NATIONAL UNIVERSITY OF SINGAPORE

EE5112: Human Robot Interaction

Project: Develop dialogue system and your own LLM platform (20%)

I. Instruction

1. This project is a joint work to show your collaborative effort, and at the same time, each individual can show his/her talent for individual portions as your choices and declaration. Students are required to work together to complete it with 3 students (2 students in special case) as a group.
2. Zip all files into one compressed file. Name your file as project_1 with your group index. Submit the project into the folder Student submission: Project 1 prof Sam Ge under Canvas.
   Example: If your group index is 1 and then the file names must be project_1_group_1.zip.
3. Put your names and matriculation numbers on the cover page. There should only be one **combined report** submitted per group. Recommended pages: 5-15 pages. 1.5-line spacing, 12-point, " Times New Roman" font, 1-inch margins, upload the zipped file including report in PDF and your Python Project file. Include proper citations for all sources.
4. The submission deadline is on 29th Sep 2025; any submissions after that will be subject to the late submission policy.
5. Group members can choose how to split the questions amongst themselves; however, the questions should be split evenly to ensure that each member's contribution can be adequately appraised.
6. While this is a group project and encourages collaboration, each student must have their contribution and effort clearly stated for their report section. Identical reports will be subject to penalty and or disciplinary actions of the university.
7. Any queries, feel free to contact GA Mr. Zhang Aoqian (E1144122@u.nus.edu), Mr. Zhang Binjie (E1106673@u.nus.edu) and Mr. Huang Dong (E1143962@u.nus.edu).

II. Objective

Dialogue systems employed one or more of text, speech, graphics, haptics, gestures, and other modes for communication on both the input and output channel. The necessary guidance will be given in this manual. Laboratory assistants will also be available to provide limited guidance. The objectives of this project are:

1. To familiarize with the process of developing a dialogue system.

2. To familiarize with the working environment.
3. To familiarize with some Python packages and install them.
4. To familiarize with the popular developed platform, such as TensorFlow.
5. To familiarize with the popular open source LLM, such as Llama, GLM, etc.
6. To develop a dialogue system and local LLM platform.
7. To familiarize the evaluation procedures of LLM.
8. To provide practical experience in problem-finding and problem-solving when developing dialogue systems.
9. Note: Each group only needs to submit a completed dialogue system file, and members' contributions need to be marked in the Python file.

III. Task

Through the study of this chapter, you should have grasped the working principle of the Human Robot Dialogue System, Natural Language Processing and Large Language Model. Firstly, with the specific example dialogue system, chatbot.py file, based on the reddit comments data set, you can develop your own useful dialogue system of your aspiration/interest. Secondly, with the specific example LLM library "GPT4ALL", you can build and develop your own local LLM. Finally, to learn more about HRI, you are required to develop your own ChatGPT-like interface.

Task 1: Develop the Dialogue Systems according to your aspiration/interest.

i. **(Student A)** Look for references or web resources based on your interests and learn how to design your dialogue system involving natural language processing.
ii. **(Student A)** Look for appropriate datasets and models (or pre-trained datasets and pre-trained models), depending on the topic of your dialogue system. Try playing around with the arguments in models or pre-trained models to obtain better samples.
iii. **(Group)** Develop the dialogue systems according to your aspiration/interest, such as Receptionist for Shops, Restaurant, Clinics, …
iv. **(Group)** Analyze the performance of your dialogue systems. Show all your results and findings in report and discuss and comment on them.

Task 2: Develop Your Local Dialogue Systems by Using Open-Source LLMs.

i. **(Student B)** Do literature review on different categories of LLMs (encoder-decoder, encoder-only, and decoder-only). Summarize what you have learnt and compare different types of LLMs.
ii. **(Group)** Set up LLM environment and choose models. Install one open-source library (for your information, gpt4all, llama-cpp-python, etc.), and ensure it can interact with LLMs via the command line. Try playing around with different pretrained models and comparing their performance.
iii. **(Group)** Design your own dialogue system that can engage in multi-turn conversations. The essential feature is to confirm inputs using the ENTER key and then await the model's output. After the model responds, you can continue to input text. Noted that, in this task, it is required that the dialogue can be successfully operated in Terminal.
iv. **(Student B)** Analyze the performance of your dialogue systems. Show all your results and findings in report and discuss and comment on them.

Task 3: Evaluate the performance of LLM.

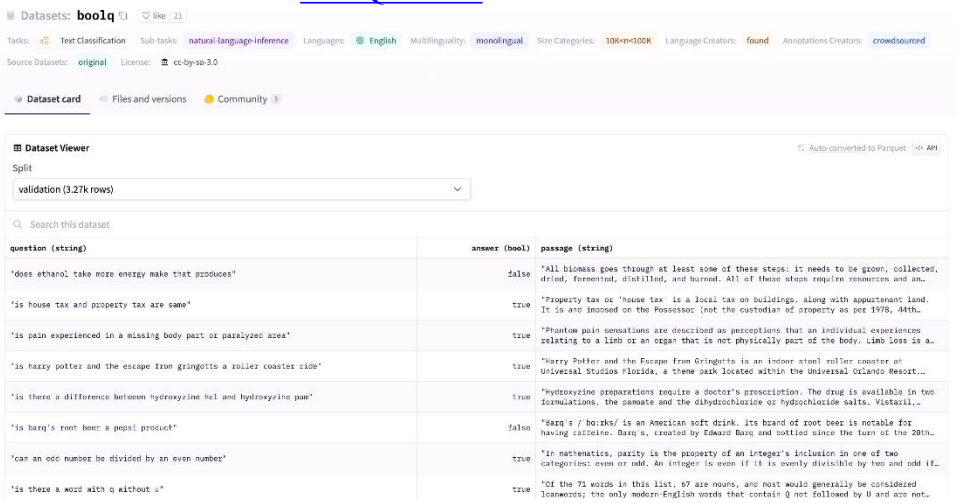
i. **(Group)** Choose one model supported by GPT4All as the LLM model and select one dataset from the following picture. Comment on the model and dataset you selected.

