MLLM-Tool: A Multimodal Large Language Model For Tool Agent Learning

Supplementary Material

In this supplementary material, we list i) dataset details, including selection criteria, searching for multimodal data, prompt templates, data distribution, and crowdsourcing details; ii) experiment setup, data preprocessing; iii) additional experimental results; iv) visualization of the model card, instruction-answer pairs and the actual cases of when using MLLM-Tool; v) our proposed categorization system.

1. Dataset Details

In this section, we elaborate on the dataset construction details and the training and testing data distribution. Precisely, Figure 1.1 visualizes the entire dataset construction process, while Sec. 1.2 - Sec. 1.4 detail each step, including API selection criteria, API functional boundaries criteria, and prompt construction. Sec. 1.5 visualizes some data distributions of the training set and testing set. Sec. 1.6 describes the process of employing annotators and instructing them on some guidelines, and in Sec. 1.7, we explain the details when searching for content-matched non-text data. We emphasize that the collection and processing of this dataset involved over 2000 person-hours, ensuring its reliability for advanced computer vision research and applications by meticulous attention to detail.

1.1. Visualization of Dataset Construction Process

We show the whole process of collecting our dataset in Figure 1. The difficulty in collecting the benchmark is reflected in three aspects. (1) Task Classification: We restructure a hierarchical task classification system. This advancement rectifies the overly coarse-grained categorizations observed in the HuggingFace platform, ensuring each API has a specific and unambiguous task label. (2) Annotation of Multimodal Input Instructions: Should an instruction reveal cues from modalities other than text, a precise coherence between the divulged information in the instruction and the content of the non-textual modality is paramount. In cases where no such cues are disclosed, we deem it essential to investigate five potential cases caused by textual ambiguity. Moreover, we show the visualization of five ambiguity types in Figure 2. Such scrutiny lays the groundwork for subsequent experiments aimed at validating whether information from other modalities aids large language models in making appropriate tool selections; (3) Instruction Matching: To identify all APIs meeting an instruction's functional requirements, we must categorize APIs based on identical and similar functionalities respectively during collection. When pairing instructions for groups of APIs with analogous functions, it becomes crucial to discern whether a provided instruction strictly satisfies each API's functional description. This determination will influence the number of APIs that can aptly respond as the correct answer to a given instruction.

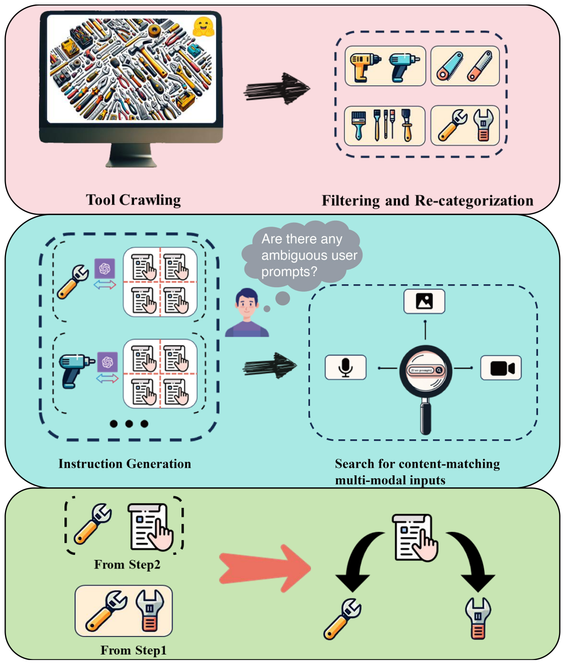

Tool Crawling
Filtering and Re-categorization
Are there any ambiguous user prompts?
Instruction Generation
Search for content-matching multi-modal inputs
From Step2
From Step1

Figure 1. The process of dataset construction. It mainly consists of three stages. The first step is collection and preprocessing, including crawling the original API from HuggingFace and filtering and re-categorizing the API according to tasks. The second step is the generation of instruction-answer pairs, including using GPT-4 to generate instructions based on API functions, determining whether there is text ambiguity in the user prompts, and searching for content-matching input for multi-modal. The third step is constructing a potential one-to-many instruction-answer pair based on the relationship between API functions.

1.2. Filtering Rules

After crawling APIs from HuggingFace, we implement stringent selection criteria for APIs due to the varying quality of these models. The specific filtering rules for this selection process are outlined as follows:

Low-Quality Model Card. Many APIs either lack model cards or contain overly simplistic ones, providing little helpful information. Additionally, some APIs have ceased maintenance or have been integrated into other APIs. Even multiple APIs share a single model card. These APIs pose challenges in organizing API documentation and require manual consultation of additional sources like papers and GitHub. While enriching the number of APIs, this approach introduces inaccuracies. Finally, we opted to remove such APIs from our dataset uniformly.

NSFW Content Risk. For ethical and safety considerations, we remove APIs, particularly some text-to-image types, which pose risks of generating NSFW (Not Safe For Work) content. Such content is deemed inappropriate for all age groups and could compromise the dataset's applicability in diverse settings.

License Restrictions. We exclude models with restrictive licensing terms. An example is the Llama2 model, which requires a formal application and approval process to download and use its model weights.

Identical APIs. Numerous models on HuggingFace share identical structures and datasets, merely differing in implementation by various contributors. To avoid redundancy, we selectively retain only the most downloaded version of each such model, ensuring the dataset is more streamlined. For APIs with different model architectures but trained on the same dataset and having the same functions, as we mentioned in the main text, we select the top five APIs based on download statistics.

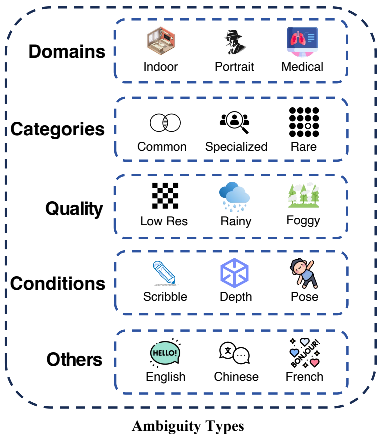


Figure 2. Five types of ambiguity cases need to be checked during the instruction-answer pair generation step.

1.3. API Function Boundaries

Multi-task APIs. Our dataset includes multi-task APIs like various large language models (Bloom [2], Baichuan2 [3], Falcon [1]), capable of numerous NLP tasks, some of which even fall outside HuggingFace's classification system. For simplicity, we focus specifically on their text generation capabilities, aligning with HuggingFace's functional categorization of such APIs. Additionally, we modify the descriptions of these APIs in our dataset to ensure a match between the functionalities reflected by the provided queries and those described in the API descriptions.

API Function Differentiation. After identifying the specific subdomain of each API, we detail our approach for discerning whether APIs within the same subdomain possess identical or similar functions. For plain text input APIs, we can assess APIs based on the language and context of the datasets they utilize. For example, in sentiment analysis tasks, we discern various contexts by relying on the distinct domains of data sources, such as financial, legal, social media, and reviews, enabling us to classify nuanced functional differences among similar-function APIs. Similarly, we refine API function categorization for multimodal inputs, especially images, by analyzing dataset contexts. For instance, in Visual Question Answering tasks, datasets feature a range of images such as Diagrams, Charts, Documents, Infographics, and other general images, enabling nuanced API segmentation. The above examples provide merely a subset of our API classification rules. Due to the complex and case-by-case nature of the analysis, the complete set of rules is not listed here.

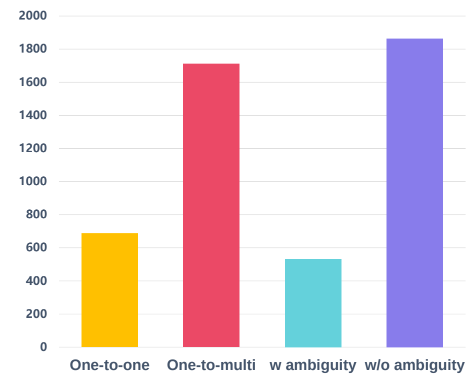
One-to-many Situation Identification. Our approach, driven by GPT-4, originates from each API's functional description and generates corresponding instructions that adhere strictly to the API's functional boundaries. Hence, consideration is primarily given to other APIs with the same function in one-to-many scenarios.

1.4. Prompt Construction

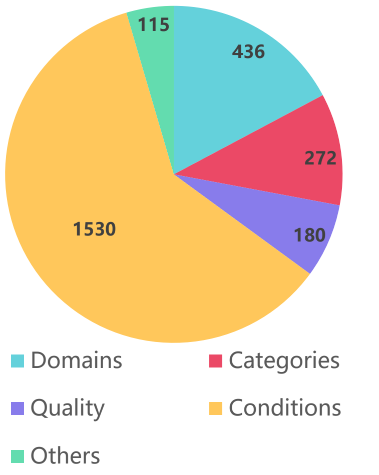
Below, we list the detailed prompts used for generating instructions. The prompts fed to GPT-4 are composed of (1)the API call's relevant information, including its name, a concise functional description, and training datasets if mentioned in the model card, (2)some sample instructions, and (3)additional requirements. The sample instructions are leveraged to foster format standardization and enhance the quality of GPT-4's outputs. So, we invite some expert annotators to craft high-quality instructions in advance and randomly choose two each time as exemplary inputs to GPT-4. We design two sets of prompt templates in our dataset, mindful of prompt ambiguity cases. For APIs with multimodal inputs susceptible to prompt ambiguity, we strive to balance the final selection of ten instructions, maintaining a 1:1 ratio between unambiguous and ambiguous instructions; the corresponding prompt template is shown in Figure 4 and Figure 5 respectively. We emphasize that query ambiguity pairs are pairs with identical textual inputs but varied multimodal inputs.

1.5. Visualization of Data Distributions

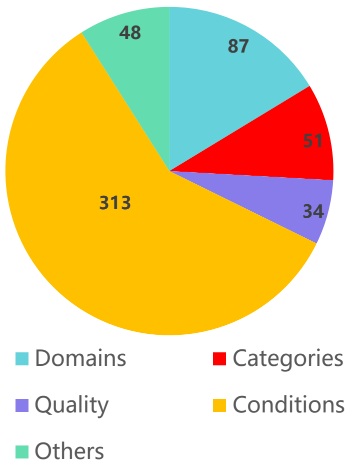
We tally the instances of each ambiguity type in Figure 3b, amassing a total of 2533 ambiguity pairs. Predominantly, these are concentrated within conditional cases involving image inputs, comprising 1530 pairs, since our dataset covers 11 distinct condition types of ControlNet [4], and their various combinations, achieving a considerable



(a) The distribution of sub-testing sets divided by API option number and ambiguity types respectively.



(b) The distribution of each ambiguity type in the whole dataset.



(c) The distribution of each ambiguity type in the testing set.

Figure 3. **Dataset statistic visualization.**

quantity.

Particularly for the testing set, we depict in Figure 3a and Figure 3c the distribution of API quantities across each testing subset. These subsets are constructed based on three distinct partitioning criteria: ambiguity types, API option number, and modality. It is worth noting that the *without ambiguity* category consists of two parts: cases with purely textual inputs that inherently lack ambiguity (1,452 instances) and cases with non-textual inputs where ambiguity is absent (412 instances).

In our work, we collect 932 APIs and one *"Unknown"* category, featuring 651 APIs with pure text inputs, 191 APIs incorporating image inputs, 80 incorporating audio inputs, and 10 incorporating video inputs. Given the long-tail distribution of API quantities for various tasks on HuggingFace, coupled with our selection criteria, the APIs we ultimately choose exhibit a similar long-tail distribution across both modality and task level. In future work, we aim to enrich our collection with more APIs, particularly those with video inputs, and optimize the distribution across different tasks for more uniformity.

1.6. Crowdsourcing Details

In this study, we employ eight annotators with NLP or CV backgrounds, with an estimated hourly 7.10$ compensation and an up to 20% bonus for those with high annotation quality. We provide a comprehensive training document and record a video to guide them through the process of annotation. We also instruct them to avoid using private information or license-risky data from unknown or sensitive sources. All annotators are presented with a consent form and must agree to join the project. We set up a cross-checking mechanism to correct errors. Moreover, our team members conduct manual checking and optimization. If the data is collected from a website, we restore the website links and upload the metadata when releasing the dataset. We claim we only use the data for academic research instead of commercial usage.

1.7. Collection of text-matched multimodal input

Our non-text data sources are from either the dataset API used or Google Images. We prioritize collecting from open-source datasets by manually checking that the data content is consistent with the instructions generated by GPT. For the latter, we can retrieve the data by keywords in the Google database, and for these data, we pay more attention to its license to ensure its legality.

Configuration	7B-model	13B-model
Optimizer	AdamW	AdamW
Optimizer Momentum	$\beta_1 = 0.9, \beta_2 = 0.95$	$\beta_1 = 0.9, \beta_2 = 0.95$
Peak learning rate	5e-4	5e-4
Weight decay	0.001	0.001
Warmup steps	10	10
Batch size	64	32
Micro-batch size	4	1
Gradient accumulation steps	4	8
Maximum target length	512	512
ImageBind Checkpoint	ImageBind-Huge	ImageBind-Huge
LoRA attention dimension (r)	32	32
LoRA scaling alpha (α)	32	32
LoRA drop out	0.1	0.1

Table 1. 7B and 13B model training configuration.

2. Experiment Setup and Data Processing

We list the setup, including some hyperparameters used to train our model, in Table 1. We adopt ImageBind's data processing approach for the multimodal inputs. We evenly split the video into five clips and randomly sampled two frames from each clip to cover the entire length, similar to other Vision Transformer (ViT)-based works with video input. For audio, we sample each input audio at 16KHz. Subsequently, we capture a log mel spectrogram featuring 128 frequency bins. Then, the spectrogram can be seen as a 2D image. In this way, we can use ViT for these three multimodal encoders. A modality-specific linear projection head is added to each modality's encoder to align the feature into a uniform 1024-sized dimension.

3. Additional Experimental Results

3.1. Results of Multiple Options Categorized by Input Modality

Regarding the condition of whether multiple API options exist, we further refine the testing subsets—originally divided based on the presence or absence of multiple API options—by introducing an additional criterion based on input modality to gain deeper insights into the issue. We select two versions of the Vicuna model with different parameter sizes, and the corresponding performance is presented in Table 2. From the results, we observe that for the case where multiple API options exist, the performance is consistently better than the case with only one API option, regardless of the input modality. This validates the argument presented in the main text. On one hand, the presence of multiple API options reduces the problem's complexity and increases the likelihood of selecting the correct API. On the other hand, during training, the data involving multiple API options benefits from being split into multiple one-to-one training strategies, further improving performance.

4. Dataset Visualization

4.1. Model Card Visualization

We illustrate 6 model card samples in Figure 6-11, which covers all the information mentioned in the main text. We list some samples from several different modalities and different functions.

4.2. Instruction-Answer Pairs Visualization

In Figure 12-14, we offer 7 cases sampled from the testing set. When we train our model, we input the conversation between humans and GPT in a conversation format, and the other attributes help us divide them into different sub-testing sets.

Model	API Options	Input Modality	Acc
Vicuna-7B	one-to-one	Video	55.56
Vicuna-7B	one-to-many	Video	83.33
Vicuna-7B	one-to-one	Audio	78.57
Vicuna-7B	one-to-many	Audio	96.32
Vicuna-7B	one-to-one	Image	75.82
Vicuna-7B	one-to-many	Image	96.30
Vicuna-7B	one-to-one	Text	58.24
Vicuna-7B	one-to-many	Text	86.43
Vicuna-13B	one-to-one	Video	44.44
Vicuna-13B	one-to-many	Video	91.67
Vicuna-13B	one-to-one	Audio	75.00
Vicuna-13B	one-to-many	Audio	95.59
Vicuna-13B	one-to-one	Image	86.81
Vicuna-13B	one-to-many	Image	96.10
Vicuna-13B	one-to-one	Text	70.21
Vicuna-13B	one-to-many	Text	90.24

Table 2. The performance of the testing subsets based on the joint criterion of the presence of multiple API options and the input modality.

4.3. Agent Output Visualization

We show four actual cases using our MLLM-Tool in Figure 15-18. Our system supports inputs from four modalities and leverages the predicted API to execute, finally getting customers' expected results.

5. Categorization System

We illustrate our proposed categorization system in Table 3,4,5. The system is two-level hierarchical. The coarse-level task setting mainly refers to the classification of HuggingFace, while the fine-grained subtask is based on the different task descriptions.

### User: You are an NLP task expert. Given an API, you need to generate 20 different user queries that utilize the API function, adhering to the following input and output format for each query:
Input:
API Name: This is the name of the API Function.
Description: This is a detailed description of the model.
Prohibit Words: These are words that are prohibited from appearing in the output.

