Veille informationnelle automatisée via IA

March 13, 2024

Veille informationnelle automatisée via IA

Ce notebook est un essi sur l'utilisation de Mistral.AI dans le cadre d'une veille informationnelle sur internet. Le but est de récupérer des articles sur internet et d'en faire un résumé en utilisant `mistral.ai`, puis de les envoyer par e-mail en format HTML généré par `Mistral.ai`

le code se découpe en 3 parties:

- récuperation des articles via url avec `BeautifulSoup`
- génération de résumé avec `Mistral.AI`
- génération du contenu HTML pour l'envoyer par e-mail automatisé via `Mistral.AI`
- envoi du message par e-mail

nous d'étaillerons le code pas à pas dans le reste du notebook.

0.1 Chargement des Packages :

[ ]:
```
import requests
from bs4 import BeautifulSoup
from mistralai.client import MistralClient
from mistralai.models.chat_completion import ChatMessage
import smtplib
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
```

0.2 Extraction des articles

Une fois les packages chargés, nous allons écrire le code permettant de récupérer les différent articles que l'on veut résumer. chaque article est alors extrait sur base de l'adresse URL disponible dans le l'objet BeautifulSoup et stocké dans un dataframe nommé `Messages` qui permettra de traiter individuellement les articles par l'IA :

[ ]:
```
response = requests.get("https://www.oppbtp.com")
soup = BeautifulSoup(response.text, 'html.parser')
articles = soup.find_all('a', class_ = "card card-column card-light␣
 ↪no-decoration")

Messages = []
```

```
for article in articles:
    text = article.get("href")
    test = requests.get(text)
    Messages.append(BeautifulSoup(test.text, 'html.parser').find('div',␣
 ↪{'class':"entry-content"}).text.replace('\n', ' ').strip())
```

0.3 Génération des résumés

On défini le role de notre IA pour lui demander de nous fournir le résumé de l'article que l'on va lui donner en entrée :

[ ]:
```
messages = [
    ChatMessage(role="system", content="Tu es un programme de génération de␣
 ↪résumé à partir de texte que l'on te donne, tu ne fais que fournir un résumé␣
 ↪du texte que l'on te donne en input")
]
```

une fois le role défini, on va, pour chaque Message, demander au programme de nous fournir le résumé de l'article :

[ ]:
```
model = "mistral-medium-latest"
api_key="dXhF7z625frqL76yHgWY5PZJ910298mg" # ne fonctionne plus après le 31␣
 ↪mars 2024

responses = []

for message in Messages:
    messages.append(ChatMessage(role="user", content=message))
    client = MistralClient(api_key=api_key)
    Mistral = client.chat(model=model, messages=messages)
    responses.append(Mistral.choices[0].message.content.replace('\n', ' '))
    messages.pop(-1)
```

Il va falloir regrouper toutes les reponses dans une seule chaîne de caractères pour pouvoir demander à l'IA de générer un mail sur base de tous les résumés qu'il a généré précedement :

[ ]:
```
test = ""

for i in range(len(responses)):
    test = test + "- titre de l'article : " + articles[i].find("h4").text+",␣
 ↪résumé de l'article : "+ responses[i] + "\n"
```

0.4 Génération du corps du mail

On défini ici un nouveau role au modèle, ici le but et de générer automatiquement un corp de mail, au format html, pour pouvoir générer la mise en forme, et envoyer par la suite au destinataire choisi :

[ ]:
```
Mailer = [
    ChatMessage(role="system", content="tu es un assistant de creation de mail,␣
 ↪tu ne reponds que par un corps de mail mis en page, étant un mail contenant␣
 ↪les informations dans le message que tu reçois, tu commenceras par un␣
 ↪message de politesse de 20 mots pour dire bonjour, de façon amicale, je veux␣
 ↪simplement le corps du texte, sans titre. tu réponds au format HTML.")
]
```

une fois le role défini, on va demander au programme de nous fournir le corps de mail :

[ ]:
```
model = "mistral-medium-latest"
api_key="dXhF7z625frqL76yHgWY5PZJ910298mg" # ne fonctionne plus après le 31␣
 ↪mars 2024

Mailer.append(ChatMessage(role="user", content=test))
client = MistralClient(api_key=api_key)
Mistral = client.chat(model=model, messages=Mailer)
responses = Mistral.choices[0].message.content
```

on défini la fonction d'envoi du mail :

[ ]:
```
def send_mail():
    msg = MIMEMultipart()
    msg['From'] = 'jbastruz@gmail.com'
    msg['To'] = 'jmastruz@gmail.com'
    msg['Subject'] = 'résumé d\'article Mistral.AI'
    body = responses

    msg.attach(MIMEText(body, 'html'))

    server = smtplib.SMTP('smtp.gmail.com', 587)
    server.starttls()
    server.login('jbastruz@gmail.com', 'pogw xkis yjdf buff')
    server.sendmail("jbastruz@gmail.com", 'jmastruz@gmail.com', msg.as_string())
    server.quit()
```

on execute la fonction d'envoi du mail :

[ ]:
```
send_mail()
```

Ce programme est un exemple d'utilisation de Mistral.AI, il est possible de l'utiliser pour d'autres projets. il est possible de récupérer du texte depuis une autre source. et ainsi travailler sur un autre type d'information. il faudra penser à travailler sur la manière d'extraire la donnée, qui peut être différente d'un site à l'autre.