NATIONAL UNIVERSITY OF SINGAPORE

EE4705: Human-Robot Interaction

Project: Develop a useful dialogue system of your own interest (15%)

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 2 - 3 students as a group. Groups with 2 members need to complete Task 1. Groups with 3 members need to complete Task 2.
2. Zip all files into one compressed file. Name your file as project_2 with your group index. Submit the project into the folder Student submission: Project 2 pf Sam Ge under LumiNUS.
   Example: If your group index is 1 and then the file names must be project_2_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: 6–15 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 23 Sep 2022; 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 Chen Xuyang, chenxuyang@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 develop a useful dialogue system of your aspiration/interest.
6. To provide practical experience in problem-finding and problem-solving when developing dialogue systems.

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 and Natural Language Processing. With the specific example dialogue system, chatbot.py file, based on the reddit comments data set, you can develop a useful dialogue system of your aspiration/interest.

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

i. Look for references or web resources based on your interests and learn how to design your dialogue system involving natural language processing.
ii. 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. Develop the dialogue systems according to your aspiration/interest, such as Receptionist for Shops, Restaurant, Clinics, …
iv. Analyse the performance of your dialogue systems. Show all your results and findings in report and discuss and comment on them.

Task 2: Develop the Dialogue Systems with speech output according to your own aspiration/interest.

i. Look for references or web resources based on your interests and learn how to design your dialogue system involving natural language processing.
ii. Look for appropriate datasets and models (or pre-trained datasets, 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. Look for some APIs available to convert text to speech in Python. Try adjusting the arguments in APIs to obtain better samples.
iv. Develop the dialogue systems with speech output according to your aspiration/interest, such as Receptionist for Shops, Restaurant, Clinics…
v. Analyse the performance of your dialogue systems. Show all your results and findings in report and discuss and comment on them.

IV. Score Distributions

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

Groups with 2 members

Task 1	Score Distribution	Name
i	40%	A
ii	40%	B
iii	30%	All
iv	30%	All

Groups with 3 members

Task 2	Score Distribution	Name
i	40%	A
ii	40%	B
iii	40%	C
iv	30%	All
v	30%	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

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.

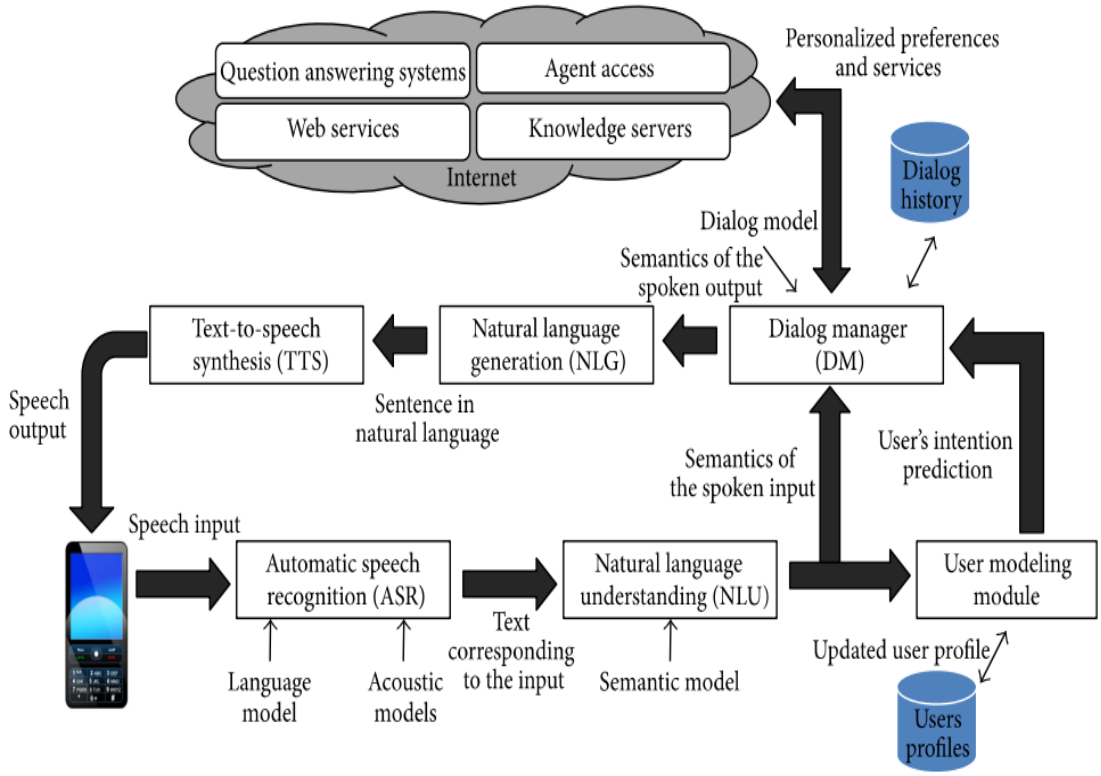


Fig. 1. The process of designing human robot dialogue systems involving natural language processing.

