Sammy Pardes
IST 664
Homework 1
2/4/21

Introduction

Jeopardy! is an American game show where contestants provide the questions to answers of trivia from a variety of categories. First airing in 1964, there have been over 8,000 episodes in its 37 seasons. Still, Jeopardy! continues to captivate audiences around the world and watching it nightly has become a tradition in many households.

While it seems like a straightforward game to play, there are certainly strategies players utilize to win big. Famous players like James Holzhauer are known for wagering a lot and going after high value questions.

Using a repository of previous questions and some NLP, can we determine what to study to have better chances of winning a game by knowing the answers to the highest-valued questions?

About the Data and Preprocessing Steps

The data used for this analysis came from Kaggle.com and originally contained over 200,000 Jeopardy! questions with 7 different attributes. After reading in the .csv file using pandas.read_csv(), the columns were renamed with the columns function to remove white space and capital letters.

Questions with a point value of “None” were removed from the data set. The value column was then cleaned by removing dollar signs and commas. Next, the value strings were converted to integers with the astype() function so that they could be arranged numerically.

From this large repository of questions, two smaller data sets were created. The 20,000 lowest valued questions, about 10% of the total number of questions, were saved in a variable called jeopardylow using the nsmallest() function. Conversely, the 20,000 highest valued questions were stored in jeopardyhigh using nlargest(). Each question in both the high and low data sets were converted to all lowercase using str.lower().

Looking through the .csv file, I noticed some rogue HTML within the questions column, mostly consisting of anchor and image tags. To further prepare the questions for analysis, an empty list called html was created. A variable called pattern was used to store the following regular expression: ‘<.*[\s]?/?.*>?|target="_blank">.*\.?’. This was used to ensure that any token in a question containing a left angle bracket followed by any number of characters, an optional space, an optional backslash, more potential characters, and one or zero right angle brackets, would be stored in the html variable. This is also accounted for anchor tags where the string ‘target=”_blank”>’ followed by any number of word characters appeared. A for loop was used to tokenize each question in the high and low value data sets. For every question, each word in the question was split on whitespace. If the word appeared matched the RegEx pattern, the word was appended to the html list.

A list of stop words was defined containing the english stopwords from the NLTK package. After viewing the unigrams generated using only the default stopwords list, a short list of custom words were added. For example, in Jeopardy! the “Clue Crew” are a group of people who travel to different locations to

record questions for the show. These words both appeared often in the data set and I determined that these words could be removed because they did not add insight to the difference between high and low value questions. “Clue” and “crew” were listed as custom stopwords. Additionally, I noticed a few unwanted tokens slipped by because they contained some extraneous punctuation or were not in the original list. The following stopwords were added: ‘like’, ‘also’, ‘may’,'"it\'ll', '"the', and 'this,'. The numbers ‘1’, ‘2’, and ‘3’ appeared often and seemed unrelated to the contents of the questions, so were also added to the list of stopwords.

Now that the list of words to be removed from each question were compiled, I tokenized each data set of questions. Two empty lists were initialized called lowtokens and hightokens. For each question in a list, every word in the question was split on the whitespace to create tokens. If the word was not in the stopword or html list and was not considered punctuation (based on string.punctuation), the word was appended to its respective list. I decided not to use any stemming or lemmatization on the questions.

While the steps outlined above took care of most of the unwanted words, there were still a few issues with my tokenization. Symbols appeared in some of the tokens, like “(video”. Additionally, I kept in the names of the Clue Crew because their first names could also be the names of other important, potentially frequently mentioned individuals.

Data Models - Unigrams, Bigrams, and Trigrams, Oh My!

FreqDist() and most_common() were used to get the top 50 unigrams for both the high and low questions list. These were then normalized by dividing by the len() of the respective list and printed using a for loop.

The bigram frequency distribution displays the most common pairs of words. For example, ‘last' and 'name’ often appear together. However, ‘last’ and ‘name’ could also appear in different contexts, like ‘last time’ or ‘first name’.

Another way to analyze bigrams is with the mutual information score. Mutual information calculates the likelihood of one word following another. As shown in the high value bigrams by PMI list, ‘agatha’ is often followed by ‘christie’. This makes sense because there aren’t very many famous Agathas. While, ‘agatha christie’ may not appear as often as other pairs of words in the data set, if the word ‘agatha’ appears in a question, it is very likely that it is followed by the word ‘christie’.

To create the lists of bigrams by frequency distribution, I created a shorthand variable, called bgmeasures, for nltk.collocations.BigramAssocMeasures(). To create the finder of the bigrams, I used BigramCollocationFinder.from_words() on each list. Then, I plugged the finder variable into score_ngrams(bgmeasures.raw_freq) to the score ngrams by frequency and printed the top 50 with a for loop. To generate the bigrams based on PMI, I implored a similar method as gathering frequency distribution but instead of bgmeasures.raw_freq, I used bgmeasures.pmi to score the ngrams.

A near identical process was taken to generate the top 50 trigrams for each set of questions. The only difference was that I used TrigramAssocMeasures() and TrigramCollocationFinder to create the finder variable, as opposed to BigramAssocMeasures() and BigramCollocationFinder.

Examining the Text

Gathering the unigrams for both high and low value questions yielded some interesting results. While many words appeared in both lists, there were some unigrams unique to one list or the other that stood out. “Tv”, “song”, “show”, and “hit” are words that are only turned up for the low value questions. Although “show” and “hit” have multiple meanings, these words seem to indicate that the low value questions may be about pop culture. “British”, “greek”, “latin”, “french”, “king” are words representative of the high value questions. Initially, it seems like questions that are worth more tend to be about history or geography.

Looking at the bigrams, sorted by frequency distribution, a similar trend appears. “No. hit”, “theme song”, “tv show”, and “ice cream” only show up in the low value list. “African country”, “name greek”, “latin to”, and “bears name” are high value bigrams. It still seems my initial assessment that low value questions skew towards pop culture may hold true. The high value questions may also include linguistics in addition to history and geography.

Further supporting this observation, in the top 20 bigrams sorted by PMI, we see “annie hall”, “wheel fortune”, “barbara streisand”, “warner bros”, and “mick jagger” for the bottom 10% of questions. For the top 10%, we see historical figures and authors like “agatha christie”, “e.m. foster”, “edgar allan”, “julius caesar”, and “h.w. bush”.

Evaluating the trigrams for each data set gave even more insight into the types of questions and their corresponding values. We see “talk show host”, “monday night football”, “saturday night fever”, and “ice cream flavor” appearing frequently for lower value questions. “Name comes latin”, “south american country”, “comes greek word”, and “nobel prize literature” are unique to the high value q’s.

Conclusion

While it’s impossible to know exactly what will be asked in any game of Jeopardy!, we now have a better idea of what to study to conquer the board and what categories tend to be worth more points. If you’re looking to impress your friends, or perhaps try out for the show, I suggest reading up on historical figures (especially writers), latin and greek words, and african and south american geography.

With a bit more analysis, I think this study list could be narrowed down even further. Perhaps using a smaller data set (for example, the top and bottom 5% of questions), and doing stemming/lemmatization would yield more specific and insightful results.