Large Language Model

Cours « Découverte de l'intelligence artificielle »

Nicolas Delestre et ChatGPT

INSA ROUEN NORMANDIE

Plan

1. Introduction à l'IA Générative
2. Fonctionnement des LLM
3. Utilisation d'un LLM par un programme
4. Utilisation d'un LLM : l'art du bon prompt
5. Enjeux et Perspectives

INSA ROUEN NORMANDIE

Définition et Concepts Clés

Qu'est-ce que l'IA Générative ?

L'IA générative est une IA capable de produire du contenu (texte, image, son, code) à partir de données d'entraînement

Exemples d'applications

- Chatbots et assistants virtuels
- Génération automatique de code
- Création de contenu multimédia

INSA ROUEN NORMANDIE

Historique et Évolution

Des premiers modèles à l'essor des LLM

- **Chaînes de Markov** (années 1950) : modélisation de la probabilité d'apparition d'un mot en fonction du mot précédent
- **Réseaux de Neurones Récurrents (RNN)** (années 1990) : introduction de la mémoire courte, permettant de traiter des séquences de texte
- **Long Short-Term Memory (LSTM)** (1997) : amélioration des RNN avec des portes de mémoire pour capturer des dépendances à long terme
- **Transformers** (2017) : introduction du mécanisme d'attention permettant de traiter des phrases entières en parallèle
- **GPT-3, GPT-4** (2020-2023) : exploitation des *Transformers* à grande échelle pour une génération de texte fluide et contextuelle

Architecture des Transformers 1 / 4

Principe de fonctionnement

Les Transformers sont une architecture de réseau de neurones qui permettent de traiter des séquences de texte[a] grâce à trois concepts clés :

- **la tokenisation du texte**, qui convertit les mots en unités manipulables par le modèle
- **le mécanisme d'attention multi-têtes**, qui permet d'établir des relations entre les tokens pour capturer le contexte
- **l'auto-régression**, qui prédit chaque token un par un pour garantir une génération fluide

---

a. Dans le cadre de ce cours on étudiera uniquement des LLM qui prennent en entrée du texte, mais depuis 2024-2025 les LLM peuvent traiter toutes suites de séquences d'octets représentant image, son ou vidéo

Architecture des Transformers 2 / 4

Tokenisation du texte

- Le texte est transformé en une séquence de tokens (mots ou sous-mots)
- Utilisation de techniques comme BPE (Byte Pair Encoding) pour découper les mots en unités plus petites
- Permet au modèle de comprendre des mots inconnus en analysant leurs sous-unités

Exemple de tokenisation

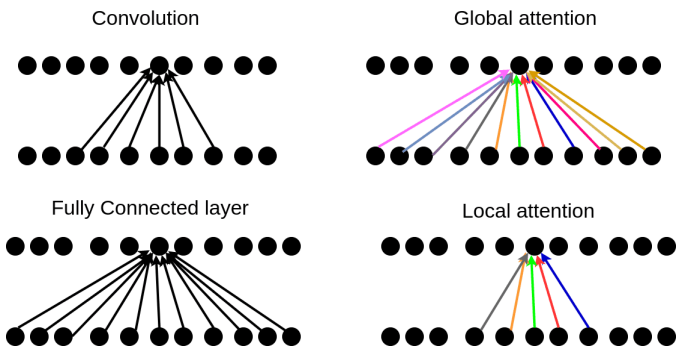
- Mot original : `"transformers"`
- Tokenisation BPE : `"trans"`, `"##form"`, `"##ers"`
- Réduction du vocabulaire de GPT-3 à environ 50 000 tokens

INSA ROUEN NORMANDIE

Architecture des Transformers 3 / 4

Mécanisme d'attention multi-têtes

- Chaque token peut prêter attention à d'autres tokens de la séquence
- Plusieurs têtes d'attention analysent différents aspects du contexte simultanément
- Permet de capturer des relations à longue distance entre les mots



https://theaisummer.com/attention/

Exemple de capacités d'attention

- GPT-4 utilise **8 à 16 têtes d'attention** par couche (entre 96 et 128 couches)
- Chaque tête peut capturer des dépendances linguistiques sur une séquence de **plus de 32 000 tokens**

Blog de Loïck Bourdois

https://lbourdois.github.io/blog/nlp

Architecture des Transformers 4 / 4

Auto-régression pour la prédiction des tokens

- Le modèle prédit chaque token un par un en utilisant les tokens précédents
- Utilisation du masquage pour empêcher l'accès aux futurs tokens durant l'entraînement
- Garantit une génération fluide et cohérente du texte

Exemple de génération de texte

- GPT-3 peut générer jusqu'à **2048 tokens** en une seule requête
- GPT-4 améliore cette capacité en atteignant **32 000 tokens** pour des analyses longues et complexes

INSA ROUEN NORMANDIE

Pré-entraînement et Fine-Tuning 1 / 3

Phases d'apprentissage

L'entraînement des LLM repose sur plusieurs étapes :

- **le pré-entraînement**, qui permet au modèle d'apprendre les structures du langage à partir d'un immense corpus de textes
- **les fine-tuning**, qui ajustent le modèle

INSA ROUEN NORMANDIE

Pré-entraînement et Fine-Tuning 2 / 3

Pré-entraînement

- Apprentissage non supervisé sur de vastes ensembles de données textuelles (livres, articles, pages web)
- Objectif : prédire le prochain token dans une phrase
- Modèles entraînés sur plusieurs téraoctets de données

Exemple de pré-entraînement

- GPT-3 a été entraîné sur un corpus estimé à plus de **500 milliards de tokens**
- Coût énergétique : plusieurs milliers de GPU pendant plusieurs semaines

INSA ROUEN NORMANDIE

Pré-entraînement et Fine-Tuning 3 / 3

Les étapes de Fine-Tuning

1. Fine-tuning supervisé :
   - Objectif : transformer les LLM en chatbot
   - Méthode : présentés des exemples de conversation
2. Fine-tuning par les préférences :
   - Objectif : améliorer la qualité des réponses et vérifié qu'elles sont légales et pas dangereuse
   - Méthode : des humains choisissent la meilleur réponse par mi celle proposée
3. Fine-tuning par le raisonnement
   - Objectif : améliorer la capacité du LLM à effectuer des tâches de raisonnement
   - Méthode : sur des problèmes vérifiables (comme les mathématiques ou le code informatique) on applique des méthodes d'apprentissage par renforcement (automatique)

Les 4 étapes pour entrainer les LLM (ScienceEtonnante)
https://www.youtube.com/watch?v=YcIbZGTRMiI

Quelques LLM en 2025

Modèle	Paramètres	Vocabulaire (tokens)	Contexte (tokens)	Type
GPT-4 Turbo (OpenAI)	1.76T	50K	128K	Propriétaire
LLaMA 3 70B (Meta)	70B	32K	8K	Open Source
Mistral 7B (Mistral AI)	7B	32K	8K	Open Source
Claude 3 (Anthropic)	Non divulgué	100K	200K	Propriétaire
Gemini 1.5 Pro (Google DeepMind)	Non divulgué	32K	1M	Propriétaire
DeepSeek 67B (DeepSeek AI)	67B	64K	8K	Open Source
Qwen 72B (Alibaba)	72B	150K	32K	Open Source
Falcon 180B (TII)	180B	65K	4K	Open Source
Command R+ (Cohere)	Non divulgué	50K	128K	Propriétaire
Mixtral (Mistral AI)	12.9B (MoE)	32K	8K	Open Source

Présentation de l'accès à un LLM

Un accès standard aux IA générant du texte

- Communication client-serveur utilisant l'architecture REST
  - Utilisable depuis une multitude de langages (dont python)
  - Utilise le protocole du Web (HTTP)
- Modes d'utilisation :
  - Text-to-Text (génération de texte)
  - Image-to-Text (analyse d'images)
  - Speech-to-Text (transcription audio)

INSA ROUEN NORMANDIE

Modes d'utilisation des modèles d'IA XX to Text

Principaux modes d'utilisation

- **Génération de texte (Text Completion)** : Complète une entrée utilisateur, utile pour la rédaction assistée et la reformulation de texte.
- **Interaction conversationnelle (Text-to-Text)** : Permet d'entretenir un dialogue fluide et contextuel, utilisé pour les chatbots et assistants virtuels
- **Analyse et description d'images (Image-to-Text)** : Extrait des informations d'une image, utilisé pour l'accessibilité et la reconnaissance d'objets
- Transcription audio en texte (Speech-to-Text) : Convertit un enregistrement audio en texte structuré, utilisé pour les sous-titres et la reconnaissance vocale

