombre

Chaque mot ou morceau de mot est converti en **nombre** à l’aide d’un *tokenizer*.

Le modèle ne voit donc **que des nombres**, pas des mots.

Le LLM ne traite **pas des phrases**, mais des **nombres**.
Il doit donc **convertir chaque mot en nombre**, grâce à une étape appelée **tokenisation**.

🧠 La tokenisation

- Elle repose sur un **référentiel appelé “vocabulaire du modèle”**, créé lors de l’entraînement.
- Ce vocabulaire contient **tous les morceaux de mots connus** du modèle, chacun ayant un identifiant unique.

Exemple concret :

Imaginons que le vocabulaire simplifié soit :

Morceau de texte	Token ID	Commentaire
“Un”	51	Mot complet
“pare”	5021	Première partie du mot composé
“-”	7001	Tiret
“feu”	1947	Deuxième partie du mot composé
“est”	102	Verbe
“un”	52	Article (forme minuscule distincte)
“outil”	2501	Nom commun
“informatique”	8274	Nom commun
“.”	9001	Ponctuation

👉 La phrase : “Un pare-feu est un outil informatique.”
peut devenir :
[51, 5021, 7001, 1947, 102, 52, 2501, 8274, 9001]

Le modèle ne voit que cette suite de nombres.
Il apprend grâce à des **relations statistiques** entre ces tokens à prédire la suite logique.

💡 Chaque LLM a son propre **tokenizer** et son propre **vocabulaire** (GPT, Claude, Mistral n’ont pas les mêmes).

⌨️ Étape 4 — Le LLM traite les tokens

C’est ICI que la magie opère !

Le **réseau de neurones du modèle** (GPT, Claude, Mistral, etc.) calcule, pour chaque position, **le mot le plus probable suivant**.
Il s’appuie sur tout ce qu’il a appris pendant son entraînement (textes, livres, code, etc.).

👉 Exemple : il “devine” que le mot suivant après “pare-feu” est probablement “protège” ou “réseau”.

Le modèle génère ainsi la réponse **token par token**.

💡 Le LLM **ne produit pas encore du texte** ici, mais une **séquence de tokens numériques** représentant la réponse.

Ainsi, la sortie peut ressembler à :
[88, 931, 105, 77, 51, 1947]

🔁 Étape 5 — Le LLM renvoie la réponse au chatbot

Le résultat est une suite de tokens (nombres).

Une fois la suite de **tokens** générée, le **LLM la reconvertit en texte** grâce au **tokenizer inverse**.

Exemple :

[88, 931, 105, 77, 51, 1947]
devient :
“Un pare-feu protège le réseau.”

Le LLM renvoie alors au chatbot du texte formaté **JSON** contenant ce texte :

```
{
  "id": "chatcmpl-12345",
  "choices": [
    {
      "message": {
        "role": "assistant",
        "content": "Un pare-feu protège le réseau."
      }
    }
  ]
}
```

⚠ Important :

- Oui, le LLM **renvoie du texte clair** dans ce format JSON.
- Non, le LLM **ne fait pas de mise en forme avancée** (gras, couleur, images…).
  Il renvoie simplement du texte brut (parfois avec du Markdown léger, ex : **mot** ou - liste).

💬 Étape 6 — Le chatbot affiche la réponse

Le **chatbot reçoit le JSON** du LLM, extrait la partie "content" et l'affiche joliment à l'utilisateur.

C'est lui qui gère :

- la **mise en page** (police, couleurs, sauts de ligne, gras, listes…)
- les **émojis**, les **bulles de dialogue**, ou les **blocs de code**
- éventuellement un **retard d'affichage progressif** (comme une écriture "en direct")

💡 En d'autres termes :

- Le **LLM génère du contenu** (du texte brut)
- Le **chatbot gère la présentation** (le "design" et l'expérience utilisateur)

🟢 **En résumé :**

- Le **chatbot** gère la communication (interface, formatage, contexte).
- Le **LLM** fait le travail d'analyse et de génération.
- L'utilisateur voit simplement une réponse en langage naturel, mais en coulisse tout passe par des nombres et des calculs.

Synthèse

Étape	Qui agit ?	Format / contenu	Rôle principal
1	Utilisateur	Langage naturel	Pose une question
2	Chatbot	JSON (system + user prompt)	Prépare la requête
3	LLM	Tokens (nombres)	Convertit et analyse
4	LLM	Tokens (nombres générés)	Prédit la réponse
5	LLM	JSON avec texte brut	Retourne la réponse textuelle
6	Chatbot	Texte + mise en page	Affiche joliment la réponse

Comment le chatBot maintient une **conversation fluide** avec un **LLM sans mémoire** (*stateless*), grâce notamment à la **fenêtre de contexte** (*context window*) ?

🧩 1. Le LLM est “sans état” : il oublie tout entre deux requêtes (prompts)

Un **LLM** (comme GPT, Mistral, Claude, etc.) est dit **stateless**. Cela signifie qu’il **n’a aucune mémoire permanente** entre deux appels.