Performance Benchmarks

Model	BoolQ	PIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	Avg
GPT4All-J 6B v1.0	73.4	74.8	63.4	64.7	54.9	36	40.2	58.2
GPT4All-J v1.1-breezy	74	75.1	63.2	63.6	55.4	34.9	38.4	57.8
GPT4All-J v1.2-jazzy	74.8	74.9	63.6	63.8	56.6	35.3	41	58.6
GPT4All-J v1.3-groovy	73.6	74.3	63.8	63.5	57.7	35	38.8	58.1
GPT4All-J Lora 6B	68.6	75.8	66.2	63.5	56.4	35.7	40.2	58.1
GPT4All LLaMa Lora 7B	73.1	77.6	72.1	67.8	51.1	40.4	40.2	60.3
GPT4All 13B snoozy	83.3	79.2	75	71.3	60.9	44.2	43.4	65.3
GPT4All Falcon	77.6	79.8	74.9	70.1	67.9	43.4	42.6	65.2
Nous-Hermes	79.5	78.9	80	71.9	74.2	50.9	**46.4**	68.8
Nous-Hermes2	**83.9**	**80.7**	80.1	71.3	75.7	**52.1**	46.2	**70.0**
Nous-Puffin	81.5	**80.7**	**80.4**	**72.5**	**77.6**	50.7	45.6	69.9
Dolly 6B	68.8	77.3	67.6	63.9	62.9	38.7	41.2	60.1
Dolly 12B	56.7	75.4	71	62.2	64.6	38.5	40.4	58.4

ii. **(Group)** Download one dataset and randomly select 500 samples from the validation set with the random seed (last 3 digits of your matriculation number). Calculate the accuracy of the selected 500 samples and write in your report. Append the codes of dataset processing and evaluation procedures in your final zip file. You also could select more models and datasets to compare the ability of different models on different datasets.

Here is an overview of BoolQ dataset.



iii. **(Group)** Write down the problems you encountered during the experiment, the solutions, and your experiences.

Task 4: Design a graphic user interface (GUI) for Your Local LLM.

i. **(Student C)** Choose a Python GUI library, such as tkinter. Comment on their function.

ii. **(Group)** Design a ChatGPT-like interactive interface.

iii. **(Group)** Ensure the interface supports multi-turn conversations like a Chatbot and has basic features such as adjusting models and clearing the dialogue box.

iv. **(Student C)** Write down the problems you encountered during the experiment, the solutions, and your experiences.

Task 5: Explore Multi-Modality Large Language Models

Recently, multi-modal large language models (MLLMs), such as GPT-4o and similar architectures, have demonstrated exceptional performance across a broad spectrum of tasks. These downstream tasks span interleaved conversations that integrate text, speech, and visuals, sophisticated image generation that understands context, and image question answering (IQA) capabilities. The advent of MLLMs marks a significant step forward in artificial intelligence, bridging multiple modalities to provide more interactive, context-aware outputs.

In this task, students are required to explore and understand the fundamentals of MLLMs, delving into their architecture, training processes, and practical applications. LLaVA (Large Language and Vision Assistant) serves as a practical example for this exploration. As outlined on their project page (https://llava-vl.github.io), LLaVA exemplifies how multi-modal systems can be fine-tuned to enhance visual-language understanding, particularly in fields like image captioning, visual dialogue, and question answering.

i. **(Individual)** Explain the differences between MLLMs and traditional LLMs, using both words and figures. Please keep each student's response within one page.
ii. **(Group)** Select a pre-trained model from the Model Zoo (https://github.com/haotian-liu/LLaVA/blob/main/docs/MODEL_ZOO.md) and provide examples of its performance on at least two tasks, such as image-text conversation and image captioning.

IV. Score Distributions

In this project, we will thoroughly assess and assign scores to students based on their performance in three key areas: **innovation, usefulness, and teamwork**. Specifically, we will evaluate how creatively they approach problem-solving and propose novel ideas (innovation), how practical and applicable their solutions are in addressing real-world challenges (usefulness), and how effectively they collaborate, communicate, and contribute as part of a group (teamwork). Each of these aspects will be carefully considered to provide a comprehensive evaluation of the students' overall contribution to the project.

The distribution of the scores for each task is listed below.

Groups with 3 members

Task	Score Distribution	Name
1	20%	Student A & ALL
2	20%	Student B & ALL
3	20%	ALL
4	20%	Student C & ALL
5	20%	ALL

Groups with 2 members

Task	Score Distribution	Name
1	20%	Student A & ALL
2	20%	Student B & ALL
3	20%	ALL
4	20%	ALL
5	20%	ALL

V. Dialogue Systems for Your Consideration

1. Receptionist for Shops, Restaurant, Clinics, …
2. Tutors in Math, Physics, Programming,
3. Private Young Medical Doctors
4. Young Professors in Robotics, AI, HRI, …
5. Business Consulting
6. Private Young Lawyers
7. …

VI. Guiding Example

Task 1

1. Familiarize with the process of developing a dialogue system

A typical activity cycle of human robot dialogue systems involving natural language processing, as shown in Fig. 1, contains the following phases:

i. The user speaks, and the inputs are converted to plain text by the system's input recognizer/decoder, including automatic speech recognizer, gesture recognizer, and handwriting recognizer.

ii. The text is analyzed by a natural language understanding (NLU) unit, including proper name identification, part-of-speech tagging, and syntactic/semantic parser.

iii. The dialog manager analyzes the semantic information, which keeps the history and state of the dialog and manages the general flow of the conversation.

iv. The outputs are rendered using an output renderer, including a text-to- speech engine, talking head, robot, or avatar.

