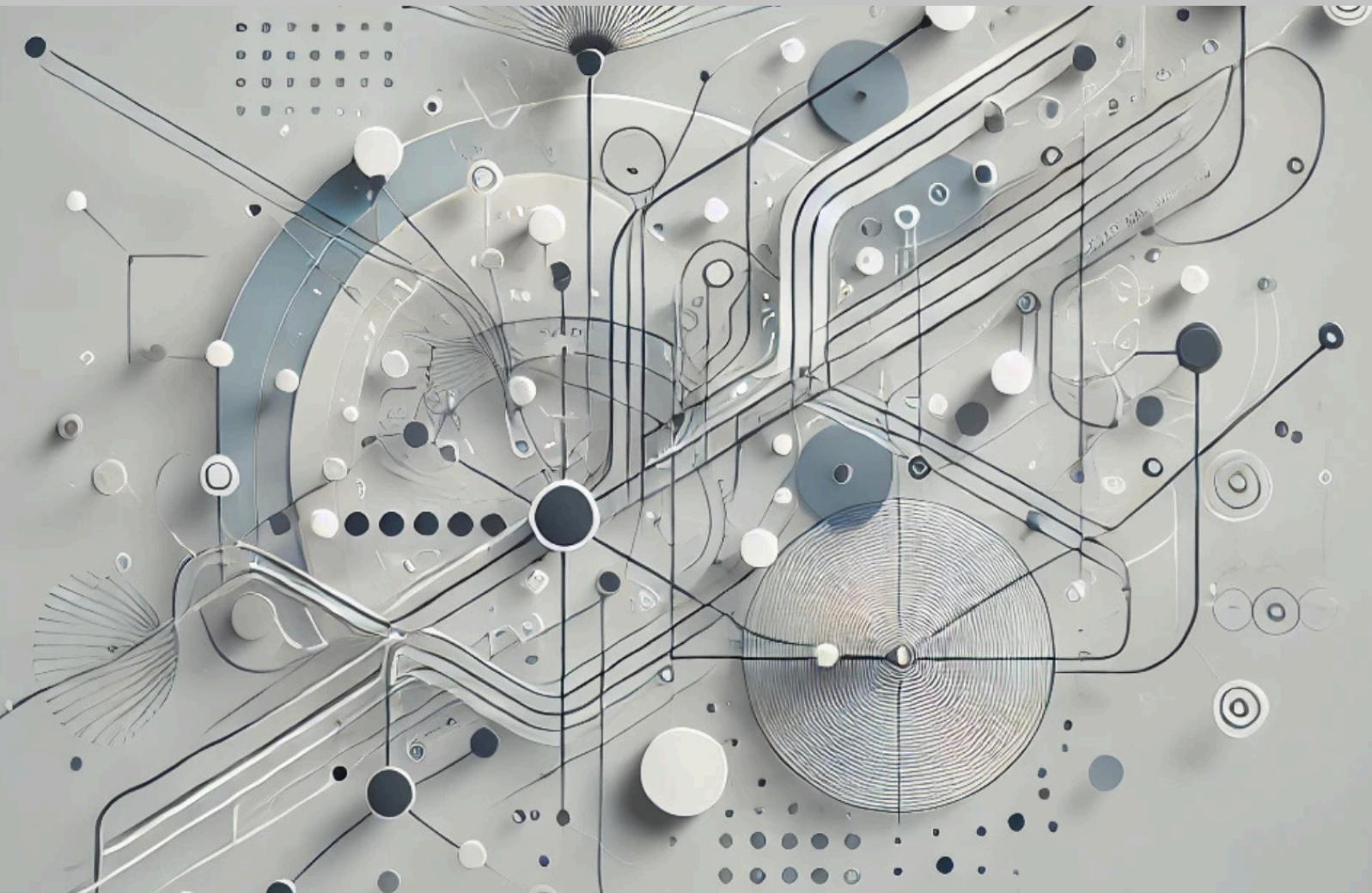


RÉALISÉ PAR :

KYLIAN GACHET & ENZO FOURNET

# IA GÉNÉRATIVE ET RAG

Modélisation  
Mathématiques



<b>Introduction.....</b>	<b>3</b>
Origines et évolutions de l'intelligence artificielle.....	3
Les avancées récentes et l'émergence des IA génératives.....	4
Limites des IA génératives actuelles.....	5
Présentation de la problématique : le besoin d'une approche augmentée.....	6
Objectifs du mémoire.....	6
<b>1. Qu'est-ce que l'IA ?.....</b>	<b>7</b>
1.1. Les différents types d'IA.....	7
1.1.1. IA étroite ou faible (ANI - Artificial Narrow Intelligence).....	7
1.1.2. IA générale (AGI - Artificial General Intelligence).....	8
1.1.3. Super IA (ASI - Artificial Super Intelligence).....	8
<b>2. Le fonctionnement de l'IA - introduction rapide.....</b>	<b>9</b>
2.1. Neurone et réseau de neurones.....	9
2.2. La régression linéaire.....	9
2.3. Probabilités et statistiques.....	10
2.4. Apprentissage.....	10
<b>3. Retrieval-Augmented Generation (RAG).....</b>	<b>11</b>
3.1. Besoin initial.....	11
3.2. Concepts.....	11
3.2.1. Tokenisation : exemple avec le package tiktoken d'OpenAI.....	12
3.2.1. Base de données vectorielle.....	16
3.2.2. Comparaison vectorielle : similarité cosinus.....	16
3.3. Le fonctionnement du RAG.....	19
3.4. Exemple concret.....	21
3.4.1. Fonctionnement.....	21
3.4.2. Demo.....	26
3.5. Avantages.....	30
3.6. Inconvénient.....	30
3.7. Applications et cas d'usages de la RAG.....	30
<b>Conclusion.....</b>	<b>32</b>

# Introduction

## Origines et évolutions de l'intelligence artificielle

L'histoire de l'intelligence artificielle (IA) remonte à plusieurs siècles, bien avant l'apparition des ordinateurs modernes. Dès le **XIIIe siècle**, Al-Jazari, un inventeur musulman, concevait des **automates** sophistiqués, dont une fontaine animée qui pouvait, en quelque sorte, être considérée comme une première tentative de créer des machines "intelligentes".

En **1642**, Blaise Pascal inventa la **machine à calculer**, une machine mécanique capable de réaliser des opérations arithmétiques de base, posant ainsi les fondements des futures calculatrices. Le développement des machines de calcul se poursuivit avec l'invention du **métier à tisser Jacquard** en **1804**, utilisant des cartes perforées pour automatiser le tissage, une technique qui inspirera plus tard la programmation informatique.

En parallèle, des **concepts mathématiques** essentiels, tels que la **logique formelle** (remontant aux travaux d'Aristote et formalisée au XIXe siècle par des mathématiciens comme Boole et Frege) et les **probabilités** (étudiées depuis Pascal et Fermat au XVIIe siècle), ont permis de développer les fondements théoriques nécessaires pour la construction de systèmes d'IA. Ces théories mathématiques sont à la base des algorithmes modernes, en particulier pour la prise de décision et la modélisation de l'incertitude.

L'un des premiers concepts d'un programme informatique est proposé en **1842** par **Ada Lovelace**, qui imagina la possibilité d'instruire une machine pour réaliser une série d'opérations, devenant ainsi l'une des premières à concevoir l'idée d'une machine programmable.

Dans les années **1930**, **Alan Turing** développa la notion de **machine abstraite**, ou "machine de Turing", qui est aujourd'hui reconnue comme un modèle théorique pour les ordinateurs modernes. Ce concept fondamental de la logique et du calcul ouvre la voie à la création de véritables ordinateurs.

