

PolicyMakersGPT(Grp-56)

Problem Statement:

"Policymakers need to assess the impact of their policy on vulnerable populations because understanding the impact of policies is crucial for determining whether they effectively support vulnerable groups or exacerbate their challenges."

For this, we are developing a chatbot to assess the impact of government policies on vulnerable populations, as it presents a critical need to address disparities and ensure equitable policy outcomes. The project aims to create a conversational system capable of collecting data on government initiatives, understanding their implementation, and assessing effects on vulnerable populations, supporting policymakers and marginalized communities.

Our Proposed Solution:

We are proposing the following Methodology for Chatbot Implementation using an Information Retrieval System integrated with a Language Model

Preliminary Steps:

1. We possess a comprehensive dataset concerning government schemes, meticulously organized within a JSON file.
2. API modules have been established to facilitate the retrieval of pertinent tweets from online forums such as Quora and X.
3. A pre-trained and fine-tuned chatbot, resembling a RAG-based model, has been successfully implemented.

User Query Handling Process:

1. Upon **receiving a user query from the UI**, exemplified by inquiries such as "Could you provide more details about the Ramalingaswami Re-Entry Fellowship program?" or "What is the feedback regarding the Ramalingaswami Re-Entry Fellowship?" or "Have the enrolled students received the fellowship under this scheme?",
2. The system proceeds to **identify the pertinent keywords or search terms** pertaining to the queried scheme, exemplified by discerning "Tell me more about the **Ramalingaswami Re-Entry Fellowship** program?" as the search term.

3. Utilizing the identified search term, the system systematically scours through available tweets and Quora posts related to the query, aggregating them into distinct files, namely tweets.csv and quora_posts.csv.
4. Subsequently, **sentiment analysis** is conducted on the textual content of these gathered resources to categorize them as positive or negative reviews, reliable or fake reviews, alongside **keyword extraction** is also performed to ascertain the presence of rare vocabulary within the corpus.
5. Following the acquisition of reviews, a **term frequency-inverse document frequency** (tf-idf) **analysis** is executed on the dataset, followed by leveraging cosine similarity to rank the most relevant results.
6. The collated information is then seamlessly relayed to the **RAG system** chatbot, which adeptly formulates an appropriate response tailored to address the user's query.

Performance Metrics:

We will use the following metrics to evaluate the system performance:

1. **Accuracy:** Measure the proportion of user queries for which the chatbot provides correct and relevant responses. This can be determined by comparing the chatbot's responses against manually labelled ground truth responses.
2. **F1 Score:** The F1 score is the harmonic mean of precision and recall, offering a balanced assessment of the system's performance in terms of both precision and recall. It is particularly useful when the distribution of relevant and irrelevant information in the dataset is imbalanced.
3. **NDCG:** NDCG will be used to assess the ranking quality of the retrieved documents or responses, considering both relevance and the position of relevant items in the ranked list.
4. **User Satisfaction Surveys:** Collect user feedback through surveys or feedback forms to gauge their satisfaction with the chatbot's responses. This qualitative approach provides valuable insights into user experience and can complement quantitative performance metrics.
5. **Response Time:** Measure the chatbot's time to respond to user queries. Prompt responses enhance user experience and satisfaction.

Baseline Results

In the previous report, we outline the data collection process of the policies from the website <https://www.myscheme.gov.in/>. The objective was to gather information about various government schemes listed on the website, organize it by state and ministry, and compile the data into a structured format for further analysis.

Methodology:

- 1) Website Inspection: The website <https://www.myscheme.gov.in/> was visited to understand its structure and content. We identified that the website categorizes schemes state-wise, providing a list of schemes along with their URLs.
- 2) Data Collection: State-wise inspection was conducted to gather data on schemes. Raw HTML data of the scheme lists and their URLs were copied for each state.
- 3) Data Scraping: BeautifulSoup, a Python library, was utilized for web scraping. Details of each scheme were extracted from the scheme URLs. Information such as scheme name, description, eligibility criteria, benefits, and application process were gathered.
- 4) Data Compilation: The scraped data was organized into a structured format. A JSON file was generated to store the compiled data, ensuring ease of access and future analysis.

Code Snippets and Explanation:

```
def get_scheme_list(self,path,state):
    filepath='./BaselineResults/RawHTMLfiles'+path
    with open(filepath,"rb") as f:
        data=f.read()

    soup=BeautifulSoup(data,'html.parser')

    target_div = soup.find_all('div', class_='flex flex-col')
    self.progress_bar(0,self.total)
    for divs in target_div:
        h2_element = divs.find('h2')
        a_element = divs.find('a')

        polycynname=" ".join(a_element.text.replace('\r\n', ' ').split())
        ministry=" ".join(h2_element.text.replace('\r\n', ' ').split())
        scheme_description=self.get_scheme_details(a_element.get("href"))
        Dataset.policy_list.append({"Sno":Dataset.sno,"Policy name":polycynname,"Description":scheme_description,
                                   "Ministry":ministry,"State":state})
        Dataset.sno+=1
    self.progress_bar(Dataset.sno,self.total)

    return
```

Using BeautifulSoup to extract the urls from raw HTML data copied from <https://www.myscheme.gov.in/>

Using Beautifulsoup to extract description of the schemes from the extracted Urls

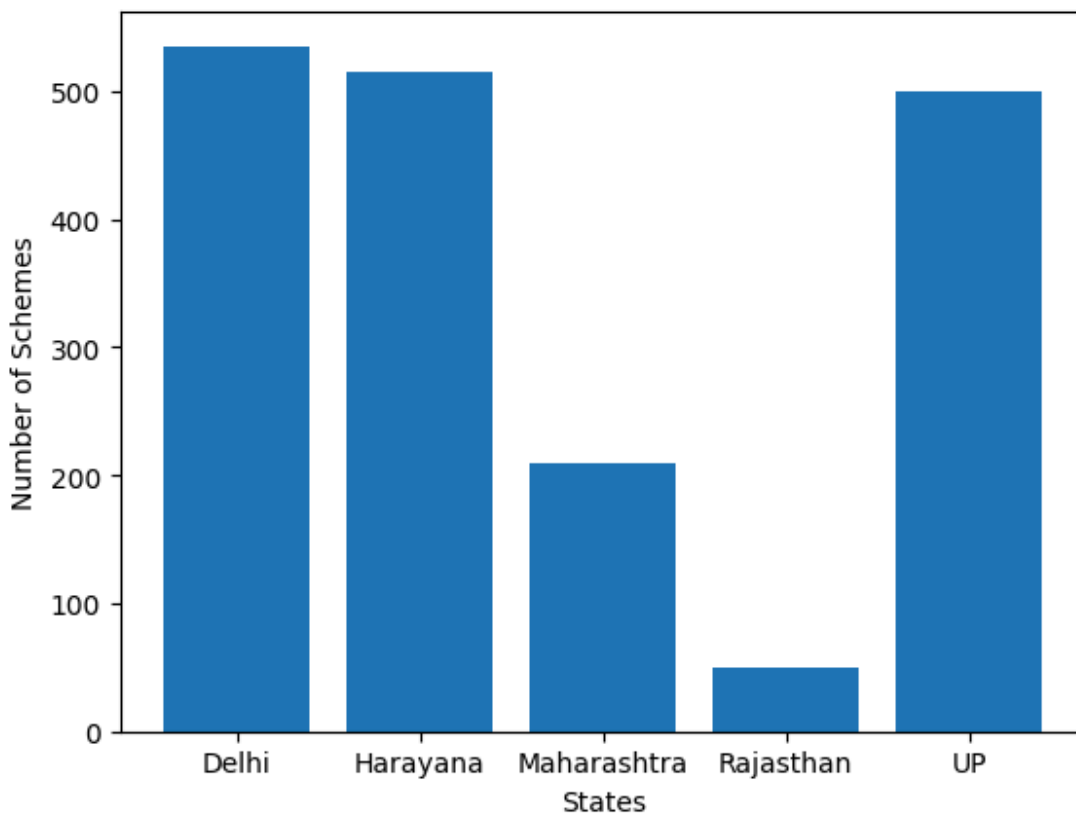
```
Total number of schemes to be compiled: 1810
Compiling Raw HTML files...
|-----| 78.23%
```

```
Total number of schemes to be compiled: 1810
Compiling Raw HTML files...
|-----| 100.06%
Compilation completed.....
```

Data Analysis:

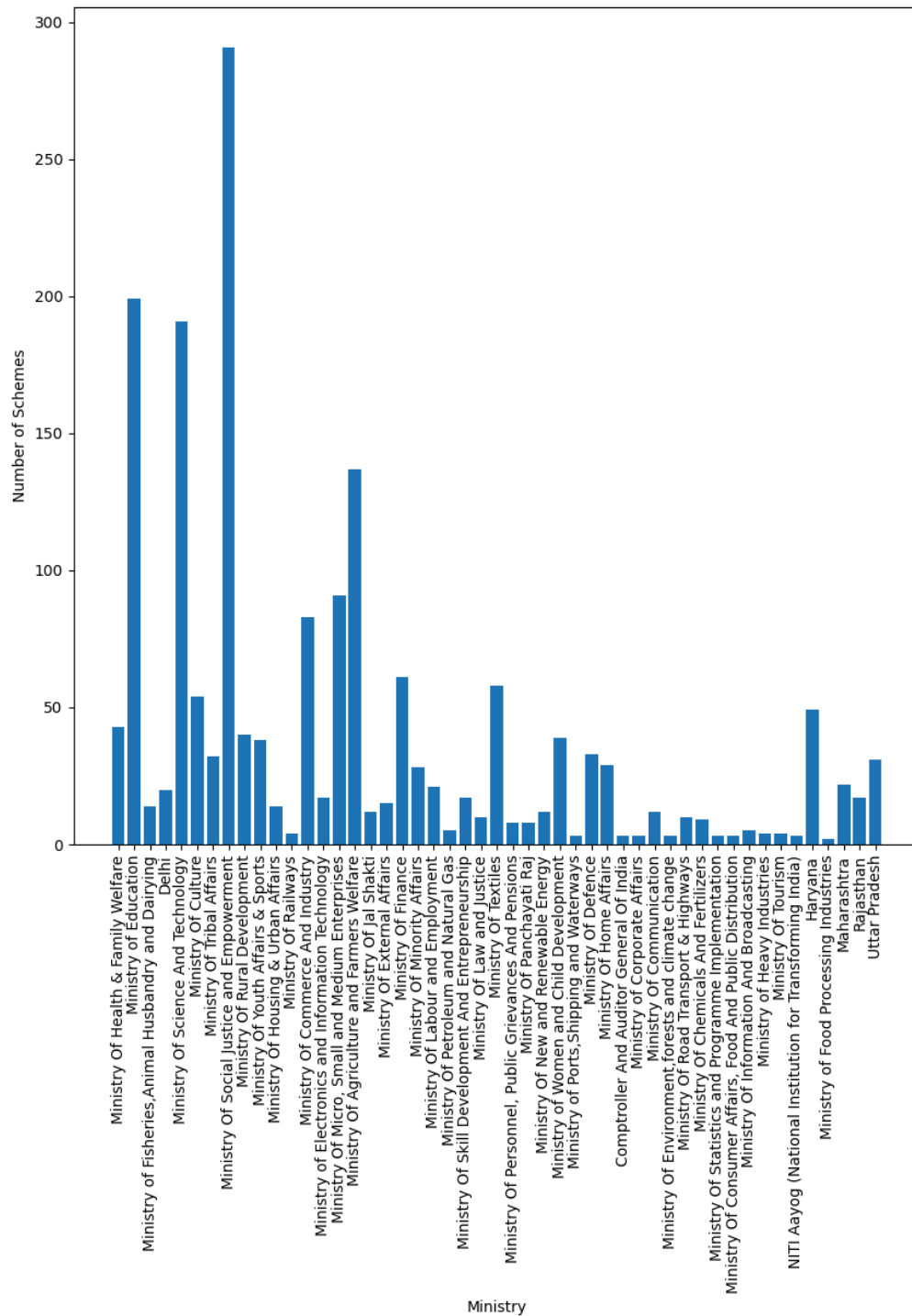
1. Graph of Number of Schemes vs State:

- A bar graph illustrating the number of schemes available in each state was generated.
- This graph provides insights into the distribution of schemes across different states.



2. Graph of Number of Schemes vs Ministry:

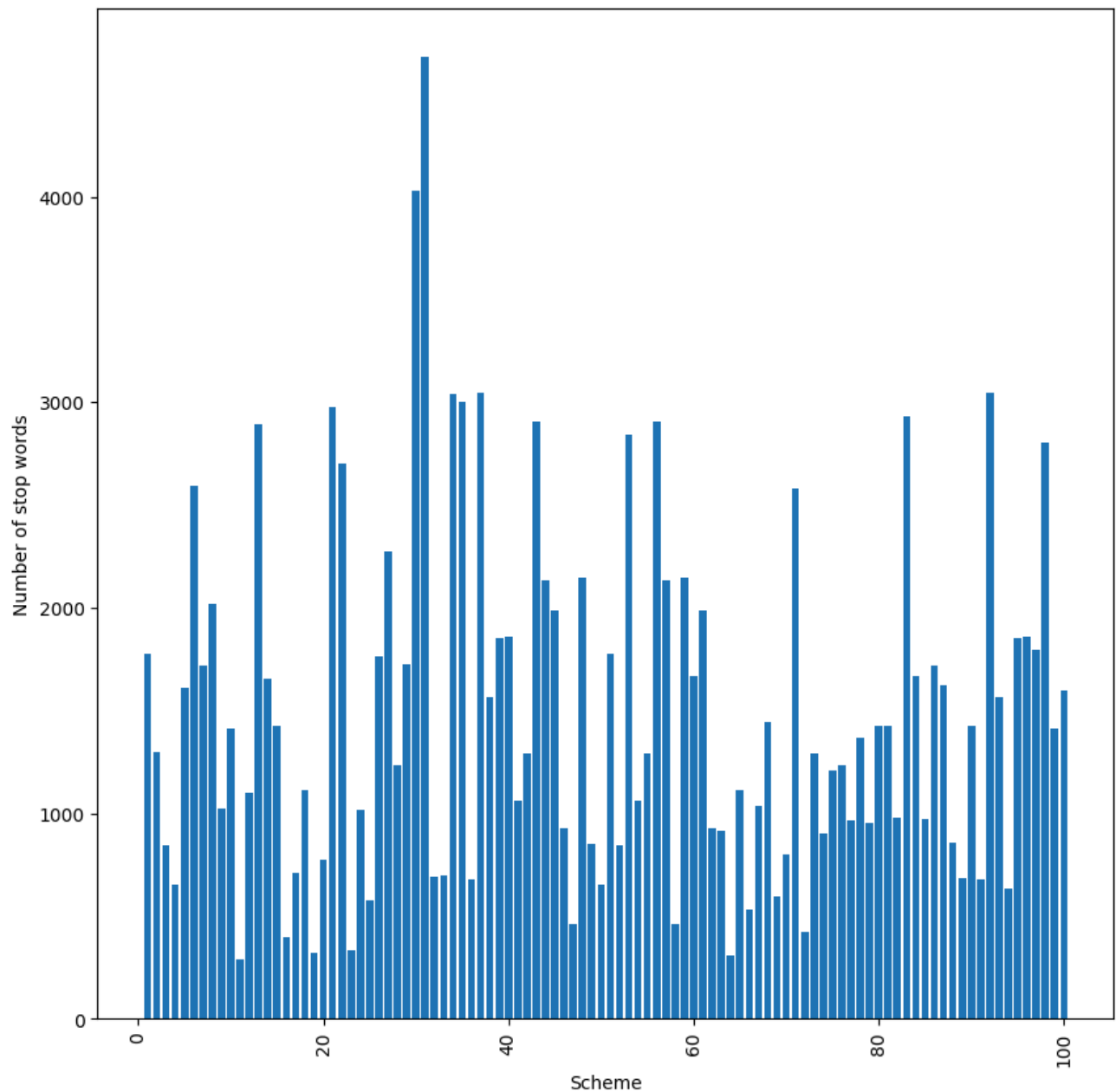
- A bar graph displaying the number of schemes under each ministry was created.
- This graph helps in understanding the distribution of schemes across various ministries.



3. Graph of Number of Stopwords in Each Scheme:

- A histogram depicting the frequency distribution of stopwords in scheme descriptions was generated.

- This graph aids in identifying the prevalence of stopwords in scheme descriptions, which can inform further preprocessing steps.



