General organizational information:

* The scripts are in EMS-pipeline/ETC/Performance Evaluation
* When I run the pipeline, I usually copy all my results into a Google Sheets file to keep everything together. So when I do the below steps, I have copied the results from the pipeline/spreadsheet into Google Sheets first.

Troubleshooting notes:

* As the pipeline and its output is updated, the pre-processing of each piece of information has to change as well. Usually this involves adding or subtracting a newline or space or taking another set of parentheses into account. I usually print the output as I’m trying to fix the parsing until it looks like each piece of information (concept, list of concepts, etc) is in its own discrete index of the array

To evaluate pipeline performance:

* In the terminal, run “jupyter notebook” and navigate to the above file
* For interventions:
  1. Copy the interventions results for a set of data into a text file. In “interventions.ipynb”, modify the variable “path\_results” at the top of main() to be the full path to this file.
  2. Your ground truth for this set of data should also be in a text file. The variable “path\_truth” in main() should reflect this file path as well.
  3. Set the variable “cases” to be the number of scenarios you are testing at once. For example, if you are evaluating the results from 10 different audio recordings, it would be 10.
  4. Run the program. You can choose to display precision, recall, and/or f1 score at once. For each of these, copy the printed results into wherever you are aggregating your data:
     + For precision: Uncomment “print(calc\_precision(truth[i], results[i]))”
     + For recall: Uncomment “print(calc\_recall(truth[i], results[i]))”
     + For f1 score: Uncomment “print(calc\_f1(truth[i], results[i]))”
* For concepts:
  1. Copy the concepts for a set of data into a text file. In “concepts.ipynb”, modify the variable “path\_results” at the top of main() to be the full path to this file.
  2. Your ground truth for this set of data should also be in a text file. The variable “path\_truth” in main() should reflect this file path as well.
  3. Set the variable “cases” to be the number of scenarios you are testing at once. For example, if you are evaluating the results from 10 different audio recordings, it would be 10.
  4. Decide what you consider relevant for concept evaluation; you will use a different function to process the concepts depending on what you want. I tend to use process\_custom. You will uncomment ONLY the lines with the appropriate processing under the headers “process truth options” and “process results options” in main().
     + process\_all\_values: You want to look at concept, negation, and value/trigger text for EVERY concept, not just numerical ones
     + process\_tf: You want to look ONLY at concept name and negation
     + process\_custom: Looks at concept name and negation for every concept, but only checks value for numerical concepts
  5. Run the program. You can choose to display precision, recall, and/or f1 score at once. For each of these, copy the printed results into wherever you are aggregating your data:
     + For precision: Uncomment “print(calc\_precision(truth[i], results[i]))”
     + For recall: Uncomment “print(calc\_recall(truth[i], results[i]))”
     + For f1 score: Uncomment “print(calc\_f1(truth[i], results[i]))”
* For audio (WER and Accuracy):
  1. Copy the text transcriptions into a text file. If you are processing more than one text chunk at once for any given file, there will be newlines between pieces of the same file, and to keep these from throwing off the alignment with the golden transcripts, you will have to preprocess the text file.
     + There is also a file called process\_transcripts.ipynb. This turns a single file with n transcripts into n files with one transcript each. Change the filepath and the variable “newfile” to specify location and naming convention of each file. It isn’t necessary to do this to run multiple evaluations; one file with all the recordings can still be evaluated.
     + There is a file called CondenseTranscripts.ipynb that you can call with the appropriate path and file name that should remove unnecessary newlines. It is not currently in the repo as of 8/4/2020, which means it’s probably locally on the computer at the Link Lab. See if you can access that if you need it. If that still doesn’t work, you can manually remove newlines if you have a small amount of information.
  2. You may have to modify the naming convention and path names in main() to make the program work. You can choose to display WER, Accuracy, or both; but note that you MUST at least run the calculations for WER in the same running as Accuracy because Accuracy depends on WER.