

ASSISTANT DE DISCUSSION CONTEXTUELLE AVEC IA

SIMPLIFIEZ LA RECHERCHE
D'INFORMATIONS AVEC L'IA
GÉNÉRATIVE



KEROUDINE BELLADJO

The screenshot shows a window titled "Assistant" with a dark gray header bar featuring red, yellow, and green window control buttons. Below the title, there is a menu bar with French labels: Edition, Zoom, Ajouter une page, Insérer, Tableau, Graphique, Texte, Figure, and Données manuelles. The main area of the window is a large white space. At the bottom, there is a horizontal row of four buttons:

- A rectangular button with a green border containing the text "Poser votre question ici".
- A button labeled "Afficher les résultats" in blue text.
- A button labeled "Mettre un contexte" in black text.
- A button labeled "Réglage" in black text.

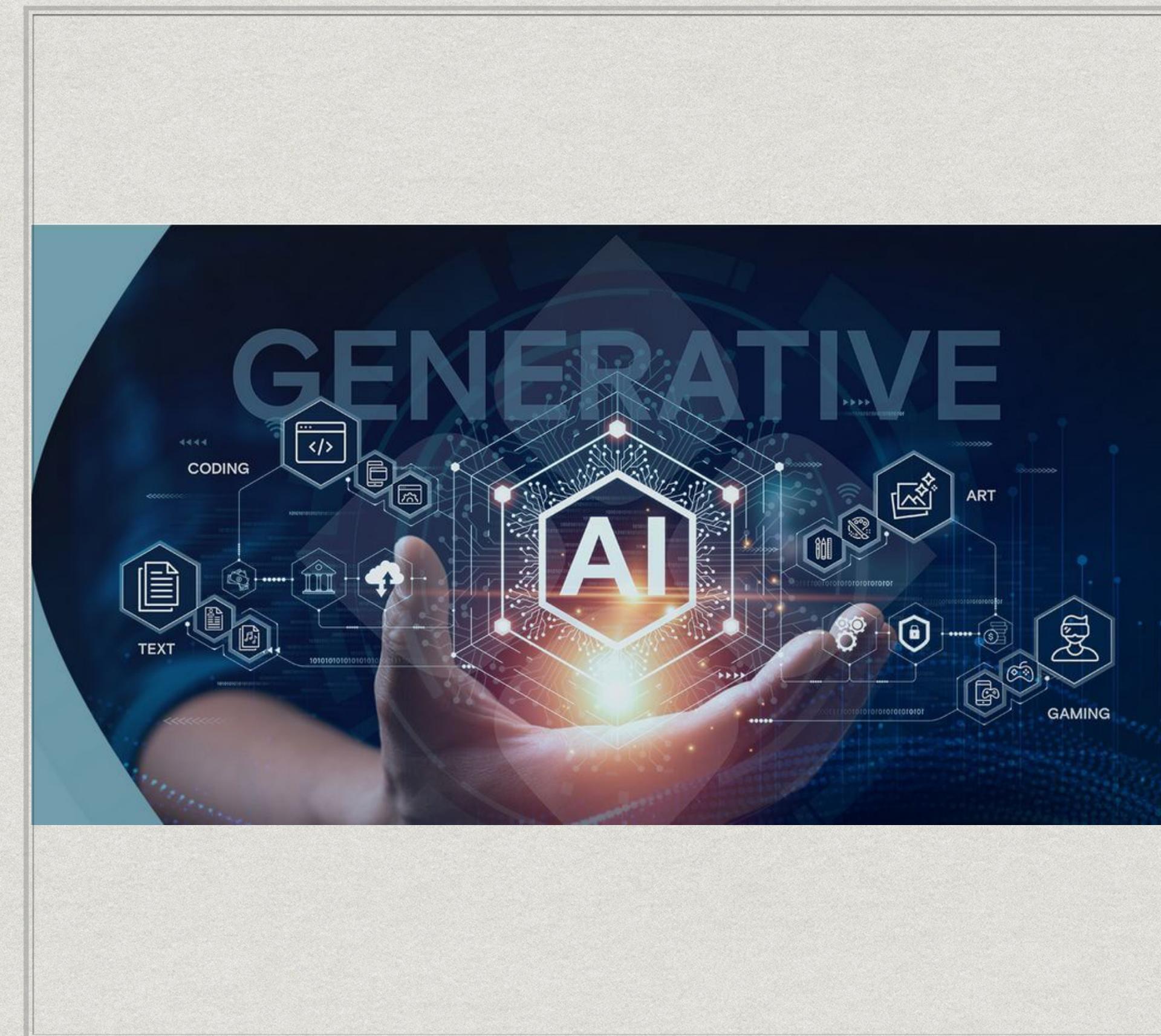
Introduction

- * Dans un monde où l'accès à l'information est devenu omniprésent grâce à la prolifération des médias numériques, trouver des réponses pertinentes est de plus en plus difficile. Face à cette surcharge informationnelle, notre application offre une solution novatrice en combinant l'intelligence artificielle à une interface utilisateur conviviale. Notre objectif est de simplifier la recherche d'informations dans ce dédale de données, répondant ainsi aux besoins variés des utilisateurs confrontés à des défis complexes. Notre application exploite les dernières avancées en matière d'intelligence artificielle pour extraire efficacement les informations pertinentes des documents numériques, améliorant ainsi l'expérience utilisateur de manière significative.



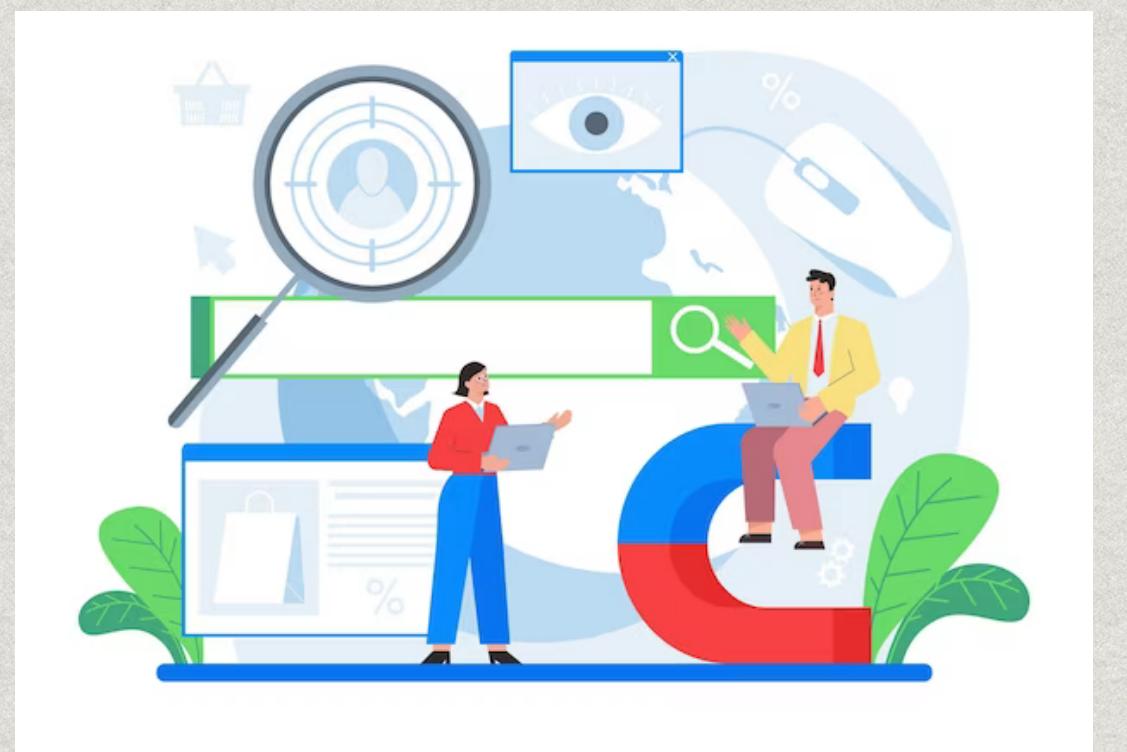
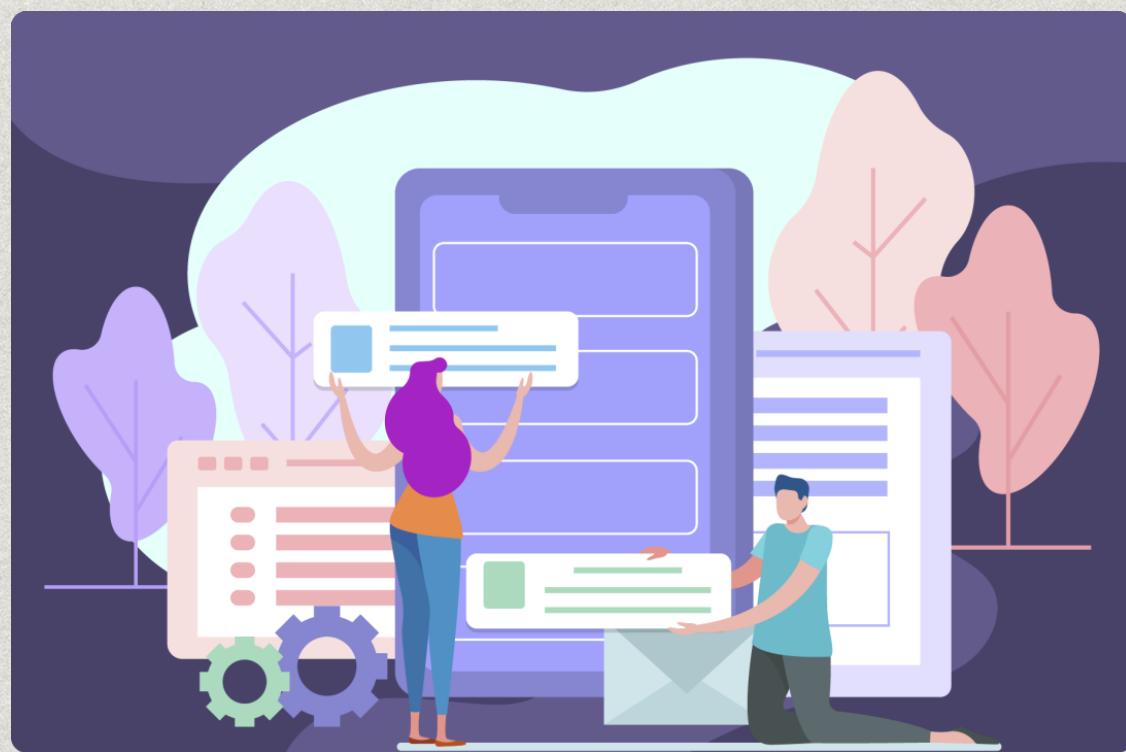
Contexte

- * Dans notre société moderne, l'accès rapide à des données pertinentes est essentiel pour la prise de décision et la résolution de problèmes. Cependant, la surabondance d'informations pose un défi majeur : extraire les données pertinentes et les comprendre dans un contexte spécifique. Traditionnellement, cette tâche fastidieuse impliquait des heures de lecture et de tri manuel des documents volumineux, comme les PDF. Notre application intervient en proposant une approche novatrice qui exploite l'intelligence artificielle et le traitement du langage naturel pour simplifier ce processus. En permettant aux utilisateurs d'interagir de manière contextuelle avec leurs documents, notre application vise à libérer les utilisateurs des contraintes de recherche manuelle, leur offrant ainsi plus de temps pour leurs activités essentielles.



Fonctionnalités Clés

- * **Extraction de texte de PDF :** Notre application simplifie le processus d'extraction de texte à partir de fichiers PDF. Les utilisateurs peuvent charger facilement un fichier PDF dans l'application, et celle-ci extrait automatiquement le texte pertinent du document. Cela élimine le besoin fastidieux de copier et coller le contenu manuellement, permettant aux utilisateurs d'économiser du temps et des efforts.
- * **Communication contextuelle :** L'intégration de l'intelligence artificielle dans notre application permet une interaction contextuelle avancée. Les utilisateurs peuvent poser des questions dans le contexte spécifique de leur document PDF, et notre IA comprend le contexte pour fournir des réponses précises et pertinentes. Cette fonctionnalité offre une expérience de discussion fluide et naturelle, où les utilisateurs peuvent obtenir des informations sans avoir à quitter l'application.
- * **Interface utilisateur conviviale :** Nous avons accordé une attention particulière à la conception de notre interface utilisateur pour la rendre aussi conviviale que possible. Les fonctionnalités sont clairement étiquetées et organisées de manière logique, rendant l'interaction avec l'application intuitive pour les utilisateurs de tous niveaux. Grâce à une interface bien pensée, les utilisateurs peuvent naviguer facilement dans l'application et accéder rapidement aux fonctionnalités dont ils ont besoin.



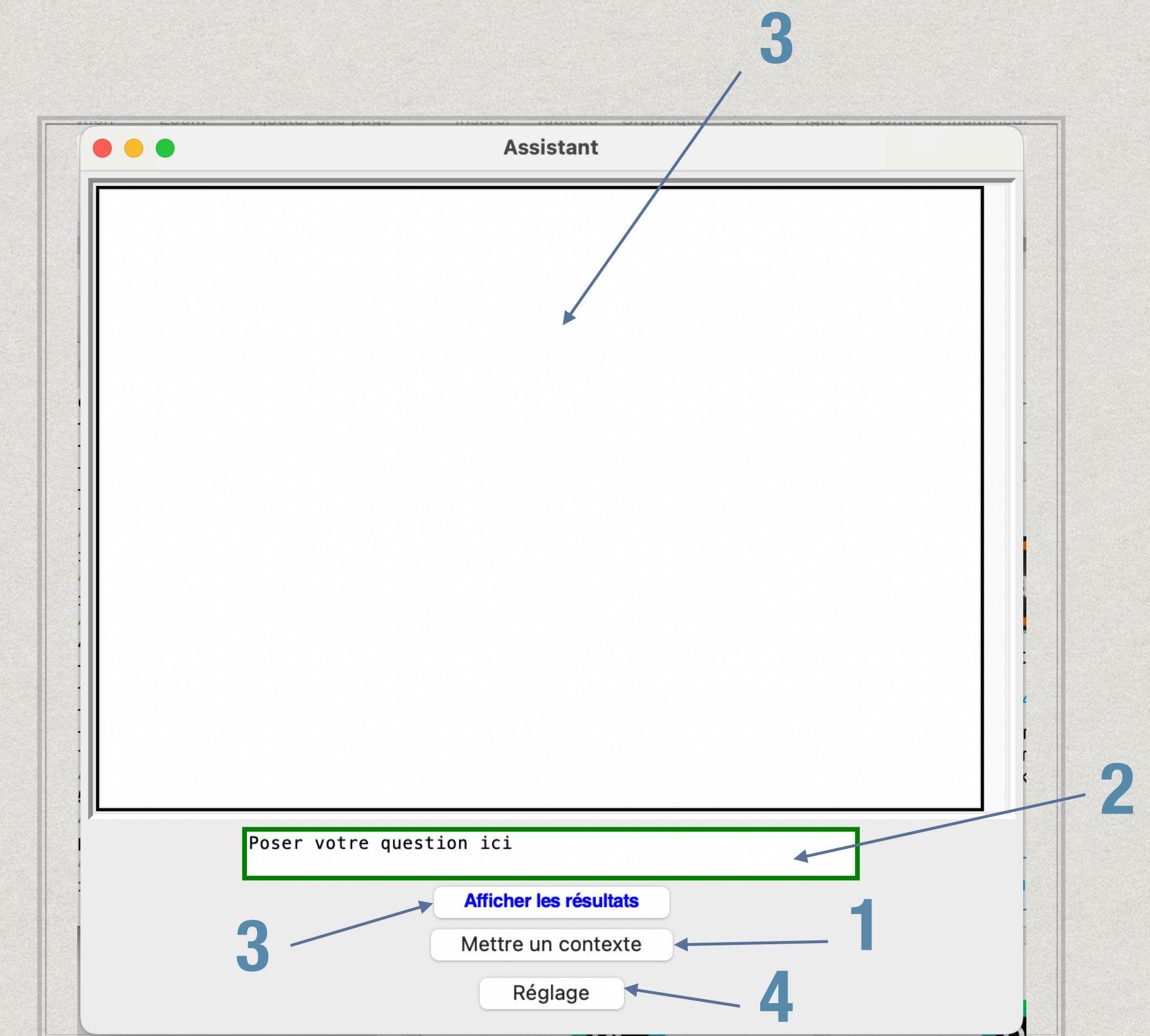
Technologies Utilisées

- * **OpenAI API** : Pour fournir une communication contextuelle avancée, notre application utilise l'API OpenAI. Cette API permet à notre application d'interagir avec une intelligence artificielle performante, capable de comprendre le contexte des documents PDF et de fournir des réponses pertinentes aux questions des utilisateurs.
- * **Python** : Le langage de programmation Python a été choisi pour le développement de l'application en raison de sa polyvalence et de sa robustesse. Python assure une intégration harmonieuse avec l'API OpenAI, offrant ainsi une flexibilité pour l'implémentation des fonctionnalités et des algorithmes avancés.
- * **Tkinter** : Pour créer une interface utilisateur interactive, nous avons utilisé Tkinter, une bibliothèque graphique standard de Python. Tkinter offre une manière simple et efficace de concevoir des interfaces utilisateur conviviales, ce qui nous a permis de fournir une expérience utilisateur fluide et intuitive.
- * **Fitz** : Pour l'extraction de texte à partir des fichiers PDF, notre application utilise la bibliothèque Fitz. Cette bibliothèque permet un traitement efficace des documents PDF, assurant ainsi une extraction précise du texte et une analyse approfondie du contenu.



Utilisation de l'Application

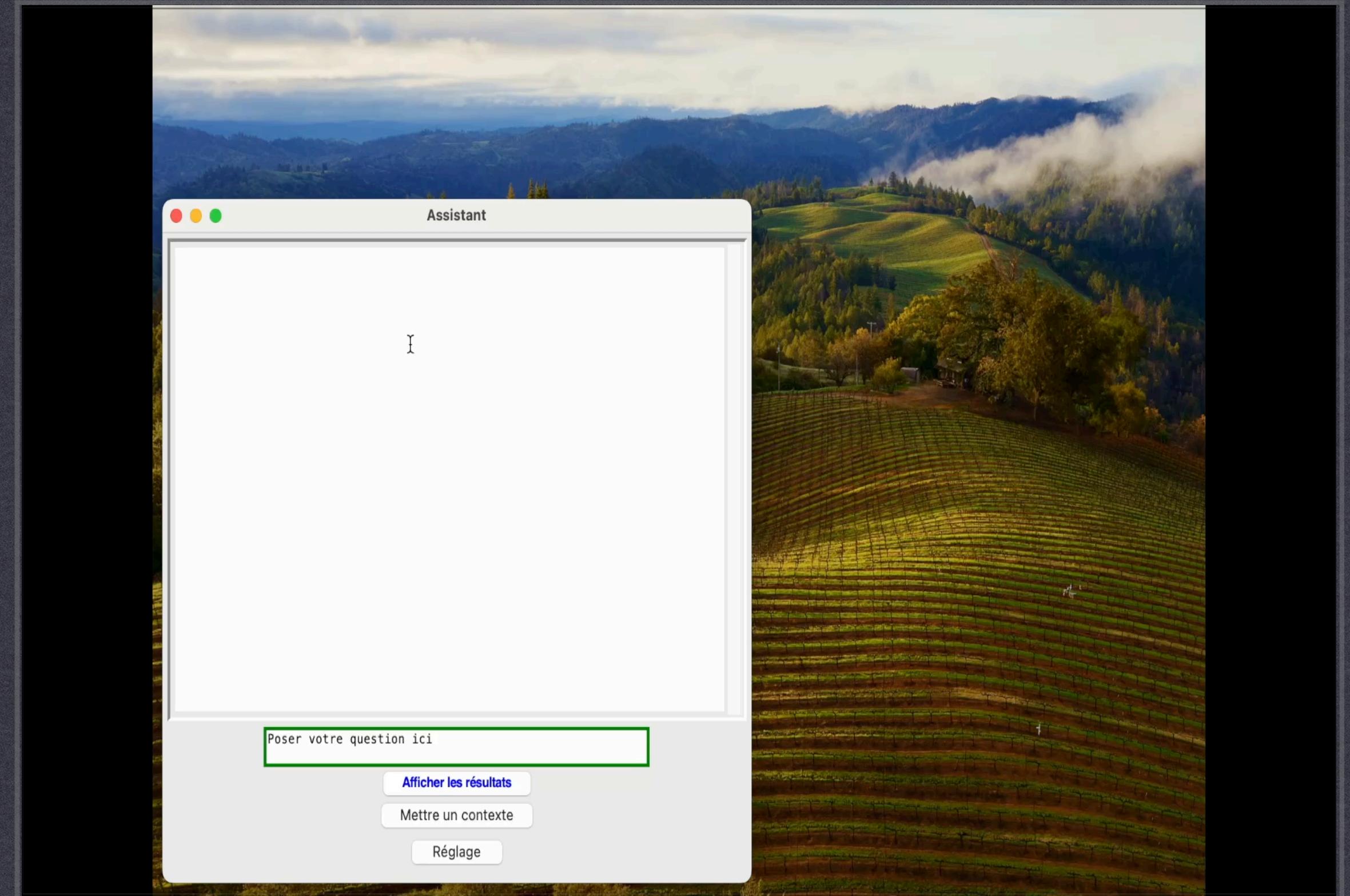
- Chargement du fichier PDF :** Les utilisateurs peuvent charger facilement leur fichier PDF en un seul clic à l'aide du bouton "Mettre un contexte". Cette étape simple permet aux utilisateurs de commencer rapidement à utiliser l'application sans tracas.
- Communication avec l'IA :** Une fois le document chargé, les utilisateurs peuvent poser leurs questions dans le champ de saisie dédié. L'application utilise alors l'API OpenAI pour comprendre le contexte de la question et fournir des réponses pertinentes en fonction du contenu du document PDF.
- Affichage des résultats :** Les réponses de l'IA et les détails de la requête sont affichés dans le journal de discussion, offrant ainsi une vue claire de l'interaction entre l'utilisateur et l'IA. Cette fonctionnalité permet aux utilisateurs de suivre facilement leurs interactions précédentes et d'accéder rapidement aux informations dont ils ont besoin.
- Personnalisation de l'interface :** Les utilisateurs ont la possibilité de personnaliser l'apparence de l'interface selon leurs préférences. Ils peuvent modifier le thème, la couleur de fond et d'autres paramètres pour créer une expérience utilisateur sur mesure. Cette fonctionnalité garantit que chaque utilisateur peut adapter l'application à ses besoins spécifiques et à ses préférences esthétiques.



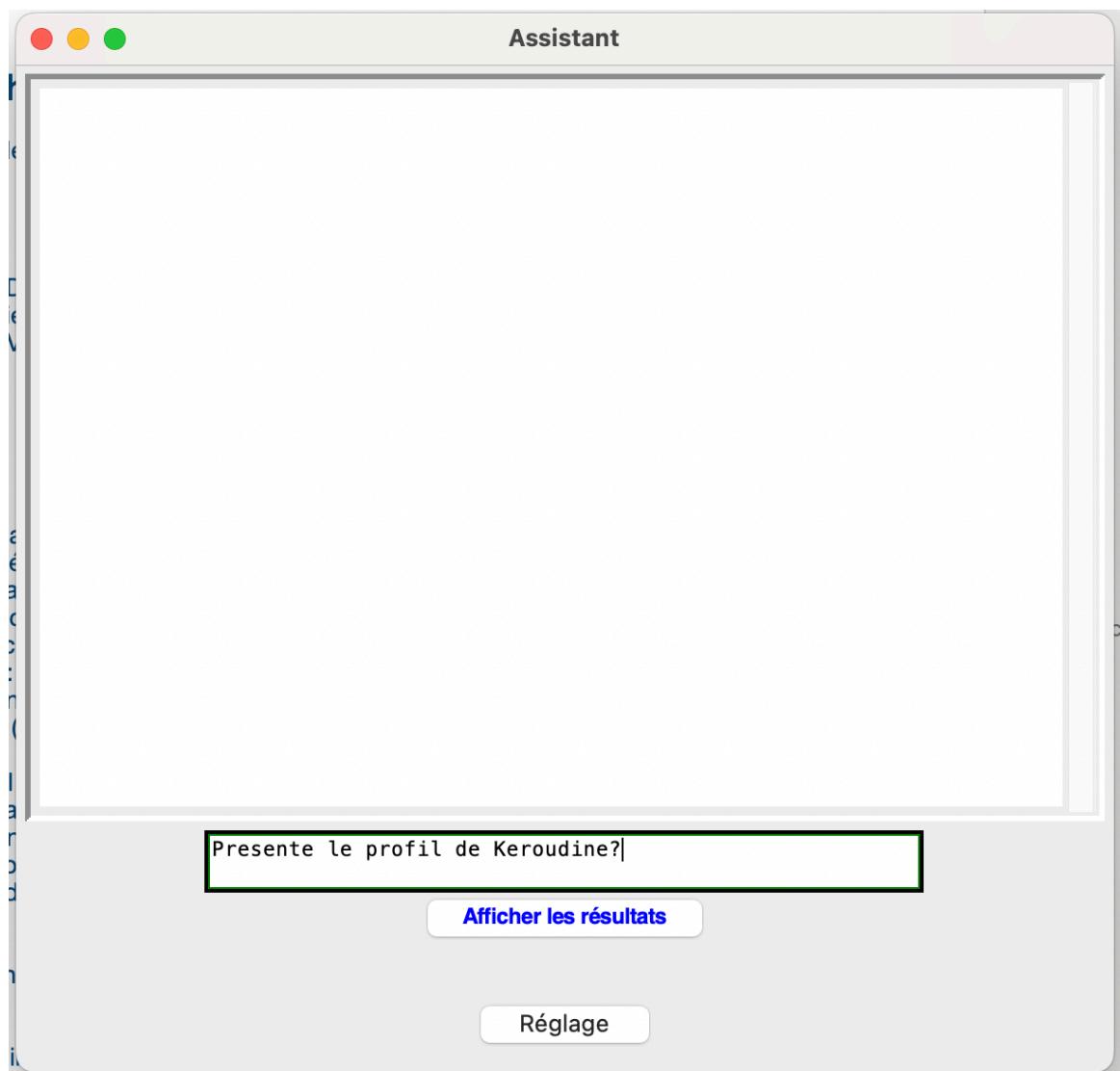
CONCLUSION

NOTRE APPLICATION PRÉSENTE UNE AVANCÉE SIGNIFICATIVE DANS L'ACCÈS À L'INFORMATION EN OFFRANT UNE SOLUTION PRATIQUE ET EFFICACE POUR OBTENIR DES RÉPONSES PERTINENTES À PARTIR DE DOCUMENTS PDF. AVEC NOTRE ASSISTANT DE DISCUSSION CONTEXTUELLE AVEC IA, LA RECHERCHE D'INFORMATIONS DEVIENT PLUS SIMPLE, RAPIDE ET PRÉCISE. LES FONCTIONNALITÉS TELLES QUE L'EXTRACTION AUTOMATIQUE DU TEXTE PERTINENT, LA COMMUNICATION FLUIDE AVEC UNE INTELLIGENCE ARTIFICIELLE PERFORMANTE ET UNE INTERFACE UTILISATEUR CONVIVIALE GARANTISSENT UNE EXPÉRIENCE OPTIMALE. QUE CE SOIT POUR DES BESOINS ACADÉMIQUES, PROFESSIONNELS OU PERSONNELS, NOTRE APPLICATION RÉPOND AUX BESOINS VARIÉS DES UTILISATEURS LES PLUS EXIGEANTS. EN SIMPLIFIANT LA RECHERCHE D'INFORMATIONS ET EN FOURNISSANT DES RÉPONSES PRÉCISES, ELLE ENRICHIT L'EXPÉRIENCE UTILISATEUR ET ENCOURAGE L'APPRENTISSAGE, LA DÉCOUVERTE ET L'INNOVATION. NOUS SOMMES CONVAINCUS QUE NOTRE APPLICATION CONTINUERA À ÉVOLUER POUR RÉPONDRE AUX BESOINS CHANGEANTS DES UTILISATEURS DANS UN MONDE EN CONSTANTE ÉVOLUTION.

DEMO



INDEX



Assistant

Utilisateur: Presente le profil de Keroudine?

Assistant: Keroudine Belladjo est un étudiant en master 2 MIASHS (Mathématiques et Informatique Appliquées aux Sciences Humaines et Sociales) parcours MQME (Méthodes Quantitatives et Modélisation pour l'Entreprise) à l'Université de Lille . Il est à la recherche d'un stage de fin d'études à partir de février 2024 dans le domaine de la data, plus précisément en tant que Data Analyst ou Data Scientist.

Keroudine possède des compétences en extraction de texte, création d'ensembles de données, analyse exploratoire des données, représentation des données et fragmentation des textes. Il maîtrise également des outils tels que Python, Jupyter Notebook, SQL et Rstudio.

Il a réalisé un projet de détection de langue à l'aide de l'algorithme k-plus proche voisin (K-NN) et a effectué un stage en tant que Data Scientist chez Decisys Consulting. Ses principales compétences incluent le data preprocessing, l'analyse de données, le machine learning et la modélisation statistique.

Keroudine est décrit comme curieux, rigoureux, adaptable et capable de travailler en équipe. Il est également intéressé par les domaines du data engineering, de la data visualization, du cloud computing et de l'économétrie.

Vous pouvez consulter son profil LinkedIn à l'adresse suivante : <https://www.linkedin.com/in/keroudine-belladjo-9a5b28184>

total de tokens utilisés: 2205
coût de la requête : 0.00441 \$ USD

Afficher les résultats

Réglage

assistant

February 8, 2024

1 Le script

```
[1]: import tkinter as tk
from tkinter import scrolledtext
from tkinter import ttk
import random
from tkinter import colorchooser
from threading import Thread

import openai
from dotenv import load_dotenv
import time
import requests
import json
import fitz

[2]: def chat_bot(messages):
    url = "https://api.openai.com/v1/chat/completions"

    # Requête à l'API OpenAI avec l'historique des messages
    payload = json.dumps({
        "model": "gpt-3.5-turbo",
        "messages": messages
    })

    headers = {
        'Content-Type': 'application/json',
        'Authorization': 'Bearer VOTRE_CLE_OPENAI',
    }

    response = requests.post(url, headers=headers, data=payload)
    return response.json()

def extract_text_from_pdf(pdf_path):
    doc = fitz.open(pdf_path)
    text = ""
```

```

for page_num in range(doc.page_count):
    page = doc[page_num]
    text += page.get_text()
doc.close()
return text

initial_message = [
    {"role": "system", "content": "Vous êtes un assistant virtuel qui aide avec\u2192des informations sur le cv de KEROUDINE BELLADJO"}, 
    {"role": "user", "content": "Aider KEROUDINE BELLADJO à trouver un stage"}, 
    {"role": "assistant", "content": "Bien sûr, je peux vous aider à chercher\u2192votre stage?"}
]

def send_message():

    pdf_path = "Keroudine_cv.pdf"
    document_text = extract_text_from_pdf(pdf_path)
    initial_message.append({"role": "assistant", "content": document_text})

    message_history = initial_message.copy()

    user_input = user_input_text.get('1.0', tk.END).strip()

    if user_input != "":
        start_time = time.time()
        # Ajout du message de l'utilisateur à l'historique
        message_history.append({"role": "user", "content": user_input})

        # Récupération de la réponse du bot
        response = chat_bot(message_history)
        bot_reply = response['choices'][0]['message']['content']

        total_tokens = response['usage']['total_tokens']
        #cout = 0.00000011 * total_tokens
        cout = (total_tokens/5000)*0.01

        #n
        chat_log.tag_config("user_input", foreground="blue")
        chat_log.insert(tk.END, "Utilisateur: " + user_input + "\n", "user_input")
        chat_log.configure(font=("Times New Roman", 12, "bold"))

```

```

#n

chat_log.insert(tk.END, "\n")

#n
chat_log.tag_config("bot_reply", foreground="black")
chat_log.insert(tk.END, "Assistant: " + bot_reply + "\n", "bot_reply")
chat_log.configure(font=("Times New Roman", 12))
#n

chat_log.insert(tk.END, "\n")
chat_log.insert(tk.END, "\n")
chat_log.insert(tk.END, "-----" + "\n")

#n
chat_log.tag_config("total_tokens", foreground="green")
chat_log.insert(tk.END, "total de tokens utilisés: " + str(total_tokens) + "\n", "total_tokens")
chat_log.configure(font=("Courier", 12))
#n

#n
chat_log.tag_config("cout", foreground="red")
chat_log.insert(tk.END, "coût de la requête : " + str(cout) + " $ USD" + "\n", "cout")
chat_log.configure(font=("Courier", 12))
#n

# Ajout de la réponse du bot à l'historique
message_history.append({"role": "assistant", "content": bot_reply})

# Efface le champ de saisie
user_input_text.delete('1.0', tk.END)

end_time = time.time()
response_time = end_time - start_time

# le temps de réponse
progress = "Réponse en {:.2f} secondes".format(response_time)

#n
chat_log.tag_config("progress", foreground="green")

```

```

    chat_log.insert(tk.END, progress + "\n", "progress")
    chat_log.configure(font=("Courier", 12))
    #n
    #chat_log.insert(tk.END, progress + "\n")
    chat_log.insert(tk.END, "\n")
    chat_log.insert(tk.END, "\n")

def thred_send_message():
    tache1 = Thread(target=send_message)
    tache1.start()

# Fonction pour changer le thème de manière aléatoire
def change_theme():
    themes = ["alt", "default", "classic"]
    new_theme = random.choice(themes)

    ttk.Style().theme_use(new_theme)

    # Changer la couleur de fond et d'arrière-plan des widgets spécifiques
    window.configure(bg=background_color.get()) # Change la couleur de fond de la fenêtre principale
    ↪
    chat_log.configure(bg=background_color.get(), fg=text_color.get()) # ↪
    ↪Change la couleur de fond et de premier plan du journal de discussion
    user_input_text.configure(bg=text_entry_bg_color.get(), fg=text_color.
    ↪get()) # Change la couleur de fond et de premier plan du champ de saisie

# Fonction pour choisir la couleur de fond de l'interface
def choose_background_color():
    color = colorchooser.askcolor()[1]
    background_color.set(color)
    change_theme()

# Fonction pour choisir la couleur et la forme des zones de saisie de texte ou de cadre
def choose_text_entry_style():
    color = colorchooser.askcolor()[1]
    text_entry_bg_color.set(color)
    change_theme()

# Options pour le menu déroulant de réglage
settings_options = ["Changer de theme", "Choisir la couleur de fond", "Choisir la couleur du cadre de saisie"]

```

```

# Fonction pour gérer les sélections du menu déroulant
def handle_settings_selection(selected_option):
    if selected_option == "Changer de theme":
        change_theme()
    elif selected_option == "Choisir la couleur de fond":
        choose_background_color()
    elif selected_option == "Choisir la couleur du cadre de saisie":
        choose_text_entry_style()

# Fonction pour afficher ou cacher le menu déroulant
def toggle_settings_dropdown():
    if settings_dropdown.winfo_ismapped():
        settings_dropdown.place_forget()
    else:
        settings_dropdown.place(relx=0.5, rely=0.92, anchor=tk.CENTER)

# Création de la fenêtre principale
window = tk.Tk()
window.title("Assistant")
window.geometry("600x550")

# Variables pour les couleurs
background_color = tk.StringVar(value="white")
text_color = tk.StringVar(value="black")
text_entry_bg_color = tk.StringVar(value="white")

# Création du journal de discussion
chat_frame = ttk.Frame(window, style="ResultFrame.TFrame")
chat_frame.pack(padx=5, pady=5, fill=tk.BOTH, expand=False)

# Configuration du style du cadre du journal de discussion
ttk.Style().configure("ResultFrame.TFrame", relief="sunken", borderwidth=3)

chat_log = scrolledtext.ScrolledText(chat_frame, width=55, height=30)
chat_log.pack(padx=5, pady=5, fill=tk.BOTH, expand=False)

# Création du champ de saisie de l'utilisateur
def clear_comment(event):
    if user_input_text.get("1.0", "end-1c") == "Poser votre question ici":
        user_input_text.delete("1.0", "end-1c")

user_input_text = tk.Text(window, width=55, height=2)

```

```

user_input_text.insert("1.0", "Poser votre question ici")
user_input_text.pack()
user_input_text.bind("<FocusIn>", clear_comment)

user_input_text.configure(highlightbackground="green")

# Cration des boutons d'envoi
send_button = ttk.Button(window, text="Afficher les rsultats",  

    ↪command=thred_send_message, style="Custom.TButton")
send_button.pack()

# Configuration du style du bouton
ttk.Style().configure("Custom.TButton", foreground="blue", font=("Helvetica",  

    ↪12, "bold"))

# Cration du bouton "Rglage"
settings_button = ttk.Button(window, text="Rglage",  

    ↪command=toggle_settings_dropdown)
settings_button.place(relx=0.5, rely=0.95, anchor=tk.CENTER)

# Variable pour le menu droulant
settings_menu = tk.StringVar(window)
settings_menu.set(settings_options[0]) # Definir la premire option comme  

    ↪celle slectionnee par deault

# Cration du menu droulant
settings_dropdown = tk.OptionMenu(window, settings_menu, *settings_options,  

    ↪command=handle_settings_selection)

# Initialisation de l'historique de discussion
message_history = initial_message.copy()

window.mainloop()

```

[]: