An NLP-based tool that generates structured data in tabular format from unstructured information contained in emails - Extracting & Pre-processing

Mini Project Report

by **Rakesh Roshan Gouda**

Regd.No-22016

Supervised by: **Prof. Pallav Kumar Baruah and Sri Satya Sai Mudigonda**

SRI SATHYA SAI INSTITUTE OF HIGHER LEARNING

(Deemed to be University)

Department of Mathematics and Computer Science

Prasanthi Nilayam Campus

April 2023

Table of Contents:

1.Introduction

Emails are a popular mode of communication for financial transactions, but extracting critical information from them can be time-consuming and challenging. Our project aims to develop an automated tool that uses Natural Language Processing (NLP) and the Gmail API to extract key details such as sender names, transaction amounts, payment methods, email dates, and email addresses from unstructured email messages. The tool will output the data in a structured format, saving time and effort in managing and organizing financial data from emails. This report will discuss the methodology used in the project, the challenges encountered during the development process, and the results achieved.

Problem Statement:

Our project aims to solve the challenge of managing and extracting financial data from emails by developing an automated tool that uses NLP and the Gmail API to extract critical details such as sender names, transaction amounts, payment methods, email dates, and email addresses from unstructured email messages. The tool outputs the data in a structured format, thereby saving time and effort in managing and organizing financial data from emails.

2.Motivation

The inspiration for this project came from the real-world need to efficiently manage financial data from emails. We realized that the current manual process of extracting and organizing data from emails is time consuming and prone to errors. By utilizing advanced data processing techniques, we aim to provide a solution to the problem of manually extracting and organizing financial information from large volumes of emails. The tool we have developed saves time and effort, as it automatically retrieves important data from emails and presents it in a tabular format.

3.Methodology

Our methodology involved using the Gmail API to retrieve email data, extracting the HTML content, and storing it for processing. We then used natural language processing (NLP) techniques, specifically regular expressions, to extract financial information such as the transaction amount, currency symbol, sender name, and payment method.

Next, we used the Pandas library to organize the extracted data into a tabular format, which was displayed to the user and also saved in a CSV file for future reference. Throughout the process, we employed various data cleaning and preprocessing techniques to ensure the accuracy and reliability of the output.

Overall, our approach allowed for efficient and accurate extraction of financial information from large volumes of emails, providing a valuable tool for managing and organizing financial data.

4. Work

Accessing Gmail data: In this step, we used the Gmail API to access the email data, including the subject, sender, and body of the email. Here is a code snippet that shows how we used the Gmail API to authenticate and get the email data.

HTML extraction: In this step, we extracted the HTML part of the email and stored it for further processing. Here is a code snippet that shows how we used the BeautifulSoup library to extract the HTML part.

NLP processing: In this step, we used NLP techniques to extract financial information such as transaction amount, currency symbol, sender name, and payment method using regular expressions. Here is a code snippet that shows how we used regular expressions to extract the transaction amount.

Tabular representation: In this step, we used the Pandas library to convert the extracted financial data into a tabular format, which makes it easy to read and analyze. Here is a code snippet that shows how we created a Pandas DataFrame to store the financial data.

Output representation: In this step, we showcased the financial data on the screen and saved it in a CSV file for future reference. Here is a code snippet that shows how we displayed the financial data on the screen and saved it in a CSV file.

Here is my code snippet:

```
from google.oauth2.credentials import Credentials
from googleapiclient.discovery import build
from googleapiclient.errors import HttpError
import base64
from email.mime.text import MIMEText
import os
import pandas as pd
from bs4 import BeautifulSoup
import re
import locale
locale.setlocale(locale.LC_ALL, '')
creds = Credentials.from_authorized_user_file('token.json', ['https://www.googleapis.com/auth/gmail.readonly'])

# Replace with the email address of the sender you want to retrieve messages from
#sender_email = 'satyajit1288@gmail.com'
#sender_email = 'dashbabu9@gmail.com'
amount_words = ['rupees','Rs','RS','USD', '$', 'AUD', 'EUR', 'GBP', 'INR', 'CAD', 'JPY','£', '€']
# Payment methods to search for
payment_methods = ['credit card', 'debit card', 'net banking', 'UPI','GPay']
currency_map = {
    'rupees': '₹',
    'Rs': '₹',
    'RS': '₹',
    'USD': '$',
    '$': '$',
    'AUD': '$',
    'EUR': '€',
    'GBP': '£',
    'INR': '₹',
    'CAD': '$',
    'JPY': '¥',
    '£': '£',
    '€': '€'
}


df = pd.DataFrame(columns=["sender name","Purpose","Amount", "Payment method"])

# Create a Gmail API client
service = build('gmail', 'v1', credentials=creds)
```

- `google.oauth2.credentials` is used for authenticating the Gmail API.

- `googleapiclient.discovery` is used for building the Gmail API client.
- `googleapiclient.errors` is used for handling errors that may occur while using the Gmail API.
- `base64` is used for decoding the base64-encoded message body.
- `email.mime.text` is used for creating a MIME message object to send emails.
- `os` is used for accessing the operating system functionalities.
- `pandas` is used for creating and manipulating dataframes.
- `bs4` is used for parsing HTML and XML documents.
- `re` is used for performing regular expressions.
- `locale` is used for formatting numbers and currency based on the user's locale.

These are lists and dictionaries used to match and map the currency words and payment methods in the email messages.

```
df = pd.DataFrame(columns=["sender name","Purpose","Amount", "Payment method"])
```

This line creates an empty pandas dataframe with four columns: `sender name`, `Purpose`, `Amount`, and `Payment method`.

`service = build('gmail', 'v1', credentials=creds)` This line creates a Gmail API client object, which is used to interact with the Gmail API.

```
try:
    # Search for messages from the specified sender
    #query = f'from:{sender_email}'
    query = "is:unread"
    result = service.users().messages().list(userId='me', q=query,maxResults=15).execute()

    # Print the message IDs and snippets
    if 'messages' in result:
        messages = result['messages']
        for message in messages:
            msg_id = message['id']
            msg = service.users().messages().get(userId='me', id=msg_id).execute()
            payload = msg['payload']
            html_part = None
            for part in payload['parts']:
                if part['mimeType'] == 'text/html':
                    html_part = part
                    soup = BeautifulSoup(base64.urlsafe_b64decode(html_part['body']['data']).decode('utf-8'), 'h
                    break

            #print('Subject:', payload['headers'])
            print('Subject:', payload['headers'][16]['value'])
            if html_part:
                print('Body:', base64.urlsafe_b64decode(html_part['body']['data']).decode('utf-8'))
                # Find payment amount

                for header in payload['headers']:
                    if header['name'] == 'Subject':
                        purpose = header['value']
                        print('Subject:', header['value'])
                    elif header['name'] == 'From':
                        sender_element = header['value']
                        sender_name = re.search(r'[A-Za-z\s]+', sender_element).group()

                        print("Sender name:", sender_name or "None")

                    # Find payment amount
                for word in amount_words:
                  instances = soup.find_all(text=lambda text: text and word in text)
                  #print("sdgh")
                  #print(instances)
                  for instance in instances:
```

This code searches for unread messages in the user's inbox using Gmail API. It then retrieves the message IDs and snippets, and for each message, it extracts the HTML body of the email using BeautifulSoup.It extracts sender name and below provided code searches for amount ,currency symbol and payment method.Finally it prints in tabular form in output.csv file using pandas library.

```
                currency_symbol = currency_map.get(word, word)
                match = re.search(r'(?:(\d{1,3}(?:,\d{3})*))?\s*' + re.escape(word) + r'\s*(?:(\d{1,3}(?:,\d{3})
                #match = re.search(r'(?:(\d+)\s*)?' + re.escape(word) + r'(?:(\s*)?(\d+))?', instance)
                if match:

                  amount = match.group(1) or match.group(2)
                  amount = locale.atof(amount.replace(',', '')) if amount else 0.0
                  #amount = float(amount.replace(',', ''))
                  print("Amount: ",currency_symbol,amount)
                  amount = str(currency_symbol)+' '+str(amount)

            # Find payment method
                  parent = instance.parent
                  while parent:
                        payment_method = re.search(r"(?i)\b(via|through|on|by)\b\s+(\w+[\w\s]*?)\W", parent.text
                        if payment_method:
                            payment_method_str = payment_method.group(2)
                            break
                        parent = parent.parent
                  print(f"Payment method: {payment_method_str}")

                  df = df.append({"sender name":sender_name or "None","Purpose":purpose,"Amount":amount, "Paymen
                  print(df)
                  print("\n")
                  df.to_csv('output.csv', index=False)
                else:
                    print('Body:', msg['snippet'])
                    print('\n')
    else:
        print('No messages found.')

except HttpError as error:
    print(f'An error occurred: {error}')
df
```

5.Results

By using the Gmail API and NLP techniques to extract financial data, our approach can be beneficial in automating tasks related to tracking financial transactions. This project can be useful for both individuals and organizations to keep track of their expenses and for auditing purposes. For instance, our approach can help in easily and automatically extracting financial data from emails, thus reducing the manual effort required to track expenses and improving accuracy.

Output of our work

```
Subject: Rahul Bandari <rahulbandari09@gmail.com>
Body: <div dir="ltr">Sairam Ram<div>I have sent you Rs 50,000 through
SBcollect.Please  check it out and confirm back to me.</div></div>

Sender name: Rahul Bandari
Subject: Donation
Amount: ₹ 50000.0
Payment method: SBcollect
 sender name Purpose Amount Payment method
0 Rahul Bandari Donation ₹ 50000.0 SBcollect
```

	sender name	Purpose	Amount	Payment method
0	Rahul Bandari	Donation	₹ 50000.0	SBcollect
1	Rahul Bandari	Refund	€ 10000.0	Zelle
2	Tejanshu Sekhar Panda	Donation	₹ 50000.0	SBcollect
3	Satyajit Behera	Donation	€ 2000.0	PayPal
4	Parth Nagar	Receipt	$ 15000.0	PhonePe
5	Parth Nagar	Refund	$ 10000.0	Gpay
6	Nirajit pandey	Donation	$ 10000.0	wire

6.Conclusion

This project has successfully demonstrated how the use of the Gmail API and NLP techniques can be leveraged to automate the process of extracting financial data from emails. The approach presented here has the potential to be used in various applications, such as tracking expenses for individuals or auditing purposes for organizations. The use of Pandas for storing and displaying the extracted data in a tabular format has made the process of analyzing financial transactions more efficient and user-friendly. Overall, this project highlights the value of combining advanced technologies like NLP and APIs to automate repetitive tasks and improve productivity.

Learning Outcome:

Through this project, I gained valuable experience in Python programming and relevant libraries such as Pandas, NumPy, and NLTK. I also gained familiarity with web scraping, text mining, predictive analysis, and data preprocessing techniques. Additionally, I learned how to use APIs, work with CSV file format, and apply NLP techniques using libraries such as beautifulsoup.

Possible future work:

Moving forward, our future work will focus on improving the tool by enabling it to extract data from any kind of email attachment, expanding its usability beyond Gmail. We also plan to incorporate machine learning techniques to train the model on larger datasets, making it more accurate and reliable.

7.References

Website: *https://developers.google.com/gmail/api/*

Course: Hands on Predictive Analysis using python