Health Hackers

User Guide

Version 1

Table of Contents

Environment Setup

1. Install the Prerequisites

Name	Version
Python	3.12.5
CUDA	12.4

2. Create a Virtual Environment

On Windows:

1. **Open Command Prompt**:
   Press Win + R, type cmd, and hit Enter.
2. **Navigate to Your Project Folder**:
   Use the cd command to change to your project directory: `cd path\to\your\project`
3. **Create the Virtual Environment**:
   Run the following command: `python -m venv venv_name`
   Replace venv_name with your desired virtual environment name.
4. **Activate the Virtual Environment**: `venv_name\Scripts\activate`
   After activation, you should see (venv_name) at the start of your command prompt, indicating the virtual environment is active.

On Linux:

1. **Open a Terminal**.
2. **Install python3-venv (if not already installed)**: `sudo apt install python3-venv`
3. **Navigate to Your Project Folder**: `cd path/to/your/project`
4. **Create the Virtual Environment**: `python3 -m venv venv_name`
   Replace venv_name with your desired virtual environment name.
5. **Activate the Virtual Environment**: `source venv_name/bin/activate`
   You should see (venv_name) at the start of your terminal prompt, indicating the virtual environment is active.

3. Install the Required Python Packages

On Windows: Once the virtual environment is activated, install the necessary dependencies using a requirements.txt file: `pip install -r requirements-windows.txt`

On Linux: Once the virtual environment is activated, install the necessary dependencies using a requirements.txt file: `pip install -r requirements-linux.txt`

How to Run the Topic Modelling Pipeline Script

Overview

This guide explains how to execute the script for processing text data and performing topic modelling using BERTopic.

Prerequisites

Data File
Place the input CSV file (features_Z140Hep.csv) in the /data/raw/ directory relative to the script's location. Ensure the file has columns named NarrativeLE and NarrativeCME.

Output Directory
Ensure the output directories exist: /data/outputs/topic_modelling/

File Paths

- **Input File:**
  /data/raw/features_Z140Hep.csv

Running the Script

1. **Open Terminal**
   Navigate to the directory containing the script.
2. **Navigate to the Script Directory**
   Ensure you are in the directory where the script is located: `cd path/to/your/src`
3. **Run the Script**
   Execute the script:
   a. On Windows: `py topic_modelling.py`
   b. On Linux: `python3 topic_modelling.py`
4. **Processing**
   The script will:
   - Load the input CSV file.
   - Normalize and preprocess the text data.
   - Generate embeddings using sentence-transformers.
   - Apply BERTopic for topic modelling on the specified columns (NarrativeLE and NarrativeCME).

- Save the topic modelling results as CSV files.

5. **Completion**
   The processed files will be saved in the output directory.

Expected Output

After running the script, you will find:

1. **Topic Modeling Results for NarrativeLE:**
   /data/outputs/topic_modelling/output_with_topics_LE.csv
2. **Topic Modeling Results for NarrativeCME:**
   /data/outputs/topic_modelling/output_with_topics_CME.csv

Troubleshooting

Missing Dependencies:
If the script fails due to missing libraries, ensure they are installed in your virtual environment: `pip list`

File Not Found Errors:
Verify that the input file path and output directories exist as specified.

Spacy Model Not Found:
Ensure the en_core_web_sm model is installed by running: `python -m spacy download en_core_web_sm`

Output Issues:
If the script doesn't generate output files, check for errors in the terminal log or ensure input data contains the required columns.

How to Run the LLM Responses Clustering Pipeline Script

Overview

This documentation provides a detailed guide to configure, execute, and understand the operations performed by the provided scripts, which processes narratives from a dataset using a Hugging Face LLM pipeline.

1. Processes narrative text data using a Hugging Face LLM to identify novel variables.
2. Extracts and cleans these variables.
3. Generates embeddings using Sentence-BERT.
4. Clusters the variables into meaningful groups using various clustering methods.
5. Visualizes and saves the clustered variables.

Prerequisites

1.Data File

Place the input CSV file (features_Z140Hep.csv) in the /data/raw/ directory relative to the script's location. Ensure the file has columns named NarrativeLE and NarrativeCME.

2. Authentication Token

The script requires a Hugging Face token to download the meta model. Replace the placeholder HUGGINGFACE_TOKEN in the script with your token, which can be generated from the Hugging Face account page. Before using the LLaMA 3.2 3B Instruct model via Hugging Face, it is mandatory to review and accept the model's license agreement provided by Meta AI. This ensures compliance with their terms of use, which may include restrictions on redistribution, commercial use, and adherence to ethical guidelines.

