Université Abou Bekr Belkaid
Tlemcen Algérie
جامعة أبي بكر بلقايد
تلمسان الجزائر

République Algérienne Démocratique et Populaire
Université Abou Bakr Belkaid– Tlemcen
Faculté des Sciences
Département d'Informatique

Mémoire de fin d'études

pour l'obtention du diplôme de Master en Informatique

Option: *Modèle Intelligent et Décision (MID)*

Thème

Réalisation d'un Système de Question/Réponse pour la Langue Arabe

Réalisé par :

- **ACHACHI MOHAMMED**
- **OUBACHIR MOHAMMED IHAB**

Présenté le 23 Juin 2024 devant les jerrys :

Présidente :	-	*Mme AMGHAR DJAZIA*
Encadreur :	-	*Mr ABDERRAHIM MOHAMMED ALAEDDINE*
Examinatrice :	-	*Mme HAMITOU MEHIAOUI ASMA*

Année universitaire : 2023-2024

Dédicace

Je dédie ce travail :

À ma chère mère et à mon cher père, qui n'ont jamais cessé de me soutenir, m'encourager et m'accompagner tout au long de mes années d'études. J'espère qu'ils trouveront ici une expression de ma profonde gratitude et de mon appréciation.

À mes chères sœurs WISSAM, CHAHRAZED et ASMA pour leur soutien continu et leur soutien moral.

À la famille ACHACHI pour leur soutien constant.

À tous mes amis (le groupe وحدوه) qui m'ont toujours soutenu, je leur souhaite encore plus de succès.

À mon frère et partenaire dans le projet, IHAB, que tu restes toujours mon ami le plus cher.

À la personne la plus chère et la plus aimée pour moi, SARAH merci d'être dans ma vie et merci pour tout le soutien.

À mes chères Sœurs HADJER et MANEL pour leur soutien continu et leur soutien moral.

Enfin, je dédie ce travail à mon frère et ami RADHWANE, merci pour tout le soutien que tu m'as apporté, que Dieu te garde pour moi.

Merci à vous tous !

Mohammed

Dédicace

Je dédie ce travail :

À ma chère mère et à mon cher père, qui n'ont jamais cessé de me soutenir, m'encourager et m'accompagner tout au long de mes années d'études. J'espère qu'ils trouveront ici une expression de ma profonde gratitude et de mon appréciation.

À mes chères frères AZIZ et ILIYAS pour leur soutien continu et leur soutien moral.

À mon frère et partenaire dans le projet, MOHAMMED, que tu restes toujours mon ami le plus cher.

À tous mes amis (le groupe وحدوه) qui m'ont toujours soutenu, je leur souhaite encore plus de succès.

À la personne la plus chère et la plus aimée pour moi, HADJER merci d'être dans ma vie et merci pour tout le soutien.

À mes chères Sœurs HADJER et BOUCHRA et IKHLASS pour leur soutien continu et leur soutien moral.

Merci à vous tous !

MOHAMMED IHAB

Remerciements

Je tiens, avant de présenter mon travail, à exprimer ma grande reconnaissance envers les personnes qui m'ont - de près ou de loin - apporté leurs soutiens. Qu'ils trouvent ici collectivement et individuellement l'expression de toute ma gratitude et ma reconnaissance.

Je tiens à remercier également mon enseignant encadrant **M. ABDERRAHIM Mohammed Alaeddine**, en dépit de ses multiples charges, ses aides et les renseignements précieux qu'il m'a fournis ainsi que pour tous les conseils et les informations qu'ils m'aient prodiguées avec un degré de patience et de professionnalisme sans égal.

Je tiens à remercier également les membres de jury madame**AMGHAR**et madame**MEHIAOUI**, pour avoir fait le plaisir d'accepter d'examiner ce travail.

Je tiens aussi à adresser mes plus sincères remerciements à l'ensemble du corps enseignant et administratif de l'Université Abou Bakr Belkaid, pour avoir porté un vif intérêt à notre formation et nous avoir accordé de l'attention et de l'énergie, ce, dans un cadre agréable de respect.

Table des matières

Introduction générale 1

Chapitre 1 : Les modèles de Questions /Réponses 3

1 Introduction ……..……...……... 4

2 Définition de modèles de Questions/Réponses …….……4

3 Les modèles existent de Questions/Réponses ……..……...……..5

3.1 BERT (Représentations bidirectionnelles des encodeurs à partir de Transformers) ……...……...……...…..5

3.2 T5 (Texte-To-texte Transfer Transformer) ……...…….……6

3.3 GPT (Transformer génératif pré-entraîné) ……...…….……8

3.4 Electra (Efficiently Learning an Encoder that Classifies Token Replacements Accurately)……...……..11

3.5 Xlnet ……...……..13

3.6 RoBERTa (Robustly optimized BERT approach) ……...……15

3.7 ALBERT (A Lite BERT) ……...……...……17

4 Conclusion …….

Chapitre 2 : Contribution

1 Introduction……..21

2 Fine-Tuning……..21

2.1 Le principe de Fine-Tuning……21

2.2 Étapes de Fine-Tuning……21

2.2.1 Choix de modèle……..21

2.2.2 La base de données pour le Fine-Tuning…….22

2.2.3 Le code de Fine-Tuning……...23

2.2.3.1 *Les bibliothèques* utilisées……23

2.2.3.2 Étape de chargement du modèle QA……23

2.2.3.3 Prétraitement des données……23

2.2.3.4 Arguments d'entrainement……...25

3 LangChain……29

3.1 Définition……29

3.2 Implémentation QA avec LangChain…….29

3.2.1 Création QA-LangChain basant sur PDF……29

3.2.1.1 Les étapes de QA-LangChain basant sur PDF……..29

3.2.1.2 Base de données de modèle QA-LangChain basant sur PDF…...30

3.2.1.3 La partie Code de modèle QA-LangChain……...31

3.2.2 Création QA-LangChain avec LLMs……..36

3.2.2.1 Les étapes QA-LangChain avec LLMs……36

3.2.2.2 Base de données de QA-LangChain avec LLMs……..37

3.2.2.3 Code de de QA-LangChain avec LLMs……...37

4 Comparaison……40

4.1 Comparaison entre modèle QA non fine-tunée modèle QA fine-tunée……..40

4.2 Comparaison entre QA-Langchain sur PDF et modèle QA non fine-tunée…...42

4.3 Comparaison entre QA-Langchain sur PDF et modèle QA fine-tunée……..44

5 Conclusion……...46

Conclusion générale **47**

Références **48**

Table des figures

Figure 1 : Architecture du model de Questions réponses [1] …….5

Figure 2 : Différences dans les architectures de modèles de bert et gpt [2] ……...6

Figure 3 : Un schéma du cadre T5 [3] ……7

Figure 4 : Structure du modèle T5[4] …….8

Figure 5 : Architecture du modèle GPT [5] ……...9

Figure 6 : Évolution des modèles GPT [6] ……...10

Figure 7 :La détection des tokens remplacés surpasse systématiquement le modèle de langage masqué (MLM) avec le même budget de calcul. [7] ……..12

Figure 8 : Fonctionnement du générateur et du discriminateur dans le modèle ELECTRA [8] ……...12

Figure 9 : Architecture de XLNet [9] ……...14

Figure 10 : Hyperparamètres utilisés pour fine-tune XLNet [10] ……14

Figure 11 : Masquage statique vs masquage dynamique [11] ……..16

Figure 12 : Hyperparamètres utilisés pour fine tune RoBERTa [12] ……...16

Figure 13 : L'architecture du modèle ALBERT dans notre tâche [13] …….17

Figure 14 : Une partie de la base de données……22

Figure 15 : Les bibliothèques utilisées……..……...23

Figure 16 : Le code de chargement du modèle QA……...……23

Figure 17 : Le code de prétraitement de données…...……...24

Figure 18 : La première initialisation……25

Figure 19 : La valeur des arguments……..……...26

Figure 20 : Le résultat de test 2 après le Fine-Tuning…..……..…26

Figure 21 : Le résultat de test 3 après le Fine-Tuning…..……..……27

Figure 22 : Le temps nécessaire pour l'entraînement(test1)……..…..28

Figure 23 : Le temps nécessaire pour l'entraînement(test2)……....28

Figure 24 : Les PDFs……...…….30

Figure 25 : Les bibliothèques de LangChain...……...…..31

Figure 26: Fonction get_pdf_text(pdf_folder).…….....32

Figure 27: Fonction get_text_chunks(text)……...……33

Figure 28: Fonction get_Vectorstores(text_chunks)……..……...33

Figure 29: Fonction get_conversation_chain(vectorstore)……..…….33

Figure 30: Fonction handle_userinput(user_question)..……...34

Figure 31 : Code de Streamlit…..…….34

Figure 32 : Code css pour streamlit..……...35

Figure 33 : Code html pour streamlit...…….35

Figure 34 : Les Bibliothèques utilisée..……37

Figure 35 : La fonction is_arabic(text)..……...38

Figure 36 : La fonction get_reponse(user_query , chat_history).……...38

Figure 37 : Un petit code d’interface streaml……...39

Figure 38 : Le code pour affichez le message d’erreur..……...39

Introduction générale

Notre monde contemporain se distingue aujourd'hui par la présence d'une énorme quantité d'informations dans divers domaines. Cette abondance d'information pose des difficultés d'accès d'une part et des difficultés à obtenir les informations nécessaires d'autre part. Les individus consacrent beaucoup de temps à naviguer entre les informations et à déterminer celles dont ils ont besoin.

Actuellement, avec le succès dans le domaine de l'intelligence artificielle et les technologies modernes, des modèles d'intelligence artificielle ont été développés pour répondre aux questions en se basant sur l'apprentissage à partir de données à grande échelle. Cette avancée scientifique a joué un rôle efficace dans la résolution du problème mentionné précédemment, en inventant plusieurs méthodes telles que le développement d'algorithmes spéciaux pour stocker et classifier les informations pour faciliter leur accès, ainsi que des méthodes de cryptage spéciales pour réduire la taille des informations.

Il est remarquable de noter que le saut qualitatif effectué par les scientifiques dans le développement des modèles d'intelligence artificielle est incroyable, et ces technologies sont devenues un élément indispensable de la vie humaine dans de nombreux domaines, y compris la vie quotidienne, l'éducation, la recherche scientifique, et même la vie sentimentale.

En conséquence de cette évolution, nous, en tant qu'Arabes, devons avoir une empreinte et une place dans ce domaine. C'est pourquoi notre projet vise à créer un modèle d'intelligence artificielle pour répondre aux questions en arabe, notamment dans le domaine de l'islam, afin de faciliter l'accès aux informations concernant cette religion.

À travers cette recherche, nous fournirons un aperçu complet de toutes les étapes par lesquelles le projet est passé pour atteindre l'objectif souhaité, et nous aborderons également toutes les méthodes et stratégies que nous avons suivies pour atteindre cet objectif, sans oublier les problèmes auxquels nous avons été confrontés afin que le lecteur ne tombe pas dans les mêmes erreurs.

Ce mémoire est composé de deux chapitres : Le premier chapitre présente les modèles de questions/Réponses. Il se focalise particulièrement sur la description de Question/Réponse. Le deuxième chapitre décrit notre contribution dans le domaine de Question/Réponse pour la langue Arabe.

Chapitre 1

Les modèles de Questions /Réponses

1 Introduction :

À l'ère de la technologie avancée et de l'abondance de données disponibles, la recherche d'informations précises et pertinentes est devenue cruciale. Dans ce contexte, les progrès rapides dans le domaine du traitement du langage naturel, en particulier dans le domaine de la Question/Réponse (en anglais Question/Answering**QA**), revêtent une importance capitale. Le QA représente un défi passionnant dans l'analyse et l'extraction d'informations cruciales à partir de textes, fournissant ainsi des réponses claires et précises aux questions formulées en langage naturel. Ce domaine repose sur le développement de modèles d'intelligence artificielle capables de comprendre et d'interpréter le langage naturel avec une précision croissante, ainsi que sur la mise en place de techniques efficaces pour extraire et résumer les informations.

2 Définition de modèles de Questions/Réponses :

Un modèle de QA est un système informatique conçu pour comprendre et traiter des questions posées en langage naturel et fournir des réponses précises et appropriées en se basant sur des données textuelles provenant de diverses sources telles que des documents, des articles ou des bases de données. Ces modèles utilisent des techniques avancées d'analyse de données textuelles, d'extraction d'informations et les algorithmes d'apprentissage automatique, y compris l'apprentissage profond (Deep Learning) pour interpréter les questions et extraire les réponses pertinentes. Ils se composent généralement de plusieurs modules, tels-que le traitement de la question, la recherche et l'analyse des documents, et l'extraction de réponse. Les modèles de QA sont utilisés dans divers domaines, y compris l'éducation, la recherche d'informations en ligne, l'assistance clientèle, et bien d'autres, pour fournir une assistance efficace et précise aux utilisateurs en répondant à leurs questions de manière automatisée.

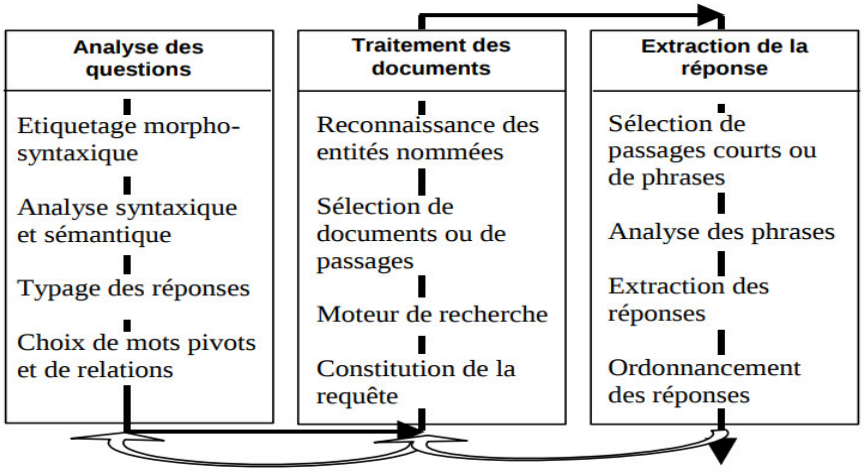


Figure 1 : Architecture du model de Questions réponses [1]

3 Les modèles existent de Questions/Réponses :

Il existe un grand nombre de modèles dédiés à la réponse aux questions, chacun présentant des différences significatives. Chaque modèle se distingue des autres par certains aspects, et chacun possède un point fort qui le distingue des autres modèles. Dans ce qui suit, nous examinerons les modèles les plus importants actuellement disponibles.

