Documentation

Meme Generator Using Text Similarity

Date: June 19, 2025

Contents

1 Project Overview

1.1 Project Domain/Area

This project falls within the domain of **Natural Language Processing (NLP)**. It leverages unsupervised vectorization and similarity matching techniques to retrieve contextually relevant memes based on user input.

1.2 Project Description

The AI Meme Generator is designed to allow users to enter a punchline or topic and automatically retrieve the most contextually relevant meme from a captioned meme dataset. Given the limitation of freely available template datasets, we used a Kaggle dataset containing 6,992 pre-captioned memes. Users can remove the old captions and insert new ones using built-in editing tools such as white box overlays, cropping, and text input. This provides an interactive way to personalize meme content while using real-world captions for better relevance.

1.3 Project Objectives

The objectives of this project are to implement an efficient meme matching and generation system using TF-IDF and cosine similarity, develop a clean GUI for user interaction, allow comprehensive meme editing options, evaluate model performance using Top-K similarity, and deliver a complete offline desktop solution.

1.4 Target Audience/Users

The project is targeted at content creators, social media users, digital marketers, and general users who wish to generate customized memes quickly and efficiently without needing design experience.

2 Technical Approach

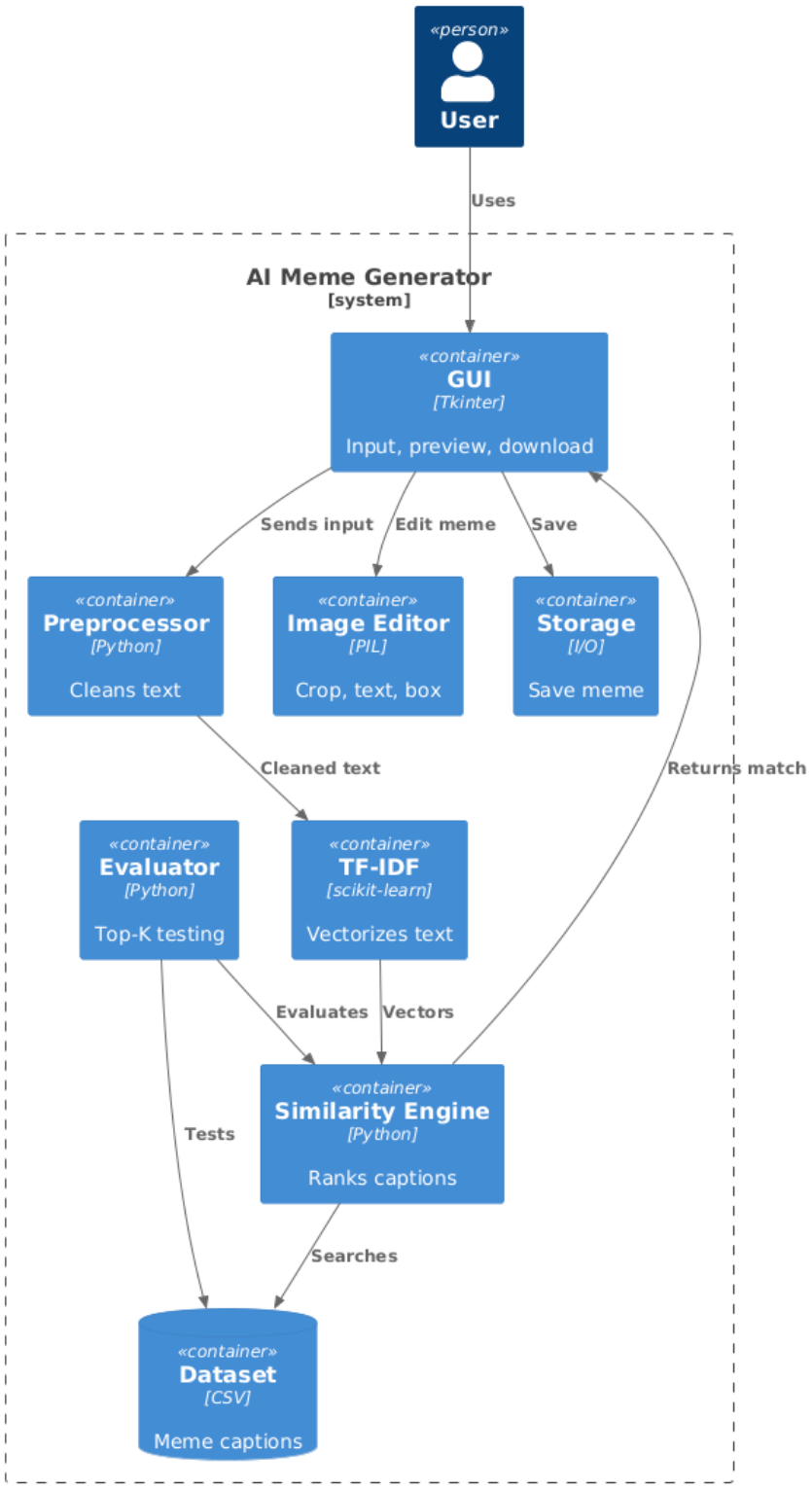
2.1 AI Techniques

We utilize **TF-IDF (Term Frequency-Inverse Document Frequency)** to convert captions and user input into weighted word vectors. These vectors are then compared using **cosine similarity** to determine relevance. The system uses **instance-based lazy retrieval**, meaning it does not involve training in a traditional sense. Instead, similarity computation is performed in real-time based on the input.

2.2 Data Requirements

The dataset used is "6992 Labeled Meme Images" from Kaggle, containing file names and associated meme captions. OCR-corrected fields are also included where available. Due to the unavailability of large-scale meme templates, we repurposed this dataset to suit our needs.

2.3 System Architecture



2.3.1 Architecture Components

The system comprises several modular components. The **Preprocessing Module** handles text normalization, including lowercasing and punctuation removal, using the `enhanced_preprocess` function. The **TF-IDF Vectorizer** converts meme captions and user input into numeric vector representations. The **Similarity Matching Engine** ranks meme captions using

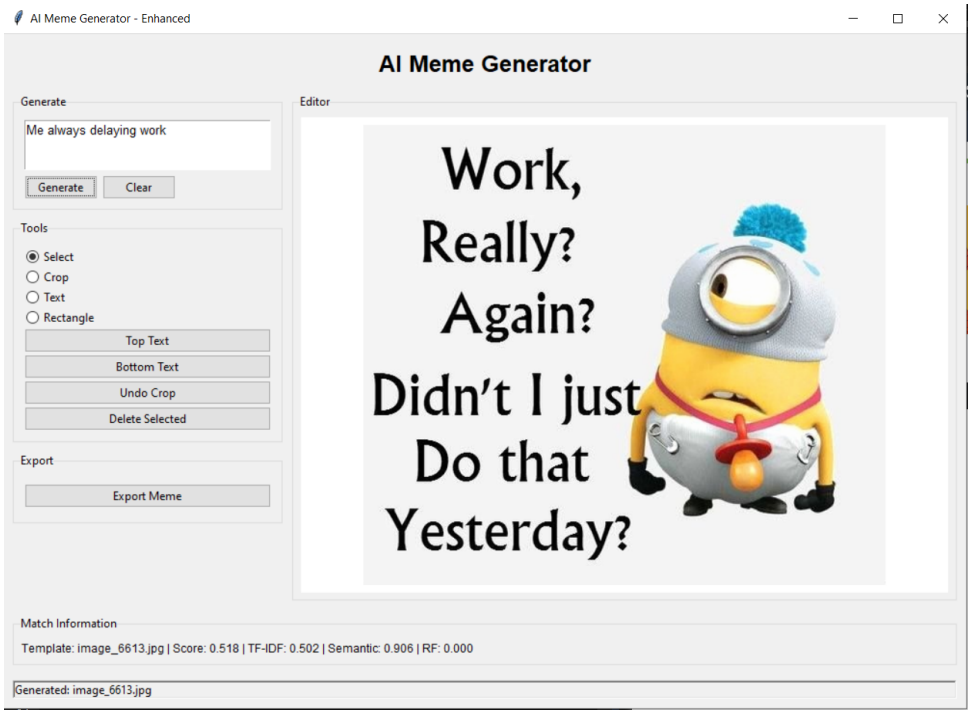
cosine similarity and selects the top results. The **Image Editor** provides cropping, text overlay, and masking features. Finally, the **GUI (Tkinter)** manages all user interactions and displays output results.