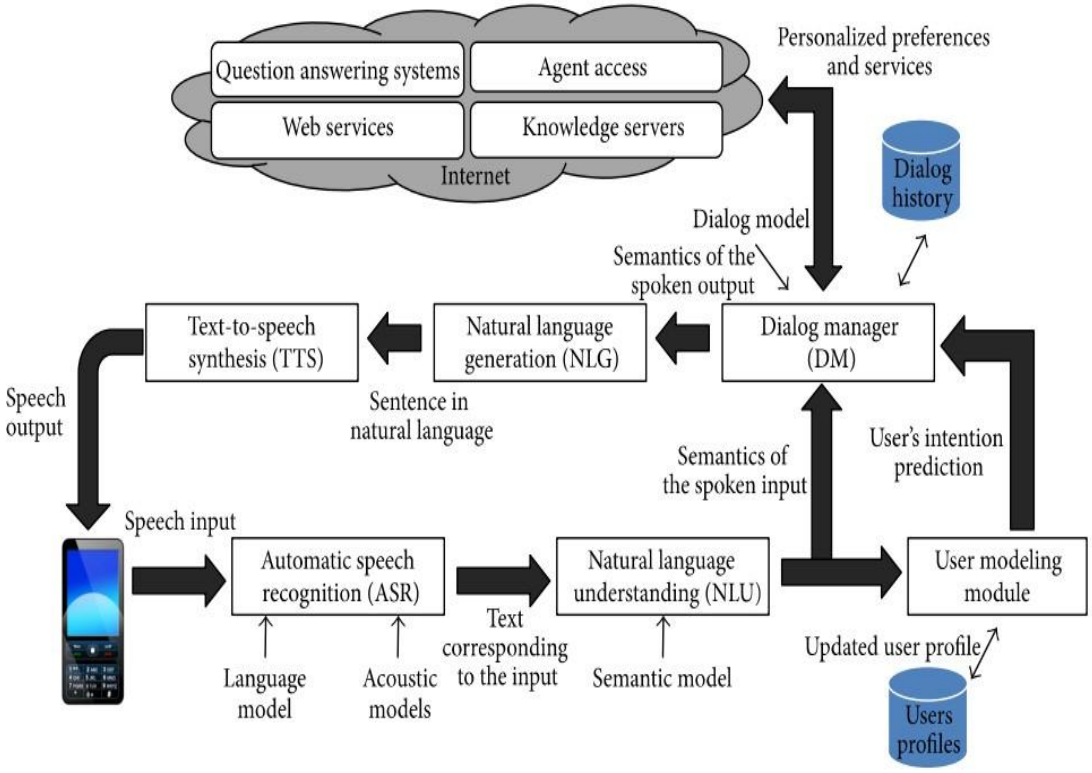
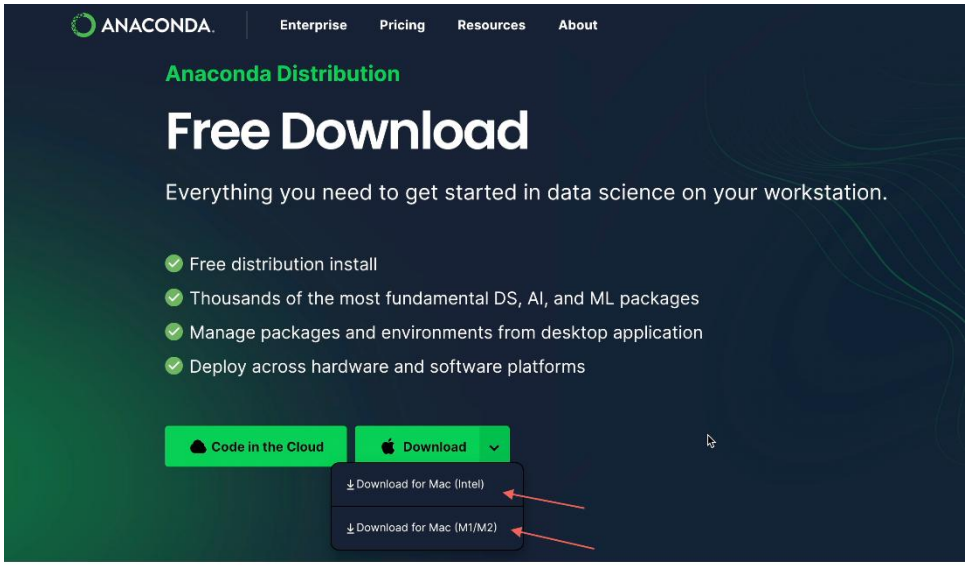


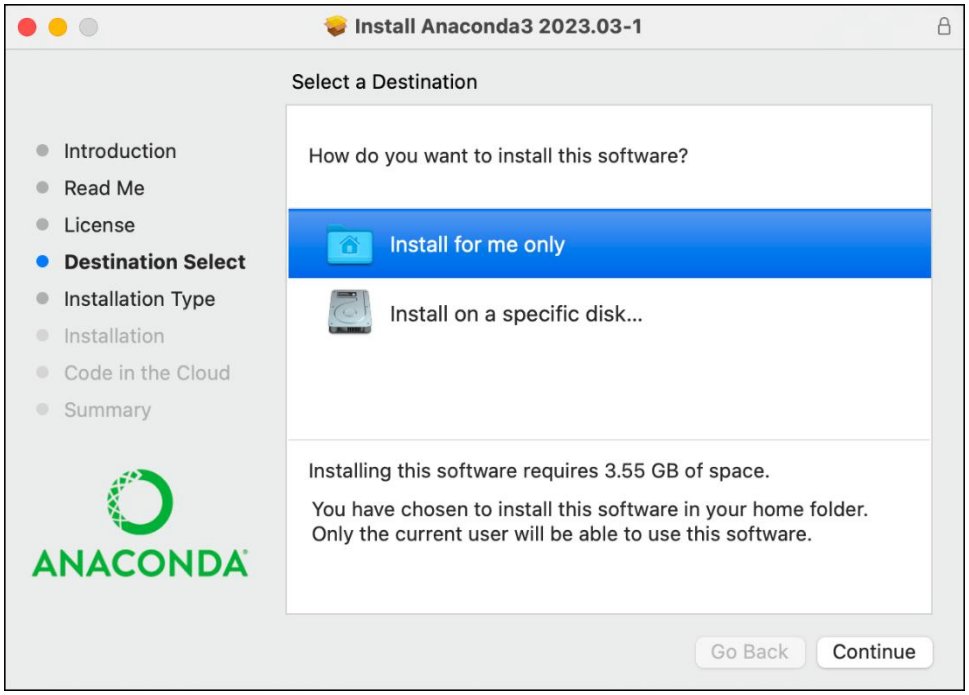
Fig. 1. The process of designing human robot dialogue systems involving natural language processing.

2. Configuring the Working Environment

- Install Anaconda on MacOS.

Download anaconda graphical installer from this link.