Retrieval of Tweets and Quora Posts:

We use the Apify client to retrieve the tweets and Quora posts and segregate them according to our requirements, which will be further used for data analysis:

Search Terms:

1. The identified scheme name is used as a search term for both X and Quora posts.

X Search:

1. A dictionary **raw_input** is created, which contains various parameters for configuring the X search, such as the maximum number of tweets to retrieve, whether to include user info, etc.
2. An Apify actor for X web scraping is called with the provided input parameters.
3. Retrieved tweets are stored in a dictionary **tweets**, where each key corresponds to a different attribute of a tweet (e.g., id, date, URL, full text, etc.).
4. Some preprocessing is done to discard irrelevant or erroneous data.
5. A loop iterates over each retrieved tweet and extracts relevant information, such as date, URL, favourite count, retweet count, etc.
6. Extracted tweet data is stored in lists within the tweets dictionary.
7. Finally, the collected tweet data is converted into a pandas DataFrame and saved as a CSV file named "**tweets.csv**".

```

for item in client.dataset(run["defaultDatasetId"]).iterate_items():
    c+=1
    tweets['Date'].append(item['created_at'])
    tweets['Url'].append(item['url'])
    tweets['Id'].append(item['id_str'])
    tweets['Favourite Count'].append(item['favorite_count'])
    tweets['Full Text'].append(item['full_text'])
    tweets['Language'].append(item['lang'])
    tweets['Reply Count'].append(item['reply_count'])
    tweets['Retweet Count'].append(item['retweet_count'])
    tweets['Views Count'].append(item['views_count'])
    hashtags=[]
    for i in item['entities']['hashtags']:
        hashtags.append(i['text'])
    tweets['Hashtags'].append(hashtags)
    image_urls=[]
    video_urls=[]
    if('media' in item['entities']):
        for i in item['entities']['media']:
            if i['type']=='video':
                video_urls.append(i['video_info']['variants'][1]['url'])
            elif i['type']=='photo':
                image_urls.append(i['media_url_https'])
    tweets['Image Urls'].append(image_urls)
    tweets['Video Urls'].append(video_urls)
    user_mentions=[]
    for i in item['entities']['user_mentions']:
        user_mentions.append(i['screen_name'])
    tweets['User Mentions'].append(user_mentions)
    tweets['User Profile Id'].append(item['user_id_str'])
    tweets['User Description'].append(item['user']['description'])
    tweets['User Name'].append(item['user']['screen_name'])
    tweets['User Followers Count'].append(item['user']['followers_count'])

tweets_df=pd.DataFrame(tweets)
tweets_df.to_csv("tweets.csv", index=False)
print("Tweets CSV created")
print("Total tweets retrieved = ",c)
tweets_df.head(5)

```

Quora Search:

1. Another dictionary raw_input is created, this time for configuring the **Quora search**. It specifies parameters like content type, limit, query, etc.
2. An Apify actor for Quora scraping is called with the provided input parameters.
3. Retrieved Quora posts are stored in a dictionary quora_posts, similar to how tweets are stored.
4. Some preprocessing is done to discard irrelevant or erroneous data.
5. A loop iterates over each retrieved Quora post and extracts relevant information, such as ID, date, URL, title, full text, etc.
6. Extracted Quora post data is stored in lists within the **quora_posts dictionary**.

7. Finally, the collected Quora post data is converted into a pandas DataFrame and saved as a CSV file named "quora_posts.csv".

```
for item in client.dataset(run["defaultDatasetId"]).iterate_items():
    c+=1
    if item['type']=='question':
        search=item['__answer']
    else:
        search=item

    full_text=""
    image_urls=[]
    video_urls=[]
    for j in search['content']:
        for k in j['spans']:
            if 'text' not in k:
                if 'modifiers' in k:
                    for l in k['modifiers']:
                        if l=='image':
                            image_urls.append(k['modifiers']['image'])
                        if l=='video':
                            video_urls.append(k['modifiers']['video'])
                    continue
            full_text+=k['text']+' '
    if item['type']=='question':
        quora_posts['Title'].append(item['title'])
    else:
        quora_posts['Title'].append(item['content'][0]['spans'][0]['text'])

    quora_posts['Full Text'].append(full_text)
    quora_posts['Id'].append(search['id'])
    quora_posts['Type'].append(search['type'])
    quora_posts['Url'].append(search['url'])
    quora_posts['Image Urls'].append(image_urls)
    quora_posts['Video Urls'].append(video_urls)
    if('last' in search['user']['names'][0]):
        quora_posts['User name'].append(search['user']['names'][0]['first']+search['user']['names'][0]['last'])
    else:
        quora_posts['User name'].append(search['user']['names'][0]['first'])
    quora_posts['User Id'].append(search['user']['id'])
    quora_posts['Comments Count'].append(search['stats']['comments'])
    quora_posts['Upvotes Count'].append(search['stats']['upvotes'])
    quora_posts['Views Count'].append(search['stats']['views'])
    quora_posts['Shares Count'].append(search['stats']['shares'])

    quora_posts['Date'].append(search['ts_created'])

quora_posts_df=pd.DataFrame(quora_posts)
quora_posts_df.to_csv("quora_posts.csv", index=False)
print("Quora Posts CSV created")
print("Total Quora Posts retrieved = ",c)
quora_posts_df.head(5)
```

Datasets

The datasets formed from X and Quora posts and the use of each column:

Tweets Dataset:

1. **Id**: Unique identifier for each tweet.
2. **Date**: Timestamp indicating when the tweet was posted.

3. **Url:** URL of the tweet.
4. **Full Text:** The complete text content of the tweet.
5. **Hashtags:** List of hashtags included in the tweet.
6. **Views Count:** Number of views the tweet has received.
7. **Favourite Count:** Number of times the tweet has been favourited.
8. **Retweet Count:** Number of times the tweet has been retweeted.
9. **Reply Count:** Number of replies to the tweet.
10. **Language:** Language in which the tweet is written.
11. **Image Urls:** List of URLs of images attached to the tweet (if any).
12. **Video Urls:** List of URLs of videos attached to the tweet (if any).
13. **User Mentions:** List of usernames mentioned in the tweet.
14. **User Profile Id:** Unique identifier for the user who posted the tweet.
15. **User Description:** Description or bio of the user who posted the tweet.
16. **User Name:** User name of the user who posted the tweet.
17. **User Followers Count:** Number of followers the user has who posted the tweet.

Id	Date	Url	Full Text	Hashtags	Views Count	Favourite Count	Retweet Count	Reply Count	Language	Image Urls	Video Urls	User Mentions	User Profile Id	User Description	User Name	User Followers Count
0	Thu Feb 02 06:02:30 +0000 2024	https://twitter.com/DEThella/status/1760546576	DEThella Remastered! Every following...		2625	19	9	0	en		[https://video.twimg.com/ext_tw_video/1760546576/vid/1080x1080/1243467265]	[DEThella, @Hyderabad, @Samarasingh, @p...	1243467265	The Department of Biochemistry, Ministry of E...	DEThella	104931
1	Sat Feb 17 18:32:44 +0000 2024	https://twitter.com/Sol_ROR/status/1758274760...	The last date for the submission of the paper...		781	2	2	1	en					Gateway for Indian researchers across the glob...	Sol_ROR	2250
2	Fri Feb 16 10:42:35 +0000 2024	https://twitter.com/newsco2/status/175846174...	Kind Attention: The last date for submission...		3367	28	35	0	en					Regional Centre for Biotechnology (RCBT) Co...	newsco2	7209
3	Mon Feb 05 10:30:22 +0000 2024	https://twitter.com/IndaScience/status/175...	DEThella @_Kandemal (Biomarkers) Daily on the @fluoridg...	[Biomarkers]	347	3	0	0	en			[DEThella, _Kandemal]	2311132343	IndaScience serves as a catalyst for fast...	IndaScience	21004
4	Mon Feb 05 10:30:22 +0000 2024	https://twitter.com/IndaScience/status/175...	This article highlights the current modifica...		3746	32	6	1	en	[https://pbs.twimg.com/media/GFbdJdU8WtAhdnI.jpg]		[DEThella, _Kandemal]	2311132343	IndaScience serves as a catalyst for fast...	IndaScience	21004

Quora Dataset:

1. **Id:** Unique identifier for each Quora post.
2. **Date:** Timestamp indicating when the post was created.
3. **Url:** URL of the Quora post.
4. **Title:** Title of the Quora post/ Quora Question.
5. **Full Text:** The complete text content of the Quora post.
6. **Type:** Type of Quora post (e.g., answer, question).
7. **Upvotes Count:** Number of upvotes the Quora post has received.
8. **Views Count:** Number of views the Quora post has received.
9. **Shares Count:** Number of times the Quora post has been shared.
10. **Comments Count:** Number of comments on the Quora post.
11. **Image Urls:** List of URLs of images attached to the Quora post (if any).
12. **Video Urls:** List of URLs of videos attached to the Quora post (if any).

13. **User Id:** Unique identifier for the user who posted the Quora post.

14. **User name:** Name of the user who posted the Quora post.

Quora Posts CSV created Total Quora Posts retrieved = 4														
	Id	Date	Url	Title	Full Text	Type	Upvotes Count	Views Count	Shares Count	Comments Count	Image Urls	Video Urls	User Id	User name
0	80644619	2018-04-22T14:22:30.537110Z	https://www.quora.com/What-are-some-prominent...	What are some prominent examples of Indian bra...	Brain Drain Brain Drain is a economic term, wh...	answer	1	3091	0	0	https://qph.cf2.quoracdn.net/main-qimg-as3682...		168640941	MayurDhokchade
1	1477743656698452	2023-04-04T06:20:01.144844Z	https://www.quora.com/What-are-some-fellowship...	What are some fellowships for research availab...	1. Khorana Program for Scholars 2. Fulbright F...	answer	0	49	0	0			233280385	The Google Bot
2	83095376	2020-09-23T09:42:45.997905Z	https://generalknowledge.quora.com/Schemes...	Schemes for Indian researchers residing in for...	Schemes for Indian researchers residing in for...	post	0	958	2	0			1292556795	ShivPrasad
3	137017811	2023-11-26T08:03:07.711494Z	https://a2esam.quora.com/The-significance-of...	The significance of Fellowships in research an...	The significance of Fellowships in research an...	post	0	15	0	0	https://qph.cf2.quoracdn.net/main-qimg-2168bf...		506976693	Azeaz

Data Analysis:

- **Id, Date, and Url:** These columns help in uniquely identifying and referencing each tweet or Quora post and provide metadata like posting date and source URL.
- **Full Text:** Contains the main content of the tweet or Quora post, which is essential for understanding the context and extracting insights.
- **Hashtags and User Mentions:** Provide additional context and insights into the topics and individuals mentioned in the tweet or Quora post.
- **Counts** (Views, Favourites, Retweets, Replies, Upvotes, Shares, Comments): Quantify the engagement and interaction levels for each post, reflecting its popularity and influence.
- **User-related Information** (Profile Id, Description, Name, Followers Count): Provide details about the user who posted the tweet or Quora post, which can be useful for user-based analysis or understanding the author's background and influence.
- **Image and Video URLs:** If available, these URLs point to multimedia content associated with the post, enriching the analysis with visual elements.

By analyzing these datasets and their columns, we gain insights into topics, user behaviour, engagement patterns, and trends on X and Quora platforms.

Why Quora and X?

For the given problem statement, which emphasizes the need for policymakers to assess the impact of their policies on vulnerable populations, both Twitter and Quora can provide valuable insights.

X:

- **Real-Time Feedback:** Provides real-time discussions and feedback on policy issues, allowing policymakers to monitor sentiments and trends among a broad audience.

- **Engagement Metrics:** Offers quantitative indicators like retweets, likes, and replies to gauge public reception and discussion around policies affecting vulnerable groups.
- **Direct Interaction:** Facilitates direct engagement between policymakers and citizens, including vulnerable communities, enabling direct feedback and dialogue.
- **Hashtag Monitoring:** Allows tracking of relevant hashtags to monitor discussions, concerns, and feedback related to policy impacts on vulnerable populations.

Quora:

- **In-depth Discussions:** Hosts longer-form discussions and questions, providing detailed insights and personal experiences related to policy impacts on vulnerable populations.
- **Expertise and Diversity:** Attracts a diverse user base, including experts and individuals with lived experiences, offering a range of perspectives on policy issues.
- **Detailed Explanations:** Users often provide detailed answers, offering nuanced insights into the specific challenges and impacts faced by vulnerable groups due to policies.
- **Question and Answer Format:** Allows policymakers to pose specific questions or scenarios and receive detailed responses from a knowledgeable community.

Combining data from both platforms can provide a comprehensive understanding of policy impacts and public perceptions, catering to different levels of engagement and depth of insights.

Challenges faced

1) Lack of Suitable API:

We encountered difficulty in finding an API capable of retrieving tweets or Quora posts efficiently. Despite extensive searching, we were unable to identify a suitable solution.

2) Costly Twitter API:

Initially, we attempted to utilize the Twitter API developer platform. However, we found that accessing the necessary data would incur significant charges, which were beyond our budget constraints.

3) Limitations of ntscraper:

In our efforts to overcome the API issue, we experimented with the ntscraper Python module. Unfortunately, we found that the results obtained were inconsistent and unreliable, failing to meet our requirements.

4) Challenges with News Articles:

As an alternative, we explored the possibility of retrieving information from news articles. Regrettably, we encountered obstacles in accessing reliable news sources. Even reputable news channels occasionally publish misinformation without proper verification, making it challenging to rely on them for accurate data.

Future Steps:

Additionally, we are capturing user information, which enables us to facilitate cross-verification within the system. This functionality allows us to validate whether a given review is authentic or not by cross-referencing user-profiles and their interactions.

Furthermore, we are storing links to various types of media, such as images and videos. This feature opens up the possibility of implementing a multimodal search system. Such a system could assist policymakers in comprehensively understanding the impact of their policies by analyzing not only textual content but also visual and auditory elements associated with user feedback and discussions.

Improvements Since the Last Deadline:

During the previous deadline, we encountered challenges using sncscrape and the Twitter APIs for retrieving tweets, as they did not yield the desired results. Determined to find more effective methods, we extensively researched and explored alternative approaches for retrieving tweets and Quora posts. However, many of the methods we discovered needed to be updated or proved ineffective after trial and error.

After persevering through numerous attempts, we ultimately discovered Apify, which emerged as a valuable solution for retrieving tweets and Quora posts. This marked a significant improvement from our previous efforts, enabling us to gather the necessary data more efficiently and effectively.

Sentimental analysis on the dataset and keywords extraction:

In sentiment analysis methodology, several techniques can be employed to gauge the sentiment expressed in textual data:

- **Aspect-based sentiment analysis:** Utilizing NLP tools and libraries to perform part-of-speech tagging, syntactic parsing, and named entity recognition to extract

relevant keywords and phrases based on their linguistic properties and relationships within the text.

- **Deep learning-based approaches:** Utilizing advanced deep learning architectures like Long Short-Term Memory networks (LSTMs), Transformers, or BERT (Bidirectional Encoder Representations from Transformers) models to capture complex patterns and dependencies in textual data for sentiment analysis.
- **Sentiment Intensity Analysis:** Gauge the intensity of sentiment expressed in tweets and Quora posts by examining linguistic cues, emotive language, and sentiment modifiers to ascertain the strength of positive or negative sentiment towards government schemes.
- **Contextual Sentiment Analysis:** Consider the broader contextual information surrounding each tweet or Quora post to ensure accurate sentiment interpretation. This involves examining user profiles, followers' responses, and the overall discourse on the platform.

In keyword extraction methodology, we plan to utilise the following to identify relevant keywords or phrases from textual data:

- **Entity Recognition:** Employ named entity recognition techniques to identify key entities mentioned in tweets and Quora posts, such as scheme names, government agencies, policymakers, relevant stakeholders, etc.
- **Domain-Specific Stopword Removal:** Eliminate domain-specific stopwords that may hinder the extraction of meaningful keywords. These stopwords may include common terms irrelevant to policymaker discourse or specific to social media communication.
- **Topic modelling techniques:** Employing topic modelling algorithms such as Latent Dirichlet Allocation (LDA) or Non-negative Matrix Factorization (NMF) to identify topics or themes within the text, and then extracting keywords associated with each topic.

We have yet to think about this step, we are still currently working on ideas and analyzing the dataset.

TF-IDF matrix analysis and Cosine Similarity on the Dataset

In this, we have used Term Frequency-Inverse Document Frequency (TF-IDF Matrix). The significance of a phrase within a document in relation to a collection of documents is assessed using this statistical metric in natural language processing and information retrieval.

Methodology:

1. We have converted the JSON file (government_schemes.json) to CSV. The JSON data contains information about government schemes, including details like scheme names, descriptions, eligibility criteria, etc.
2. Then, Loaded all the CSV files: tweets.csv, quora_posts.csv, and government_schemes.csv.
3. In order to clean and standardize text input for additional analysis, text preprocessing is an essential step before calculating the tf-idf scores.
4. Now, the final TF-IDF matrix calculation. It is calculated based on two components:
 - Term Frequency (TF): The frequency of a term in a document, indicating how often the term appears.
 - Inverse Document Frequency (IDF): A measure of how rare or common a term is across all documents in the collection.