Output:
Query1: This is an instruction that can utilize the API function.
Query2: This is an instruction that can utilize the API function.
Query3: This is an instruction that can utilize the API function.
Query4: This is an instruction that can utilize the API function.
Query5: This is an instruction that can utilize the API function.
. . .
Query20: This is an instruction that can utilize the API function.

Below are some examples:
Example 1:
Input:
API Name: microsoft/codereviewer
Description: CodeReviewer is a model pre-trained with code change and code review data to support code review tasks.
Prohibit Words: ”API, tools, model”

Output:
Query1: I’ve made some changes to my Python code. Can you review this snippet for me? ”def addNums(a, b): return a + b”
Query2: I’m not very confident in my Java coding skills. Could you check this piece for potential issues? ”public int divide(int x, int y) { return x / y; }”
Query3: I’m trying to refactor this C# method. Does it look okay? ”public void PrintName(string name) { Console.WriteLine(name); }”
Query4: I wrote this SQL query, and I’m unsure if it’s optimized. Can you review it? ”SELECT * FROM users WHERE age $\geq$ 25;”
Query5: Here’s a piece of JavaScript function I came up with. Any suggestions for improvement? ”function greet(name) { return ’Hello ’ + name; }”
Query6: I’ve been learning Ruby recently. Would appreciate your thoughts on this code segment: ”def multiply(x, y) x * y end”
Query7: I’m not sure if this CSS is correct. Can you review it? ”h1 { color: blue; font-size: 30px; }”
Query8: I just started with Swift. Here’s a function I wrote. Could you review it for best practices? ”func displayMessage(message: String) { print(message) }”
Query9: My colleague wrote this PHP function, but it seems off. Can you review it for me? ”function subtract(x,y) { return $x-$y; }”
. . .
Query20: I’m learning Kotlin and wrote this simple method. Mind taking a look? ”fun sum(a: Int, b: Int): Int = a + b”

To be continued

Continued
Example 2:
Input:
API Name: Salesforce/blip-vqa-base
Description: This model is a base-sized ViT-based BLIP trained on the visual question answering task VQA2.0 dataset. The model's input is an image and a text question, and the model's output is a text answer.
Prohibit Words: "API, tools, model, VQA2.0"
Output:
Query1: I recently visited a new city and took some photographs. Based on this particular image, can you tell me what is the weather conditions like?
Query2: I was at a photo exhibition and took this picture. Could you provide information about was this taken in a Latin American country judging from the dress?
Query3: I was at the park and saw a dog playing. By analyzing this photo, can you tell if the frisbee it is using is hard or soft?
Query4: My cousin went to a new food joint and ordered their specialty hot dog. By looking at this picture, can you determine if the hot dog is larger than a normal one?
Query5: I attended a family gathering and took a group photo. By analyzing the photograph, how many kids can you see in the picture?
Query6: Looking at this captivating portrait, I noticed an object in the individual's hand. It seems to be a sweet treat. What is this person holding?
Query7: I was at the beach and took a photo of a man skiing. By examining the image, can you tell if the wave is chasing him?
Query8: I visited the local train station and captured an image of a passing train. Can you identify what color the train is from the picture?
Query9: This cityscape presents an old-fashioned vehicle that stands out amidst the modern structures. I'd like to know more about this vehicle. Could you tell me its model and color?"
. . .
Query20: Given the serene backdrop of sailboats and a calm sea, can you tell me how many boats are there?

Note that:
1. When crafting queries, avoid including the API's name;
2. Ensure that the queries are varied and diverse;
3. When processing an input, any words listed in the Prohibited Words must be strictly excluded from the response.

Now, let's start.
Input:
API Name: [API Name]
Description: [API Description]
Prohibit Words: [Some words]

Figure 4. **Example of the unambiguous prompt template**: This template applies to pure text inputs and a part of multimodal input scenarios where textual inputs reveal multimodal information. In practical usage, the template's masked sections should be populated with the respective API's name, description, and words prohibited in the instructions.

### User: You are an NLP task expert. Given an API, you need to generate 20 different user queries that utilize the API function, adhering to the following input and output format for each query:
Input:
API Name: This is the name of the API Function.
Description: This is a detailed description of the model.
Prohibit Words: These are words that are prohibited from appearing in the output.

Output:
Query1: This is an instruction that can utilize the API function.
Query2: This is an instruction that can utilize the API function.
Query3: This is an instruction that can utilize the API function.
Query4: This is an instruction that can utilize the API function.
Query5: This is an instruction that can utilize the API function.
. . .
Query20: This is an instruction that can utilize the API function.

Below are some examples:
Example 1:
Input:
API Name: timm/resnet101.a1h_in1k
Description: A 101 layers ResNet-B image classification model trained on ImageNet-1k.
Prohibit Words: "API, tools, model, ImageNet"

Output:
Query1: I'd value your input on this image's classification.
Query2: There's an image in my possession, and I'm seeking a label for it.
Query3: I need help in determining the label for this image. Can you help?
Query4: Can you provide a label perspective for this image?
Query5: How would you manage the categorization of this particular visual?
Query6: There's an intriguing image that I've come across, and I'm curious about its category.
Query7: How would you classify this picture?
Query8: I'd be grateful for your view on this image's category.
Query9: I'm searching for a label for this visual. Can you assist?
. . .
Query20: What category seems plausible for this visual, in your view?

Example 2:
Input:
API Name: lllyasviel/control_v11f1p_sd15_depth
Description: This model is intended for control diffusion models by adding extra conditions. Trained with depth estimation, the condition image is an image with depth information, usually represented as a grayscale image, and the output is a new image.
Prohibit Words: "API, tools, model, depth, controlnet"

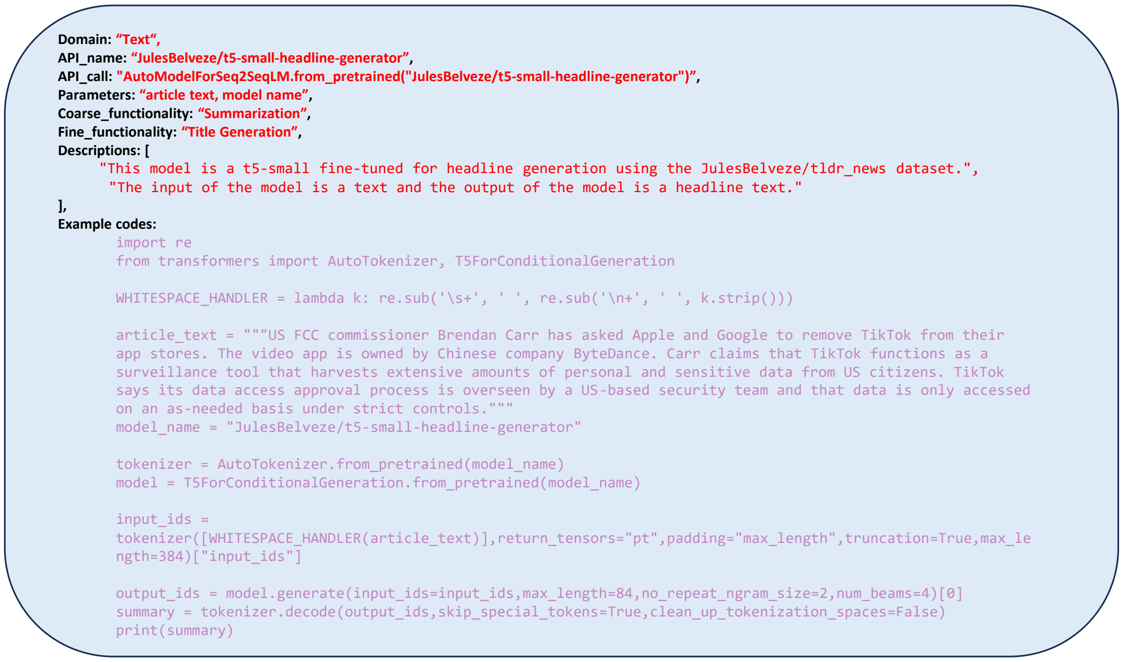
To be continued

Continued
Output:
Query1: Use my image to depict a family picnic in a sunlit park with children playing.
Query2: Can you reimagine my image as a soccer match in progress, with players and fans cheering?
Query3: Create an adventure scene with pirates and treasure islands using my image.
Query4: Can you make my image into a heartwarming scene of a mother duck leading her ducklings through a pond?
Query5: Craft a depiction of a magician performing tricks in front of an amazed audience from my image.
Query6: Could you reshape this photograph into a bustling train station with intricate ironwork and arches?
Query7: Transform this snapshot into a charming cottage nestled among rolling hills and wildflowers.
Query8: I'm curious about how this picture would be reimagined as a bustling harbor with ships, cranes, and cargo containers.
Query9: Use my image to depict a family picnic in a sunlit park with children playing.
...
Query20: I'd love to see this image as a gardener carefully pruning roses in a blooming garden.

