GRIHAM-C: Grievance Hierarchical Analysis and Monitoring – through Clustering

Developed and Presented By: S. Meenakshi, B.E., Computer Sci. & Engg

Email id: smeenakshi1997@gmail.com   Phone No: +91 9486780698

Abstract:

"GRIHAM-C: Grievance Hierarchical Analysis and Monitoring - through Clustering" is a comprehensive system designed to enhance the analysis and monitoring of grievances through advanced natural language processing techniques, like a home to monitoring the grievances once they are submitted. The project leverages state-of-the-art transformers language models, specifically tailored for zero-shot classification of grievances and textual submissions. The system integrates an independent summarizer model and translation capabilities, contributing to efficient grievance monitoring until it reaches the last-mile officer.

Introduction:

The heart of the project is a transformers language model fine-tuned for zero-shot classification, enabling the system to categorize grievances using predefined labels that are pertinent with user-defined clustering use cases and that do not pertain to common language labels. This flexibility allows the model to adapt to varying datasets and efficiently cluster grievances into hierarchical categories.

Additionally, an independent summarizer model is incorporated to provide concise representations of lengthy textual submissions. This summarization aids in quick comprehension and decision-making by extracting essential information from the grievances.

Furthermore, a translation model is integrated to facilitate multi-lingual support. This feature ensures that grievances submitted in different Indian languages can be translated into a common language, fostering a more inclusive and accessible grievance monitoring system.

"GRIHAM-C" offers a robust and versatile solution for the hierarchical analysis and monitoring of grievances. The combination of a specialized zero-shot classification model, an efficient summarizer, and a translation module collectively enhances the system's capabilities, providing valuable insights for effective grievance resolution. The system's outputs are generated in JSON and CSV formats, facilitating seamless sharing with last-mile officers for further analysis and timely resolution of grievances.

Technology used:

Programming Language: Python

- NLP tool: The proposed transformer model (NLP specific open source neural networks) for text classification, is modelled using Deep Learning frameworks such as Tensorflow and PyTorch, in zero-shot classification task. The model trained using custom dataset is an mnli model **cluster_model_worksfine**, an inference NLP model trained in zero-shot classification task for demonstration purposes.
- Front-end tool: Streamlit library is utilized to realize the solution and the front-end of this demo has been built with it. It requires the report classification_result.json, generated by the backend system for clustering, for effective visualization of the clustering offered by the solution model.
- Training Module: Training module consists of a dataset preparation code which assigns the necessary parameter to the data and prepares it for training. Training code is as per the steps involved in the fine tuning of a large inference language for demonstration purpose.

Process:

There are 3 main modules in which the classification model forms the primary task of the solution. The 3 modules are:

- Classification module
- Summarization module
- Translation module (for Indian language grievances)

Classification module:

The model used in the solution is a self-trained mnli model: cluster_model_worksfine like the BART-large-mnli utilized for zero-shot-classification task.

The zero-shot classification system using models like BART-large-mnli involves using a pre-trained language model to classify text into multiple classes without any prior training on examples of those specific classes. Here is a step-by-step explanation:

1. **Model Architecture:**
    - **Model Choice:** In this case, the model chosen is BART-large-mnli. BART (Bidirectional and Auto-Regressive Transformers) is a transformer-based neural network architecture.
    - **MNLI Task:** The model is pre-trained on the MNLI (MultiNLI) dataset. MNLI is a natural language inference dataset, where the model learns to predict whether one sentence entails, contradicts, or is neutral with respect to another sentence.
2. **Zero-Shot Classification:**
    - **Input Encoding:** Given a text input, the model encodes the input using its pre-trained weights.
    - **Prompt Engineering:** Instead of traditional fine-tuning on a specific classification task, zero-shot classification relies on prompt engineering. A prompt is a textual description that guides the model to perform a specific task without explicit training data. In this case, it is the model that is trained with provided dataset and the task is zero-shot classification.
    - **Multi-Class Classification:** The model is designed to handle multiple classes. It can perform multi-class classification even if it hasn't been explicitly trained on examples from each class.
3. **Task-Specific Labels:**
    - **Candidate Labels:** The user provides a set of candidate labels or classes that they want the model to classify the input into. These labels act as potential categories or topics for the classification task.
4. **Inference:**
    - **Prediction:** The model generates predictions for each candidate label based on the input and the provided prompt. The output is a probability distribution over the candidate labels.
    - **Top-K Prediction:** The top-K predicted labels with the highest probabilities are often considered as the model's classification.
5. **Output:**
    - **Probabilities:** The model returns a probability distribution indicating the likelihood of the input belonging to each of the provided candidate labels.

In summary, zero-shot classification with models like BART-large-mnli leverages pre-trained language models and prompt engineering to classify text into user-defined categories without the need for specific training examples for those categories. The model generalizes from its pre-training on diverse language understanding tasks to perform classification on a wide range of tasks.

Note: Fine-tuning to create a custom transformer model maybe a heavy process, but essentially done only once before the deployment of the clustering tool in the server. The fine-tuning itself is independent and is used to determine the accuracy and fit the model in fulfilling the requirements, especially when the format of input varies than that of the pre-training. This model **cluster_model_worksfine**, has been tuned with **Accuracy: 1.**0, and to **run on GPU and CPU**.

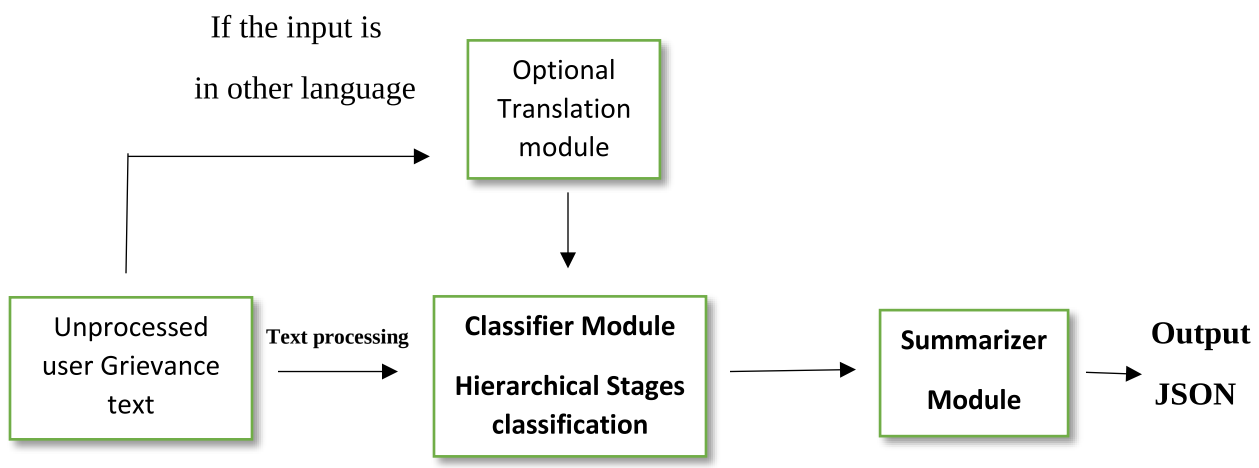
Summarization Module:

An Independent summarization transformers model **bart_large_cnn_fine_tuned_summarizer**, is fine-tuned for this specific grievance text input, thus offering a comprehensive access to the last-mile officer to get the gist and important details of the grievance, thus allowing the process to be streamlined and automated for fast response time. The fine-tuning code is provided with the solution code

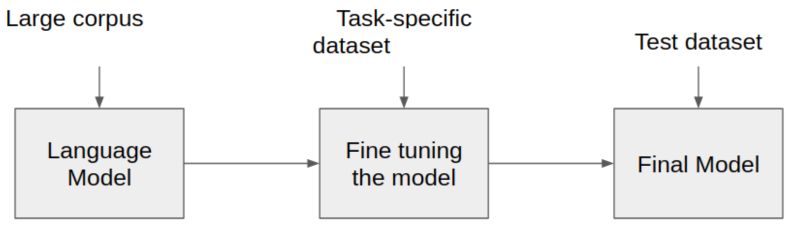
Translation Module:

Independent translation module consists of the transformer model for translation off the open source models that enables the system to automatically translate the other Indian language grievances with an additional identifier of the input language entered during the submission of the grievance. Does not require third party translation tools and can be fine-tuned with Indic language datasets.

Proposed system schema:


If the input is
in other language
Optional Translation module
Unprocessed user Grievance text
Text processing
Classifier Module
Hierarchical Stages classification
Summarizer Module
Output
JSON

Proposed training of above mentioned transformers language models as a flow diagram:



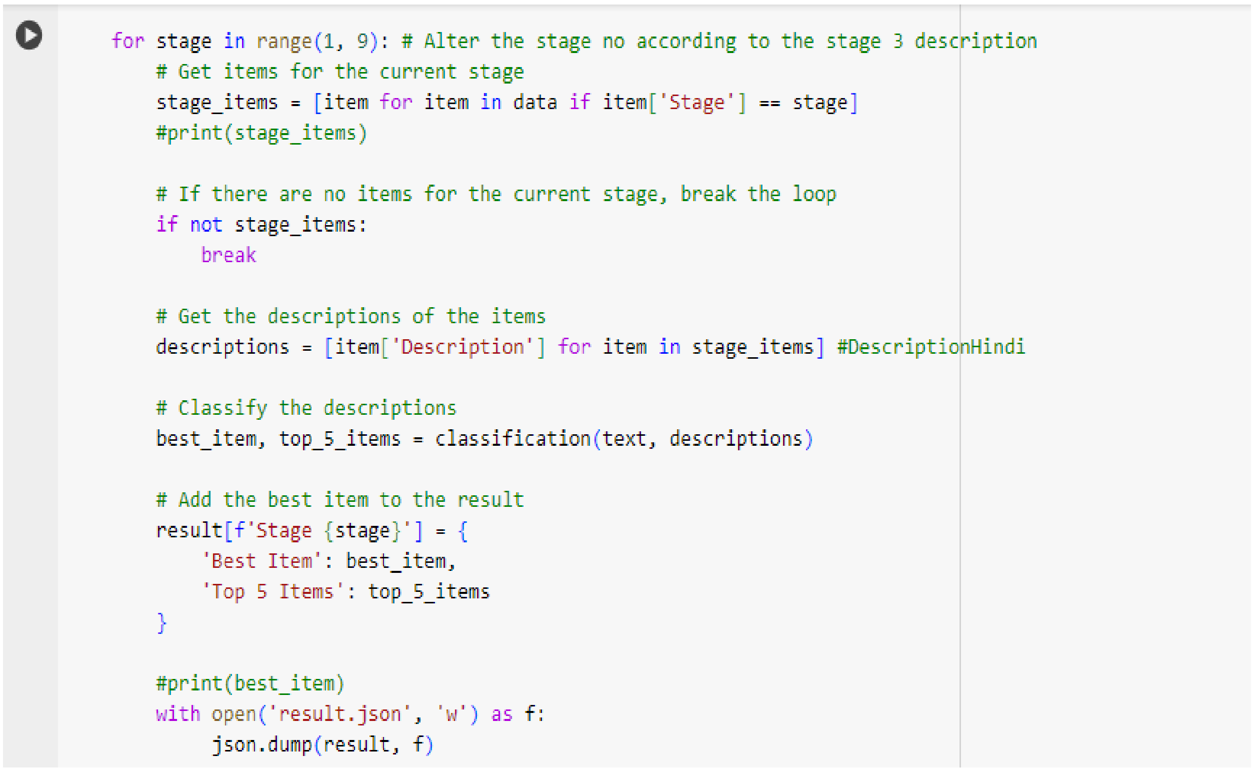
Key Features in the Classification module in the backend submission portal:

