# MentorPAL

Blocked due to NDA and ARL Data

Data Preparation

---------------

To obtain transcripts for a particular video, do the following:

0. After the interview is done, watch it fully and note down the start and end timestamps for each question. Format: HH:MM:SS. Download the timestamps sheet and video. Look at existing timestamp sheets for exact format of the sheet.

1. Place the video and the timestamps file inside \*\*recordings/session1/\*\*, \*\*recordings/session2/\*\*, etc.

2. The video must be named as \*\*session1part1.mp4\*\*. The timestamps file must be \*\*session1part1\_timestamps.csv\*\*. The session number and part number will change based on what you have. If a session has only one part, then it will be session3part1.mp4, session3part1\_timestamps.csv. Part numbering begins at 1.

3. Run the command `python src/data\_pipeline/prepare\_data.py /example/path/to/mentor/sessionN` from root directory of the git repository. \*N\* is the session number.

4. The program will first convert the video to audio and create \*\*session1part1.wav.\*\*, etc. based on what the video name is. The audio file has same name, except the extension

5. Then, the program will cut the audio into chunks, based on the timestamps provided. The chunks are stored inside \*\*recordings/session1/audiochunks\*\* and the chunks are named from \*\*a0.wav\*\* to \*\*qm.wav\*\* where \*m\* is the number of questions asked in the session.

6. Once the chunks have been obtained, the program will automatically call the IBM Watson method in `ibm\_stt.py`.

7. The transcript will be automatically generated and stored as \*\*transcript.csv\*\* inside \*\*recordings/session\_n.\*\*

8. Once the transcripts have been generated, put them on Google Sheets and manually edit them to make them coherent.

9. Work on generating the paraphrases for each question. Also assign Tags and Helper Tags to each question.

10. Combine the paraphrases and answers and add them to the \*\*Questions\_Paraphrases\_Answers\*\* sheet inside MentorPAL/Dialogue on Google Drive. Do the same with the prompts and utterances and add them to \*\*Prompts\_Utterances\*\*.

11. Download that sheet and \*\*Prompts\_Utterances\*\* as XLSX and name them \*\*Questions\_Paraphrases\_Answers.xlsx\*\* and \*\*Prompts\_Utterances.xlsx\*\*. Place them inside MentorPAL/data on your local machine.

12. Run `python src/post\_process\_data.py /example/path/to/mentor <start\_session> <end\_session>`. start\_session and end\_session are session numbers. Example, giving 1 4 will make the code run from session1 to session4. Giving 2 2 will make code run only for session2.

13. This will generate \*\*classifier\_data.csv\*\*, \*\*NPCEditor\_data.xlsx\*\* and \*\*metadata.csv\*\*. The first file is for use by the classifier. The second file should be imported into NPCEditor (More on this later) and the third file contains metadata about what was processed. If you have 5 sessions so far and then, you record a sixth session, you don't need to run the post\_process\_data.py from scratch. You can just run python src/post\_process\_data.py /example/path/to/mentor 6 6` and data from the sixth session will be appended to \*\*classifier\_data.csv\*\* and \*\*NPCEditor\_data.xlsx\*\*. \*\*metadata.csv\*\* will also be updated and you can add more new sessions like this. Yould still have to run prepare\_data.py for the new sessions every time and have to do the manual work (paraphrases, tags, manual transcription cleanup) for each session.

14. Now, data for the ensemble classifier is ready and you can proceed to training the classifier and NPCEditor.

\*\*WARNING: Opening any csv/xlsx file downloaded from Google Sheets in Excel will mess up the encoding of text in the files. Google Sheets uses UTF-8 encoding whereas Excel uses Windows-1252. The files must be in UTF-8 only. Always edit the file in Google Sheets and just download it. Never open it on your host machine.\*\*

\*\*If using LibreOffice Calc, the default encoding seems to be UTF-8. Hence, it should be safe to open the csv/xlsx files in LibreOffice\*\*

NPCEditor Setup

---------------

The following steps will setup NPCEditor with the data from \*\*NPCEditor\_data.xlsx\*\*

1. Open NPCEditor (on Mac, it is an application. On Windows/Linux, it is a .jar file which can be opened by just double-clicking it). Make sure you have Java 8 installed.

