Faculté
des **sciences économiques** et de **gestion**
Université de Strasbourg

Automatisation de la Recherche et de l'Envoi de Candidatures

Xinyue ZHANG; Jingjing JIANG; Huiyue LI

Plan de présentation

1. Introduction
2. Processus de réalisation
   - Etape 1: Collecte d'informations via le scraping
   - Etape 2 : Génération automatisée du contenu des e-mail de candidature
   - Etape 3 : Envoi automatique des emails de candidatures
3. Difficultés rencontrées
4. Critiques et conclusions

1. Introduction

- Buts : faciliter la recherche et la postulation d'emploi
- Objectifs : addresses mails, contenus du mails, envoyer le mail

```
file_path = "job_descriptions.xlsx"
df = pd.read_excel(file_path)
cv_path = "CV.pdf"

for index, row in df.iterrows():
    recipient = row['Email']
    description = row['Description']
    send_email(recipient, gpt(description), cv_path)
    print(f"Send OK {index}")
```

2.1 Collecte d'informations via le scraping

- **Extraction des liens d'offres d'emploi**

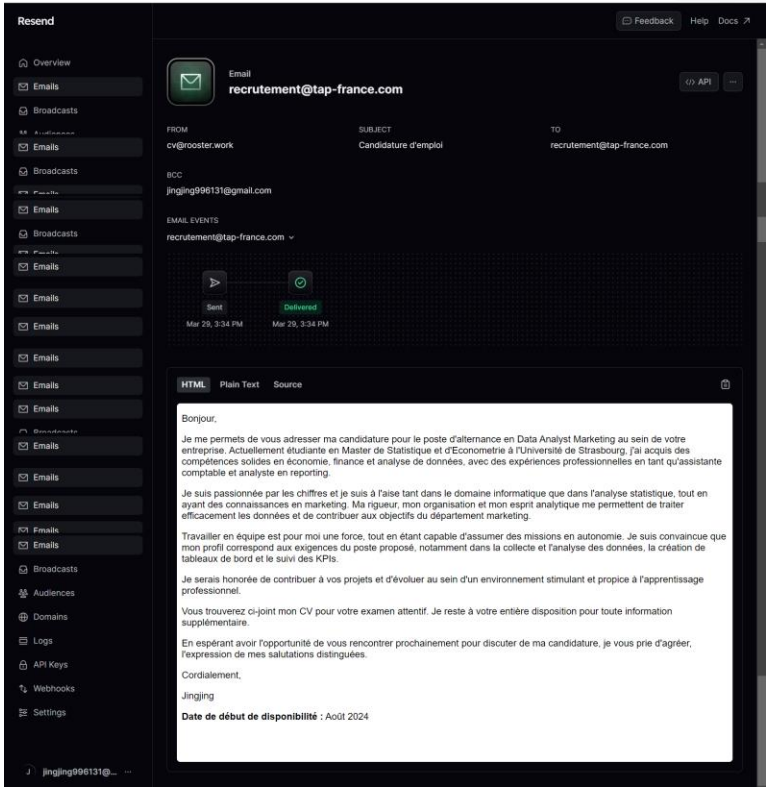
```
def job_links(domains, num_pages):
    links = []
    driver = webdriver.Chrome(service=cService)
    for domain in domains:
        for page in range(1, num_pages + 1):
            start = page * 10
            url = f"https://fr.indeed.com/emplois?q={domain}&l=%C3%8Ele-de-
France&sc=0kf%3Ajt(apprenticeship)%3B&start={start}"
            driver.get(url)
            time.sleep(4)
            for i in range(1, 18):
                xpath_expression =
'/html/body/main/div/div[2]/div/div[5]/div/div[1]/div[5]/div/ul/li[{}]/div/div[1]/div/div/div/ta
ble[1]/tbody/tr/td/div[1]/h2/a'.format(i)
                elements = driver.find_elements(By.XPATH, xpath_expression)
                if not elements:
                    continue
                link = elements[0].get_attribute('href')
                links.append(link)

    driver.quit()
    return links

links = job_links(['data','analyst'], 15)
```

- **Collecte les adresses mail et les descriptions de poste.**
- **Enregistrement des informations collectées dans un fichier Excel.**

2.2 Création automatisée de contenu de candidature avec l'AI

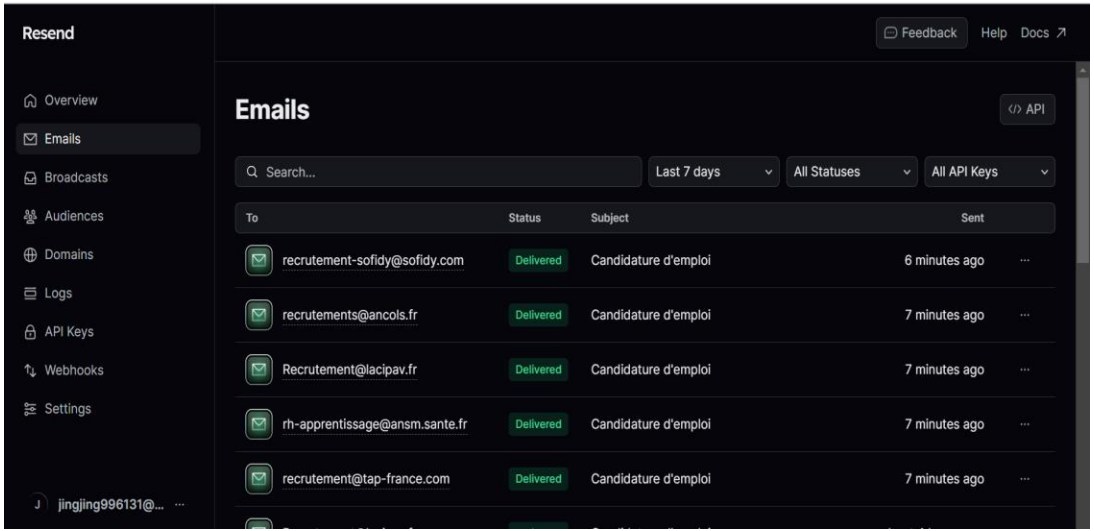
- Utilisation de ChatGPT pour personnaliser le contenu des e-mails de candidature.

```
from openai import OpenAI
client = OpenAI(API_key="XXXX")

def gpt(text):
    role_system = """Demander au Chatgtp les exigences"""
    completion = client.chat.completions.create(
        model="gpt-3.5-turbo-0125",
        messages=[
            {"role": "system", "content": role_system},
            {"role": "user", "content": text}
        ]
    )
    return completion.choices[0].message.content
```



2.3 Envoi automatique des e-mails de candidatures via Resend

```
import resend
resend.api_key = "XXXX"

def send_email(recipient, content, cv_path):
    f = open(cv_path, "rb").read()
    params = {
        "from": "cv@rooster.work",
        "to": recipient,
        "subject": "Candidature d'emploi",
        "html": content,
        "bcc": ["jingjing996131@gmail.com"],
        "attachments": [{"filename": "CV.pdf", "content": list(f)}],
    }
    resend.Emails.send(params)
```



3. Difficultés Rencontrées

- **Extraction des données à partir du site d'emploi**

  => Indeed qui utilise une structure dynamique.

- **Envoi automatique d'emails existence des problèmes de sécurité**

  => comme le problème de l'authentification.

4. Conclusion et critiques

Caractéristiques :

- Personnalisation : contenu du mail plus approprié
- Flexibilité : chercher des emplois dont les gens ont besoin
- Efficacité : postuler les emplois sans besoin visiter le site

Critiques :

- Fiabilité du scraping
- Qualité du contenu généré par ChatGPT
- Sécurité de l’envoi des emails