Le développement de l'IA moderne commença véritablement dans les années **1950**. En **1956**, lors de la conférence de Dartmouth, des scientifiques de diverses disciplines se réunirent pour discuter de la création de machines capables d'apprendre et de simuler l'intelligence humaine. C'est à cette occasion que le terme "**intelligence artificielle**" fut officiellement utilisé pour la première fois. Cette décennie vit également le développement du premier **ordinateur ENIAC** (1946) et des **premiers algorithmes d'apprentissage** (1957), posant ainsi les bases de la programmation des machines à penser.

Les décennies suivantes furent marquées par de grands progrès et des phases de doute, souvent appelées les "**hivers de l'IA**" (comme celui de **1974-1980**), où les financements et l'enthousiasme pour l'IA déclinaient faute de résultats satisfaisants. Cependant, des percées

comme le **Chatbot ELIZA** (1966), un programme de conversation simulée, démontrèrent des avancées en matière de traitement du langage.

L'intelligence artificielle connaît un nouvel essor dans les années **1980** avec **l'âge d'or des systèmes experts**, des programmes capables d'imiter le raisonnement humain dans des domaines spécifiques. En **1997**, **Deep Blue**, un superordinateur conçu par IBM, a battu le champion du monde d'échecs Garry Kasparov, symbolisant l'ascension de l'IA dans le domaine des jeux stratégiques et de la résolution de problèmes complexes.

La fin du XXe siècle et le début du XXIe siècle marquent l'émergence d'applications de l'IA dans la vie quotidienne. En **2007**, l'apparition du **premier smartphone** permet à l'IA d'interagir directement avec les utilisateurs via des interfaces vocales et des assistants personnels (Siri, Ok Google...). Des avancées comme la **reconnaissance faciale** (1994) et le développement du **World Wide Web** (1989) contribuent également à la collecte massive de données, essentielles pour l'apprentissage des machines.

Avec l'apparition de **GPT-3** en **2020** et d'autres modèles de génération de texte, l'IA franchit une nouvelle étape, capable de produire du contenu de manière autonome et de s'adapter à des tâches de plus en plus variées, ouvrant ainsi la voie à la génération augmentée par la récupération (RAG), une solution prometteuse pour pallier certaines limites des modèles actuels.

## **Les avancées récentes et l'émergence des IA génératives**

Comme mentionné précédemment, les concepts mathématiques à la base des IA d'aujourd'hui, tels que l'algèbre linéaire, la logique formelle et les probabilités, existaient déjà depuis plusieurs siècles. Cependant, ce n'est qu'à partir des années 2010, l'IA connaît une véritable révolution avec l'émergence de l'**apprentissage profond** (deep learning), rendu possible par l'augmentation des capacités de calcul et la disponibilité de vastes ensembles de données. Les réseaux de neurones profonds permettent des progrès sans précédent dans des domaines tels que la vision par ordinateur et le traitement du langage naturel.

En 2017, l'introduction de l'architecture **Transformer** par Google marque un tournant majeur, permettant une amélioration significative des performances des modèles d'IA dans le traitement du langage. Cette architecture inspire des modèles tels que **BERT**, **GPT** (Generative Pre-trained Transformer) et leurs successeurs, qui repoussent les limites de ce que les machines peuvent accomplir en termes de compréhension et de génération de texte.

En 2020, l'arrivée de **GPT-3** d'OpenAI, un modèle d'IA générative capable de produire du texte de manière fluide et contextuelle, marque un jalon dans le domaine de la génération de contenu automatique. Désormais, les IA génératives sont capables de répondre à des questions, rédiger des articles, créer des images, et même assister les développeurs en écrivant du code. Cette avancée ouvre de nouvelles perspectives pour l'automatisation de tâches créatives et intellectuelles.

Aujourd'hui, **GPT-4o** d'OpenAI, l'un des successeurs de GPT-3, accompagné par les innovations de **NVIDIA** en matière de puissance de calcul, permet à ces modèles de fonctionner avec des performances accrues et une précision encore améliorée. Ces progrès technologiques ouvrent des possibilités sans précédent, rendant les IA génératives capables de s'adapter à des contextes plus diversifiés et de fournir des réponses plus nuancées. Ces avancées posent cependant des défis, notamment en matière de fiabilité de l'information et de biais, soulevant la question de savoir comment optimiser ces modèles pour répondre aux attentes croissantes en matière de pertinence et d'actualité.

## **Limites des IA génératives actuelles**

Malgré les performances impressionnantes des modèles génératifs récents, des limitations importantes subsistent. Ces modèles sont souvent décrits comme des « machines à prédiction de texte » : ils génèrent du contenu en s'appuyant sur les probabilités statistiques calculées à partir des données d'entraînement. Cela signifie qu'ils sont pour la plupart incapables de mettre à jour leurs connaissances ou d'accéder à des informations actualisées en temps réel, ce qui peut conduire à des réponses obsolètes ou inexactes.

En outre, ces IA peuvent être influencées par des biais présents dans leurs données d'entraînement, produisant parfois des réponses partiales ou inappropriées. Les IA génératives actuelles peuvent également manquer de précision pour répondre à des questions complexes nécessitant des connaissances spécialisées ou actualisées, limitant ainsi leur utilité dans certains contextes professionnels.

Ces limites soulèvent une question fondamentale pour le développement de l'IA : comment enrichir les modèles génératifs pour qu'ils soient capables de produire des réponses non seulement créatives, mais aussi précises, pertinentes et adaptées au contexte ?

## Présentation de la problématique : le besoin d'une approche augmentée

Pour pallier ces lacunes, l'approche **Retrieval-Augmented Generation (RAG)**, ou génération augmentée par la récupération, apparaît comme une solution prometteuse. Elle combine la puissance des modèles génératifs avec un mécanisme de récupération d'informations en temps réel, permettant de produire des réponses mieux informées et davantage contextualisées. Ce mémoire s'intéressera à la problématique suivante : **comment le RAG peut-il apporter une plus-value significative aux modèles d'IA générative ?**

En explorant cette question, ce travail vise à démontrer que le RAG représente une avancée importante dans l'évolution de l'IA, comblant les limites des modèles de génération pure et ouvrant de nouvelles perspectives pour l'automatisation et l'optimisation des tâches complexes.

## Objectifs du mémoire

L'objectif de ce mémoire est d'abord de présenter rapidement le fonctionnement des IA génératives et les différents types d'IA existant aujourd'hui, en se concentrant particulièrement sur les **LLM** (Large Language Models). Nous expliquerons ensuite le fonctionnement du **RAG** (Retrieval-Augmented Generation) et en proposerons un exemple que nous aurons mis en œuvre. Enfin, nous tenterons de répondre à notre problématique en explorant les cas d'usage et la plus-value que le RAG peut apporter aux LLM dans divers domaines.

## 1. Qu'est-ce que l'IA ?

L'intelligence artificielle (IA) est souvent perçue par le grand public comme une technologie capable de simuler l'intelligence humaine dans des machines programmées pour effectuer des tâches et prendre des décisions de manière autonome.

Cette vision populaire s'accompagne fréquemment d'abus de langage et de clichés, tels que l'idée que l'IA possède une véritable conscience ou qu'elle pourrait un jour remplacer complètement l'humain. D'un point de vue plus rigoureux, l'IA peut être définie comme un ensemble de théories et de techniques visant à réaliser des machines capables de simuler certaines fonctions cognitives humaines.

Elle fait appel à des disciplines telles que les mathématiques, les sciences cognitives, la neurobiologie computationnelle et l'informatique pour résoudre des problèmes à forte complexité logique ou algorithmique.

