Creating Fun With Phonemes and Sentiment Analysis

Introduction

What makes a joke funny? Each joke contains some kind of idea, irony, or unexpected twist. The form, or the delivery, of the joke, also matters. Part of the delivery of a joke is phonological in nature. People respond differently to different sounds. For example, one marketing psychology study claims that people are able to more frequently recall and recognize words that begin with a plosive (Bergh 1984). You can decide if this is true (and perhaps even effective) based on whether or not you can still remember the title of this project.

In comedy, there is a widespread idea that the plosive /k/ is the funniest sound. For example, humor writer Mike Sacks claims, “Ks are funny - Hoboken. Plastics has a K sound, Fresca” (Simon 2009). Many such claims abound in a Google search; they appear to have originated in a bit from the film “The Sunshine Boys”, in which the main character claims that the K sound is funniest. However, this film in turn seems to have borrowed this idea from writer H. L. Mencken. In one study that worked to determine which words were funniest, the author states that the ideal word is one that:

> “...is a short, infrequent word composed of uncommon letters that is likely to include one or more specific letter or phonemes (/u/, consonant -le, y, and/or /k/), with an animate (or animacy-related) referent, especially one that is human and insulting, profane, diminutive, and/or related to good times” (Westbury & Hollis 2019).

Since humor is super complex and entails content from a variety of fields, this project will only focus on the phonemes /u/ and /k/. I wrote a small program to make a very superficial analysis of how funny a comedian is based on these sounds. I also worked with a sentiment analyzer that can evaluate how positive or negative a comedian is. The sentiment analyzer is trained off of datasets of IMDB posts. I will use 20-minute transcriptions from these performances: “Old Yeller” from Lewis Black, and a Dave Chappelle performance.

Hypothesis

Since the /k/ and /u/ phonemes are apparently the funniest, I hypothesize that the funnier comedian will use more of these sounds. Referring to the above ‘ideal word’, I also hypothesize that the funnier comedian will also be more generally negative, since being profane and insulting is associated with being funny. To test this hypothesis, I will use the quantitative analysis generated from these programs to determine who is funnier. I personally think Lewis Black is funnier, so I assume he uses more of the /k/ and /u/ phonemes, while being more generally negative.

Data Collection

Phoneme Counter

To collect this data, I first made a program that counts phonemes. First, I had to modify the text by making everything lower-case. Then, I used the Regular Expression Tokenizer from NLTK to tokenize the text. Then, I used an NLTK dictionary to convert the words into Arpabet so that I could count the phonemes. Next, I wrote a function called ‘phoneme_counter’. I created a variable KCount and set it equal to zero. I first had it print the entire Arpabet generated by the corpus. Then, I wrote a for loop that contained an if statement that counted the number of “K” strings in the generated Arpabet list, and populated the variable KCount with that list. I had a lot of problems with KeyErrors concerning the contractions. I bypassed these errors using try/except, and tested a few smaller text strings with the function to ensure that my phoneme count wasn't being ruined. I added an identical group of things to count the /u/ phonemes ("UW1" in the Arpabet). Then, I wrote some print statements to show the counts of the /k/, /u/, and total of the two phonemes found in the transcripts.

Sentiment Analyzer

I worked with a tutorial program to do the sentiment analysis. A link to the tutorial is in a comment in line 1 of the program sentiment_analyzer. Since I have never worked with machine learning or TorchText, I will attempt to provide a hopefully accurate summary below of how the program works. Each section of the program is marked with a #1, #2, and so on.

I. For #1 and #2, this program begins with a bigram generator similar to that of a previous homework in this class. A variable n_grams is created and then populated with a list. Then, a for loop that uses append to join the various combinations of two words together. Next, there is a sample print statement showing the bigram function in action.

II. Next, for #3, I imported the PyTorch and TorchText libraries, along with the sample datasets used to train the programs. The datasets are the IMDB posts used for training the program. In #4, the IMDB data is used to train the program.

III. For #5 and #6, the vocabulary data is created. In machine learning, a vector is a 1-dimensional array used for storing data. A tensor is a generalization of matrices using n-dimensional arrays. These are used for organizing data. The device 'cuda' is my computer; during my ordeal with installing PyTorch, I was able to configure the CUDA thing on my laptop so that it could run these processes natively.

IV. In #7-#9, I imported the neural network and got a function together that would arrange the text into tensors for processing. Then, the parameters were defined into three dimensions: input, embedding, and output. In #9, we can see how many parameters there are.

V. In #10-#12, the TorchText optimizer was initialized.

VI. In #13-19, the function to test the accuracy of the training was created. Then, the model is trained in 5 epochs. An epoch is a measure of the number of times all training vectors were used one time to update weights. The number of epochs can affect the accuracy of the training. This program has a thing to show how accurate it thinks the prediction will be with the number of epochs.

VII. Finally, in #20 and #21, using SpaCy, a module that does things similar to NLTK, I put together the function for tokenizing text and executing the prediction of whether a string has a negative or positive sentiment. This uses the bigram function from the beginning, along with the tensors. Then, I added my transcripts as variables and ran the predict_sentiment function on my transcripts.

Transcripts

I transcribed the two comedy specials mentioned in the introduction into a text file. It's not a super exact transcription because I did not include things like "Um", and I mostly wrote things like "gonna" as "going to". These didn't affect the results -- no /k/ or /u/ sounds got lost. I transcribed 20 minutes of performance from both Lewis Black and Dave Chappelle.

Results

	Lewis Black	**Dave Chappelle**
/k/ phoneme count	213	307
/u/ phoneme count	203	210
Total of /k/ and /u/	416	517

Sentiment: 1 = totally positive 0 = totally negative	0.07230111211538315	0.0020793809089809656

Conclusion

Based on the above data, Dave Chappelle is funnier. I also noticed, while transcribing, that his punchlines almost always contain the /k/ phoneme somewhere (example: "Let's sprinkle some crack on him" -- used three times). Both had roughly the same number of /u/ phonemes used. Dave Chappelle is also significantly more negative. This is, I assume, because he uses far, far more curse words, which themselves are a metric of how funny something is (Westbury & Hollis 2019). If you consider financial net worth to be a measure of success, Dave Chappelle is also significantly richer than Lewis Black -- another correlation with the results above. If a person wishes to succeed in comedy based on these metrics, then a good start would be to use more /k/ phonemes and be as negative as possible.

Work Cited:

Bergh, B. G. V., Collins, J., Schultz, M., & Adler, K. (1984). Sound advice on brand names. *Journalism Quarterly*, *61*(4), 835-840.

Simon, S. (2009, August 1). Comedy Writing: How To Be Funny. Retrieved from https://www.npr.org/templates/story/story.php?storyId=111456667.

Westbury, C., & Hollis, G. (2019). Wriggly, squiffy, lummox, and boobs: What makes some words funny?. *Journal of Experimental Psychology: General*, *148*(1), 97.

Performances:

Lewis Black: Old Yeller
https://tubitv.com/movies/487599/lewis_black_old_yeller_live_at_the_borgata

Dave Chappelle: STAND UP SPECIAL Live San Francisco 2018
https://www.youtube.com/watch?v=5UeJEJilnII