3. File Locations

Ensure the input file and output directories exist:

- **Input File**: /data/raw/features_Z140Hep.csv
- **Output Files**:
    - LLM responses: /data/outputs/llm_with_clustering/
    - Extracted variables: /data/outputs/llm_with_clustering/
    - Clusters: /data/outputs/clusters/

Ensure all input directories exist and have the appropriate permissions.

Configuration Parameters

Parameter	Description	Value
HUGGINGFACE_TOKEN	A token for Hugging Face authentication.	Replace with your token.
MODEL_ID	The identifier for the pre-trained LLM to be used.	meta-llama/Llama-3.2-3B-Instruct
INPUT_CSV	Path to the input dataset containing narratives.	../data/raw/features_Z140Hep.csv
OUTPUT_CSV	Path for saving LLM-generated responses.	../data/outputs/llm_with_clustering/llm_responses.csv
EXTRACTED_VARIABLES_FILE	File to save cleaned variable names extracted from LLM responses.	../data/outputs/llm_with_clustering/cleaned_variable_list.txt
QUERY_FOR_LLM	The prompt/query to be used for generating LLM responses.	"What are the novel variables can you understand from the given context?"

MAX_NEW_TOKENS	Maximum tokens generated in LLM responses.	512
ROWS_TO_PROCESS	Number of narrative rows interact with the LLM context.	4000
SENTENCE_TRANSFORMER_MODEL	Sentence-BERT model for embeddings.	all-MiniLM-L6-v2

Steps to Run the Script

1. Activate the environment as before.

2. Workflow Execution

Step 1: Generate LLM Responses

Run the script to process narratives, extract variables, and save results:

On Windows: `py llm_iteratives.py`

On Linux: `python3 llm_iteratives.py`

This script:

1. Reads narratives from /data/raw/features_Z140Hep.csv.
2. Uses an LLM to generate responses based on QUERY_FOR_LLM.
3. Saves responses to /data/outputs/llm_with_clustering/llm_responses.csv.
4. Extracts bolded variables from responses and saves them in /data/outputs/llm_with_clustering/cleaned_variable_list.txt.

Step 2: Clean Variables and Generate Embeddings

Run the clustering script to process extracted variables:

On Windows: `py clustering.py`

On Linux: `python3 clustering.py`

This script:

1. Cleans the extracted variables for consistent formatting.
2. Generates semantic embeddings using Sentence-BERT.

3. Perform Clustering using DBSCAN
4. Visualize and Save Clusters

   The script visualizes the clusters using PCA and saves them:

   - Individual cluster files: /data/outputs/clusters/cluster_<id>.txt
   - Other All noises in one file: /data/outputs/clusters/clusters.txt

Script Workflow

Step 1: Authentication

The script authenticates with Hugging Face using the provided HUGGINGFACE_TOKEN.

Step 2: Load Model and Pipeline

The specified model (MODEL_ID) is loaded along with its tokenizer and configured for text generation.

Step 3: Process Dataset

- Reads the input CSV file.
- Combines the NarrativeLE and NarrativeCME columns for each row.
- Generates responses from the LLM for each narrative using the provided query (QUERY_FOR_LLM).

Step 4: Save Results

The LLM responses are saved in a CSV file at the specified location (OUTPUT_CSV).

Step 5: Extract Variables

- Parses the LLM responses for variables denoted in bold (e.g., **variable**).
- Converts variable names to snake_case.
- Saves the unique, cleaned variable names to the specified file (EXTRACTED_VARIABLES_FILE).

Step 6: Perform Clustering

Choose a clustering method:

- **DBSCAN**: Use a precomputed cosine distance matrix to identify clusters.

Step 7: Visualize and Save Clusters

The script visualizes the clusters using PCA and saves them

Output Files

LLM Responses:

- File: /data/outputs/llm_with_clustering/llm_responses.csv
- Contains responses generated for each narrative.

Extracted Variables:

- File: /data/outputs/llm_with_clustering/cleaned_variable_list.txt
- Contains unique, cleaned variables in snake_case format.

Clustered Variables:

- Individual files: /data/outputs/clusters/cluster_<id>.txt
- Consolidated file: /data/outputs/clusters/clusters.txt

Troubleshooting

Authentication Errors:
Verify the Hugging Face API token and its permissions.

File Not Found:
Check input file paths and directory structures.

Model Loading Issues:
Ensure the MODEL_ID and SENTENCE_TRANSFORMER_MODEL are correctly specified and compatible.

Clustering Issues:
Use the elbow method or k-NN plots to determine optimal clustering parameters.

Performance Issues:
Use a machine with GPU acceleration for processing large datasets or LLMs.

How to Run the Combined Pipeline Script

Overview

This script processes text data, retrieves relevant chunks using FAISS indexing, and generates responses using a language model. Follow the steps below to configure and execute the script

1. Authenticates with Hugging Face using a token.
2. Loads and initializes an LLM (Meta-Llama) and an embedding model (Sentence-BERT).
3. Processes narrative text data into manageable chunks.
4. Builds a FAISS index for efficient similarity-based text retrieval.
5. Retrieves relevant text chunks based on search queries.
6. Generates responses using retrieved text and an LLM.
7. Saves the generated responses to individual files.

Prerequisites

1. Authentication Token

The script requires a Hugging Face token to download the meta model. Replace the placeholder HUGGINGFACE_TOKEN in the script with your token, which can be generated from the Hugging Face account page. Before using the LLaMA 3.2 3B Instruct model via Hugging Face, it is mandatory to review and accept the model's license agreement provided by Meta AI. This ensures compliance with their terms of use, which may include restrictions on redistribution, commercial use, and adherence to ethical guidelines.

2. File Locations

Ensure the input file and output directories exist:

- **Input File**: /data/raw/features_Z140Hep.csv
- **Output File**: /data/outputs/llm_with_faiss/

Ensure all input and output directories exist and have the appropriate permissions.

Configuration Parameters

Parameter	Description	Value
HF_TOKEN	Hugging Face API token for authentication.	Replace with your token.
MODEL_ID	LLM model used for processing narratives.	meta-llama/Llama-3.2-3B-Instruct

INPUT_CSV	Path to the input dataset.	../data/raw/features_Z140Hep.csv
OUTPUT_DIR	Directory to save the generated responses.	../data/outputs/llm_with_faiss
queryForLLM	Query for generating LLM responses.	"What are the novel variables you can understand from the given context?"
queryForSS_list	List of search queries for FAISS retrieval.	(List of pre-defined queries that selected from topic modelling.)

Steps to Run the Workflow

1. Workflow Execution
   To run the FAISS-based LLM script, follow these steps:

   Step 1: Authenticate Hugging Face
   The script automatically authenticates with Hugging Face using the token provided in the HF_TOKEN configuration. Ensure you replace the placeholder token with your actual Hugging Face API token.

   Step 2: Prepare Input Files
   Ensure that the input file (../data/raw/features_Z140Hep.csv) exists and contains the required columns (NarrativeLE and NarrativeCME).

   Step 3: Create the Output Directory
   The script automatically creates the output directory (../data/outputs/llm_with_faiss) if it does not exist. However, ensure the script has write permissions for this directory.

   Step 4: Run the Script
   Run the script using the following command:

   On Windows: `` `py llm_faiss.py` ``

   On Linux: `` `python3 llm_faiss.py` ``

Step 5: Script Execution Workflow

Once executed, the script will:

1. **Authenticate with Hugging Face**: Logs in using the provided HF_TOKEN.
2. **Load LLM and Embedding Models**: Initializes the specified LLM (meta-llama/Llama-3.2-3B-Instruct) and Sentence-BERT (all-MiniLM-L6-v2).
3. **Process the Dataset**:
    - Combines NarrativeLE and NarrativeCME columns for each row.
    - Splits long text into smaller chunks, respecting token limits.
    - Stores these chunks in a separate DataFrame (df_chunks).
4. **Build the FAISS Index**:
    - Computes embeddings for the text chunks using Sentence-BERT.
    - Builds a FAISS index for efficient similarity-based retrieval.
5. **Generate Responses**:
    - For each query in queryForSS_list, retrieves relevant text chunks from the FAISS index.
    - Combines the retrieved text into a single context.
    - Generates a response using the LLM and saves it to a text file in the output directory.

Step 6: Check Output

After the script completes, navigate to the output directory:
/data/outputs/llm_with_faiss
You will find response files named response_<index>.txt (e.g., response_1.txt, response_2.txt).

Troubleshooting

1. **Authentication Issues**:
    - Verify the Hugging Face token and ensure it's active.
2. **Model Loading Errors**:
    - Confirm that the specified MODEL_ID and embedding model ID are correct.
3. **File Not Found**:
    - Ensure the input file path and output directories exist.
4. **Performance Issues**:
    - Use a machine with GPU acceleration for faster processing, especially for FAISS indexing and LLM generation. Try to use faiss-gpu instead of faiss-cpu.
5. **Indexing Errors**:
    - Ensure the df_chunks DataFrame contains valid text data before building the FAISS index.