### 1.1. Les différents types d'IA

Aujourd'hui, les niveaux de capacités d'intelligence artificielle sont généralement classés en trois catégories principales.

#### 1.1.1. IA étroite ou faible (ANI - Artificial Narrow Intelligence)

L'ANI représente le niveau actuel de l'IA. Elle est conçue pour effectuer des tâches spécifiques dans un domaine particulier. Les systèmes d'ANI excellent dans leur domaine d'expertise, mais sont limités à celui-ci. ChatGPT et les IA génératives similaires à celui-ci, par exemple, font partie de cette catégorie d'IA étroite. Bien que très performant dans le traitement du langage naturel, il ne possède pas de compréhension générale du monde ni de capacités cognitives comparables à celles d'un être humain.

### **1.1.2. IA générale (AGI - Artificial General Intelligence)**

L'AGI représente un niveau d'intelligence artificielle hypothétique qui serait capable d'effectuer pratiquement n'importe quelle tâche cognitive propre aux humains. Contrairement à l'ANI, une AGI pourrait travailler en autonomie sur une grande variété de problèmes, sans intervention humaine. Ce niveau d'IA n'existe pas encore et fait l'objet de nombreuses recherches et débats quant à sa faisabilité et son impact potentiel sur la société.

Depuis l'essor des IA génératives en 2020, chaque année nous rapproche davantage du développement de systèmes d'intelligence artificielle capables de prendre en charge certaines activités humaines, voire de les remplacer dans des domaines spécifiques.

### **1.1.3. Super IA (ASI - Artificial Super Intelligence)**

L'ASI est un concept encore plus avancé et spéculatif que l'AGI. Il s'agirait d'une intelligence artificielle surpassant largement les capacités cognitives humaines dans tous les domaines. Cette forme d'IA reste pour l'instant du domaine de la science-fiction et soulève de nombreuses questions éthiques et philosophiques sur l'avenir de l'humanité face à une intelligence supérieure artificielle.

## 2. Le fonctionnement de l'IA - introduction rapide

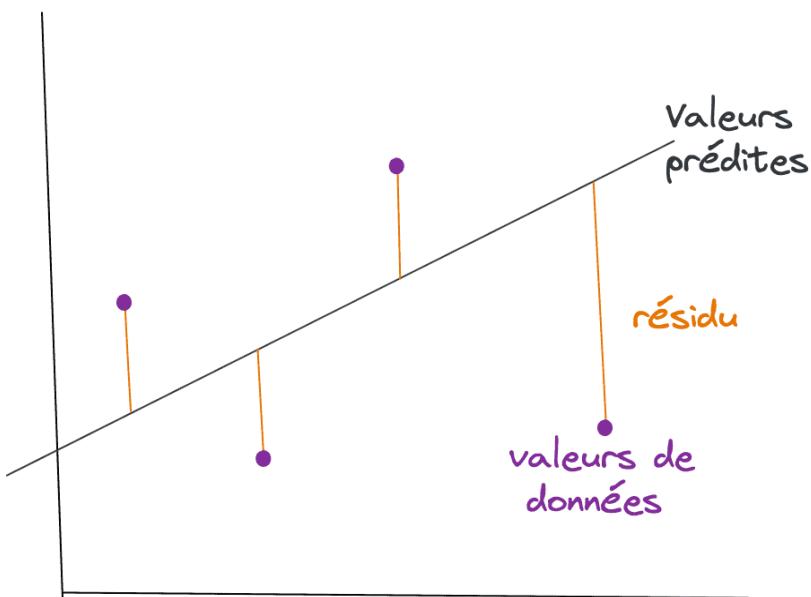
### 2.1. Neurone et réseau de neurones

Le neurone artificiel est l'unité de base des réseaux de neurones, imitant le fonctionnement des neurones biologiques. Chaque neurone reçoit des entrées, les transforme à l'aide de poids et d'une fonction d'activation, puis transmet le résultat aux neurones suivants.

Les réseaux de neurones sont organisés en couches : une couche d'entrée pour les données, des couches cachées pour le traitement et une couche de sortie pour les résultats. Cette structure permet aux réseaux d'apprendre des représentations complexes à partir des données.

### 2.2. La régression linéaire

La régression linéaire est une méthode statistique utilisée pour modéliser la relation entre une variable dépendante et une ou plusieurs variables indépendantes. Elle cherche à établir une ligne droite qui minimise la distance entre les points de données et la ligne, permettant ainsi de prédire des valeurs futures. Bien que simple, cette technique est fondamentale en apprentissage automatique et sert de base à des modèles plus complexes.



Représentation graphique d'une régression linéaire

### 2.3. Probabilités et statistiques

Les probabilités et les statistiques sont essentielles au fonctionnement de l'IA. Elles permettent aux modèles de gérer l'incertitude et d'interpréter les données de manière significative. Voici comment ces concepts s'appliquent :

- **Évaluation de l'incertitude** : les probabilités aident les modèles d'IA à quantifier le degré de confiance dans leurs prédictions ou décisions.
- **Interprétation des résultats** : les statistiques fournissent des outils pour analyser et comprendre les tendances dans les données traitées par l'IA.
- **Inférences à partir des données** : les algorithmes d'apprentissage automatique utilisent des méthodes statistiques pour tirer des conclusions générales à partir d'échantillons de données.
- **Modélisation des erreurs** : la distribution normale, par exemple, est souvent utilisée pour représenter les erreurs dans les prédictions des modèles.

Ces concepts permettent aux systèmes d'IA de prendre des décisions éclairées en tenant compte de l'incertitude inhérente aux données et aux prédictions.

### 2.4. Apprentissage

L'apprentissage est au cœur du fonctionnement de l'IA, permettant aux modèles d'améliorer leurs performances au fil du temps. Il existe plusieurs types d'apprentissage :

- **Supervisé** : le modèle apprend à partir d'exemples étiquetés. C'est comme un professeur qui guide l'apprentissage en montrant des exemples avec des réponses correctes. Le modèle apprend à reconnaître des motifs et à faire des prédictions précises en s'appuyant sur ces exemples préétiquetés.
- **Non supervisé** : ici, le modèle travaille sans étiquettes, cherchant à découvrir des structures et des motifs cachés dans les données. C'est comme un détective qui analyse des informations sans indications préalables, regroupant des éléments similaires et identifiant des tendances naturelles dans les données.
- **Par renforcement** : le modèle apprend par essais et erreurs en recevant des récompenses pour les actions positives ou des pénalités pour les actions négatives.

Ces différents types de méthodes permettent aux systèmes d'IA de s'adapter à des environnements variés.

## 3. Retrieval-Augmented Generation (RAG)

### 3.1. Besoin initial

Le RAG, ou Génération Augmentée par Récupération, répond aujourd’hui à plusieurs besoins cruciaux dans le domaine de l’IA générative :

- **Améliorer la précision et la fiabilité** des réponses générées par les LLMs, afin de pouvoir se rapprocher vers des réponses de plus en plus cohérentes et pertinentes.
- Fournir des **informations à jour et vérifiables/sourcées**.
- **Réduire le phénomène d’hallucination** des modèles d’IA générative.
- Permettre l'accès à des connaissances spécifiques sans réentraînement coûteux du modèle.

### 3.2. Concepts

Le RAG fonctionne en deux phases principales :

- **Récupération (Retrieval)** : lorsqu'une requête est soumise, le système recherche des informations pertinentes dans une base de connaissances externe.

Par exemple, celui-ci ira récupérer des informations dans :

- Des **bases de données et de connaissances spécialisées** : comme *Wikidata*, qui est une base de connaissances libre et collaborative, conçue pour centraliser et structurer les données utilisées par les différents projets du mouvement *Wikimedia*, tels que *Wikipédia*, *Wikivoyage*, et *Wikisource*.
- Des **APIs externes** : comme des services d'information météorologique, financière ou d'actualités. Ces APIs permettent d'accéder à des données en temps réel ou régulièrement mises à jour dans des domaines spécifiques.
- Des **bases de données internes à l'entreprise** : comme des documentations, rapports internes, bases de connaissances d'entreprise...

Cette recherche utilise généralement des techniques de recherche sémantique et des bases de données vectorielles pour trouver rapidement les informations les plus pertinentes.

- **Génération (Generation)** : les informations récupérées sont ensuite triées et intégrées au prompt initial, en sélectionnant les éléments les plus pertinents. Elles sont ensuite transmises au model. Ce dernier utilise ces informations, combinées à ses connaissances internes, pour générer une réponse à la fois plus précise et plus adaptée au contexte.

### 3.2.1. Tokenisation : exemple avec le package tiktoken d'OpenAI

Un token dans le contexte de l'IA générative est une unité de base utilisée pour traiter et analyser le texte. Ces tokens interviennent à plusieurs étapes clés dans le traitement du langage naturel par les systèmes d'IA générative, notamment :

#### 1. Prétraitement du texte :

- 1.1. Lors de l'entrée de texte dans le système, le texte est découpé en tokens (mots, sous-mots ou caractères) par le tokeniser du modèle.
- 1.2. Ces tokens, après découpage du texte initial, deviennent des unités de texte et sont ensuite convertis en *token IDs*.
- 1.3. Les token IDs permettent de transformer les unités de texte en une forme numérique que le modèle peut traiter efficacement, facilitant ainsi l'analyse et la génération de langage naturel.

#### 2. Traitement par le modèle :

- 2.1. Les tokens sont convertis en représentations vectorielles.
- 2.2. Le modèle traite ces vecteurs pour comprendre le contexte et générer une réponse.

#### 3. Génération de la sortie :

- 3.1. Le modèle produit une séquence de tokens qui est ensuite décodée en texte lisible.
- 3.2. Le nombre de tokens générés peut être limité par les paramètres du modèle.

La **tokenisation** est donc un processus essentiel dans le traitement du langage naturel (NLP). Le package Python [tiktoken](#) d'OpenAI permet de gérer cette tâche efficacement, notamment pour les modèles de langage comme GPT.

### 3 - Retrieval-Augmented Generation (RAG)

Voici un exemple simple d'utilisation de tiktoken :

```
 1 import tiktoken
 2
 3 # Nom de l'encodage
 4 #   `o200k_base` : `gpt-4o`, `gpt-4o-mini`
 5 #   `cl100k_base` : `gpt-4-turbo`, `gpt-4`, `gpt-3.5-turbo`, `text-embedding-ada-002`, `text-embedding-3-small`, `text-embedding-3-Large`
 6 #   `p50k_base` : Codex models, `text-davinci-002`, `text-davinci-003`
 7 #   `r50k_base` (or `gpt2`) : GPT-3 models like `davinci`
 8 encoding_name = "cl100k_base"
 9
10 # Initialise le tokeniser avec l'encodage spécifié
11 tokenizer = tiktoken.get_encoding(encoding_name)
12
13 # Exemple de texte à tokeniser
14 text = "Bonjour, comment ça va ?"
15
16 # Tokenise le texte pour obtenir les IDs des tokens
17 tokensIDs = tokenizer.encode(text)
18
19 # Récupère l'encodage pour décoder les valeurs des tokens
20 encoding = tiktoken.get_encoding(encoding_name)
21
22 # Décode chaque ID de token en sa valeur correspondante
23 tokensValues = [encoding.decode_single_token_bytes(token) for token in tokensIDs]
24
25 # Affichage du texte initial
26 print("Affichage du texte initial : ", text) # Affiche les tokens
27
28 # Affichage des IDs des tokens
29 print("Affichage des tokenIDs : ", tokensIDs) # Affiche les IDs des tokens
30
31 # Affichage des valeurs de chaque token
32 print("Affichage des valeurs de chaque token : ", tokensValues) # Affiche les valeurs des tokens
```

Code utilisant tiktoken

```
PS C:\Users\User\OneDrive\Bureau\BUT3\ModelisationMathematique\rag-qa-system> & c:/Users/User/OneDrive/Bur  
eDrive/Bureau/BUT3/ModelisationMathematique/rag-qa-system/Tokenisation/tokenisation_openai.py  
Affichage du texte initial : Bonjour, comment ça va ?  
Affichage des tokenIDs : [82681, 11, 4068, 39043, 11412, 949]  
Affichage des valeurs de chaque token : [b'Bonjour', b', ', b' comment', b' \xc3\xaa7a', b' va', b' ?']
```

Résultat de test de tiktoken

(Vous pouvez retrouver cet exemple [ici](#))

Et ci-dessous un exemple de tokenisation du model GPT-4o & 4o mini d'OpenAI :

### 3 - Retrieval-Augmented Generation (RAG)

The screenshot shows the Hugging Face Tokenizer interface. On the left, a text input field contains "Bonjour, comment ça va ?". Below it, a table shows "Tokens" (6) and "Characters" (24). A preview window shows the tokens: [45751, 11, 5375, 13590, 3423, 1423]. On the right, a detailed view shows "Tokens" (6) and "Characters" (24), with a list of token IDs: [45751, 11, 5375, 13590, 3423, 1423].

Exemple de tokenisation de GTP40

Trouvable [ici](#).

Ce qu'il faut savoir, c'est que la **tokenisation** est un processus entièrement déterministe. Cela signifie qu'une même liste de mots ou de caractères sera toujours tokenisée de manière identique.

Par exemple, si l'on prend le tokenizer **cl100k\_base**, en examinant le code permettant d'effectuer la tokenisation, on peut constater qu'une liste de mots, encodée en **base64**, correspond directement à des **identifiants (IDs)**. Ces identifiants sont ensuite utilisés pour représenter chaque mot ou sous-mot dans le modèle.

The screenshot shows a web-based tool for encoding and decoding text to Base64. On the left, a "Text\*" input field contains "Bonjour", with "copy", "clear", and "download" buttons above it. On the right, a "Base64" output field contains "Qm9uam91cg==", also with "copy", "clear", and "download" buttons above it. Below the input field is a blue button labeled "Encode text to Base64". Below the output field is a blue button labeled "Decode Base64 to text".

Conversion en base 64 du mot "Bonjour"

82678	IERhdGFDb250ZXh0 82677
82679	Y29uc3RydWN0ZWQ= 82678
82680	QU5ESU5H 82679
82681	LS0tLS0tLS0tLS0qLwo= 82680
82682	Qm9uam91cg= 82681
82683	X1BIUA= 82682
82684	cHJvZ3Jlc3NiYXI= 82683
82685	Tm90U3VwcG9ydGVkRXhjZXB0aW9u 82684

Correspondance du mot “Bonjour” pour [cl100k\\_base](#)

```

4
5     def cl100k_base():
6         mergeable_ranks = load_tiktoken_bpe(
7             "https://openai-public.blob.core.windows.net/encodings/cl100k_base.tiktoken",
8             expected_hash="223921b76ee99bde995b7ff738513eef100fb51d18c93597a113bcffe865b2a7",
9         )
10        special_tokens = {
11            ENDOFFTEXT: 100257,
12            FIM_PREFIX: 100258,
13            FIM_MIDDLE: 100259,
14            FIM_SUFFIX: 100260,
15            ENDOFPROMPT: 100276,
16        }
17        return {
18            "name": "cl100k_base",
19            "pat_str": r"""\b(i:sdmt|ll|ve|re)\b|\b(^r\n\np\{L\}\p\{N\})?+\p\{L\}+\b|\p\{N\}{1,3}+\b|\b(^s\np\{L\}\p\{N\})\b""",
20            "mergeable_ranks": mergeable_ranks,
21            "special_tokens": special_tokens,
22        }

```

Extrait de code de [cl100k\\_base](#)

Et on constate, ci-dessus, dans le code de [cl100k\\_base](#), que nous rencontrons parfois des **tokens dits “spéciaux”**. Ces tokens sont utilisés, par exemple, par OpenAI pour délimiter des parties spécifiques du prompt envoyé au modèle, afin qu'il les interprète correctement.

