UNHCR Handoff Notes
CS50 Stanford
6/8/2018

Obtaining the Code
The CS50 Team will send a .zip file titled “CS50_UNHCR.zip” containing all code used to run our milestones. After unzipping the contents of that, please refer to the files inside of it for the remainder of these notes.

Audio Transcription Code
Our Audio Transcription process was done through the use of Google Cloud Speech-To-Text, leveraging the fact that Google gives $300 in free credits for every new account created to help this task have no monetary cost.

Transcriptions have already been provided to the UNHCR in previous emails, but in case further transcription from MP3 -> Text is needed, follow these procedures, as guided by the instructions at https://cloud.google.com/speech-to-text/docs/reference/libraries#client-libraries-install-python:

1. Create a Google Cloud Account
2. Install the Google Cloud SDK: https://cloud.google.com/sdk/downloads
3. Open a Terminal window
4. Set your Google Cloud Account and Google Cloud Project Name with:

   gcloud init

5. Run the following command to install Google-Cloud-Speech:

   pip install –upgrade google-cloud-speech

6. Follow the instructions here under “Setting up authentication” using the Command Line: https://cloud.google.com/speech-to-text/docs/reference/libraries#client-libraries-install-python
   a. Note: When it comes time to add your GOOGLE_APPLICATION_CREDENTIALS, so that you don’t have to type that line every single time you open Terminal, do this instead:

      Open your ‘~/.bash_profile’ and add the following on a new line:

      export GOOGLE_APPLICATION_CREDENTIALS=”[PATH]”

      where [PATH] is the global path to the JSON file containing your service account key, as stated by the tutorial linked in this step.

7. Enable data logging in your Google Cloud Project by following: https://cloud.google.com/speech-to-text/docs/enable-data-logging

8. Because the audio files are in MP3, we need to convert them to FLAC first before we can run Google Cloud Speech on them.

9. Navigate to the "Transcription" folder and create a directory for storing the FLAC transcriptions.

10. Update the directory paths in convert.py to reflect the directory locations of your MP3 files and where you want to store the FLAC transcriptions.

11. Run: *python convert.py*

    This will create copies of each MP3 file in FLAC format in the directory you specified.

12. Next, we need to upload these to Google Cloud Storage. Follow the instructions here to create a GCS Bucket: https://cloud.google.com/storage/docs/creating-buckets

13. Use this command, as followed by https://cloud.google.com/storage/docs/gsutil/commands/cp, to upload all FLAC files to your GCS Bucket:

    gsutil -m cp -r [directory of FLAC files] gs://my-bucket

14. After replacing necessary file paths in transcribe.py and creating a directory for storing Transcripts, run:

    python transcribe.py

15. Done!

Topic Modeling Code

Our topic modeling code can be found in the Topic_Modeling folder. Because our topic modeling efforts were subpar, we do not have a single file that contains our best approach. We have included all code we wrote for this task.

Overview of Topic Modeling Approaches

gensim_topic_modeling.py

This file uses the gensim topic modeling package to run unsupervised LDA on the text. The LDA Model passes over the data multiple times and tries to find a certain number of keywords pertaining to unsupervised generated topics.

Run: *python gensim_topic_modeling.py*

LDA.py
This file uses the standard LDA package commonly referenced when we did our initial research with LDA. The package uses a similar approach to the gensim topic modeling file, and also does unsupervised unguided LDA on the call transcriptions.

Run: *python LDA.py*

topic_modelr.py
topic_modelr.py uses sklearn package to run topic modeling on text. This file was our most promising approach since it allowed us to do n-gram keyword topic-modeling (so instead of just looking for individual keywords per topic, it looked for phrases). Please follow the instructions here for how to run topic_modelr.py:
https://github.com/jmausolf/Python_Tutorials/tree/master/Topic_Models_for_Text

guidedLDA-notebook.ipynb
This is the Guided LDA work done by our team. Guided LDA lets us choose the topics/keywords instead of having the LDA model choose the topics for us. There is a section with the topics that we seeded, but those can be changed as long as the words are present in the text (it will throw an error if the word is not there). The model was created following this tutorial: https://medium.freecodecamp.org/how-we-changed-unsupervised-lda-to-semi-supervised-guidedlda-e36a95f3a164

In order to run this, you need to be able to run Jupyter Notebooks on your machine. Please follow the instructions here to do so:
http://jupyter.readthedocs.io/en/latest/install.html

After installing, simply doing “jupyter notebook” in the “Topic_Modeling” directory will start the notebook, then open guidedLDA-notebook.ipynb in the screen that pops up.

To run any cell, simply do “Shift + Enter”

Topic Modeling Results
As reported in prior calls, our topic modeling efforts yielded no valuable results as of now due to the limited timeframe we had after call transcription. However, we do have some pain points we have identified for future work:

Differentiating between Callers: Our topic modeling largely struggled because it kept picking up words from the UNHCR script. One solution is to find a way to either break apart who is saying what during the Transcription phase, or to find a way to ignore words from that script during LDA processing.

Picking up Topics with LDA: As stated before, LDA is an unsupervised algorithm, meaning that it is hard to control what it will pick up as topics. We suggest going with

the Guided LDA approach since it at least allows the user to seed the LDA model with initial topics.

Sentiment Analysis Overview
Basic sentiment analysis was conducted on the call text by running a model trained on the IMDB movie review corpus. Our initial results found that most calls came back as neutral sentiment. The code for this will be sent later once we consolidate it from the team members.

In conducting Sentiment Analysis, the team realized that it is best if we can run a model that is trained on a subset of our calls so that it is more fitted towards positive/negativeness in the scope of UN Calls, not movie reviews. This was something we did not have time for due to the fact that it required we label hundreds/thousands of calls as positive or negative.