AIMS-DTU Research Intern Round 2: From Picture to Plate

Overview

This project, *From Picture to Plate*, focuses on leveraging Vision-Language Models (VLMs) to generate concise 2–3 step cooking instructions from food images and noisy or vague dish titles. The task emphasizes creativity in data curation, prompt engineering, and understanding the alignment between visual and textual data to produce practical recipe summaries.

Objective

The goal is to create a pipeline that takes a food image and a noisy or vague dish title as input and generates a concise 2–3 step cooking instruction that captures the essence of the dish. The pipeline should generalize to unseen image-title pairs using a pretrained VLM and few-shot prompting.

Dataset Setup

The dataset was curated from Food.com, a popular recipe website, to ensure diversity in dishes and reliable recipe data. The process involved:

- **Data Collection**: Scraped 15 food image samples along with their corresponding noisy titles (e.g., ”cheesy bake”, ”noodly thing”) and full cooking instructions from Food.com.
- **Manual Summary Creation**: For each of the 15 samples, concise 2–3 step cooking summaries were manually written based on the full instructions. These were stored as tuples in a Python dictionary and exported to a CSV file (`y_train.csv`) using the pandas library.
- **Test Dataset**: A separate test set of 10 image-title pairs was created similarly, with manually written summaries stored in `y_test.csv` for evaluation.
- **Image Handling**: Images were downloaded and stored locally, with paths updated in `x_train.csv` and `x_test.csv` to point to the correct file locations (e.g., `1.png`, `2.png`, etc.).

The training dataset included dishes like spaghetti bolognese and creamy Cajun chicken, with summaries formatted as three-step instructions. For example:

- **Title**: Spaghetti Bolognese
- **Summary**: Step 1: Brown hamburger and onion. Step 2: Drain grease, add sauce ingredients, and simmer. Step 3: Serve over cooked pasta with Parmesan.

Task Flow

Step 1: Data Collection & Manual Summary Creation

The dataset was collected from Food.com, focusing on recipes with clear images and detailed instructions. For each of the 15 training samples, full instructions were condensed into 2–3 step summaries, ensuring clarity and brevity. These summaries served as ground truth for few-shot prompting. The test set of 10 samples was prepared similarly to evaluate model generalization.

Step 2: Model Selection & Pipeline Design

Initially, a Facebook VLM was tested but proved unsuitable, as it generated irrelevant outputs not aligned with the task of summarizing recipes. After researching suitable models, Google's FLAN-T5-Large was selected due to its strong performance in text generation and ability to handle structured prompts effectively. The pipeline was designed as follows:

- **Tokenizer and Model**: Used `AutoTokenizer` and `AutoModelForSeq2SeqLM` from HuggingFace's Transformers library to process inputs and generate outputs.
- **Prompt Engineering**: A few-shot prompting approach was adopted, where three randomly selected training examples (title-summary pairs) were included in the prompt to guide the model. The prompt format was:

  > Generate a concise 2–3 step recipe for the given dish title.
  > Title: [Example Title 1]
  > Recipe: [Example Summary 1]
  > ...
  > Title: [Input Title]
  > Recipe:

- **Inference Settings**: The model used temperature (0.7), top-k sampling (50), and top-p sampling (0.95) to balance creativity and coherence in the generated summaries.
- **Image Verification**: Images were loaded using the PIL library to confirm accessibility and correct format (RGB), though the current pipeline focused on text-based inputs (titles) for simplicity.

The pipeline was implemented in Python 3.10, utilizing libraries like `transformers`, `torch`, `pandas`, `tqdm`, and `zipfile`. A GPU (NVIDIA GeForce RTX 3050 with CUDA 12.4) was used to accelerate inference.

Step 3: Inference on Test Set

The pipeline was tested on the 10 test samples, generating summaries for each noisy title. The outputs were saved to `flan_t5_test_results.csv`. Example outputs include:

- **Title**: Lemon Baked Cod
- **Generated Summary**: Step 1: Preheat oven, season cod. Step 2: Bake with lemon. Step 3: Serve hot.
- **Title**: Creamy Cajun Chicken
- **Generated Summary**: Step 1: Sauté chicken. Step 2: Add spices and cream. Step 3: Simmer and serve.

Step 4: Evaluation and Reflection

The generated summaries were manually compared to the original test summaries in `y_test.csv`. Key observations include:

- **Successes**: The model correctly captured the essence of dishes in most cases, producing concise and actionable instructions. For example, the summary for ”Lemon Baked Cod” accurately reflected the baking process and lemon flavor.
- **Failures**: In some cases, the model oversimplified complex dishes or missed specific ingredients (e.g., omitting garnishes like parsley), likely due to the limited number of few-shot examples. A significant challenge was integrating the PC’s GPU (NVIDIA GeForce RTX 3050 with CUDA 12.4). Without proper GPU configuration, the model would freeze during inference, causing the VS Code editor to crash. This was overcome by ensuring compatibility with PyTorch and CUDA, enabling stable and faster processing.
- **Common Failure Patterns**: The model occasionally produced generic steps (e.g., ”cook and serve”) when the noisy title was too vague, underscoring the need for descriptive prompts. Early attempts with the BLIP model failed, as it merely echoed the few-shot prompt examples without generating meaningful summaries, indicating it was not suited for this task. Similarly, the `facebook/bart-large-cnn` model produced irrelevant outputs, such as repeating the prompt itself (e.g., for ”Lemon Baked Cod,” it generated: ”You are a professional cooking assistant. Given the name of a dish, generate a short, clear 2–3 step cooking recipe...”), confirming its inadequacy as a summarizer for this context.
- **Model Choice Justification**: After extensive research across platforms to identify a VLM capable of capturing recipe complexity and generalizing

well in the local environment, Google's FLAN-T5-Large was selected. Unlike the BLIP model, which failed to generate meaningful summaries, and the `facebook/bart-large-cnn` model, which produced irrelevant outputs, FLAN-T5-Large demonstrated robust text generation and effective generalization from few-shot examples without requiring fine-tuning. Its compatibility with the GPU setup and ability to handle structured prompts made it ideal for this task.

- **Design Decisions**: The decision to focus on text-based inputs (titles) rather than images simplified the pipeline, as image processing required additional preprocessing. Few-shot prompting was effective given the small dataset size, avoiding the need for resource-intensive fine-tuning.

Deliverables

1. **Notebook/Script**: A Python script was developed to load data, process images, generate summaries using FLAN-T5, and save results. The script includes clear console logging for debugging and progress tracking.
2. **Manually Written Summaries**: 15 training and 10 test summaries were created and stored in `y_train.csv` and `y_test.csv`, respectively.
3. **Model-Generated Outputs**: Summaries for the test set were saved in `flan_t5_test_re`
4. **Video Walkthrough**: A 3–5 minute video will be prepared to explain the pipeline, prompt structure, model selection reasoning, and two sample outputs (e.g., Lemon Baked Cod and Creamy Cajun Chicken).

Conclusion

The FLAN-T5-Large model, combined with few-shot prompting, successfully generated concise and practical cooking instructions from noisy dish titles. The pipeline demonstrated the potential of pretrained VLMs for recipe summarization, even without fine-tuning. Future improvements could include incorporating image data for multimodal processing or expanding the dataset for better generalization.

Divansu Mishra

24/A03/021