2. File -> Import -> Excel

![alt text](https://cloud.githubusercontent.com/assets/2927889/26427446/c7e379d4-40aa-11e7-8ece-095f2df2d271.png)

3. Select the Excel file (.xlsx) \*\*NPCEditor\_data.xlsx\*\*.

![alt text](https://cloud.githubusercontent.com/assets/2927889/26427449/cb951740-40aa-11e7-86e2-763599d3c030.png)

4. In a few moments, you should see the questions and answers being loaded into the Utterances tab.

5. Navigate to the People tab and click on 'Add' in the bottom left corner. Name your person as you wish. This is not the name of the mentor because when multiple mentors are added, there will still be only one NPCEditor instance running.

![alt text](https://cloud.githubusercontent.com/assets/2927889/26427459/d024a168-40aa-11e7-8037-55ee2cdc446b.png)

6. Click on the Accounts tab inside the People tab. Create a new account by clicking the 'Add' button below the list of accounts. The account type must be 'Batch Processing'.

![alt text](https://cloud.githubusercontent.com/assets/2927889/26427465/d7f37a22-40aa-11e7-9564-dce78e782596.png)

7. Enter the agent name as the name of the mentor (i.e. 'clintanderson'). If there are multiple mentors, each mentor gets a separate batch processing account. This will help in directing questions to specific mentors if required. Leave the module name as such. Check the 'Connect on startup' option. Click 'Connect'.

![alt text](https://cloud.githubusercontent.com/assets/2927889/26427474/dd82c2cc-40aa-11e7-8d64-ffe59ea891d8.png)

8. Navigate to the Utterances tab. Select all the answers and from the 'Domain' dropdown, select the name of the person you have created earlier in step 5.

![alt text](https://cloud.githubusercontent.com/assets/2927889/26427479/e2f71294-40aa-11e7-857f-2a39b83fa151.png)

9. Leave the questions side untouched.

10. Now, NPCEditor is ready for training.

11. Navigate to the Classifiers tab, select the entry in the first box with Sender=Anybody. Set up the classifier as shown in the image below and click on 'Start Training'.

![alt text](https://cloud.githubusercontent.com/assets/2927889/26427493/f0ec88de-40aa-11e7-8c44-0dcf62dab005.png)

12. Once the training is finished, the details of the classifier will show up in the status box. This indicates that NPCEditor classifier model is ready.

13. Navigate to the 'Conversations' tab and make sure that the Dialog Manager is set to 'Classifier'. XML messages can now be sent to NPCEditor and answers will be returned.

14. Save the session as \*\*train.plist\*\* inside MentorPAL/NPCEditor files.

\*\*npceditor\_interface.py\*\*: Provides an interface to talk to NPCEditor and get answer(s) for question(s) and returns answer(s) along with score(s) and answer ID(s).

Classifier Setup

---------------

The components of the classifier are explained below, with data from \*\*classifier\_data.csv\*\*

\*\*classifier\_preprocess.py\*\*: This file reads the raw text data from \*\*classifier\_data.csv\*\*, converts the questions to feature vectors, obtains the sparse topic vectors and dumps all this data into pickle files for use by the classifier and topic neural network (LSTM) classifier.

\*\*lstm.py\*\*: This file creates the topic LSTM classifier and stores the newly generated topic model in \*\*train\_data/lstm\_topic\_model.h5\*\*. The train and test topic vectors are stored in \*\*train\_data/train\_topic\_vectors.pkl\*\* and \*\*test\_data/test\_topic\_vectors.pkl\*\* respectively.

