ElectoralBond QA from Tabular Data

Submission By:
Pinaki Das
T Karthikeyan

GitHub: KarpinChat
Drive: IITGN_ACM_HACKATHON

Problem Statement:

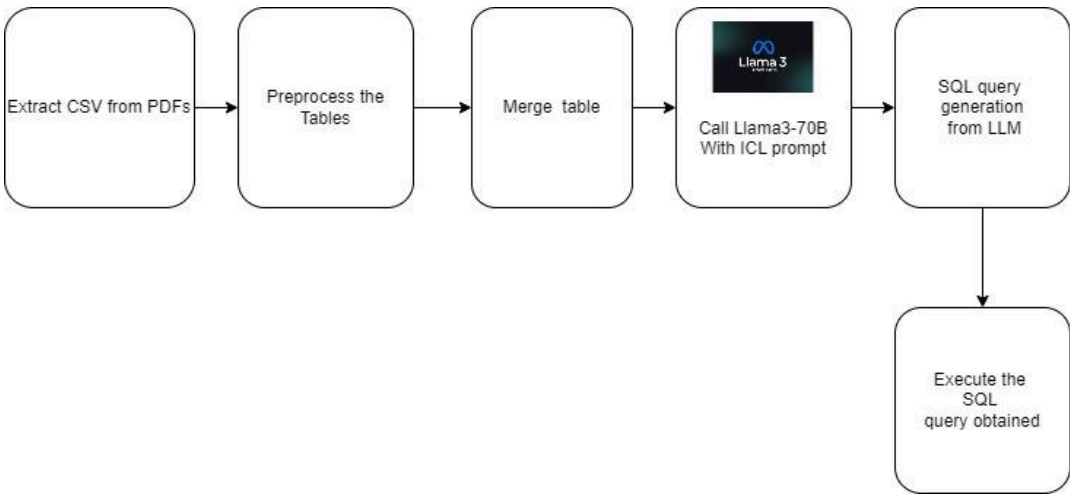
We are given two sets of PDFs containing.

i. Details of Electoral Bonds submitted by SBI on 21st March 2024 (EB_Redemption_Details) [Bonds encashed by political parties]
ii. Details of Electoral Bonds submitted by SBI on 21st March 2024 (EB_Purchase_Details) [Bonds purchased by Individuals and Companies]

These tables contain over 38K rows and 19 columns combined. Our task is to find effective ways to make a QA chat interface to query these PDFs and bring us the answers.

Primary Strategy:

Use LLMs to create effective SQL queries. Use those SQL queries to obtain the answers from LLM.



Pipeline Methodology

1. Converting PDF to CSV since it is fast to load and experiment

```python
def pdf_to_dataframe(pdf_path):
    # Open the PDF file
    with pdfplumber.open(pdf_path) as pdf:
        # Iterate over all the pages
        all_tables = []
        for page_number, page in enumerate(pdf.pages):
            print(page_number)
            # Extract table data from the page
            tables = page.extract_tables()

            # If tables are found, process them
            if tables:
                for table in tables:
                    # Convert table to DataFrame
                    df = pd.DataFrame(table[1:], columns=table[0])
                    print(f"Table found on page {page_number + 1}")
                    all_tables.append(df)

        # Concatenate all DataFrames into a single DataFrame
        combined_df = pd.concat(all_tables, ignore_index=True)
        print(len(combined_df))
        return combined_df
```

2. Preprocessing the table

```python
# Convert the csv path to a dataframe
def csv_to_df(csv_path):
    # Read the CSV file and convert it to a DataFrame
    df = pd.read_csv(csv_path)

    #Preprocess the columns names for the ease of asking sql queries
    df.rename(columns=lambda x: x.replace(' ', '_').replace('\n',
'_').replace('.','').replace('(','').replace(')',''), inplace=True)

    # Display the DataFrame
    print(df.columns)

    # Identify columns with "date" or "Date" in their names
    date_columns = [col for col in df.columns if 'date' in col.lower()]

    # Convert identified columns to date format
    for col in date_columns:
        df[col] = pd.to_datetime(df[col], format='%d/%b/%Y').dt.strftime('%Y-%m-%d')

    # Convert the 'Denominations' column from string with commas to integers
    df['Denominations'] = df['Denominations'].str.replace(',', '')  # Remove commas
    df['Denominations'] = df['Denominations'].astype(int)  # Convert to integer

    return df
```

3. Merging the two tables using the foreign key as Prefix+Bond Number. This was done because electoral bonds have a unique alphanumeric number, which is nothing but **a combination of Prefix and Bond Number.**