- The usage of this model is improvised by coding the data handling part of any grievance that is submitted by the user to **automate the identification of the labels** for classification in the backend.
- The file **CategoryCode_Mapping_v2.json** is used for the extensive **identification of the input labels for the classification** thus using **only the input organization's labels in each stage**, hence guarantee the **hierarchical analytics** of the sequence.
- The **sequence of the steps** that are followed are:
  - Input of the grievance text is classified according to the **org-code entered at the time of submission**.
  - The **Stage 1 parent code is then passed to fetch the labels in the next Stage** i.e., Stage 2, along with the parent codes of all the labels.
  - The **classifier model is run against the input text stream with the classification labels of that Stage**. The model also fetches the **next <u>Top 5 possible classifications</u>** as there is a possibility that the classifications can give the officer an idea of the classification text. This can be used as a guide for the last-mile officers.
  - This process is repeated until the **text is processed against all the possible Stages and lables** in the classification based on org_code.
- The system **supports clustering both in the forward and backward clustering**. Hence, assists in clustering the grievances based on the Stages classified, as sub-classification. For example, in the grievances classified under org_code: CBODT -> Stage 2: Direct Taxes, all the Grievances classified under Stage 2: Direct Taxes related issues will be grouped.

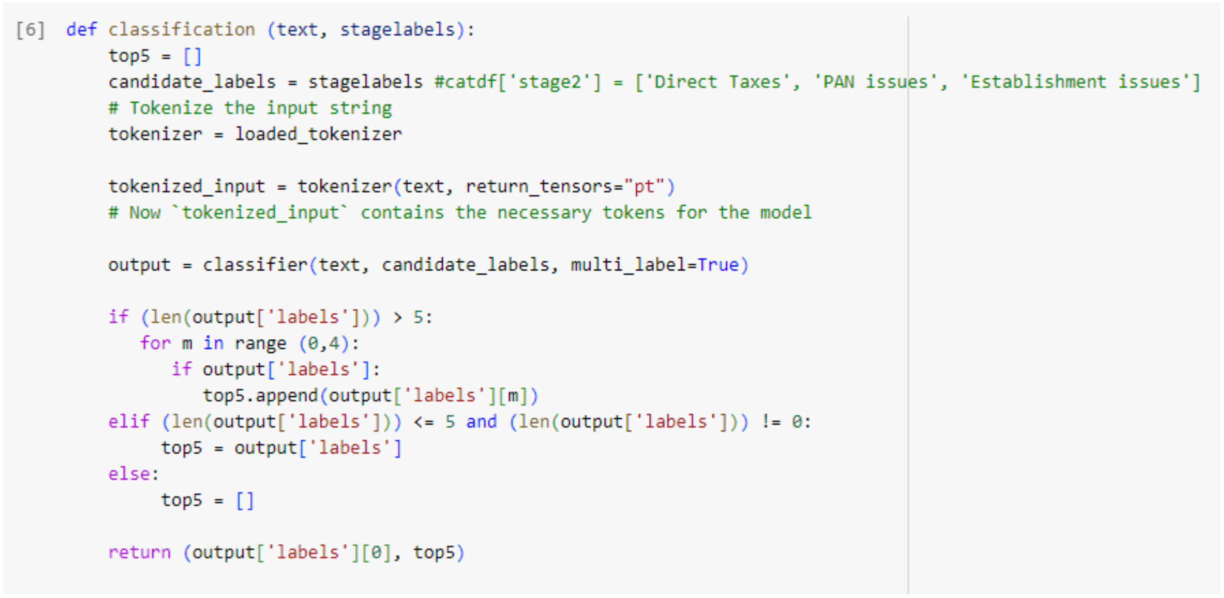
Note: All the Grievance text inputs are translated to common language: i.e., English. The raw dataset provided has multiple language texts, which can be further trained on the language model as an **additional layer after the classification labels for Stages are available** in all languages in the future. Currently only few labels of organizations available in dataset are in Hindi, which is less for effective tuning.

However, **other languages are supported in this current demo after translation and other languages expressed in English is also included in current training for better classification and summary**, which is working just as good, with similar turnaround time.

Screen grab of the code required to identify the tokens for each stage:

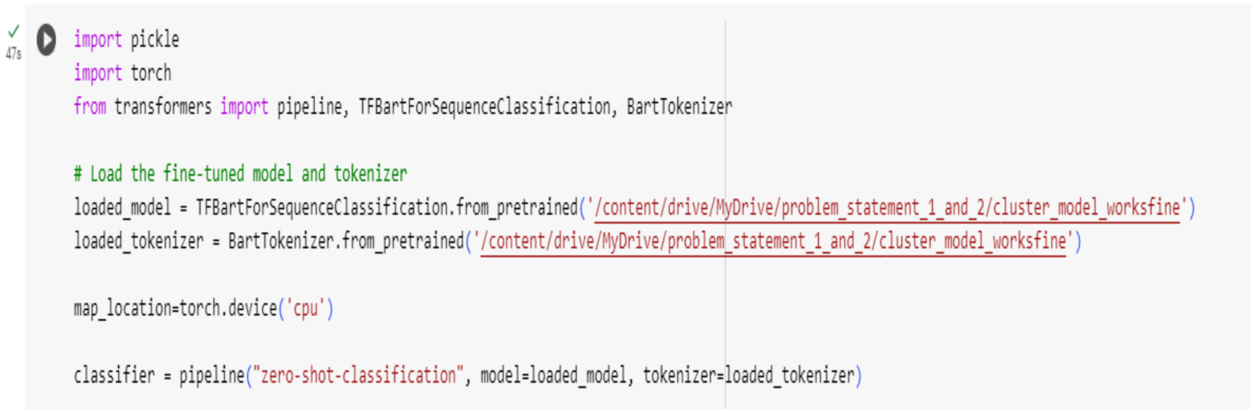


```
for stage in range(1, 9): # Alter the stage no according to the stage 3 description
    # Get items for the current stage
    stage_items = [item for item in data if item['Stage'] == stage]
    #print(stage_items)

    # If there are no items for the current stage, break the loop
    if not stage_items:
        break

    # Get the descriptions of the items
    descriptions = [item['Description'] for item in stage_items] #DescriptionHindi

    # Classify the descriptions
    best_item, top_5_items = classification(text, descriptions)

    # Add the best item to the result
    result[f'Stage {stage}'] = {
        'Best Item': best_item,
        'Top 5 Items': top_5_items
    }

    #print(best_item)
    with open('result.json', 'w') as f:
         json.dump(result, f)
```



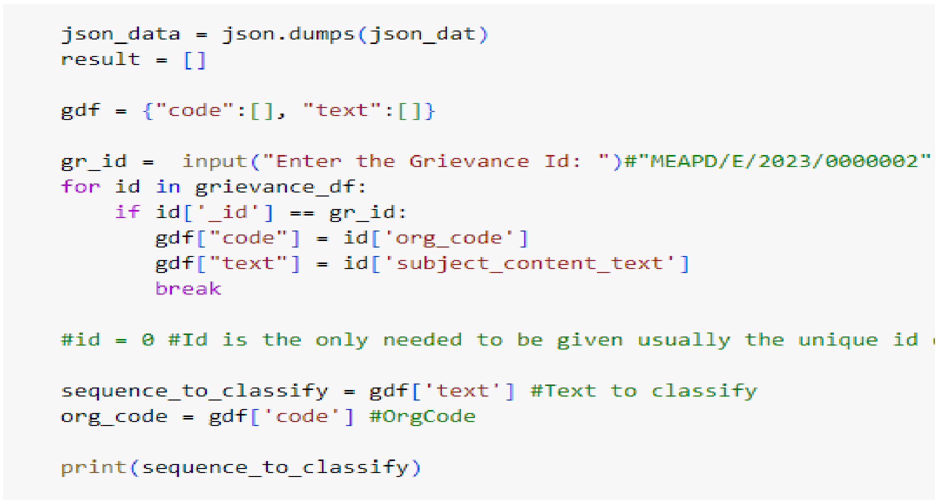
```
[6] def classification (text, stagelabels):
        top5 = []
        candidate_labels = stagelabels #catdf['stage2'] = ['Direct Taxes', 'PAN issues', 'Establishment issues']
        # Tokenize the input string
        tokenizer = loaded_tokenizer

        tokenized_input = tokenizer(text, return_tensors="pt")
        # Now `tokenized_input` contains the necessary tokens for the model

        output = classifier(text, candidate_labels, multi_label=True)

        if (len(output['labels'])) > 5:
           for m in range (0,4):
              if output['labels']:
                 top5.append(output['labels'][m])
        elif (len(output['labels'])) <= 5 and (len(output['labels'])) != 0:
             top5 = output['labels']
        else:
             top5 = []

        return (output['labels'][0], top5)
```

The model cluster_model_worksfine is loaded using transformer loader and tokenizer (loaded only once):



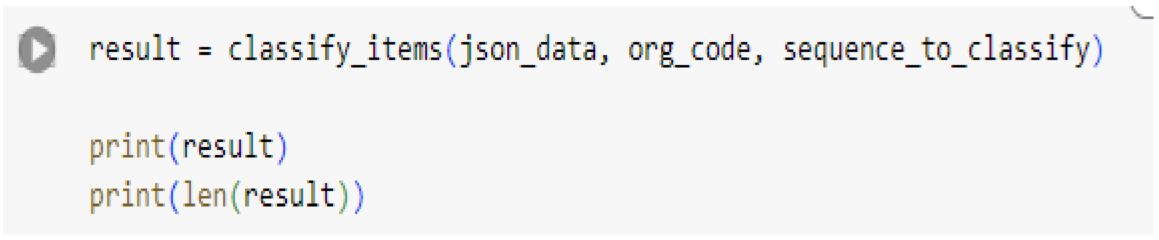
```
import pickle
import torch
from transformers import pipeline, TFBartForSequenceClassification, BartTokenizer

# Load the fine-tuned model and tokenizer
loaded_model = TFBartForSequenceClassification.from_pretrained('/content/drive/MyDrive/problem_statement_1_and_2/cluster_model_worksfine')
loaded_tokenizer = BartTokenizer.from_pretrained('/content/drive/MyDrive/problem_statement_1_and_2/cluster_model_worksfine')

map_location=torch.device('cpu')

classifier = pipeline("zero-shot-classification", model=loaded_model, tokenizer=loaded_tokenizer)
```

Input section:



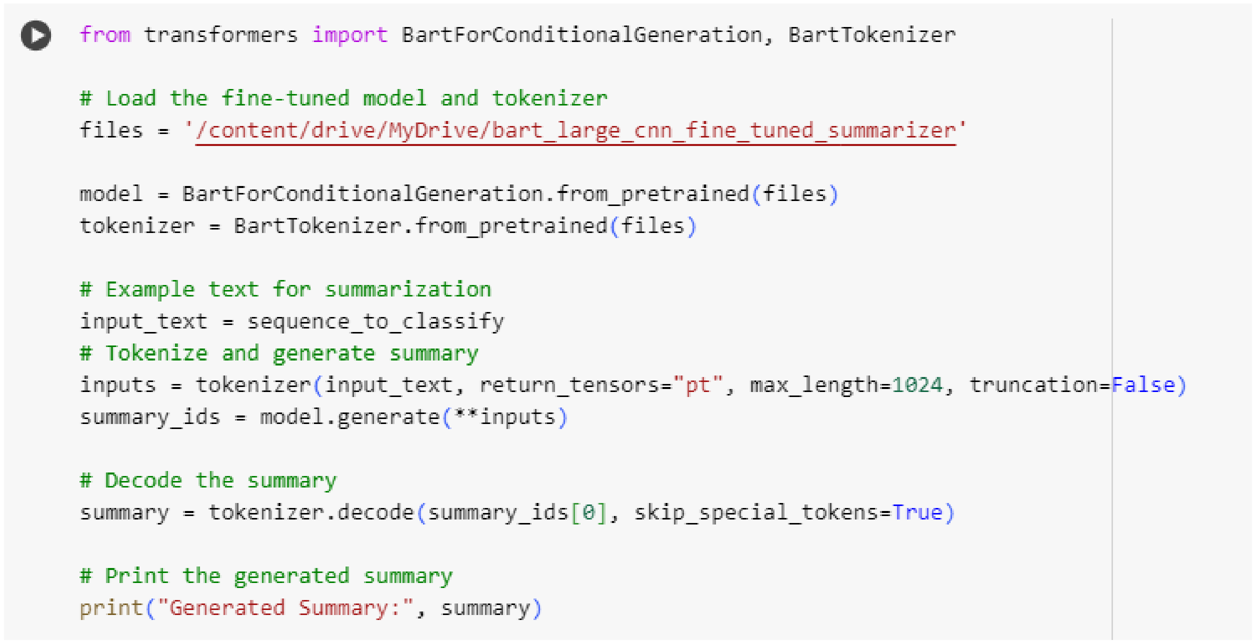
```
json_data = json.dumps(json_dat)
result = []

gdf = {"code":[], "text":[]}

gr_id =  input("Enter the Grievance Id: ")#"MEAPD/E/2023/0000002"
for id in grievance_df:
    if id['_id'] == gr_id:
        gdf["code"] = id['org_code']
        gdf["text"] = id['subject_content_text']
        break

#id = 0 #Id is the only needed to be given usually the unique id

sequence_to_classify = gdf['text'] #Text to classify
org_code = gdf['code'] #OrgCode

print(sequence_to_classify)
```

Classification Function Snippet:



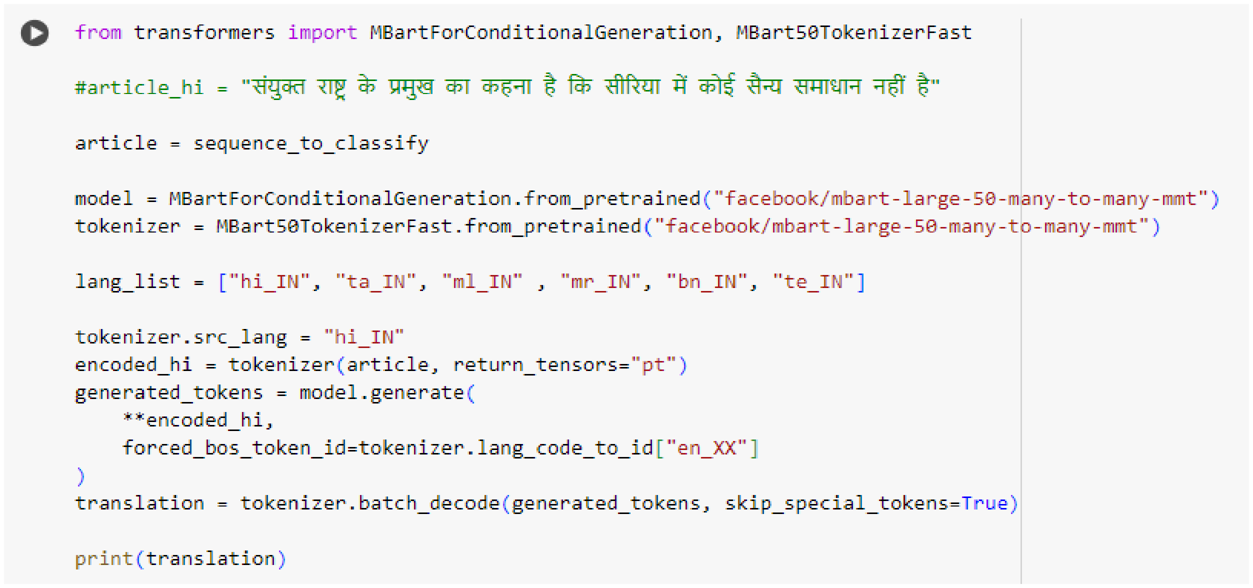
```
result = classify_items(json_data, org_code, sequence_to_classify)

print(result)
print(len(result))
```

Summary Module:



```python
from transformers import BartForConditionalGeneration, BartTokenizer

# Load the fine-tuned model and tokenizer
files = '/content/drive/MyDrive/bart_large_cnn_fine_tuned_summarizer'

model = BartForConditionalGeneration.from_pretrained(files)
tokenizer = BartTokenizer.from_pretrained(files)

# Example text for summarization
input_text = sequence_to_classify
# Tokenize and generate summary
inputs = tokenizer(input_text, return_tensors="pt", max_length=1024, truncation=False)
summary_ids = model.generate(**inputs)

# Decode the summary
summary = tokenizer.decode(summary_ids[0], skip_special_tokens=True)

# Print the generated summary
print("Generated Summary:", summary)
```

Additional Translation Module:



```python
from transformers import MBartForConditionalGeneration, MBart50TokenizerFast

#article_hi = "संयुक्त राष्ट्र के प्रमुख का कहना है कि सीरिया में कोई सैन्य समाधान नहीं है"

article = sequence_to_classify

model = MBartForConditionalGeneration.from_pretrained("facebook/mbart-large-50-many-to-many-mmt")
tokenizer = MBart50TokenizerFast.from_pretrained("facebook/mbart-large-50-many-to-many-mmt")

lang_list = ["hi_IN", "ta_IN", "ml_IN" , "mr_IN", "bn_IN", "te_IN"]

tokenizer.src_lang = "hi_IN"
encoded_hi = tokenizer(article, return_tensors="pt")
generated_tokens = model.generate(
    **encoded_hi,
    forced_bos_token_id=tokenizer.lang_code_to_id["en_XX"]
)
translation = tokenizer.batch_decode(generated_tokens, skip_special_tokens=True)

print(translation)
```

An Example of the Input and output:

Input:

```
Labour and Employment >> Transfer related issues >> Transfer in/out/Form
13/ Online transfer related issues

Name and Address of Establishment : PERSOLKELLY INDIA PRIVATE LIMITED
```

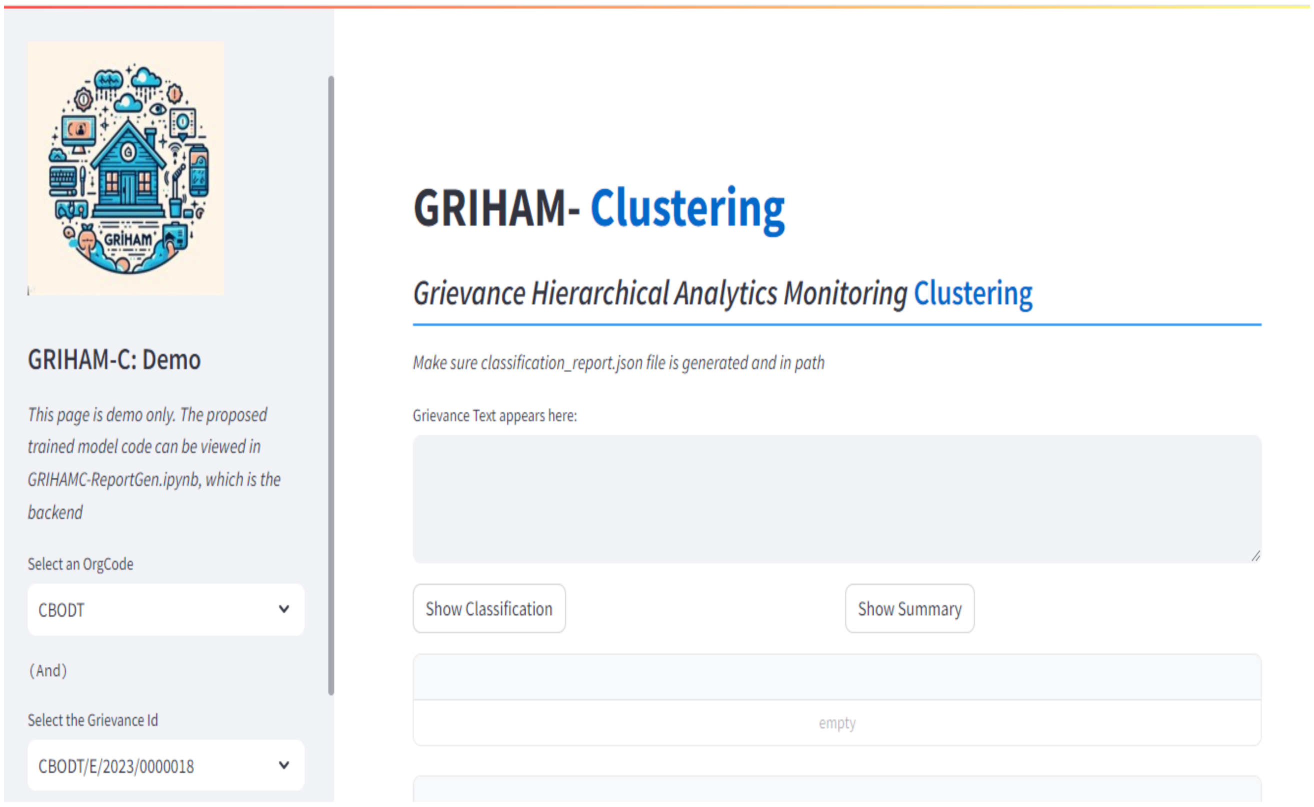
```
UAN No. : X0X3X9X5X8X0
PF Code/ PF Account No. : X-X-X-X-X
PPO No. : Not Provided
Scheme Certificate Number : Not Provided
PF Office : Regional Office, Bandra 1
-----------------------
Hello dear sir madam
My name is SYED SHEBAAZ AHMED
My uan number is X0X3X9X5X8X0
My pf number is X-X-X-X-X
My company name is PERSOLKELLY INDIA PRIVATE LIMITED
Hers about company details

Sir i apply for transfer request but epfo still rejected reason is Break
statement
Rejected reason: Your Claim  Claim Id - X-X-X-X-X  has been rejected due
to : 1) PAYMENT NOT RECEIVED FOR THE MONTH OF 05/2022 CLARIFY THE SAME

So i contact to my Hr he&#39;s given to me Clearfication Letter break
statment and also sending Epfo office regarding
So i request you to Epfo department please approve my transfer
Please find the attachment pdf file
Please reply as soon as possible
```

Output of classification:

"classification_result": {
    "Stage 1": {
      "Best Item": "Labour and Employment",
      "Top 5 Items": [
        "Labour and Employment"
      ],
      "Best Item Code": 2173
    },
    "Stage 2": {
      "Best Item": "Transfer related issues",
      "Top 5 Items": [
        "Transfer related issues",
        "Employer's grievance",
        "Compliance related Issues",
        "Online Claim Application"

```
    ],
    "Best Item Code": 2347
  },
  "Stage 3": {
    "Best Item": "Transfer in/out/Form 13/ Online transfer related issues",
    "Top 5 Items": [
      "Transfer in/out/Form 13/ Online transfer related issues",
      "Transfer of PF",
      "Others",
      "Non receipt of Annexure K"
    ],
    "Best Item Code": 2398
  }
}, ….
```

Screen Grab of the visualizing tool built using Streamlit:



GRIHAM-C: Demo

This page is demo only. The proposed trained model code can be viewed in GRIHAMC-ReportGen.ipynb, which is the backend

Select an OrgCode

CBODT

(And)

Select the Grievance Id

CBODT/E/2023/0000018

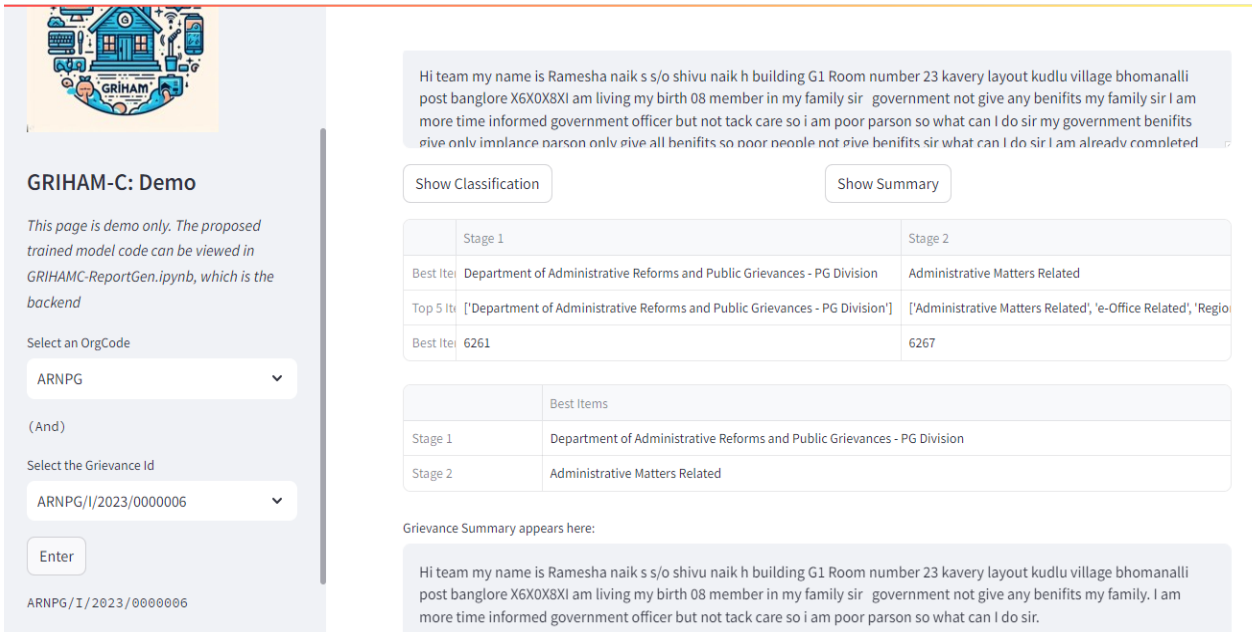
GRIHAM- Clustering

Grievance Hierarchical Analytics Monitoring Clustering

Make sure classification_report.json file is generated and in path

Grievance Text appears here:

Example of output in visualizing tool:



GRIHAM-C: Demo

This page is demo only. The proposed trained model code can be viewed in GRIHAMC-ReportGen.ipynb, which is the backend

Select an OrgCode: ARNPG

(And)

Select the Grievance Id: ARNPG/I/2023/0000006

Enter

ARNPG/I/2023/0000006

Hi team my name is Ramesha naik s s/o shivu naik h building G1 Room number 23 kavery layout kudlu village bhomanalli post banglore X6X0X8XI am living my birth 08 member in my family sir government not give any benifits my family sir I am more time informed government officer but not tack care so i am poor parson so what can I do sir my government benifits give only implance parson only give all benifits so poor people not give benifits sir what can I do sir I am already completed

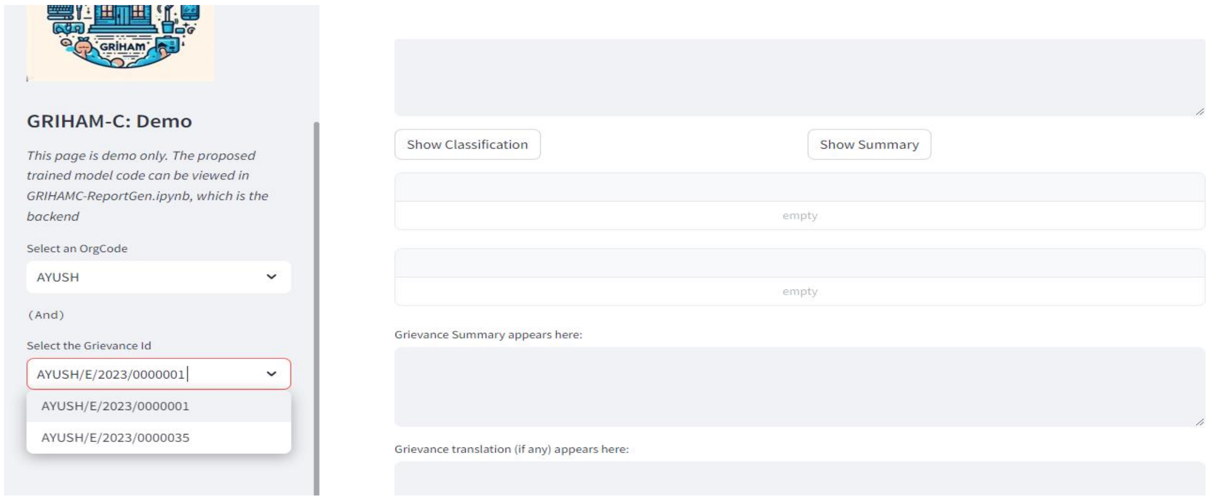
Show Classification | Show Summary

	Stage 1	Stage 2
Best Ite	Department of Administrative Reforms and Public Grievances - PG Division	Administrative Matters Related
Top 5 It	['Department of Administrative Reforms and Public Grievances - PG Division']	['Administrative Matters Related', 'e-Office Related', 'Regio
Best Ite	6261	6267

	Best Items
Stage 1	Department of Administrative Reforms and Public Grievances - PG Division
Stage 2	Administrative Matters Related

Grievance Summary appears here:

Hi team my name is Ramesha naik s s/o shivu naik h building G1 Room number 23 kavery layout kudlu village bhomanalli post banglore X6X0X8XI am living my birth 08 member in my family sir government not give any benifits my family. I am more time informed government officer but not tack care so i am poor parson so what can I do sir.



GRIHAM-C: Demo

This page is demo only. The proposed trained model code can be viewed in GRIHAMC-ReportGen.ipynb, which is the backend

Select an OrgCode: AYUSH

(And)

Select the Grievance Id: AYUSH/E/2023/0000001

AYUSH/E/2023/0000001

AYUSH/E/2023/0000035

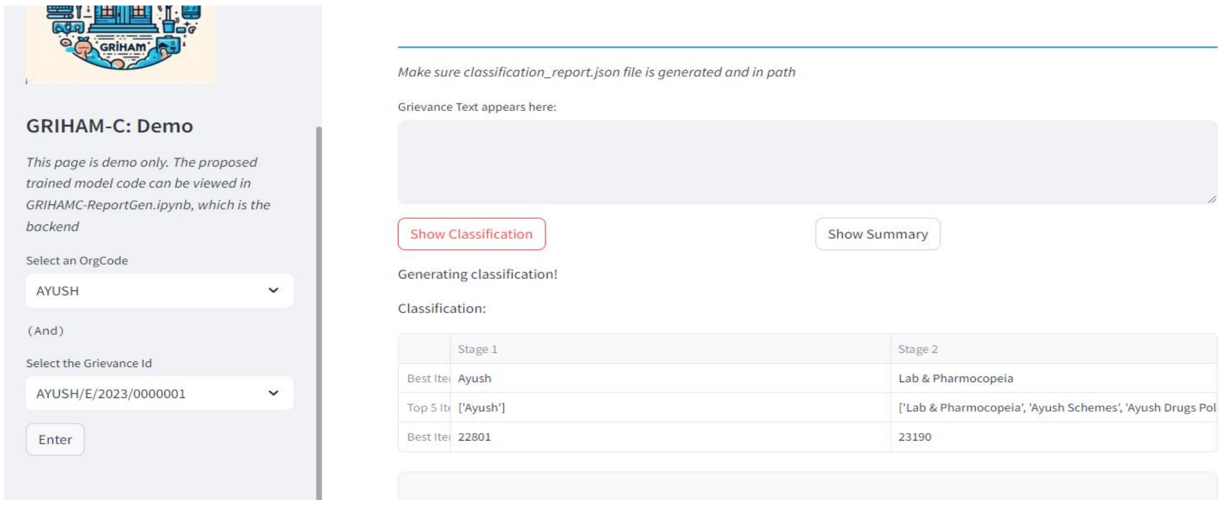
Show Classification | Show Summary

empty

empty

Grievance Summary appears here:

Grievance translation (if any) appears here:



GRIHAM-C: Demo

This page is demo only. The proposed trained model code can be viewed in GRIHAMC-ReportGen.ipynb, which is the backend

Select an OrgCode: AYUSH

(And)

Select the Grievance Id: AYUSH/E/2023/0000001

Enter

Make sure classification_report.json file is generated and in path

Grievance Text appears here:

Show Classification | Show Summary

Generating classification!

Classification:

	Stage 1	Stage 2
Best Ite	Ayush	Lab & Pharmocopeia
Top 5 It	['Ayush']	['Lab & Pharmocopeia', 'Ayush Schemes', 'Ayush Drugs Pol
Best Ite	22801	23190

Implementation and Future Improvements:

- ➢ This system generates the outputs in JSON and CSV files for each Department. Further can be extended for generating reports for each stage by simple selection operations. **The demo of clustering and writing the JSON file, grouping the grievances based on the stages is included in GRIHAMC_Front_end.ipynb, as a simple cell.** (Refer Video).
- ➢ **The report generation system will be setup as a backend and automated** with each grievance submission. The visualizer is just for demo purposes.
- ➢ Right now, the **demo can handle grievances under org_code mentioned in the CategoryCode_Mapping_v2.json**. The code can altered to handle queries not in the file without altering the system.
- ➢ All the dataset used in the development is the **datasets provided** for the competition, **converted as JSON files** for faster access, which are shared in a folder along with the model files.
- ➢ **The training code** used for the same is found in GRIHAMC_TrainingFinal.ipynb.

Improvement areas addressed:

- The model utilizes a language inference model, which is highly flexible for all sorts of text inputs suitable for public use cases, leaving us with the option of fine tuning with a layer on every Indian language, highly effective for easy maintenance.
- Offers intuitive monitoring for the last-mile officers to identify the clusters easily. In built summarizer, translator for the officer to identify the issues and track the grievances. Addition of a chat bot based on a RAG model will enhance the experience.
- Report Generation already clusters the grievances based on the ORG_CODE of the input, which can be extended for generation of report stage wise after classification already present.
- The system can be **integrated with the existing ticket support** to understand the grievances better by using the Top 5 classification list for each Grievance.
- Update, Publish and broadcast options for the last-mile officer enables them to send the message of action in a chain link based on the classification created, thus saving time in the back and forth information passage.

Citations and Resources:

1. Benchmarking Zero-shot Text Classification: Datasets, Evaluation and Entailment Approach https://arxiv.org/abs/1909.00161
2. https://huggingface.co/docs/transformers/training
3. https://arxiv.org/abs/1703.09902v1 - Survey of state of the art NLG technology
4. https://huggingface.co/

Disclaimer: All the development and code were created by the presenter i.e., Meenakshi, and with due compliance with open source licenses (MIT license)