3.1 BERT (Représentations bidirectionnelles des encodeurs à partir de Transformers) :

BERT (Représentations Bidirectionnelles des Encodeurs à partir de Transformers) est un modèle de traitement du langage naturel (NLP) développé par Google AI en 2018. Il excelle par sa capacité à comprendre le contexte des mots dans une phrase de manière bidirectionnelle, améliorant ainsi sa performance sur diverses tâches de NLP par rapport à ses prédécesseurs. BERT bénéficie d'une phase de pré-entraînement non supervisée sur de larges corpus textuels, lui permettant d'acquérir une compréhension profonde des modèles linguistiques, et il peut être affiné de manière supervisée pour des tâches spécifiques avec une faible quantité de données annotées. De plus, BERT peut fonctionner de manière multimodale, traitant à la fois le texte et les images pour des tâches combinées de traitement de langage et de vision par ordinateur. Il se distingue par sa capacité à analyser les orientations bidirectionnelles du langage naturel, ce qui facilite la compréhension des significations des mots et des phrases de manière très efficace. Cette analyse se concentre sur

les mots et les phrases avant et après chaque phrase ou mot, les rendant ainsi interdépendants les uns des autres. Cela aboutit à une compréhension précise et efficace du sens, offrant ainsi

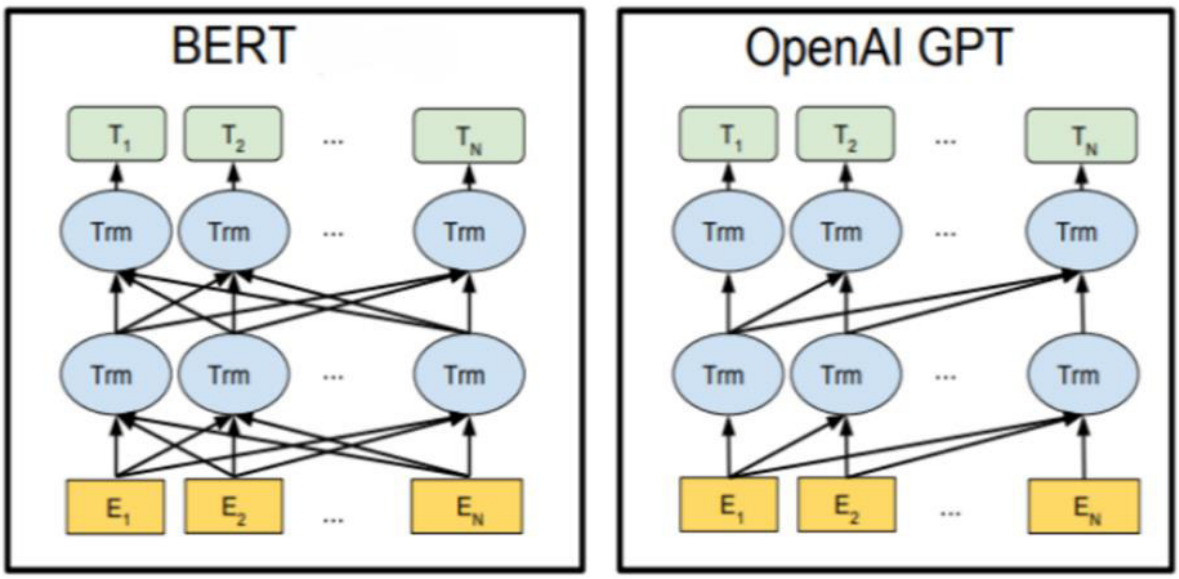


Figure 2 : Différences dans les architectures de modèles de bert et gpt [3]

des résultats plus efficaces et précis [2].

3.2 T5 (Texte-To-texte Transfer Transformer) :

Le modèle T5 est une architecture de réseau neuronal basée sur les transformateurs, développée par Google (pour le (NLP). Il propose une approche unifiée "texte à texte" où chaque tâche NLP est formulée comme une transformation texte à texte. T5 est pré-entraîné sur un ensemble diversifié de tâches, supervisées et non supervisées, lui permettant d'acquérir une compréhension profonde du langage. Sa capacité à traiter différentes tâches avec la même architecture, la même fonction de perte et les mêmes hyperparamètres en fait un choix attrayant pour les applications NLP nécessitant polyvalence et efficacité. T5 a établi des performances de pointe sur diverses tâches de NLP et continue d'être un modèle influent dans le domaine de l'IA [4].

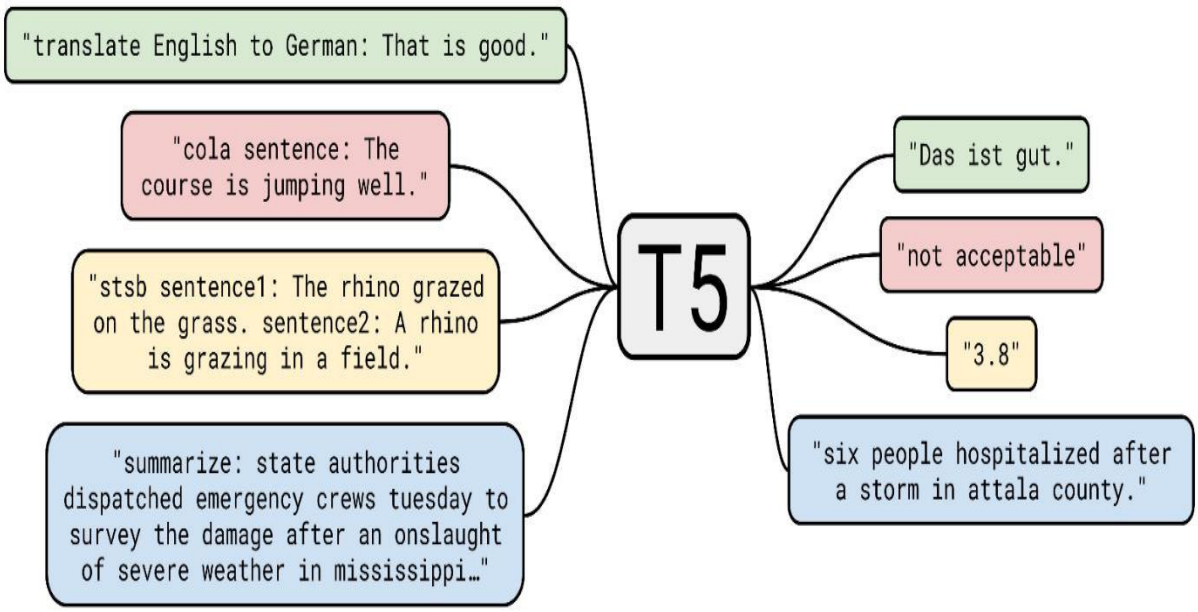


Figure 3 : Un schéma du cadre T5 [5]

Chaque tâche que nous considérons, y compris la traduction, la réponse aux questions et la classification, est formulée comme l'alimentation de modèle avec du texte en entrée et son entraînement à générer un texte cible. Cela nous permet d'utiliser le même modèle, la même fonction de perte, les mêmes hyperparamètres, etc., pour notre ensemble diversifié de tâches.

T5 est entraîné en utilisant le "teacher forcing". Cela signifie que pour l'entraînement, nous avons toujours besoin d'une séquence d'entrée et d'une séquence cible correspondante. La séquence d'entrée est alimentée dans le modèle en utilisant input_ids. La séquence cible est décalée vers la droite, c'est-à-dire préfixée par un jeton de début de séquence et alimentée dans le décodeur en utilisant decoder_input_ids.

Dans le style du "teacher forcing", la séquence cible est ensuite complétée par le jeton EOS à la fin et correspond aux étiquettes. T5 peut être entraîné / ajusté à la fois de manière supervisée et non supervisée [6].

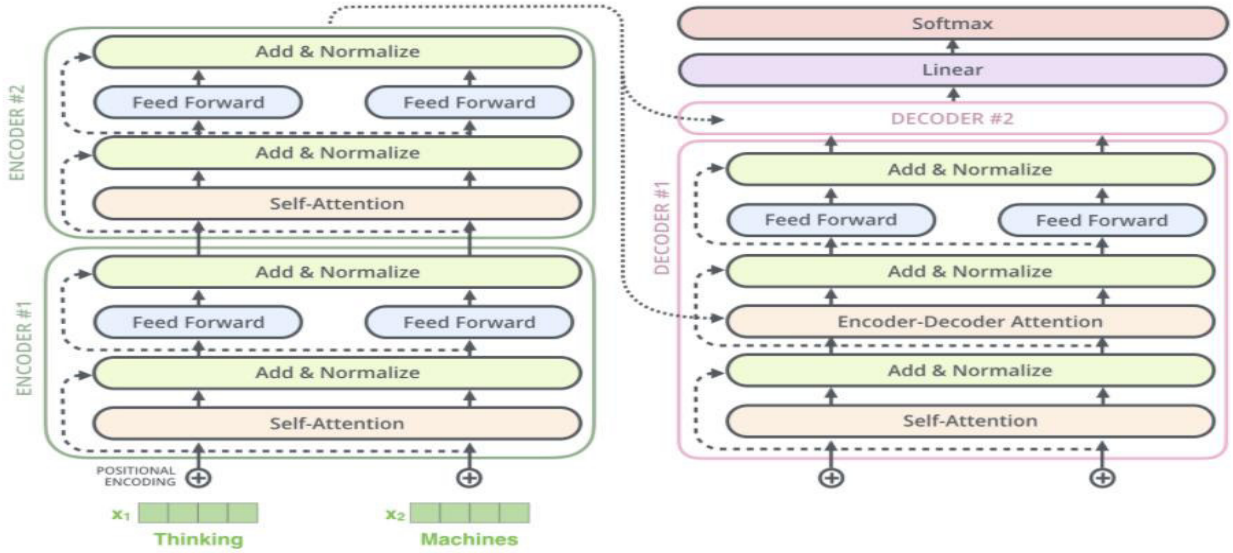


Figure 4 : Structure du modèle T5[7]

Tout comme d'autres modèles basés sur les transformateurs, T5 passe par un processus en deux étapes : le pré-entraînement et fine tuning. Pendant le pré-entraînement, T5 apprend les subtilités du langage en prédisant les parties manquantes du texte d'entrée. Cette phase dote le modèle d'une compréhension globale de la syntaxe, de la sémantique et des relations contextuelles.

Dans l'étape de fine tuning T5 est adapté à des tâches spécifiques de traitement du langage naturel en s'entraînant sur des ensembles de données spécifiques à la tâche. Ce processus affine le modèle pour exceller dans des tâches telles que la traduction, la résumé, la réponse aux questions, et plus encore [8].

3.3 GPT (Transformer génératif pré-entraîné) :

GPT est un modèle de langage basé sur les réseaux de neurones profonds, utilisé dans le domaine de NLP. L'idée principale derrière GPT est l'utilisation de l'apprentissage préalable, une technique qui consiste à entraîner un modèle sur une grande quantité de données avant de l'affiner pour une tâche spécifique.

Lors de la phase d'apprentissage préalable, le modèle GPT est entraîné sur un vaste ensemble de données textuelles, telles que des livres, des articles et des pages Web, pour apprendre les motifs statistiques et les structures du langage naturel. Ensuite, le modèle est ajusté sur des tâches spécifiques en utilisant une technique appelée **Fine-Tuning**, où des

couches de sortie spécifiques à la tâche sont ajoutées et les poids du modèle pré-entraîné sont ajustés sur les données de la tâche. Cette phase de **Fine-Tuning** permet au modèle de s'adapter aux spécificités de la tâche tout en tirant parti des connaissances générales du langage acquises lors de l'apprentissage préalable [9].

GPT se distingue par sa capacité à générer des textes cohérents et pertinents sur le plan contextuel. Cela est réalisé grâce à l'utilisation de mécanismes d'**auto-attention** qui permettent au modèle de pondérer l'importance des différentes parties de la séquence d'entrée lors de la génération du texte de sortie. Ces mécanismes d'auto-attention permettent également à GPT de capturer le contexte et les dépendances entre les différents mots et phrases, le rendant ainsi adapté aux tâches impliquant la génération de séquences de texte plus longues, telles que des articles ou des histoires.

Malgré ses performances impressionnantes dans diverses tâches de traitement du langage naturel, telles que la modélisation linguistique, la réponse aux questions et la classification de texte, GPT est confronté à des défis tels que le problème de **biais**. En effet, les modèles de langage comme GPT apprennent à partir des données sur lesquelles ils sont entraînés, ce qui peut refléter des biais et des stéréotypes présents dans ces données, pouvant conduire à la génération de texte biaisé ou inapproprié [10].

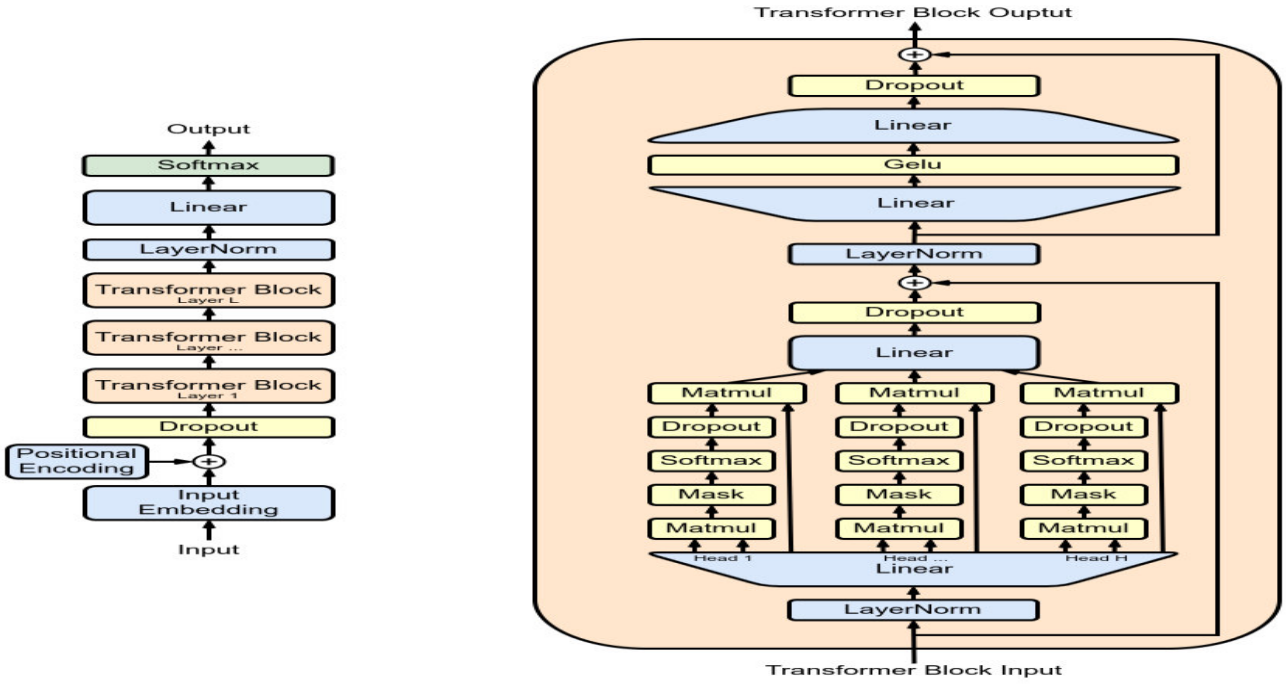


Figure 5 : Architecture du modèle GPT [11]

Depuis son lancement jusqu'à aujourd'hui, ce modèle a connu plusieurs versions, chacune bénéficiant de mises à jour qui l'ont rendu plus avancé et performant que sa version précédente [12].

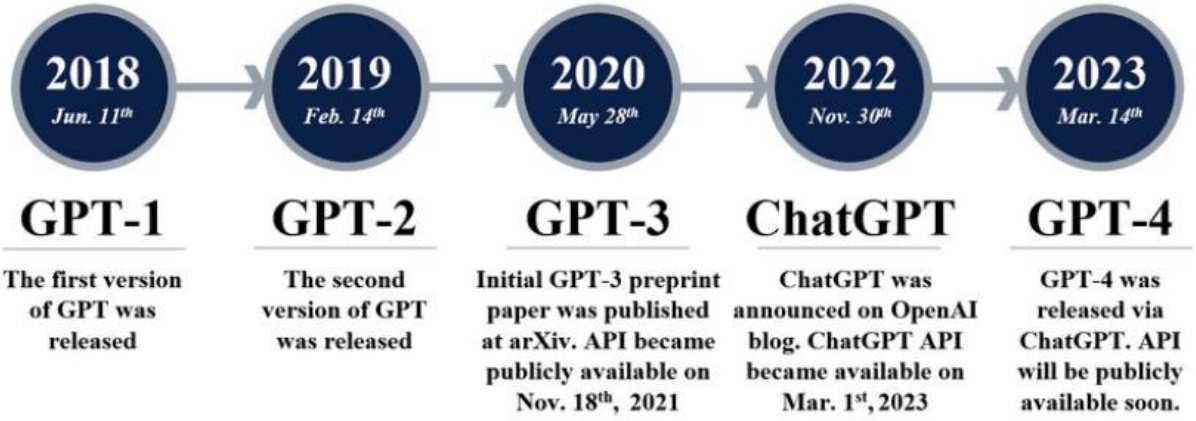


Figure 6 : Évolution des modèles GPT [13]

Il se distingue également par sa capacité à être utilisé dans de nombreux domaines qui aident les humains et économisent beaucoup de temps et d'efforts. Voici les quelques

- ✓ ***La génération de texte*** : GPT est capable de générer du texte de manière cohérente et créative dans un style humain.
- ✓ ***La traduction automatique*** : GPT peut traduire un texte d'une langue à une autre avec une bonne précision.
- ✓ ***L'extraction de résumés*** : GPT peut générer des résumés concis à partir de longs documents.
- ✓ ***Les assistants virtuels*** : GPT peut être utilisé pour construire des assistants virtuels qui comprennent les questions des utilisateurs et fournissent des réponses pertinentes

GPT peut être utilisé pour travailler avec la langue arabe. Bien que les modèles originaux de GPT aient principalement pris en charge l'anglais, de nombreuses versions et modifications de ces modèles ont été développées pour prendre en charge d'autres langues, y compris l'arabe.

Les modèles modifiés pour prendre en charge l'arabe ont été entraînés sur de grands ensembles de données contenant des textes en arabe, ce qui leur permet de comprendre et de générer des textes en arabe de manière similaire à d'autres langues. Ces modèles peuvent être utilisés dans une variété d'applications telles que la génération de texte, la traduction, les conversations automatiques, la classification de texte, et bien d'autres dans le domaine du traitement automatique du langage naturel.

3.4 Electra (Efficiently Learning an Encoder that Classifies Token Replacements Accurately):

ELECTRA est une nouvelle approche de pré-entraînement qui surpasse les techniques existantes avec le même budget de calcul. Par exemple, ELECTRA égale les performances de RoBERTa et XLNet sur le banc d'essai GLUE pour la compréhension du langage naturel en utilisant moins d'un quart de leur puissance de calcul, et obtient des résultats de pointe sur le banc d'essai de réponse aux questions SQuAD. L'excellente efficacité d'ELECTRA signifie qu'il fonctionne bien même à petite échelle - il peut être entraîné en quelques jours sur un seul GPU pour obtenir une meilleure précision que GPT, un modèle qui utilise plus de 30 fois plus de puissance de calcul. ELECTRA est publié en tant que modèle open-source sur TensorFlow et comprend plusieurs modèles de représentation linguistique pré-entraînés prêts à l’emploi [14].

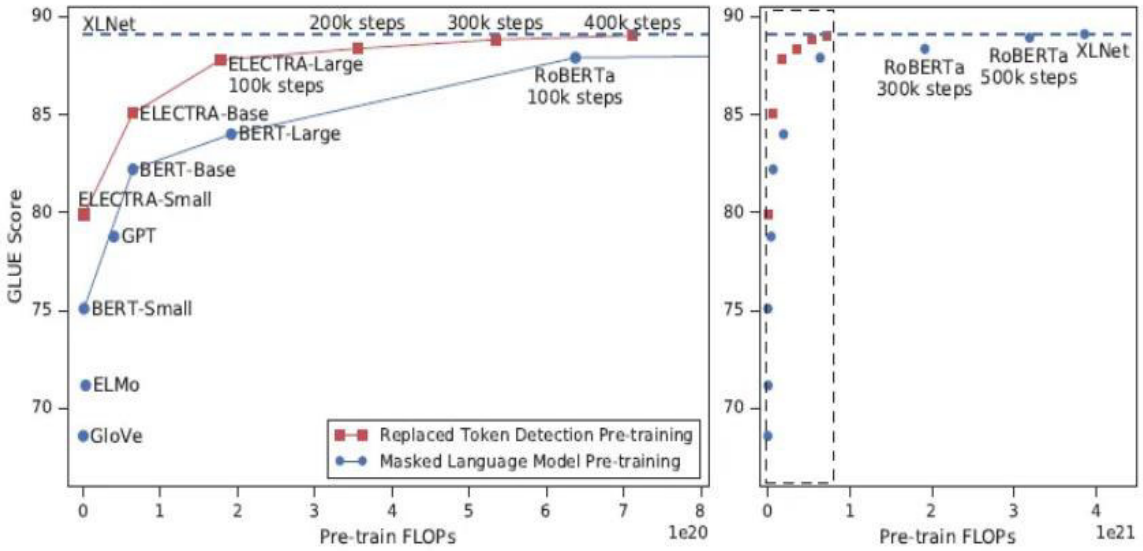


Figure 7 :La détection des tokens remplacés surpasse systématiquement le modèle de langage masqué (MLM) avec le même budget de calcul. [15]

ELECTRA utilise deux modèles pendant la pré-entraînement: un générateur et un discriminateur.

Générateur : Un petit modèle MLM (MaskedLanguage Model) qui prédit les jetons masqués.

Discriminateur : Un modèle plus grand qui tente de distinguer entre les jetons originaux et les jetons remplacés par le générateur.

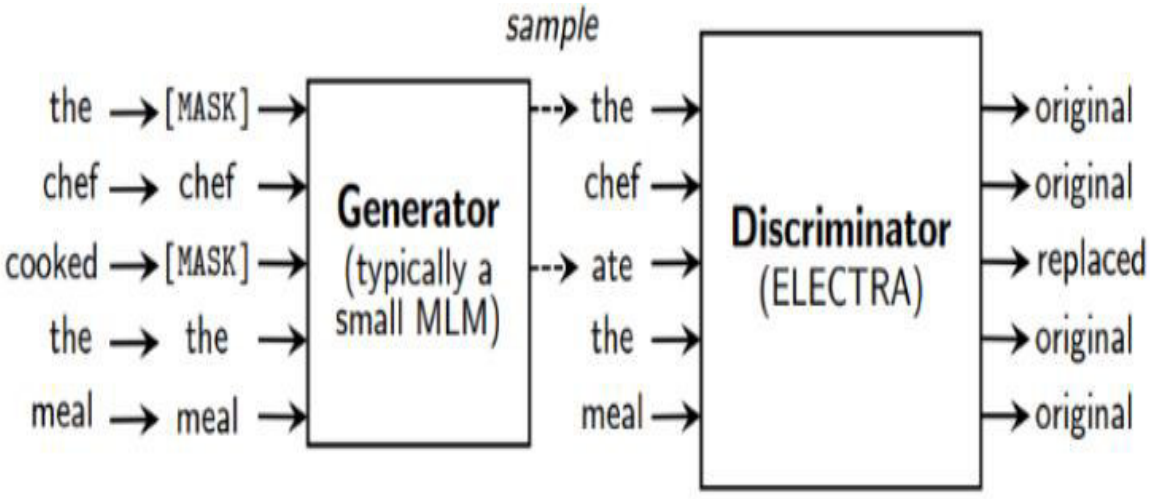


Figure 8 : Fonctionnement du générateur et du discriminateur dans le modèle ELECTRA [16]

Au lieu de masquer des mots, le générateur remplace certains jetons dans le texte d'entrée par des alternatives plausibles. Le rôle du discriminateur est de déterminer si chaque jeton dans l'entrée améliorée a été remplacé par le générateur ou non. Cette tâche force le discriminateur à comprendre des nuances linguistiques plus profondes pour détecter les différences subtiles entre les jetons réels et faux.

Tout comme BERT, ELECTRA peut être ajustéspour différentes tâches de traitement automatique du langage naturel (TALN) en aval telles que l'analyse de sentiment, la réponse aux questions, et bien d'autres.

ELECTRA se forme de manière plus efficace que les modèles basés sur MLM. Étant donné qu'il apprend de tous les jetons d'entrée (pas seulement ceux masqués), il nécessite moins de temps de calcul pour des performances comparables ou meilleures.

ELECTRA représente un changement significatif dans l'approche de la pré-entraînementdes modèles de langage. En se concentrant sur la tâche de détection des jetons remplacés, il offre une manière plus efficace et potentiellement plus efficace d'apprendre les représentations linguistiques par rapport aux modèles traditionnels basés sur MLM. Cette efficacité le rend particulièrement attrayant pour une large gamme d'applications de TALN [17].

3.5 Xlnet :

XLNet est un modèle révolutionnaire de compréhension du langage naturel et de génération de texte, développé par des chercheurs de Google Brain et de l'Université Carnegie Mellon. Contrairement aux modèles traditionnels, XLNet utilise une méthode d'entraînement autorégressive généralisée, ce qui lui permet d'apprendre des contextes bidirectionnels en maximisant la vraisemblance attendue sur toutes les permutations possibles de l'ordre de factorisation de la séquence d'entrée. En intégrant des idées du modèle Transformer-XL, XLNet surmonte les limites des approches précédentes et atteint des performances supérieures dans un large éventail de tâches de compréhension du langage naturel. XLNet représente une avancée majeure dans le domaine de la NLP, offrant une capacité de compréhension plus complète et des performances améliorées par rapport aux modèles précédents.

XLNet est construit sur l'architecture Transformer-XL, qui intègre des mécanismes de récurrence pour capturer les dépendances à long terme et permet au modèle de gérer plus efficacement les séquences plus longues que le modèle Transformer standard.

XLNet introduit un mécanisme de récurrence de segment et un schéma d'encodage relatif pour modéliser la dépendance entre les segments, renforçant ainsi sa capacité à capturer les dépendances à longue portée [18].

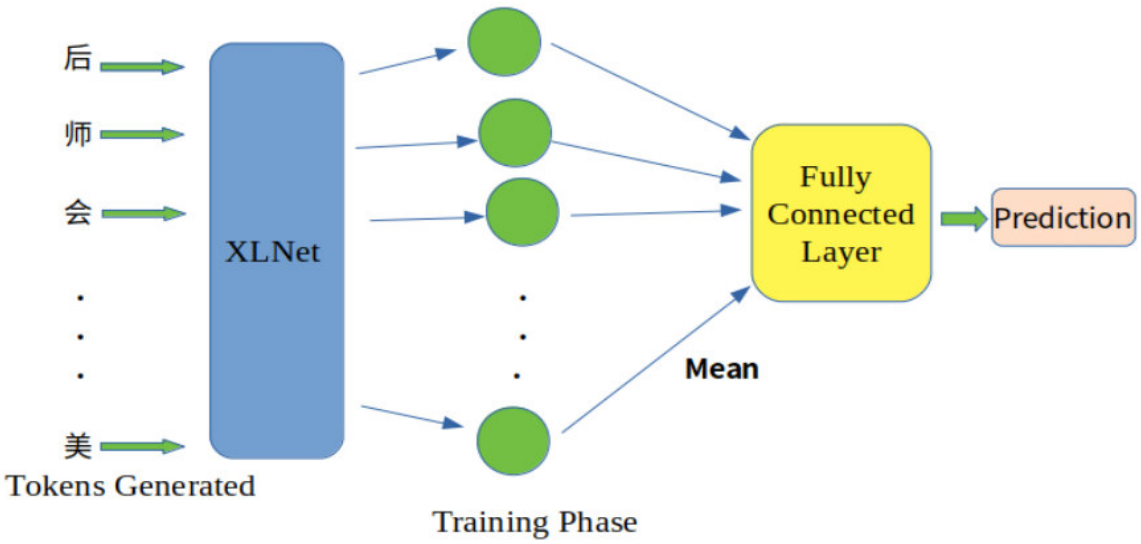


Figure 9 : Architecture de XLNet [19]

Les modèles pré-entraînés comme XLNet offrent une base solide, ajustésur des données spécifiques à la tâche est crucial pour obtenir les meilleurs résultats. Cela implique de former le modèle sur un ensemble de données plus petit qui est spécifique à la tâche ciblée.

Hyperparameter	Values
adam epsilon	1e-8
eval batch size	56
gradient accumulation steps	2
learning rate	1e-5, 5e-5
num train epoch	5, 10, 15
train batch size	56
scheduler	constant schedule with warmup linear schedule with warmup cosine schedule warmup
warmup ratio	0.06
warmup steps	50
weight decay	0.1

Figure 10 : Hyperparamètres utilisés pour fine-tune XLNet [20]

L'émergence de XLNet a ouvert des opportunités de carrière passionnantes dans le domaine de NLP et de la science des données. Alors que la demande de solutions NLP continue de croître, la maîtrise de XLNet et de modèles similaires peut donner aux data scientistes un avantage concurrentiel.

Les professionnels intéressés par une carrière dans le NLP devraient envisager d'acquérir une expertise en XLNet, de comprendre son architecture et d'explorer ses applications dans divers domaines. Se tenir informé des derniers articles de recherche, assister à des conférences et participer à des compétitions Kaggle peuvent également contribuer à la croissance professionnelle.

XLNet est susceptible d'inspirer de nouvelles avancées dans la modélisation du langage. Les chercheurs explorent activement des moyens d'améliorer son efficacité, de réduire le temps d'entraînement et d'étendre ses capacités à des domaines spécifiques. Alors que XLNet continue d'évoluer [21].

3.6 RoBERTa (Robustly optimized BERT approach):

RoBERTa est un modèle pour la compréhension du langage naturel qui constitue une amélioration de BERT Introduit par Facebook AI en 2019, RoBERTa réexamine la méthodologie de pré-entraînement de BERT et apporte des améliorations clés qui ont conduit à des performances améliorées.

Une des améliorations clés de RoBERTa est sa stratégie de masquage dynamique pendant l'entraînement. En masquant aléatoirement des mots dans les séquences d'entrée, RoBERTa utilise une technique appelée modélisation de langage masquée. Cela force le modèle à prédire les mots manquants en se basant sur des indices contextuels, lui permettant d'apprendre des représentations de mots plus robustes et adaptables.

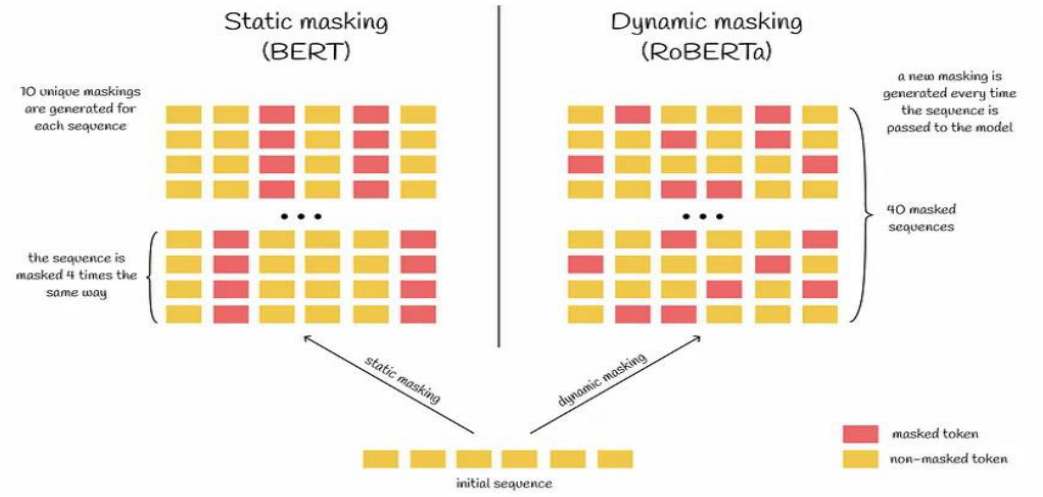


Figure 11 : Masquage statique vs masquage dynamique [22]

RoBERTa réside dans sa polyvalence pour l'entraînement des modèles d'IA par fine tuning. En l'exposant à des ensembles de données étiquetés adaptés à des tâches de NLP spécifiques, telles que l'analyse de sentiment, RoBERTa peut s'adapter et améliorer considérablement ses performances. Cette adaptabilité fait de RoBERTa un choix privilégié pour les applications de recherche et commerciales, servant de modèle fondamental pour le développement de solutions de NLP hautement efficaces.

Pour fine tuneRoBERTa pour les tâches de NLP, il convient de suivre certaines bonnes pratiques. Celles-ci incluent l'expansion de la taille de l'ensemble de données d'affinage et du nombre d'époques, l'expérimentation avec diverses méthodes d'affinage, y compris la prédiction de la phrase suivante, les méthodes de phrases complètes et de phrases de document, et le tokenisation du texte en utilisant le tokenizer pré-entraîné utilisé pour RoBERTa.

Hyperparameter	Value
LoRA r	8
LoRA α	16
Dropout Probability	0.1
Weight Decay	0
Learning Rate	5×10^{-5}
Learning Rate Scheduler	Linear
Batch Size	4
Max Sequence Length	300

Figure 12 : Hyperparamètres utilisés pour fine tuneRoBERTa [23]

RoBERTa est un hommage aux progrès remarquables réalisés dans la modélisation du langage, alors que nous continuons à repousser les limites de l'IA et du NLP. Il a indéniablement joué un rôle significatif dans la détermination de l'avenir des applications de NLP grâce à sa résilience et son optimisation. Alors, préparez-vous à découvrir les opportunités que RoBERTa ouvre alors que nous entrons dans une nouvelle ère du traitement du langage [24].

3.7 ALBERT (A Lite BERT) :

Le modèle ALBERT (A Lite BERT) représente une avancée majeure dans le domaine du traitement du langage naturel. Il est une évolution du modèle BERT bien connu, utilisé pour la représentation et la compréhension du langage. ALBERT se distingue par son caractère plus léger et optimisé, visant à réduire la taille du modèle et à augmenter la vitesse de prédiction, tout en conservant des performances similaires ou améliorées par rapport à BERT.

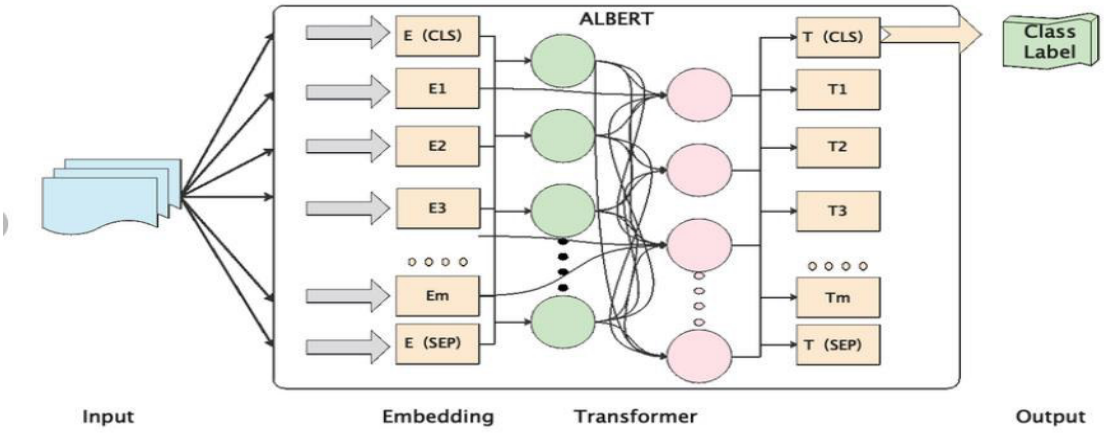


Figure 13 : L'architecture du modèle ALBERT dans notre tâche [25]

ALBERT utilise le même principe d'apprentissage auto-supervisé que BERT, où il est pré-entraîné sur de grandes quantités de données textuelles non annotées, telles que des articles de Wikipédia et d'autres textes disponibles en ligne. L'entraînement repose sur des critères d'auto-apprentissage tels que la prédiction de mots masqués et la prédiction de l'ordre des phrases.

Les principales améliorations d'ALBERT incluent le partage des paramètres entre les couches, la factorisation des matrices d'incorporation et l'utilisation de critères d'auto-apprentissage améliorés. Ces techniques aident à réduire la taille du modèle et à améliorer son efficacité computationnelle.

ALBERT a démontré d'excellentes performances dans une variété de tâches de traitement du langage naturel, mettant en lumière son efficacité remarquable dans la compréhension et l'exécution des fonctions linguistiques. Avec sa disponibilité en tant qu'implémentation open-source et la mise à disposition de modèles pré-entraînés, ALBERT est rapidement devenu un outil privilégié par la communauté de recherche et industrielle en NLP. Son utilisation s'étend à un large éventail d'applications dans le domaine, profitant de ses capacités de représentation puissantes et efficaces.

Nous mentionnons ci-dessous quelques domaines :

- ✓ Analyse des sentiments
- ✓ Reconnaissance d'entité nommée (NER)
- ✓ Réponse aux questions
- ✓ Récapitulation
- ✓ Traduction
- ✓ Recommandation de contenu
- ✓ Moteurs de recherche
- ✓ Chat bot et assistants virtuels

4 Conclusion :

Dans ce chapitre, nous avons présenté une définition générale des modèles de réponse aux questions, ainsi qu'une variété des modèles les plus importants actuellement disponibles.

Le lecteur pourra ainsi découvrir ces modèles de manière détaillée, ainsi que comprendre les caractéristiques distinctives de chacun et les éléments communs entre eux. Nous avons également mentionné les principaux domaines dans lesquels chaque modèle est utilisé.

Il convient de noter qu'il existe un grand nombre de modèles qui n'ont pas été mentionnés, en raison de leur faible utilisation. Nous avons donc concentré notre attention sur les modèles les plus couramment utilisés, que ce soit sur le marché du travail ou dans le domaine de la recherche scientifique. Les modèles mentionnés continuent à évoluer et à se mettre à jour pour devenir plus efficaces.

Ainsi, à travers ce qui a été mentionné, le lecteur, s'il est dans le domaine professionnel, pourra identifier les modèles qui pourraient l'aider dans ses recherches ou son travail.

Chapitre 2 :

Contribution

1 Introduction :

Le **Fine-Tuning** est une technique en intelligence artificielle utilisée pour améliorer les performances des modèles d'apprentissage automatique, en particulier dans le domaine des modèles QA.

Dans ce chapitre, nous expliquerons toutes les méthodes que nous avons utilisées pour créer ce modèle, ainsi qu'une comparaison entre ce qui est disponible sur le marché et ce que nous avons accompli. Nous présenterons également les résultats obtenus en utilisant les différentes méthodes employées.

2 Fine-Tuning :

2.1 Le principe de Fine-Tuning :

Le **Fine-Tuning**, également connu sous le nom d'ajustement fin, est un processus d'adaptation d'un modèle de machine Learning pré-entraîné à une tâche spécifique ou à un ensemble de données spécifiques. Plutôt que de construire un modèle à partir de zéro et de l'entraîner entièrement sur vos propres données, le **Fine-Tuning** consiste à utiliser un modèle pré-entraîné déjà capable de capturer des informations générales sur une grande quantité de données, puis à ajuster les poids du modèle pour qu'il se spécialise dans votre tâche spécifique.

2.2 Étapes de Fine-Tuning :

2.2.1 Choix de modèle :

Parmi les étapes cruciales du projet, celle-ci repose sur le choix du modèle, qui sera entraîné sur notre base de données. La difficulté de cette étape réside dans la variété des caractéristiques que doit posséder le modèle. Il est essentiel que le modèle soit préalablement entraîné à répondre aux questions et qu'il soit également bien formé en langue arabe.

Le modèle que nous avons choisi est : **xlm-roberta-large-arabic_qa**[1] qui est un modèle QA basé sur les transformers, spécialisé pour la langue arabe.

[1] https://huggingface.co/salti/xlm-roberta-large-arabic_qa/tree/main

2.2.2 La base de données pour le Fine-Tuning :

L'étape la plus importante consiste à choisir la base de données qui correspond à notre modèle. Nous avons choisi la base de données **tarteel-ai/quranqa**[26]**,** qui se compose de trois colonnes principales : questions, passages et réponses. Elle est également constituée des parties pour l'entrainement, validation, test et test sans réponse divisées comme suit :

Train	Validation	Test	Test sans réponse
710	190	238	274

La figure 14 présente une partie de notre base de données.

Figure 14 : Une partie de la base de données

2.2.3 Le code de Fine-Tuning :

2.2.3.1 Les bibliothèques utilisées :

La figure 15 présente toutes les bibliothèques nécessaires.

```
from transformers import AutoModelForQuestionAnswering, AutoTokenizer, pipeline
from datasets import load_dataset
from transformers import TrainingArguments, Trainer, AutoModelForQuestionAnswering, DefaultDataCollator
from accelerate import Accelerator
```

Figure 15: les bibliothèques utilisées

2.2.3.2 Étape de chargement du modèle QA :

Dans cette étape, nous avons chargé le modèle, que nous allons entraîner, de Hugging Face ainsi que la base de données sur laquelle nous allons effectuer l'entraînement.

```
[2]
model_name = 'salti/xlm-roberta-large-arabic_qa'

[ ]
nlp = pipeline('question-answering', model=model_name, tokenizer=model_name)

[4]
data = load_dataset('tarteel-ai/quranqa')

[ ]
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForQuestionAnswering.from_pretrained(model_name)
```

Figure 16: le code de chargement du modèle QA

2.2.3.3 Prétraitement des données :

Pour permettre au modèle de traiter les données efficacement, nous avons prétraité les données pour une tâche de question-réponse. Nous avons affiné les questions, les avons tokenisées avec les passages correspondants, et calculé les positions de début et de fin des réponses dans les passages. Les données sont ensuite transformées pour être compatibles avec un modèle de traitement du langage naturel. Enfin, un collateur de données est utilisé pour regrouper les données avant l'entraînement.

Les étapes du prétraitement sont illustrées dans les images suivantes :

```python
def preprocess_function(examples):
    """Courtesy of https://huggingface.co/docs/transformers/tasks/question_answering"""
    questions = [q.strip() for q in examples["question"]]
    inputs = tokenizer(
        questions,
        examples["passage"],
        max_length=384,
        truncation="only_second",
        return_offsets_mapping=True,
        padding="max_length",
    )

    offset_mapping = inputs.pop("offset_mapping")
    answers = examples["answers"]
    start_positions = []
    end_positions = []

    for i, offset in enumerate(offset_mapping):
        answer = answers[i]
        if "answer_start" in answer and answer["answer_start"]:
            start_char = answer["answer_start"][0]
            end_char = start_char + len(answer["text"][0])
        else:
            # Si "answer_start" n'est pas défini ou est une liste vide, marquez (0, 0)
            start_char, end_char = 0, 0

        sequence_ids = inputs.sequence_ids(i)
```

```python
        # Find the start and end of the context
        idx = 0
        while sequence_ids[idx] != 1:
            idx += 1
        context_start = idx
        while sequence_ids[idx] == 1:
            idx += 1
        context_end = idx - 1

        # If the answer is not fully inside the context, label it (0, 0)
        if offset[context_start][0] > end_char or offset[context_end][1] < start_char:
            start_positions.append(0)
            end_positions.append(0)
        else:
            # Otherwise it's the start and end token positions
            idx = context_start
            while idx <= context_end and offset[idx][0] <= start_char:
                idx += 1
            start_positions.append(idx - 1)

            idx = context_end
            while idx >= context_start and offset[idx][1] >= end_char:
                idx -= 1
            end_positions.append(idx + 1)

    inputs["start_positions"] = start_positions
    inputs["end_positions"] = end_positions
    return inputs
```

```python
tokenized_data = data.map(preprocess_function, batched=True, remove_columns=data["train"].column_names)
```

```python
# Data collator
data_collator = DefaultDataCollator()
```

Figure 17 : le code de prétraitement de données

2.2.3.4 Arguments d'entrainement :

Cette étape peut être considérée comme le point de rupture entre l'efficacité et l'inefficacité de l'entrainement car ces valeurs varient d'un modèle à un autre et d'une base de données à une autre, et leur complexité est accrue par le fait que ces valeurs sont liées à plusieurs facteurs tels que la taille de la base de données. Nous discutons de ces valeurs et du rôle de chaque valeur, puis nous mentionnons comment nous avons traité ces valeurs dans notre projet.

output_dir: chemin du répertoire où les résultats de l'entraînement seront sauvegardés.
evaluation_strategy: fréquence d'évaluation du modèle (ici à la fin de chaque époque).
learning_rate: taux d'apprentissage utilisé par l'optimiseur pour ajuster les poids du modèle.
per_device_train_batch_size: taille du lot d'entraînement par appareil (GPU/CPU).
per_device_eval_batch_size: taille du lot d'évaluation par appareil (GPU/CPU).
num_train_epochs: nombre total d'époques (passes complètes sur le jeu de données d'entraînement).
weight_decay: taux de décroissance des poids utilisé pour régulariser l'entraînement et éviter le surapprentissage.

Au cours de notre projet, nous avons initialisé plusieurs valeurs pour parvenir à un résultat satisfaisant, en raison de la petite taille de la base de données.

Test 1 :

```
# Training arguments
training_args = TrainingArguments(
    output_dir="./my_qa_model_xlm_roberta_arabic",
    evaluation_strategy="epoch",
    learning_rate=5e-5,
    per_device_train_batch_size=1,
    per_device_eval_batch_size=1,
    num_train_epochs=7,
    weight_decay=0.01,
)
```

Figure 18 : La première initialisation

Par exemple, après avoir pris ces valeurs, les résultats de modèle après Fine-Tuning étaient bons, mais il a pris beaucoup de temps.

Test 2 :

dans ce test nous avons changé les valeurs des arguments comme il ont illustré dans la figure18.

```
# Training arguments
training_args = TrainingArguments(
    output_dir="./my_qa_model_xlm_roberta_arabic",
    evaluation_strategy="epoch",
    learning_rate=5e-5,
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    num_train_epochs=7,
    weight_decay=0.01,
)
```

Figure 19 : la valeur des arguments

Le résultat de l'entrainement était comme suit :

```
from transformers import AutoModelForQuestionAnswering, AutoTokenizer, pipeline

# Assurez-vous que toutes les autres importations nécessaires sont également présentes dans votre script


# Charge (class) AutoModelForQuestionAnswering
model_di                                                    ic"
model = AutoModelForQuestionAnswering.from_pretrained(model_directory)
tokenizer = AutoTokenizer.from_pretrained(model_directory)

# Initialiser le pipeline de question-réponse avec le modèle fine-tuné
nlp = pipeline('question-answering', model=model, tokenizer=tokenizer)

# Exemple de passage et de question à tester
passage = "يؤذون النبي ويقولون هو أذن قل أذن خير لكم يؤمن بالله ويؤمن للمؤمنين ورحمة للذين آمنوا منكم والذين يؤذون رسول الله لهم عذاب أليم
question = "ما هي مصارف الزكاة؟"

# Obtenir la réponse
answer = nlp(question=question, context=passage)

# Afficher la réponse
print("Réponse:", answer["answer"])
```

```
✓ 14.4s                                                                    Python

c:\Python\Python312\Lib\site-packages\tqdm\auto.py:21: TqdmWarning: IProgress not found. Please update jupyter and ipywidgets. See https://ipywidgets.r
  from .autonotebook import tqdm as notebook_tqdm
Réponse: إنما الصدقات للفقراء والمساكين والعاملين عليها
```

Figure 20: le résultat de test 2 après le Fine-Tuning

La réponse correcte de la base de données est : للفقراء والمساكين والعاملين عليها والمؤلفة قلوبهم وفي الرقاب والغارمين وفي سبيل الله وابن السبيل

La réponse du modèle après l'entraînement est : انما الصدقات للفقراء و المساكين و العاملين عليها

Un autre exemple :

```python
from transformers import AutoModelForQuestionAnswering, AutoTokenizer, pipeline

# Assurez-vous que toutes les autres importations nécessaires sont également présentes dans votre script


# Charger le modèle fine-tuné
model_directory = "my_qa_model_xlm_roberta_arabic"
model = AutoModelForQuestionAnswering.from_pretrained(model_directory)
tokenizer = AutoTokenizer.from_pretrained(model_directory)

# Initialiser le pipeline de question-réponse avec le modèle fine-tuné
nlp = pipeline('question-answering', model=model, tokenizer=tokenizer)

# Exemple de passage et de question à tester
passage = "بوتكم لعلكم تشكرون، وظللنا عليكم الغمام وأنزلنا عليكم المن والسلوى كلوا من طيبات ما رزقناكم وما ظلمونا ولكن كانوا أنفسهم يظلمون
question = "ما هي انواع الحيوانات التي ذكرت في القرآن؟"

# Obtenir la réponse
answer = nlp(question=question, context=passage)

# Afficher la réponse
print("Réponse:", answer["answer"])
```

✓ 9.6s Python

Réponse: العجل

Figure 21 : le résultat de test 3 après le Fine-Tuning

La réponse correcte de la base de données est : العجل السلوى

La réponse du modèle après l'entraînement est : العجل

Dans les deux cas précédents, les résultats du modèle lors de la question après l'entraînement étaient largement acceptables, mais la différence réside dans le temps nécessaire à l'entraînement, comme illustré dans les deux images suivantes

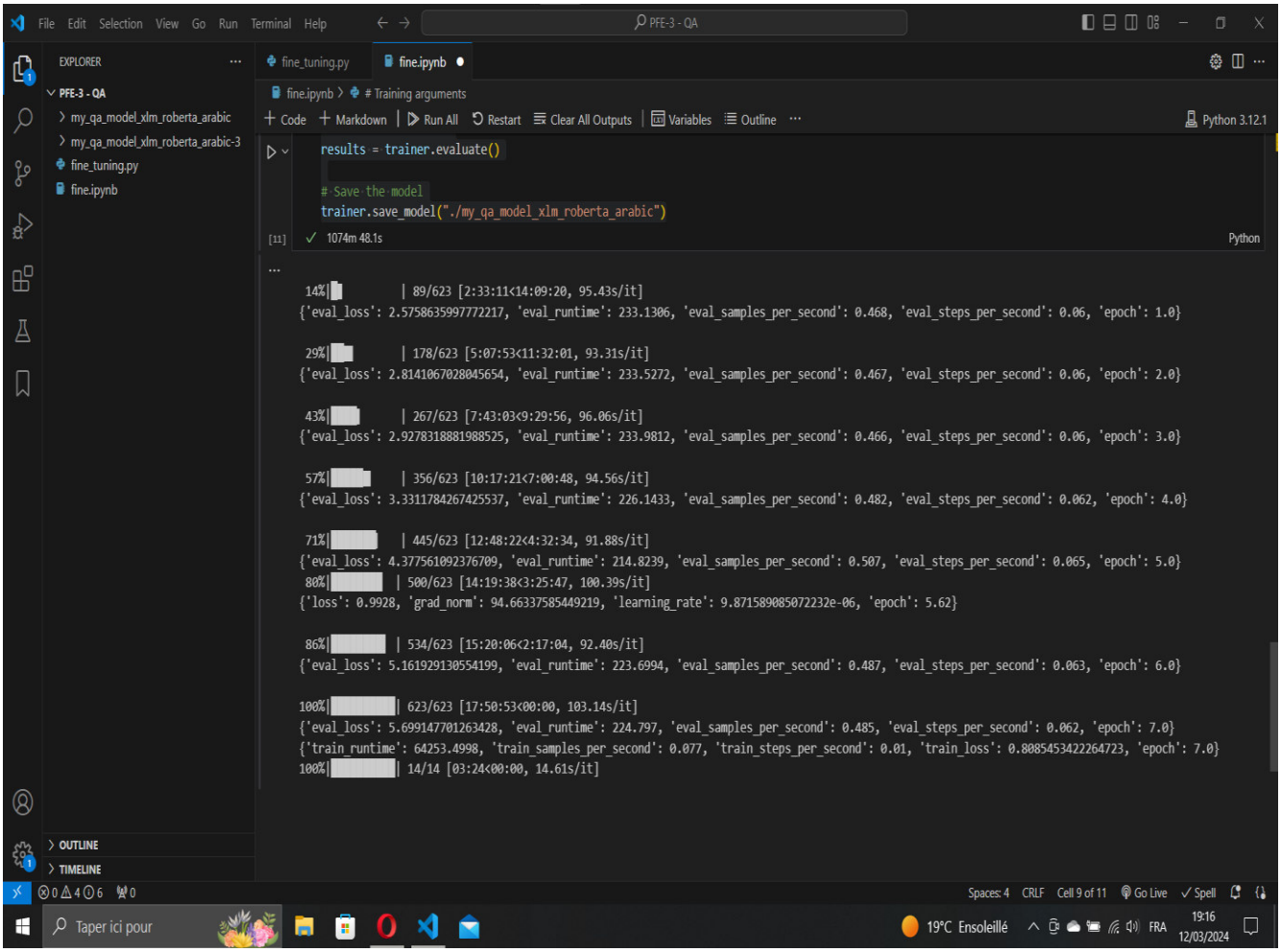


Figure 22 : Le temps nécessaire pour l'entraînement(test1)

Figure 23 : Le temps nécessaire pour l'entraînement(test2)

3 LangChain :

3.1 Définition :

LangChain est un Framework open-source conçu pour simplifier la création d'applications utilisant de grands modèles de langage (LLMs) ,disponible dans les bibliothèques Python et Java[27].

Grâce à ses schémas et modèles flexibles, LangChain simplifie la création des applications alimentées par les LLM, comme les chatbots et les agents conversationnels , et aussi facilite l'intégration et le traitement de PDF.

3.2 Implémentation QA avec LangChain :

Dans notre projet, nous avons implémenté un chatbot QA pour la langue arabe en utilisant LangChain avec deux méthodes différentes :

3.2.1 Création QA-LangChain basant sur PDF :

LangChain offre plusieurs fonctionnalités pour traiter et analyser des fichiers PDF, nous permettant de créer des applications qui interagissent avec des documents PDF de manière intelligente. Voilà les étapes du fonctionnement de LangChain avec les PDF dans notre projet :

3.2.1.1 Les étapes de QA-LangChain basée sur PDF :

- **Définir le chemin du dossier contenant les fichiers PDF**

- **Charger le texte des fichiers PDF dans le dossier :**
  Cette étape permette d'extraire le contenu textuel des fichiers PDF, ce qui constitue la base de connaissances sur laquelle le Chatbot s'appuiera pour répondre aux questions des utilisateurs.

- **Segmentation de texte :**
  Pour faciliter le processus d'inclusion et de recherche par le Chatbot en divisant le texte en segments plus petits, ce qui améliore l'efficacité et la pertinence des réponses.

- **Création de l'index vectoriel :**

  Cette étape est essentielle pour transformer le texte brut extrait des PDF en une représentation numérique compréhensible par le Chatbot. Ce processus permet au Chatbot de raisonner sur le contenu des PDF et de fournir des réponses pertinentes aux questions des utilisateurs.

- **Création de la chaîne de conversation :**

  Cette étape de création de la chaîne de conversation est l'armature de notre projet. Elle lui donne la capacité d'apprendre, de s'adapter et d'offrir une expérience conversationnelle riche.

3.2.1.2 Base de données de modèle QA-LangChain basant sur PDF :

Cette base de données n'est pas une structure de stockage statique, mais plutôt un ensemble de données dynamiques traitées qui prennent en charge les capacités de recherche et de réponse du Chatbot. En d'autres termes, il ne s'agit pas d'une base de données relationnelle traditionnelle, mais plutôt d'un ensemble de fichiers PDF et du texte extrait de ces fichiers.

Figure 24: Les PDFs

3.2.1.3 La partie Code de modèle QA-LangChain :

- **Chargement des bibliothèques :**

```
import streamlit as st
import os
from dotenv import load_dotenv
from PyPDF2 import PdfReader
from langchain.text_splitter import CharacterTextSplitter
from langchain.embeddings import OpenAIEmbeddings
from langchain_community.embeddings import HuggingFaceInstructEmbeddings
from langchain.vectorstores import FAISS
from langchain_community.chat_models import ChatOpenAI
from langchain.memory import ConversationBufferMemory
from langchain.chains import ConversationalRetrievalChain
from htmlTemplates import css, bot_template, user_template
from langchain.llms import HuggingFaceHub
```

Figure 25 : les bibliothèques de LangChain

- **Import streamlit as st :** Importe la bibliothèque Streamlit pour créer l'interface web.
- **Import os :** Importe la bibliothèque os pour gérer les chemins de fichiers.
- **from dotenv import load_dotenv:** Importe la fonction load_dotenv pour charger les variables d'environnement (non détaillées dans le code fourni).
- **from PyPDF2 import PdfReader:** Importe la classe PdfReader de la bibliothèque PyPDF2 pour lire les fichiers PDF.
- **from langchain.text_splitter import CharacterTextSplitter:** Importe la classe CharacterTextSplitter de langchain pour diviser le texte en morceaux.
- **from langchain.embeddings import OpenAIEmbeddings, HuggingFaceInstructEmbeddings:** Importe les classes OpenAIEmbeddings et HuggingFaceInstructEmbeddings de langchain pour créer des embeddings de texte.
- **from langchain.vectorstores import FAISS:** Importe la classe FAISS de langchain pour créer un "vector store" (stockage vectoriel).
- **from langchain_community.chat_models import ChatOpenAI: I**mporte la classe ChatOpenAI de langchain_community pour utiliser le modèle LLM ChatGPT d'OpenAI (commenté dans le code fourni).

- **from langchain_community.chat_models import HuggingFaceHub:** Importe la classe HuggingFaceHub de langchain_community pour utiliser un modèle LLM pré-entraîné de Hugging Face.
- **from langchain.memory import ConversationBufferMemory:** Importe la classe ConversationBufferMemory de langchain pour gérer l'historique des conversations.
- **from langchain.chains import ConversationalRetrievalChain:** Importe la classe ConversationalRetrievalChain de langchain pour créer la chaîne de conversation principale.
- **from htmlTemplates import css, bot_template, user_template:** Importe les modèles HTML pour la mise en forme de l'interface utilisateur (non détaillés dans le code fourni).
- **from langchain.llms import HuggingFaceHub:** Importe la classe HuggingFaceHub de langchain pour utiliser un modèle LLM pré-entraîné de Hugging Face.

➢ **Fonctions définies sur QA-langchain basée sur PDF :**

```
15  # Chemin du dossier contenant les fichiers PDF
16  pdf_docs =('C:/Users/mcs/Desktop/chatbot-Pdfs/fichier')
17
18  def get_pdf_text(pdf_folder):
19      text = ""
20      # Parcourir tous les fichiers dans le dossier spécifié
21      for filename in os.listdir(pdf_folder):
22          if filename.endswith(".pdf"):
23              # Construire le chemin complet du fichier PDF
24              pdf_path = os.path.join(pdf_folder, filename)
25              try:
26                  # Ouvrir le fichier PDF
27                  with open(pdf_path, "rb") as file:
28                      pdf_reader = PdfReader(file)
29
30                      # Lire chaque page du PDF
31                      for page in pdf_reader.pages:
32                          text += page.extract_text()
33              except Exception as e:
34                  print(f"Erreur lors de la lecture du PDF {pdf_path}: {e}")
35      return text
```

Figure 26 : fonction get_pdf_text(pdf_folder)

get_pdf_text(pdf_folder): extrait le texte de tous les fichiers PDF dans le dossier spécifié et le retourne comme une chaîne de caractères.

```python
def get_text_chunks(text):
    text_splitter = CharacterTextSplitter(
        separator="\n",
        chunk_size=1000,
        chunk_overlap=200,
        length_function=len
    )
    chunks = text_splitter.split_text(text)
    return chunks
```

Figure 27 : fonction get_text_chunks(text)

get_text_chunks(text): divise le texte extrait en morceaux plus petits (chunks) pour un traitement plus efficace.

```python
def get_vectorstore(text_chunks):
    embeddings = OpenAIEmbeddings()
    #embeddings = HuggingFaceInstructEmbeddings(model_name="sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2")
    vectorstore = FAISS.from_texts(texts=text_chunks, embedding=embeddings)
    return vectorstore

```

Figure 28 : fonction get_Vectorstores(text_chunks)

get_vectorstore(text_chunks): crée un stockage vectoriel qui est un élément essentiel pour la recherche efficace d'informations dans les documents PDF.

```python
def get_conversation_chain(vectorstore):
    llm = ChatOpenAI()
    #llm = HuggingFaceHub(repo_id="google/flan-t5-xxl", model_kwargs={"temperature":0.5, "max_length":512})

    memory = ConversationBufferMemory(
        memory_key='chat_history', return_messages=True)
    conversation_chain = ConversationalRetrievalChain.from_llm(
        llm=llm,
        retriever=vectorstore.as_retriever(),
        memory=memory
    )
    return conversation_chain
```

Figure 29: fonction get_conversation_chain(vectorstore)

get_conversation_chain(vectorstore): crée la chaîne de conversation principale du chatbot, en combinant un modèle LLM, le "vector store" et une mémoire de conversation.

```python
def handle_userinput(user_question):
    response = st.session_state.conversation({'question': user_question})
    st.session_state.chat_history = response['chat_history']

    for i, message in enumerate(st.session_state.chat_history):
        if i % 2 == 0:
            st.write(user_template.replace(
                "{{MSG}}", message.content), unsafe_allow_html=True)
        else:
            st.write(bot_template.replace(
                "{{MSG}}", message.content), unsafe_allow_html=True)

```

Figure 30 : fonction handle_userinput(user_question)

handle_userinput(user_question): gère la saisie de l'utilisateur, met à jour l'historique de la conversation et affiche la question de l'utilisateur et la réponse du chatbot.

- **Interface Streamlit et configuration principale :**

```python
def main():
    load_dotenv()
    st.set_page_config(page_title="Chat with multiple PDFs" , page_icon=":books:")
    st.header("Chat with multiple PDFs :books:")
    st.write(css, unsafe_allow_html=True)

    if "conversation" not in st.session_state:
        st.session_state.conversation = None

    user_question=st.chat_input("اطرح سؤالك هنا ")


    if user_question:
        handle_userinput(user_question)

    # Obtenir le texte des fichiers PDF dans le dossier
    raw_text = get_pdf_text(pdf_docs)


    # get the text chunks
    text_chunks = get_text_chunks(raw_text)

    # create vector store
    vectorstore = get_vectorstore(text_chunks)

    # create conversation chain
    st.session_state.conversation = get_conversation_chain(vectorstore)
```

Figure 31 : code de Streamlit

Dans notre code nous avons travaillé avec Streamlit qui gère l'affichage des éléments de l'interface utilisateur de notre application ,et permet d'afficher la réponse générée à l'utilisateur.

- Pour **htmlTemplates** qui fait la mise en forme et l'affichage des messages dans notre interface de conversation. Il définit des modèles HTML avec des styles CSS réutilisables pour personnaliser l'apparence de la conversation.

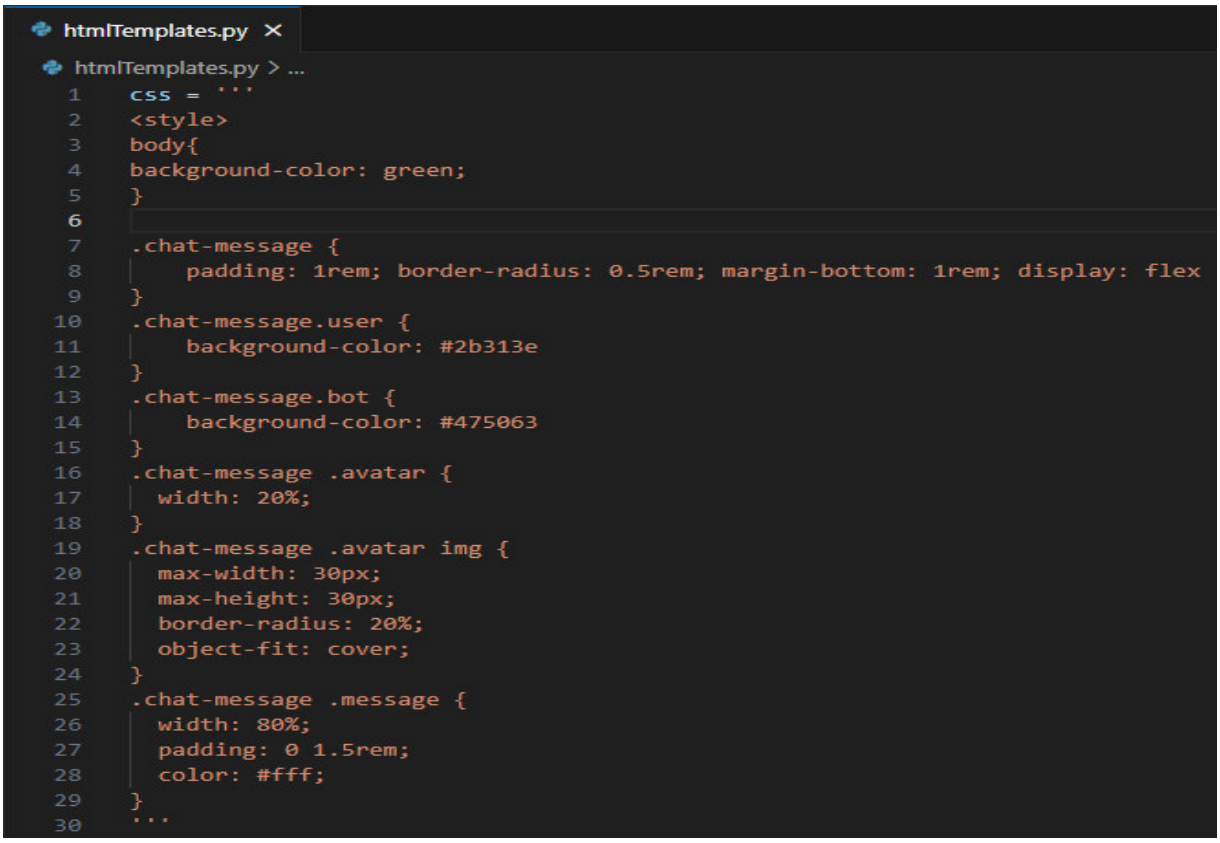


Figure 32 : code css pour streamlit

Figure 33 : code html pour streamlit

3.2.2 Création QA-LangChain avec LLMs :

L'intégration de LLMs dans LangChain offre un moyen puissant et flexible d'ajouter des fonctionnalités de traitement du langage naturel aux applications. Cela permet aux développeurs de créer des applications plus intelligentes et plus attrayantes sans avoir à se soucier des complexités des LLMs.

3.2.2.1 Les étapes QA-LangChain avec LLMs :

- **Représentations des messages :**

LangChain fournit des classes pour structurer les messages utilisateur et IA, facilitant la gestion du dialogue.

- **Génération d'utilisateurs :**

LangChain permet de construire des invites contextuelles pour le modèle d'IA, améliorant la pertinence des réponses.

- **Analyse des réponses :**

LangChain offre des outils pour analyser les réponses du modèle d'IA et les formater pour l'affichage à l'utilisateur.

- **Gestion de la conversation :**

LangChain facilite le suivi de l'historique de la conversation et la mise à jour du contexte pour des interactions fluides.

- **Gestion des erreurs :**

LangChain permet de gérer les erreurs et les exceptions de manière robuste, garantissant une expérience utilisateur stable.

3.2.2.2 Base de données de QA-LangChain avec LLMs :

Pour cette implémentation, il n'y a pas de base de données spécifique, c'est juste la base de données que le modèle contient, et si elle ne contient pas une réponse alors LangChain fait des recherches externes pour obtenir des réponses.

3.2.2.3 Code de de QA-LangChain avec LLMs :

- **Chargement des bibliothèques :**

```
1  import streamlit as st
2  from langchain_core.messages import AIMessage, HumanMessage
3  from langchain_openai import ChatOpenAI
4  from dotenv import load_dotenv
5  from langchain_core.output_parsers import StrOutputParser
6  from langchain_core.prompts import ChatPromptTemplate
7  import re
8
```

Figure 34 : les Bibliothèques utilisée

- **Import streamlit as st:**

  importe la bibliothèque Streamlit pour créer l'interface web.

- **langchain_core.messages.HumanMessage():**

  crée un objet HumanMessage pour représenter la saisie de l'utilisateur.

- **langchain_core.messages.AIMessage():**

  crée un objet AIMessage pour représenter les réponses de l'IA.

- **langchain_core.prompts.ChatPromptTemplate():**

  construit des invites pour le modèle d'IA en se basant sur des modèles.

- **langchain_openai.ChatOpenAI():**

  communique avec le modèle d'IA ChatGPT pour générer des réponses.

- **langchain_core.output_parsers.StrOutputParser():**

  analyse les réponses de l'IA en format de chaîne de caractères.

- **Import re :**

  fournit des outils pour travailler avec les expressions rationnelles en Python.

  Dans notre code elle vérifie si le texte de l'utilisateur est en arabe.

➢ **Fonctions définies :**

```
48  def is_arabic(text):
49      arabic_pattern = r"[\u0600-\u06FF\u0750-\u077F\u08A0-\u08FF]+"
50      return bool(re.search(arabic_pattern, text))
51
```

Figure 35 : la fonction is_arabic(text)

- Is_arabic(text) :

  cette fonction vérifie si un texte donné contient des caractères arabes.
  Si oui elle retourne True sinon elle retourne False

```
53
54  def get_response(user_query, chat_history):
55
56      template = """
57  أنت مساعد مفيد تم تدريبه خصيصًا للمحادثة باللغة العربية.
58  يرجى الرد على الأسئلة التالية باللغة العربية مع مراعاة سياق المحادثة:
59
60  سجل المحادثة: {chat_history}
61
62  سؤال المستخدم (باللغة العربية): {user_question}
63
64  يجب أن تكون الإجابة باللغة العربية:
65  """
66
67      prompt = ChatPromptTemplate.from_template(template)
68
69      llm = ChatOpenAI()
70
71      chain = prompt | llm | StrOutputParser()
72
73      return chain.stream({
74          "chat_history": chat_history,
75          "user_question": user_query,
76      })
77
```

Figure 36 : la fonction get_reponse(user_query , chat_history)

- get_response(user_query, chat_history) : cette fonction semble conçue pour interagir avec un LLM spécifiquement pour la conversation en arabe. Elle récupère l'historique de la conversation, formule une invite de discussion avec la question de l'utilisateur, l'envoie au LLM et récupère une réponse en arabe.

➢ **Interface Streamlit et configuration principale :**

```python
# session state
if "chat_history" not in st.session_state:
    st.session_state.chat_history = [
        AIMessage(content="مرحبا , كيف يمكنني مساعدتك ؟"),
    ]

# conversation
for message in st.session_state.chat_history:
    if isinstance(message, AIMessage):
        with st.chat_message("AI",avatar="static/quran.png"):
            st.write(message.content)
    elif isinstance(message, HumanMessage):
        with st.chat_message("Human",avatar="static/muslim.png"):
            st.write(message.content)

#user input
user_query = st.chat_input("اطرح سؤالك هنا ....")

if user_query is not None and user_query != "":
    if is_arabic(user_query):
        st.session_state.chat_history.append(HumanMessage(content=user_query))

        with st.chat_message("Human",avatar="static/muslim.png"):
            st.markdown(user_query)

        with st.chat_message("AI",avatar="static/quran.png"):

          response= st.write_stream(get_response(user_query, st.session_state.chat_history))

          st.session_state.chat_history.append(AIMessage(content=response))
    else:
        st.markdown(f"<div>{ereur}</div>",unsafe_allow_html=True)
```

```python
st.set_page_config(page_title="دليل الحكمة ", page_icon="static/read.png")


left_co, cent_co,last_co = st.columns(3)
with cent_co:
    st.image("dalil_hikma-removebg-preview.png", width=180)
```

Figure 37 : un petit code de interface streamlit

- Dans notre code nous avons travaillé avec Streamlit qui gère l'affichage des éléments de l'interface utilisateur de notre application, et permet d'afficher la réponse générée par le modèle de langage (GPT) à l'utilisateur.

- Ainsi, On a fait des modifications avec HTML et CSS pour donner une bonne interface à notre application.

```python
with open("style.css", "r") as f:
    css = f"<style>{f.read()}</style>"
    st.markdown(css, unsafe_allow_html=True)

ereur = """
<span class='ereur '>  ! يرجى إدخال السؤال باللغة العربية فقط</span>
"""
```

Figure 38 : le code pour affichez le message de erreur

4 Comparaison :

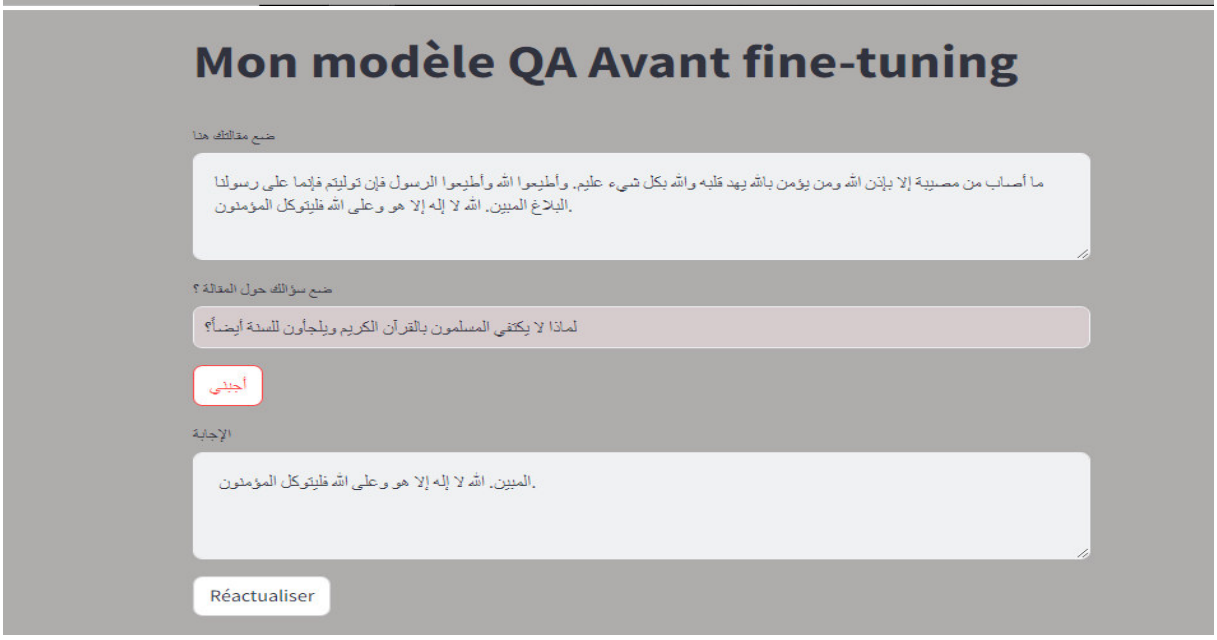
4.1 Comparaison entre modèle QA non fine-tunée modèle QA fine-tunée:

Nous avons fait une comparaison entre notre modèle Fine-Tuning avec notre model avant Fine-Tuning .Nous avons donné les mêmes passages(contextes) et les mêmes questions depuis notre base de données aux ces modèles . Les résultats étaient les suivants :

- **La première expérience**

La réponse juste c'est : " فمن شاء اتخذ إلى ربه مآبا **"**

- **La deuxième expérience:**



La réponse juste c'est : "و أطيعوا الله وأطيعوا الرسول **"**

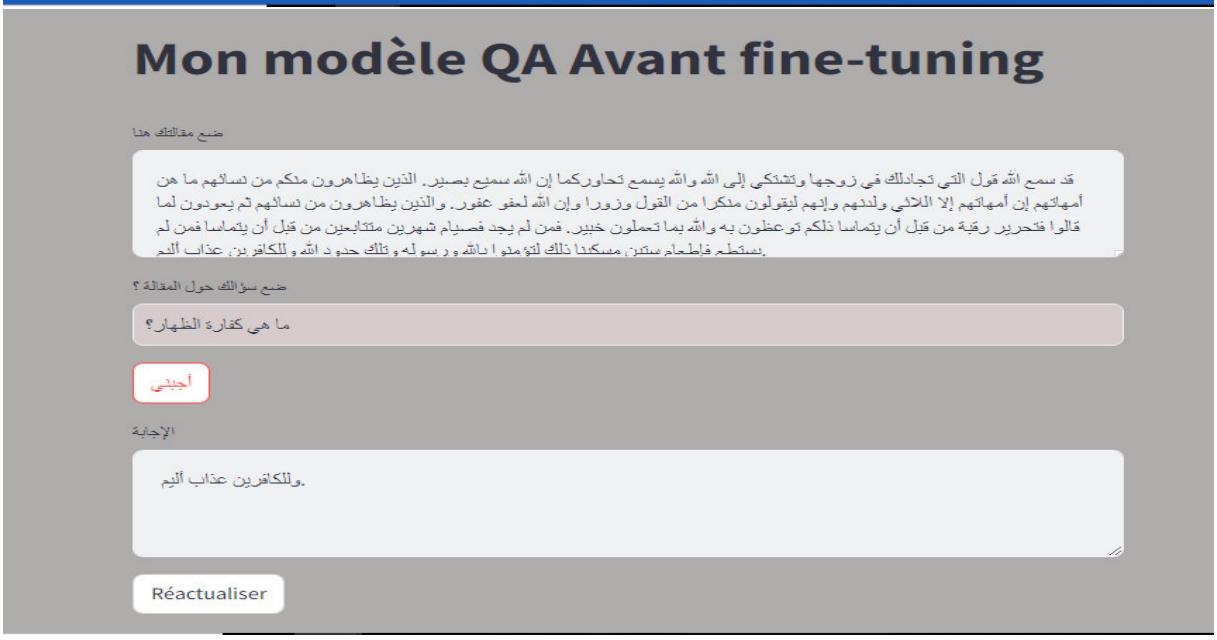
- Après les expériences que nous avons fait sur les deux modèles, on peut dire que notre modèle Fine-Tuning nous donne des résultats quasiment correctes, tandis que le modèle non entraîné donne des résultats aléatoires qui peuvent être corrects, mais qui sont souvent faux ou incomplets, ou il ne donne que la première phrase du contexte.
- Cela apparaît également dans le temps de réponse où notre modèle entraîné prend plus de temps que le modèle non entraîné pour donner un résultat précis.

4.2 Comparaison entre QA-Langchain sur PDF et modèle QA non fine-tunée:

Nous allons maintenant comparer notre modèle QA non Fine-Tuning avec la méthode de fusion PDF de LangChain.

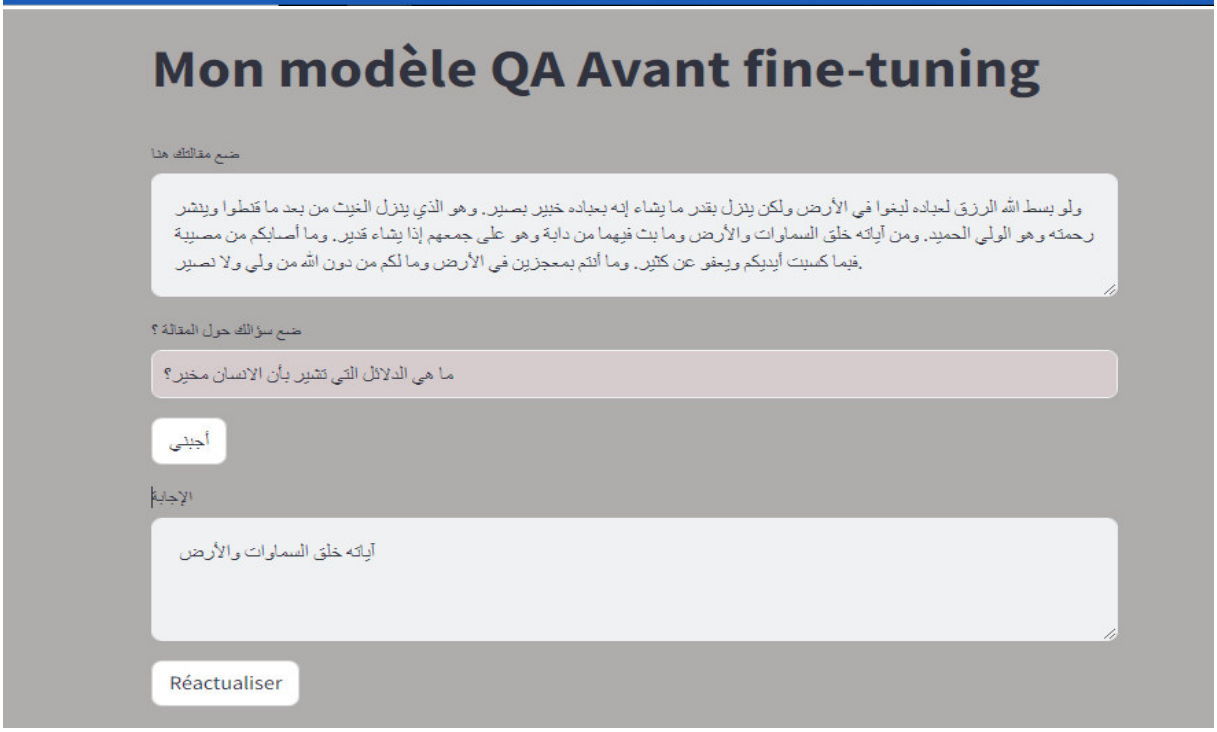
Pour le modèle QA non Fine-Tuning nous lui posons la question et donnons le contexte et pour la méthode de langchain nous lui donnons le contexte dans le pdf et lui posons la même question. Les résultats sont les suivants :

- **La première expérience :**



La réponse juste c'est : " تحرير رقبة من قبل أن يتماسا ذلكم توعظون به والله بما تعملون خبير. فمن لم يجد فصيام شهرين متتابعين من قبل أن يتماسا

- **La deuxième expérience :**



La réponse juste c'est : "ما أصابكم من مصيبة فبما كسبت أيديكم"

- Après les expériences que nous avons faites, nous pouvons dire que QA-LangChain est meilleur que le modèle QA dans le sens où il donne des réponses correctes et régulières dans la phrase, contrairement au modèle qui donne des réponses approximatives ou incorrectes.

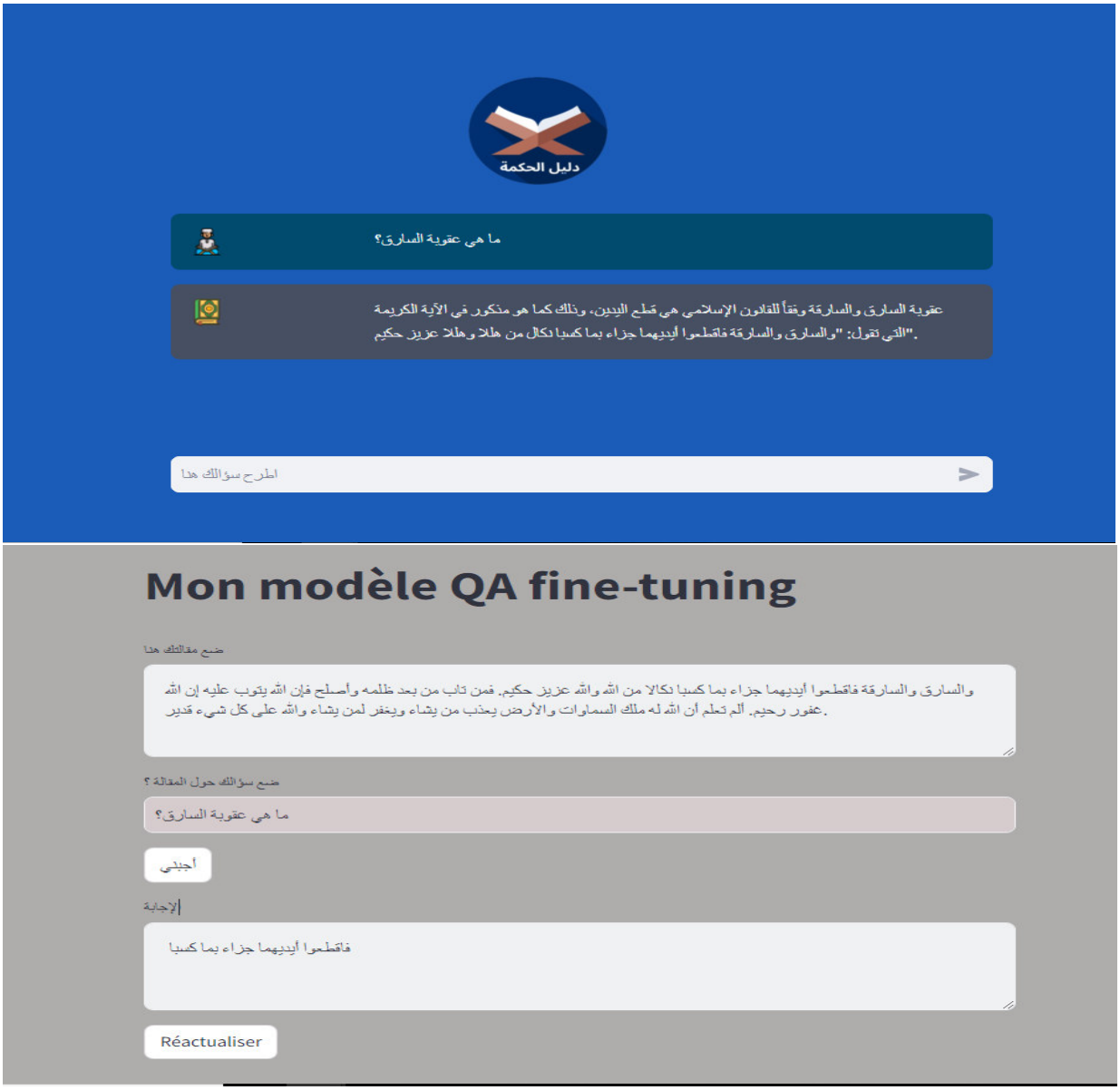
- De plus, en termes du temps de réponse. Sachant que le modèle QA réponde rapidement comme nous l'avons mentionné, mais LangChain nous donne une réponse plus rapide et régulière, ce qui le rend Parfait.