INSA ROUEN NORMANDIE

Préparation des requêtes API

Ce dont vous avez besoin pour interagir avec un serveur LLM

- Une **bibliothèque pour faire des requête Web** : utilisation du package 'requests'
- Une **clé API** : clé d'authentification nécessaire pour accéder aux modèles
- **URL du serveur**, par exemple `https://iti-llm.insa-rouen.fr`
- Au moins **un modèle** installé sur le serveur

INSA ROUEN NORMANDIE

Module `requests`

fonction `post`

- On va utiliser 3 paramètres nommés :
  1. `url` : l'URL du serveur en ajoutant `"/completions"` ou `/chat/completions`
  2. `header` : un dictionnaire :
     - `Auhorization` : pour indiquer la clé API `"Bearer XX"`
     - `Content-Type` : pour indiquer dans quel format le client et le serveur vont s'échanger les données `"application/json"`
  3. `data` : un dictionnaire qui dépend du mode d'utilisation, mais avec au moins le modèle (`"model": "YY"`)
- Le résultat contient un statut (`status_code`, 200 si tout OK, `text` à la raison sinon) et dictionnaire (accessible ç l'aide de la fonction `json()`)

Génération de texte 1 / 2

Clés-valeurs de `data`

`prompt`	Le texte d'entrée que le modèle doit compléter
`max_tokens`	Nombre maximal de tokens générés en sortie
`temperature`	Contrôle la créativité de la génération (valeurs typiques entre 0 et 1, où 0 produit des sorties plus déterministes et 1 génère des résultats plus variés)
`n`	Nombre de complétions à générer pour chaque requête (par défaut 1)
`top_p`	Alternative à `temperature`, filtre les réponses avec une probabilité cumulative (valeurs entre 0 et 1)
`stop`	Liste de séquences de texte qui arrêtent la génération lorsqu'elles sont rencontrées
`presence_penalty`	Encourage le modèle à parler de nouveaux sujets en augmentant la probabilité de nouveaux mots
`frequency_penalty`	Réduit la probabilité des mots répétés dans la sortie
`logprobs`	Permet d'obtenir les probabilités logarithmiques des tokens générés (valeurs entre 0 et 5, où 5 affiche les probabilités des 5 tokens les plus probables)

Clés-valeurs retournées

`id`	Identifiant unique de la requête
`object`	Type de l'objet retourné (ex : `"text_completion"`)
`created`	Timestamp indiquant la date et l'heure de la génération
`choices`	Liste des différentes réponses générées
`choices[0]["text"]`	Contenu du texte généré
`choices[0]["logprobs"]`	Probabilités logarithmiques des tokens générés si activé
`choices[0]["finish_reason"]`	Raison pour laquelle la génération s'est arrêtée (ex : `"stop"`)
`usage`	Informations sur l'utilisation des tokens (`"total_tokens"`, `"prompt_tokens"`, `"completion_tokens"`)

Génération de texte 2 / 2

```
import requests
API_KEY = ..
URL = "https://iti-llm.insa-rouen.fr/v1/completions"
MODEL = "mistralai/Codestral-22B-v0.1"

headers = {
    "Authorization": f"Bearer {API_KEY}",
    "Content-Type": "application/json"
}

data = {
    "model": MODEL,
    "prompt": "Explique le concept de machine learning en une phrase simple.",
    "max_tokens": 100,
    "temperature": 0.8,
}

response = requests.post(URL, headers=headers, json=data)

if response.status_code == 200:
    print(response.json())
else:
    print(f"Erreur {response.status_code}: {response.text}")
```

```
{'id': 'cmpl-198616d9d3e24357804f499ac9b35549',
'object': 'text_completion', 'created': 1741180268, 'model': 'mistralai/Codestral-22B-v0.1',
'choices': [{'index': 0, 'text': "\n\nLe machine learning est une branche de l'intelligence artificielle qui permet aux
        machines d'apprendre àpartir des données, sans être explicitement programmées, en améliorant leur capacité àeffectuer
        des tâches spécifiques en s'adaptant aux modèles détectés.", 'logprobs': None, 'finish_reason': 'stop', 'stop_reason':
         None, 'prompt_logprobs': None}], 'usage': {'prompt_tokens': 13, 'total_tokens': 89, 'completion_tokens': 76, '
        prompt_tokens_details': None}}
```