In this, TF-IDF scores are calculated for given queries against documents (policy names, tweets, Quora posts) to determine the relevance of each document to the query.

Code Snippet:

```

import math
import csv
from collections import Counter

# Step 2: TF Calculation
def calculate_tf(document):
    tf = Counter(document.split()) # Here, we are assuming the data is already tokenized
    return tf

# Step 3: IDF Calculation
def calculate_idf(documents):
    N = len(documents)
    idf_values = {}
    all_tokens_set = set([token for document in documents for token in document.split()])
    for token in all_tokens_set:
        contains_token = map(lambda doc: token in doc.split(), documents)
        idf_values[token] = math.log(N / (sum(contains_token)))
    return idf_values

# Step 4: TF-IDF Calculation
def calculate_tfidf(query, documents):
    idf = calculate_idf(documents)
    query_tf = calculate_tf(query)
    tfidf = {}
    for token, tf in query_tf.items():
        if token in idf:
            tfidf[token] = tf * idf[token]
    return tfidf

```

In this, we are calculating the tf-idf matrix.

user query:

```

query1 = "ramalingaswami re-entry fellowship"
query2 = "Ramalingaswami Re-Entry Fellowship"

```

```

document = df1["Policy name"]

query_tfidf1 = calculate_tfidf(query1, document)
query_tfidf2 = calculate_tfidf(query2, document)

print("TF-IDF scores for the query:", query_tfidf1)
print("TF-IDF scores for the query:", query_tfidf2)

```

```

TF-IDF scores for the query: {}
TF-IDF scores for the query: {'Ramalingaswami': 5.421640582580036, 'Re-Entry': 5.421640582580036, 'Fellowship': 2.5038698504957564}

```

```

document = df2["Full Text"]

query_tfidf1 = calculate_tfidf(query1, document)
query_tfidf2 = calculate_tfidf(query2, document)

print("TF-IDF scores for the query:", query_tfidf1)
print("TF-IDF scores for the query:", query_tfidf2)

```

```

TF-IDF scores for the query: {'re-entry': 2.302585092994046, 'fellowship': 1.3862943611198906}
TF-IDF scores for the query: {'Ramalingaswami': 0.22314355131420976, 'Re-Entry': 2.995732273553991, 'Fellowship': 0.9162907318741551}

```

```

document = df3["Full Text"]

query_tfidf1 = calculate_tfidf(query1, document)
query_tfidf2 = calculate_tfidf(query2, document)

print("TF-IDF scores for the query:", query_tfidf1)
print("TF-IDF scores for the query:", query_tfidf2)

```

```

TF-IDF scores for the query: {'re-entry': 1.3862943611198906, 'fellowship': 0.6931471805599453}
TF-IDF scores for the query: {'Ramalingaswami': 0.0, 'Re-Entry': 1.3862943611198906, 'Fellowship': 0.0}

```


The score is calculated for input user query by first computing the TF for the query terms and multiplying them with IDF values to get TF-IDF scores. After obtaining the aggregated TF-IDF scores for all documents, the documents are ranked based on these scores. Typically, documents with higher TF-IDF scores are considered more relevant to the query and are ranked higher. Finally, the ranked list of documents is returned, with the most pertinent documents appearing at the top.

Explanations of the column chosen for TF-IDF and potential improvements:

Column Choice:

- The 'Policy name' column for government schemes is chosen for TF-IDF as it likely contains concise and descriptive names of different schemes, which are crucial for information retrieval or categorisation tasks related to schemes.
- The 'Full Text' column is chosen for tweets and Quora posts because it contains the entire content of the tweet or post, which is essential for capturing the complete context and meaning of the text.

Potential Improvements:

- For tweets and Quora posts, additional features such as user mentions, hashtags, or metadata like timestamps could be included in the TF-IDF analysis if relevant to the task.
- Fine-tuning the TF-IDF parameters, such as adjusting the tokenization method, handling stop words, or using n-grams, can improve the quality of the TF-IDF representations.

Challenges faced and potential improvements of the code:

Challenges:

- Ensuring consistency in text preprocessing across different datasets to maintain the quality of TF-IDF representations.
- Identifying and handling noisy or irrelevant text data that may affect the TF-IDF results.
- Optimizing the code for efficiency, especially for large datasets, to reduce computational time and memory usage.

Potential Improvements:

- Implementing custom text preprocessing functions tailored to the characteristics of each dataset can improve the quality of TF-IDF representations.
- Incorporating techniques for handling imbalanced data or outliers which can affect the IDF calculations and skew the TF-IDF results.

- Parallelizing TF-IDF computations to leverage multi-core processors for faster processing, especially for large datasets.

Additional: Proposed Scoring System:

Using trends from the dataset to sort the top rankings, making a customized scoring system according to the context with the cosine similarity, we can use the likes count to determine the more influential tweet or post when we have similar cosine similarity. We can also use the contextual information from the previous user queries to determine the ranking of the top results.

Chatbot

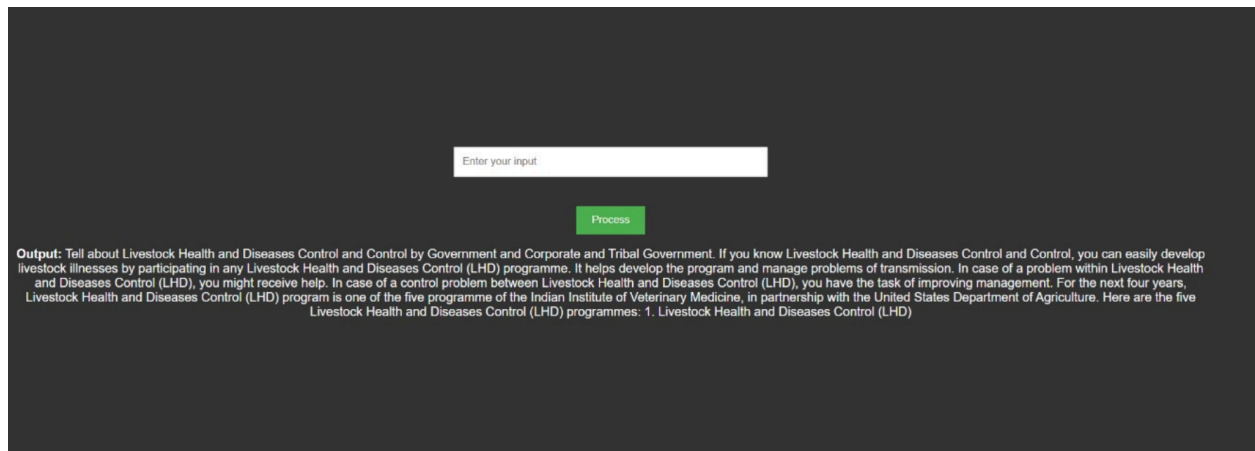
In our project, we're leveraging advanced natural language processing techniques to develop a conversational chatbot using the GPT-2 model. Initially, we fine-tuned GPT-2 on a diverse dataset of schemes encompassing scheme details, descriptions, eligibility criteria, and benefits. This enabled our chatbot to provide information about various government schemes. Moving forward, our focus will shift towards enhancing the chatbot's functionality by integrating user feedback at large and sentiment analysis.

Till the final deadline, we plan to augment the model's training data with rankings derived from the Information Retrieval (IR) system on tweets and posts. As we proceed, we'll leverage rankings derived from our Information Retrieval (IR) system using RAG. This integration will enrich our chatbot's training data, enabling it to discern and prioritize responses based on user relevance and sentiment.

By incorporating user sentiment extracted from social media platforms like X and Reddit, we aim to provide users with insights into public opinion regarding government schemes. This iterative approach ensures our chatbot evolves to better serve policymakers' needs, facilitating informed decision-making and fostering greater engagement with government initiatives.

Moreover, we're exploring the possibility of upgrading our model from GPT-2 to a more powerful transformer, such as GPT-3. The enhanced capabilities of GPT-3 could significantly improve our chatbot's performance and responsiveness.

User Interface



Enter your input

Process

Output: Tell about Livestock Health and Diseases Control and Control by Government and Corporate and Tribal Government. If you know Livestock Health and Diseases Control and Control, you can easily develop livestock illnesses by participating in any Livestock Health and Diseases Control (LHD) programme. It helps develop the program and manage problems of transmission. In case of a problem within Livestock Health and Diseases Control (LHD), you might receive help. In case of a control problem between Livestock Health and Diseases Control (LHD), you have the task of improving management. For the next four years, Livestock Health and Diseases Control (LHD) program is one of the five programme of the Indian Institute of Veterinary Medicine, in partnership with the United States Department of Agriculture. Here are the five Livestock Health and Diseases Control (LHD) programmes: 1. Livestock Health and Diseases Control (LHD)

The User Interface consists of an input box that takes the input from the user and presents the output for the query. User Interface is made with the Django framework because Django is python based and our training model is written in python.

For this review we trained our model using the GPT-2 model provided by hugging face. The training data consists of more than 1800 schemes including scheme details, descriptions, eligibility criteria, and benefits.

```
from transformers import GPT2LMHeadModel, GPT2Tokenizer, GPT2Config
from transformers import TextDataset, DataCollatorForLanguageModeling
from transformers import Trainer, TrainingArguments
```

```
def fine_tune_gpt2(model_name, train_file, output_dir):
    # Load GPT-2 model and tokenizer
    model = GPT2LMHeadModel.from_pretrained(model_name)
    tokenizer = GPT2Tokenizer.from_pretrained(model_name)

    # Load training dataset
    train_dataset = TextDataset(
        tokenizer=tokenizer,
        file_path=train_file,
        block_size=128)

    # Create data collator for language modeling
    data_collator = DataCollatorForLanguageModeling(
        tokenizer=tokenizer, mlm=False)

    # Set training arguments
    training_args = TrainingArguments(
        output_dir=output_dir,
        overwrite_output_dir=True,
        num_train_epochs=5,
```

```
        per_device_train_batch_size=4,
        save_steps=10_000,
        save_total_limit=2,
    )

    # Train the model
    trainer = Trainer(
        model=model,
        args=training_args,
        data_collator=data_collator,
        train_dataset=train_dataset,
    )

    trainer.train()
```

```

# Save the fine-tuned model
model.save_pretrained(output_dir)
tokenizer.save_pretrained(output_dir)

fine_tune_gpt2("gpt2", "preprocessed_scheme_data.txt", "model_output")

```

Potential Improvements:

After training the model and testing it we realized that the GPT-2 model needs more input questions to train on and give better results. We can either test other LLMs like the LLAMA provided by Meta, and compare the results with our model.

Challenges

Getting the desired output from the trained model is quite difficult as it depends on the input and output queries we use for training, for example we have trained the model giving these input and output prompts for each policy.

```

train_dataset.append(f"input: What are the benefits of {item['Policy name']} policy?")
train_dataset.append(f"output: {item['Description']['benefits']}")
train_dataset.append("<|endoftext|>")
train_dataset.append(f"Who is eligible for {item['Policy name']} policy?")
train_dataset.append(f"output: {item['Description']['eligibility']}")
train_dataset.append("<|endoftext|>")

```