Ces tokens spéciaux jouent un rôle crucial dans la communication avec le modèle. Ils peuvent servir à structurer les entrées, comme indiquer le début ou la fin d'une instruction, différencier les rôles (système, utilisateur, assistant), ou encore définir des balises pour des tâches particulières. Cela permet au modèle de comprendre le contexte ou le type de contenu qu'il doit générer.

Cette approche garantit la reproductibilité et la cohérence du processus de tokenisation, ce qui est essentiel pour l'entraînement et l'inférence des modèles de traitement automatique du langage.

Cette approche permet aux modèles d'apprentissage automatique de traiter les données textuelles sous une forme numérique, facilitant ainsi leur compréhension et leur analyse par le LLM.

Les tokens jouent donc un rôle crucial tout au long du processus, de l'entrée à la sortie, en passant par le traitement interne du modèle. Ils permettent de transformer le langage naturel en une forme que l'IA peut traiter efficacement, tout en conservant le sens et le contexte du texte original.

### 3.2.1. Base de données vectorielle

Les bases de données vectorielles sont utilisées pour **stocker et rechercher des données** sous forme de **vecteurs**. Chaque entrée dans la base est représentée par un vecteur dans un **espace multidimensionnel**. Cela permet d'effectuer des recherches rapides basées sur la **similarité entre vecteurs**.

Par exemple, dans le contexte du NLP, chaque mot ou phrase peut être transformé en un vecteur à l'aide de techniques telles que [Word2Vec](#), [BERT](#) ou d'embedding via [des modèles spécialisés](#). Ces représentations vectorielles permettent ensuite d'évaluer la proximité sémantique entre différentes entrées.

### 3.2.2. Comparaison vectorielle : *similarité cosinus*

La similarité cosinus est une méthode couramment utilisée pour mesurer la similarité entre deux vecteurs dans un espace vectoriel. Elle est définie par la formule suivante :

$$\text{similarité cosinus} = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \cdot \|\mathbf{B}\|}$$

Où :

- $\mathbf{A} \cdot \mathbf{B}$  est le produit scalaire des vecteurs A et B.

$$\mathbf{A} \cdot \mathbf{B} = \sum_{i=1}^n A_i \cdot B_i$$

Où  $A_i$  et  $B_i$  sont les  $i$ -èmes éléments des vecteurs A et B.

*Exemple : Si A = [4, 2, 3] et B = [3, 5, 8] :*

$$\mathbf{A} \cdot \mathbf{B} = (4 \cdot 3) + (2 \cdot 5) + (3 \cdot 8) = 12 + 10 + 24 = 46$$

- $\|A\|$  et  $\|B\|$  sont les normes (ou longueurs) des vecteurs.

$$\|A\| = \sqrt{\sum_{i=1}^n A_i^2}$$

Exemple : Pour  $A = [4, 2, 3]$  et  $B = [3, 5, 8]$

Norme A :

$$\|A\| = \sqrt{4^2 + 2^2 + 3^2} = \sqrt{16 + 4 + 9} = \sqrt{29}$$

Norme B :

$$\|B\| = \sqrt{3^2 + 5^2 + 8^2} = \sqrt{9 + 25 + 64} = \sqrt{98}$$

Grâce à ces deux élément, on peut enfin calculer la similarité cosinus de A et B :

$$\text{similarité}(A, B) = \frac{46}{\sqrt{29} \cdot \sqrt{98}} = \frac{46}{5.385 \cdot 9.899} = \frac{46}{53.291} \approx 0.863$$

$$\text{similarité}(A, B) \approx 0.863$$

Le résultat de la **similarité cosinus (0.863)** indique que les vecteurs **A et B sont fortement similaires** (très proche de 1), ce qui suggère qu'ils pointent dans des directions proches dans l'espace vectoriel, bien qu'ils ne soient pas identiques (auquel cas le résultat aurait été 1).

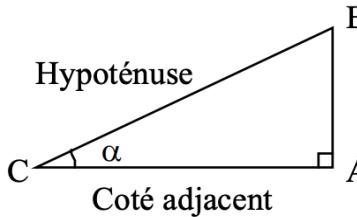
Cette mesure varie entre -1 et 1, où 1 signifie que les vecteurs sont **identiques**, 0 indique qu'ils sont **orthogonaux (sans similitude)**, et -1 signifie qu'ils sont **opposés**. La similarité cosinus est particulièrement utile dans le domaine du traitement du langage naturel pour comparer des documents ou des phrases.

Finalement, pour connaître la similarité cosinus entre deux vecteurs, il suffit de calculer le cosinus de l'angle formé par ces vecteurs. Ce calcul est en réalité un concept très simple en trigonométrie.

Le cosinus, une fonction qui exprime le rapport entre la longueur du côté adjacent à un angle donné et la longueur de l'hypoténuse dans un triangle rectangle. Cela peut être rappelé grâce au moyen mémotechnique :

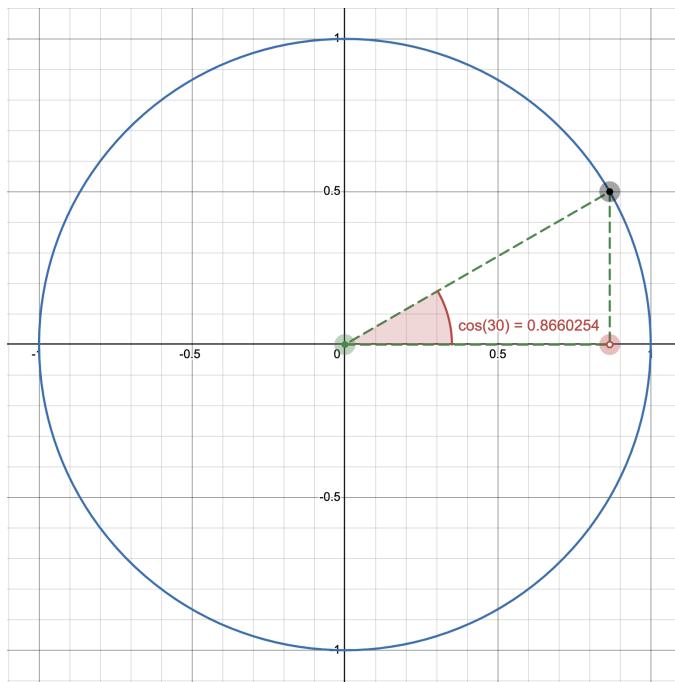
**CAH SOH TOA**

**CAH : Cosinus = Adjacent / Hypoténuse**



$$\cos(\alpha) = \frac{\text{côté adjacent}}{\text{hypoténuse}}$$

Dans le contexte des vecteurs, ce calcul se traduit par le rapport entre le produit scalaire des deux vecteurs (équivalent à la projection d'un vecteur sur l'autre) et le produit de leurs normes (ou longueurs) comme expliqué précédemment. Qu'on pourrait aussi représenter comme tel dans un cercle trigonométrique :



Représentation en 2D de la similarité cosinus/cosinus de l'angle entre les vecteurs

On peut donc constater avec cette représentation graphique que vous pouvez retrouver [ici](#) que l'angle entre les deux vecteur de l'exemple précédent dans un monde en deux dimensions serait d'environ 30°.

Ce calcul, bien que simple, est extrêmement puissant, car il fournit une mesure quantitative de l'alignement entre deux vecteurs. Cette simplicité explique pourquoi la similarité cosinus est si populaire dans des domaines comme le traitement du langage naturel et la recherche

d'informations.

En résumé, ces concepts mathématiques interconnectés permettent de comprendre comment les réseaux de neurones fonctionnent, comment les données textuelles sont préparées pour l'analyse, et comment les similarités sémantiques peuvent être mesurées efficacement.

### **3.3. Le fonctionnement du RAG**

Le RAG, dans le traitement, présente des similitude avec le fonctionnement d'un réseau de neurone, notamment dans :

- La représentation vectorielle de l'information
- Le découpage de l'information initiale en bloc
- Le traitement parallèle des données
- L'utilisation de mécanismes de pondération pour évaluer la pertinence
- La sélection des éléments les plus importants basée sur des scores (ici les vecteurs)
- La génération d'une sortie en fonction des entrées sélectionnées, afin de fournir une réponse basée sur les informations pertinentes récupérées

Ces similitudes reflètent des principes communs de traitement de l'information, bien que les deux systèmes aient des architectures et des objectifs distincts.

Voici, dans l'ordre, les étapes de réalisation du RAG :

**1. Conversion du document en input :**

- Le document PDF est converti en texte brut. Il est également possible d'utiliser des fichiers texte, Markdown...

**2. Chunking (fragmentation) :**

- Le texte extrait est divisé en fragments plus petits (chunks) pour faciliter le traitement.

**3. Création des embeddings<sup>1</sup> :**

- Chaque chunk de texte est converti en un vecteur numérique (embedding) de 100 à 1000 dimensions.

---

<sup>1</sup> Un *embedding* est une représentation vectorielle numérique d'un élément (comme un mot, une phrase, une image ou une vidéo) dans un espace multidimensionnel.

### **3 - Retrieval-Augmented Generation (RAG)**

- Ces vecteurs représentent le contenu sémantique du texte.

#### **4. Indexation :**

- Les embeddings sont stockés dans une base de données vectorielle pour une recherche rapide.

#### **5. Traitement de la requête :**

- La requête de l'utilisateur est également convertie en un vecteur d'embedding.

#### **6. Comparaison vectorielle :**

- On calcule la similarité cosinus entre le vecteur de la requête et chaque vecteur de chunk.
- La similarité cosinus donne un score entre -1 et 1, où 1 indique une similarité parfaite.

#### **7. Sélection des chunks pertinents :**

- Les chunks avec les scores de similarité les plus élevés sont sélectionnés. Ces chunks sont considérés comme les plus pertinents pour la requête.

#### **8. Génération de la réponse :**

- Les chunks sélectionnés sont fournis comme contexte à un modèle de langage.
- Le modèle génère une réponse en s'appuyant sur ce contexte et la requête initiale.

Pour illustrer le concept du RAG, nous avons utilisé l'exemple de fichiers PDF, contenant du texte (Attention certains PDF peuvent contenir des images et non du texte lisible). Cependant, il est important de noter que les systèmes d'IA génératifs actuels appliquent le processus du RAG principalement pour traiter et exploiter les informations issues des pages web et d'autres sources en ligne comme indiqué précédemment dans la partie [Concepts](#).

Contrairement aux simples fichiers PDF, ces sources représentent de grandes masses de données (Big Data), permettant ainsi aux systèmes d'IA d'accéder à des données vastes et constamment mises à jour, enrichissant ainsi leurs capacités de génération de réponses pertinentes et actualisées.

### 3.4. Exemple concret

Nous avons décidé de réaliser un exemple concret de mise en pratique du RAG sur la base du travail de [pixegami](#) un youtubeur connu pour son travail sur Python et sur les IA. Nous nous inspirerons du code présenté dans sa vidéo intitulé [Python RAG Tutorial \(with Local LLMs\): AI For Your PDFs](#) cependant nous irons plus loin et nous expliquerons en détail le fonctionnement du code et nous l'implémenterons dans un petit serveur Django afin de proposer une démo !

Vous pouvez trouver tout notre travail sur ce [repo GitHub](#).

#### 3.4.1. Fonctionnement

Ci-dessous, se trouve un schéma illustrant un des possibles fonctionnements du RAG, en l'occurrence celui que nous allons mettre en œuvre plus tard. Ce schéma se concentre uniquement sur le principe du RAG, sans entrer dans les détails de l'application Django qui l'entoure.

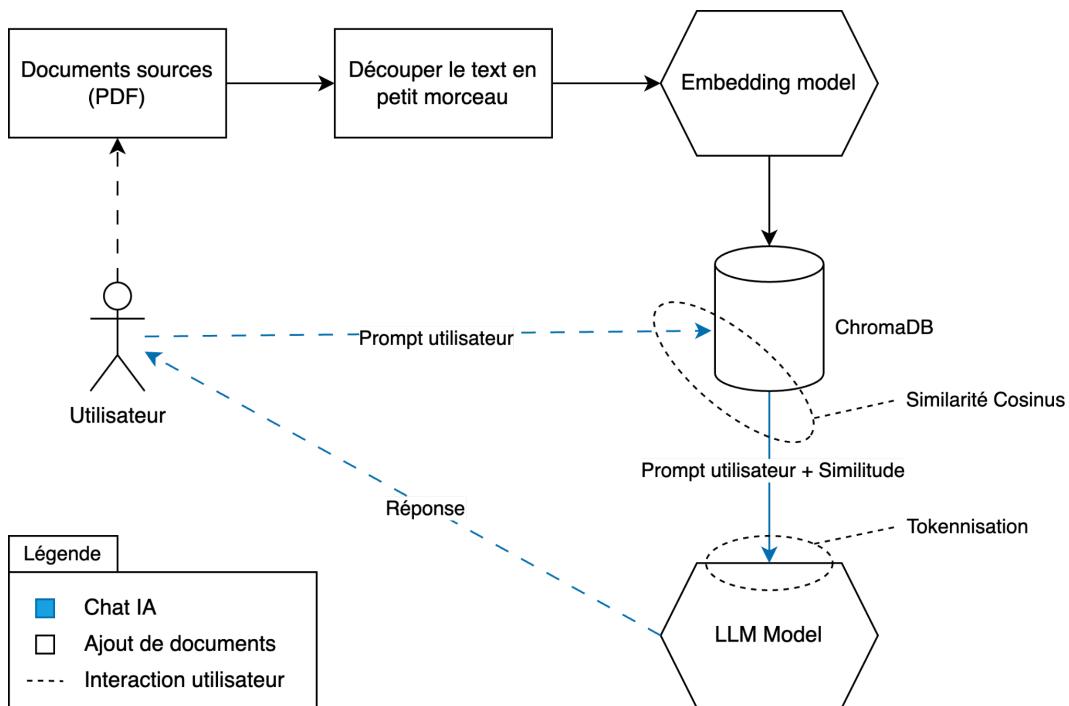


Schéma représentant le fonctionnement du RAG dans notre démo.

Et similaire au fonctionnement du RAG en général.

On observe sur ce schéma deux chemins distincts :

- L'ajout de documents, représenté en noir.
- Le processus de discussion (chat), représenté en bleu et les pointillés pour symboliser les interactions

de l'utilisateur.

