Identifying Ads in TV News Closed Captions

By Linnea Hartsuyker

Problem Statement

TV News closed captioning captions the ads as well as the news itself. This closed caption text is a rich source of information about what news stories are being covered and what language is used to discuss them, but when it's contaminated with ads, it's harder to extract information about the news itself.

On the other hand, knowing who advertises on which show is also useful information for media watchdog groups, so being able to identify ads is an important processing step in doing natural language processing on TV News closed captions

Data

The data will be downloaded from archive.org, which maintains a huge repository of TV news closed captions.

Data Gathering

I modified the download scripts from https://github.com/notnews/archive_news_cc to help me download TV News Closed Captioning from archive.org. I followed the following steps:

1. Download the list of all available tv shows
2. Use that list to build two lists, one of all FOX News shows, and one of all CNN shows
3. Feed those lists into the scraper that downloads HTML and XML files, one for each show

Data Parsing

I wrote code to parse metadata from the XML files including:

- Contributor: TV affiliate
- Runtime: the length of the program in HH:MM:SS
- Start_time: the datetime when the program started

- End_time: the datetime when the program ended
- Subject: a list of subjects covered in the program

The HTML from each show divides the closed captioning text into 60-second snippets, so I parsed:

- Snip_start: the start of the snippet in seconds since the beginning of the program
- Snip_end: the end of the snippet in seconds since the beginning of the program
- Snippet: the text of the snippet

This is combined with the metadata and output to a CSV of snippets, one for CNN and one for FOX.

Please find the code to do this downloading and parsing here:
https://github.com/LinneaHarts/ad_finder_cc/tree/master/src/data

Data Cleaning and Feature Creation

One of the biggest challenges with NLP classification is coding the text. I was that unsupervised learning could help make the task of coding sentences of closed captioning as ad or news easier.

Before experimenting with vectorizing and clustering, I used NLTK to tokenize the snippets into sentences and combined sentences of fewer than 3 words into the previous sentence. I removed all punctuation and some other special characters.

Then I tried several different approaches to using unsupervised learning to (a) help with hand-coding and (b) create features that could help with machine learning. My process was as follows:

1. Using TFIDF, lemmatize and then vectorize the sentences into words and bi-grams that could be used in a variety of bag-of-words analyses
2. Use Kmeans to cluster the data into 75 clusters. I then printed out representative sentences from these clusters and sorted them into ad, news, or mixed. While this could not replace hand-coding, since the vast majority of sentences sorted into one mixed cluster, I was able to identify ad clusters and news clusters among the other sentences.
3. I used the Kmeans cluster assignment to create a rough logistic regression model.
4. I applied this model to snippets to create a feature called snip_ad, which was 1 if the rough model identified an ad
5. I added features for kmeans clusters and whether I had identified that cluster as ad, news, or mixed

6. Using the topic modeling from before, I added the topic scores to the features, so each sentence is scored by how well it fits into any of 75 topics. While these topics don't sort ads particularly well, I hoped they would be useful features for a supervised machine learning model
7. With visual inspection of the data, I noticed certain words, like "next", "welcome back", "applause" and others might indicate ads in the following or previous rows, so I created features like **next_welcome** and **prev_next** among others to capture the presence of those words in nearby sentences.

After this I hand-coded 10,000 sentences of CNN closed captioning and 10,000 sentences of FOX News closed captioning for use in training supervised machine learning models.