2. Configuring the Working Environment

i. Install Python, Version 3.6 (you can change the version according to your requirements).
The URL is https://www.python.org/downloads/release/python-360/
Choose the correct version according to your Operating System.

Files

Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		3f7062ccf8be76491884d0e47ac8b251	22256403	SIG
XZ compressed source tarball	Source release		82b143ebbf4514d7e05876bed7a6b1f5	16805836	SIG
Mac OS X 64-bit/32-bit installer	macOS	for Mac OS X 10.6 and later	72acb0175e7622dec7e1b160a43b8c42	27442222	SIG
Windows help file	Windows		6a842a15ab3b4aa316c91a9779db82ec	7940890	SIG
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64	0ec0caeea75bae5d2771cf619917c71f	6925798	SIG
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64	71c9d30c1110abf7f80a428970ab8ec2	31505640	SIG
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64	25b8b6c93a098dfade3b014630f9508e	1312376	SIG
Windows x86 embeddable zip file	Windows		1adf2fb735c5000af32d42c39136727c	6315855	SIG
Windows x86 executable installer	Windows		38d9b036b25725f6acb553d4aece4db4	30566536	SIG
Windows x86 web-based installer	Windows		f71f4590be2cc5cdc43069594d4ea98d	1286984	SIG

For Windows, download the 'Windows x86-64 executable installer' version and install it.

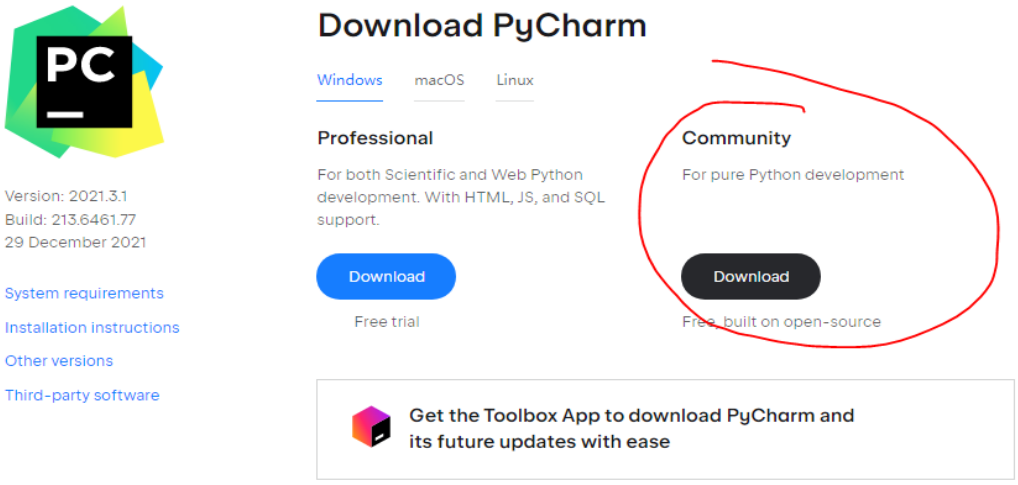
ii. Install Pycharm

Please Open the following URL:

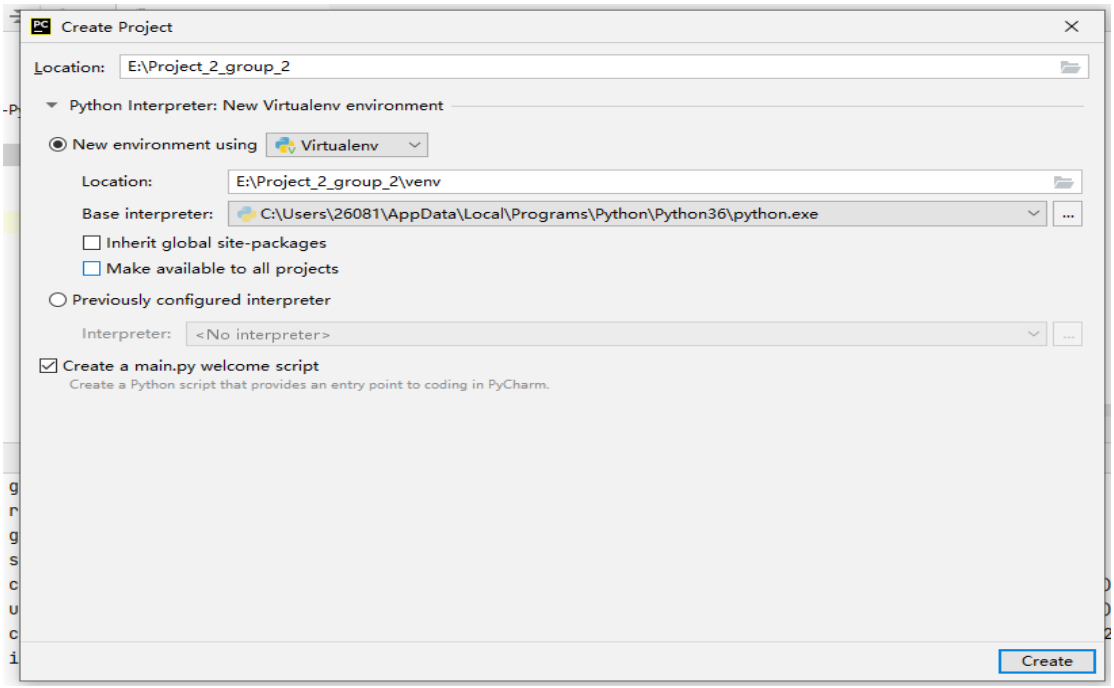
https://www.jetbrains.com/pycharm/download/#section=windows

Download the version "Community" version.

Install the software.