2.4 GUI Interface Design

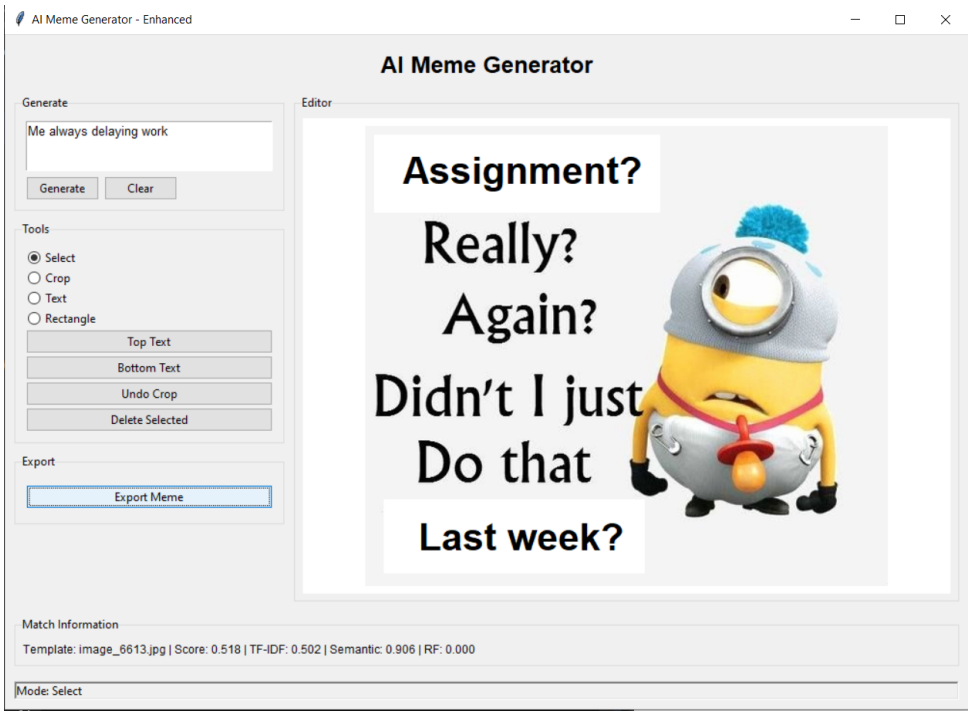
The GUI was built using **Tkinter** and provides text input, meme generation, image editing tools (crop, white block, text), and an export/download feature. It ensures an intuitive flow from input to final meme creation and enables offline usage.

2.5 Visual Mockups

2.5.1 Screen 1: Input Screen



2.5.2 Screen 2: Meme Preview/Edit Screen



3 Evaluation Methodology

3.1 Why Traditional Accuracy is Not Used

Since our system is not a classifier, traditional metrics like precision, recall, F1-score, and categorical accuracy are irrelevant. We do not use labels or train a prediction model. Instead, we focus on similarity-based ranking of text inputs.

3.2 Top-K Similarity Accuracy

Metric Used: Top-5 similarity accuracy
Definition: Measures how often the correct meme image appears in the top 5 most similar matches for a given test caption.

3.3 Testing Process

The dataset is split into 80% for fitting the vectorizer and 20% for testing. Each test caption is vectorized, and cosine similarity is calculated with all stored meme vectors. The top-K matches are ranked, and if the correct meme is present in those K results, it is counted as correct. Accuracy is calculated as:

$$accuracy = \left(\frac{correct\_matches}{total\_test\_samples} \right) \times 100$$

3.4 Accuracy Result

Our model achieved **99.21% Top-5 similarity accuracy**. This indicates that in over 99% of test cases, the correct meme appeared among the top five suggested results. This high score demonstrates the effectiveness of vector-based textual matching.

4 Testing and Limitations

4.1 Testing

Testing was conducted using unseen captions from the dataset. Evaluation was carried out using Top-K similarity scoring logic to ensure ranking relevance and model consistency.

4.2 Limitations

Despite strong performance, the system has a few limitations. The dataset contains pre-captioned memes, not clean templates. All matching is purely text-based—image content is not analyzed. Additionally, very short user input may result in sparse vectors, reducing matching accuracy.

5 Justification of Model Choice

We opted for TF-IDF and cosine similarity due to their simplicity, efficiency, and appropriateness for our dataset. Supervised models like Random Forests or Naive Bayes were unsuitable due to class uniqueness. Deep learning models like BERT or GPT were avoided to reduce complexity, computational demands, and data needs. Instead, we used instance-based lazy retrieval with unsupervised text vectorization, which aligns perfectly with the project's scope and available resources.

6 Resources and Tools

This project was developed using Python 3. Libraries include Tkinter for GUI development, Pillow for image editing, and scikit-learn for vectorization and similarity matching. Pandas and NumPy were used for data handling. The dataset used was the 6,992 labeled meme images dataset from Kaggle.