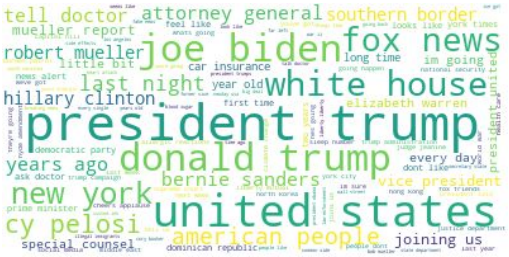
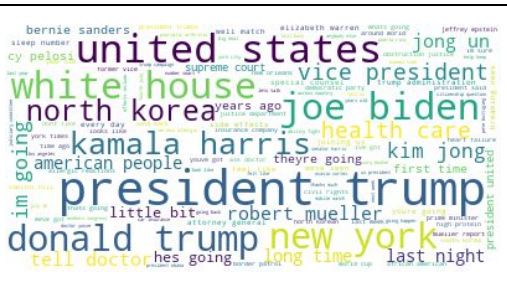
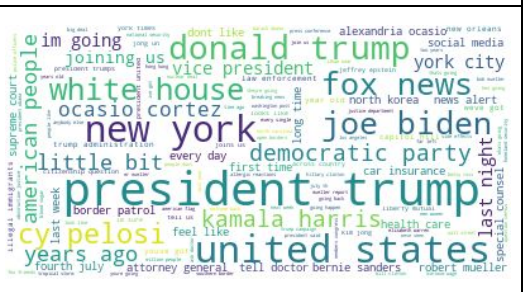
Initial Findings

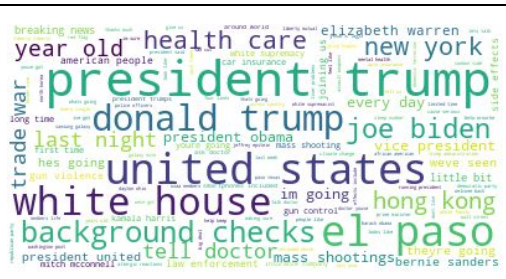
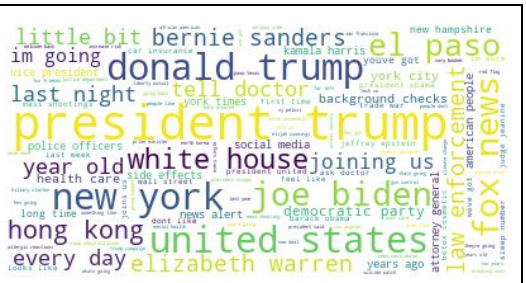
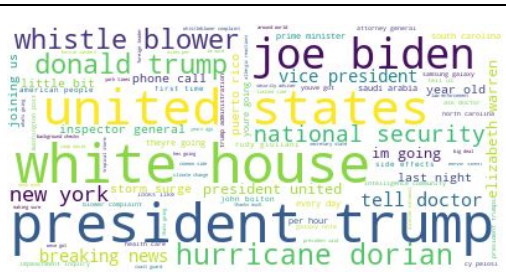
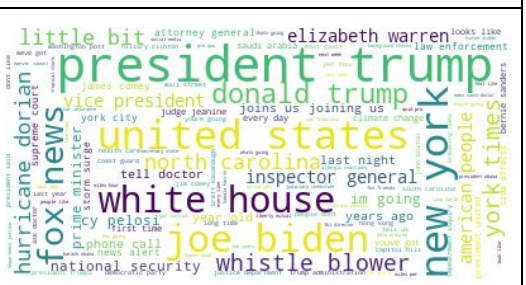
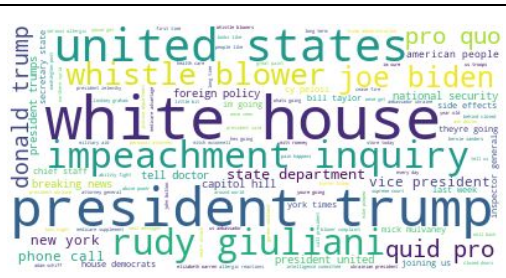
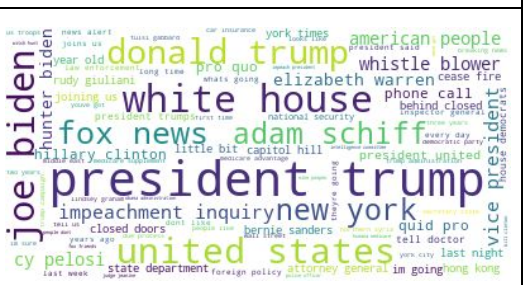
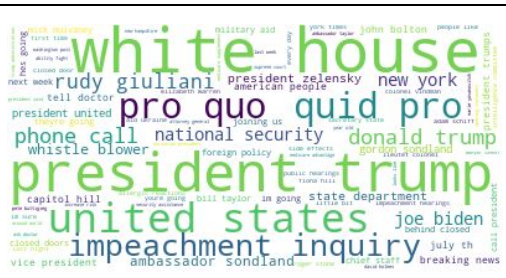
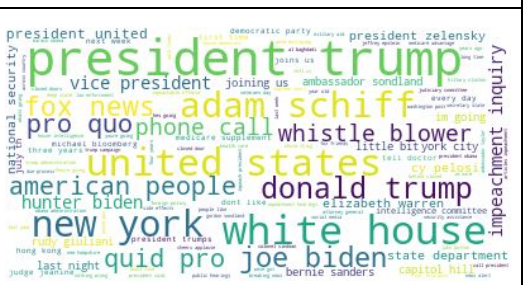
Exploratory Data Analysis

