



**ECOLE MAROCAINE DES
SCIENCES DE L'INGENIEUR**
Membre de
HONORIS UNITED UNIVERSITIES

RAPPORT D'ÉTUDE DE NOTRE PROJET
Ingénieur Informatique et Réseau

L'Ecole Marocaine des Sciences de l'ingénieur
Département : Informatique
Filière : IIR

**Développement d'un Chatbot Avancé de
Questions-Réponses sur Documents**
**Utilisant des Modèles de Langage de Grande
Taille Basés sur RAG**

Réalisé par :
BELAHRECH Abderrahmane
Dalil Ali
FATHALLAH Oussama
BENLAMRABET Abdelwadoud

Encadré par :
Mr. Aissam Outchakoucht

Ce travail est dédié à nos parents

Remerciements

Le présent travail est le fruit d'une collaboration précieuse et du soutien de nombreuses personnes auxquelles nous souhaitons exprimer notre profonde gratitude.

Nous tenons à remercier chaleureusement notre école **EMSI** pour nous avoir offert l'opportunité de réaliser ce projet et d'approfondir nos connaissances dans le cadre de nos études. Nous souhaitons également exprimer notre reconnaissance à **M. Aissam Outchakoucht**, notre professeur de matière, pour son travail acharné et sa capacité à rendre des concepts complexes, notamment dans le domaine de l'intelligence artificielle (IA), accessibles et simples à comprendre.

Enfin, nous remercions toutes les personnes dont les conseils et les enseignements ont été déterminants dans la réalisation de ce travail collectif.

Abstract

Natural Language Processing (NLP) and deep learning models play a crucial role in the development of intelligent applications, particularly in the realm of question-answering chatbots. This project focuses on the development of an advanced chatbot system capable of processing large documents and providing precise answers to user queries. The primary goal is to leverage the capabilities of NLP models to efficiently extract relevant information from documents and deliver contextually appropriate responses.

A key feature of this system is its ability to indicate the source of the information within the document, ensuring greater transparency and reliability in the generation of answers. This functionality enhances the user experience by allowing users to quickly access information while maintaining a clear reference to its origin in the document.

By building on recent advancements in NLP and deep learning, this project explores the challenges of document-based information retrieval and highlights innovative solutions that make these systems more accurate and tailored to user needs.

Résumé

Le traitement automatique du langage naturel (NLP) et les modèles de deep learning jouent aujourd’hui un rôle majeur dans la conception d’applications intelligentes, en particulier dans le domaine des **Documents Q&A Bot** (chatbot de questions-réponses) chatbots de questions-réponses. Ce projet se concentre sur le développement d’un système avancé de chatbot capable de traiter des documents volumineux et de fournir des réponses précises aux questions des utilisateurs. L’objectif principal est d’exploiter les capacités des modèles de NLP pour extraire efficacement les informations pertinentes à partir de documents et les restituer sous forme de réponses adaptées au contexte.

Une caractéristique notable de ce projet est que le chatbot indique également la source des informations dans le document, garantissant ainsi une transparence et une fiabilité accrues dans la génération des réponses. Cette fonctionnalité est essentielle pour améliorer l’expérience utilisateur, en permettant aux utilisateurs d’accéder rapidement aux informations tout en conservant la trace de leur origine dans le document.

En s’appuyant sur les avancées récentes en matière de NLP et de deep learning, ce projet explore les défis liés à la récupération d’informations à partir de documents et met en avant les solutions innovantes qui permettent de rendre ces systèmes plus précis et adaptés aux besoins des utilisateurs.

Sommaire

Liste des Figures	vii
Liste des Abréviations	ix
Introduction Générale	1
1 État de l'art	2
1.1 Introduction	3
1.2 Intelligence Artificielle	4
1.3 Machine learning	5
1.3.1 Type de ML	5
1.3.2 Algorithme d'apprentissage automatique	6
1.4 Deep learning	6
1.4.1 Réseaux neurones	7
1.4.2 Le modèle du perceptron	8
1.4.3 MLP	8
1.5 Qu'est-ce que la NLP appliquée ?	9
1.5.1 Cas d'utilisation	12
1.6 Modèles de langage en traitement automatique du langage naturel (NLP)	13
1.6.1 Qu'est-ce qu'un modèle en apprentissage automatique ?	13
1.6.2 Qu'est-ce qu'un modèle de langage ?	14
1.6.3 Embeddings en Traitement du Langage Naturel	14
1.6.3.1 Pourquoi Utiliser des Embeddings ?	14
1.6.3.2 Types d'Embeddings Utilisés dans ce Projet	15
1.6.3.3 Rôle des Embeddings dans le Système Q&A	15
1.6.4 LLMs (Large Language Models)	16
1.6.5 Modèles Transformer	16
1.6.5.1 Explication du Modèle Transformer	16
1.6.5.2 Composants du Modèle Transformer	17
1.6.5.3 Fonctionnement Global	19
1.6.5.4 Avantages des Transformers	19

1.6.5.5	Perspectives	19
1.6.6	Entraîner un modèle de langage	20
1.6.7	Modèles de langage généraux par rapport à ceux spécifiques à un domaine	20
1.6.8	Que peuvent faire les modèles de langage ?	21
1.6.9	Fine-tuning a language model	22
1.6.10	Où trouver des modèles ?	23
1.6.11	Comment gérer un langage spécifique à un domaine ?	23
1.6.12	Choisir les bons modèles	24
1.7	Conclusion	25
2	Conception et outils de développement	27
2.1	Introduction	27
2.2	Conception de l'application	28
2.3	Outils de Développement Utilisés	28
2.3.1	LangChain	28
2.3.2	FAISS (Facebook AI Similarity Search)	29
2.3.3	Hugging Face Transformers	29
2.3.4	Streamlit	31
2.3.5	Docker	31
2.3.6	Visual Studio Code	32
2.4	Architecture de l'application	33
2.5	Conclusion	34
3	Phase de Réalisation	35
3.1	Introduction	35
3.2	Démarrage de l'Application	35
3.3	Téléchargement de Document	36
3.4	Processus de Génération de Réponses et Source des Informations	37
3.5	Conclusion	38
Conclusion Générale		39
Bibliography		40

Liste des Figures

1.1	Chema qui montre la structure d'un RNA.	7
1.2	Schéma qui montre un perceptron simple.	8
1.3	Schéma qui montre une représentation graphique d'un MLP.	9
1.4	Pre-training and sharing models.	10
1.5	Applied NLP frameworks.	11
1.6	Représentation visuelle d'un modèle d'embedding transformant différents types de données en vecteurs pour une comparaison de similarité dans un espace multidimensionnel.	15
1.7	The trasformer-model architecture.	17
1.8	some members of the bert family.	20
1.9	fine-tuning.	22
1.10	fine-tuning a language model.	23
1.11	domain adaption with bert.	24
1.12	choice of the llm.	25
2.1	LangChain logo.	28
2.2	Hugging Face logo.	29
2.3	Exemple de comparaison de similarité sémantique entre des phrases avec le modèle sentence-transformers/all-MiniLM-L6-v2.	30
2.4	Exemple de génération de texte contextuelle et précise avec le modèle Mistral-7B-Instruct-v0.1.	31
2.5	Streamlit logo.	31
2.6	Docker logo.	32
2.7	Visual Studio Code logo.	32
2.8	Illustration détaillée de l'architecture de l'application Document Q&A Bot utilisant la méthodologie RAG.	34
3.1	Lancement de l'application.	36
3.2	Téléchargement d'un fichier PDF dans l'application Document Q&A Bot pour l'analyse et la génération de réponses basées sur le contenu du document.	36
3.3	Processus de génération de réponses basé sur les informations contenues dans les documents fournis par l'utilisateur.	37

3.4 Exemple de la source des informations extraites des documents pour générer des réponses précises et contextuelles.	37
--	----

Liste des Abréviations

NLP	Natural Language Processing.
LLM	Large Language Model.
AI	Intelligence Artificielle.
DL	Deep Learning.
ML	Machine Learning.
RAG	Retrieval-Augmented Generation.
FAISS	Facebook AI Similarity Search.
BERT	Bidirectional Encoder Representations from Transformers.
GPT	Generative Pre-trained Transformer.
RNA	Réseaux de Neurones Artificiels.
MLP	Multilayer Perceptron.
Q&A	Question and Answer.
PDF	Portable Document Format.
ERP	Enterprise Resource Planning.
SaaS	Software as a Service.

Introduction Générale

Dans un contexte où les avancées technologiques sont en perpétuelle évolution, l'intelligence artificielle (IA) et le traitement du langage naturel (NLP) sont devenus des domaines de recherche et d'application incontournables. L'essor des technologies basées sur les modèles de langage a ouvert la voie à de nouvelles possibilités en matière d'interaction homme-machine, transformant la manière dont les utilisateurs accèdent à l'information et interagissent avec les systèmes numériques.

Ce projet s'inscrit dans cette dynamique en proposant le développement d'un système innovant de questions-réponses, capable d'interagir avec les utilisateurs pour fournir des informations précises et pertinentes à partir de documents variés. En intégrant des outils tels que **LangChain**, **FAISS**, le modèle de langage **mistralai/Mistral-7B-Instruct-v0.1**, ainsi que les embeddings de **sentence-transformers/all-MiniLM-L6-v2**, ce projet explore les défis et les opportunités liés à l'application des modèles de langage de pointe dans des contextes pratiques.

L'objectif principal de ce projet est de créer un chatbot sophistiqué qui non seulement comprend les questions des utilisateurs, mais est également capable de récupérer et d'analyser des documents pour fournir des réponses contextualisées. Cette approche se distingue par son utilisation de bases de données vectorielles pour optimiser la recherche de documents et la personnalisation des réponses en fonction des besoins des utilisateurs.

Au-delà de l'aspect technique, ce projet met également en avant l'importance de l'expérience utilisateur en réduisant le temps de recherche et en permettant aux utilisateurs d'accéder rapidement aux informations pertinentes à partir de documents volumineux. De plus, le système indique clairement la source des réponses extraites des fichiers téléchargés.

1

État de l'art

Sommaire

1.1	Introduction	3
1.2	Intelligence Artificielle	4
1.3	Machine learning	5
1.3.1	Type de ML	5
1.3.2	Algorithme d'apprentissage automatique	6
1.4	Deep learning	6
1.4.1	Réseaux neurones	7
1.4.2	Le modèle du perceptron	8
1.4.3	MLP	8
1.5	Qu'est-ce que la NLP appliquée ?	9
1.5.1	Cas d'utilisation	12
1.6	Modèles de langage en traitement automatique du langage naturel (NLP)	13
1.6.1	Qu'est-ce qu'un modèle en apprentissage automatique ?	13
1.6.2	Qu'est-ce qu'un modèle de langage ?	14
1.6.3	Embeddings en Traitement du Langage Naturel	14
1.6.3.1	Pourquoi Utiliser des Embeddings ?	14
1.6.3.2	Types d'Embeddings Utilisés dans ce Projet	15
1.6.3.3	Rôle des Embeddings dans le Système Q&A	15
1.6.4	LLMs (Large Language Models)	16
1.6.5	Modèles Transformer	16
1.6.5.1	Explication du Modèle Transformer	16
1.6.5.2	Composants du Modèle Transformer	17
1.6.5.3	Fonctionnement Global	19
1.6.5.4	Avantages des Transformers	19
1.6.5.5	Perspectives	19
1.6.6	Entraîner un modèle de langage	20

1.6.7	Modèles de langage généraux par rapport à ceux spécifiques à un domaine	20
1.6.8	Que peuvent faire les modèles de langage ?	21
1.6.9	Fine-tuning a language model	22
1.6.10	Où trouver des modèles ?	23
1.6.11	Comment gérer un langage spécifique à un domaine ? . .	23
1.6.12	Choisir les bons modèles	24
1.7	Conclusion	25

1.1 Introduction

Le **Traitemet du Langage Naturel (NLP)** a fait des avancées significatives, permettant aux développeurs d'extraire des informations de grandes bases de données textuelles de manière intuitive. Cependant, en tant que développeur, comprendre comment construire un prototype d'application NLP robuste peut être un défi. Ce chapitre vise à guider les développeurs et les data scientists à travers le processus de mise en œuvre d'un système NLP. Il fournit des explications accessibles des concepts fondamentaux du NLP moderne et vous équipe des outils nécessaires pour concevoir, construire et maintenir une application de bout en bout alimentée par le NLP.