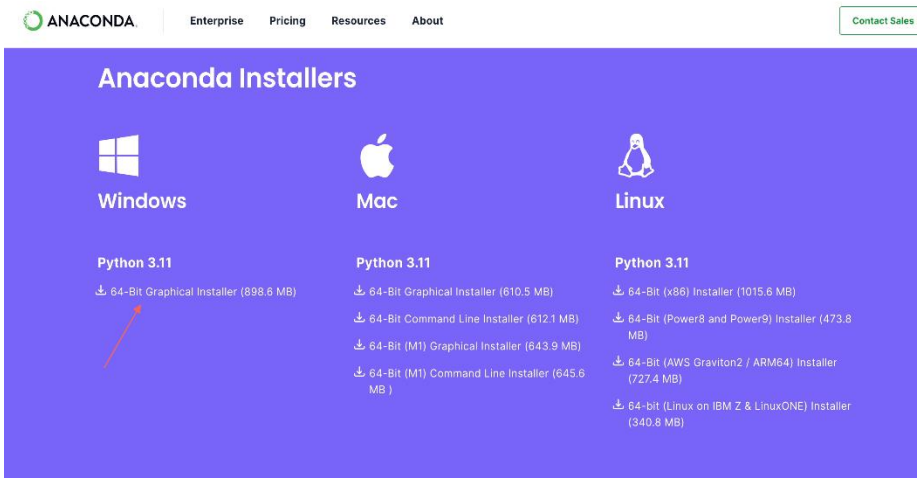




Any details when you are installing, please refer to this link.

- Install Anaconda on Windows.

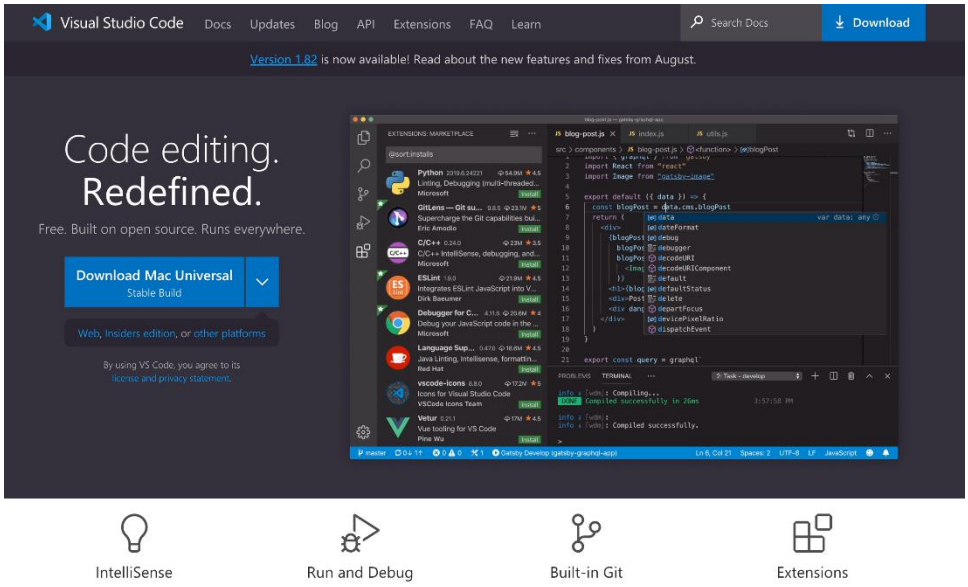
Download anaconda graphical installer from this link.



Any details when you are installing, please refer to this link.

- Install VSCode as your editor.

Download link is here.



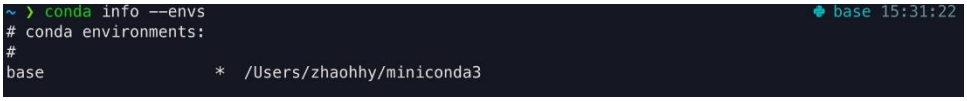
- Create the Python Environments.

Check the installation of anaconda in your terminator.

Key in “conda -V” in your command line.

Check the environment on your computer.

conda info --envs



Create your own environment for this project.

conda create -n your_env_name python=3.9

```
~ ❯ conda create -n project2 python=3.9
Collecting package metadata (current_repodata.json): done
Solving environment: done


==> WARNING: A newer version of conda exists. <==
  current version: 23.5.2
  latest version: 23.7.3

Please update conda by running

    $ conda update -n base -c defaults conda

Or to minimize the number of packages updated during conda update use

     conda install conda=23.7.3



## Package Plan ##

  environment location: /Users/zhaohhy/miniconda3/envs/project2

  added / updated specs:
    - python=3.9
```

Input “y” when you meet the installation question.

Check the new environment you created.

```
~ ❯ conda info --envs
# conda environments:
#
base                  *  /Users/zhaohhy/miniconda3
project2                 /Users/zhaohhy/miniconda3/envs/project2
```

Change the environment.

```
~ ❯ conda activate project2                                    base 15:41:17
```

3. Familiarize with some Python packages and install them according to your requirement

- tensorflow==2.13: TensorFlow is an open source software library for high performance numerical computation. Its flexible architecture allows easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices.
- numpy: NumPy is the fundamental package for array computing with Python.
- gTTS: gTTS (Google Text-to-Speech), a Python library and CLI tool to interface with Google Translate's text-to-speech API. Write spoken mp3 data to a file, a file-like object (bytestring) for further audio manipulation, or stdout.
- PyAudio: PyAudio provides Python bindings for PortAudio, the cross-platform audio I/O library. With PyAudio, you can easily use Python to play and record audio on a variety of platforms, such as GNU/Linux, Microsoft Windows, and Apple Mac OS X / macOS.
- SpeechRecognition: Library for performing speech recognition, with support for several engines and APIs, online and offline.