\*\*logisticregression.py\*\*: This file creates the Logistic Regression classifier and stores the trained models in \*\*train\_data/fused\_model.pkl\*\* and \*\*train\_data/unfused\_model.pkl\*\*, which are the models with the topic vectors and without the topic vectors respectively.

\*\*classify.py\*\*: This file handles the above three files. It contains master methods which explicitly preprocess the data for the classifier (by calling methods in classifier\_preprocess.py), train the topic LSTM (by calling methods in lstm.py) and train the classifier (by calling methods in logisticregression.py).

Ensemble Classifier

-----------------

\*\*IMPORTANT: Make sure to download the Google News Vectors from <a href="https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit">here</a> and place it in the same folder as MentorPAL (the git repo). The git repo and GoogleNews-vectors-negative300.bin must be siblings.\*\*

A sample program \*\*src/classifier/run.py\*\* is provided to demonstrate how to use the ensemble classifier.

The program \*\*src/classifier/ensemble.py\*\* handles the ensemble. Specifically, when it receives a question, it sends it to \*\*src/classifier/npceditor\_interface.py\*\* and \*\*src/classifier/classify.py\*\* to get answers from NPCEditor and the classifier. This program has the rules to decide which answer to return. The answer is returned to run.py or whichever interface you call this program from.

The Google News Vectors bin file is around 3.6 GB in size and will be loaded every time a new instance of ensemble.py is created. ensemble.py will have the ability to handle session after session and will keep track of session activity. A new instance will be needed only when the program is started from scratch. When multiple users interact with the system one after the other, they will use the same instance of ensemble.py. ensemble.py has the ability to start a new user session, which is different from a new instance of the program itself.

System Setup

------------

The following are required to clone the project from git and run:

1. python 3.5.3 (python 2.7 is no longer supported)

2. git

3. <a href="https://git-lfs.github.com/">git-lfs</a>. Track .pkl and .json files when using git-lfs.

4. ffmpeg (make sure to install it with --with-theora, --with-libvorbis, --with-opus when using brew on macOS. When compiling from source code, use --enable-theora, --enable-libvorbis, --enable-libopus)

5. <a href="activemq.apache.org">Apache Activemq</a>

6. Java 8

Python Dependencies

------------

The following packages are required for the program to run. All these can be installed from pip3 or easy\_install3.

1. \*\*watson\_developer\_cloud\*\* : IBM Watson's Python wrapper to communicate with their STT service.

2. \*\*ffmpy\*\* : A wrapper for FFmpeg, which is used to convert video to audio and also cut the audio file into chunks.

3. \*\*keras\*\* : Neural network library

4. \*\*scikit-learn\*\* : Python Machine Learning library

5. \*\*numpy\*\* : Python scientific computing

6. \*\*pandas\*\* : High-performance data structures for Python

7. \*\*gensim\*\* : Topic Modelling - needed for word2vec

8. \*\*openpyxl\*\* : Pandas dependency for writing to excel files

10. \*\*tensorflow\*\* : Deep learning library

11. \*\*nltk\*\* : Natural Language Toolkit

12. \*\*h5py\*\* : Python interface to HDF5 binary format

Windows Setup

------------

1. Get Python 3.5.3 and install TensorFlow from <a href="http://www.lfd.uci.edu/~gohlke/pythonlibs/#tensorflow">here</a>.

2. Install Visual C++ Redistributable 2015 x64 as MSVCP140.dll

3. To install keras and gensim, you have to first install scipy from <a href="http://www.lfd.uci.edu/~gohlke/pythonlibs/#scipy">here</a>.

4. Install numpy+mkl from <a href="http://www.lfd.uci.edu/~gohlke/pythonlibs/#numpy">here</a>.

5. If you are unable to install scikit-learn using pip in Python3, install it from <a href="http://www.lfd.uci.edu/~gohlke/pythonlibs/#scikit-learn">here</a>