Avant de plonger dans le NLP, il est essentiel de comprendre son contexte plus large au sein de l'**intelligence artificielle (IA)**. L'IA regroupe un ensemble de techniques visant à permettre aux machines de simuler l'intelligence humaine. Parmi ces techniques, le **machine learning (ML)** joue un rôle crucial en permettant aux ordinateurs d'apprendre à partir de données sans être explicitement programmés. Le **deep learning**, un sous-domaine du ML, utilise des réseaux neuronaux artificiels pour traiter des données complexes, et il a été particulièrement influent dans les progrès du NLP moderne, notamment grâce aux **modèles de langage de grande taille (LLM)**. Ces LLMs, tels que **GPT-3** et **BERT**, ont révolutionné le domaine en offrant des capacités inégalées pour comprendre et générer du texte.

La réponse automatisée aux questions, les interfaces de recherche en langage naturel intuitives, le contrôle vocal, et bien d'autres choses encore - le NLP a le potentiel d'améliorer considérablement nos vies. Bien que des outils permettant de mettre en place un prototype NLP existent pour différents cas d'utilisation, superviser un projet qui met en œuvre un système de bout en bout peut être une tâche intimidante pour les développeurs. Ce chapitre vise à combler cette

lacune, en fournissant une perspective pratique et technique sur le processus de mise en œuvre du NLP.

Ce chapitre a pour objectif de répondre à certaines questions cruciales pour la mise en œuvre d'un système basé sur le NLP :

1. Quels sont les éléments clés à prendre en compte pour comprendre l'architecture du modèle **Transformer** dans le cadre de la mise en œuvre NLP ?
2. Comment construire un prototype capable d'être mis à l'échelle pour devenir un système prêt pour la production ?
3. Comment puis-je comparer différentes architectures de pipeline et différents paramètres d'hyperparamètres afin de trouver ce qui fonctionne le mieux pour notre projet ?
4. Comment puis-je adapter un modèle de langage existant à notre cas d'utilisation spécifique ?

Ces questions fondamentales seront abordées dans cette sous-section, fournissant ainsi une base solide pour la mise en œuvre d'un système NLP. Nous nous plongerons dans les processus essentiels pour mettre en place avec succès un système NLP appliqué en production, en commençant par une brève introduction sur le sujet du NLP appliqué.

À la fin de ce chapitre, nous aurons une compréhension solide des étapes nécessaires pour mettre en œuvre un système NLP et nous serons équipé(e) des connaissances et des conseils nécessaires pour surmonter les défis rencontrés. Commençons le voyage de la construction d'applications **NLP** puissantes !

1.2 Intelligence Artificielle

L'intelligence artificielle met en œuvre un certain nombre de techniques visant à permettre aux machines d'imiter une forme d'intelligence réelle. Exemples d'applications de l'intelligence artificielle :

- **Système de recommandation**
- **Traitements d'image, son, texte...**
- **Voiture autonome**
- **Reconnaissance faciale**

1.3 Machine learning

La Machine Learning, aussi appelé apprentissage automatique en français, est une forme d'intelligence artificielle permettant aux ordinateurs d'apprendre sans avoir été programmés explicitement à cet effet. Cette technique permet de développer des programmes informatiques pouvant changer en cas d'exposition à de nouvelles données. Découvrez la définition, le fonctionnement et les secteurs d'applications de machine Learning.

1.3.1 Type de ML

On distingue trois types de Machine Learning :

- **Apprentissage supervisé** : consiste à entrer des caractéristiques et une cible comme input pour entraîner le modèle, afin de rendre l'algorithme capable, une fois entraîné, de prédire cette cible sur de nouvelles données non annotées. Dans l'apprentissage supervisé, il existe des problèmes de classification et de régression.
 - **Classification:** consiste à prédire à quelle classe appartient quelque chose (par exemple, prédire si un emprunteur fera défaut sur son prêt)
 - **Régression:** consiste à prédire une valeur numérique (par exemple, prédire le prix auquel une maison se vendra).
- **Apprentissage non supervisé:** En informatique et en intelligence artificielle, l'apprentissage non supervisé fait référence à la situation du machine learning où les données ne sont pas étiquetées. Il faut donc découvrir les structures sous-jacentes à ces données sans étiquetage.
- **Apprentissage de renforcement:** En intelligence artificielle, plus précisément en apprentissage automatique, l'apprentissage par renforcement consiste, pour un agent autonome, à apprendre les actions à prendre, à partir d'expériences, de façon à optimiser une récompense quantitative au cours du temps.

1.3.2 Algorithme d'apprentissage automatique

Il existe plusieurs types d'algorithmes d'apprentissage automatique parmi eux:

- **Régression linéaire simple:** C'est lorsqu'on a une seule variable explicative x et une seule variable indépendante...
- **Logistic regression:** La régression logistique utilise principalement une fonction logistique:
 $y = 1 / (1 + \exp(-x))$ pour modéliser une sortie binaire. Si la probabilité est supérieure à 0,5 les prédictions seront classées en classe 0. Sinon, la classe 1 sera attribuée.
- **Arbre de decision:** L'arbre de decision est un algorithme non paramétrique qui ressemble à un organigramme de questions à lesquelles on répond pour prédire la sortie .
- **Forêts aléatoires :** Composé de plusieurs arbres de décision, travaillant de manière indépendante pour résoudre un problème spécifique. L'assemblage des sorties des arbres de décision va donner une estimation globale.

1.4 Deep learning

Sous-ensemble du **Machine Learning**, le Deep Learning imite le fonctionnement du cerveau humain par des **réseaux neuronaux artificiels**. L'objectif est de créer des modèles à partir de données afin d'éclairer la prise de décision. Cette méthode peut aussi bien faire partie d'un apprentissage supervisé, semi-supervisé ou non-supervisé.

Les architectures de Deep Learning ont été appliquées à des domaines tels que :

- **La vision par ordinateur**
- **La reconnaissance vocale**
- **Le traitement du langage naturel**
- **Le filtrage des réseaux sociaux**
- **La bioinformatique**
- **La traduction automatique**

Le Deep learning concerne un nombre illimité de couches de taille limitée qu'il utilise pour extraire progressivement des caractéristiques de niveau supérieur à partir de l'entrée brute. Par exemple, dans le traitement d'images, les couches inférieures peuvent identifier les bords, tandis que les couches supérieures peuvent identifier les concepts pertinents pour un humain tel que des visages.

1.4.1 Réseaux neurones

Un réseau de neurones artificiels, ou Artificial Neural Network en anglais, est un système Informatique matérielle et / ou logiciel dont le fonctionnement est calqué sur celui des neurones du cerveau humain. Il s'agit là d'une variété de technologie Deep Learning/apprentissage profond), qui fait elle-même partie de la sous-catégorie d'intelligence artificielle du Machine Learning(apprentissage automatique). Pour introduire les réseaux de neurones, nous allons commencer par présenter un modèle composé d'un seul neurone, appelé modèle du perceptron. Celui-ci va nous permettre de mettre en évidence les mécanismes de base de tout réseau de neurones.

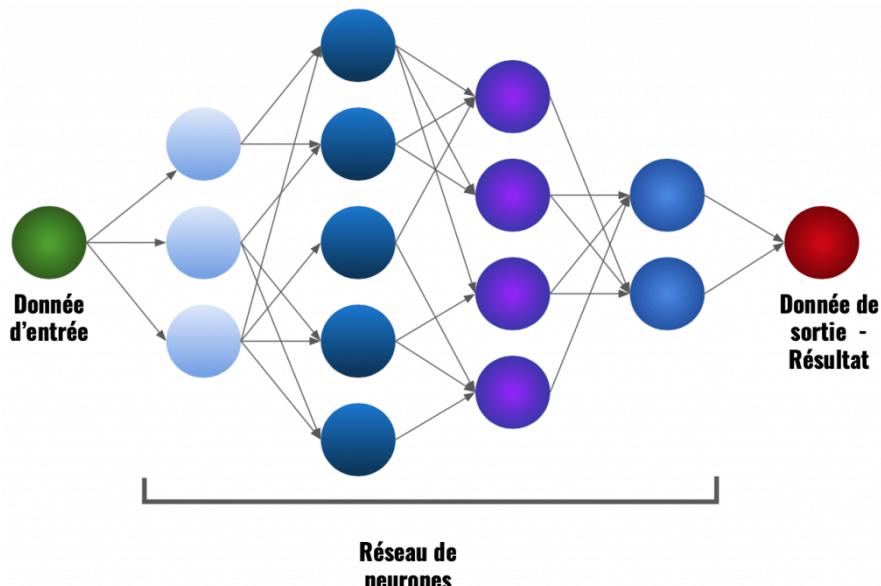


Figure 1.1: Schéma qui montre la structure d'un RNA.

1.4.2 Le modèle du perceptron

Dans sa version la plus simple, le perceptron est un réseau de neurones composé de seulement un neurone, qui prend en entrée n données binaires. Chacune de ses entrées i est pondérée par un poids noté w_i . Le neurone peut prendre les états "1" ou "0" (respectivement actif ou non actif) en fonction de ses entrées pondérées et d'un biais. Cependant ce modèle est limité car il ne peut pas résoudre des problèmes non linéairement séparables.

Équation de sortie d'un neurone formel :

$$y = f((w, x) + b) \quad (1.1)$$

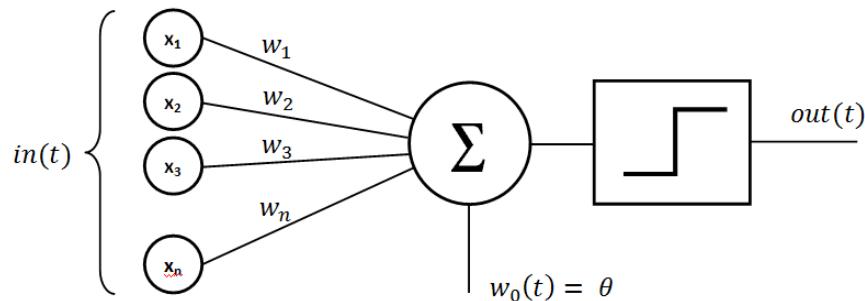


Figure 1.2: Schéma qui montre un perceptron simple.

1.4.3 MLP

Les perceptrons multicouches, appelés aussi MLP (pour Multilayer Perceptron), sont des réseaux de neurones plus généraux que le perceptron. Ils sont composés d'une multitude de neurones interconnectés et organisés en couches successives. Un MLP peut être représenté par un graphe acyclique dans lequel chaque noeud représente un neurone. Les arcs orientés représentent les relations entre chaque neurone : un arc du noeud i au noeud j signifie que le neurone j prend la valeur d'activation du neurone i en entrée. La Figure 1.3 montre une représentation graphique d'un MLP

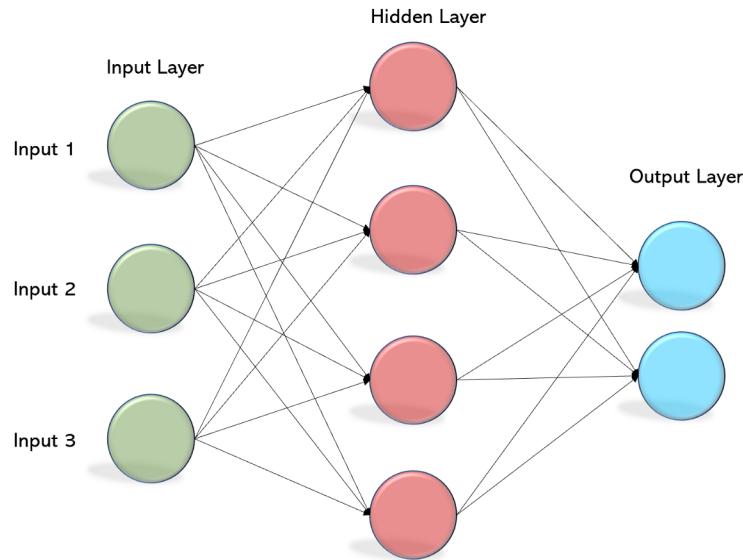


Figure 1.3: Schéma qui montre une représentation graphique d'un MLP.