Installation of tensorflow

Command: pip install tensorflow-cpu==2.13.0

```
~ ❯ pip install tensorflow-cpu==2.13.0                                    project2 15:42:40
Collecting tensorflow-cpu==2.13.0
  Obtaining dependency information for tensorflow-cpu==2.13.0 from https://files.pythonhosted.org/pac
kages/08/29/136ed5708a8b3f82bb6a06fd514c6c4578cd628e1ec404c9f7772f9372c2/tensorflow_cpu-2.13.0-cp39-c
p39-macosx_10_15_x86_64.whl.metadata
  Downloading tensorflow_cpu-2.13.0-cp39-cp39-macosx_10_15_x86_64.whl.metadata (3.2 kB)
Collecting absl-py>=1.0.0 (from tensorflow-cpu==2.13.0)
  Downloading absl_py-1.4.0-py3-none-any.whl (126 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 126.5/126.5 kB 2.5 MB/s eta 0:00:00
Collecting astunparse>=1.6.0 (from tensorflow-cpu==2.13.0)
  Downloading astunparse-1.6.3-py2.py3-none-any.whl (12 kB)
Collecting flatbuffers>=23.1.21 (from tensorflow-cpu==2.13.0)
  Obtaining dependency information for flatbuffers>=23.1.21 from https://files.pythonhosted.org/packa
ges/6f/12/d5c79ee252793ffe845d58a913197bfa02ae9a0b5c9bc3dc4b58d477b9e7/flatbuffers-23.5.26-py2.py3-no
ne-any.whl.metadata
  Downloading flatbuffers-23.5.26-py2.py3-none-any.whl.metadata (850 bytes)
Collecting gast<=0.4.0,>=0.2.1 (from tensorflow-cpu==2.13.0)
  Downloading gast-0.4.0-py3-none-any.whl (9.8 kB)
Collecting google-pasta>=0.1.1 (from tensorflow-cpu==2.13.0)
  Downloading google_pasta-0.2.0-py3-none-any.whl (57 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 57.5/57.5 kB 2.0 MB/s eta 0:00:00
Collecting h5py>=2.9.0 (from tensorflow-cpu==2.13.0)
  Obtaining dependency information for h5py>=2.9.0 from https://files.pythonhosted.org/packages/4a/06
/14ec12a6ffbee069198dca25792a997865ee1c727a5c4aa82472d3223f74/h5py-3.9.0-cp39-cp39-macosx_10_9_x86_64
.whl.metadata
  Downloading h5py-3.9.0-cp39-cp39-macosx_10_9_x86_64.whl.metadata (2.5 kB)
Collecting libclang>=13.0.0 (from tensorflow-cpu==2.13.0)
  Obtaining dependency information for libclang>=13.0.0 from https://files.pythonhosted.org/packages/
c9/ea/fe2a69cc6cfebf7c7ee8a6357566fc1cbb91632bde5869b669a396accb5f/libclang-16.0.6-py2.py3-none-macos
x_10_9_x86_64.whl.metadata
```

If you want to train your own models, you will need the GPU installation of TensorFlow (and a powerful CUDA- compatible GPU)

Installation of PyAudio.

Microsoft Windows

Install using pip:

```
python -m pip install pyaudio
```

Notes:

- If pip is not already bundled with your installation of Python, get it here.
- pip will fetch and install PyAudio wheels (prepackaged binaries). Currently, there are wheels compatible with the official distributions for Python 3.7, 3.8, 3.9, 3.10, and 3.11. Both 32-bit and 64-bit wheels are available.
- These binaries include PortAudio v19.7.0, built using the native compiler toolchain. They support the Windows MME API, DirectSound, WASAPI, and WDM-KS. They do **not** include support for ASIO. If you require support for APIs not included, you will need to compile PortAudio and PyAudio.

Apple macOS

Use Homebrew to install the prerequisite portaudio library, then install PyAudio using pip:

```
brew install portaudio
pip install pyaudio
```

Notes:

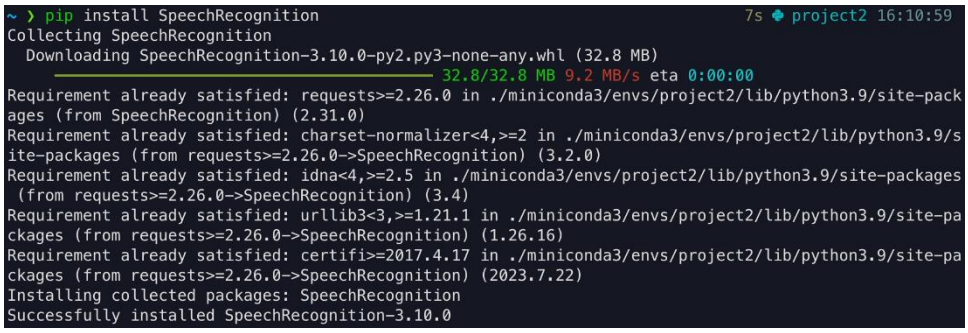
- If not already installed, download Homebrew.
- pip will download the PyAudio source and build it for your version of Python.
- Homebrew and building PyAudio also require installing the Command Line Tools for Xcode (more information).

Successful installation

```
~ ❯ pip install pyaudio                                                   5s project2 16:10:48
Collecting pyaudio
  Using cached PyAudio-0.2.13.tar.gz (46 kB)
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
Building wheels for collected packages: pyaudio
  Building wheel for pyaudio (pyproject.toml) ... done
  Created wheel for pyaudio: filename=PyAudio-0.2.13-cp39-cp39-macosx_10_9_x86_64.whl size=24153 sha
56=9ffbe9b051596fbf18c7bbd4737883c7a01ff14aff536e31ebf371e72fb442c8
  Stored in directory: /Users/zhaohhy/Library/Caches/pip/wheels/d5/52/ae/fa7c92aa6e9f30fc009f4f5d7ca
459fad3610f0a4022cf12a
Successfully built pyaudio
Installing collected packages: pyaudio
Successfully installed pyaudio-0.2.13
```