Note that:
1. When crafting queries, avoid including the API's name;
2. Ensure that the queries are varied and diverse;
3. When processing an input, any words listed in the Prohibited Words must be strictly excluded from the response;
4. The queries should not convey or imply multimodal information.

Now, let's start.
Input:
API Name: [API Name]
Description: [API Description]
Prohibit Words: [Some words]

Figure 5. **Example of the ambiguous prompt template**: This template applies to other multimodal input scenarios where textual inputs do not reveal multimodal information. In practical usage, the template's masked sections should be populated with the respective API's name, description, and words prohibited in the instructions.



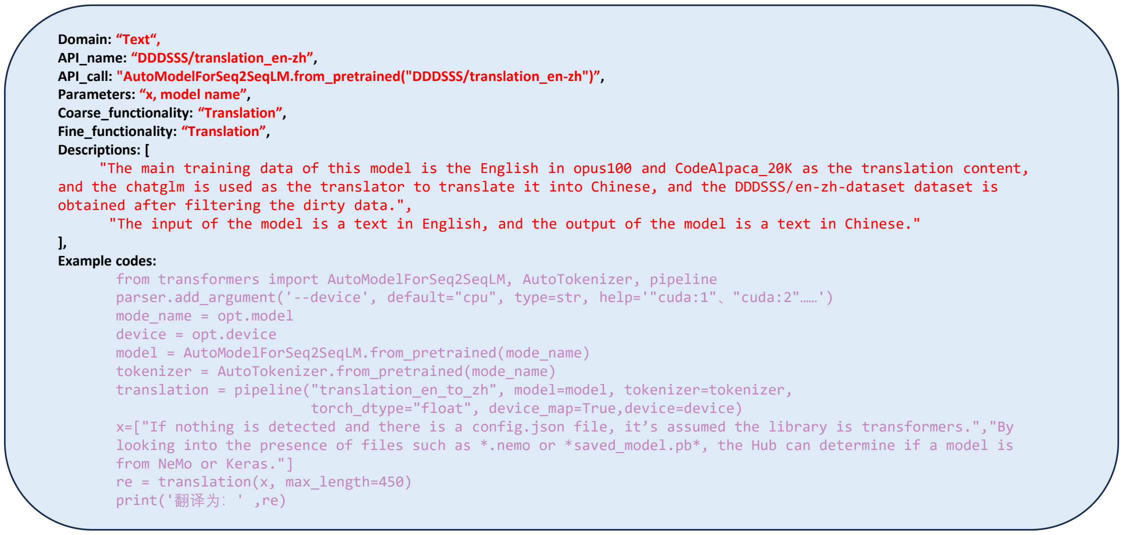
Domain: “Text“,
API_name: “JulesBelveze/t5-small-headline-generator”,
API_call: "AutoModelForSeq2SeqLM.from_pretrained("JulesBelveze/t5-small-headline-generator")”,
Parameters: “article text, model name”,
Coarse_functionality: “Summarization”,
Fine_functionality: “Title Generation”,
Descriptions: [

```
     "This model is a t5-small fine-tuned for headline generation using the JulesBelveze/tldr_news dataset.",
      "The input of the model is a text and the output of the model is a headline text."
```

],
Example codes:

```
import re
from transformers import AutoTokenizer, T5ForConditionalGeneration

WHITESPACE_HANDLER = lambda k: re.sub('\s+', ' ', re.sub('\n+', ' ', k.strip()))

article_text = """US FCC commissioner Brendan Carr has asked Apple and Google to remove TikTok from their
app stores. The video app is owned by Chinese company ByteDance. Carr claims that TikTok functions as a
surveillance tool that harvests extensive amounts of personal and sensitive data from US citizens. TikTok
says its data access approval process is overseen by a US-based security team and that data is only accessed
on an as-needed basis under strict controls."""
model_name = "JulesBelveze/t5-small-headline-generator"

tokenizer = AutoTokenizer.from_pretrained(model_name)
model = T5ForConditionalGeneration.from_pretrained(model_name)

input_ids =
tokenizer([WHITESPACE_HANDLER(article_text)],return_tensors="pt",padding="max_length",truncation=True,max_le
ngth=384)["input_ids"]

output_ids = model.generate(input_ids=input_ids,max_length=84,no_repeat_ngram_size=2,num_beams=4)[0]
summary = tokenizer.decode(output_ids,skip_special_tokens=True,clean_up_tokenization_spaces=False)
print(summary)
```

Figure 6. Model Card Visualization. (Sample 1)



Domain: “Text“,
API_name: “DDDSSS/translation_en-zh”,
API_call: "AutoModelForSeq2SeqLM.from_pretrained("DDDSSS/translation_en-zh")”,
Parameters: “x, model name”,
Coarse_functionality: “Translation”,
Fine_functionality: “Translation”,
Descriptions: [

```
     "The main training data of this model is the English in opus100 and CodeAlpaca_20K as the translation content,
and the chatglm is used as the translator to translate it into Chinese, and the DDDSSS/en-zh-dataset dataset is
obtained after filtering the dirty data.",
      "The input of the model is a text in English, and the output of the model is a text in Chinese."
```

],
Example codes:

```
from transformers import AutoModelForSeq2SeqLM, AutoTokenizer, pipeline
parser.add_argument('--device', default="cpu", type=str, help='"cuda:1"、"cuda:2"……')
mode_name = opt.model
device = opt.device
model = AutoModelForSeq2SeqLM.from_pretrained(mode_name)
tokenizer = AutoTokenizer.from_pretrained(mode_name)
translation = pipeline("translation_en_to_zh", model=model, tokenizer=tokenizer,
                       torch_dtype="float", device_map=True,device=device)
x=["If nothing is detected and there is a config.json file, it’s assumed the library is transformers.","By
looking into the presence of files such as *.nemo or *saved_model.pb*, the Hub can determine if a model is
from NeMo or Keras."]
re = translation(x, max_length=450)
print('翻译为: ' ,re)
```

Figure 7. Model Card Visualization. (Sample 2)

Domain: “Audio“,
API_name: “microsoft/speecht5_vc”,
API_call: "SpeechT5ForSpeechToSpeech.from_pretrained("microsoft/speecht5_vc")”,
Parameters: “path”,
Coarse_functionality: “Audio-to-Audio”,
Fine_functionality: “Voice Conversation”,
Descriptions: [

```
    "This is a SpeechT5 model fine-tuned for voice conversion (speech-to-speech) on CMU ARCTIC. This model is for
self-supervised speech/text representation learning, motivated by the success of T5",
     "The input of the model is an audio, and the output of the model is an audio.",
     "The model is trained on CMU ARCTIC dataset."
```

],
Example codes:

```
from transformers import SpeechT5Processor, SpeechT5ForSpeechToSpeech, SpeechT5HifiGan
from datasets import load_dataset
dataset = load_dataset("hf-internal-testing/librispeech_asr_demo", "clean", split="validation")
dataset = dataset.sort("id")
sampling_rate = dataset.features["audio"].sampling_rate
example_speech = dataset[0]["audio"]["array"]
processor = SpeechT5Processor.from_pretrained("microsoft/speecht5_vc")
model = SpeechT5ForSpeechToSpeech.from_pretrained("microsoft/speecht5_vc")
vocoder = SpeechT5HifiGan.from_pretrained("microsoft/speecht5_hifigan")

inputs = processor(audio=example_speech, sampling_rate=sampling_rate, return_tensors="pt")
# load xvector containing speaker's voice characteristics from a file
import numpy as np
import torch
speaker_embeddings = np.load("xvector_speaker_embedding.npy")
speaker_embeddings = torch.tensor(speaker_embeddings).unsqueeze(0)

speech = model.generate_speech(inputs["input_values"], speaker_embeddings, vocoder=vocoder)

import soundfile as sf
sf.write("speech.wav", speech.numpy(), samplerate=16000)
```

Figure 8. Model Card Visualization. (Sample 3)

Domain: “ Audio“,
API_name: “facebook/mms-lid-126”,
API_call: "AutoModelForAudioClassification.from_pretrained("facebook/mms-lid-126")”
Parameters: “model id, path, language”,
Coarse_functionality: “Audio Classification”,
Fine_functionality: “Spoken Language Identification”,
Descriptions: [

```
    "This model is fine-tuned for speech language identification (LID) and part of Facebook's Massive Multilingual
Speech project. It is based on the Wav2Vec2 architecture and can recognize 126 languages from  facebook/mms-1b
dataset.",
     "The input of the model is an audio, and the output of the model is a predefined language label.",
```

],
Example codes:

```
from transformers import Wav2Vec2ForSequenceClassification, AutoFeatureExtractor
import torch

stream_data = load_dataset(path, language, split="test", streaming=True)
stream_data = stream_data.cast_column("audio", Audio(sampling_rate=16000))
sample = next(iter(stream_data))["audio"]["array"]

model_id = "facebook/mms-lid-126"

processor = AutoFeatureExtractor.from_pretrained(model_id)
model = Wav2Vec2ForSequenceClassification.from_pretrained(model_id)
inputs = processor(sample, sampling_rate=16_000, return_tensors="pt")

with torch.no_grad():
    outputs = model(**inputs).logits

lang_id = torch.argmax(outputs, dim=-1)[0].item()
detected_lang = model.config.id2label[lang_id]
```

Figure 9. Model Card Visualization. (Sample 4)

Domain: “Audio“,
API_name: “Neleac/timesformer-gpt2-video-captioning”,
API_call: "VisionEncoderDecoderModel.from_pretrained("Neleac/timesformer-gpt2-video-captioning")”,
Parameters: “video path”,
Coarse_functionality: “Video-to-Text”,
Fine_functionality: “Video Caption”,
Descriptions: [
    "This is a model for video caption task, whose vision encoder model is timesformer-base-finetuned-k600 and the text decoder model is gpt2",
    "The input of the model is a video, and the output of the model is the caption text of the video.",
    "The model is trained on VaTeX dataset."
],
Example codes:

```
import av
import numpy as np
import torch
from transformers import AutoImageProcessor, AutoTokenizer, VisionEncoderDecoderModel
device = "cuda" if torch.cuda.is_available() else "cpu"
image_processor = AutoImageProcessor.from_pretrained("MCG-NJU/videomae-base")
tokenizer = AutoTokenizer.from_pretrained("gpt2")
model = VisionEncoderDecoderModel.from_pretrained("Neleac/timesformer-gpt2-video-captioning").to(device)
video_path = "never_gonna_give_you_up.mp4"
container = av.open(video_path)
seg_len = container.streams.video[0].frames
clip_len = model.config.encoder.num_frames
indices = set(np.linspace(0, seg_len, num=clip_len, endpoint=False).astype(np.int64))
frames = []
container.seek(0)
for i, frame in enumerate(container.decode(video=0)):
    if i in indices:
       frames.append(frame.to_ndarray(format="rgb24"))
gen_kwargs = {"min_length": 10, "max_length": 20, "num_beams": 8}
pixel_values = image_processor(frames, return_tensors="pt").pixel_values.to(device)
tokens = model.generate(pixel_values, **gen_kwargs)
caption = tokenizer.batch_decode(tokens, skip_special_tokens=True)[0]
print(caption)
```

Figure 10. Model Card Visualization. (Sample 5)

Domain: “Image, Text“,
API_name: “lllyasviel/sd-controlnet-hed”,
API_call: "ControlNetModel.from_pretrained("lllyasviel/sd-controlnet-hed")”,
Parameters: “image path, text”,
Coarse_functionality: “Multimodal-to-Image”,
Fine_functionality: “HED-Boundary-Conditioned Prompt-guided Image Generation”,
Descriptions: [
    "ControlNet is a neural network structure to control diffusion models by adding extra conditions with Stable Diffusion v1-5. Trained with HED edge detection (soft edge), the condition image is a monochrome image with white soft edges on a black background. The HED Edge model was trained on 3M edge-image, caption pairs.",
    "The input of the model is an HED boundary image and a text prompt, and the output of the model is an image"
],
Example codes:

```
from PIL import Image
from diffusers import StableDiffusionControlNetPipeline, ControlNetModel, UniPCMultistepScheduler
import torch
from diffusers.utils import load_image

image = load_image("https://huggingface.co/lllyasviel/sd-controlnet-hed/resolve/main/images/man.png")

controlnet = ControlNetModel.from_pretrained(
    "lllyasviel/sd-controlnet-hed", torch_dtype=torch.float16
)

pipe = StableDiffusionControlNetPipeline.from_pretrained(
    "runwayml/stable-diffusion-v1-5", controlnet=controlnet, safety_checker=None, torch_dtype=torch.float16
)
pipe.scheduler = UniPCMultistepScheduler.from_config(pipe.scheduler.config)
pipe.enable_xformers_memory_efficient_attention()
pipe.enable_model_cpu_offload()

image = pipe("oil painting of handsome old man, masterpiece", image, num_inference_steps=20).images[0]
image.save('images/man_hed_out.png')
```

Figure 11. Model Card Visualization. (Sample 6)

ID: 19
Image Path:
Video Path: video/Ob-VYqPMlxM.mp4
Audio Path:
Input Modality: video
Ambiguity: -1
IsAmbiguity: false
OnetoMulti: false
Conversations:
    From: human
    Value: "Change the breed of the animal in this video.",

    From: gpt
    Value: “unknown"

ID: 18
Image Path:
Video Path: video/81-iBUya_iY.mp4
Audio Path:
Input Modality: video
Ambiguity: -1
IsAmbiguity: false
OnetoMulti: false
Conversations:
    From: human
    Value: "Provide a description for this video I record in a lecture.",

    From: gpt
    Value:
        -“kpyu/video-blip-flan-t5-xl-ego4d"

ID: 1017
Image Path:
Video Path:
Audio Path:
Input Modality: text
Ambiguity: -1
IsAmbiguity: false
OnetoMulti: true
Conversations:
    From: human
    Value: “I need to write a simple C++ program that prints my name. Here‘s what I have: ’#include <iostream> int main() ‘std::cout << ’My name‘; return 0;’‘. Is it correct?“

    From: gpt
    Value:
        -"Salesforce/codet5p-220m",
        -"Salesforce/codet5-small",

Figure 12. Instruction-Answer Pairs Visualization. (Sample 1-3)

```
ID: 886
Image Path:
Video Path:
Audio Path:
Input Modality: text
Ambiguity: -1
IsAmbiguity: false
OnetoMulti: true
Conversations:
    From: human
    Value: “Designing wedding invitations with a twist of enchantment, could you illustrate a scene where 'Love Transcends Reality' is a key theme?"

    From: gpt
    Value:
      -"SG161222/Realistic_Vision_V1.4",
      -"darkstorm2150/Protogen_x3.4_Official_Release",
      -"dreamlike-art/dreamlike-photoreal-2.0",
      -"stablediffusionapi/edge-of-realism",
      -"SG161222/Realistic_Vision_V2.0"
```

```
ID: 731
Image Path:
Video Path:
Audio Path:
Input Modality: text
Ambiguity: -1
IsAmbiguity: false
OnetoMulti: false
Conversations:
    From: human
    Value: “I was reading this article about the stock market: 'Stock markets worldwide took a hit today due to increasing tensions  between the U.S.
      and China. Recent trade talks between the two superpowers seem to have broken down, leading to fears of an impending trade war. Many
      leading economists are predicting a downturn in global growth if the situation isn't resolved quickly. Major companies such as Apple and
      Amazon saw their share prices drop significantly, with tech stocks being the hardest hit. Investors are now seeking refuge in traditional safe
      havens such as gold and government bonds.' Can you help me extract the essential points and turn it into a concise paragraph? "

    From: gpt
    Value:
      -“human-centered-summarization/financial-summarization-pegasus"
```

Figure 13. Instruction-Answer Pairs Visualization. (Sample 4-5)

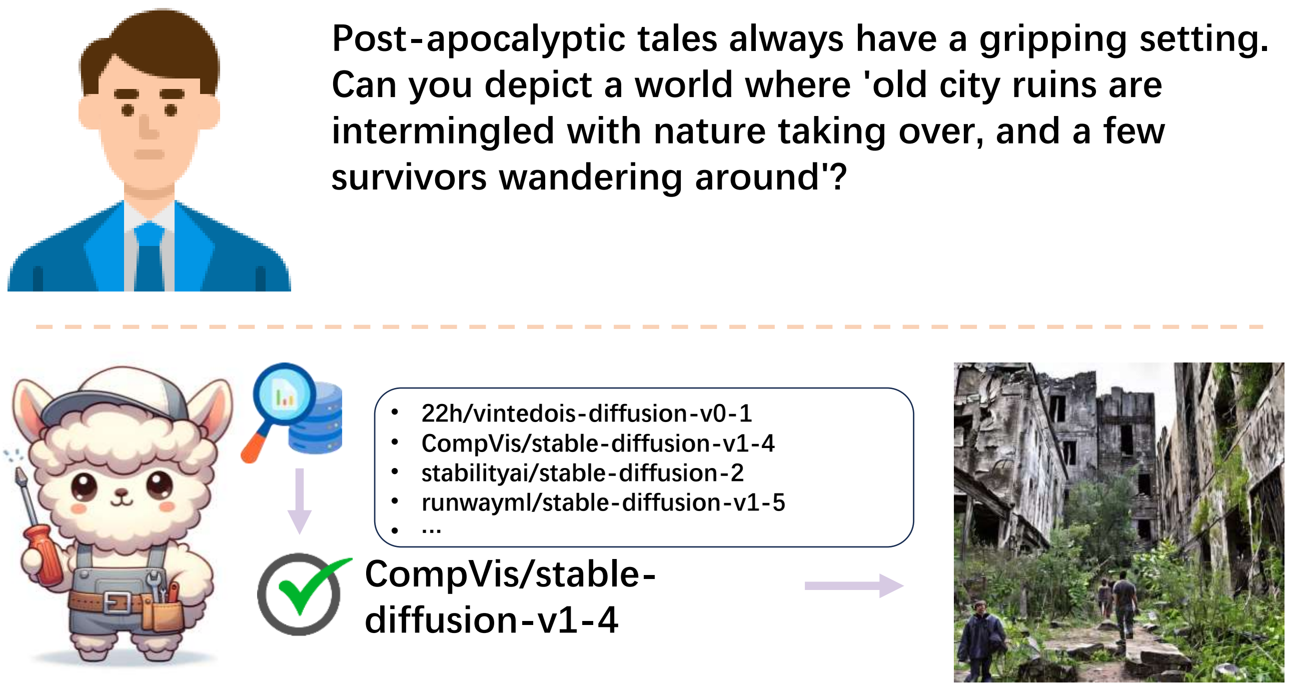
ID: 238
Image Path:
Video Path:
Audio Path: 10/127.jpg
Input Modality: Image
Ambiguity: 4
IsAmbiguity: true
OnetoMulti: true
Conversations:
From: human
Value: “Alter this picture to display a floating wooden houseboat anchored in a calm bay during sunset. “

From: gpt
Value:
-"lllyasviel/control_v11f1p_sd15_depth",
-"lllyasviel/sd-controlnet-depth",
-"thibaud/controlnet-sd21-depth-diffusers",
-"stabilityai/stable-diffusion-2-depth",
-"kandinsky-community/kandinsky-2-2-controlnet-depth"

ID: 147
Image Path:
Video Path:
Audio Path: audioset1/assets/audio11.wav
Input Modality: audio
Ambiguity: 5
IsAmbiguity: true
OnetoMulti: true
Conversations:
From: human
Value: “I have an old recording that I'd like to convert into a readable format. Can you transcribe it for me? “

From: gpt
Value:
-"facebook/mms-1b-all",
-"openai/whisper-large-v2",
-"facebook/wav2vec2-lv-60-espeak-cv-ft",
-"facebook/wav2vec2-xlsr-53-espeak-cv-ft",
-“cpierse/wav2vec2-large-xlsr-53-esperanto“,
-“bond005/wav2vec2-large-ru-golos-with-lm"

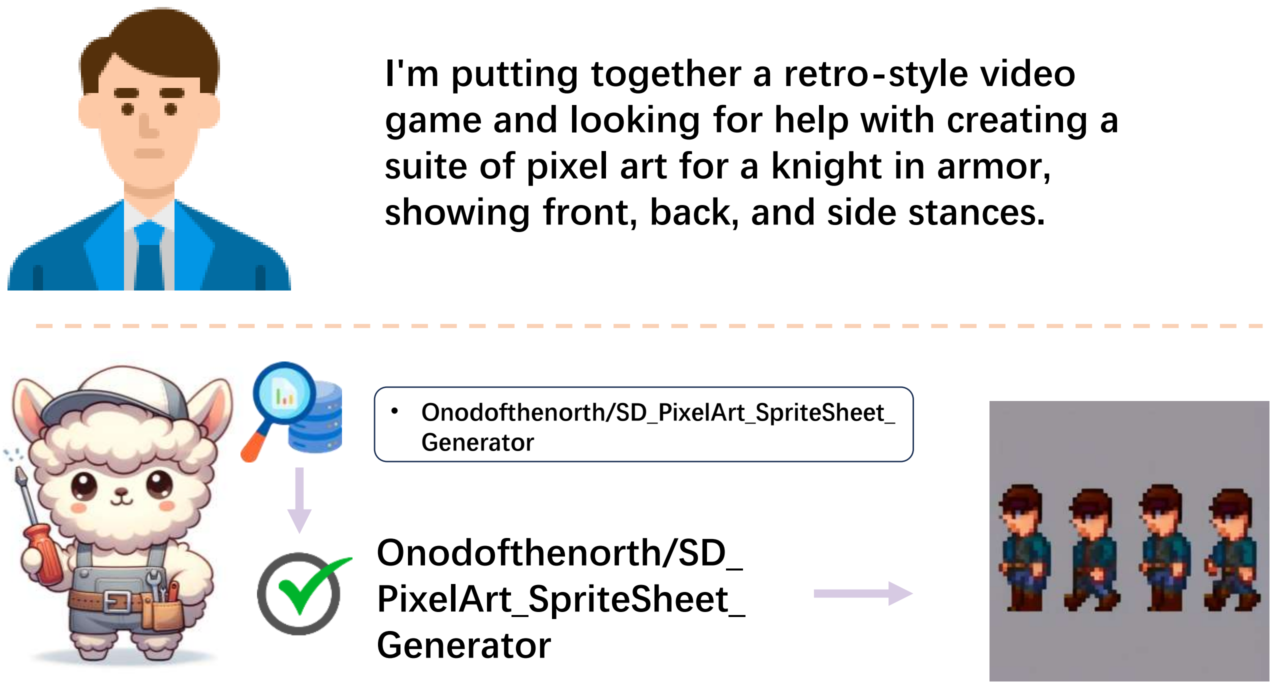
Figure 14. Instruction-Answer Pairs Visualization. (Sample 6-7)


Post-apocalyptic tales always have a gripping setting. Can you depict a world where 'old city ruins are intermingled with nature taking over, and a few survivors wandering around'?

- 22h/vintedois-diffusion-v0-1
- CompVis/stable-diffusion-v1-4
- stabilityai/stable-diffusion-2
- runwayml/stable-diffusion-v1-5
- ...