1.5 Qu'est-ce que la NLP appliquée ?

La NLP appliquée se distingue de la NLP théorique en fournissant aux développeurs des outils pratiques pour exploiter les modèles linguistiques pré-entraînés dans des applications concrètes. Alors que la complexité des grands modèles linguistiques basés sur les **Transformers** peut être intimidante pour les novices en NLP théorique, elle est moins un obstacle dans le domaine de la NLP appliquée.

Il est rare que nous ayons besoin d'entraîner un modèle linguistique à partir de zéro. Grâce aux avancées dans le partage de modèles, l'accès aux **modèles pré-entraînés** est devenu plus pratique. Des plateformes centralisées comme le hub de modèles de **Hugging Face** proposent des dizaines de milliers de modèles pré-entraînés disponibles gratuitement.

Entraînement préalable et partage de modèles Les modèles Transformer sont grands et puissants, et leur entraînement nécessite de l'argent et du temps. Heureusement, la plupart des chercheurs en traitement du langage naturel pratiquent le partage de ressources pour économiser ces précieux éléments : ils téléversent les paramètres de leur modèle entraîné sur une plateforme accessible au public, telle que le hub de modèles de Hugging Face. À partir de là, tout le monde peut télécharger le modèle de langage pré-entraîné et l'utiliser directement, ou le soumettre à des passes d'entraînement supplémentaires pour l'affiner en fonction d'un cas d'utilisation spécifique.

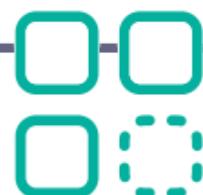


Figure 1.4: Pre-training and sharing models.

En NLP depuis l'introduction de l'architecture **Transformer** pour la modélisation du langage. Les frameworks axés sur la NLP appliquée visent donc à aider leurs utilisateurs dans les domaines suivants : Cette infrastructure est généralement fournie sous forme de **pipelines**. Les pipelines sont des architectures préconçues avec des espaces réservés pour les modèles linguistiques. Ils gèrent la logique de l'entrée d'une requête en langage naturel, la transformation en une ou plusieurs étapes, et la sortie du résultat. Le grand avantage des pipelines est qu'ils permettent aux développeurs de comparer différentes configurations, de construire et de tester rapidement des prototypes.

Frameworks de NLP appliqués

De nombreuses bibliothèques et frameworks de NLP se concentrent aujourd'hui sur la mise en œuvre de produits basés sur le NLP. En association avec les plateformes de partage de modèles, ils font partie d'un écosystème qui permet, même aux non-experts en NLP, de mettre en œuvre un système prêt pour la production avec des capacités de traitement du langage naturel alimentées par des Transformers. Des exemples de tels frameworks sont spaCy, Haystack et Transformers de Hugging Face.

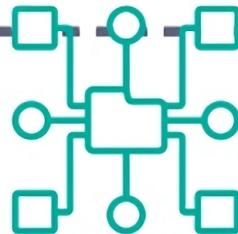


Figure 1.5: Applied NLP frameworks.

Les pipelines facilitent également la connexion à différents types de bases de données, la configuration et l'ajustement des **hyperparamètres** des modèles, ainsi que la gestion des **ressources informatiques** puissantes nécessaires lors de la manipulation de grands modèles pré-entraînés basés sur les Transformers.

Le concept de pipelines en tant qu'unités conteneurisées est tellement utile qu'ils ne sont pas seulement utilisés pour les requêtes et les inférences. En plus des pipelines de requête, la plupart des systèmes de NLP appliquée mettent également en œuvre des pipelines de **prétraitement**. Ces pipelines gèrent la transformation des données, les étapes de prétraitement telles que le nettoyage et la division du texte, ainsi que le stockage des données transformées et prétraitées dans une base de données.

En plus des différentes conceptions de pipelines pour l'inférence et le prétraitement, les frameworks de NLP appliquée offrent également des interfaces vers les différents emplacements où les **modèles pré-entraînés** sont hébergés. Ainsi, nous pouvons simplement brancher différents modèles dans notre pipeline de requête et observer les résultats pour déterminer ce qui fonctionne le mieux.

Non seulement nous pouvons utiliser des modèles linguistiques pré-entraînés tels quels, mais nous pouvons également les utiliser comme base pour **affiner** nos propres modèles. L'affinage (fine-tuning) fait référence à la tâche consistant à nourrir notre modèle avec des données supplémentaires, spécifiques à la tâche ou au domaine, afin de créer un modèle adapté à notre cas d'utilisation. C'est pourquoi la gestion de jeux de données est un autre pilier de la NLP appliquée.

Outre les bibliothèques et les frameworks open source, il existe également des **solutions gérées** pour la mise en œuvre de projets de NLP appliquée. Ces solutions offrent une approche plus accessible et conviviale pour mettre en place un système fonctionnel de bout en bout. L'interface utilisateur facilite la mise en œuvre et la comparaison de différentes configurations système, la collecte des commentaires des utilisateurs finaux dès le début du processus, ainsi que le déploiement final de l'application NLP.

1.5.1 Cas d'utilisation

Pour mieux comprendre les différentes façons dont la NLP appliquée peut bénéficier aux organisations dans la pratique, examinons les trois **cas d'utilisation** de modèles suivants. Bien que strictement hypothétiques, ces exemples sont inspirés de cas d'utilisation réels que nous avons rencontrés au fil des années.

- **Cas d'utilisation 1 : Construction d'un système de questions-réponses sur une base de données de manuels techniques:**

Un fabricant aérospatial a accumulé au fil des années des milliers de manuels pour leurs produits. Avec autant d'informations textuelles dans des silos privés, il est difficile pour les pilotes en formation de trouver rapidement les réponses à leurs problèmes (ils ne peuvent pas simplement parcourir sémantiquement l'ensemble de données à l'aide des moteurs de recherche sur Internet). Par conséquent, l'entreprise met en place une interface de **question-réponse** (QA). Leur configuration utilise un modèle de recherche sémantique provenant du hub de modèles de Hugging Face, ainsi qu'un modèle de QA adapté au domaine technique.

- **Cas d'utilisation 2 : Exploitation de modèles de traduction pour alimenter des chatbots multilingues:**

Un chatbot gère les messages d'aide sur une application avec des utilisateurs situés dans de nombreux endroits différents. Il utilise la NLP pour extraire les réponses aux questions des utilisateurs à partir d'une grande collection de documents. Étant donné que le corpus de documents lui-même est monolingue, mais que les utilisateurs peuvent poser des questions dans plusieurs langues, l'entreprise ajoute une couche de **traduction** en plus du pipeline du chatbot.

- **Cas d'utilisation 3 : Extraction d'informations à partir de rapports commerciaux:**

Une institution fédérale évalue les facteurs de risque financier des entreprises en se basant sur leurs rapports commerciaux. Ils font passer chaque rapport par un **pipeline d'extraction d'informations**, qui met en évidence les sections pertinentes à l'aide d'un modèle qui a été **affiné** pour le jargon financier. Bien que les résultats à faible confiance doivent encore être vérifiés manuellement, le système permet à leurs agents d'évaluer les entreprises individuelles beaucoup plus rapidement.

Pour mieux comprendre les différentes façons dont la NLP appliquée peut bénéficier aux organisations dans la pratique, examinons le cycle de mise en œuvre d'un système NLP dans son ensemble.

1.6 Modèles de langage en traitement automatique du langage naturel (NLP)

Les **modèles de langage** occupent une place centrale en **NLP**. Mais qu'est-ce qu'un **modèle de langage** ? Pour répondre à cette question, clarifions d'abord le terme "modèle" et son utilisation en **apprentissage automatique**.

1.6.1 Qu'est-ce qu'un modèle en apprentissage automatique ?

Le monde réel est complexe et confus. Les **modèles** servent à représenter un domaine particulier d'intérêt de manière plus simple. Par exemple, les **modèles** météorologiques sont des représentations simplifiées des phénomènes météorologiques et de leurs interactions. Ces **modèles** nous aident à mieux comprendre le domaine de la météo et à faire des prédictions à ce sujet.

En **apprentissage automatique**, les **modèles** sont assez similaires. Ils servent principalement à prédire des événements en se basant sur des données passées, c'est pourquoi on les appelle également des **modèles de prévision** ou de **prédition**.

Les données que nous alimentons à un algorithme d'**apprentissage automatique** lui permettent de créer un **modèle** du domaine de ces données. Ces données doivent représenter au mieux la réalité, de sorte que les **modèles** qui en sont basés puissent approximer le monde réel le plus fidèlement possible.

1.6.2 Qu'est-ce qu'un modèle de langage ?

Un **modèle de langage** est un **modèle d'apprentissage automatique** conçu pour représenter le domaine du langage. Il peut être utilisé comme base pour un certain nombre de tâches basées sur le langage, par exemple :

- Réponse aux questions
- Recherche sémantique
- Résumé

et bien d'autres tâches qui opèrent sur le langage naturel.

1.6.3 Embeddings en Traitement du Langage Naturel

Les embeddings jouent un rôle crucial en traitement du langage naturel (NLP), notamment dans les systèmes modernes d'apprentissage automatique. Ils permettent de représenter les mots, phrases ou documents sous forme de vecteurs numériques dans un espace de haute dimension. Ces vecteurs capturent des relations sémantiques, telles que des similitudes entre mots, facilitant ainsi la compréhension du texte par les modèles de NLP.

1.6.3.1 Pourquoi Utiliser des Embeddings ?

L'un des principaux avantages des embeddings est leur capacité à capturer les significations contextuelles des mots. Par exemple, dans un espace d'embeddings, les mots « roi » et « reine » apparaîtront proches l'un de l'autre, car ils partagent des propriétés sémantiques similaires. Cette caractéristique est particulièrement utile pour diverses tâches de NLP comme la recherche sémantique, la classification de texte, et la réponse aux questions. Dans le contexte de ton application, ces embeddings sont essentiels pour extraire les informations pertinentes à partir de documents juridiques en arabe, en assurant que les réponses fournies soient à la fois précises et contextuellement appropriées.

1.6.3.2 Types d'Embeddings Utilisés dans ce Projet

Pour ce projet, nous utilisons des embeddings fournis par HuggingFace, en particulier le modèle "*sentence-transformers/all-MiniLM-L6-v2*". Ce modèle est choisi pour sa capacité à capturer les relations complexes dans les données textuelles. Il s'agit d'un modèle léger mais performant, optimisé pour des tâches telles que la recherche sémantique et la réponse aux questions. Sa capacité à générer des vecteurs d'embeddings efficaces en fait un outil idéal pour des applications nécessitant un traitement de texte multilingue, comme dans le cas de l'analyse de documents juridiques en arabe.

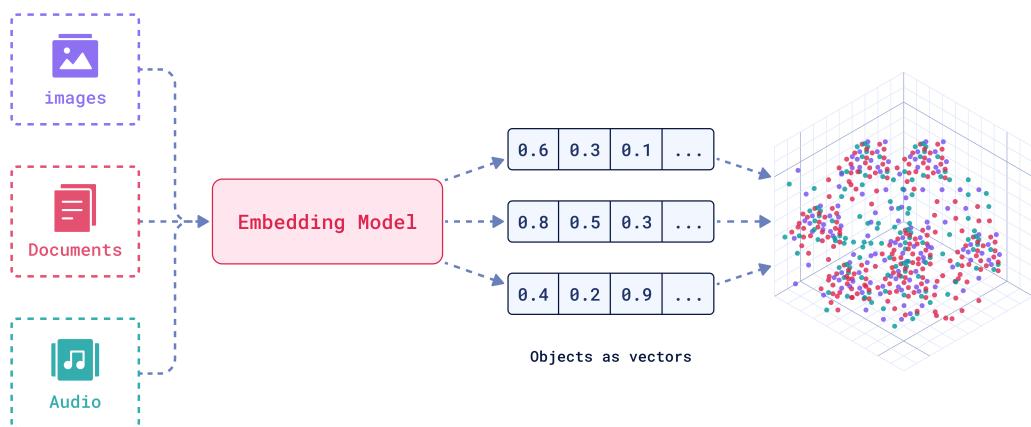


Figure 1.6: Représentation visuelle d'un modèle d'embedding transformant différents types de données en vecteurs pour une comparaison de similarité dans un espace multidimensionnel.

1.6.3.3 Rôle des Embeddings dans le Système Q&A

Dans le cadre de l'application de Document Q&A Bot, les embeddings jouent un rôle fondamental. Lorsqu'une question est posée, les embeddings sont utilisés pour représenter à la fois la question et les documents en vecteurs, permettant ainsi de mesurer la similarité entre eux. Ce processus est essentiel pour identifier les passages pertinents dans les documents qui peuvent contenir la réponse à la question posée. En utilisant des embeddings, le système peut rapidement et efficacement comparer la requête avec l'ensemble des documents indexés et fournir une réponse pertinente, même dans un contexte multilingue.

Ce processus améliore la précision du modèle en réduisant les erreurs d'interprétation et en assurant que les réponses fournies sont non seulement pertinentes, mais aussi basées sur des contextes textuels appropriés.

1.6.4 LLMs (Large Language Models)

Les modèles de langage de grande taille (LLMs) sont une classe spécifique de modèles de langage, caractérisés par leur nombre élevé de paramètres, généralement comptés en milliards. Des modèles comme GPT-3, BERT, et leurs variantes, ont démontré des performances inégalées dans des tâches variées allant de la génération de texte à la réponse aux questions, en passant par la traduction automatique et le résumé de texte.

Ces modèles exploitent l'architecture des Transformers pour capturer des relations complexes entre les mots dans de grands corpus de texte. En raison de leur capacité à généraliser sur de vastes ensembles de données, les LLMs sont devenus un outil incontournable dans le développement d'applications NLP modernes. Ils permettent d'améliorer la précision des systèmes en interprétant mieux les nuances contextuelles du langage humain.

1.6.5 Modèles Transformer

1.6.5.1 Explication du Modèle Transformer

Le modèle Transformer est une architecture de réseau neuronal introduite par Vaswani et al. en 2017 dans leur article intitulé "[Attention is All You Need](#)", principalement utilisée pour des tâches de traitement du langage naturel (NLP) comme la traduction automatique, la génération de texte, et bien d'autres applications.

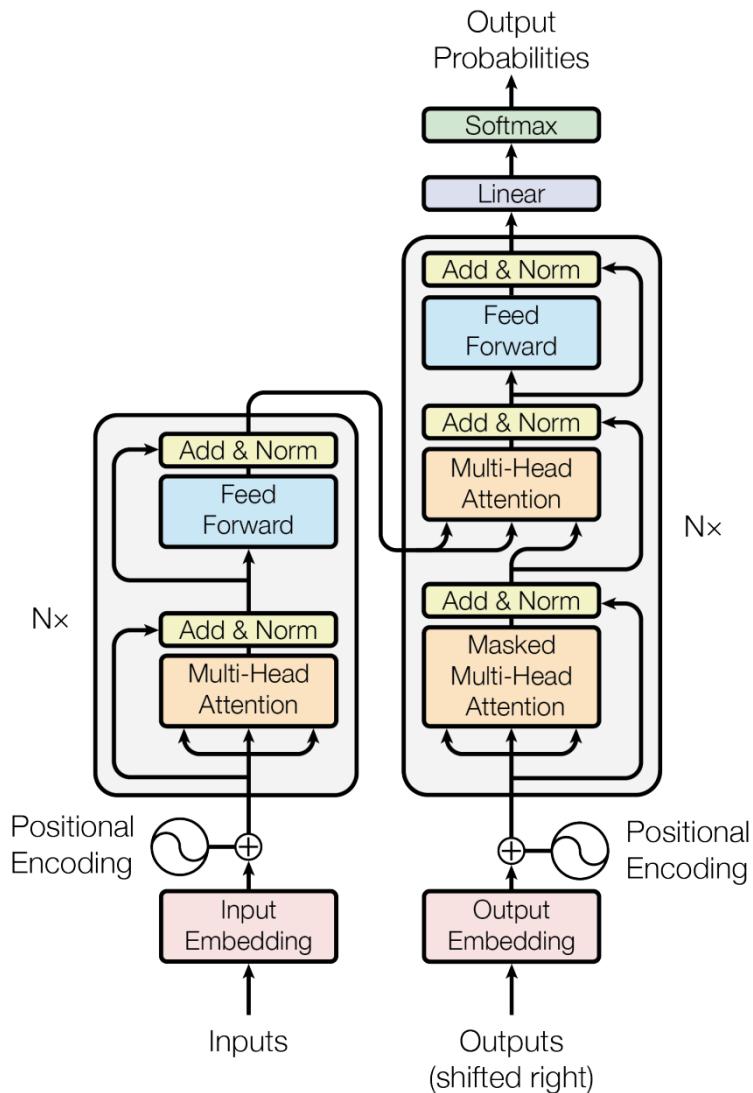


Figure 1.7: The transformer-model architecture.

1.6.5.2 Composants du Modèle Transformer

- **Encoders et Decoders :**

- Le modèle Transformer est composé d'une série d'encoders et de decoders, représentés par les blocs à gauche et à droite respectivement dans l'illustration.
- Chaque encoder est constitué de deux sous-couches : une couche d'attention multi-têtes (Multi-Head Attention) et une couche de réseau feed-forward (Feed Forward).

- Les decoders, en plus des sous-couches présentes dans les encoders, possèdent une sous-couche d'attention masquée (Masked Multi-Head Attention) pour s'assurer que chaque position dans la sortie ne puisse pas assister aux positions futures.

- **Encodage Positionnel (Positional Encoding)** :

- Comme les Transformers ne traitent pas les séquences dans un ordre particulier, une information positionnelle est ajoutée aux embeddings d'entrée pour permettre au modèle de prendre en compte la position relative des mots dans la séquence.

- **Attention Multi-têtes (Multi-Head Attention)** :

- Ce mécanisme permet au modèle de se concentrer sur différentes parties de la séquence d'entrée simultanément en utilisant plusieurs têtes d'attention. Chaque tête d'attention calcule une pondération différente pour les éléments de la séquence, permettant au modèle de capturer diverses relations contextuelles.

- **Add & Norm** :

- Après chaque sous-couche, une opération de normalisation est appliquée pour stabiliser l'entraînement. Un mécanisme de résiduel (Add) est également utilisé pour faciliter la propagation du gradient à travers les couches.

- **Feed Forward** :

- Cette couche est une simple couche de réseau de neurones entièrement connectée appliquée de manière indépendante à chaque position. Elle ajoute de la complexité non-linéaire et augmente la capacité d'expression du modèle.

- **Sortie (Output)** :

- Le module de décodage produit finalement une séquence de probabilités sur le vocabulaire possible, permettant ainsi de générer la séquence de sortie.

1.6.5.3 Fonctionnement Global

1. **Encodage** : Le texte d'entrée est d'abord converti en un format numérique à l'aide d'embeddings, et l'encodage positionnel est ajouté pour conserver les informations de séquence. Ensuite, chaque mot passe à travers les couches d'encoders qui appliquent successivement les mécanismes d'attention et les réseaux feed-forward.
2. **Décodage** : Le décodeur prend les représentations intermédiaires produites par les encoders ainsi qu'un décalage de la séquence de sortie précédente (dans les tâches de génération de séquences) et génère un mot à la fois, en tenant compte des contextes des mots précédents grâce à l'attention masquée.

1.6.5.4 Avantages des Transformers

- **Parallélisation** : Contrairement aux RNN, les Transformers peuvent traiter tous les éléments d'une séquence simultanément, permettant une parallélisation et des temps d'entraînement plus rapides.
- **Longue portée contextuelle** : Le mécanisme d'attention permet de capturer des dépendances à longue distance entre les mots, crucial pour des tâches de compréhension de texte.

Les modèles de langage de grande taille (LLM) tels que BERT et GPT ont révolutionné le domaine du traitement du langage naturel en permettant une meilleure compréhension contextuelle des textes. Ces modèles, basés sur l'architecture des Transformers, sont capables de traiter et de générer du texte avec un niveau de précision et de fluidité qui n'était pas possible avec les approches précédentes.

1.6.5.5 Perspectives

Le modèle Transformer est devenu la norme pour de nombreuses tâches de NLP grâce à sa flexibilité, son efficacité et sa capacité à capturer des relations complexes dans des séquences textuelles. Il constitue la base de nombreux modèles modernes, tels que BERT, GPT, et leurs variantes, qui ont montré des performances exceptionnelles dans une variété de tâches de traitement du langage naturel.

1.6.6 Entraîner un modèle de langage

L'état de l'art en **NLP** aujourd'hui est soutenu par de grands réseaux neuronaux. Les **modèles de langage** neuronaux tels que **BERT** apprennent quelque chose qui ressemble à l'intuition linguistique en traitant des millions de points de données. En **apprentissage automatique**, ce processus est appelé "entraînement".

Pour entraîner un **modèle**, nous devons proposer des tâches qui le poussent à apprendre une représentation d'un domaine donné. Pour la modélisation du langage, une tâche courante consiste à compléter le mot manquant dans une phrase, tout comme dans notre exemple précédent. Grâce à cette tâche et à d'autres tâches d'entraînement, un **modèle de langage** apprend à encoder les significations des mots et des passages de texte plus longs.

Alors, comment passe-t-on d'une représentation informatique des propriétés sémantiques d'une langue à un **modèle** capable d'effectuer des tâches spécifiques telles que la **réponse aux questions** ou le **résumé** ?

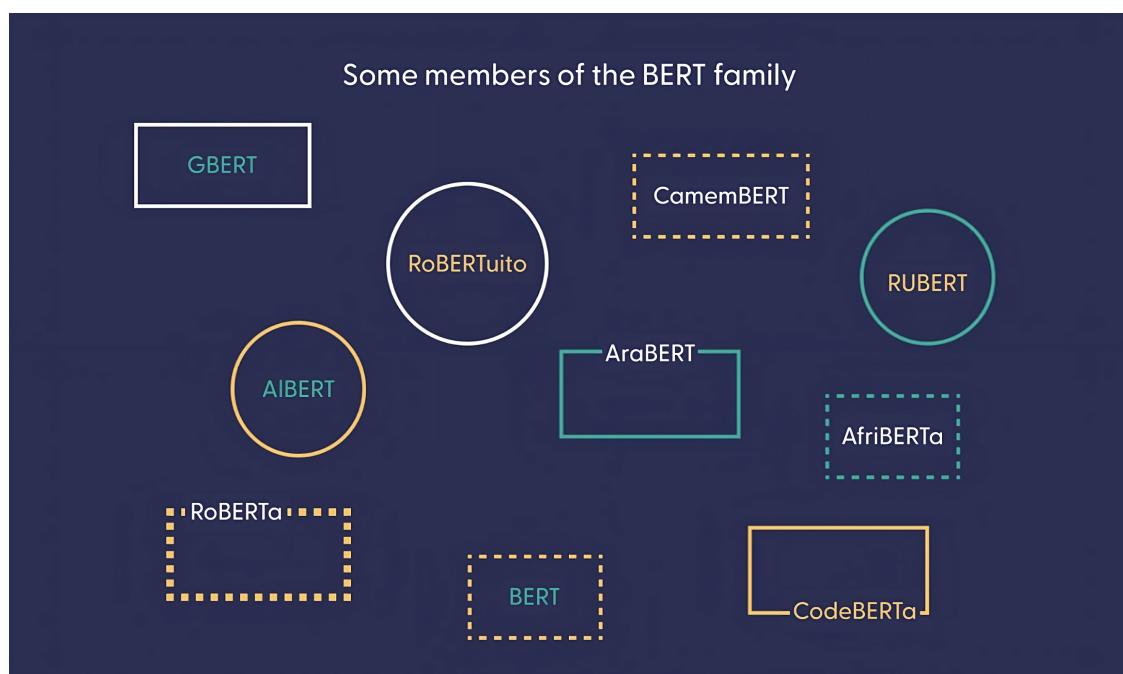


Figure 1.8: some members of the bert family.