➡️ Concrètement :

- Il **ne sait pas** ce que tu as dit dans ton message précédent.
- À chaque requête, il reçoit **tout ce qu’il doit savoir dans le prompt**, puis il génère une réponse **comme si c’était la première fois**.

💬 Exemple :

Si tu envoies “Rappelle-moi ce que tu viens de dire”,
le LLM ne peut pas le faire **s’il ne reçoit pas à nouveau la phrase précédente** dans le contexte.

⚙️ 2. Le rôle du chatbot : donner au LLM l’illusion de la mémoire

C’est là que **le chatbot entre en scène** 🧠

Le **chatbot** (ex : ChatGPT, Claude.ai, Copilot, Gemini, etc.) est une **couche logicielle au-dessus du LLM**. Il agit comme un **gestionnaire de conversation**.

Son rôle principal :

Maintenir l’historique complet du dialogue et le reformater et le renvoyer au LLM à chaque nouvel échange.

💡 Ainsi, à chaque fois que tu écris un nouveau message :

1. Le chatbot récupère **toute la conversation précédente** (toi + l’IA).
2. Il la reformate en JSON (messages, rôles, contenu, etc.).
3. Il renvoie **l’ensemble** au LLM comme *prompt complet*.

Le LLM, voyant tout l’historique dans sa **fenêtre de contexte**, a l’impression de “se souvenir” de la discussion.
Mais en réalité, c’est le **chatbot qui lui redonne la mémoire** à chaque fois.

🧠 3. La fenêtre de contexte : la “mémoire à court terme” du LLM

La **fenêtre contextuelle** (*context window*) correspond à la **quantité maximale de texte**
que le LLM peut lire et utiliser **en une seule fois**.

Elle est mesurée en **tokens**, pas en mots :

- GPT-4 Turbo → environ **128 000 tokens** (≈ 90 000 mots)
- Claude 3.5 → **200 000 tokens**
- Mistral 7B → **32 000 tokens**

La **fenêtre de contexte** (ou *context window*) est **la mémoire de travail temporaire du modèle**.

➡️ C’est la **zone interne du LLM** dans laquelle il **charge les tokens** avant de produire une réponse.
Elle contient tout ce que le modèle “voit” et peut utiliser **pour raisonner à ce moment-là**.

Autrement dit :

Quand on dit qu’un texte est “dans la fenêtre de contexte”, cela veut dire que ce texte (sous forme de tokens) **est actuellement accessible au modèle pour générer sa réponse.**

⚠️ Si la conversation devient trop longue, et que l’historique dépasse la taille de la **fenêtre de contexte**, le chatbot doit alors :

- **supprimer les anciens messages**, ou
- **les résumer** pour libérer de la place.

🧩 4. Exemple concret

Imaginons une conversation :

Utilisateur : Bonjour
LLM : Bonjour ! Comment allez-vous ?
Utilisateur : Peux-tu me rappeler ce qu'est un pare-feu ?
LLM : Un pare-feu est...
Utilisateur : Et peux-tu m'en donner un exemple concret ?

Ce que le chatbot envoie réellement au LLM :

```
[
  {"role": "system", "content": "Tu es un assistant pédagogique."},
  {"role": "user", "content": "Bonjour"},
  {"role": "assistant", "content": "Bonjour ! Comment allez-vous ?"},
  {"role": "user", "content": "Peux-tu me rappeler ce qu'est un pare-feu ?"},
  {"role": "assistant", "content": "Un pare-feu est..."},
  {"role": "user", "content": "Et peux-tu m'en donner un exemple concret ?"}
]
```

➡️ Le LLM voit **tout** cet historique à chaque requête.
Il peut donc répondre **en tenant compte du contexte complet**,
même s'il ne "s'en souvient" pas entre les appels.

⚙️ 5. Ce que le chatbot gère pour le LLM

Fonction	Rôle du chatbot
🧠 Mémoire courte (conversation)	Maintient l'historique et le renvoie au LLM à chaque fois.
🧩 Structure	Formate les échanges (system, user, assistant).
✂️ Compression / Résumé	Réduit le contexte quand il devient trop long.
⚖️ Cohérence	Garde un ton, un rôle, un style cohérent.
🛡️ Filtrage / Sécurité	Supprime ou reformule des contenus sensibles avant envoi.

---

7. En résumé

Élément	Rôle
LLM (modèle)	Génère des réponses à partir du contexte fourni (mais sans mémoire entre deux appels).
Fenêtre de contexte	Zone de lecture temporaire : tout ce que le modèle peut “voir” et utiliser pour répondre.
Chatbot	Gère l’historique, reformate les échanges, et simule la mémoire en renvoyant tout le contexte à chaque fois.

🟢 **Conclusion :**

Le **LLM** ne garde aucune mémoire : il raisonne uniquement sur le **contexte qu’on lui donne**.
C’est le **chatbot** qui maintient la conversation en **conservant et réinjectant** l’historique des messages dans la **fenêtre de contexte**, donnant ainsi **l’illusion d’une discussion continue et cohérente.**