4.3 Comparaison entre QA-Langchain sur PDF et modèle QA fine-tunée:

Dans cette dernier partie , Nous allons comparer notre modèle fine-tunée avec la méthode d"intégration des PDFs dans LangChain.

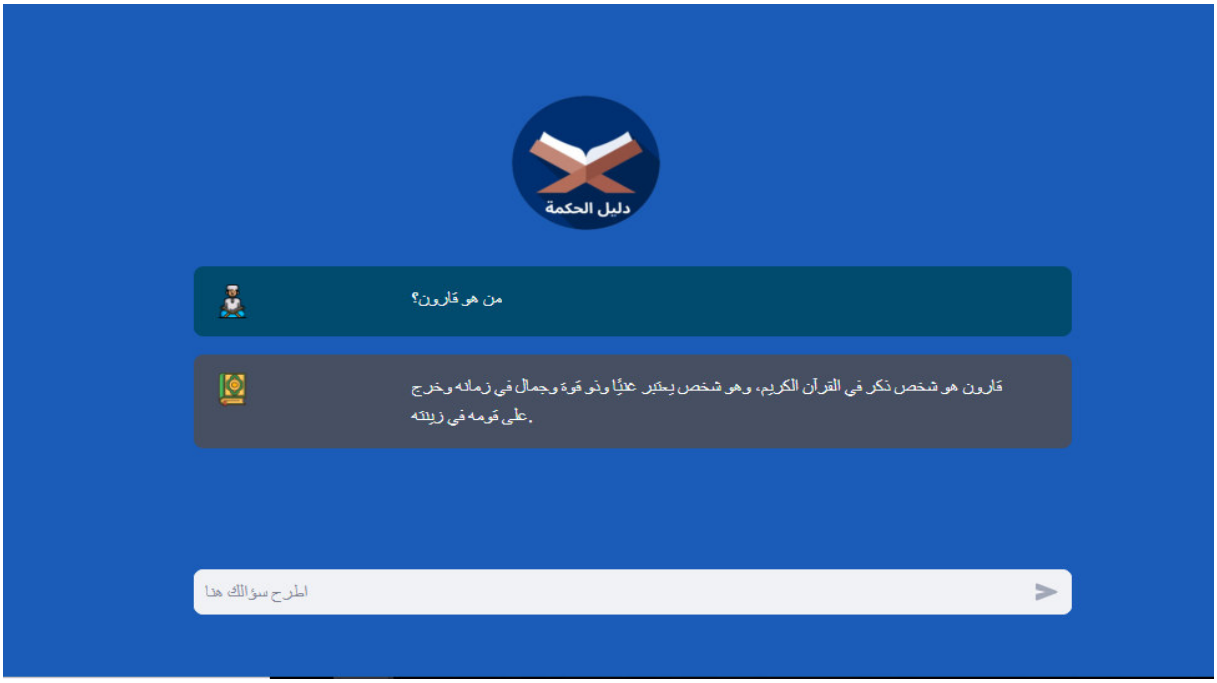
Pour notre modèle QA fine-tuneé, nous lui posons la question et donnons le contexte et pour la méthode de QA-LangChain basée sur pdf nous lui donnons le contexte dans le pdf et lui posons la question. Les résultats sont les suivants :

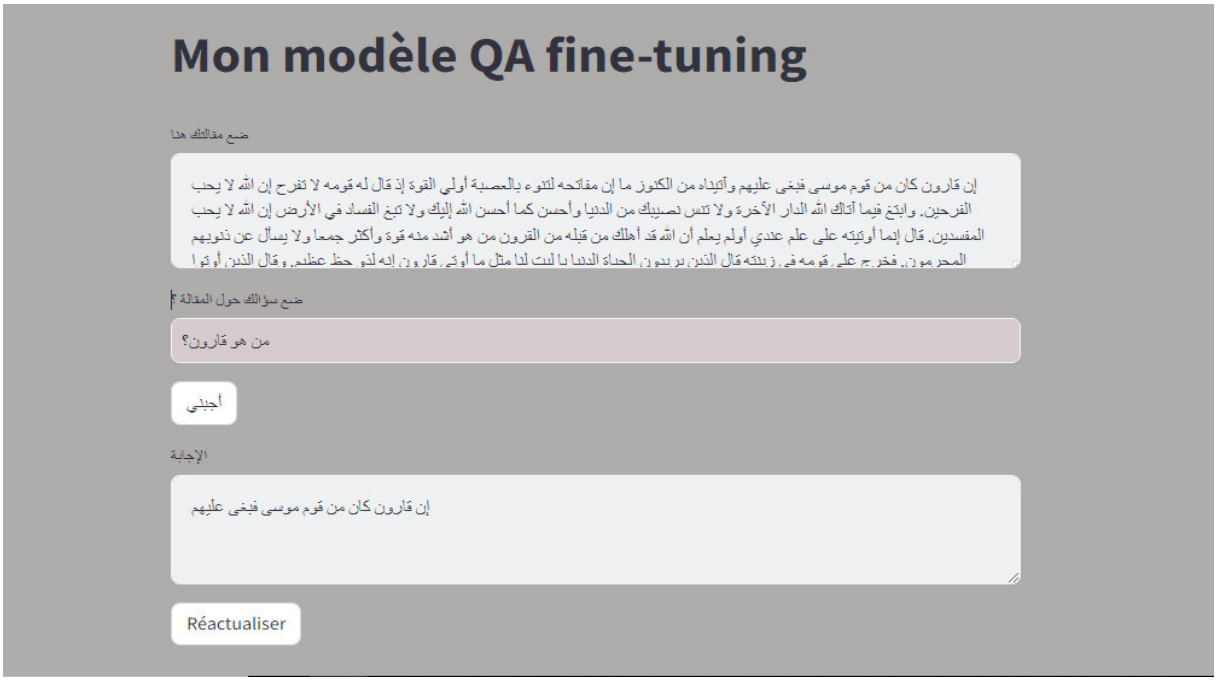
- **La première expérience :**



La réponse juste c'est :" السارق والسارقة فاقطعوا أيديهما جزاء بما كسبا **"**

- **La deuxième expérience :**





La réponse juste c'est : " إن قارون كان من قوم موسى فبغى عليهم " , " وآتيناه من الكنوز ما إن مفاتحه لتنوء بالعصبة أولي القوة", "قال له قومه لا تفرح إن الله لا يحب الفرحين", "قال إنما أوتيته على علم عندي"

- Après les expériences que nous avons faites, nous pouvons dire que notre QA-LangChain est meilleur que notre modèle fine-tunée dans le sens où il donne des réponses correctes et est plus régulier dans la phrase .Contrairement à notre modèle qui donne des réponses justes par rapport au modèle précédent mais parfois il donne des petites erreurs.

- De plus, en termes du temps de réponse . on peut dit que notre QA-LangChain est très rapide que notre modèle Fine-Tuning . mais les deux donne des bonnes réponses.

5 Conclusion :

Après ce chapitre, nous pouvons dire que la meilleure méthode de QA est LangChain, car elle donne toujours des réponses correctes et également dans une phrase régulière et le plus important c'est la rapidité de sa réponse, car elle donne la réponse plus rapide que les modèles QA existant et plus correcte qu'eux. Il suffit de lui fournir des fichiers pdfs qui contient des informations complètes et nécessaires.

D'un autre côté, cela ne signifie pas que les modèles QA existant sont inutiles ou inefficaces. Cependant, le modèle QA-Fine-Tuning donne des meilleurs résultats que le modèle normal. Il existe plusieurs facteurs pour choisir le meilleur modèle de langage d'assurance qualité pour une tâche particulière, tels que le type de questions auxquelles il faut répondre, la taille et la qualité de l'ensemble de données disponible, et les ressources de calcul disponibles.

Et le plus important, c'est de tester différents modèles et d'évaluer leurs performances sur un ensemble de données de validation avant de choisir un modèle pour une utilisation en production.

Conclusion générale

Dans cette recherche, nous avons proposé diverses méthodes pour résoudre le problème de l'accès aux informations religieuses en créant un système de chat automatisé qui permet à l'utilisateur de poser des questions directement et d'obtenir des informations rapidement, ce qui l'aide à approfondir sa compréhension de la religion.

Pour créer ce système, nous avons proposé deux approches différentes : la première est le fine-tuning et la seconde est LangChain, les deux méthodes se distinguant par leur efficacité et leur précision. Nous considérons que nous avons réussi à atteindre une grande partie de l'objectif de ce projet, malgré la contrainte de temps et la complexité de la langue arabe, et nous avons fait de bons choix en ce qui concerne les outils de mise en œuvre, faisant ainsi de notre travail un excellent point de départ pour d'autres projets futurs. Cependant, notre travail reste incomplet et peut être amélioré dans plusieurs domaines, notamment la création d'une base de données exhaustive sur l'Islam contenant toutes les informations religieuses afin de permettre un entraînement plus efficace des modèles.

Références

[1]https://www.researchgate.net/figure/Architecture-classique-dun-systeme-de-questions-reponses_fig1_231361280 [20/04/2024]

[2]https://www.intelligence-artificielle-school.com/ecole/technologies/abert-modele-nlp/

[3]https://www.researchgate.net/figure/Model-structure-of-BERT-and-GPT_fig1_354908597 [18/03/2024]

[4]https://databasecamp.de/en/ml-blog/t5-model

[5]https://paperswithcode.com/method/t5 [18/03/2024]

[6]https://www.bravrez.fr/article/intelligence-artificielle/article-les-transformateurs--expliques---comprendre-le-modele-derriere-le-gpt-3--le-bert-et-le-t5-2.html

[7]https://medium.com/@sharathhebbar24/t5-overview-05327690fa93 [20/03/2024]

[8]https://ludo-louis.fr/les-transformateurs-comprendre-modele-gpt-3-bert-t5/#google_vignette

[9]https://datascientest.com/generative-pretrained-transformer

[10] https://alberteam.com/gpt-tout-comprendre-sur-le-modele-dia-dopenai/

[11]https://pandia.pro/guide/quest-ce-que-gpt-generative-pre-trained-transformer-dans-le-domaine-de-lia/ [20/03/2024]

[12]https://hellofuture.orange.com/fr/le-modele-de-langage-gpt-3-revolution-ou-evolution/

[13]https://generativeai.pub/a-look-into-the-evolution-of-gpt-models-from-gpt-1-to-gpt-438b68c2f275b [20/03/2024]

[14] https://openreview.net/pdf?id=r1xMH1BtvB

[15]https://sh-tsang.medium.com/brief-review-electra-pre-training-text-encoders-as-discriminators-rather-than-generators-9568050d3a86 [21/03/2024]

[16]https://next.gr/news/news/290/the-best-nlp-papers-from-iclr-2020 [23/03/2024]

[17]https://openreview.net/pdf?id=r1xMH1BtvB

[18]https://ai-jobs.net/insights/xlnet-explained/

[19]https://www.researchgate.net/figure/Architecture-of-XLNet_fig2_352364036 [23/03/2024]

[20]https://www.researchgate.net/figure/Hyperparameters-Used-for-Fine-tuning-XLNet_tbl2_366981657 [25/03/2024]

[21] https://ai-jobs.net/insights/xlnet-explained/

[22]https://neildias.medium.com/language-model-catboost-a-study-in-an-nlp-competition-6e40d9a37c5d [25/03/2024]

[23]https://www.researchgate.net/figure/Hyperparameters-used-in-fine-tuning-RoBERTa-base-and-RoBERTa-large-with-LoRA_tbl3_373437741 [25/03/2024]

[24] https://towardsdatascience.com/roberta-1ef07226c8d8

[25]https://www.researchgate.net/figure/The-architecture-of-the-ALBERT-model-in-our-task_fig1_355434928 [26/03/2024]

[26]https://huggingface.co/datasets/tarteel-ai/quranqa/viewer/shared_task

[27]https://pandia.pro/guide/cest-quoi-langchain-tout-comprendre/

Résumé :

Le développement scientifique et technologique que connaît le monde a entraîné la disponibilité d'une grande quantité d'informations, ce qui a engendré certains problèmes pour l'homme, notamment en ce qui concerne la rapidité d'accès à ces informations. Par conséquent, ce projet vise à fournir un système de chat automatisé pour obtenir des informations rapidement et avec précision. L'idée principale de ce projet est de permettre à l'utilisateur de poser des questions religieuses et d'obtenir des réponses précises et rapides, sans avoir à se donner la peine de chercher dans de multiples sources.

Mots-clés :

chatbot ,nlp ,modele qa,gpt, bert,robirta, fine-tuning , langchain

Abstract:

The scientific and technological advancements in the world have resulted in the availability of large amounts of information, which has caused some problems for people, especially regarding the speed of accessing this information. Therefore, this project aims to provide an automated chat system to obtain information quickly and accurately. The main idea of this project is to enable the user to ask religious questions and receive precise and rapid answers, without the need to search through multiple sources.

Keywords:

Chatbot ,nlp ,model QA,gpt,bertroberta,fine-tuning,langchain

ملخص :

نـتج عـن التطـور العلمـي والتكنولـوجي الـذي يشـهده العـالم تـوفر كميـات كبيـرة مـن المعلومـات، ممـا أدى إلـى إحـداث بعـض المشـاكل للإنسـان، خاصـة فيمـا يتعلـق بسـرعة الحصـول علـى هـذه المعلومـات. لـذلك، يهـدف هـذا المشـروع إلـى تـوفير نظـام دردشـة مـع آلـة للحصـول علـى معلومـات سـريعة ودقيقـة. الفكـرة الرئيسـية مـن هـذا المشـروع هـي تمكـين المسـتخدم مـن طـرح أسـئلة دينيـة والحصـول علـى اجابـات دقيقـة و بسـرعة ، دون الحاجـة إلـى تكبـد عنـاء البحث في مصادر متعددة.

الكلمـــات الرئيســـية :شـــات بـــوت , معالجـــة اللغـــة الطبيعيـــة ,نمـــودج الاســـالة و الاجوبـــة ,جـــي بـــي تـــي ,بيرت,روبيرتا,التخصيص الدقيق ,لانغ شين