1.6.7 Modèles de langage généraux par rapport à ceux spécifiques à un domaine

Les **modèles de langage** généraux comme **BERT** ou sa grande sœur **RoBERTa** nécessitent de très grandes quantités de données pour apprendre les régularités

d'une langue. Les praticiens du **NLP** utilisent souvent Wikipédia et d'autres collections de données textuelles disponibles gratuitement pour les entraîner. À présent, des **modèles** similaires à **BERT** existent pour pratiquement toutes les langues disposant d'une Wikipédia suffisamment vaste.

Alors, que peut-on faire avec ces **modèles**? Pourquoi sont-ils si populaires? Eh bien, **BERT** peut être utilisé pour améliorer la compréhension du langage, par exemple dans le moteur de recherche Google. Mais sans doute la plus grande valeur des **modèles de langage généraux** est qu'ils peuvent servir de base pour d'autres tâches basées sur le langage, telles que la **réponse aux questions**. En les exposant à différents ensembles de données et en ajustant l'objectif de l'entraînement, nous pouvons adapter un **modèle de langage général** à un cas d'utilisation spécifique.

1.6.8 Que peuvent faire les modèles de langage ?

Les **modèles de langage** peuvent sembler très intelligents. Dans ce projet, par exemple, je montre à quel point notre **modèle** peut bien répondre aux questions sur les lois juridiques. Il est important de noter, cependant, que ce **modèle de langage** ne sait en réalité rien du tout. Il est simplement très doué pour extraire les bonnes réponses à partir des documents, grâce à sa maîtrise de la langue humaine et au réglage fin qu'il a reçu sur un ensemble de données de questions-réponses. Il fonctionne de manière similaire à un agent humain qui lit des documents pour en extraire des informations, mais beaucoup, beaucoup plus rapidement! D'autres types de **modèles de langage** adoptent une approche totalement différente. Par exemple, la célèbre famille de **modèles de langage** génératifs GPT mémorise en réalité des informations. Ils ont tellement de paramètres - des milliards - qu'ils peuvent stocker les informations acquises lors de l'entraînement en plus d'apprendre les régularités de la langue. Alors, que peut faire un **modèle de langage**? Exactement ce pour quoi il a été entraîné - ni plus, ni moins. Certains **modèles** sont entraînés pour extraire des réponses à partir de textes, d'autres pour générer des réponses à partir de zéro. Certains sont entraînés pour résumer du texte, d'autres simplement pour apprendre à représenter le langage. Si vos documents n'utilisent pas un langage très spécialisé, un **modèle pré-entraîné** pourrait très bien fonctionner, sans nécessiter d'entraînement supplémentaire. Cependant, d'autres cas d'utilisation pourraient bénéficier d'étapes d'entraînement supplémentaires.

1.6.9 Fine-tuning a language model

Il existe de nombreuses tâches qui bénéficient d'une représentation de l'intuition linguistique. Des exemples de telles tâches sont l'analyse des sentiments, la reconnaissance des entités nommées, la **réponse aux questions**, et bien d'autres. Adapter un **modèle de langage** général à une telle tâche est ce qu'on appelle **fine-tuning**.

Fine-tuning

Un modèle pré-entraîné est caractérisé par ses poids - les paramètres qui ont été définis lors de l'entraînement et qui déterminent le comportement du modèle lors de l'inférence. Lorsque vous effectuez un fine-tuning, les poids sont initialisés aux valeurs pré-entraînées, puis vous les adaptez davantage à vos données en effectuant des étapes d'entraînement supplémentaires. Comparé à la pré-entraînement, le fine-tuning nécessite moins de données (mais généralement annotées). Souvent, il suffit d'environ 500 points de données annotées pour adapter un modèle au nouveau domaine cible, et rarement plus de 5000.

a subheading

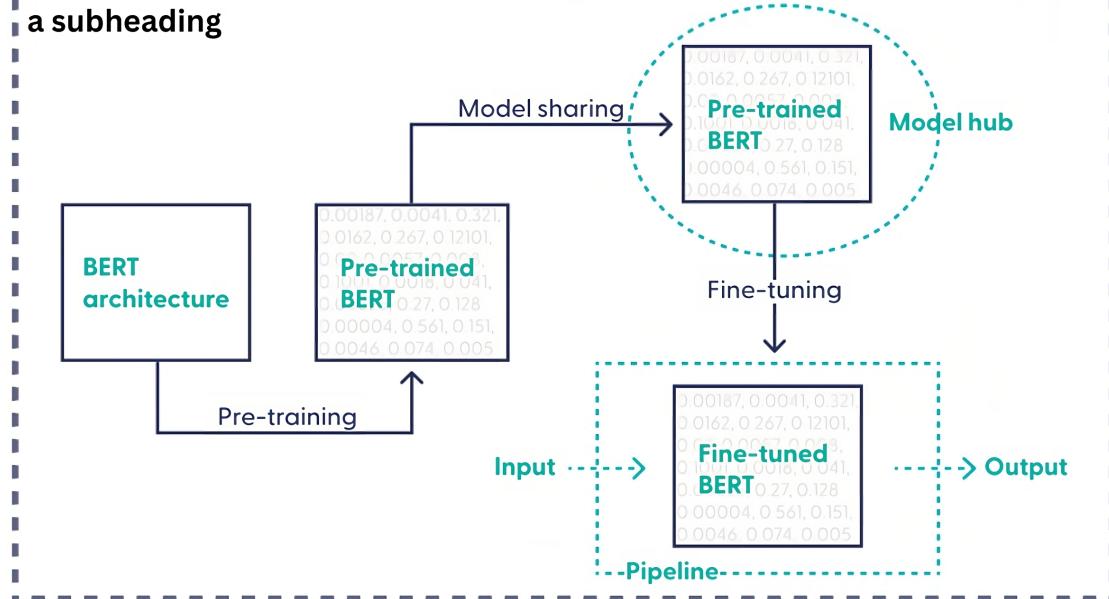


Figure 1.9: fine-tuning.

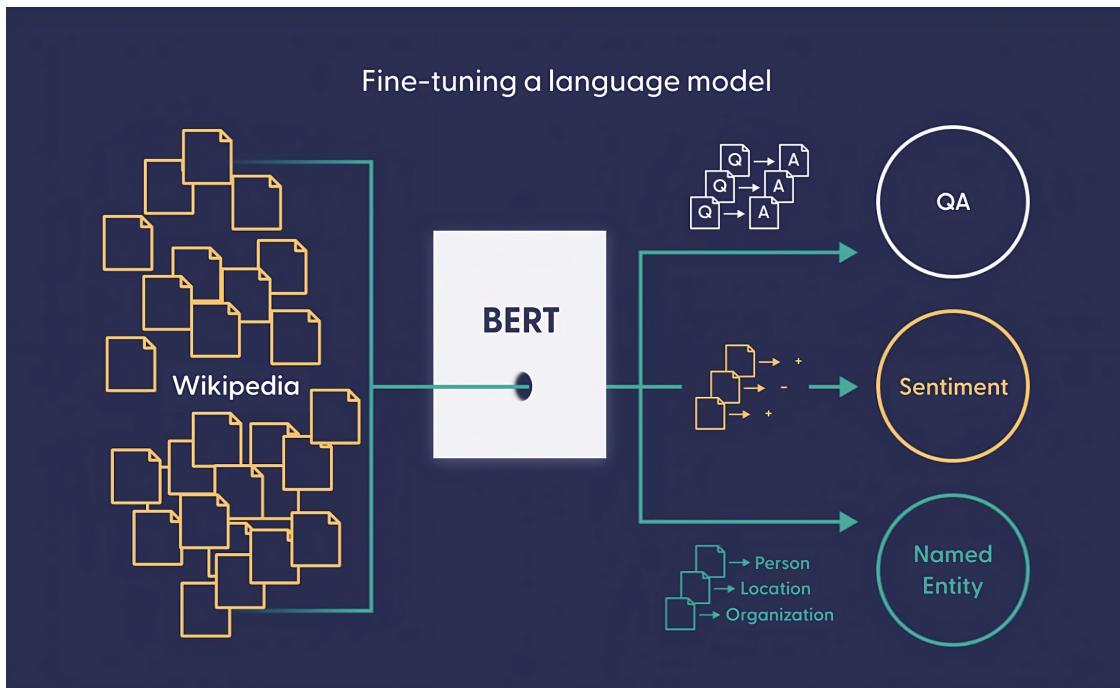


Figure 1.10: fine-tuning a language model.

1.6.10 Où trouver des modèles ?

Les **modèles** à usage général ainsi que les **modèles réglés** peuvent être sauvegardés et partagés. Le modèle hub de **Hugging Face** est la plateforme la plus populaire pour le partage de **modèles**, avec des dizaines de milliers de **modèles** de différentes tailles, pour différentes langues et cas d'utilisation. Il y a de fortes chances que votre propre cas d'utilisation soit déjà couvert par l'un des **modèles** disponibles sur le modèle hub. Pour vous aider à trouver un **modèle** qui pourrait répondre à vos besoins, vous pouvez utiliser l'interface sur le côté gauche de la page du modèle hub pour filtrer par tâche, langue et autres critères. Cela vous permet de rechercher spécifiquement des **modèles** qui ont été entraînés pour la **réponse aux questions**, le **résumé**, et bien d'autres tâches. Une fois que vous avez trouvé un **modèle** adapté, il vous suffit de l'intégrer dans votre **pipeline NLP**, de le connecter à votre base de données et de commencer à expérimenter.

1.6.11 Comment gérer un langage spécifique à un domaine ?

Bien que nous parlions souvent des langues comme s'il s'agissait d'entités homogènes, la réalité en est très loin. Il existe, par exemple, certains domaines professionnels, tels que le droit ou la médecine, qui utilisent un jargon très spécialisé, que les non-experts peuvent à peine comprendre. De même, lorsqu'un **modèle BERT** général

est utilisé pour traiter des données provenant de l'un de ces domaines, il peut avoir de mauvaises performances, tout comme une personne sans diplôme dans le domaine.

Une technique appelée **adaptation au domaine** fournit la solution : dans ce cas, le **modèle pré-entraîné** subit des étapes d'entraînement supplémentaires, cette fois sur des données spécialisées telles que des documents juridiques ou des articles médicaux.