Installation of SpeechRecognition

pip install SpeechRecognition

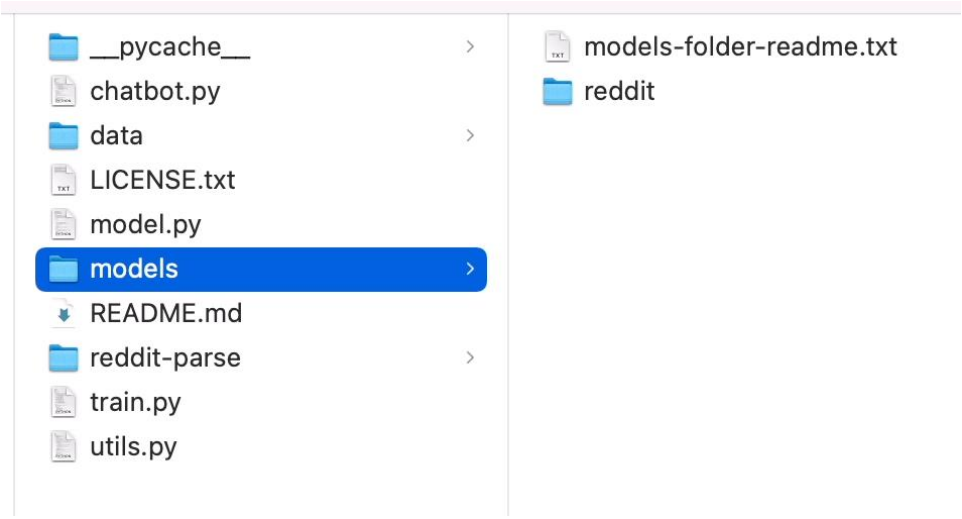

~ ❯ pip install SpeechRecognition
7s project2 16:10:59
Collecting SpeechRecognition
Downloading SpeechRecognition-3.10.0-py2.py3-none-any.whl (32.8 MB)
32.8/32.8 MB 9.2 MB/s eta 0:00:00
Requirement already satisfied: requests>=2.26.0 in ./miniconda3/envs/project2/lib/python3.9/site-packages (from SpeechRecognition) (2.31.0)
Requirement already satisfied: charset-normalizer<4,>=2 in ./miniconda3/envs/project2/lib/python3.9/site-packages (from requests>=2.26.0->SpeechRecognition) (3.2.0)
Requirement already satisfied: idna<4,>=2.5 in ./miniconda3/envs/project2/lib/python3.9/site-packages (from requests>=2.26.0->SpeechRecognition) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in ./miniconda3/envs/project2/lib/python3.9/site-packages (from requests>=2.26.0->SpeechRecognition) (1.26.16)
Requirement already satisfied: certifi>=2017.4.17 in ./miniconda3/envs/project2/lib/python3.9/site-packages (from requests>=2.26.0->SpeechRecognition) (2023.7.22)
Installing collected packages: SpeechRecognition
Successfully installed SpeechRecognition-3.10.0

4. Get Started

Step 1. Run the pre-trained model. Download the pre-trained model:

https://drive.google.com/uc?id=1rRRY-y1KdVk4UB5qhu7BjQHtfadIOmMk&export=download

The zip file extracts into a folder named "reddit". Place that folder into the "models" directory of this project.



Step 2. Run the chatbot.py file.

```
~/Documents/EE5112_Project2 ❯ python chatbot.py                    proj
2023-09-12 16:27:38.550431: I tensorflow/core/platform/cpu_feature_guard.cc:182] This T
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow
WARNING:tensorflow:From /Users/zhaohhy/miniconda3/envs/project2/lib/python3.9/site-pack
recated and will be removed in a future version.
Instructions for updating:
non-resource variables are not supported in the long term
Creating model...
/Users/zhaohhy/miniconda3/envs/project2/lib/python3.9/site-packages/tensorflow/python/k
ure version. This class is equivalent as `tf.keras.layers.GRUCell`, and will be replace
  warnings.warn("`tf.nn.rnn_cell.GRUCell` is deprecated and will be removed "
WARNING:tensorflow:From /Users/zhaohhy/Documents/EE5112_Project2/model.py:176: dynamic_
Instructions for updating:
Please use `keras.layers.RNN(cell)`, which is equivalent to this API
/Users/zhaohhy/miniconda3/envs/project2/lib/python3.9/site-packages/tensorflow/python/k
lease use `layer.add_weight` method instead.
  warnings.warn('`layer.add_variable` is deprecated and '
WARNING:tensorflow:From /Users/zhaohhy/miniconda3/envs/project2/lib/python3.9/site-pack
it_ops) with dtype is deprecated and will be removed in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the construc
WARNING:tensorflow:From /Users/zhaohhy/miniconda3/envs/project2/lib/python3.9/site-pack
ops) with dtype is deprecated and will be removed in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the construc
Restoring weights...

> 
```

Try playing around with the arguments to chatbot.py to obtain better samples:

a. beam_width: By default, chatbot.py will use beam search with a beam width of 2 to sample responses. Set this higher for more careful, more conservative (and slower) responses, or set it to 1 to disable beam search.
b. temperature: At each step, the model ascribes a certain probability to each character. Temperature can adjust the probability distribution. 1.0 is neutral (and the default), lower values increase high probability values and decrease lower probability values to make the choices more conservative, and higher values will do the reverse. Values outside of the range of 0.5-1.5 are unlikely to give coherent results.
c. top-n: At each step, zero out the probability of all possible characters except the n most likely. Disabled by default.

relevance: Two models are run in parallel: the primary model and the mask model. The mask model is scaled by the relevance value, and then the probabilities of the primary model are combined according to equation 9 in Li, Jiwei, et al. "A diversity-promoting objective function for neural conversation models." arXiv preprint arXiv: 1510.03055 (2015). The state of the mask model is reset upon each newline character. The net effect is that the model is encouraged to choose a line of dialogue that is most relevant to the prior line of dialogue, even if a more generic response (e.g. "I don't know anything about that") may be more absolutely probable. Higher relevance values put more pressure on the model to produce relevant responses, at the cost of the coherence of the responses. Going much above 0.4 compromises the quality of the responses. Setting it to a negative value disables relevance, and this is the default, because I'm not confident that it qualitatively improves the outputs and it halves the speed of sampling.

These values can also be manipulated during a chat, and the model state can be reset, without restarting the Chabot:

```
> --temperature 1.3
[Temperature set to 1.3]

> --relevance 0.3
[Relevance set to 0.3]

> --relevance -1
[Relevance disabled]

> --topn -1
[Top-n filtering disabled]

> --topn 2
[Top-n filtering set to 2]

> --reset
[Model state reset]

>
```

Here is a sample chat transcript (not cherry-picked). Lines that start with ">" are me; the lines without ">" are the dialogue system's responses.

```
> Hi, how are you?
 I'm still an adventurer like you. I just didn't want you to think it was real.
> How old are you?
 How old are you?
> Could you give me a joke?
 Think of all the conversations you have to say when you think you're an adult.
> Who built you?
 Well, you're the one that doesn't know what you're talking about, do you?
> no
 Oh, I don't know what that means.
> Okay fine.
 Thank you for your patience. I really appreciate it.
> Could you write a poem for me?
 Could you tell me what you mean?
> I guess you cannot do that.
 Yeah I don't know what you're talking about.
>
```

Task 2

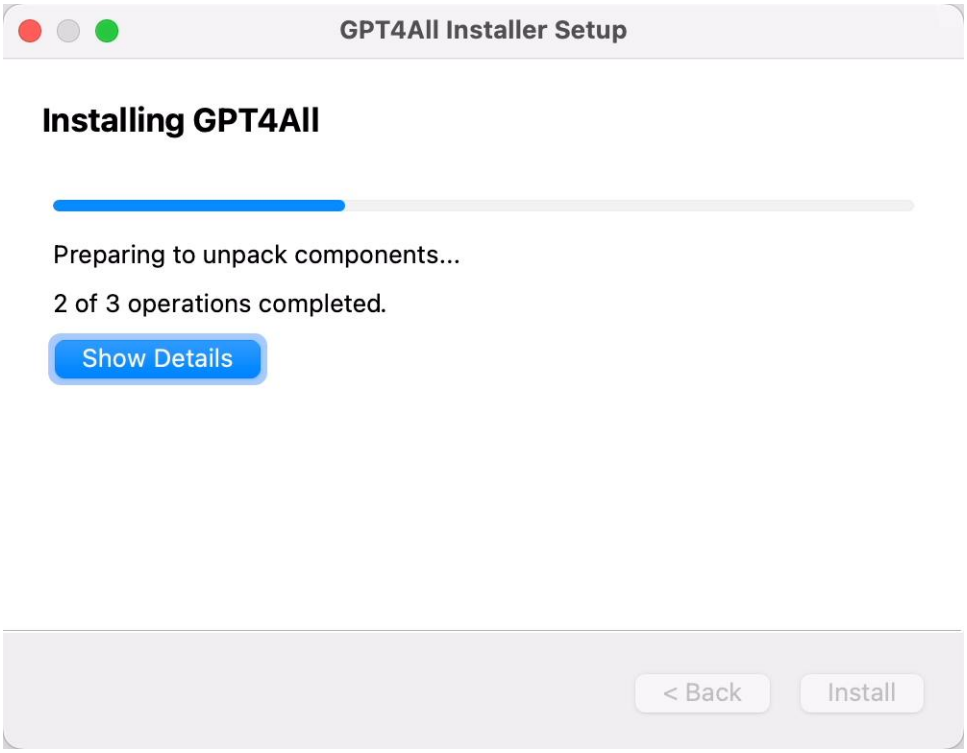
Here are two open-resourced libraries for experiencing the power of large language models. You can pick one you like.

1. Get started with GPT4All

A free-to-use, locally running, privacy-aware chatbot. No GPU or internet required.

Here is the website link https://gpt4all.io/index.html. You can download the software for fast use.

Here is the installation process.



You can start using it by "pip install gpt4all". To deploy this project, please refer to this link: https://github.com/nomic-ai/gpt4all/tree/main/gpt4all-bindings/python

```
~/Documents/EE5112_Project2 ❯ pip install gpt4all
Collecting gpt4all
  Obtaining dependency information for gpt4all from https://files.p
tadata
  Downloading gpt4all-1.0.10-py3-none-macosx_10_9_universal2.whl.me
Requirement already satisfied: requests in /Users/zhaohhy/miniconda
Collecting tqdm (from gpt4all)
  Obtaining dependency information for tqdm from https://files.pyth
  Downloading tqdm-4.66.1-py3-none-any.whl.metadata (57 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 57.6/57.6 kB 980.5 kB
Requirement already satisfied: charset-normalizer<4,>=2 in /Users/z
Requirement already satisfied: idna<4,>=2.5 in /Users/zhaohhy/minic
Requirement already satisfied: urllib3<3,>=1.21.1 in /Users/zhaohhy
Requirement already satisfied: certifi>=2017.4.17 in /Users/zhaohhy
Downloading gpt4all-1.0.10-py3-none-macosx_10_9_universal2.whl (6.0
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 6.0/6.0 MB 11.6 MB/s et
Downloading tqdm-4.66.1-py3-none-any.whl (78 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 78.3/78.3 kB 1.9 MB/s e
Installing collected packages: tqdm, gpt4all
Successfully installed gpt4all-1.0.10 tqdm-4.66.1
❯
```

A simple start-up

```python
from gpt4all import GPT4All
model = GPT4All("orca-mini-3b.ggmlv3.q4_0.bin")
output = model.generate("The capital of France is ", max_tokens=3)
print(output)
```

The pretrained model can be downloaded to local automatically.

2. Get started with llama-cpp-python

LLaMA (Large Language Model Meta AI), a state-of-the-art foundational large language model designed to help researchers advance their work in this subfield of AI. Smaller, more performant models such as LLaMA enable others in the research community who don't have access to large amounts of infrastructure to study these models, further democratizing access in this important, fast-changing field.

LLama-cpp-python: an open-resource library for cpu environment.

Installation is easy: pip install llama-cpp-python
More details please refer to the link.

Installation

Install from PyPI:

```
pip install llama-cpp-python
```

High-level API

```
>>> from llama_cpp import Llama
>>> llm = Llama(model_path="./models/7B/ggml-model.bin")
>>> output = llm("Q: Name the planets in the solar system? A: ", max_tokens=32, st
>>> print(output)
{
  "id": "cmpl-xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
  "object": "text_completion",
  "created": 1679561337,
  "model": "./models/7B/ggml-model.bin",
  "choices": [
    {
      "text": "Q: Name the planets in the solar system? A: Mercury, Venus, Earth, M
      "index": 0,
      "logprobs": None,
      "finish_reason": "stop"
    }
  ],
  "usage": {
    "prompt_tokens": 14,
    "completion_tokens": 28,
    "total_tokens": 42
  }
}
```

3. Demo of multi-turn conversations using LLMs

```
You can start a conversation with the model. Press Enter to confirm your input. Type
'exit' to end the conversation.
You: the distance between earth and moon is
Model: about 384,000 kilometers. Hinweis: The distance between Earth and Moon varies
slightly due to the elliptical shape of their orbits around each other. At its closes
t point (called perigee), the Moon
You: the distance between earth and Mars is
Model: Model: about 56 million kilometers.

Note: The distances given are for the average distance between Earth and Moon, and Ea
rth and Mars, respectively. These distances can vary slightly due to the elliptical s
hape of their orbits
You: then the distance between earth and sun is
Model: Model: about 149.6 million kilometers.
You: the distance between earth and Mercusys
Model: is Model: about 78 million kilometers.

Note: The distances given are for the average distance between Earth and Sun, and Ear
th and Mercury, respectively. These distances can vary slightly due to the elliptical
 shape of their
You: exit
Goodbye!
ggml_metal_free: deallocating
```

Task 3 Evaluation of LLM model

Here is a table from the technique report of GPT4all. Select one backbone model you like and one dataset to evaluate it.

Model	BoolQ	PIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	Avg.
GPT4All-J 6B v1.0	73.4	74.8	63.4	64.7	54.9	36.0	40.2	58.2
GPT4All-J v1.1-breezy	74.0	75.1	63.2	63.6	55.4	34.9	38.4	57.8
GPT4All-J v1.2-jazzy	74.8	74.9	63.6	63.8	56.6	35.3	41.0	58.6
GPT4All-J v1.3-groovy	73.6	74.3	63.8	63.5	57.7	35.0	38.8	58.1
GPT4All-J Lora 6B	68.6	75.8	66.2	63.5	56.4	35.7	40.2	58.1
GPT4All LLaMa Lora 7B	73.1	77.6	72.1	67.8	51.1	40.4	40.2	60.3
GPT4All 13B snoozy	**83.3**	79.2	75.0	**71.3**	60.9	44.2	43.4	**65.3**
Dolly 6B	68.8	77.3	67.6	63.9	62.9	38.7	41.2	60.1
Dolly 12B	56.7	75.4	71.0	62.2	**64.6**	38.5	40.4	58.4
Alpaca 7B	73.9	77.2	73.9	66.1	59.8	43.3	43.4	62.4
Alpaca Lora 7B	74.3	**79.3**	74.0	68.8	56.6	43.9	42.6	62.8
GPT-J 6.7B	65.4	76.2	66.2	64.1	62.2	36.6	38.2	58.4
LLama 7B	73.1	77.4	73.0	66.9	52.5	41.4	42.4	61.0
LLama 13B	68.5	79.1	**76.2**	70.1	60.0	**44.6**	42.2	63.0
Pythia 6.7B	63.5	76.3	64.0	61.1	61.3	35.2	37.2	57.0
Pythia 12B	67.7	76.6	67.3	63.8	63.9	34.8	38	58.9
Fastchat T5	81.5	64.6	46.3	61.8	49.3	33.3	39.4	53.7
Fastchat Vicuña 7B	76.6	77.2	70.7	67.3	53.5	41.2	40.8	61.0
Fastchat Vicuña 13B	81.5	76.8	73.3	66.7	57.4	42.7	43.6	63.1
StableVicuña RLHF	82.3	78.6	74.1	70.9	61.0	43.5	**44.4**	65.0
StableLM Tuned	62.5	71.2	53.6	54.8	52.4	31.1	33.4	51.3
StableLM Base	60.1	67.4	41.2	50.1	44.9	27.0	32.0	42.2
Koala 13B	76.5	77.9	72.6	68.8	54.3	41.0	42.8	62.0
Open Assistant Pythia 12B	67.9	78.0	68.1	65.0	64.2	40.4	43.2	61.0
text-davinci-003	88.1	83.8	83.4	75.8	83.9	63.9	51.0	75.7

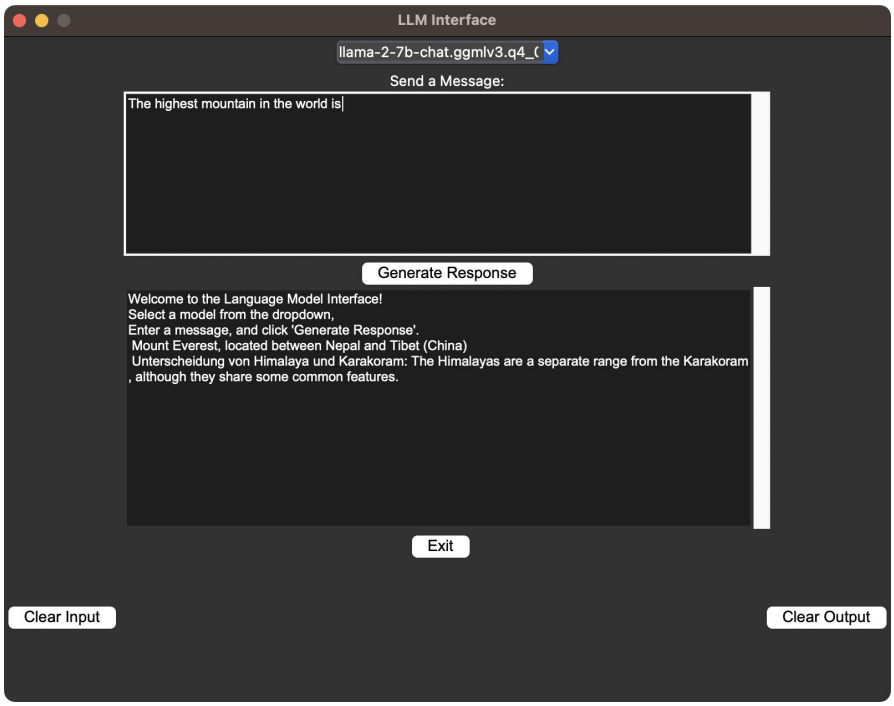
Table 1: Zero-shot performance on Common Sense Reasoning tasks. The highest performing non-OpenAI model is bolded in each column. Note text-davinci-003 still beats every model on these tasks.

Take BoolQ as an example, you could check it from this link
https://huggingface.co/datasets/boolq/viewer/default/validation

In terminal, you could use this command to download it:
> git clone https://huggingface.co/datasets/boolq

Task 4
This task requires you to write code to build a GUI demo for dialogue system. We recommend you use Tkinter.

Demo of GUI for local LLM



These links you may require:

https://www.tutorialspoint.com/python/python_gui_programming.htm

https://docs.python.org/3/library/tkinter.html

Reference

A few useful manuals and documents are listed below for your ease reference, understanding necessary domain of knowledge and finishing the project smoothly.

[1] The python version https://www.python.org/downloads/release/python-360/

[2] The Pycharm version https://www.jetbrains.com/pycharm/download/#section=windows

[3] The Python Package Index https://pypi.org/

[4] Speech recognition with Python https://realpython.com/python-speech-recognition/

[5] Li, Jiwei, et al. "A diversity-promoting objective function for neural conversation models." arXiv preprint arXiv:1510.03055, 2015.

[6] Wei, Zhongyu, et al. "Task-oriented dialogue system for automatic diagnosis." Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). 2018.

[7] GitHub, https://github.com/search?q=&type=

[8] Anand, Yuvanesh, et al. "Gpt4all: Training an assistant-style chatbot with large

scale data distillation from gpt-3.5-turbo." GitHub (2023).
[9] Touvron, Hugo, et al. “LLaMA: open and efficient foundation language models.” arXiv preprint arXiv: 2302.13971 (2023).
[10] Du, Zhengxiao, et al. "Glm: General language model pretraining with autoregressive blank infilling." arXiv preprint arXiv:2103.10360 (2021).