Project Structure

Directories and Files

- **CODE** - Contains the code files.
- **DATA** - Contains the extracted embeddings, the collected news, and images.
- **REPORT.pdf** - The report explaining the results.

./CODE:

- **analyse.ipynb** - Main notebook for making all the plots and results.
- **Emotion_classification.py** - Classifying emotions in the text.
- **Extract_embeddings.py** - Cleans content and saves the text and cleaned image paths.
- **embed_content.py** - Extract embeddings for the content of the news articles.
- **Joint_new.py** - Extract text, image, and joint embeddings using ImageBind.
- **Sentiment_analysis.py** - Label positive and negative sentiment in text.

./DATA:

- **images** - Holds the representative image of each article.
- **News_embeddings.pkl** - Embeddings of article content.
- **Vision_embeddings.pkl** - Embeddings for the article representative image.
- **news.xlsx** - Collected news data including the source, content, and header.
- **Text_embeddings.pkl** - Embeddings of the news article heading.

./DATA/images:

Images saved as `article source_article number`:

- BBC_1.png
- IndianExpress_1.png
- NYTimes_2.png
- BBC_2.png
- IndianExpress_2.png
- NYTimes_3.png
- BBC_3.png
- NYTimes_1.png

CODE README

Sentiment Analysis

This script, `sentiment_analysis.py`, performs sentiment analysis on a collection of articles. It uses the Hugging Face's Transformers library to classify the sentiment of each sentence in the articles. The final output is a list of scores indicating the proportion of positive and negative sentiments for each article.

How it works:

1. **Loading Data**: The script begins by loading the article contents from a pickle file named `text_content.pkl`.
2. **Sentence Tokenization**: Each article is split into individual sentences using the NLTK library.
3. **Sentiment Analysis**: The Hugging Face's sentiment analysis pipeline is used to classify the sentiment of each sentence.
4. **Aggregation**: The sentiments for each article are aggregated to provide a single score for the entire article.

Dependencies:

- NLTK
- Transformers

Output:

The script outputs a list of dictionaries, where each dictionary represents the sentiment scores for an article. The keys in the dictionary are "POS" and "NEG", representing the proportion of positive and negative sentiments, respectively.

News Content Embeddings

The script, `embed_content.py`, is designed to generate embeddings for a collection of news articles using the BERT model from the Hugging Face's Transformers library. Additionally, it predicts the news category for each article.

Workflow:

1. **Loading Data**: The script loads the article contents from a pickle file named `text_content.pkl`.
2. **BERT Model Setup**: Initializes the BERT model and tokenizer from a pretrained model.
3. **Embedding Generation & Classification**: For each article, the script tokenizes the content, generates embeddings, and predicts the news category.
4. **Saving Embeddings**: The embeddings are saved to a pickle file named `news_embeddings.pkl`.

Dependencies:

- Transformers
- torch
- pickle

Outputs:

- A list of predicted news categories for each article.
- A list of probabilities associated with each prediction.
- A pickle file, `news_embeddings.pkl`, containing the embeddings for each article.

Emotion Classification

The script, `emotion_classification.py`, performs emotion classification on a collection of articles. It leverages the Hugging Face's Transformers library and a pretrained model to classify the emotion of each sentence in the articles. The final output is a list of emotion counts normalized by the number of lines in each article.

Workflow:

1. **Loading Data**: The script loads the article contents from a pickle file named `text_content.pkl`.
2. **Sentence Tokenization**: Each article is split into individual sentences using the NLTK library.
3. **Emotion Classification**: The script uses the `SamLowe/roberta-base-go_emotions` model to classify the emotion of each sentence.
4. **Aggregation**: The emotions for each article are aggregated and normalized by the number of lines to provide a consolidated count for each article.

Dependencies:

- Transformers
- NLTK
- pickle

Output:

The script outputs a list of dictionaries, where each dictionary represents the normalized emotion counts for an article.

Extract Embeddings

The script, `extract_embeddings.py`, is designed to process a collection of news articles and their associated images. It cleans the text content, loads the images, and prepares them for embedding extraction (though the actual embedding extraction is commented out in the provided code).

Workflow:

1. **Loading Data**: The script loads the article contents from an Excel file named `news.xlsx`.
2. **Text Cleaning**: Each article's header and content are tokenized and cleaned.
3. **Image Loading**: The associated images for each article are loaded and processed.
4. **Embedding Extraction**: The script contains commented-out sections that would extract embeddings for the text and images using a model.
5. **Saving Processed Data**: The cleaned text headers, image paths, and main content are saved as pickle files for future use.

Dependencies:

- os
- pickle
- pandas
- torch
- numpy
- PIL
- torchvision
- models (local module)
- nltk

Outputs:

- A pickle file, `text_list.pkl`, containing the cleaned text headers for each article.
- A pickle file, `image_paths.pkl`, containing the paths to the processed images.
- A pickle file, `text_content.pkl`, containing the cleaned main content for each article.

Joint Embeddings Extraction

The script, `joint_new.py`, is designed to extract joint embeddings for a collection of news article headers and their associated images. It uses the `imagebind_model` from a local module to generate embeddings for both text and vision modalities.

Workflow:

1. **Loading Data**: The script loads the cleaned text headers and image paths from pickle files named `text_list.pkl` and `image_paths.pkl`, respectively.
2. **Model Initialization**: Initializes the `imagebind_huge` model from the local `imagebind_model` module.
3. **Data Transformation**: The text and images are transformed into a format suitable for the model.
4. **Embedding Extraction**: The script extracts embeddings for both the text and vision modalities.
5. **Similarity Calculation**: Calculates the cosine similarity between the embeddings.
6. **Saving Results**: The embeddings and similarity results are saved as pickle files.

Dependencies:

- models (local module)
- data (local module)
- torch
- pickle

Outputs:

- A pickle file, `vision_text_similarity.pkl`, containing the similarity scores between vision and text embeddings.
- A pickle file, `vision_embeddings.pkl`, containing the embeddings for the images.
- A pickle file, `text_embeddings.pkl`, containing the embeddings for the text headers.