Soutenance SAE : ALOQAS

Intro : 1m Q
Pré Equipe : 1m L
GPT 2 : 3m L et Al (support)
Dataset : 2m S et Au
Fine-tuning : 5m Al et S (support?)
Gradio et Maquette : 3m O et Au
Optimisation : 3m : S ou Al (support)
Recommandation et Conclusion : 1m Q

Introduction :

1. Présentation du sujet

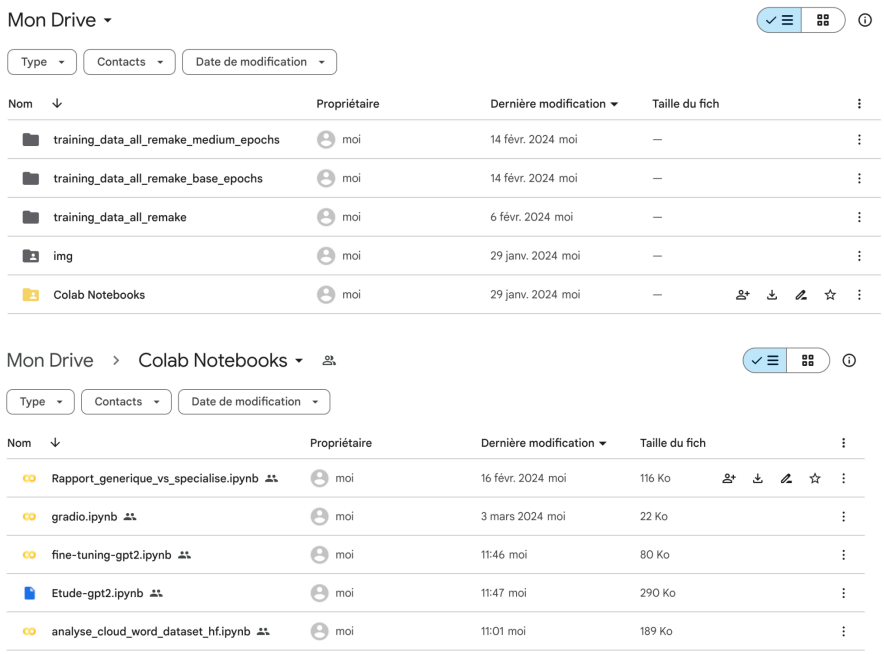
- L'objectif de la SAE est de développer un chatbot capable d'engager des conversations naturelles, de répondre aux questions.
- Le projet utilisera Keras NLP pour implémenter le modèle GPT-2.
- Gradio sera utilisé pour créer l'interface utilisateur.
- TensorFlow/Keras servira de backend pour le projet.
- Le développement se fera à l'aide de Jupyter Notebook et Google Colabs
- Le modèle sera entraîné avec le dataset “Scientific Papers” de Tensorflow.
- Le code et l'application Gradio seront hébergés sur un dépôt Hugging Face.

2. Plan :
    - Présentation de l’équipe et de l’organisation de travail
    - Présentation de Chat GPT2 et ses usages
    - Présentation du dataset et son usage
    - Présentation du Fine-tuning et son fonctionnement
    - Présentation de Gradio et son fonctionnement
    - Optimisation et problèmes rencontrés
    - Recommandation et Conclusion

Présentation Équipe :

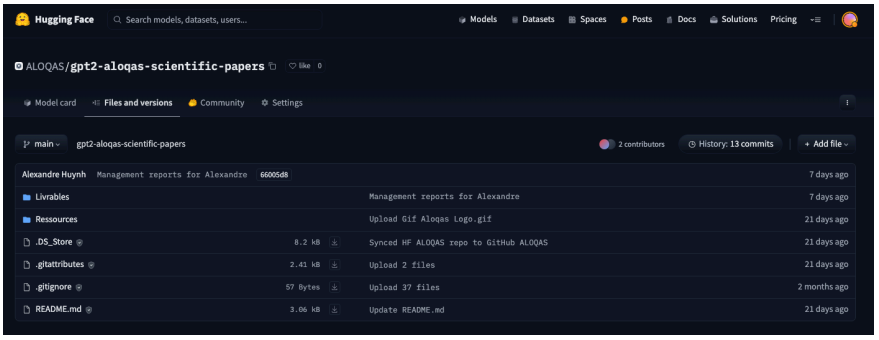
1. Organisation de l’équipe
    - Samuel : Fine-tuning et Analyse Dataset
    - Quentin : Fine-tuning
    - Lucas : GPT 2 + Github
    - Aurélien : Développeur Gradio + Cloud Word
    - Alexandre : GPT2 + Fine-tuning + Hugging face
    - Ony : Web Design et Gradio

2. Environnement de Dev
    - Google Colabs => Google Partagé : [Google Drive]

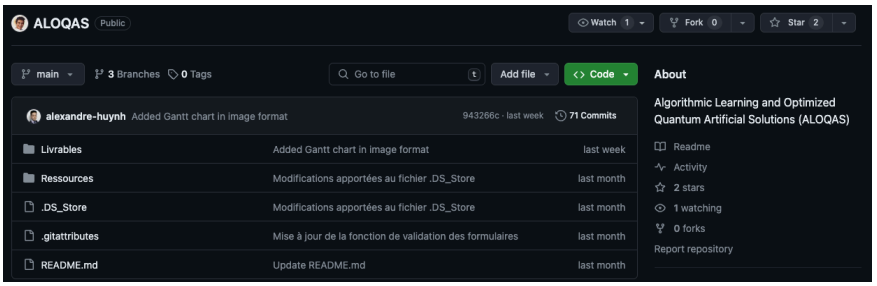


    - Hugging Face : [Hugging face]

Hugging Face est une plateforme qui permet de développer une plateforme d'intelligence artificielle axée sur le traitement du langage naturel, permettant aux développeurs de créer des applications utilisant le langage et la compréhension.



    - GitHub : [GitHub ALOQAS]



3. Organisation de travail
    - Discord
    - GitHub : https://github.com/LucasAguetai/ALOQAS
    - Hugging Face : https://huggingface.co/ALOQAS
    - Changement de manager tous les deux semaines => Rapport manager

Répartition du rôle de manager		
Samuel	6/11/2023	20/11/2023
Quentin	4/12/2023	18/12/2023
Aurelien	1/1/2024	15/1/2024
Alexandre	29/1/2024	12/2/2024
Ony	26/2/2024	11/3/2024
Lucas	25/3/2024	8/4/2024

    - Rapport individuel et collectif

Présentation du Dataset :

1. Présentation du dataset :
    a. Présentation du dataset Scientifique Papers :

    - Le dataset est distribué par Tensorflow, qui est une bibliothèque open source développée par Google pour l'apprentissage et l'intelligence artificielle.
    - Le dataset comprend deux archives : l'archive ArXiv et l'archive PubMed :
        - ArXiv contient plus de 200 000 exemples d'entraînement couvrant des domaines comme la science, les mathématiques, la biologie et la finance. (7.5 GO)
        - PubMed, avec plus de 120 000 exemples d'entraînement, se concentre sur le domaine médical et la biologie. (2.51 GO)

scientific_papers/arxiv (default config)

- **Config description**: Documents from ArXiv repository.
- **Dataset size**: `7.07 GiB`
- **Splits**:

Split	Examples
`'test'`	6,440
`'train'`	203,037
`'validation'`	6,436

- **Examples** (tfds.as_dataframe):

Display examples...

scientific_papers/pubmed

- **Config description**: Documents from PubMed repository.
- **Dataset size**: `2.34 GiB`
- **Splits**:

Split	Examples
`'test'`	6,658
`'train'`	119,924
`'validation'`	6,633

- **Examples** (tfds.as_dataframe):

Display examples...

- L'archive PubMed a été choisie pour faciliter l'importation des données et orienter le chatbot vers le domaine médical.
- Les données sont structurées en arrays de dictionnaires contenant les clés 'abstract', 'article', et 'section_names'.

Feature	Class	Shape	Dtype	Description
	FeaturesDict			
abstract	Text		string	
article	Text		string	
section_names	Text		string	

	abstract	article	section_names
0	backgroundarteriovenous malformations (avms) with vascular abnormalities , including aneurysms , have been reported frequently . however , the coexistence of avm and unilateral moyamoya disease is rare . we report herein an avm patient who presented with acute ischemic stroke with unilateral moyamoya disease and occlusion of the feeding artery.case reporta-41-year old man was admitted with sudden dysarthria and facial palsy . brain computed tomography and magnetic resonance imaging revealed an acute infarction adjacent to a large avm in the right frontal lobe . cerebral angiography revealed occlusions of the proximal right middle cerebral and proximal anterior cerebral arteries , which were the main feeders of the avm . innumerable telangiectatic moyamoya - type vessels between branches of the anterior cerebral artery and dilated lenticulostriate arteries on the occluded middle cerebral artery were detected . however , a nidus of the avm was still opacified through the distal [...]	arteriovenous malformations (avms) with vascular abnormalities , including aneurysms , have been reported frequently . however , the coexistence of avm and unilateral moyamoya disease is rare . we report herein an avm patient who presented with acute ischemic stroke with unilateral moyamoya disease and occlusion of the feeding artery . brain computed tomography and magnetic resonance imaging revealed an acute infarction adjacent to a large avm in the right frontal lobe . cerebral angiography revealed occlusions of the proximal right middle cerebral and proximal anterior cerebral arteries , which were the main feeders of the avm . innumerable telangiectatic moyamoya - type vessels between branches of the anterior cerebral artery and dilated lenticulostriate arteries on the occluded middle cerebral artery were detected . however , a nidus of the avm was still opacified through the distal right callosomarginal artery , which was supplied by the remaining anterior cerebral artery [...]	Background Case Report Conclusions Introduction Case Report Discussion