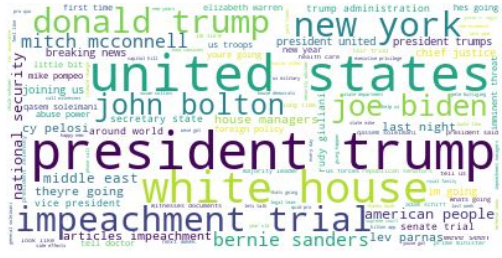
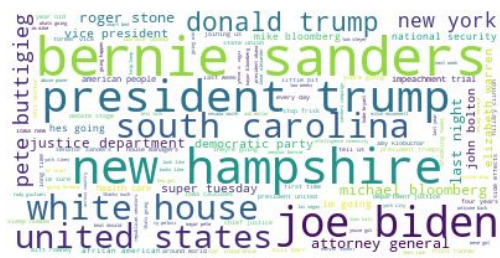
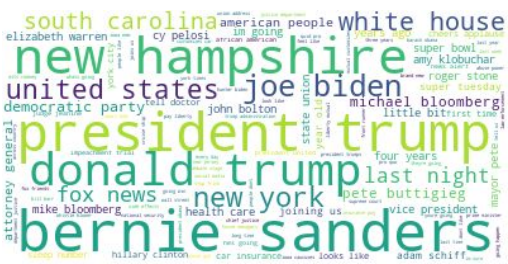
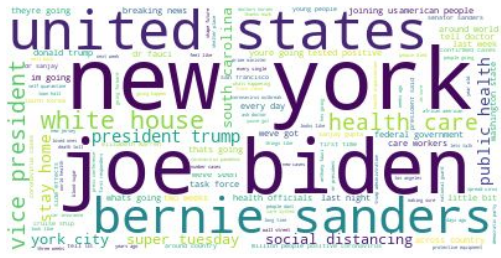
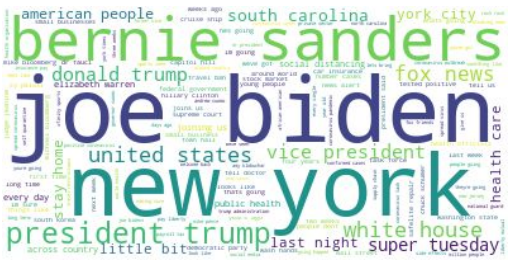
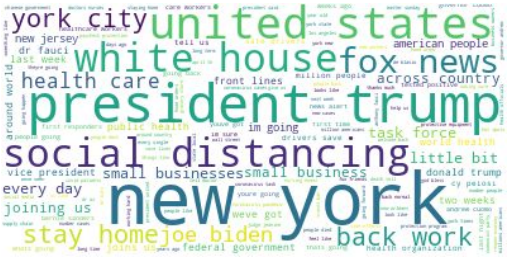
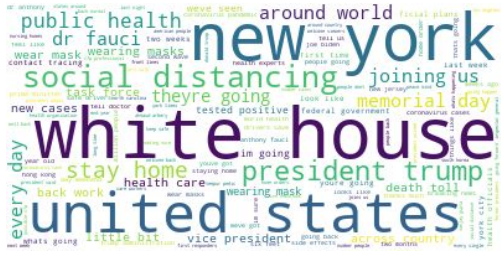
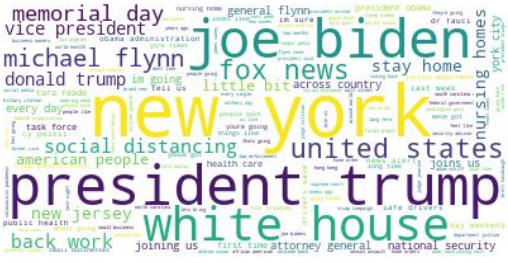
Exploratory analysis can be less than useful for NLP projects, but I still used some tools to visualize what I could.