#### **3.4.1.1. Étapes du fonctionnement de l'ajout des documents**

##### **1. Ajout de documents :**

Pour être pertinent, le RAG a besoin d'une source de fichiers originaux. Il est donc nécessaire d'ajouter des documents sources. Dans notre démonstration, il s'agit de fichiers PDF. (Notez qu'il existe différents types de PDF : ceux construits à partir d'images ne pourront pas être utilisés directement dans notre démonstration, à moins d'être préalablement traités par un logiciel OCR).

##### **2. Prétraitement des documents :**

Le texte des documents est extrait, puis divisé en chunks ou morceaux à l'aide d'une bibliothèque spécialisée.

##### **3. Fourniture des chunks au modèle :**

Tous ces chunks sont ensuite fournis à un modèle spécialisé dans l'embedding, qui joue un rôle central dans le fonctionnement du RAG.

---

#### **Mais qu'est-ce que l'embedding ?**

L'embedding est une technique qui transforme les données (ici, les morceaux de texte ou chunks) en vecteurs numériques dans un espace de dimensions fixes. Ces vecteurs, appelés embeddings, capturent les informations sémantiques du texte de manière compacte, permettant au système d'effectuer des calculs mathématiques pour établir des relations entre les données.

#### **Exemple :**

##### **Prenons le document suivant :**

“Django permet de créer des applications web et il est rapide et efficace.”

##### **1. Découpage en chunks :**

Le texte est divisé en petits morceaux (chunks), par exemple :

- Chunk 1 : “Django permet de créer des applications web.”
- Chunk 2 : “et il est rapide et efficace.”

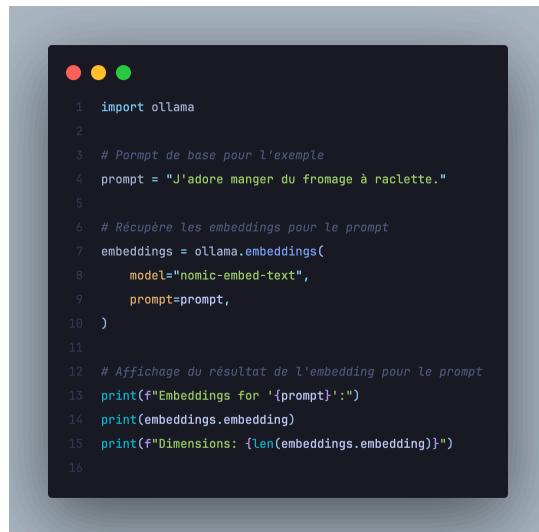
## 2. Transformation en vecteurs

Chaque chunk est transformé en un vecteur numérique à l'aide du modèle d'embedding. À noter que ces modèles sont entraînés pour retourner des vecteurs d'un nombre de dimensions prédéfini. Dans notre exemple, ce sera 768.

Par exemple :

- Chunk 1 → [0.8, 0.4, -0.6, ...]
- Chunk 2 → [0.5, 0.7, -0.2, ...]

### Exemple de code :



```
 1 import ollama
 2
 3 # Prompt de base pour l'exemple
 4 prompt = "J'adore manger du fromage à raclette."
 5
 6 # Récupère les embeddings pour le prompt
 7 embeddings = ollama.embeddings(
 8     model="nomic-embed-text",
 9     prompt=prompt,
10 )
11
12 # Affichage du résultatat de l'embedding pour le prompt
13 print(f"Embeddings for '{prompt}':")
14 print(embeddings.embedding)
15 print(f"Dimensions: {len(embeddings.embedding)}")
16
```

*Code de test/démo embedding*



### **3.4.1.2. Étapes du fonctionnement du chat avec l'IA**

#### **1. Envoie du prompt par l'utilisateur**

L'utilisateur soumet une requête ou une question (prompt).

#### **2. Similarité cosinus au sein de la base de données vectoriel**

Le système de la base de données transforme le prompt en un vecteur numérique. Le vecteur du prompt est comparé aux vecteurs des chunks déjà indexés dans la base de données vectorielle, telle que ChromaDB, en utilisant une mesure de similarité cosinus. Cela permet d'identifier les chunks les plus pertinents.

#### **3. Récupération du prompt et des fragments pertinents**

Les fragments de texte (chunks) les plus proches du prompt, selon la similarité cosinus, sont récupérés avec leur contexte d'origine.

#### **4. Construction d'un contexte enrichi**

Le prompt original de l'utilisateur est enrichi avec les fragments de texte pertinents, créant un contexte adapté à la question posée.

#### **5. Envoie au LLM**

Le prompt enrichi est envoyé à un grand modèle de langage (LLM), comme GPT, pour générer une réponse.

#### **6. Récupération de la réponse**

La réponse générée par le LLM est renvoyée à l'utilisateur via l'interface. Le système peut inclure les sources des fragments pour garantir la traçabilité et la transparence.

### 3.4.2. Demo

Nous allons exécuter la démonstration que nous avons réalisée, disponible dans le dossier `/server` de ce [dépôt GitHub](#). Cette démo illustre un exemple de RAG (Retrieval-Augmented Generation) construit à partir du travail de [pixegami](#), avec une application Django conçue pour le rendre plus accessible et utilisable.

#### Pré-requis

Avant de commencer, il sera nécessaire de :

1. Installer **Python** ainsi que les dépendances requises (listées dans le fichier `requirements.txt` ou équivalent).
2. Installer **Ollama**, qui devra être en fonctionnement sur la machine.

#### Lancement de l'application

Une fois ces pré-requis satisfaits, lancez l'application comme indiqué dans le fichier `README`. Après démarrage, voici l'interface obtenue :

### Chat avec IA

*En fonction de la puissance de calcul de votre machine certaines requêtes peuvent prendre du temps.*

Entrez votre requête :

Envoyer

### Gestion des Documents

[Rafraîchir les Documents](#)

Aucun document trouvé.

### Uploader un Fichier vers le RAG

*Attention certains pdf pourront ne pas fonctionner. (parfois des PDF sont construits à partir d'image, et cette démo ne supporte pas ce genre de pdf)*

Choisir des fichiers :

Sélect. fichiers | Aucun fichier choisi

Uploader

*Interface Django de la démonstration*

### 3 - Retrieval-Augmented Generation (RAG)

L'interface proposée est simple et explicite.

Les éléments dans leur ordre d'apparition :

- Un premier **disclaimer** en italique informe que, sur certaines machines, les requêtes peuvent être relativement longues à traiter.
- La zone pour écrire votre requête.
- Bouton d'envoi de la requête.
- Bouton pour rafraîchir la liste de documents inclus.
- Liste des documents inclus.
- Un second **disclaimer** en rouge concerne les fichiers PDF : comme mentionné précédemment, seuls les PDF contenant du texte exploitable sont pris en charge par cette application.
- Formulaire d'upload des documents.
- Bouton d'upload des documents.

**Vous pouvez donc uploader des fichiers par exemple le rapport d'alternance d'Enzo Fournet :**

#### Uploader un Fichier vers le RAG

Attention certains pdf pourraient ne pas fonctionner. (parfois des PDF sont construits à partir d'image, et cette démo ne supporte pas ce genre de pdf)

Choisir des fichiers :

Rapport d'alternance.pdf

#### Uploader un Fichier vers le RAG

Attention certains pdf pourraient ne pas fonctionner. (parfois des PDF sont construits à partir d'image, et cette démo ne supporte pas ce genre de pdf)

Choisir des fichiers :

Rapport d'alternance.pdf

Files added successfully

*Upload de mon rapport d'alternance*

#### Gestion des Documents

Document : rapport-dalternance.pdf

*Rapport uploadé*

### **3 - Retrieval-Augmented Generation (RAG)**

Nous savons donc que dans ce rapport, il a été rédigé une partie qui explique ce qu'Enzo pense de l'IA dans l'apprentissage, nous pouvons donc essayer de demander à notre App ce qu'Enzo pense de l'IA dans l'apprentissage (des petits points de chargement apparaîtront puis le texte morceau par morceau).