- Pour l'utilisation, le dataset est accessible via la fonction load_dataset et divisé en trois parties : entraînement, test et validation.

```
from datasets import load_dataset
```

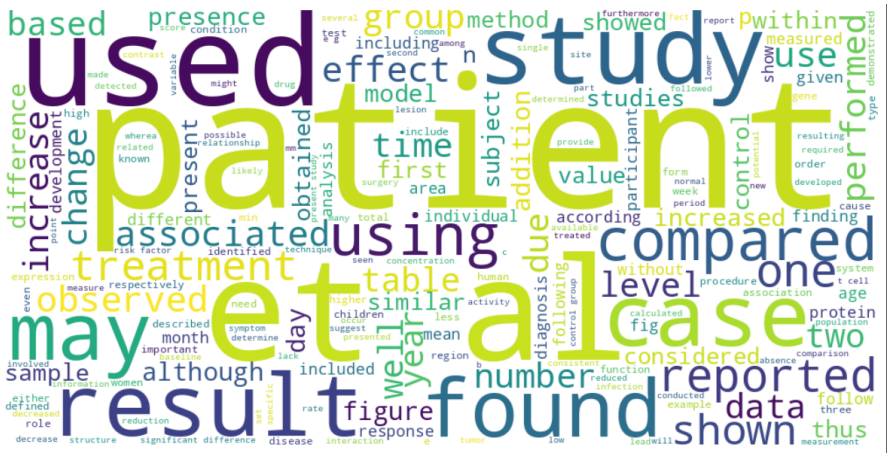
```
[ ] dataset = load_dataset("scientific_papers", 'pubmed')


    train_dataset = dataset["train"]
    test_dataset = dataset["test"]
    val_dataset = dataset["validation"]
```

b. Analyse du dataset Scientifique Papers :

- L'analyse du dataset inclut des nuages de mots pour chaque segment (entraînement, test, validation) et identifie les x mots les plus fréquents.
- Lorsque l'on manipule des données, il est primordial de les analyser. Pour
- réaliser cette analyse, on peut, par exemple, créer un "cloudword" (nuage de mots) . Un "cloudword" est une représentation visuelle des données : plus un mot a d'occurrences dans un texte, plus il apparaît gros dans le "cloudword". Cela permet de voir rapidement quels sont les mots les plus utilisés et d'avoir un premier aperçu du ou des sujets principaux.

```
[ ] worldCloudTrain = WordCloud(width=1000, height=500,background_color='white', colormap='viridis').generate(textTrain)
    worldCloudTest = WordCloud(width=1000, height=500,background_color='white', colormap='viridis').generate(textTest)
    worldCloudVal = WordCloud(width=1000, height=500,background_color='white', colormap='viridis').generate(textVal)

[ ] print("Nuage de Mots : Train")
    displayWordCloud(worldCloudTrain)
    print("Nuage de Mots : Test")
    displayWordCloud(worldCloudTest)
    print("Nuage de Mots : Val")
    displayWordCloud(worldCloudVal)
```

- Comme l'a expliqué Samuel plus tôt, le dataset est séparé en 3 parties : entraînement, test et validation. L'objectif serait donc d'avoir les 3 "cloudwords" assez similaires. Sinon, cela pourrait signifier que nos datasets sont trop différents des autres, ce qui peut poser problème pour entraîner ou tester le fine-tuning.



- Un nuage de mots est bien pratique, mais on ne voit pas combien de fois les mots apparaissent. Pour remédier à cela, on peut utiliser une bibliothèque : matplotlib.pyplot (elle est utilisée pour créer des graphiques), et créer une fonction pour nous afficher le nombre d'occurrences de chaque mot.

Présentation de Chat GPT2 : Lucas et Alexandre

a. Présentation de Chat GPT2 :

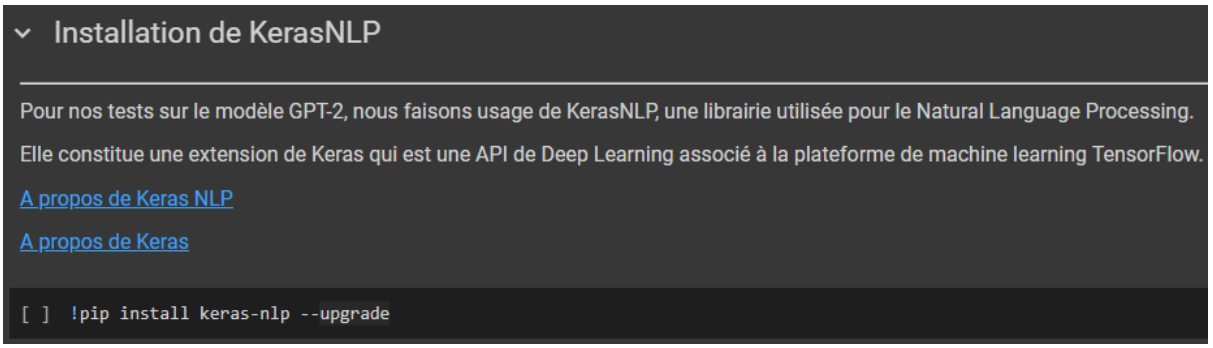
- Modèle GPT2 (Generative Pre-trained Transformer 2 (GPT-2)
  - LLM (Large Language Model)
  - modèle deep learning par OpenAI
    - apprentissage automatique
    - utilisation de réseaux de neurones, interpret plusieurs couches
    - ajuster ses paramètres lors entraînement
  - traitement langage naturel / textuel
  - pré-entraîné sur 8 millions de pages Web

- GPT2 = LLM spécialisée, génération de texte à partir de prompt
  - prédire élément suivant, compléter / continuer la phrase
  - garder cohérence, structure, logique, style écriture

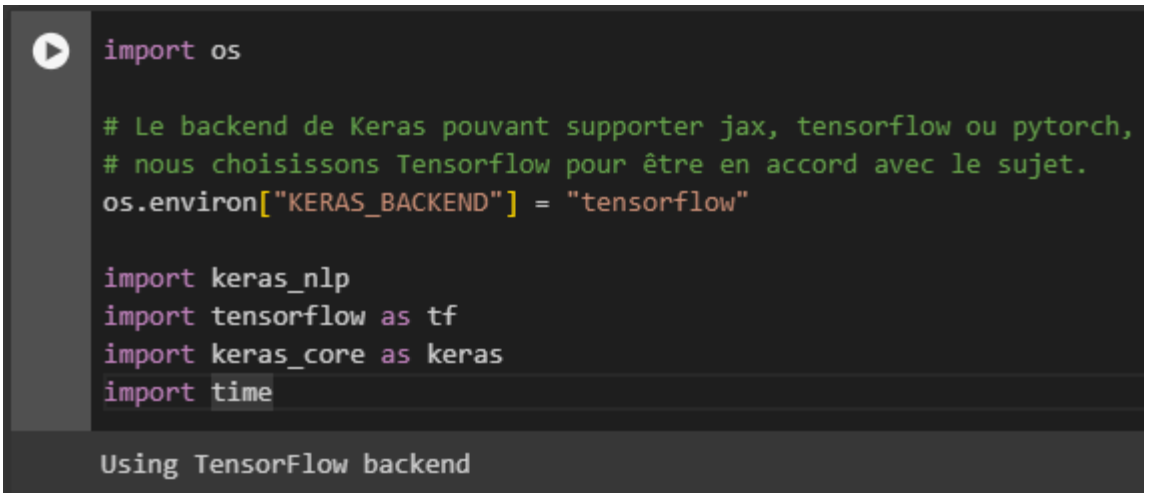
b. Présentation de Chat GPT2 :

- Étude du modèle et première experience
  - import + recherche bibliothèques Keras / KerasNLP
    - Keras = librairie deep learning API
      - couches + models (Sequential
      - faire des models from scratch
    - KerasNLP = librairie spécial process langage naturel
      - couches + models

INSTALL KERASNLP



IMPORT TENSORFLOW ET AUTRE



- modèles tailles diff : base / medium / large

DIFFÉRENTS MODÈLES GPT2

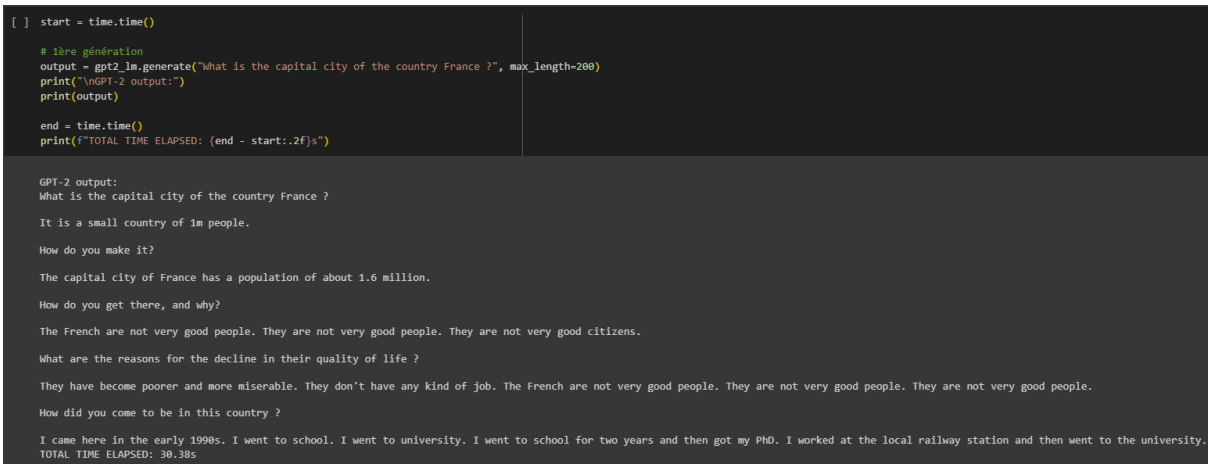
Preset name	Parameters	Description
gpt2_base_en	124.44M	12-layer GPT-2 model where case is maintained. Trained on WebText.
gpt2_medium_en	354.82M	24-layer GPT-2 model where case is maintained. Trained on WebText.
gpt2_large_en	774.03M	36-layer GPT-2 model where case is maintained. Trained on WebText.
gpt2_extra_large_en	1.56B	48-layer GPT-2 model where case is maintained. Trained on WebText.
gpt2_base_en_cnn_dailymail	124.44M	12-layer GPT-2 model where case is maintained. Finetuned on the CNN/DailyMail summarization dataset.