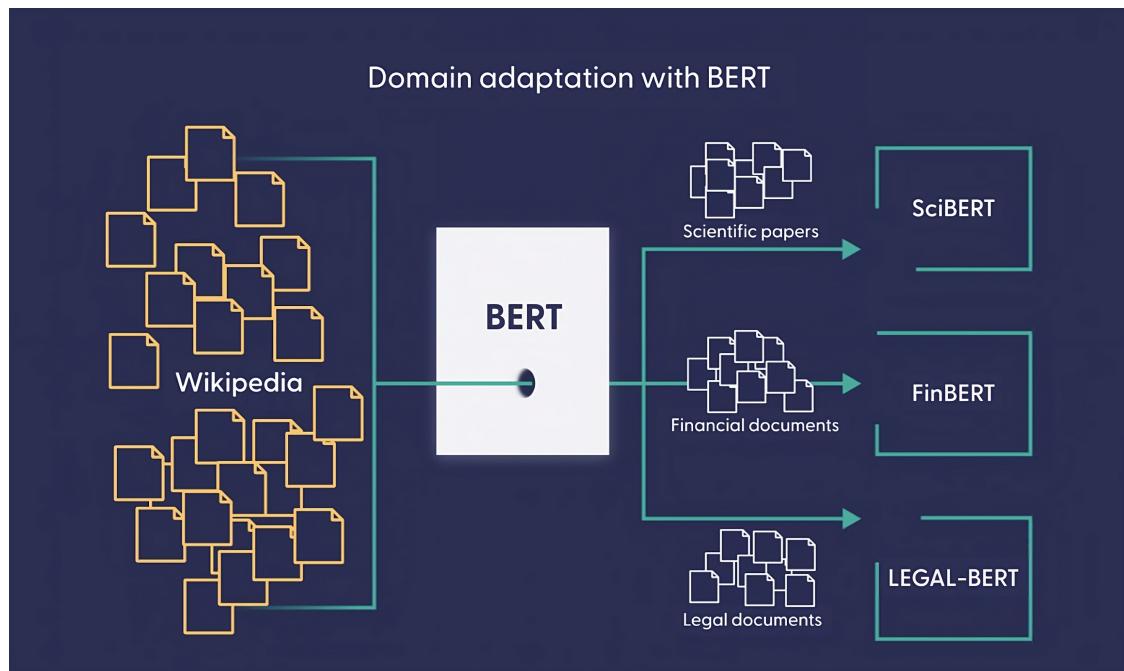


Figure 1.11: domain adaption with bert.

Le **modèle hub** de Hugging Face contient des **modèles de langage** basés sur **BERT** qui ont été adaptés aux domaines scientifique, médical, juridique ou financier. Ces **modèles de langage spécifiques à un domaine** peuvent ensuite servir de base pour d'autres tâches en aval. Par exemple, ce **modèle hautement spécialisé** extrait des **entités nommées** (comme des noms de cellules et de protéines) à partir de textes biomédicaux en anglais et en espagnol.

1.6.12 Choisir les bons modèles

Il est très rare que ceux qui déploient des systèmes de **NLP appliqués** aient besoin d'entraîner leurs propres modèles à partir de zéro. Au lieu de cela, travailler avec des **modèles de langage** basés sur les **Transformers en NLP appliqué** consiste à :

- Choisir le bon **modèle pré-entraîné** pour votre cas d'utilisation.

- Affiner le modèle pour le domaine exact et la tâche de votre application, si nécessaire.
- Évaluer le modèle.
- Surveiller et ré-entraîner périodiquement le modèle.

Dans cette section, nous examinerons la première étape, qui comprend probablement une visite au **modèle hub**. Le modèle hub est un peu comme un grand magasin. Il est préférable de savoir à l'avance ce que nous recherchons, car cela facilitera le choix des modèles que nous souhaitons essayer sans errer dans les allées pendant des heures. Ces modèles peuvent ensuite être intégrés à notre **pipeline**, où nous pouvons les évaluer sur nos propres données. À la fin de ce processus, nous obtiendrons le modèle linguistique qui convient le mieux à notre cas d'utilisation.

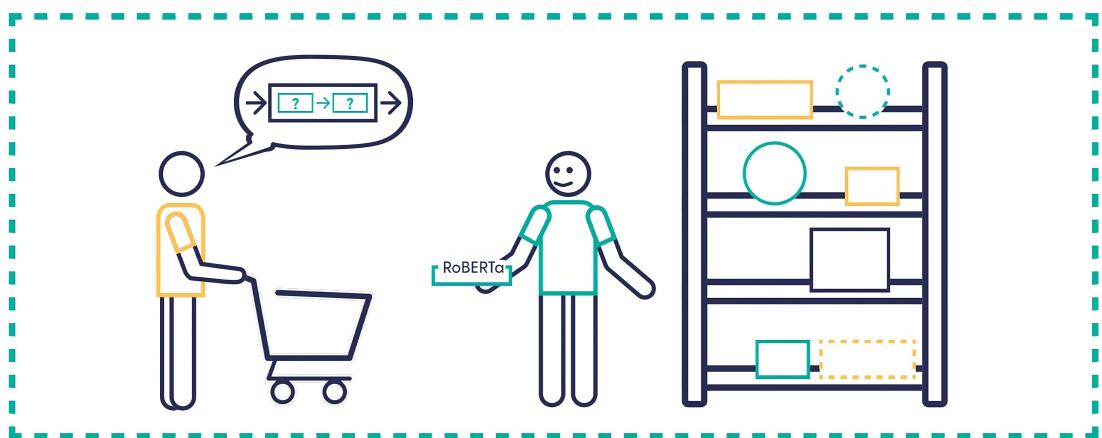


Figure 1.12: choice of the llm.

1.7 Conclusion

Ce chapitre a offert une vue d'ensemble des concepts fondamentaux et des technologies clés dans le domaine du **traitement du langage naturel (NLP)** et de l'**intelligence artificielle (IA)**. Nous avons exploré les bases de l'intelligence artificielle, en passant par le **machine learning** et le **deep learning**, jusqu'aux architectures modernes comme les **Transformers** et les **modèles de langage de grande taille (LLM)**.

Les avantages des Transformers, tels que la parallélisation et la capacité à capturer des dépendances contextuelles à longue portée, ont été soulignés, ainsi que

leur rôle dans la révolution des applications NLP. Nous avons également discuté des techniques d'entraînement et de *fine-tuning* des modèles de langage, ainsi que de l'importance des modèles spécifiques à un domaine pour des applications spécialisées.

En somme, ce chapitre a posé les fondations nécessaires pour comprendre comment les technologies modernes en NLP et IA peuvent être appliquées dans divers cas d'utilisation. Il a également fourni les outils théoriques et pratiques essentiels pour aborder les défis rencontrés lors de la mise en œuvre de systèmes basés sur le NLP. Dans les chapitres suivants, nous approfondirons ces concepts à travers des applications concrètes et des études de cas pratiques, illustrant comment ces technologies peuvent être utilisées pour développer des systèmes performants et robustes dans le domaine du traitement du langage naturel.

2

Conception et outils de développement

Sommaire

2.1	Introduction	27
2.2	Conception de l'application	28
2.3	Outils de Développement Utilisés	28
2.3.1	LangChain	28
2.3.2	FAISS (Facebook AI Similarity Search)	29
2.3.3	Hugging Face Transformers	29
2.3.4	Streamlit	31
2.3.5	Docker	31
2.3.6	Visual Studio Code	32
2.4	Architecture de l'application	33
2.5	Conclusion	34

2.1 Introduction

La conception d'un système d'information est complexe et nécessite une planification détaillée de l'organisation. Elle implique la création d'un modèle virtuel qui met en évidence les aspects essentiels. Les outils informatiques sont cruciaux dans ce processus, jouant un rôle central, en particulier dans le contexte de ce projet. Le chapitre discutera des technologies utilisées pour développer l'application, en commençant par la conception du système et en se terminant par l'examen des langages de programmation utilisés dans différentes parties de l'application.

2.2 Conception de l'application

L'application **Document Q&A Bot** a été conçue pour fournir une solution de question-réponse basée sur des documents en utilisant des techniques avancées de traitement du langage naturel (NLP). La conception a été orientée autour de trois objectifs principaux :

- **Facilité d'utilisation** : Offrir une interface intuitive permettant aux utilisateurs de télécharger divers formats de documents (PDF, DOCX, TXT) et de poser des questions sur le contenu de ces documents.
- **Précision des réponses** : Utiliser des modèles de langage sophistiqués pour générer des réponses précises et contextuellement pertinentes, en se basant uniquement sur les informations contenues dans les documents fournis.
- **Extensibilité** : Créer une architecture flexible capable d'intégrer facilement des améliorations futures et des fonctionnalités supplémentaires, telles que de nouveaux types de documents ou des modèles de langage différents.

2.3 Outils de Développement Utilisés

Pour atteindre ces objectifs, plusieurs outils et technologies ont été intégrés dans le développement de l'application :

2.3.1 LangChain

Ce framework a été utilisé pour orchestrer le processus de génération de réponses, en facilitant l'intégration des modèles de langage et le flux de données entre les différents composants de l'application.



Figure 2.1: LangChain logo.

2.3.2 FAISS (Facebook AI Similarity Search)

FAISS est une bibliothèque développée par Facebook AI, spécialisée dans la recherche de similarités entre des vecteurs. Elle est principalement utilisée pour l'indexation et la recherche dans des ensembles de données de grande dimension, comme des segments de texte ou des images convertis en représentations vectorielles.

Recherche vectorielle FAISS permet de rechercher efficacement des vecteurs similaires dans de grands ensembles de données. Chaque morceau de texte ou document est converti en vecteur via des modèles d'embeddings, et FAISS permet de retrouver rapidement les morceaux les plus proches d'un vecteur donné (par exemple, une question posée par l'utilisateur).

Performance FAISS est conçu pour être extrêmement rapide et efficace, même avec des ensembles de données très volumineux contenant des millions de vecteurs. Il utilise des algorithmes d'approximation et d'optimisation pour accélérer les recherches.

Indexation FAISS propose différents types d'index pour optimiser la recherche en fonction des besoins spécifiques (vitesse, précision). Par exemple, les index IVFPQ et HNSW sont souvent utilisés pour gérer des recherches dans des espaces à grande dimension.

En résumé, FAISS est un outil clé pour les systèmes basés sur des embeddings, tels que les chatbots ou les moteurs de recherche, où la rapidité et la pertinence des résultats sont cruciales. Plus de détails sont disponibles dans la documentation officielle de FAISS : <https://faiss.ai/index.html>.

2.3.3 Hugging Face Transformers

Hugging Face est une plateforme open-source spécialisée dans le développement et l'accès à des modèles de traitement du langage naturel (NLP), notamment ceux basés sur l'architecture Transformer. Ces modèles sont utilisés pour diverses tâches NLP, telles que la génération de texte, la classification et la recherche de similarité.



Figure 2.2: Hugging Face logo.

Dans le cadre de cette application, deux modèles de Hugging Face ont été intégrés pour répondre à des besoins spécifiques.

Le modèle [sentence-transformers/all-MiniLM-L6-v2](#) a été utilisé pour créer des embeddings de documents, transformant ainsi le contenu textuel en représentations vectorielles, facilitant une recherche de similarité efficace à l'aide de FAISS.

