Summary for Metis Project 4

Using NLP to Explore 'Love' in TED Talks

Sean Davern
Seattle, Fall Cohort, weeks 6.5-8

The Apple Pages version of this document contains playable videos that are meant to be helpful in understanding some of the clustering discussion.

The project's README.md contains a file-by-file explanation of the project organization.

Overview

This summary is intended to provide Metis staff with supporting information about work completed beyond what was described in the presentation slide_deck.

The data analyzed in this project came from the Kaggle TED Talks dataset compiled by Rounak Banik. In particular, the transcripts.csv file was mined for talks containing words stemming from 'love'. The 2,550 talk transcripts contain more than 1,100 that have at least 1 sentence using a love-related word. The transcripts are relatively clean and didn't require much pre-processing. However, part way through the project I realized that frequent sentences with no spacing between their final period and the first word of the next sentence were distorting my analysis. The fix was relatively simple. Thanks to help from a fellow student, I expanded my experience with cleaning using regular expressions.[1] Interestingly, the fix resulted a pretty significant improvement in my modeling because many unrelated sentences weren't being randomly carried into the analyses. A previous realization solved a similar issue when it became clear that selection of sentences containing 'love' were picking up sentences containing 'glove' and 'Strangelove', etc.[2] Those were valuable lessons about the importance of data cleaning!

My hope was that the range of ways that the TED authors talked about love would provide a rich data set to analyze. In hind sight, while some talks do explore the concept of love more extensively, most use the term in passing and analysis of the way it is used didn't provide a lot of richness. One metric that demonstrates this is that more than 500 of the talks use love in only one sentence out of the, on average, approximately 150 sentences in an average TED talk[3]. Despite that, the corpus did provide a perfectly adequate collection of sentences to run

---

[1] See the file fix_run_together_sents.ipynb for explanation of where I was working out run-together sentences. This file is also the file that is ultimately used to save the pickled data frame that is used in all the modeling.

[2] See the file love_not_glove.ipynb

[3] See the modeling file cv_binary-lsa.ipynb for analysis of the frequency of use of love. In my pre-proposal EDA I had identified that more than half the talks contained the word love. However, I didn't explore that further to assess how extensively the word was used in each talk.

the NLP methods on to get some experience with the tools. Overall, I feel like the deeper dive I took into clustering (discuss below) was interesting and helpful and has instilled a motivation to dig into it more deeply in the future.

Modeling

Note that slide 3 of my slide_deck shows the NLP methods I utilized (listed below chevrons). Generally modeling effort can be found in jupyter notebooks in the Models repo folder.

Cluster Analysis

A very significant portion of this project involved clustering and trying to understand what the clusters represent. Efforts to investigate clustering and cluster meaning is sprinkled throughout all my modeling Jupyter notebooks. Especially until I overcame the data cleaning issues discussed above, neither looking at words scoring highly in topics nor documents representative of clusters were easy to make sense of.

I encounter a wide variety of cluster shapes from long, stringy clusters[4] (Figure 1) to relatively dense blobs[5] (double click movie Figure 2) to more diffuse clouds[6] (double click movie Figure 3). Fortunately, all these examples are in a dimensional space that I could visualize to validate if

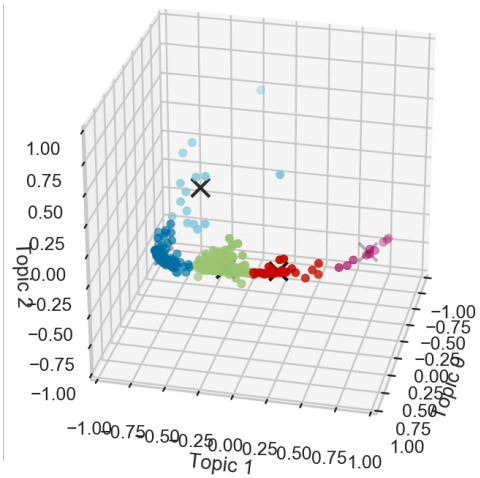

Figure 1 - stringy clusters

Topic Modeling Clusters

Topic Modeling Clusters

Movies playable by double clicking in Pages document.

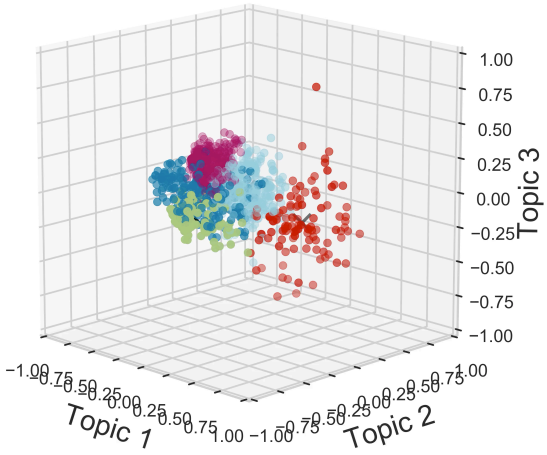

Figure 2 - Tight Clusters (movie)

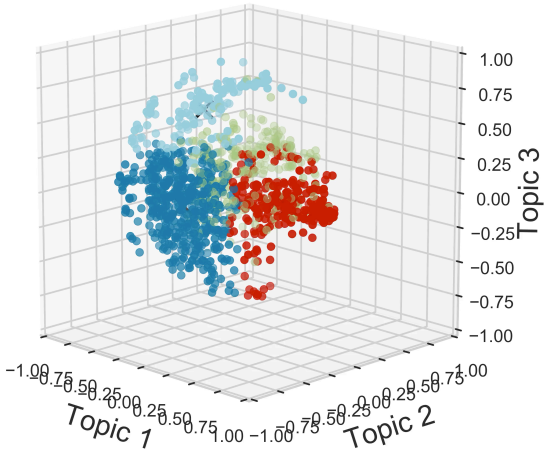

Figure 3 - Diffuse Clusters (movie)

[4] From cv-lsa.ipynb "The 3-Topic Result".

[5] From cv_binary-lsa.ipynb "The 4-Topic Result"

[6] From tfidf-lsa.ipynb "The 4-Topic Result"

clusters were distinct and not just arbitrary separations of a single cloud or merging points between obviously separated parts of distinct clouds.

In all the cases shown here the cluster are the result of K-Means clustering using the optimum number of clusters as arrived at by elbow analysis. However, I implemented DBSCAN, MeanShift, Hierarchical, and Spectral clustering methods too.[7] I found each method had unique peculiarities. For example, tuning DBSCAN was extremely sensitive to the eps parameter leading to wide variations in the numbers of clusters identified. Similarly, the quantile parameter for MeanShift clustering also lead to wide variations in results. I used ward, average, and complete linkages in Hierarchical clustering with variation in results as expected. Validating or evaluating the results of clustering methods would be overwhelming if only done comparing topic words or document texts. In my case I was at least able to visually assess if clusters "made sense" looking at cloud shapes. Validating cluster in higher dimensional spaces would have been even harder. Exploring this issue is something I will likely continue in the future.

I'll also mention that I built the animations and movies shown here with matplotlib's Axes3D and Animation libraries and FFMpeg writer for mp4 video file creation as shown in their notebooks.

Python Modules

Building upon failed efforts during the previous project and with help from Cliff, I was able to implement a Python module in this project. The results of this reside in the functions.py file in the src folder. Among functions important to my project (e.g. get_sents), I also created a unit_normal function to normalize doc_topic matrices before I realized a more capable implementation was available in sklearn.preprocessing.normalize. I did validated that my function and normalize gave equivalent results[8] before abandoning my function. This added to my learning the lesson discussed multiple times regarding not reinventing the wheel - look for available libraries before implementing one yourself.

Remote Processing

Finally, I spent some effort early in the project working to figure out how to get my AWS instance up and running with the 'metis' conda environment.[9] I thought I might need to use an AWS machine to run model training. I didn't end up needing that which was fortunate because I was not able to transfer the 'metis' environment from my computer to the AWS machine. I could certainly have manually installed all the libraries, but transferring the environment in whole with yaml resulted only in errors. I suspect that the number of libraries and library version dependencies are too complicated and causing conflicts. Its also possible that some overlap of library installations via both conda and pip is enabling my computer to work but not transfer easily. I abandoned this effort when I found processing on my computer was feasible.

---

[7] See tfidf-lsa.ipynb "The 3-Topic Result"

[8] See normalize_vs_unit_norm.ipynb. This helped me verify that results I was getting using my function were not flawed due to this function.

[9] See the AWS_setup folder for artifacts of that effort.