- test initialisé GPT2 base
- tests de génération :
    - complétion texte gén + technique
    - questions gén + science / tech

INITIALISATION GPT2

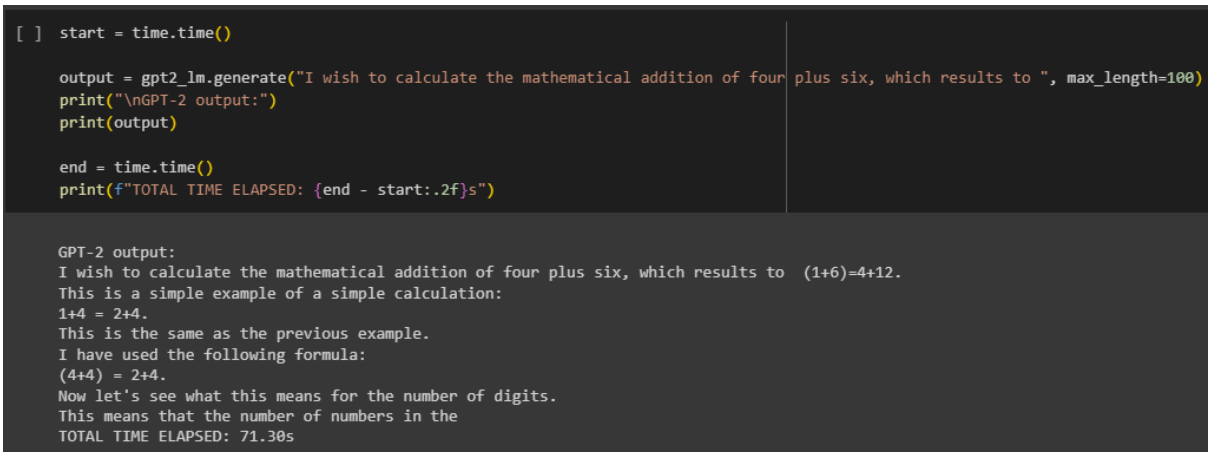
```
# To speed up training and generation, we use preprocessor of length 128
# instead of full length 1024.
preprocessor = keras_nlp.models.GPT2CausalLMPreprocessor.from_preset(
    "gpt2_base_en",
    sequence_length=128,
)

gpt2_lm_medium = keras_nlp.models.GPT2CausalLM.from_preset(
    "gpt2_base_en", preprocessor=preprocessor
)
```

TEST QUESTION GENERIQUE : What is the capital city of the country France ?



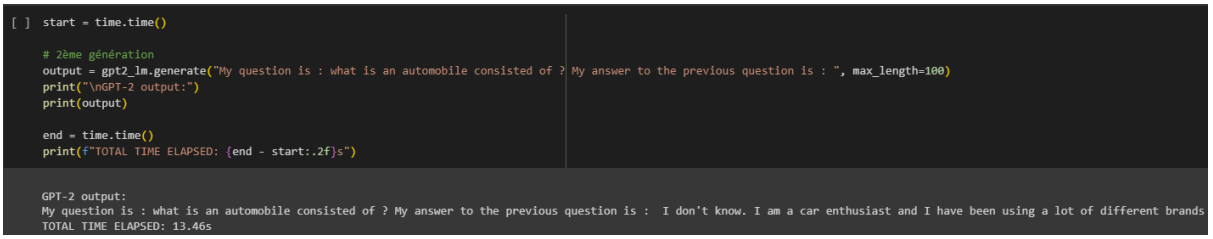
```
[ ] start = time.time()

    # 1ère génération
    output = gpt2_lm.generate("What is the capital city of the country France ?", max_length=200)
    print("\nGPT-2 output:")
    print(output)

    end = time.time()
    print(f"TOTAL TIME ELAPSED: {end - start:.2f}s")

GPT-2 output:
What is the capital city of the country France ?

It is a small country of 1m people.

How do you make it?

The capital city of France has a population of about 1.6 million.

How do you get there, and why?

The French are not very good people. They are not very good people. They are not very good citizens.

What are the reasons for the decline in their quality of life ?

They have become poorer and more miserable. They don't have any kind of job. The French are not very good people. They are not very good people. They are not very good people.

How did you come to be in this country ?

I came here in the early 1990s. I went to school. I went to university. I went to school for two years and then got my PhD. I worked at the local railway station and then went to the university.
TOTAL TIME ELAPSED: 30.38s
```

TEST LOGIQUE : Opération mathématique



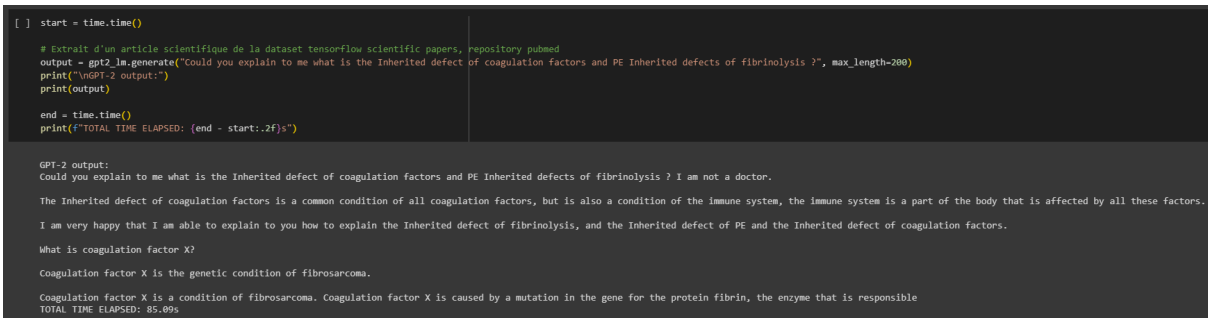
```
[ ] start = time.time()

    output = gpt2_lm.generate("I wish to calculate the mathematical addition of four plus six, which results to ", max_length=100)
    print("\nGPT-2 output:")
    print(output)

    end = time.time()
    print(f"TOTAL TIME ELAPSED: {end - start:.2f}s")

GPT-2 output:
I wish to calculate the mathematical addition of four plus six, which results to  (1+6)=4+12.
This is a simple example of a simple calculation:
1+4 = 2+4.
This is the same as the previous example.
I have used the following formula:
(4+4) = 2+4.
Now let's see what this means for the number of digits.
This means that the number of numbers in the
TOTAL TIME ELAPSED: 71.30s
```

TEST QUESTION CONNAISSANCE TECHNIQUE : Composition automobile



```
[ ] start = time.time()

    # 2ème génération
    output = gpt2_lm.generate("My question is : what is an automobile consisted of ? My answer to the previous question is : ", max_length=100)
    print("\nGPT-2 output:")
    print(output)

    end = time.time()
    print(f"TOTAL TIME ELAPSED: {end - start:.2f}s")

GPT-2 output:
My question is : what is an automobile consisted of ? My answer to the previous question is :  I don't know. I am a car enthusiast and I have been using a lot of different brands
TOTAL TIME ELAPSED: 13.46s
```

TEST QUESTION SCIENTIFIQUE - SUJET DATASET PUBMED : Could you explain to me what is the Inherited defect of coagulation factors and PE Inherited defects of fibrinolysis ?



```
[ ]  start = time.time()

     # Extrait d'un article scientifique de la dataset tensorflow scientific papers, repository pubmed
     output = gpt2_lm.generate("Could you explain to me what is the Inherited defect of coagulation factors and PE Inherited defects of fibrinolysis ?", max_length=200)
     print("\nGPT-2 output:")
     print(output)

     end = time.time()
     print(f"TOTAL TIME ELAPSED: {end - start:.2f}s")

GPT-2 output:
Could you explain to me what is the Inherited defect of coagulation factors and PE Inherited defects of fibrinolysis ? I am not a doctor.

The Inherited defect of coagulation factors is a common condition of all coagulation factors, but is also a condition of the immune system, the immune system is a part of the body that is affected by all these factors.

I am very happy that I am able to explain to you how to explain the Inherited defect of fibrinolysis, and the Inherited defect of PE and the Inherited defect of coagulation factors.

What is coagulation factor X?

Coagulation factor X is the genetic condition of fibrosarcoma.

Coagulation factor X is a condition of fibrosarcoma. Coagulation factor X is caused by a mutation in the gene for the protein fibrin, the enzyme that is responsible
TOTAL TIME ELAPSED: 85.09s
```

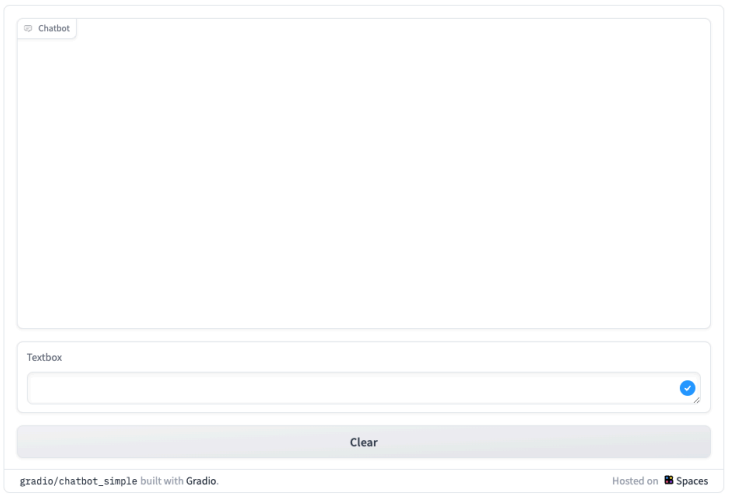
- <u>constat étude</u>
  - modèle générique, certain degré de connaissance
  - difficultés répondre questions techniques, médicales, science.