Word Clouds

I used to explore to visualize the top bi-grams for FOX and CNN over the past year.

Month	**CNN**	**FOX News**
June 2019		
July 2019		

August 2019		
September 2019		
October 2019		
November 2019		
December 2019		

January 2020		
February 2020		
March 2020		
April 2020		
May 2020		

While many months did not have much difference, since keywords from news stories tended to dominate, it's interest to note a few things:

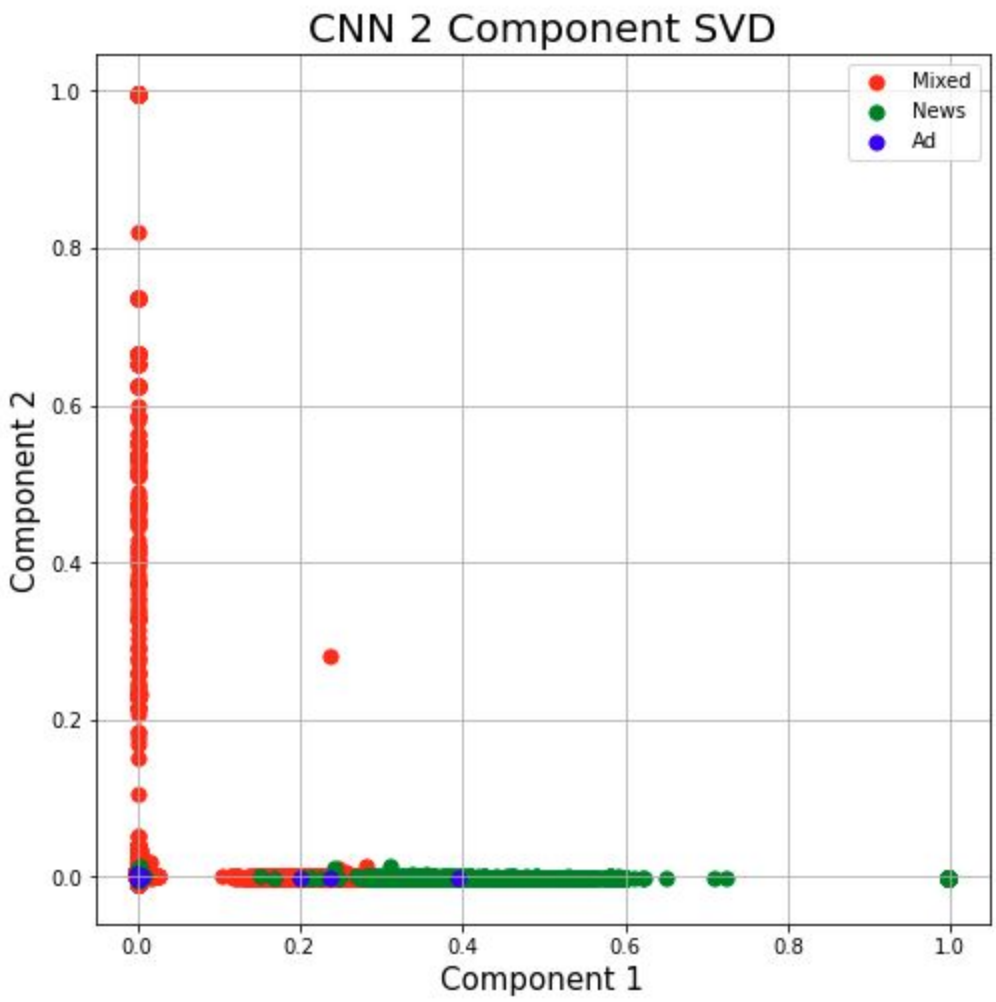
- August 2019 shows "background checks" along with "el paso" for CNN, showing that CNN was talking about solutions to gun violence, while FOX was report on it but not talking solutions

