Project: NYTimes Topic Trends from 2011 – 2017
By: Michael Du
Objective: The purpose of my project was to track how topic coverage has varied over the years.

Data Scraping:

File: nyt_scraper.py
The original plan was to utilize the NYTimes API to obtain lead sentences from all articles, but following discussion with Debbie, we determined that 250 character length snippets would not be sufficient, despite having over 800 thousand articles across the last 10 years. As a result, I built a scraper to retrieve full articles. There was around 3700 articles per month on average, for a total of 268,000 articles that I scraped across 3 EC2. The data was stored in CSV format.

Data Modeling with LSI:

File: NYTLSA.ipynb
I began modeling the data in the NYTSVM notebook, where I parsed the merged csv file (containing all 268k articles) into a dataframe. I then took a **random** subset of 20k articles from each year, for 140,000 articles, and used TFIDF vectorizer. This was chosen because I think the methodology is great for drawing distinctions between articles. After some gensim conversion, I applied **LSI** modeling and generated an elbow plot with 150 topics, so I could visualize how many topics would be ideal moving forward.
Although 10 seemed to cover most of the variance, I didn't think it would hurt to go with 50, since I could capture additional noise/variance. Since my next step involved clustering, I didn't see the need to narrow the topics down at this point.

At this point, I created a MatrixSimilarity, and was able to pull similar articles so that I could build a recommendation system around it, but this was generally useless for my case. The code is still available in the notebook, but commented out.

K-Means

With the LSI transformed set of data, I first created a silhouette plot for up to 30 clusters, and once I noticed the score would keep going up, I decided to settle with 20, just because I felt 20 would be near the limits of making visual sense (in terms of plotting charts for visuals).

Retro Fitting the Data

The step after this was to fit individual year's data back into this general model, so I could get an accurate breakdown of the number of articles that are categorized in each of the 20 clusters. This taking an individual year's articles and:

- Converting with the pre-fitted **TFIDF vectorizor** (on the entire year's data),
- Transforming with the pre-trained **LSI model** (50 terms) and then
- Predicting the clusters with the pre-trained **KMeans cluster**(20 clusters)

This effectively gave me a total counts of articles per category, per year.

Cluster Definition

The problem with this approach is that each cluster was defined by the set of reduced features that itself was messy to interpret, due to the nature of LSA feature reduction. Negative values and repeat terms meant I could not look at the topics themselves to figure out what belonged in the cluster. However, I was able to manually sift through random samples of each cluster to see how articles were related. I would say 3/4 were very clearly defined (basketball, performance art, finance, politics, etc..) but there were a few clusters that were a bit messy, and felt like an amalgamation of leftover articles that didn't belong to any existing cluster but was limited by the number of clusters available. If I were to build upon this project, I would definitely look into the clusters number and go with a higher cluster number.

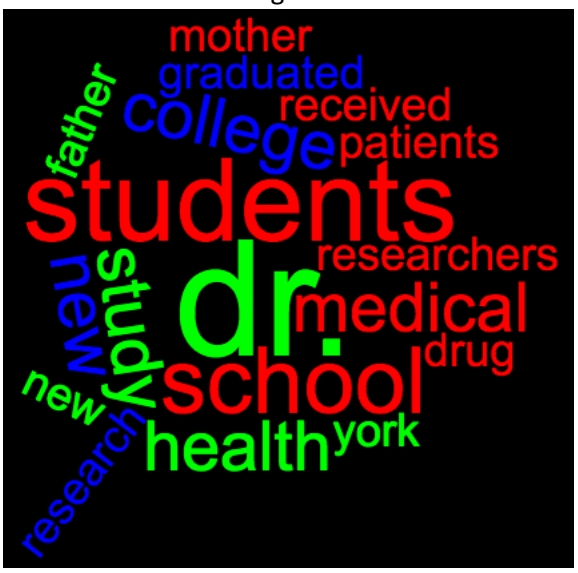
Data Modeling with LDA

File: NYTLDA.ipynb
After consulting with Joe, I determined I wanted better defined topics, and so I went with LDA. The overall process was generally the same, with a few exceptions:

- I used CountVectorizer instead of TFIDF because it's better suited for LDA

- I chose 20 topics instead of 50, because with LDA, the more well defined topics meant I could interpret each feature individually, now that they were sets of distinct words that more or less described an actual news topic
- This also meant I could skip clustering itself

However, the word weights alone didn't necessarily give a clear indication of the topic. For instance, the following word cloud clustered marriage announcements:



As a result, I still manually read articles with highest weights in each category to determine what the topic categorized.

Conclusion: