**Overview**

This documentation covers a LinkedIn Post Generator application that uses few-shot learning and LLM (Language Model) to generate professional LinkedIn posts based on user preferences for topic, length, and language.

**File Structure**

**1. few\_shot.py**

Handles loading and filtering example posts for few-shot learning.

**Key Features:**

* Loads and processes posts from a JSON file
* Categorizes posts by length (Short, Medium, Long)
* Provides filtered posts based on user criteria

**Methods:**

* \_\_init\_\_(file\_path): Initializes and loads posts
* load\_posts(file\_path): Loads and processes JSON data
* categorize\_length(line\_count): Categorizes post length
* get\_tags(): Returns unique tags
* get\_filtered\_posts(length, language, tag): Returns filtered posts

**2. llm\_helper.py**

Handles LLM (Language Model) integration using Groq API.

Key Features:

* Loads environment variables
* Initializes Groq chat model (meta-llama/llama-4-scout-17b-16e-instruct)

**3. main.py**

Streamlit application for the user interface.

**Key Features:**

* Provides dropdown selectors for:
  + Title (tags)
  + Length (Short, Medium, Long)
  + Language (English)
* Generate button to create posts

**4. post\_generator.py**

Generates LinkedIn posts using LLM with few-shot examples.

Key Features:

* Constructs prompts with few-shot examples
* Handles length specifications
* Generates posts using LLM

Methods:

* get\_length\_str(length): Converts length category to line count
* get\_prompt(length, language, tag): Constructs the LLM prompt
* generate\_post(length, language, tag): Generates the final post

**5. preprocess.py**

Processes raw post data into structured format with metadata.

Key Features:

* Extracts metadata (line count, language, tags) from raw posts
* Unifies and merges similar tags
* Saves processed data to JSON

**Methods:**

* process\_post(raw\_file\_path, processed\_file\_path): Main processing function
* get\_unified\_tags(posts\_with\_metadata): Unifies similar tags
* extract\_metadata(post): Extracts metadata from a single post

Workflow

1. **Data Preparation:**
   * Raw posts are processed through preprocess.py to extract metadata and unify tags
   * Processed data is saved to processed\_posts.json
2. **User Interaction**:
   * User selects preferences (tag, length, language) in the Streamlit UI (main.py)
   * On clicking "Generate":
     + Relevant examples are fetched (few\_shot.py)
     + Prompt is constructed (post\_generator.py)
     + LLM generates the post (llm\_helper.py)
3. **Post Generation**:
   * The system uses few-shot learning by including relevant examples in the prompt
   * Generated post is displayed to the user

Dependencies

* Python 3.x
* Required packages:
  + pandas
  + json
  + streamlit
  + langchain\_groq
  + python-dotenv
  + langchain\_core