- In January 2020, CNN was talking about John Bolton, who had a tell-all book about Trump coming out, but FOX did not cover it as much
- In May 2020, CNN was still talking about Dr. Fauci, public health, and social distancing since the COVID-19 epidemic was still ongoing, while FOX was pushing stories about Michael Flynn

The code to create the CNN word clouds is here: https://github.com/LinneaHarts/ad_finder_cc/blob/master/notebooks/CNN%20Word%20Clouds.ipynb and the FOX word clouds here: https://github.com/LinneaHarts/ad_finder_cc/blob/master/notebooks/FOX%20Word%20Clouds.ipynb

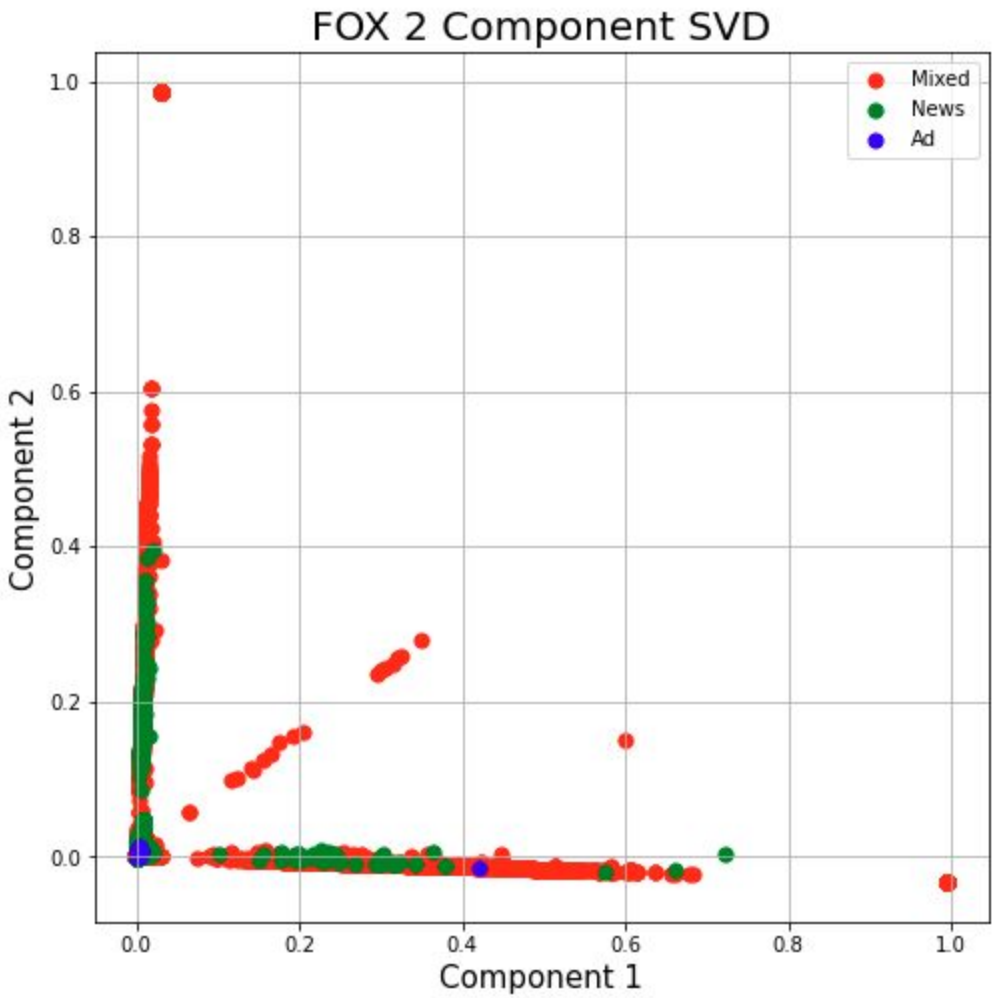
Visualize K-means

I had hoped that doing a dimension reduction on my data and visualizing the clusters might show some interesting groups. I performed Kmeans clustering on CNN sentences and coded the clusters as ad, news, or mixed. I then used TruncatedSVD to collapse the data to two dimensions and plotted it:



This doesn't provide much insight, but it is interesting to note that the news clusters do stay together, even if mixed (which are behind the news clusters) are along both axes.

For completeness, here is the same image for FOX:

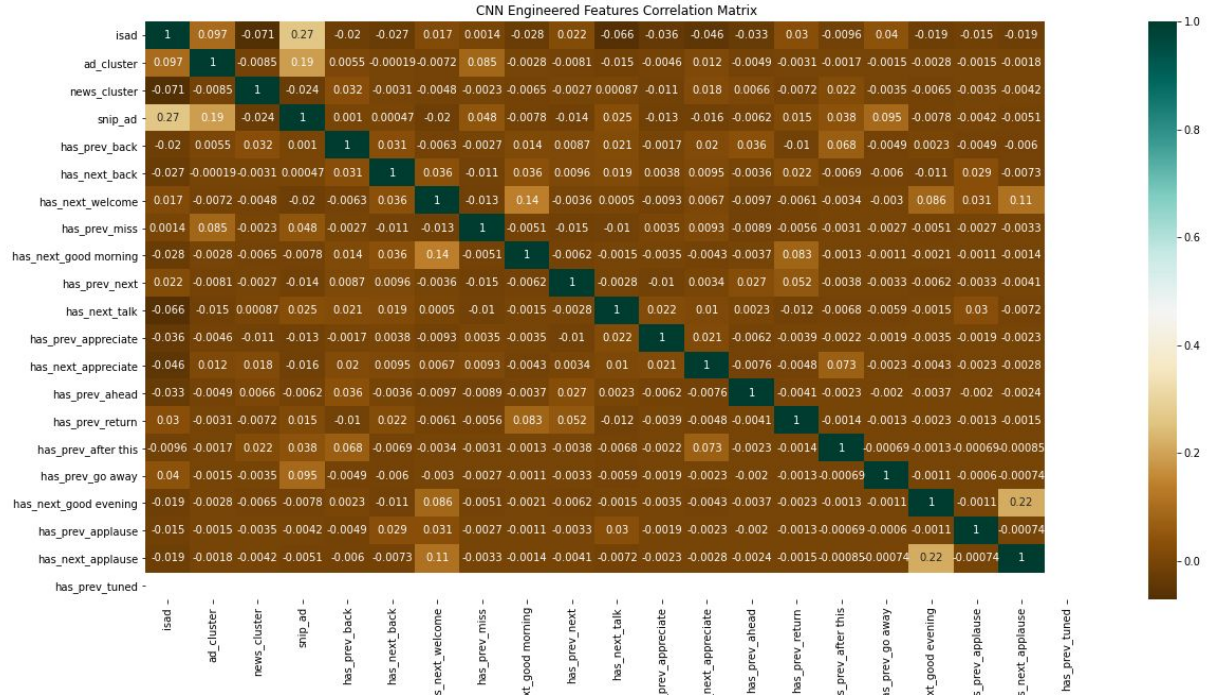


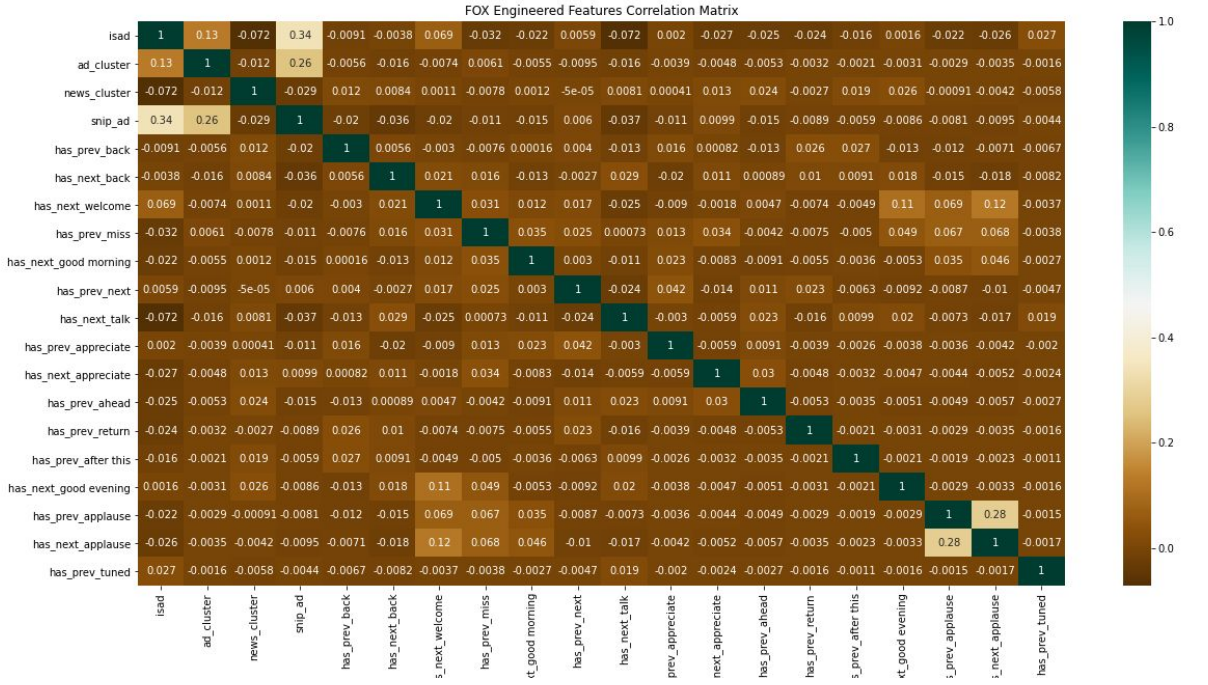
The code for this clustering and dimension reduction can be found here: https://github.com/LinneaHarts/ad_finder_cc/blob/master/notebooks/Clustering%20CNN%20Data.ipynb and https://github.com/LinneaHarts/ad_finder_cc/blob/master/notebooks/Clustering%20FOX%20Data.ipynb

Correlation Matrices

I wanted to get insight into how closely correlated my engineered features were with identified ads and also to see if there were any overlaps that were too strong between topics.

These are correlation matrices for the engineered features, showing that some at least have a good positive correlation with ads being present:





I also examined correlation matrices for all of the topics to see if any stood out as being correlated with ads, or too strongly correlated with each other. Here is an example, but they all show pretty much the same thing: weak correlation with ads and with one another:

See these notebooks for code to create correlation matrices: https://github.com/LinneaHarts/ad_finder_cc/blob/master/notebooks/CNN%20Correlations%20and%20Statistics.ipynb and https://github.com/LinneaHarts/ad_finder_cc/blob/master/notebooks/FOX%20Correlations%20and%20Statistics.ipynb

Next Steps

I now have a Vectorizer that seems to work well, corpuses of coded text, and engineered features. My next steps include:

- Apply and tune supervised machine learning models
- Determine what features help the most with predicting whether or not a sentence is an add
- Apply those models to wild data to see how well they perform in the real world
- Apply the CNN model to FOX and vice versa to see if one can be used to predict the other