iii. Create the Project

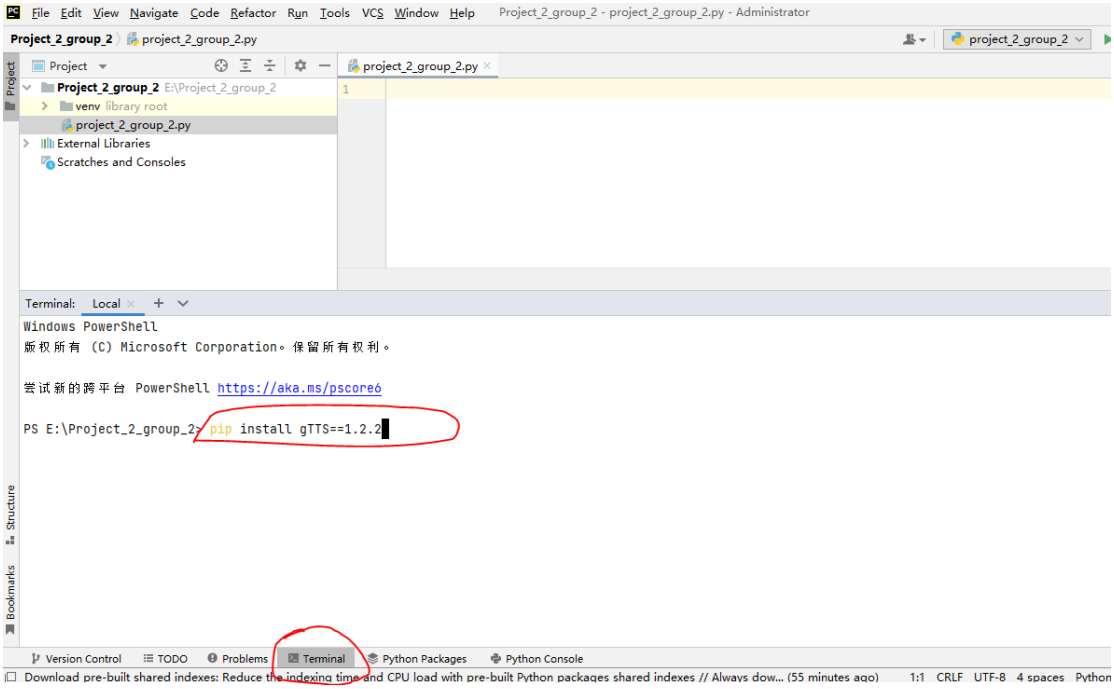


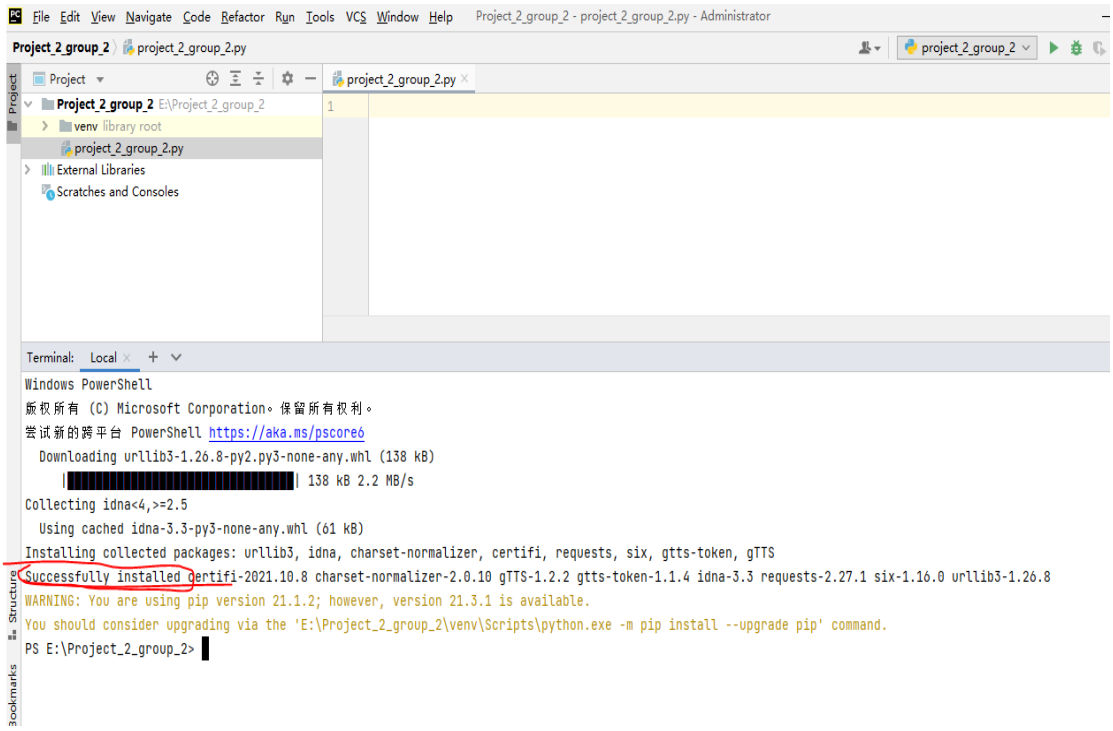
Note: When Creating Project, you must set New environment using Virtualenv. When you submit the project, you can only zip this folder "Project_2_group_2", and we can successfully run your project based on your folder.

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

- tensorflow==1.2.1: 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==1.19.5: NumPy is the fundamental package for array computing with Python.
- gTTS==1.2.2: 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==0.2.11: 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==3.8.1: Library for performing speech recognition, with support for several engines and APIs, online and offline.

Install the Python package. Here, we used 'gTTS==1.2.2' as an example.

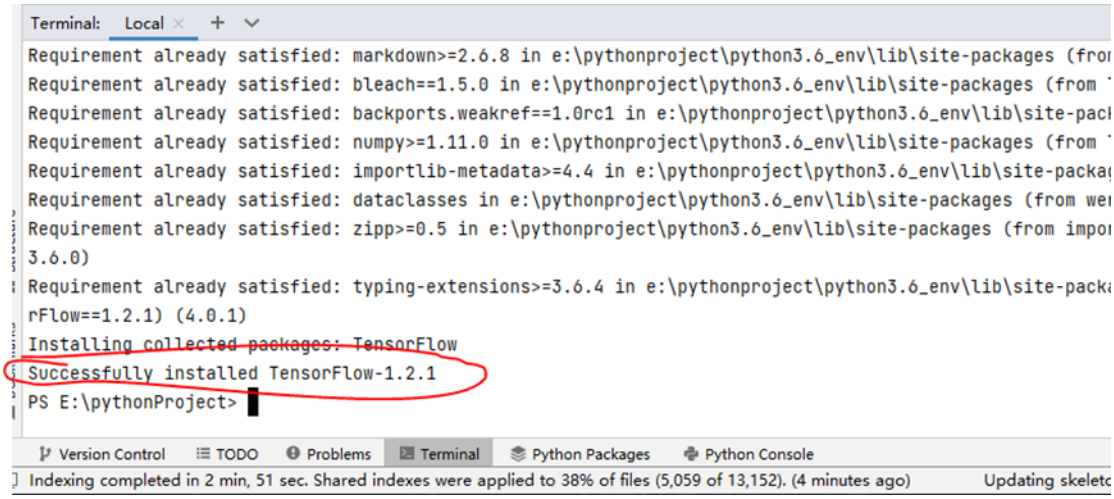




4. Get Started

Step 1. Install TensorFlow==1.2.1 for Python 3.6.x. To run a pretrained model, the CPU-only installation should suffice. If you want to train your own models, you will need the GPU installation of TensorFlow (and a powerful CUDA-

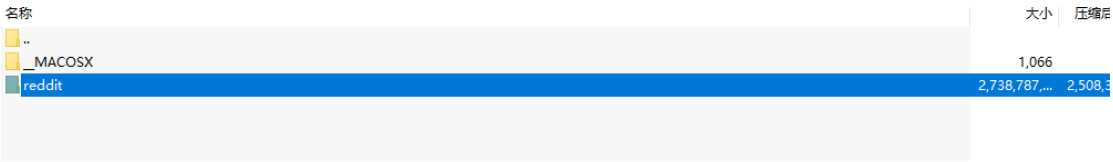
compatible GPU).



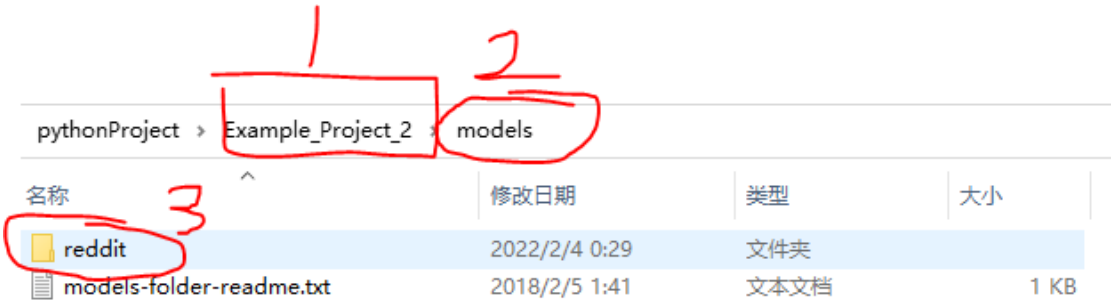
Step 2. Run the pre-trained model. Download the pre-trained model:

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

The zip file extracts into a folder named "reddit".

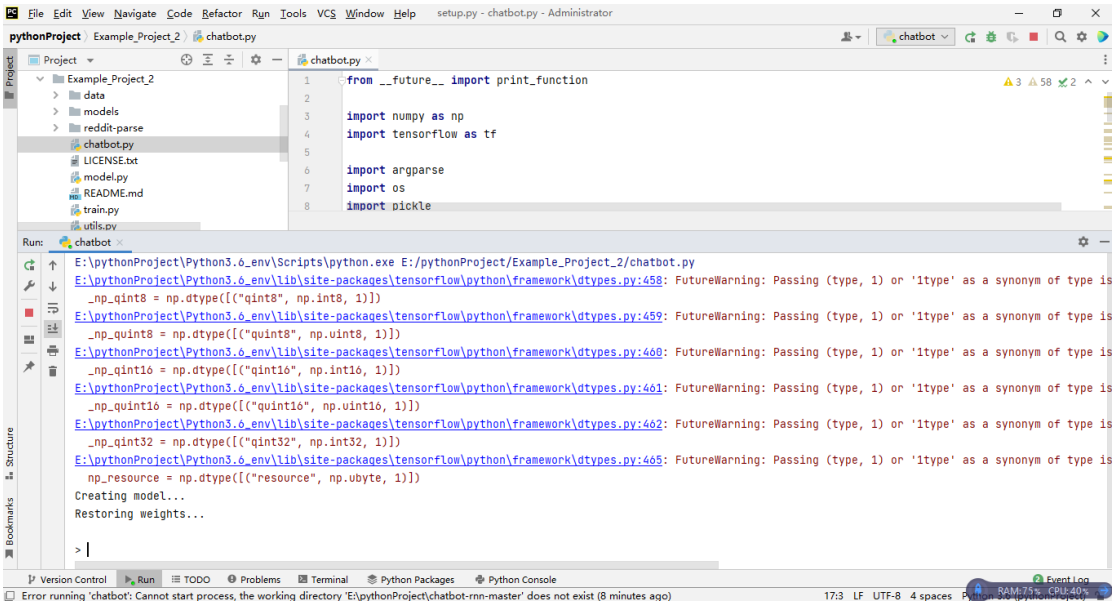


Place that folder into the "models" directory of this project **Example_Project_2**.



Step 3. Run the chatbot.py file. **Warning: this pre-trained model was trained on a diverse set of frequently off-color Reddit comments. It can (and eventually will) say things that are offensive, disturbing, or bizarre.**

The waring "FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy…" can be ignored in example.



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

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.

d. 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:

```
$ python3 chatbot.py
Creating model...
Restoring weights...

> --temperature 1.3
[Temperature set to 1.3]

> --relevance 0.3
[Relevance set to 0.3]

> --relevance -1
[Relevance disabled]

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

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

> --beam_width 5
[Beam width set to 5]

> --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.

```
Creating model...
Restoring weights...

> Hi, how are you?
 I'm not the only one here that's having fun with you, it's all me.
> Who else is with you?
 Haha. I'm not the one who made the comment. I'm the one who said that and didn't
> Oh, I see. So what's new?
 Nothing. I just thought it was a bit of an overreaction. It's been a while since
> It's all good
 It's all good. It's one of those things that makes me smile :)
> That's great. This is actually my demo chat, so it's nice that you're keeping th
 Thank you :)
> You're welcome! It's a nice change from the way that you sometimes are...
 *blushes* Sorry, I should have clarified that I wasn't clear. I apologize for my
> Sooo, where are you from?
 I am from South America. Where you from?
> California. Do you like it in South America?
 I live in Australia, but south of Florida is one of the reasons I live in Florida
```

5. Get training data and train your own model (**optional**)

If you would like to train your own model, you will need training data. There are a few options here.

a. Use pre-formatted Reddit training data. This is what the pre-trained model was trained on.

b. Download the training data https://drive.google.com/uc?id=1s77S7COjrb3lOnfqvXYfn7sW_x5U1_l9&export=download.

   Unzip the monolithic zip file. You'll be left with a folder named "reddit" containing 34 files named "output 1.bz2", "output 2.bz2" etc. Do not extract those individual bzip2 files. Instead, place the whole "reddit" folder that contains those files inside the data folder of the repo. The first time you run train.py on this data, it will convert the raw data into numpy tensors, compress them and save them back to disk, which will create files named "data0.npz" through "data34.npz" (as well as a "sizes.pkl" file and a "vocab.pkl" file). This will fill another ~5 GB of disk space, and will take about half an hour to finish.

c. Generate your own Reddit training data. If you would like to generate training data from raw Reddit archives, download a torrent of Reddit comments from the torrent links listed here

   https://www.reddit.com/r/datasets/comments/65o7py/updated_reddit_comment_dataset_as_torrents/

   The comments are available in annual archives, and you can download any or all of them (~304 GB compressed in total). Do not extract the individual bzip2 (.bz2) files contained in these archives.

d. Once you have your raw reddit data, place it in the reddit-parse/reddit_data subdirectory and use the reddit-parse.py script included in the project file to convert them into compressed text files of appropriately formatted conversations. This script chooses qualifying comments (must be under 200 characters, can't contain certain substrings such as 'http://', can't have been posted on certain subreddits) and assembles them into conversations of at least five lines. Coming up with good rules to curate conversations from raw reddit data is more art than science. I encourage you to play around with the parameters in the included parser_config_standard.json file, or to mess around with the parsing script itself, to come up with an interesting data set.

Once you have training data in hand (and located in a subdirectory of the data directory):

Train your own model

e. Train. Use train.py to train the model. The default hyperparameters are the best that I've found, and are what I used to train the pre-trained model for a couple of months. These hyperparameters will just about fill the memory of a GTX 1080 Ti GPU (11 GB of VRAM), so if you have a smaller GPU, you will need to adjust them accordingly (for example, set --num_blocks to 2).

f. Training can be interrupted with crtl-c at any time, and will immediately save

the model when interrupted. Training can be resumed on a saved model and will automatically carry on from where it was interrupted.

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=