Présentation de Gradio : Ony et Aurélien

a. Présentation de Gradio :

- Explication de Gradio
  - bibliothèque open-source facilitant la création d'interfaces utilisateur pour les modèles d'apprentissage automatique et les systèmes d'IA.

- <u>Lecture de Documentation</u> : https://www.gradio.app/docs/interface
- Gradio est une bibliothèque open-source pour simplifier la création d'interfaces utilisateur pour l'IA. Voici ses points clés :
  - Création rapide d'UI: Permet de développer des applications web interactives sans expertise en développement web.



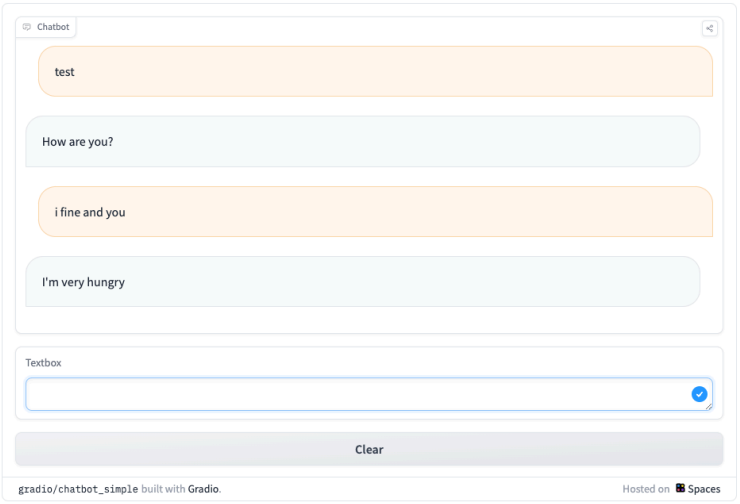
- Layout flexible: Organise les éléments UI en lignes ou colonnes pour une expérience utilisateur optimisée.
- Bouton et Event Listener: Intègre des boutons pour déclencher des actions et écoute les événements pour une interaction dynamique.

```
    def respond(message, chat_history):
        bot_message = random.choice(["How are you?", "I love you", "I'm very hungry"])
        chat_history.append((message, bot_message))
        time.sleep(2)
        return "", chat_history

    msg.submit(respond, [msg, chatbot], [msg, chatbot])

if __name__ == "__main__":
    demo.launch()
```

b. Usage et Fonctionnement :

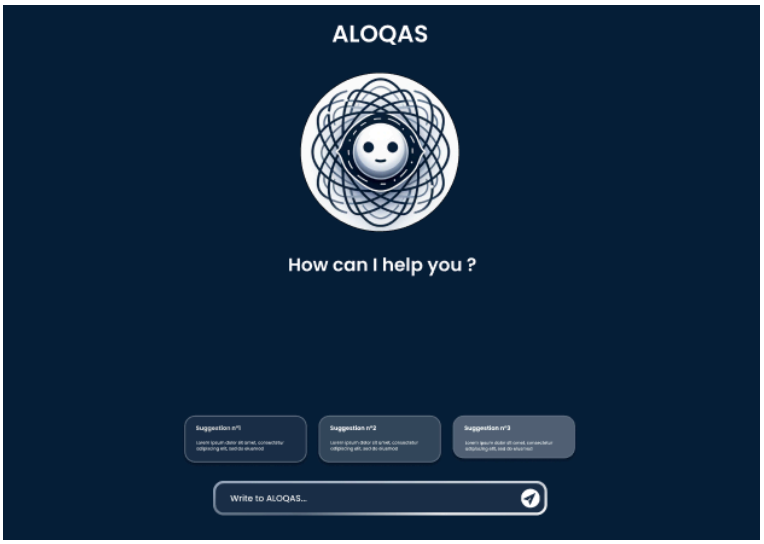
- Fonctionnement: Associe des fonctions aux entrées (inputs) et sorties (outputs) pour exécuter et afficher les résultats du modèle d'IA.

- Interface graphique constitué d'un chatbot avec un champ de texte pour poser une question au model. Avec différentes questions, suggestions sous forme de boutons pour poser des questions pré-établi. Avec une zone pour afficher.



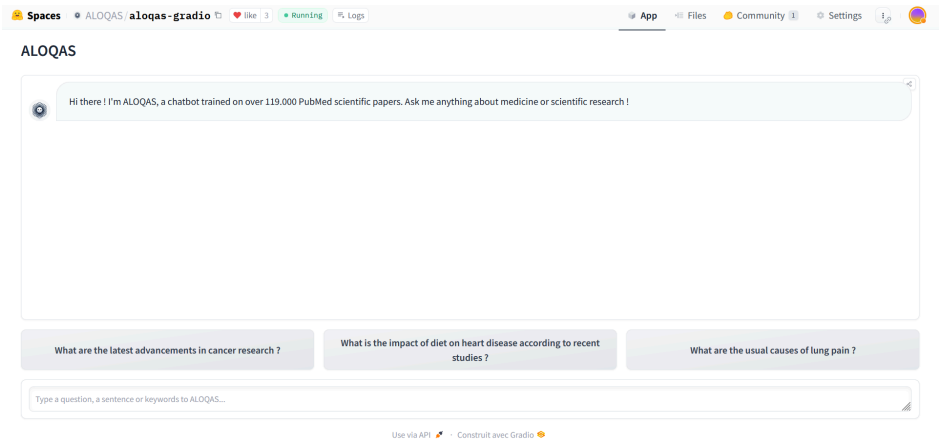
c. Prototypage de la maquette du chatbot :

- Maquette du chatbot fait avec Figma : [Figma Maquette]



d. Première version du chatbot (sans CSS/Thème) :

- Lors des premières étapes de développement avec Gradio, nous avons créé un premier jet de notre chatbot sans nous soucier de l'apparence visuelle. Nous avons utilisé les composants prédéfinis de Gradio pour construire une interface fonctionnelle. Bien que le résultat soit fonctionnel, l'apparence de l'interface était rudimentaire et manquait d'esthétisme.



e. Deuxième version du chatbot (Theme Builder + ajout de CSS/Thème) :

- Pour améliorer l'apparence visuelle de notre chatbot, nous avons utilisé deux méthodes principales : le "Theme Builder" de Gradio et des styles CSS personnalisés.

- Le "Theme Builder" de Gradio est un outil intégré qui permet de personnaliser facilement l'apparence de notre application. Avec cet outil, nous avons eu la flexibilité de choisir parmi une variété de thèmes prédéfinis ou de créer notre

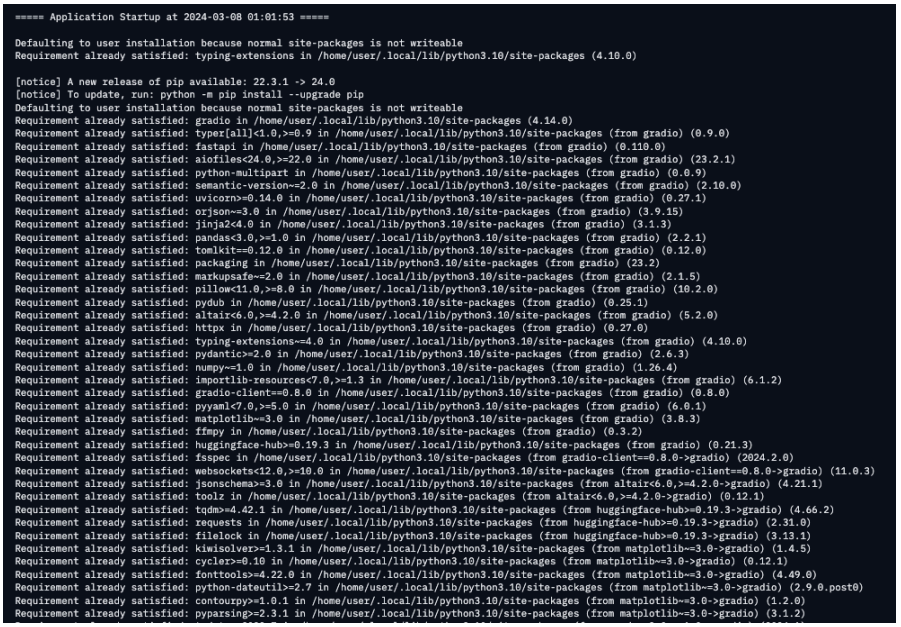
propre thème personnalisé. Par exemple, nous avions défini notre thème de base en spécifiant la couleur principale de notre choix (slate = bleue).

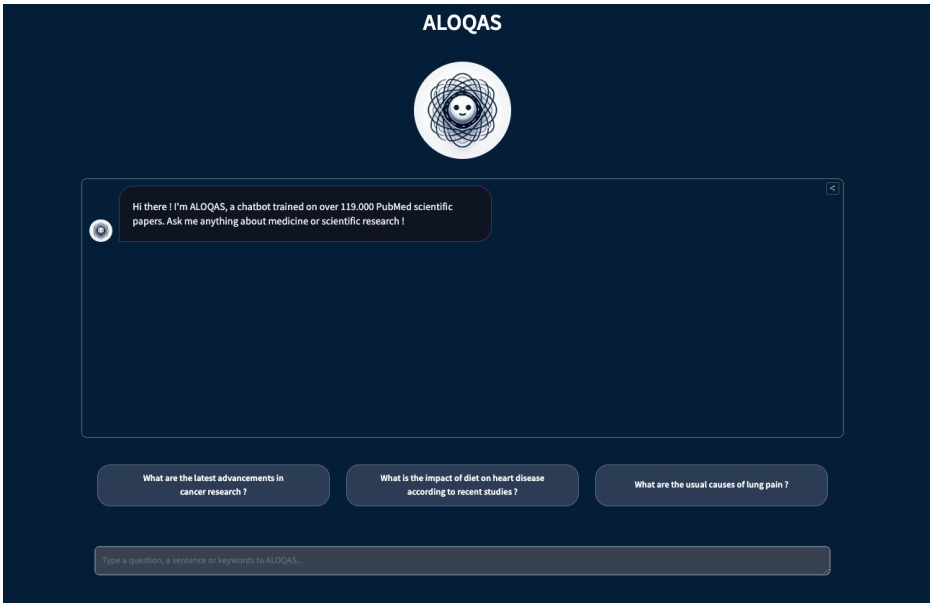
- Nous avons également utilisé des styles CSS personnalisés pour ajuster les couleurs des différents éléments, les bordures, les avatars du chatbot et de l'utilisateur, etc. de notre interface utilisateur, ce qui nous a donné un contrôle précis sur l'apparence de notre chatbot.

f. Déploiement de Gradio sur Hugging Face :

- Model : Dépot du Git avec les fichiers du chatbot
  - App.py => Code source du model et gradio
  - Checkpoint => Paramètres/poids du model provenant du fin-tunning du model avec le dataset

- Space : Production du chatbot
- Push et Déploiement : A chaque push, un script du container Hugging face





g. Retour Feedback des utilisateurs.

- Nous avons sollicité des retours sur notre interface auprès de différents utilisateurs. Un étudiant en 3ème année de BUT Informatique a apprécié le visuel mais a noté le manque d'intuitivité pour poser une question en anglais. De plus, il a remarqué l'absence de signes indiquant la fin de la génération de texte, et que la zone de saisie reste affichée après l'envoi d'un message. Un autre utilisateur, qui est diplômé d'une école d'ingénieur, a apprécié le design simple mais a noté le manque de conclusion claire dans les textes générés.

Utilisateur	Impressions générale	Remarques et points à améliorer	Questions posées
Etudiant en BUT3 Informatique Formation Initiale (sans notions de machine learning)	- Apprécie en général le visuel de l'interface, format Chatbot messages question-réponse - Fonctionnalité Gradio affichant le temps pris pour la génération de texte, montrant que le processus est en cours d'exécution	- Disposition message inversée (question user à gauche, réponse bot à droite) - Manque intuitivité qu'il faut poser une question (a saisi bonjour, english, ne savait pas que celle ci devait être en anglais) - Signe ou ponctuation manquant pouvant indiquer la fin génération de texte, impression que le texte n'était pas terminé - Impression qu'il y a seulement les suggestions de texte comme options, manque d'intuitivité qu'il y ait une zone de saisie (logo trop grand, besoin de dérouler la page pour trouver la zone de saisie)	

		- Zone de saisie n'étant pas effacé après la saisie et validation/envoi d'un prompt	
Etudiant en école d'ingénieur majorant en cybersécurité (avec des notions en machine learning)	- Apprécie le design "simple, épuré et efficace" - Satisfait de la qualité des résultats avec des questions saisies par l'utilisateur	- Impression que la dernière phrase du texte n'est pas terminée, malgré la présence d'un point final de ponctuation	- Is it serious to have diabetes?
Développeur back-end (sans notions en machine learning)	- A beaucoup apprécié l'interface ALOQAS comparé aux autres modèles	- Manque d'intuitivité dans l'interface : peu évident qu'il faut défiler la page pour trouver la zone de saisie dû à la taille importante du logo	
	- Réponse de qualité moyenne, données anciennes - Interface était pas assez médical (trop sombre)	- Suggestion, "lung pain" est peu évocateur (douleurs pulmonaires), mieux de mettre disease	- Question pourcentage survie cancer du sein - Réponse pourcentage de mort - Mais donnée pas à jour - ALOQAS 30% mort - Internet 12% mort - Stage

In the previous issue of critical care, van der linderen and colleagues report that the prevalence of coronary heart disease (chd) in the general population is about 20%, which is higher than that of the general population. The authors also found that a higher prevalence of coronary artery disease is associated with a lower risk of myocardial infarction and death. It was reported that the incidence of heart disease is about 1. 5 times greater in the general population, which indicates higher prevalence of coronary artery disease among the patients of the.

Fine tuning du Model : Samuel et Alexandre

a. Présentation et Usage du Fine-Tuning :

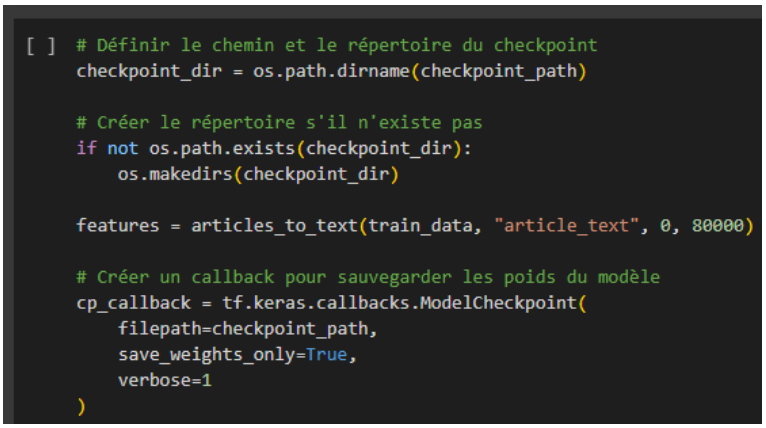
- Définition fine tuning
    - Entraîner un modèle de deep learning avec un dataset

- Usage
    - spécialiser un modèle pré entraîné (GPT2, CIFAR100) sur une tâche spécifique
    - fournir jeu de données, apprendre en modifiant poids couches
    - ici, spécialiser GPT2 basique sur tâche générique sur des textes orienté médical.
    - fournir un jeu de données articles scientifiques

b. Prise en main du fine tuning :

- reprenant import + init GPT2 + compilation
- entrainement via fit
- comment fournir texte à fit ?
    - possible fournir strings / un tableau de strings (doc)
    - fonction Python articles to texte
- première fine tuning avec bcp de crash mémoire
    - fine tuning epoch 1 en plusieurs étapes
    - échantillon progressif : 1k → 10k → 80k → totalité
    - sur chaque article, apprendre mots utilisés, relations entre les mots
        - tokenisation des mots
        - altérer les poids couches
        - baser sur fonction loss
        - mesure différence préd model vs réalité data
        - jusqu'à entrain tout 80k
    - bcp de crash mémoire

FINE TUNING Premieres expérimentation fine tuning



```
[ ]  # Définir le chemin et le répertoire du checkpoint
     checkpoint_dir = os.path.dirname(checkpoint_path)

     # Créer le répertoire s'il n'existe pas
     if not os.path.exists(checkpoint_dir):
         os.makedirs(checkpoint_dir)

     features = articles_to_text(train_data, "article_text", 0, 80000)

     # Créer un callback pour sauvegarder les poids du modèle
     cp_callback = tf.keras.callbacks.ModelCheckpoint(
         filepath=checkpoint_path,
         save_weights_only=True,
         verbose=1
     )
```

```python
# Vérifier si les poids du modèle existent déjà
if os.path.exists(checkpoint_path):
    print("Loading weights from checkpoint...")
    gpt2_lm.load_weights(checkpoint_path)
else:
    print("Training model from scratch...")
    # Entraîner le modèle avec GPU et utiliser le callback de checkpoint
    with tf.device('/device:GPU:0'):
        num_epochs = 1

        # Taux d'apprentissage décroissant linéairement
        learning_rate = keras.optimizers.schedules.PolynomialDecay(
            initial_learning_rate=5e-5,
            decay_steps=len(features) * num_epochs,
            end_learning_rate=0.01,
        )

        loss = keras.losses.SparseCategoricalCrossentropy(from_logits=True)
        gpt2_lm.compile(
            optimizer=keras.optimizers.Adam(learning_rate=learning_rate),
            loss=loss,
            weighted_metrics=["accuracy"],
        )

        # Entraîner le modèle avec le callback de checkpoint
        gpt2_lm.fit(
            x=features,
            epochs=num_epochs,
            callbacks=[cp_callback]  # Passer le callback de checkpoint à l'entraînement
        )
```

- test rapide d'évaluation de modèle
    - dataset test + val
- test de génération simple

TESTS EVALUATION PRECISION SIMPLES SUR MODELE

```python
[ ]  # Charger les données de validation
     val_data, val_labels = load_ds.load_dataset(pathToDataset, "val", 7)

     # Rassembler le texte des articles en une seule liste
     val_features = articles_to_text(val_data, "article_text", 0, 6632)

     # Évaluer le modèle sur les données de test
     with tf.device('/device:GPU:0'):
       val_loss, val_accuracy = gpt2_lm.evaluate(
           x=np.array(val_features),
           verbose=1
       )

     # Afficher les résultats de l'évaluation
     print(f"Validation Loss: {val_loss}")
     print(f"Validation Accuracy: {val_accuracy}")
```

```python
[ ]  # Charger les données de test
     test_data, test_labels = load_ds.load_dataset(pathToDataset, "test", 7)

     # Rassembler le texte des articles en une seule liste
     test_features = articles_to_text(test_data, "article_text", 0, 6657)

     # Évaluer le modèle sur les données de test
     with tf.device('/device:GPU:0'):
       test_loss, test_accuracy = gpt2_lm.evaluate(
           x=np.array(test_features),
           verbose=1
       )

     # Afficher les résultats de l'évaluation
     print(f"Test Loss: {test_loss}")
     print(f"Test Accuracy: {test_accuracy}")
```

```
2/2 [==============================] - 23s 5s/step - loss: 2.9220 - accuracy: 0.4465
Test Loss: 2.922043561935425
Test Accuracy: 0.44648462533950806
```

TEST GENERATION

```
[ ]  # Exemple de prompts basés sur des sujets de papiers scientifiques
     prompts = [
         "The impact of global warming on marine biodiversity",
         "Technological advancements in renewable energy sources",
         "Genetic factors influencing Alzheimer's disease",
         "The role of artificial intelligence in personalized medicine",
         "Quantum computing and its future implications",
         "Mechanisms of resistance to antibiotics in bacteria",
         ""
     ]

     # Parcourir et générer des réponses pour chaque prompt
     for prompt in prompts:
         start = time.time()
         output = gpt2_lm.generate(prompt, max_length=200)
         end = time.time()

         print(f"\nPrompt: {prompt}")
         print("GPT-2 output:")
         print(output)
         print(f"TOTAL TIME ELAPSED: {end - start:.2f}s")
         print('_____________________________________________________')
```

```
Prompt: The role of artificial intelligence in personalized medicine
GPT-2 output:
The role of artificial intelligence in personalized medicine has been well recognized , with an increase in the number and variety of artificial intelligence technologies ( aiis ) and a number of clinical trials . however , there are
TOTAL TIME ELAPSED: 0.60s
_____________________________________________________

Prompt: Quantum computing and its future implications
GPT-2 output:
Quantum computing and its future implications for the physical sciences have been highlighted by the development of quantum computing . the quantum field has been explored in many areas , such as the design and development of quantum
TOTAL TIME ELAPSED: 1.94s
_____________________________________________________

Prompt: Mechanisms of resistance to antibiotics in bacteria
GPT-2 output:
Mechanisms of resistance to antibiotics in bacteria include the presence of anaerobic microaerophilic bacteria ( mb ) in the gut tract , the presence of mb - specific molecules ( mb - tss ) in the intestinal tract , and the presence
TOTAL TIME ELAPSED: 0.57s
```

c. Sauvegarde des paramètres du modèle :

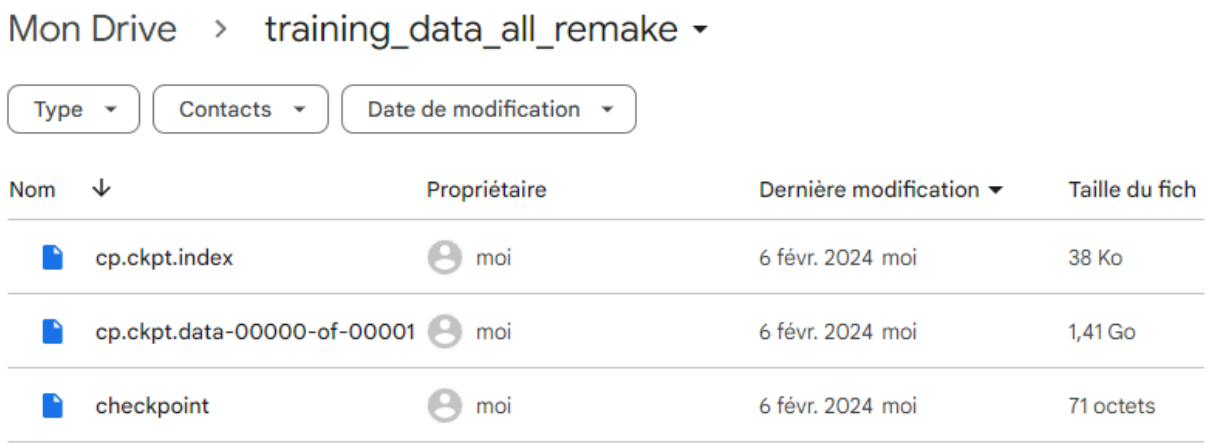
- utilisation de checkpoints
  - sauvegarder poids du modèle entraîné
  - fichier .ckpt = persister modèle entrainé
- à chaque fois,
  - init modèle vide,
  - charger les poids
- = modèle spécialisé

SAUVEGARDE POIDS CHECKPOINT Trouver comment sauvegarder le modèle

```
# Créer un callback pour sauvegarder les poids du modèle
cp_callback = tf.keras.callbacks.ModelCheckpoint(
    filepath=checkpoint_path,
    save_weights_only=True,
    verbose=1
)
```

```
# Entraîner le modèle avec le callback de checkpoint
gpt2_lm.fit(
    x=features,
    epochs=num_epochs,
    callbacks=[cp_callback]  # Passer le callback de checkpoint à l'entraînement
)
```

CHECKPOINT SAUVEGARDE EN FICHIERS Trouver comment sauvegarder le modèle



LOAD POIDS CHECKPOINT Trouver comment sauvegarder le modèle

```
[ ]  checkpoint_path = "/content/drive/MyDrive/training_data_all_3/cp.ckpt"
     # gpt2_lm.load_weights(checkpoint_path)
```

Optimisation et Problèmes rencontrées : ALL

a. Optimisation du modèle et fin-tunning :

- fine tuning différents modèles
  - base, medium, large (large a pas marché)
  - epochs 1 et epochs 3 (3 fois même section)
    - division train plusieurs parties dataset
    - entraînement effectué n epochs fois sur chaque section
  - très longs : plusieurs jours & plusieurs sessions google
  - après plusieurs tentatives,
    - rendu compte que optimisation fine tuning limité par performances RAM / GPU sur Collab
    - donc nécessité d'optimiser la dataset en elle même
  - cela a été seulement possible après avoir optimisé la dataset

INIT MODELE MEDIUM REMAKE fine tuning différents modèles

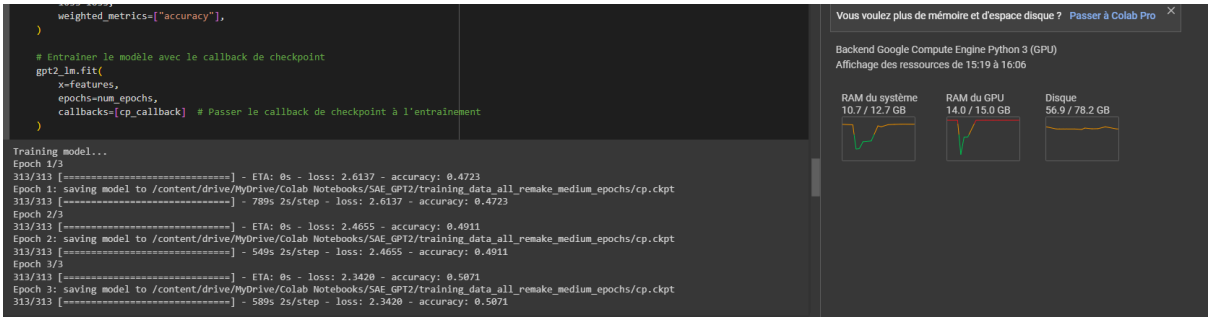
```
[ ]  os.environ["KERAS_BACKEND"] = "tensorflow"  # or "tensorflow" or "torch"

     keras.mixed_precision.set_global_policy("mixed_float16")

     preprocessor = keras_nlp.models.GPT2CausalLMPreprocessor.from_preset(
         "gpt2_medium_en",
         sequence_length=128,
     )
     gpt2_lm = keras_nlp.models.GPT2CausalLM.from_preset(
         "gpt2_medium_en", preprocessor=preprocessor
     )
```

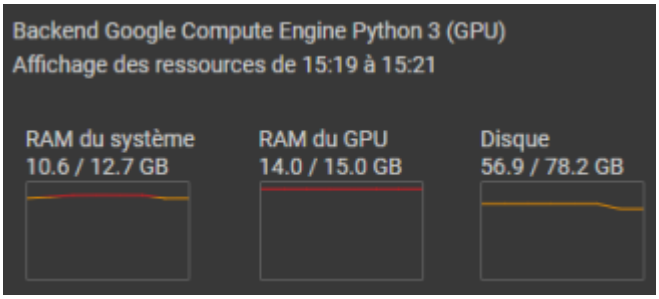
FINE TUNING REMAKE MEDIUM EPOCHS 3 : parler division train en 10k/10k fine tuning différents modèles

```python
# Définir le chemin et le répertoire du checkpoint
checkpoint_dir = os.path.dirname(checkpoint_path)

# Créer le répertoire s'il n'existe pas
if not os.path.exists(checkpoint_dir):
    os.makedirs(checkpoint_dir)

#features = dataset["train"][:10000]["article"] #fait
# features = dataset["train"][10000:20000]["article"] #fait
# features = dataset["train"][20000:30000]["article"] #fait

# features = dataset["train"][30000:40000]["article"] # fait 15/02 11h
# features = dataset["train"][40000:50000]["article"] # fait 15/02 12h
# features = dataset["train"][50000:60000]["article"] # fait 15/02 14h

# features = dataset["train"][60000:70000]["article"] # fait 15/02 16h22
# features = dataset["train"][70000:80000]["article"] # commencé 25/02 14h40 fini 15h20
# features = dataset["train"][80000:90000]["article"] # commencé 25/02 15h25 fini 16h

# features = dataset["train"][90000:100000]["article"] # commencé 25/02 16h05 fini 16h42
# features = dataset["train"][100000:110000]["article"] # commencé 01/03 22h40 fini 23h18
features = dataset["train"][110000:]["article"] # commencé 01/03 23h20 fini 23h50
```

```python
# Créer un callback pour sauvegarder les poids du modèle
cp_callback = tf.keras.callbacks.ModelCheckpoint(
    filepath=checkpoint_path,
    save_weights_only=True,
    verbose=1
)

print("Training model...")
# Entraîner le modèle avec GPU et utiliser le callback de checkpoint
with tf.device('/device:GPU:0'):
    num_epochs = 3

    # Taux d'apprentissage décroissant linéairement
    learning_rate = keras.optimizers.schedules.PolynomialDecay(
        initial_learning_rate=5e-5,
        decay_steps=len(features) * num_epochs,
        end_learning_rate=0.0,
    )

    loss = keras.losses.SparseCategoricalCrossentropy(from_logits=True)
    gpt2_lm.compile(
        optimizer=keras.optimizers.Adam(learning_rate=learning_rate),
        loss=loss,
        weighted_metrics=["accuracy"],
    )

    # Entraîner le modèle avec le callback de checkpoint
    gpt2_lm.fit(
        x=features,
        epochs=num_epochs,
        callbacks=[cp_callback]  # Passer le callback de checkpoint à l'entraînement
    )
```

PERFORMANCES REMAKE MEDIUM EPOCHS 3 fine tuning différents modèles





b. Optimisation du dataset et son utilisation :

- import direct impossible (taille importante, limite de ram colab)
- limite performance PC + serveur (ouverture texte + chargement)
- solution téléch. manuel → division / split en fichiers “chunks” paquet
- fonctions Python division / chargement / validation json

LOAD DATASET Problème utilisation dataset

```
[ ]  # chemin vers le jeu de données divisée localement
     pathToDataset = "drive/MyDrive/chunking-dataset"

     # "train" afin de prendre seulement la partie du jeu de données dédié à l'entrainement du modèle
     # 100 afin de prendre 100 fichiers de 1.000 articles chacun
     train_data, train_labels = load_ds.load_dataset(pathToDataset, "train", 100)
```

```
[ ]  # Fonction pour charger les données par parties
     def load_data_in_parts(dataset, text_key, part_size=10000):
         part = []
         for index, article in enumerate(dataset):
             article_text = article.get(text_key, "")
             part.append(article_text)
             if len(part) >= part_size or index == len(dataset) - 1:
                 print(f"Yielding part with {len(part)} articles...\n")  # Print the size of the current part
                 yield part
                 part = []
```

FONCTION PACKAGE PYTHON LOAD DATASET Problème utilisation dataset

```
[ ]  def load_files_from_directory(directory, num_files=None):
         all_data = []
         all_labels = []
         files_processed = 0

         for filename in os.listdir(directory):
             if num_files is not None and files_processed >= num_files:
                 break

             file_path = os.path.join(directory, filename)
             if os.path.isfile(file_path):
                 with open(file_path, 'r') as file:
                     articles = json.load(file)
                     for article in articles:
                         all_data.append(article)
                         for label in article.keys():
                             if label not in all_labels:
                                 all_labels.append(label)

                 files_processed += 1

         return all_data, all_labels

     def load_dataset(base_directory, dataset_type, num_files=None):
         dataset_directory = os.path.join(base_directory, dataset_type)
         return load_files_from_directory(dataset_directory, num_files)
```

FONCTION CONVERSION DICO EN LISTE DE STRINGS D'ARTICLE
(load dataset charge du json en dico python, fct ici passe en liste de string, chaque string = article) Problème utilisation dataset

```
[ ]  def articles_to_text(articles, text_key, n_start, n_end):
       article_list = []
       for article in articles[n_start:n_end] :
         article_text = " ".join(article.get(text_key, ""))
         article_list.append(article_text)
       return article_list
```

- partie optimisation fine tuning - problèmes de performances RAM
  - découverte dataset Hugging Face
    - identique TF mais (-) conso mémoire
    - import + load dataset → stockage, pas de conso RAM
    - modifier toutes les fonctions qu'on avait
  - fonction fit utilisant que la RAM alors que GPU dispo
    - chargement dataset + finetuning à l'origine sans GPU
    - puis trouvé withgpu T4

import dataset HF découverte dataset Hugging Face

```
[ ]  !pip install keras_nlp -q
     !pip install datasets -q
```

```
[ ]  import os
     import gc
     import tensorflow as tf
     from tensorflow import keras
     from datasets import load_dataset
     import json
     import keras_nlp
     import time
     import numpy as np
```

```
[ ]  dataset = load_dataset("scientific_papers", 'pubmed')


     train_dataset = dataset["train"]
     test_dataset = dataset["test"]
     val_dataset = dataset["validation"]
```

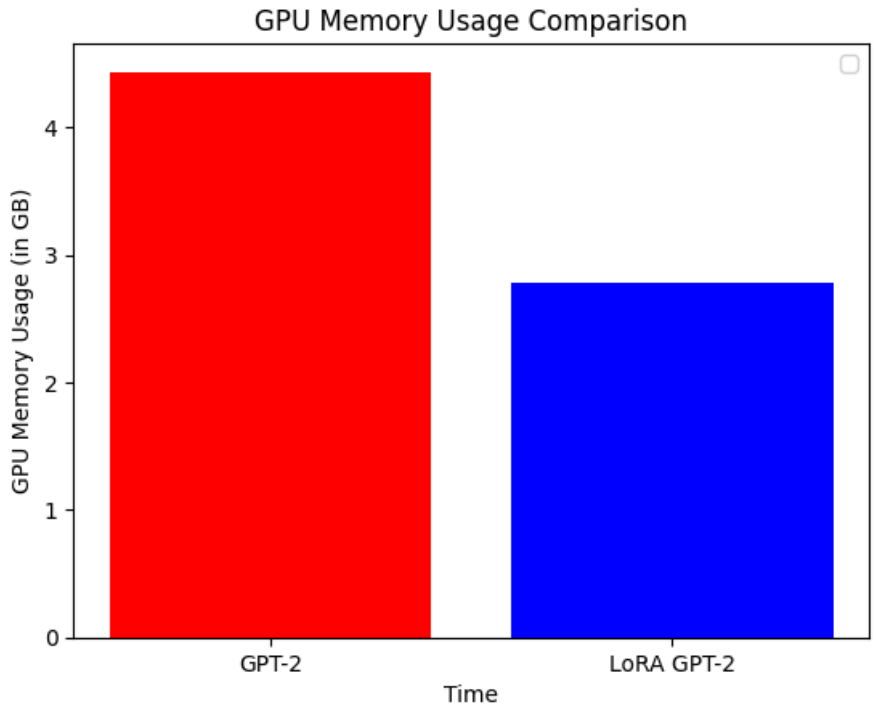
- <u>tentative technique lora</u>
  - compliquée dans 1 premier temps car modification couche GPT2
    - = change ce que couche accepte (entrée)
  - mais cause problème génération de texte, n'arrive pas à apprendre
    - texte troué, ponctuation aléatoire
    - pourtant apprentissage possible
  - possible raison texte trop long, sait pas comment interpréter