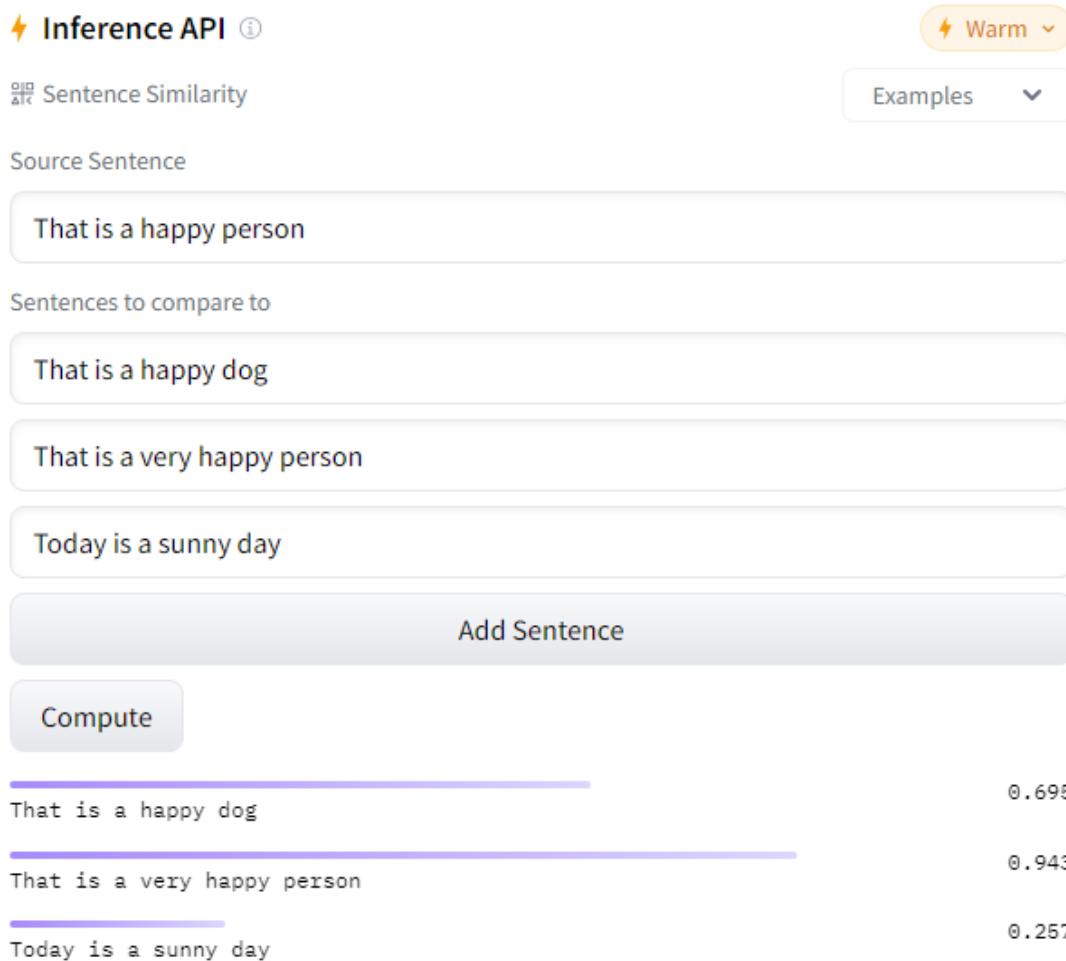


Figure 2.3: Exemple de comparaison de similarité sémantique entre des phrases avec le modèle sentence-transformers/all-MiniLM-L6-v2.

De plus, le modèle [mistralai/Mistral-7B-Instruct-v0.1](#) a été employé pour la génération de réponses, en exploitant les segments textuels les plus pertinents afin de fournir des réponses contextuelles et précises basées sur les documents traités.

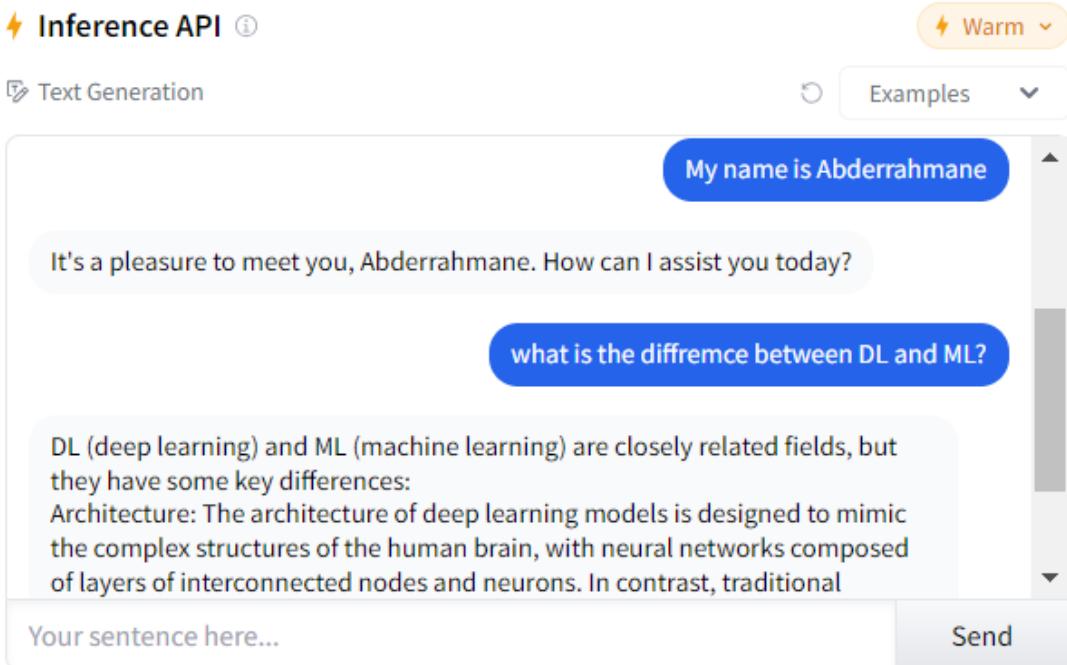


Figure 2.4: Exemple de génération de texte contextuelle et précise avec le modèle Mistral-7B-Instruct-v0.1.

2.3.4 Streamlit

Streamlit est une bibliothèque Python utilisée pour créer l'interface utilisateur de l'application. Streamlit permet de construire des applications web interactives et réactives avec un minimum de code, offrant ainsi une expérience utilisateur fluide et professionnelle.



Figure 2.5: Streamlit logo.

2.3.5 Docker

Docker est une plateforme open-source qui permet de créer, déployer et gérer des conteneurs. Un conteneur regroupe une application et toutes ses dépendances dans

un environnement isolé, garantissant ainsi une exécution cohérente sur différentes machines. Grâce à Docker, les développeurs peuvent standardiser les environnements de développement et de production, facilitant ainsi la portabilité des applications tout en améliorant la gestion des ressources et la sécurité.

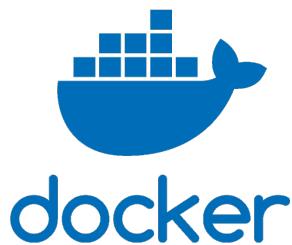


Figure 2.6: Docker logo.

2.3.6 Visual Studio Code

Visual Studio Code est un éditeur de code extensible développé par Microsoft pour Windows, Linux et macOS². Les fonctionnalités incluent la prise en charge du débogage, la mise en évidence de la syntaxe, la complétion intelligente du code, les snippets, la refactorisation du code et Git intégrer. Les utilisateurs peuvent modifier le thème, les raccourcis clavier, les préférences et installer des extensions qui ajoutent des fonctionnalités supplémentaires.

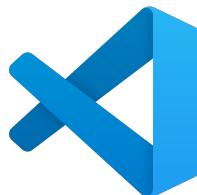


Figure 2.7: Visual Studio Code logo.

Python et Bibliothèques Associées :

- **PyPDFLoader** : Pour charger et traiter des fichiers PDF.
- **Docx2txtLoader** : Pour extraire le texte des documents DOCX.
- **TextLoader** : Pour gérer les fichiers texte simples.
- **dotenv** : Pour gérer les variables d'environnement, notamment les clés d'API et les configurations sensibles.

2.4 Architecture de l'application

L'application **Document Q&A Bot** repose sur la méthodologie **Retrieval-Augmented Generation (RAG)** pour améliorer la précision et la pertinence des réponses générées. Cette approche combine des techniques avancées de récupération d'informations avec la génération de texte, assurant ainsi que les réponses fournies sont non seulement contextuelles, mais également basées sur des informations extraites directement des documents téléchargés par l'utilisateur.

Le processus commence par l'extraction du contenu du document, qu'il s'agisse de fichiers PDF, DOCX, ou TXT. Ce contenu est ensuite divisé en segments plus petits (ou *chunks*) à l'aide de `RecursiveCharacterTextSplitter`, ce qui permet de structurer les données pour un traitement plus efficace. Chaque segment de texte est alors transformé en un vecteur d'embedding à l'aide du modèle `sentence-transformers/all-MiniLM-L6-v2`. Ce modèle est spécialement conçu pour capturer le sens et le contexte du texte sous une forme numérique, facilitant ainsi la comparaison entre les différents segments de texte.

Ces embeddings sont ensuite indexés dans une base de données vectorielle à l'aide de **FAISS (Facebook AI Similarity Search)**, une technologie qui permet d'effectuer des recherches de similarité extrêmement rapides et efficaces parmi des ensembles de données volumineux. Cette étape est cruciale, car elle permet de préparer le terrain pour la récupération rapide des segments les plus pertinents lorsque l'utilisateur pose une question.

Lorsqu'une question est soumise par l'utilisateur, celle-ci est convertie en embedding de requête en utilisant le même modèle `sentence-transformers/all-MiniLM-L6-v2`. Une recherche sémantique est ensuite effectuée dans l'index FAISS pour identifier les segments de texte les plus pertinents par rapport à la question posée. Ces segments récupérés fournissent le contexte nécessaire pour la génération de la réponse.

La génération de la réponse est réalisée par le modèle `mistralai/Mistral-7B-Instruct-v0.1`, un modèle de langage génératif capable de produire des réponses cohérentes et précises en fonction du contexte fourni. Le modèle prend les segments récupérés et la question de l'utilisateur pour générer une réponse qui est à la fois contextuellement adaptée et basée sur des informations exactes extraites du document original.

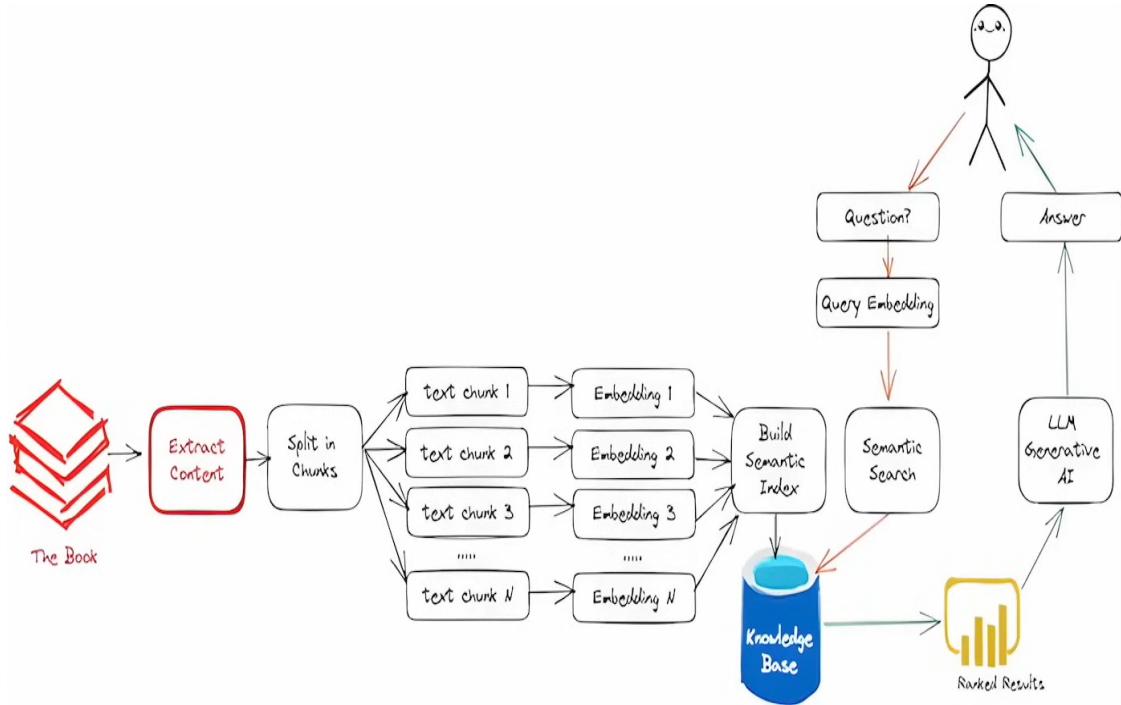


Figure 2.8: Illustration détaillée de l'architecture de l'application Document Q&A Bot utilisant la méthodologie RAG.