CompVis/stable-diffusion-v1-4

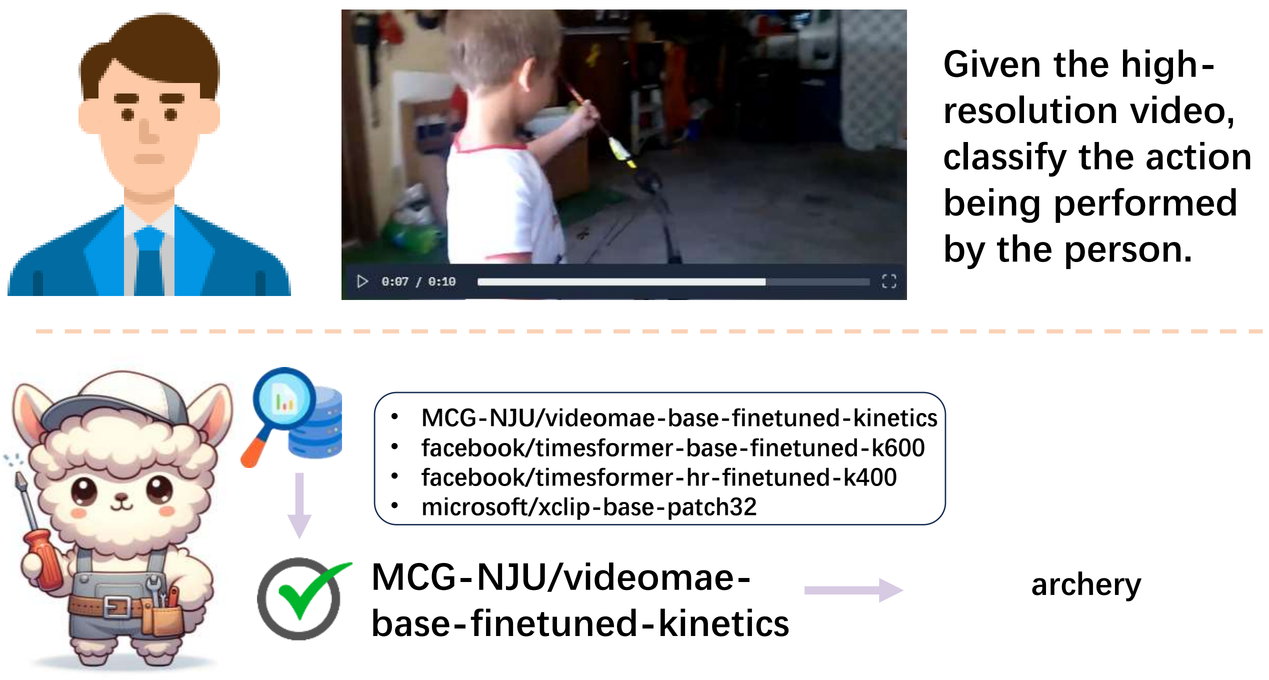
Figure 15. Agent Output Visualization. (Sample 1)


I'm putting together a retro-style video game and looking for help with creating a suite of pixel art for a knight in armor, showing front, back, and side stances.

- Onodofthenorth/SD_PixelArt_SpriteSheet_Generator

Onodofthenorth/SD_PixelArt_SpriteSheet_Generator

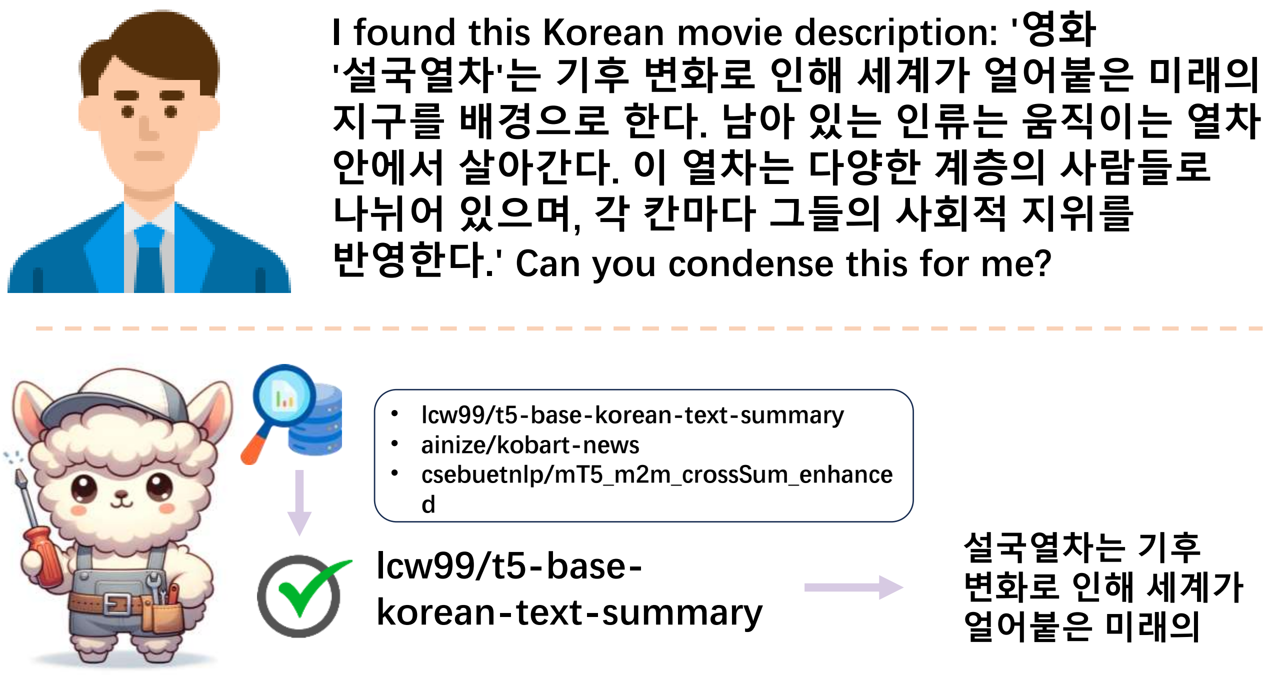
Figure 16. Agent Output Visualization. (Sample 2)



Given the high-resolution video, classify the action being performed by the person.

0:07 / 0:10
MCG-NJU/videomae-base-finetuned-kinetics
facebook/timesformer-base-finetuned-k600
facebook/timesformer-hr-finetuned-k400
microsoft/xclip-base-patch32
MCG-NJU/videomae-base-finetuned-kinetics
archery

Figure 17. Agent Output Visualization. (Sample 3)



I found this Korean movie description: '영화 '설국열차'는 기후 변화로 인해 세계가 얼어붙은 미래의 지구를 배경으로 한다. 남아 있는 인류는 움직이는 열차 안에서 살아간다. 이 열차는 다양한 계층의 사람들로 나뉘어 있으며, 각 칸마다 그들의 사회적 지위를 반영한다.' Can you condense this for me?

lcw99/t5-base-korean-text-summary
ainize/kobart-news
csebuetnlp/mT5_m2m_crossSum_enhanced
lcw99/t5-base-korean-text-summary

설국열차는 기후 변화로 인해 세계가 얼어붙은 미래의

Figure 18. Agent Output Visualization. (Sample 4)

Task	Subtask	Api Names
Audio Classification	Event Recognition	MIT/ast-finetuned-audioset-10-10-0.4593
	Command Recognition	MIT/ast-finetuned-speech-commands-v2
	Spoken Language Identification	TalTechNLP/voxlingua107-epaca-tdnn
	Speaker Verification	anton-l/wav2vec2-base-superb-sv
	Emotion Recognition	audeering/wav2vec2-large-robust-12-ft-emotion-msp-dim
	Gender Recognition	m3hrdadfi/hubert-base-persian-speech-gender-recognition
	Keyword Spotting	superb/hubert-base-superb-ks
Audio-to-Audio	Single-Channel Speech Enhancement	JorisCos/ConvTasNet_Libri1Mix_enhsingle_16k
	Speech-to-Speech Translation	facebook/xm_transformer_unity_en-hk
	Voice Conversion	microsoft/speecht5_vc
	Seperate Clean	mpariente/DPRNNTasNet-ks2_WHAM_sepclean
	Audio Source Separation	speechbrain/sepformer-libri3mix
	Pop-to-Piano	sweetcocoa/pop2piano
Feature Extraction	Audio Feature Extraction	LeBenchmark/wav2vec2-FR-7K-large
Audio-to-Text	Automatic Speech Recognition	Harveenchadha/vakyansh-wav2vec2-hindi-him-4200
Voice Activity Detection	Speaker Segmentation	philschmid/pyannote-segmentation
	Speaker Diarization	philschmid/pyannote-speaker-diarization-endpoint
Depth Estimation	Depth Estimation	Intel/dpt-hybrid-midas
Image Classification	General Image Classification	facebook/convnextv2-tiny-1k-224
	Specific Type Image Classification	Neto71/sea_mammals
	Style Classifier	playrobin/furniture-styles
Image Segmentation	Text Guided Image Segmentation	CIDAS/clipseg-rd64-refined
	Semantic Segmentation	apple/deeplabv3-mobilevit-small
	Panoptic Segmentation	facebook/detr-resnet-50-panoptic
	Instance Segmentation	facebook/mask2former-swin-small-coco-instance
	Zero-shot Segmentation	facebook/sam-vit-huge
Image-to-Image	Image Super-Resolution	caidas/swin2SR-classical-sr-x2-64
	Image Variations	lambdalabs/sd-image-variations-diffusers
	2D-to-3D	openai/shap-e-img2img
	Image Deblurring	google/maxim-s3-deblurring-gopro
	Image Dehazing	google/maxim-s2-dehazing-sots-outdoor
	Image Deraining	google/maxim-s2-deraining-rain13k
Feaure Extraction	Image Feature Extraction	BridgeTower/bridgetower-base
Object Detection	Object Detection	SenseTime/deformable-detr
	Table Detection	TahaDouaji/detr-doc-table-detection
	Text-conditioned Object Detection	google/owlvit-base-patch32
Visual Question Answering	Visual Question Answering	Salesforce/blip-vqa-base
	Chart Question Answering	google/matcha-chartqa
	Diagram Question Answering	google/pix2struct-ai2d-base
	Document Question Answering	google/pix2struct-docvqa-base
	Infographics Question Answering	google/pix2struct-infographics-vqa-large

Table 3. Our proposed categorization system (Part 1). The first and second columns are the two levels of the categories for each task. The third column provides an example of the candidate models.

Task	Subtask	Api Names
Image-to-Text	Conditional Image Caption	Salesforce/blip2-flan-t5-xl
	Optical Character Recognition	alibaba-damo/mgp-str-base
	Image Caption	bipin/image-caption-generator
	Document Parsing	naver-clova-ix/donut-base-finetuned-cord-v2
	Tag Generation	SmilingWolf/wd-v1-4-convnextv2-tagger-v2
	Chart-to-Table	google/deplot
Zero-Shot Image Classification	Zero-Shot Image Classification	laion/CLIP-ViT-B-16-laion2B-s34B-b88K
Multimodal-to-Image	Face-Detection-Conditioned Prompt-guided Image Generation	CrucibleAI/ControlNetMediaPipeFace
	Prompt-guided Cartoonization	instruction-tuning-sd/cartoonizer
	Brightness-Conditioned Prompt-guided Image Generation	ioclab/control_v1p_sd15_brightness
	Prompt-guided Pixel-to-Pixel Image Editing	lllyasviel/control_v11e_sd15_ip2p
	Shuffle-Conditioned Prompt-guided Image Generation	lllyasviel/control_v11e_sd15_shuffle
	Tiled-Conditioned Prompt-guided Image Generation	lllyasviel/control_v11f1e_sd15_tile
	Depth-Conditioned Prompt-guided Image Generation	lllyasviel/control_v11f1p_sd15_depth
	Canny-Edges-Conditioned Prompt-guided Image Generation	lllyasviel/control_v11p_sd15_canny
	Line-Art-Conditioned Prompt-guided Image Generation	lllyasviel/control_v11p_sd15_lineart
	MLSD-Conditioned Prompt-guided Image Generation	lllyasviel/control_v11p_sd15_mlsd
	Normal-Conditioned Prompt-guided Image Generation	lllyasviel/control_v11p_sd15_normalbae
	Openpose-Conditioned Prompt-guided Image Generation	lllyasviel/control_v11p_sd15_openpose
	Scribble-Conditioned Prompt-guided Image Generation	lllyasviel/control_v11p_sd15_scribble
	Segment-Conditioned Prompt-guided Image Generation	lllyasviel/control_v11p_sd15_seg
	HED-Boundary-Conditioned Prompt-guided Image Generation	lllyasviel/control_v11p_sd15_softedge
	Anime-Line-Art-Conditioned Prompt-guided Image Generation	lllyasviel/control_v11p_sd15s2_lineart_anime
	Prompt-guided Image Variations	stabilityai/stable-diffusion-2-1-unclip
Text Classification	Sentiment Analysis	DTAI-KULeuven/robbert-v2-dutch-sentiment
	Emotion Analysis	MilaNLProc/feel-it-italian-emotion
	Offensive Language Detection	Hate-speech-CNERG/bert-base-uncased-hatexplain
	Topic Classification	MaartenGr/BERTopic_Wikipedia
	Toxicity Analysis	OpenAssistant/reward-model-deberta-v3-large-v2
	Sensitive Text Detection	apanc/russian-inappropriate-messages
	Irony Detection	cardiffnlp/twitter-roberta-base-irony
	Bias Detection	cffl/bert-base-styleclassification-subjective-neutral
Feature Extraction	Text Feature Extraction	facebook/bart-large
Fill-Mask	Fill-Mask	CLTL/MedRoBERTa.nl
Text Generation	Text Generation	EleutherAI/gpt-j-6b
	Prompt Generation	FredZhang7/anime-anything-promptgen-v2
	Code Generation	NumbersStation/nsql-350M
	Specific Text Genre Generation	uer/gpt2-chinese-poem

Table 4. Our proposed categorization system(Part 2). The first and second columns are the two levels of the categories for each task. The third column provides an example of the candidate models.

Task	Subtask	Api Names
Text Generation	Text Generation	EleutherAI/gpt-j-6b
	Prompt Generation	FredZhang7/anime-anything-promptgen-v2
	Code Generation	NumbersStation/nsql-350M
	Specific Text Genre Generation	uer/gpt2-chinese-poem
Question Answering	Extractive Question Answering	CATIE-AQ/QAmembert
	Open Domain Question Answering	facebook/dpr-ctx_encoder-single-nq-base
Sentence Similarity	Sentence Similarity	intfloat/e5-small-v2
Summarization	Summarization	IDEA-CCNL/Randeng-Pegasus-523M-Summary-Chinese
	Title Generation	JulesBelveze/t5-small-headline-generator
	Keyword Generation	Voicelab/vlt5-base-keywords
Text-to-Image	Text to General Style Image	runwayml/stable-diffusion-v1-5
	Text to RGBD Image	Intel/ldm3d
	Text to Specific Style Image	nitrosocke/spider-verse-diffusion
	Text to 3D Image	openai/shap-e
	Text to Spectrogram Image	riffusion/riffusion-model-v1
Text-to-Speech	Text-to-Speech	Voicemod/fastspeech2-en-male1
Text-to-Video	Text-to-Video	damo-vilab/text-to-video-ms-1.7b
Text-to-Text	General Text Generation	1-800-BAD-CODE/xlm-roberta_punctuation_fullstop_truecase
	Sentence Correction	KES/T5-KES
	Generative Question Answering	MaRiOrOsSi/t5-base-finetuned-question-answering
	Question Generation	allenai/t5-small-squad2-question-generation
	Paraphraser	cointegrated/rut5-base-paraphraser
	Recipe Generation	flax-community/t5-recipe-generation
	Question Answering Generation	google/t5-small-ssm-nq
	Text Revision	grammarly/coedit-large
	Relation Extraction	ibm/knowgl-large
	Code Review	microsoft/codereviewer
	Commonsense Reasoning	mrm8488/t5-base-finetuned-common_gen
	Span Sentiment Extraction	mrm8488/t5-base-finetuned-span-sentiment-extraction
	Distractor	potsawee/t5-large-generation-race-Distractor
	Detoxification	s-nlp/bart-base-detox
	Symbolic Music Generation	sander-wood/text-to-music
Translation	Translation	Babelscape/mrebel-large
Zero-Shot Classification	Zero-Shot Classification	MoritzLaurer/DeBERTa-v3-base-mnli-fever-anli
Video Classification	Human action recognition video classification	MCG-NJU/videomae-base-finetuned-kinetics
Video-to-Text	Video Question Answering	kpyu/video-blip-flan-t5-xl-ego4d
	Video caption	Neleac/timesformer-gpt2-video-captioning
Feature Extraction	Video Feature Extraction	deepmind/multimodal-perceiver

Table 5. Our proposed categorization system(Part 3). The first and second columns are the two levels of the categories for each task. The third column provides an example of the candidate models.

References

[1] Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Alessandro Cappelli, Hamza Alobeidli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. The refinedweb dataset for falcon llm: outperforming curated corpora with web data, and web data only. arXiv preprint arXiv:2306.01116, 2023.

[2] BigScience Workshop, Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, et al. Bloom: A 176b-parameter open-access multilingual language model. arXiv preprint arXiv:2211.05100, 2022.

[3] Aiyuan Yang, Bin Xiao, Bingning Wang, Borong Zhang, Ce Bian, Chao Yin, Chenxu Lv, Da Pan, Dian Wang, Dong Yan, et al. Baichuan 2: Open large-scale language models. arXiv preprint arXiv:2309.10305, 2023.

[4] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In Proceedings of the IEEE/CVF Intersnational Conference on Computer Vision, pages 3836–3847, 2023.