0. Name of the group members and CNetID

- Christi Liongson (`christiannenic`), Kenny Shaevel (`kshaevel`), and Charlie Sheils (`casheils`)
- Git-repo: `casheils-christiannenic-kshaevel`

1. A brief overview of the final project (200 words maximum)

This project analyzes cable news transcripts from CNN, Fox, and MSNBC from January 1st, 2020 to the present. We were curious to know which speakers appeared most often and on the widest variety of different shows and networks, as well as how much speaking time they were given.

To do this, we thus created an extensive web-scraping operation to collect textual information of each episode, most importantly the text of the transcripts attributed with their respective speaker.

The Django interface allows users to input a speaker name (required) and speaker title, show name, network, and search terms to generate summary visualizations and a topic analysis (using TF-IDF modeling) that displays top tokens and sentences.

2. The overall structure of the software (1-page maximum)

Installation

All required libraries to run the full program (except selenium, see below) can be found in the file `requirements.txt` and can be installed from the command line using _, ideally in a virtual environment setup.

The one exception is the Firefox webdriver for the selenium package (for use in web-scraping Fox News), which must be downloaded from https://github.com/mozilla/geckodriver/releases and placed in the user bin.

Web scraper

For the web-scraping operation (built by Charlie), the database into which we put the transcript information we web-scraped is called `news_db_2020.sqlite3`, and the schema for this database can be found in `news.sql`. The file `run_crawlers.py` can be run from the terminal with no arguments. This file runs all associated python scripts `crawler_fox.py`, `crawler_cnn.py`, and `crawler_msnbc.py` that web-scrape from Fox, CNN, and MSNBC, and place the show, episode, speaker, and transcript information into the database.

To run the full crawler, type `python run_crawlers.py` in the terminal.

A limited version of the web-scraping operation, which just takes the first show from each network and crawls the transcripts from 2/15/2020 to 3/14/2020, can be run by typing `python run_crawlers_limited.py` in the terminal. This references the modified crawlers

`crawler_fox_limited.py`, `crawler_cnn_limited.py`, and `crawler_msnbc_limited.py`.

The file `crawler_util.py` contains functions common to each of the six aforementioned crawlers. It contains functions that add newline characters to each break in the raw transcript text, and later that assign each chunk of text to the speaker at the beginning of them, using a combination of text and regular expression operations. For the most part, within a news transcript, the first time a name appears, it appears as "<FULL NAME>, <SPEAKER TITLE/DESCRIPTION>", and subsequently the name appears as "<LAST_NAME>". Thus when attributing a quote to a speaker when inserting into the database, all subsequent quotes from the same speaker should be attributed to the same speaker name (and ID, for database purposes). The functions in this file resolve conflicts, particularly when two speakers in the transcript have the same last name. As well, in most transcripts, video and audio clips are noted by (BEGIN VIDEO CLIP) and (END VIDEO CLIP), or something similar. We removed these from the text of the transcript since we were mainly interested in speakers who had actually appeared on the show intentionally.

User Interface

The user interface is built using Django 2.0.2 (Christi) and allows users to query the database using a form and perform textual transcript analysis and view summary graphical information through the respective `analyze.py` and `visualize.py` scripts. Top tokens and sentences will appear as cards on the interface and data visualizations will open in a new window. Data visualizations will need to be closed in order for the user interface to finish loading.

To run the web interface and access the topic modeling and visualizations, navigate to `final_project/ui` and run the command `python manage.py runserver`.

Topic Modeling

The topic modeling (Christi) script in `analyze.py` takes a data frame of transcript data and applies sci-kit learn's TF-IDF vectorizer to identify top tokens. Before extracting top tokens, the transcript data is preprocessed to combine transcript blocks into each episode, remove stopwords utilizing nltk's stopwords corpora and manual additions, and tokenize. Top tokens per episode are identified with the highest modeled TF-IDF score.

Sentences are also broken down and further tokenized using nltk's `sent_tokenize`. Sentences are ranked using the mean modeled TF-IDF scores. Sentences with fewer than three words are assumed to be greetings or short asides and removed from the analysis.

Finally, episodes that have identified less than ten top tokens and three top sentences are assumed to lack substantial information or spillover from a previous episode. Those episodes are removed from the topic modeling output but may be included in other analysis.

Visualizations

The visualization module (Kenny) is in visualize.py and has two components. The first is dynamic: based on data entered through the Django interface, a graph showing the most common shows the speaker has appeared on (color-coded by network) is generated. The data from Django is passed as a dictionary to `retrieve_data`, which constructs a query to our database and turns the result into a Pandas dataframe. This dataframe is sent to `speaker_summary` through which the visual is built using `matplotlib`.

The second component is a static overall summary analysis for display on the Django webpage. The first analysis is most frequent speakers on each network with hosts/anchors excluded, to give a sense of the most impactful personalities on each network. The second analysis is on verbosity: how many words are spoken each time someone speaks. We compare the verbosity across networks, across shows, and across speakers using the same flow as for the dynamic visual. These are done using `most_frequent_speakers` and `most_verbose` respectively.

3. What the project tried to accomplish and what it actually accomplished (200 words)

What we Learned

After attempting to crawl CNN, MSNBC, and Fox transcript websites and populate a database, we found the three websites to be substantially different in terms of content layout and "quirks" in the interaction with their website. The main reason for this hurdle was that much of the content was a "rush transcript" and contained many typos and inconsistencies of dates, names, and other transcript markers such as commercial breaks and video clips. As a result, many manual changes needed to be put in place in order to handle these inconsistencies.

Sci-kit learn, while a very robust machine learning tool for python and is effective at calculating TF-IDF scores for many episodes, can take a long time to run. However, we've found that sci-kit learn is more efficient at conducting TF-IDF analysis versus the loops and dictionaries created in a previous programming assignment. The final *analyze.py* utilizes Pandas data frames and apply functions to minimize runtime.

Next Steps

With additional time, expansions for this project include:

1. Crawling previous years and adding them to the database. While this would allow us to gather a more substantial corpus for speaker analysis, this would require additional manual fixes to typos and inconsistencies that the current web crawler has not yet encountered.
2. Associating speakers with the database through a Jaro-Winkler distance algorithm instead of manual associations. This would require additional training to find the optimal thresholds for speaker name matching.
3. Speed efficiency in textual analysis.
4. User interface adjustments, such as in-line display of data visualizations and dynamic drop-down options for speaker selection
5. Additional search and filter functionalities: view, filter, and report by show and network.