En résumé, l'application **Document Q&A Bot** est un exemple d'intégration harmonieuse de technologies de traitement du langage naturel (NLP) de pointe, appuyée par une architecture flexible et des outils de développement modernes, garantissant une expérience utilisateur fluide et des réponses d'une grande précision.

2.5 Conclusion

Dans ce chapitre, nous avons exploré les différentes technologies et outils utilisés pour le développement de l'application **Document Q&A Bot**. Chaque composant, du framework LangChain à la plateforme Hugging Face, a joué un rôle essentiel dans la conception d'un système robuste et extensible, capable de traiter efficacement des documents et de générer des réponses précises et contextualisées.

Nous avons également examiné l'importance de FAISS pour la recherche rapide de similarité et la contribution des modèles de langage, tels que sentence-transformers/all-MiniLM-L6-v2 et Mistral-7B-Instruct-v0.1, dans la gestion et l'analyse des contenus textuels. L'intégration de ces outils avec une interface conviviale construite à l'aide de Streamlit, et l'utilisation de Docker pour assurer la portabilité et la fiabilité du déploiement, montrent l'importance d'une architecture bien pensée.

3

Phase de Réalisation

Sommaire

3.1	Introduction	35
3.2	Démarrage de l'Application	35
3.3	Téléchargement de Document	36
3.4	Processus de Génération de Réponses et Source des Informations	37
3.5	Conclusion	38

3.1 Introduction

Dans cette phase de réalisation, nous allons explorer le processus de démarrage de l'application **Document Q&A Bot** et son fonctionnement. L'application est conçue pour offrir une interface utilisateur conviviale permettant de poser des questions sur le contenu de documents PDF téléchargés, et d'obtenir des réponses précises, extraites directement des documents.

3.2 Démarrage de l'Application

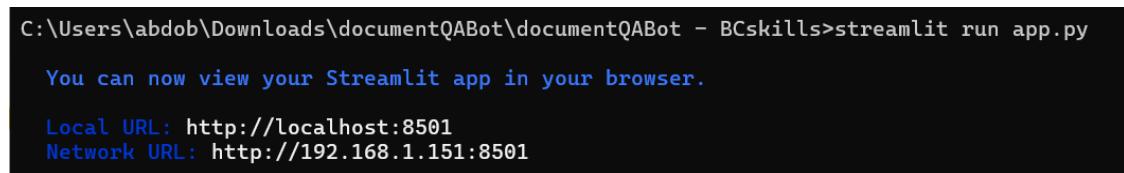
Avant de lancer l'application, il est important de s'assurer que toutes les bibliothèques nécessaires sont installées. Pour ce faire, exécutez la commande suivante dans le répertoire contenant le fichier `requirements.txt` :

```
pip install -r requirements.txt
```

Cette commande installera toutes les dépendances requises pour le bon fonctionnement de l'application.

Ensuite, pour démarrer l'application, il est essentiel de se situer dans le répertoire contenant le fichier principal `app.py`. Une fois dans le bon répertoire, l'application est démarrée en exécutant la commande suivante dans le terminal :

```
streamlit run app.py
```



```
C:\Users\abdob\Downloads\documentQABot\documentQABot - BCskills>streamlit run app.py
You can now view your Streamlit app in your browser.
Local URL: http://localhost:8501
Network URL: http://192.168.1.151:8501
```

Figure 3.1: Lancement de l'application.

Cela initialise un serveur local via le navigateur à l'adresse "`http://localhost:8501`". L'interface utilisateur s'affiche alors, prête à recevoir des documents et des questions.

3.3 Téléchargement de Document

Une fois l'application lancée, l'utilisateur peut télécharger un document en cliquant sur le bouton *Browse files* comme illustré dans la figure 3.2. Le fichier sera analysé par l'application pour permettre la génération des réponses.

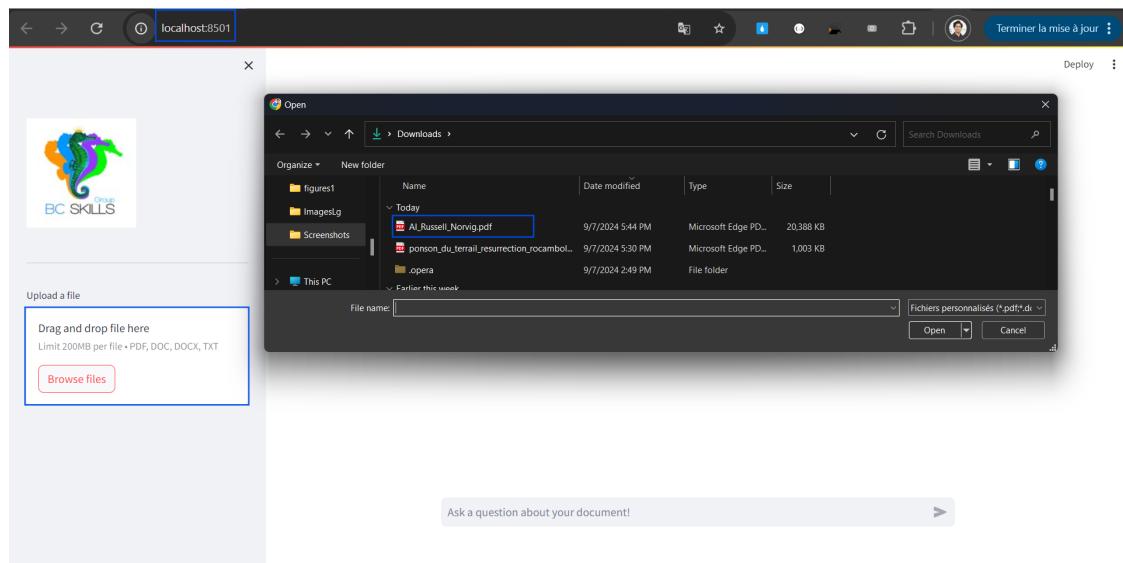


Figure 3.2: Téléchargement d'un fichier PDF dans l'application Document Q&A Bot pour l'analyse et la génération de réponses basées sur le contenu du document.

Dans l'exemple présenté, un fichier intitulé `AI_Russell_Norvig.pdf` a été téléchargé, et sera utilisé pour répondre aux questions posées par l'utilisateur.

3.4 Processus de Génération de Réponses et Source des Informations

Le processus de génération de réponses dans l'application repose sur deux étapes principales : la recherche de la source des informations pertinentes dans les documents fournis par l'utilisateur, suivie de la génération d'une réponse basée sur ces informations.

The screenshot shows the 'Document Q&A Bot' interface. On the left, there's a logo for 'BC SKILLS'. Below it is a file upload section with a placeholder 'Drag and drop file here' and a 'Browse files' button. A file named 'AI_Russell_Norvig.pdf' (19.9MB) is listed. On the right, a question is asked: 'What is the Turing Test, and how is it used to measure intelligence in machines?'. Below the question, a detailed answer is provided: 'The Turing Test is a test for a program to have a conversation (via online typed messages) with an interrogator for five minutes. The interrogator then has to guess if the conversation is with a program or a person; the program passes the test if it fools the interrogator 30% of the time. This test is used to measure intelligence in machines as it provides a satisfactory operational definition of intelligence. A computer passes the test if a human interrogator, after the usefulness of its instantiation in the Loebner Prize competition, and by Ford and Hayes (1995), who argue that the test itself is not helpful for AI. Bringsjord (2008) gives advice for a Turing Test judge. Shieber (2004) and Epstein et al. (2008) collect a number of essays on the Turing Test.' A 'Show source' button is visible. At the bottom, there's a text input field 'Ask a question about your document!' with a send button.

Figure 3.3: Processus de génération de réponses basé sur les informations contenues dans les documents fournis par l'utilisateur.

Une fois que les segments de texte pertinents sont identifiés, l'application utilise ces informations pour générer des réponses contextuelles et précises :

This screenshot shows the same interface as Figure 3.3, but the 'Show source' button is now active, displaying three expanded sections of the document content. The first section discusses the Turing Test's history and purpose. The second section talks about Alan Turing's paper 'Computing Machinery and Intelligence' (1950). The third section is titled '1.1.1 Acting humanly: The Turing Test approach' and quotes Alan Turing's proposal for the test. The rest of the interface remains the same, with the BC SKILLS logo, file upload area, and question input field.

Figure 3.4: Exemple de la source des informations extraites des documents pour générer des réponses précises et contextuelles.

Ce processus garantit que les réponses fournies sont basées uniquement sur les documents téléchargés et sont contextuellement appropriées.

3.5 Conclusion

La phase de réalisation démontre que l'application **Document Q&A Bot** est capable de fournir des réponses précises en se basant sur des documents téléchargés. Grâce à des outils tels que **FAISS** et les modèles de langage de **Hugging Face**, l'application permet de traiter rapidement de grandes quantités de texte et de répondre de manière cohérente aux questions des utilisateurs.

Conclusion Générale

Ce projet a permis de développer un **Document Q&A Bot** (un chatbot de questions-réponses) innovant, capable de traiter de grands volumes de documents et de fournir des réponses contextuelles et précises. En s'appuyant sur des technologies de traitement automatique du langage naturel (NLP) et de deep learning, le système a démontré son efficacité pour extraire des informations pertinentes de documents variés, répondant ainsi aux besoins des utilisateurs de manière rapide et fiable.

L'intégration de bases de données vectorielles, combinée aux modèles de langage sophistiqués, a permis d'optimiser la recherche d'informations et de personnaliser les réponses selon le contexte des documents fournis. Une fonctionnalité clé de cette application est la capacité à fournir la source exacte des informations, renforçant ainsi la transparence et la confiance dans les réponses générées.

En résumé, ce projet démontre la puissance des technologies NLP actuelles dans la création d'outils interactifs intelligents. Il ouvre la voie à des applications futures dans de nombreux secteurs où la gestion efficace de l'information est cruciale. Ce travail peut être enrichi et étendu en intégrant davantage de langues, en améliorant l'expérience utilisateur, et en optimisant les performances des modèles utilisés.

Bibliography

Ressources en ligne

- [1] Hugging Face. *Hugging Face Platform*. [En ligne]. Disponible sur: <https://huggingface.co/>
- [2] Sentence Transformers. *all-MiniLM-L6-v2*. [En ligne]. Disponible sur: <https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2>
- [3] Mistralai. *Mistral-7B-Instruct-v0.1*. [En ligne]. Disponible sur: <https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.1>
- [4] LangChain Documentation. *LangChain*. [En ligne]. Disponible sur: <https://langchain.readthedocs.io/en/latest/>
- [5] Streamlit Documentation. *LangChain avec Streamlit*. [En ligne]. Disponible sur: <https://docs.streamlit.io/develop/tutorials/lms/lm-quickstart>
- [6] Benny Cheung. *Ask a Book Questions with LangChain and OpenAI*. [En ligne]. Disponible sur: <https://bennycheung.github.io/ask-a-book-questions-with-langchain-openai>
- [7] FAISS. (Facebook AI Similarity Search). [En ligne]. Disponible sur: <https://github.com/facebookresearch/faiss>
- [8] Docker Documentation. *Docker Documentation*. [En ligne]. Disponible sur: <https://docs.docker.com/>

Articles

- [9] LeCun, Y., Bengio, Y., & Hinton, G. (2015). *Deep Learning*. Nature, 521(7553), 436-444. [En ligne]. Disponible sur: <https://www.nature.com/articles/nature14539>

- [10] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). *Attention is All You Need*. Advances in Neural Information Processing Systems, 5998-6008. [En ligne]. Disponible sur: <https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fb0d053c1c4a845aa-Paper.pdf>

Livres

- [11] Chollet, F. (2017). *Deep Learning with Python*. Manning Publications. [En ligne]. Disponible sur: <https://www.manning.com/books/deep-learning-with-python>