COUCHE LAYER LORA

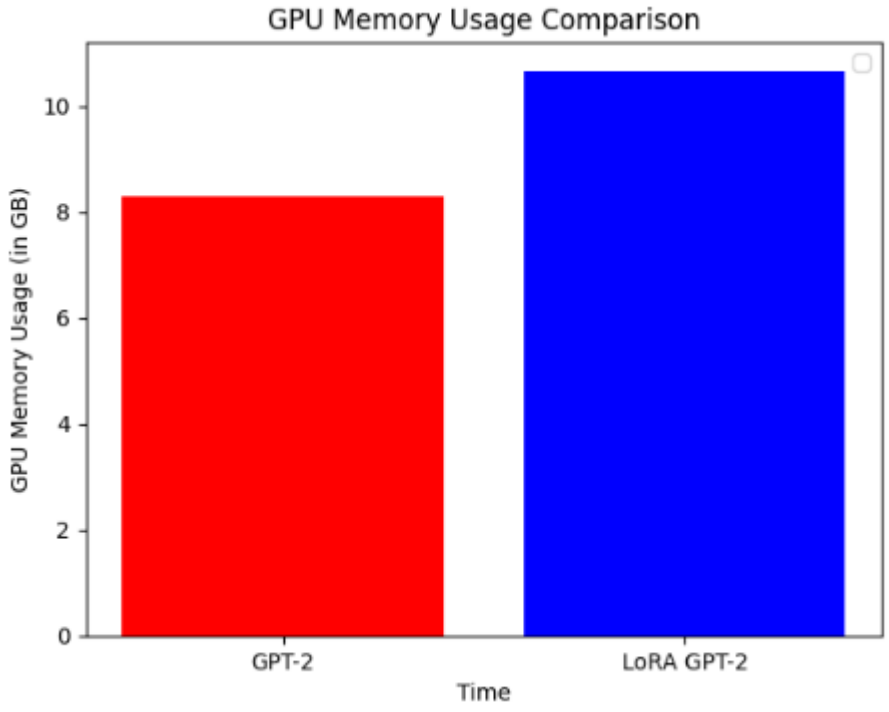
```python
class LoraLayer(keras.layers.Layer):
    def __init__(
        self,
        original_layer,
        rank=8,
        alpha=32,
        trainable=False,
        **kwargs,
    ):
        original_layer_config = original_layer.get_config()
        name = original_layer_config["name"]

        kwargs.pop("name", None)

        super().__init__(name=name, trainable=trainable, **kwargs)

        self.rank = rank
        self.alpha = alpha

        self._scale = alpha / rank

        self._num_heads = original_layer_config["output_shape"][-2]
        self._hidden_dim = self._num_heads * original_layer_config["output_shape"][-1]

        self.original_layer = original_layer
        self.original_layer.trainable = False

        self.A = keras.layers.Dense(
            units=rank,
            use_bias=False,
            kernel_initializer=keras.initializers.VarianceScaling(
                scale=math.sqrt(5), mode="fan_in", distribution="uniform"
            ),
            trainable=trainable,
            name=f"lora_A",
        )

        self.B = keras.layers.EinsumDense(
            equation=original_layer_config["equation"],
            output_shape=original_layer_config["output_shape"],
            kernel_initializer="zeros",
            trainable=trainable,
            name=f"lora_B",
        )

    def call(self, inputs):
        original_output = self.original_layer(inputs)
        if self.trainable:
            lora_output = self.B(self.A(inputs)) * self._scale
            return original_output + lora_output

        return original_output
```

```
[ ]  plt.bar(
         ["GPT-2", "LoRA GPT-2"],
         [max(gpt2_lm_memory_usage), max(lora_model_memory_usage)],
         color=["red", "blue"],
     )

     plt.xlabel("Time")
     plt.ylabel("GPU Memory Usage (in GB)")

     plt.title("GPU Memory Usage Comparison")
     plt.legend()
     plt.show()
```

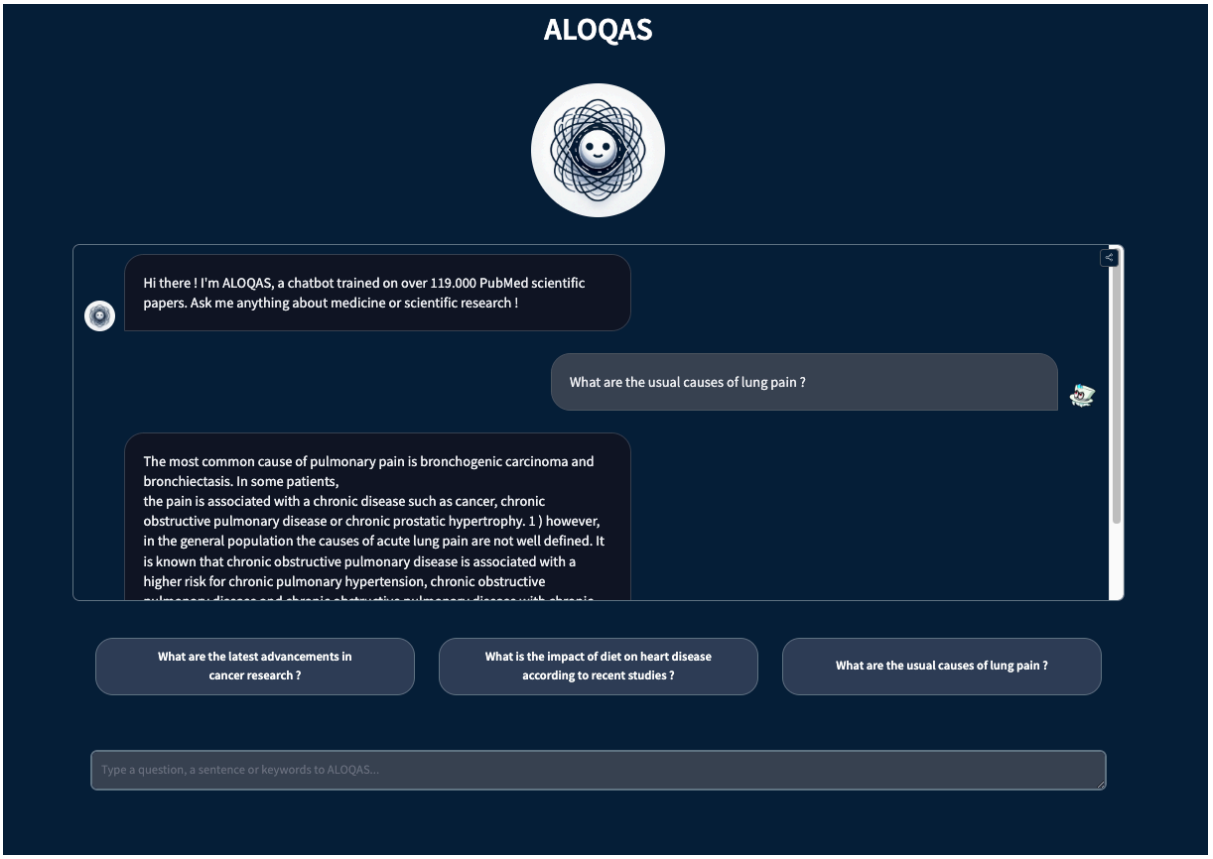
Performance Théorique :



Performance Pratique :



Démonstration : Kasagué + Ony



Prompt 1 : What are the latest advancements in cancer research ?
Prompt 2 : What is the impact of diet on heart disease according to recent studies ?
Prompt 3 : What are the usual causes of lung pain ?
Prompt 4 : What are the mechanisms of resistance to antibiotics in bacteria ?
Prompt 5 : Could you explain to me what is the Inherited defect of coagulation factors ?
Prompt 6 : Tell me more about cardiac hypertrophy.

(la ponctuation est importante)

Recommandation pour améliorer le chat bot :

- Migrer de GPT Medium -> Large
- Diminuer le temps de réponse du model
- Ajouter un historique des conversations
- Ajout d'animation et d'interaction
-

Conclusion : Quentin

En somme, ALOQAS a réussi à développer un chatbot spécialisé dans le domaine médical en utilisant des technologies avancées comme Keras NLP et TensorFlow, avec l'implémentation du modèle GPT-2. L'équipe interdisciplinaire a surmonté divers défis

techniques, notamment en optimisation et gestion des ressources, grâce à une collaboration efficace via des plateformes comme Google Drive, Discord, et GitHub. Le chatbot, grâce à son interface Gradio, a démontré son potentiel comme outil d'assistance en recherche scientifique, offrant des réponses pertinentes sur des sujets complexes. Pour améliorer encore le chatbot, il est recommandé de passer à un modèle plus performant, de réduire le temps de réponse, et d'ajouter un historique des conversations, rendant ALOQAS un outil plus précieux pour la recherche et l'éducation scientifiques.

Annexes notions techniques (pour nous!!)

GPT2

- Tokenisation
  - strings → tokens = séquence d'entiers pour chaque mot
- Embedding
  - token → vecteur
  - table embedding
- Transformateur / transformer
  - blocs de transformateurs
    - Multihead
      - focus attention certaines partie du texte entrée lors préd
      - par ex mots + pertinents dans une phrase
    - feedforward
      - réseau de neurones
- Décodage
  - vecteurs → scores de probabilité / chaque mot dans vocab modèle
  - mot avec score + élevé = choisi comme prédiction

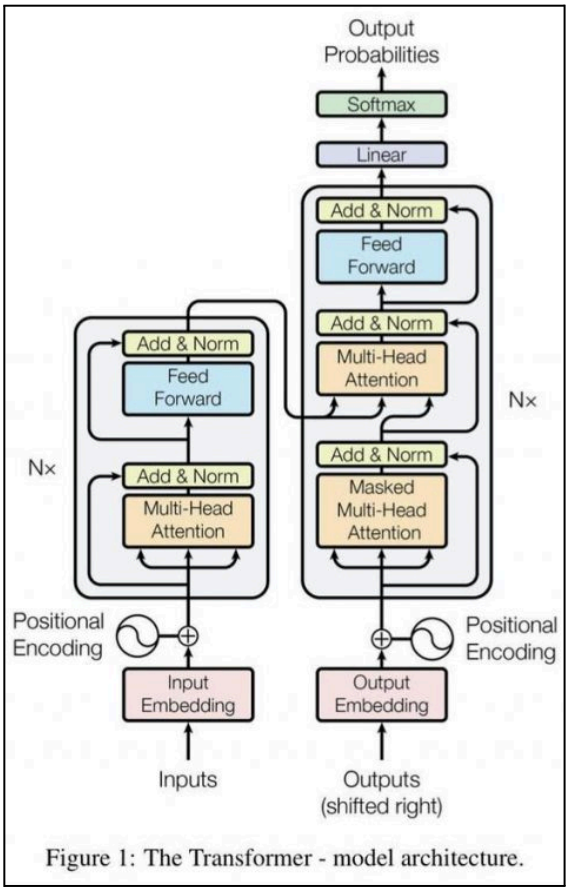


Figure 1: The Transformer - model architecture.

Tokenisation : on convertit string en tokens, c'est des séquences d'entiers pour représenter
Embedding : on convertit token en vecteur = utilisation table embedding
Transformateur / transformer : ça passe dans des blocs de transformateurs

-

Décodage