Entrez votre requête :

Qu'est-ce qu'Enzo Fournet pense de l'IA dans l'apprentissage ?

1

**Envoyer**

D'après le contexte, il semble que Enzo Fournet pense que l'intelligence artificielle (IA) a un impact profondément transformateur sur l'apprentissage. Il mentionne qu'elle permet aux apprentis de débuter leur formation à un niveau supérieur et d'accélérer le processus d'apprentissage, en permettant ainsi la réalisation de tâches plus complexes dès le début. Cependant, il souligne également que cette avancée ne vient pas sans ses défis, notamment la nécessité de développer progressivement une compréhension approfondie des bases.

Sources : rapport-dlaternance.pdf

*Réponse de l'application*

### **3 - Retrieval-Augmented Generation (RAG)**

On se rend donc compte que l'application répond correctement en utilisant le contexte du document comme l'indique la source et si l'on consulte les logs de l'application, on peut aussi retrouver le prompt complet envoyé au modèle :

Prompt:

Human:

Answer the question based only on the following context:

20/04/2024 1

(....)

---

BRUEL. Cette expérience démontre que l'IA peut effectivement augmenter l'efficacité de l'apprentissage, en permettant aux apprentis de réaliser des tâches plus complexes plus tôt dans leur formation, tout en développant progressivement une compréhension approfondie.

---

lorsqu'ils traitent des données fournies dans le cadre de discussions. Comme discuté avec Guillaume PIESSAT, notre responsable qualité, il est crucial d'éviter le copier-coller de code sous licence privée dans ces plateformes pour prévenir toute violation de droits d'auteur ou de propriété intellectuel.

Il est également important de souligner, comme nous l'avons évoqué avec Guillaume lors de nos discussions, que l'utilisation du code généré par ces modèles peut s'avérer dangereuse. En effet, les modèles peuvent avoir intégré du code issu de sources privées qui ne devraient pas être divulguées ou utilisées par d'autres entités. Nous devons donc faire preuve de prudence en gérant les informations dans les deux sens : non seulement dans ce que nous fournissons aux IA

---

Answer the question based on the above context: Qu'est-ce qu'Enzo Fournet pense de l'IA dans l'apprentissage ?

*Prompt envoyé au modèle*

### 3.5. Avantages

Le RAG offre plusieurs avantages significatifs :

- **Précision accrue** : en s'appuyant sur des sources externes vérifiables, le RAG améliore considérablement l'exactitude des réponses générées.
- **Informations à jour** : le système peut accéder à des données récentes, dépassant les limites temporelles de l'entraînement initial du LLM.
- **Réduction des hallucinations** : en se basant sur des faits externes, le RAG diminue le risque de générer des informations erronées ou "hallucinées".
- **Flexibilité et adaptabilité** : il permet d'adapter rapidement les modèles à des domaines spécifiques sans nécessiter un réentraînement complet, qui prendrait potentiellement plus de beaucoup de temps.
- **Transparence et confiance** : les sources utilisées peuvent être citées, permettant aux utilisateurs de vérifier les informations, permettant de vérifier la fiabilité des informations utilisées.

### 3.6. Inconvénient

Malgré ses nombreux avantages, le RAG présente quelques limitations :

- **Complexité accrue** : l'implémentation d'un système RAG est plus complexe qu'un simple LLM, nécessitant une infrastructure de recherche et de stockage supplémentaire.
- **Dépendance aux sources externes** : la qualité des réponses dépend fortement de la qualité et de la pertinence des sources d'information utilisées.
- **Latence potentielle** : le processus de recherche et de récupération d'informations peut augmenter le temps de réponse du système.

### 3.7. Applications et cas d'usages de la RAG

Les assistants virtuels améliorés, ou chatbot, utilisant la RAG peuvent fournir des **réponses plus précises et contextuelles** aux requêtes des utilisateurs. Ils peuvent accéder à des informations **spécifiques à l'entreprise ou au domaine** pour offrir une assistance personnalisée.

Ils interviennent dans différents types de situation et différents domaines :

### **Assistants virtuels et chatbots améliorés**

- **Service client** : utilisation de la RAG pour programmer des chatbots capables de répondre aux demandes des clients en accédant à des informations spécifiques, ce qui réduit le temps de résolution et améliore l'efficacité.
- **Assistance technique** : amélioration des systèmes d'assistance technique en accédant à la documentation interne pour fournir des réponses précises aux problèmes techniques (Exemple : développeur...).

### **Analyse et traitement de documents**

- **Extraction d'informations** : utilisation du RAG pour extraire des informations pertinentes de documents variés tels que des rapports annuels, documents marketing, avis clients, etc.
- **Recherche médicale** : aide à la recherche de documents pertinents dans les bases de données médicales, facilitant l'accès à des informations cruciales pour les chercheurs et praticiens.

### **Systèmes d'information spécialisés**

- **Analyse financière** : application de la RAG pour analyser des rapports financiers en combinant les capacités des modèles avec des données financières spécifiques et actualisées.
- **Recherche et développement** : utilisation dans la découverte de gaz et de pétrole en combinant les connaissances du modèle avec des données géologiques spécifiques.

### **Traitement des interactions client**

- **Analyse des interactions** : examen des transcriptions des échanges avec les clients dans les centres d'appels, permettant une analyse approfondie et personnalisée des interactions.

### **Systèmes de recommandation personnalisés**

- **Recommandations sur mesure** : développement de systèmes capables de fournir des recommandations personnalisées basées sur les préférences et le comportement des utilisateurs.

## Conclusion

En conclusion, ce mémoire s'est penché sur l'évolution des intelligences artificielles génératives et leurs récentes avancées, avec une attention particulière portée à la génération augmentée par récupération (RAG). Les modèles génératifs, tels que les Large Language Models (LLM), ont montré des capacités remarquables dans la création de contenu textuel. Cependant, ils rencontrent des difficultés notables, notamment en matière de mise à jour des connaissances et de biais liés aux données d'entraînement.

Le RAG émerge comme une innovation clé pour surmonter ces obstacles. En intégrant un mécanisme de récupération d'informations en temps réel, le RAG enrichit les modèles génératifs avec des données actuelles et contextuelles. Cela permet de produire des réponses plus précises et pertinentes, tout en atténuant les hallucinations souvent observées dans les modèles d'IA. Cette approche tire parti des atouts des systèmes de recherche d'information et de la créativité des modèles génératifs, ouvrant ainsi de nouvelles perspectives pour l'automatisation et l'optimisation de tâches complexes.

Les applications potentielles du RAG sont nombreuses, allant de l'amélioration des assistants virtuels à l'élaboration de systèmes d'analyse documentaire avancés. Toutefois, sa mise en œuvre requiert une infrastructure technique sophistiquée et pose des défis concernant la qualité et la pertinence des sources d'information utilisées, dû à un grand nombre de fausses informations comme les fake news sur Internet.

À l'avenir, le développement du RAG pourrait bénéficier d'une recherche approfondie visant à optimiser les algorithmes de récupération et à intégrer de nouvelles sources de données. En parallèle, l'exploration de nouvelles architectures de modèles pourrait améliorer davantage les capacités de génération et d'adaptation contextuelle. Ces progrès pourraient transformer radicalement l'utilisation des systèmes d'IA dans des secteurs tels que la finance, la santé ou l'éducation, en proposant des solutions toujours à jour et efficace.