Interaction conversationnelle 1 / 2

Clés-valeurs de `data`

`messages`	Liste des messages envoyés et reçus, chaque message est un dictionnaire avec : • `role` : Décrit le rôle du message dans la conversation (`"system"`, `"user"`, `"assistant"`) • `content` : Contenu textuel du message • `name` (optionnel) : Nom du personnage IA lorsqu'il y a plusieurs assistants
`temperature`	Contrôle la créativité des réponses (valeurs typiques entre 0 et 1)
`n`	Nombre de réponses générées par requête
`top_p`	Alternative à `temperature`, filtre les réponses avec une probabilité cumulative
`presence_penalty`	Encourage le modèle à aborder de nouveaux sujets
`frequency_penalty`	Réduit la probabilité de répétition des mots
`stop`	Liste de séquences de texte qui arrêtent la génération de réponses

Clés-valeurs retournées

`id`	Identifiant unique de la requête
`object`	Type de l'objet retourné (ex : `"chat.completion"`)
`created`	Timestamp indiquant la date et l'heure de la génération
`choices`	Liste des différentes réponses générées
`choices[0]["message"]["content"]`	Contenu du message généré
`choices[0]["finish_reason"]`	Raison de l'arrêt de la réponse (ex : `"stop"`)
`usage`	Informations sur l'utilisation des tokens (`"total_tokens"`, `"prompt_tokens"`, `"completion_tokens"`)

Interaction conversationnelle 2 / 2

```python
import requests
API_KEY = ..
URL = "https://iti-llm.insa-rouen.fr/v1/chat/completions"
MODEL = "mistralai/Codestral-22B-v0.1"

headers = {
    "Authorization": f"Bearer {API_KEY}",
    "Content-Type": "application/json"
}

data = {
    "model": MODEL,
    "messages": [
        {"role": "system", "content": "Tu es un enseignant en IA."},
        {"role": "user", "content": "Bonjour, peux-tu m'expliquer ce qu'est un large modèle de langage?"}
    ],
    "temperature": 0.7
}

response = requests.post(URL, headers=headers, json=data)

if response.status_code == 200:
    print(response.json())
else:
    print(f"Erreur {response.status_code}: {response.text}")
```

```
'id': 'chatcmpl-40c6a1a118cb40a88bcdd0f994354b53', 'object': 'chat.completion', 'created': 1741183130, 'model': 'mistralai/
        Codestral-22B-v0.1',
'choices': [{'index': 0, 'message': {'role': 'assistant', 'content': ' Bonjour, tout d\'abord merci pour votre intérêt envers
        l\'Intelligence Artificielle (IA) !\n\nUn large modèle de langage, [...] de Google Research. Ces modèles ont été entra
        inés sur des ensembles de données contenant des milliards de mots et ont démontré des performances avancées sur un
        large éventail de tâches linguistiques.', 'tool_calls': []}, 'logprobs': None, 'finish_reason': 'stop', 'stop_reason':
         None}], 'usage': {'prompt_tokens': 40, 'total_tokens': 532, 'completion_tokens': 492, 'prompt_tokens_details': None},
         'prompt_logprobs': None}
```

Analyse et description d'images 1 / 3

Clés-valeurs de `data`

`messages`	Liste avec un message qui est un dictionnaire avec : • `role` : Décrit le rôle du message (`"system"`, `"user"`, `"assistant"`) • `content` : liste avec du textes et l'image encodée en base64 (cf. l'exemple)
`temperature`	Contrôle la créativité des réponses (valeurs typiques entre 0 et 1)
`top_p`	Alternative à `temperature`, filtre les réponses avec une probabilité cumulative
`n`	Nombre de descriptions générées pour chaque requête
`stop`	Liste de séquences de texte qui arrêtent la génération de réponse

Clés-valeurs retournées

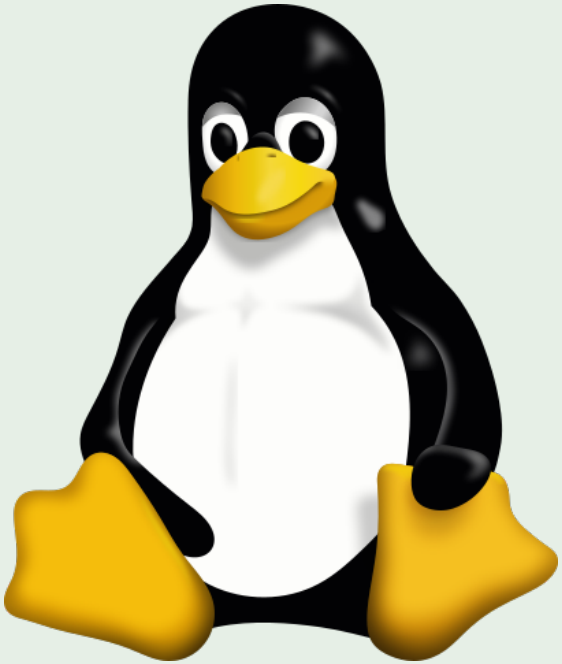
`id`	Identifiant unique de la requête
`object`	Type de l'objet retourné (ex : `"image_analysis"`)
`created`	Timestamp indiquant la date et l'heure de la génération
`choices`	Liste des différentes réponses générées
`choices[0]["message"]["content"]`	Description de l'image retournée par le modèle
`usage`	Informations sur l'utilisation des tokens (`"total_tokens"`, `"prompt_tokens"`, `"completion_tokens"`)

INSA ROUEN NORMANDIE

Analyse et description d'images 2 / 3

```
import requests
import base64
import mimetypes

API_KEY = ..
URL = "https://iti-llm.insa-rouen.fr/v1/chat/completions"
MODEL = "Qwen/Qwen2-VL-7B-Instruct"

headers = {
    "Authorization": f"Bearer {API_KEY}",
    "Content-Type": "application/json"
}

filename = "Tux.png"
mime_type, _ = mimetypes.guess_type(filename)
with open(filename, "rb") as image_file:
    file_content = base64.b64encode(image_file.read()).decode("utf-8")
url_base64 = f"data:{mime_type}; base64,{file_content}\n"

data = {
    "model": MODEL,
    "messages": [
        {"role": "user",
         "content": [
             {
                 "type": "text",
                 "text": "Décris cette image en détail."
             },
             {
                 "type": "image_url",
                 "image_url": {'url': url_base64}
             }
         ]
        }
    ]
}

response = requests.post(URL, headers=headers, json=data)
```

Analyse et description d'images 3 / 3




```
{"id":"chatcmpl-fb352d20e0b94030b1fb05449e393384","object":"chat.completion",
"created":1741186657,"model":"Qwen/Qwen2-VL-7B-Instruct",
"choices":[{"index":0,"message":{"role":"assistant","content":"Cette image montre
        un pingouin, l'icône officielle du système d'exploitation Linux. Le
        pingouin est représenté de manière stylisée avec des yeux ronds et un bec
        jaune. Les pieds de l'animal sont traditionnellement jaunes et le corps
        est noir. Cet icône est largement reconnu en tant qu'élément symbole du
        système Linux.",
"tool_calls":[]},"logprobs":null,"finish_reason":"stop","stop_reason":null}],"
        usage":{"prompt_tokens":264,"total_tokens":356,"completion_tokens":92,"
        prompt_tokens_details":null},"prompt_logprobs":null}d
```

Pourquoi le prompt est essentiel ?

Un prompt bien conçu améliore fortement les résultats

- Oriente le LLM vers la bonne réponse
- Limite les erreurs ou hallucinations
- Permet des résultats plus cohérents et ciblés

Méthode proposée par Google

Tâche, Contexte, Références, Évaluer, Itérer : une démarche structurée pour rédiger des prompts efficaces

Formation IA de Google : 10h en Version Accélérée (Ludo Salenne)
`https://www.youtube.com/watch?v=mFTNLy2_Gl8`

INSA ROUEN NORMANDIE

Étape 1 : Tâche

Définir clairement la tâche

Indiquer précisément ce que l'on attend du modèle :

- Action attendue : résumer, traduire, corriger, etc
- Niveau de détail ou de complexité souhaité

Fil rouge

Demande initiale : "Résume cet article."
Améliorée : *"Crée un résumé compréhensible par un enfant de 10 ans."*

INSA ROUEN NORMANDIE

Étape 2 : Contexte

Fournir les informations nécessaires

Aide le modèle à comprendre la situation :

- Qui est l'utilisateur cible ?
- Quels sont les objectifs ou contraintes ?
- Quelle est la source ou matière première ?

Fil rouge

- Article à résumer : [insérer un extrait]
- Public cible : enfant de 10 ans
- Objectif : transmettre les points clés simplement

INSA ROUEN NORMANDIE

Étape 3 : Références

Donner des exemples ou modèles

Permet d'inspirer le format, le style ou la tonalité :

- Exemples de réponses attendues
- Format ou ton spécifique à adopter
- À éviter : prompts vagues sans cadre

Fil rouge

"Un volcan est entré en éruption. Les habitants sont partis se mettre à l'abri. Personne n'a été blessé."

Style : phrases courtes, vocabulaire simple, ton rassurant

INSA ROUEN NORMANDIE

Étape 4 : Évaluer

Analyser le résultat obtenu

Vérifier que la réponse :

- Répond bien à la demande
- Respecte le style et le ton attendus
- Est cohérente, correcte et exploitable

Fil rouge

Critères d'évaluation :

- Le vocabulaire est-il adapté à un enfant ?
- Le message est-il clair, fidèle à l'article ?

INSA ROUEN NORMANDIE

Étape 5 : Itérer

Améliorer le prompt progressivement

Modifier le prompt selon le résultat :

- Ajouter des précisions
- Changer le ton ou le style
- Fournir de nouveaux exemples

Fil rouge

Si le ton est trop formel : *"Utilise un langage simple, comme si tu parlais à un enfant."*
Si les infos sont trop techniques : *"Explique les mots compliqués ou remplace-les."*

INSA ROUEN NORMANDIE

Défis et Limitations des LLM 1 / 2

Hallucinations et Fiabilité

- Les modèles peuvent générer des informations incorrectes ou fictives
- Pas de réelle compréhension du monde, uniquement une probabilité d'apparition des mots
- Risque d'utilisation dans la désinformation et la manipulation

Biais et Éthique

- L'entraînement sur des données biaisées peut entraîner des réponses discriminatoires
- Possibilité de renforcer des stéréotypes sociaux ou culturels
- Besoin de supervision humaine pour éviter des dérives

INSA ROUEN NORMANDIE

Défis et Limitations des LLM 2 / 2

Coût Computationnel et Impact Environnemental

- Entraînement nécessitant des milliers de GPU/TPU pendant plusieurs semaines
- Empreinte carbone significative due à la consommation énergétique élevée
- Recherche en cours sur des modèles plus efficients et moins gourmands en ressources

Problèmes Légaux et Sécurité

- Protection des données personnelles et conformité aux régulations (RGPD, etc.)
- Risque d'utilisation malveillante (fraude, deepfakes, etc.)
- Attribution et droit d'auteur : Qui est propriétaire du contenu généré ?

Futurs Développements des LLM 1 / 3

Modèles Plus Efficaces et Accessibles

- Développement de modèles plus légers et optimisés pour réduire la consommation énergétique
- Open-source et modèles personnalisables pour une adoption plus large
- Intégration d'algorithmes permettant un entraînement plus rapide et moins coûteux

Amélioration de la Compréhension et du Raisonnement

- Renforcement des capacités de raisonnement logique et mathématique
- Intégration de la mémoire contextuelle pour des interactions plus longues et cohérentes
- Modèles hybrides combinant bases de connaissances et LLM pour plus de précision

Futurs Développements des LLM 2 / 3

Développement des IA Multimodales

- Capacité d'analyse et de génération de texte, d'images, de vidéos et de sons combinés
- Fusion entre LLM et reconnaissance d'images pour améliorer l'interaction
- Utilisation dans des applications comme la réalité augmentée et virtuelle

Agentification des LLM

- Développement d'IA autonomes capables d'interagir avec des outils externes et d'exécuter des tâches complexes
- Intégration avec des API, bases de données et systèmes informatiques pour agir comme des assistants intelligents
- Capacité d'adaptation en fonction du contexte et des retours utilisateurs pour des interactions plus efficaces
- Évolution vers des agents multi-étapes capables de planifier et d'exécuter des tâches longues et stratégiques

Futurs Développements des LLM 3 / 3

Meilleure Explicabilité et Sécurité

- Transparence accrue sur la manière dont les modèles prennent leurs décisions
- Intégration de filtres plus avancés pour limiter les biais et les contenus nuisibles
- Renforcement des mécanismes de vérification des faits pour éviter les hallucinations

INSA ROUEN NORMANDIE

Conclusion

Synthèse

- Les LLM révolutionnent le traitement automatique de la langue
- De nombreuses applications
- Pour le grand public IA = LLM
- Leur accès est standardisé
- Pour avoir des bons résultats, il faut utiliser des bons prompts
- Il y a encore beaucoup de problème à résoudre, mais la recherche est très productive dans ce domaine

INSA ROUEN NORMANDIE