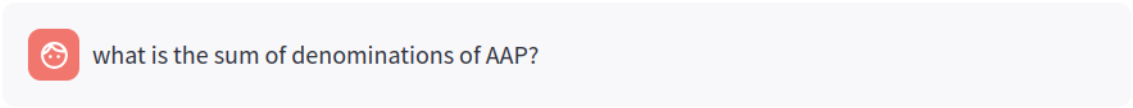
```
### CREATING MERGED DATAFRAME
csv_path1 = 'bonds_polparties.csv'
csv_path2 = 'bonds_individuals.csv'

df1 = csv_to_df(csv_path1)
df2 = csv_to_df(csv_path2)
merged_df = pd.merge( df1, df2, on=['Prefix', 'Bond_Number'], how='left') #can do other joins
merged_df.drop(columns = ["Denominations_y", "Sr_No_y" ], inplace=True)
merged_df.rename(columns={'Denominations_x': 'Denominations', 'Sr_No_x': 'Sr_No'},inplace=True)
merged_df = merged_df.fillna(0)
```

4. Using Grok's API, we call **meta-llama/Meta-Llama-3-70B.** We make use of ICL by giving a prompt which is placed below. It contains information about the table schemas, its foreign keys, and a one-liner explanation of the columns. **Very importantly**, we also add the various names of political parties to the prompt so that the LLM is aware of the different names of the same political party. For Eg: `YSR CONGRESS PARTY (YUVAJANA SRAMIKA RYTHU CONGRESS PARTY)`

```
Krompt = """
    You are a sqllite3 generator
    Generate SQLite3 queries for a table with the following schema:

    Table: Merged
    - Sr_No (INTEGER) : Unique serial number of the bond.
    - Date_of_Encashment (TEXT) : Date the bond was encashed.
    - Name_of_the_Political_Party (TEXT): Name of the political party that received the bond.
    - Account_no_of_Political_Party (TEXT): Bank account number of the political party
    - Prefix (TEXT): Prefix used for bond numbering.
    - Bond_Number (INTEGER): Unique number assigned to the bond.
    - Denominations (INTEGER): Value of the bond in monetary terms.
    - Pay_Branch_Code (INTEGER): Code of the branch where the bond was encashed.
    - Pay_Teller (TEXT): Name or ID of the teller who processed the encashment.
    - Reference_No_URN (TEXT): Unique reference number for the transaction.
    - Journal_Date (TEXT): Date the transaction was recorded in the journal.
    - Date_of_Purchase (TEXT): Date the bond was purchased.
    - Date_of_Expiry (TEXT): Expiry date of the bond
    - Name_of_the_Purchaser (TEXT): Name of the person who purchased the bond.
    - Issue_Branch_Code (TEXT): Code of the branch where the bond was issued.
    - Issue_Teller (TEXT): Name or ID of the teller who issued the bond.
    - Status (TEXT): Current status of the bond (e.g., encashed, expired).

    Total Unique Party are
        'ALL INDIA ANNA DRAVIDA MUNNETRA KAZHAGAM',
        'BHARAT RASHTRA SAMITHI', 'BHARATIYA JANATA PARTY',
        'PRESIDENT, ALL INDIA CONGRESS COMMITTEE', 'SHIVSENA',
        'TELUGU DESAM PARTY',
        'YSR CONGRESS PARTY (YUVAJANA SRAMIKA RYTHU CONGRESS PARTY)',
        'DRAVIDA MUNNETRA KAZHAGAM (DMK)', 'JANATA DAL ( SECULAR )',
        'NATIONALIST CONGRESS PARTY MAHARASHTRA PRADESH',
        'ALL INDIA TRINAMOOL CONGRESS', 'BIHAR PRADESH JANTA DAL(UNITED)',
        'RASHTRIYA JANTA DAL', 'AAM AADMI PARTY',
        'ADYAKSHA SAMAJVADI PARTY', 'SHIROMANI AKALI DAL',
        'JHARKHAND MUKTI MORCHA', 'JAMMU AND KASHMIR NATIONAL CONFERENCE',
        'BIJU JANATA DAL', 'GOA FORWARD PARTY',
        'MAHARASHTRAWADI GOMNTAK PARTY', 'SIKKIM KRANTIKARI MORCHA',
        'JANASENA PARTY', 'SIKKIM DEMOCRATIC FRONT'

    Select the most probable party name when giving query.
    Output only the sql query and nothing else. Dont include ``` in output

    This is the table\n {}

    """.format(merged_df)
```

5. Obtain the relevant SQL query from the LLM (Llama-3). We use this SQL query to the database containing the data from the PDFs

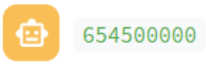
Input



Intermediate code generated by Llama3

```sql
SELECT SUM(DENOMINATIONS) FROM MERGED WHERE NAME_OF_THE_POLITICAL_PARTY = 'AAM AADMI PARTY';
```

Output



Screenshots: