DATA 620: FINAL PROJECT: JEOPARDY!

Network analysis and Text processing

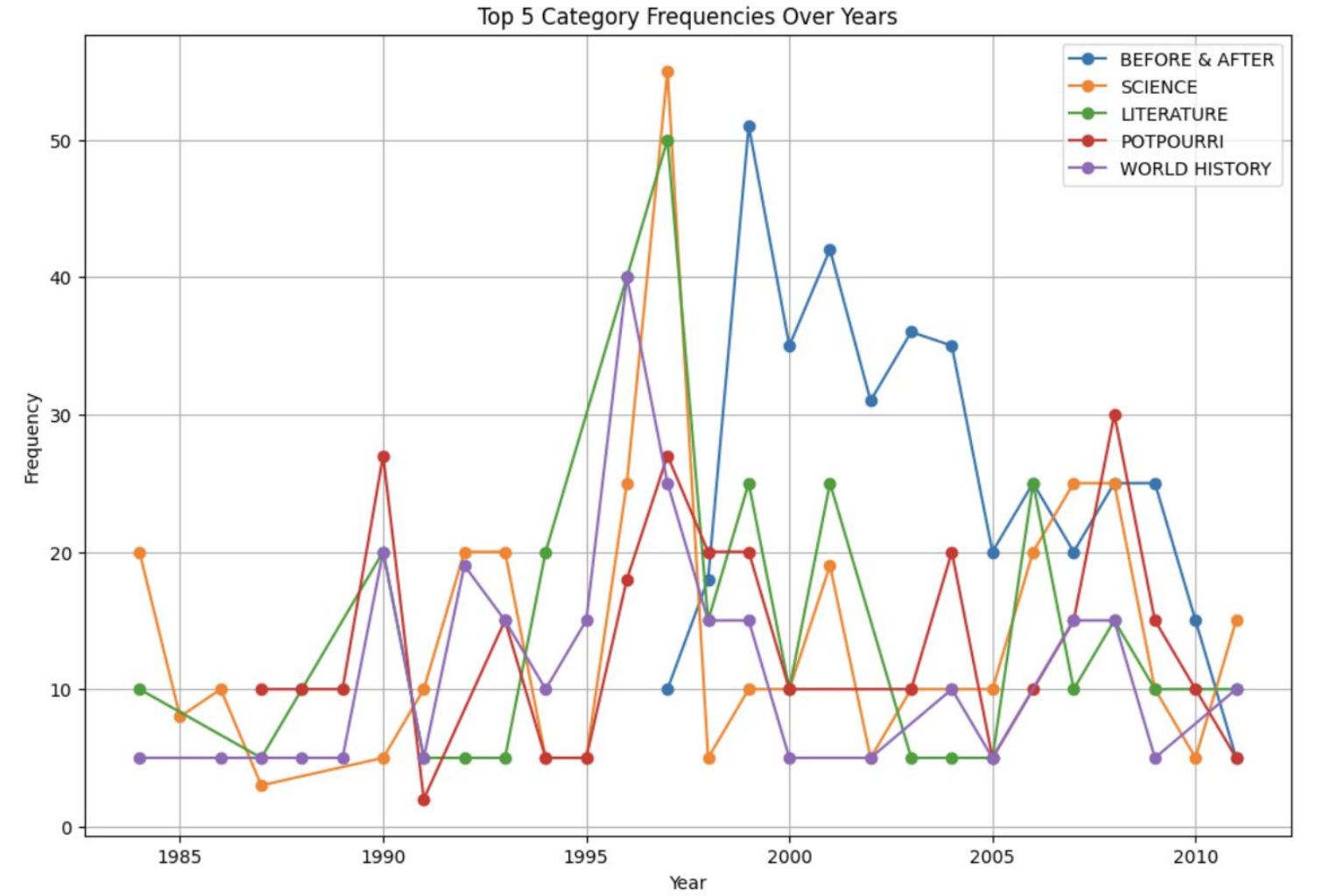
Group: *Susanna Wong, Puja Roy, Mikhail Broomes & Marjete Vucinaj*

Motivation:

- The goal of this project is to understand the patterns and relationships within Jeopardy! questions can provide insights into the shows content evolution and difficulty distribution. This analysis will be valuable to contesters and fans interested in the structure of the quiz questions and their categorizations.
- We will try to predict what category a question would be based on the body of text. We will attempt to use text processing to extract features and train initial models for predicting categories.

Dataset:

- The dataset contains 150,000 Jeopardy! questions in CSV format, sourced from www.j-archive.com. It contains questions from 1984 to 2012. This subset is part of a larger collection of over 500,000 questions aired on the show. It includes details such as question category, value (except for final rounds), question text, answer, round type, episode number, and air date.
- **Data cleaning:**
  - Cleaning 'Value' column: Dropping rows with missing values and remove dollar signs '$' and commas ',' from the values. Then converted datatype to floating-point numbers.
  - Cleaning 'Question' column: Some entries in this column contain html tags of images f



Do the top categories change through out the years?

The top categories seem to be consistent for the most part. In the past some of the categories were asked at much higher frequency, for example History was asked a 60+ times in 1997, and Before & After was asked at the highest frequency between 1997 to 2005. As we approach more recent years the categories seem to be asked in similar frequency, between 2005 and 2012 all categories were asked 30 times or less.

Frequency Analysis:

Identify the most common words or phrases in the questions or answers


```
[ ]  def clean_text(text):
         text = re.sub(r'\([^)]*\)', '', text)
         text = text.lower()
         text = text.translate(str.maketrans('', '', string.punctuation))
         text = re.sub(r'\b(clue|crew)\b', '', text)
         words = word_tokenize(text)
         stop_words = set(stopwords.words('english'))
         words = [word for word in words if word not in stop_words]
         return ' '.join(words)

     df['Cleaned_Question'] = df[' Question'].apply(clean_text)
```

```
     df['Tokenized_Question'] = df['Cleaned_Question'].apply(nltk.word_tokenize)
     stop_words = set(stopwords.words('english'))
     df['Tokenized_Question'] = df['Tokenized_Question'].apply(lambda x: [word for word in x if word not in stop_words])

     all_words = [word for tokens in df['Tokenized_Question'] for word in tokens]
     word_freq = Counter(all_words)

     wordcloud = WordCloud(width=800, height=400, background_color= 'white').generate_from_frequencies(word_freq)
     plt.imshow(wordcloud, interpolation = 'bilinear')
     plt.axis('off')
     plt.show()
```

- Text preprocessing: Define the function, convert text to lowercase, remove punctuation, parenthesis, tokenize text, and remove stop words, then apply the function to the Question column
- *Code and image reflect the ‘question’ data; same process was completed for ‘answers’*

What are the top words in ‘Answers’

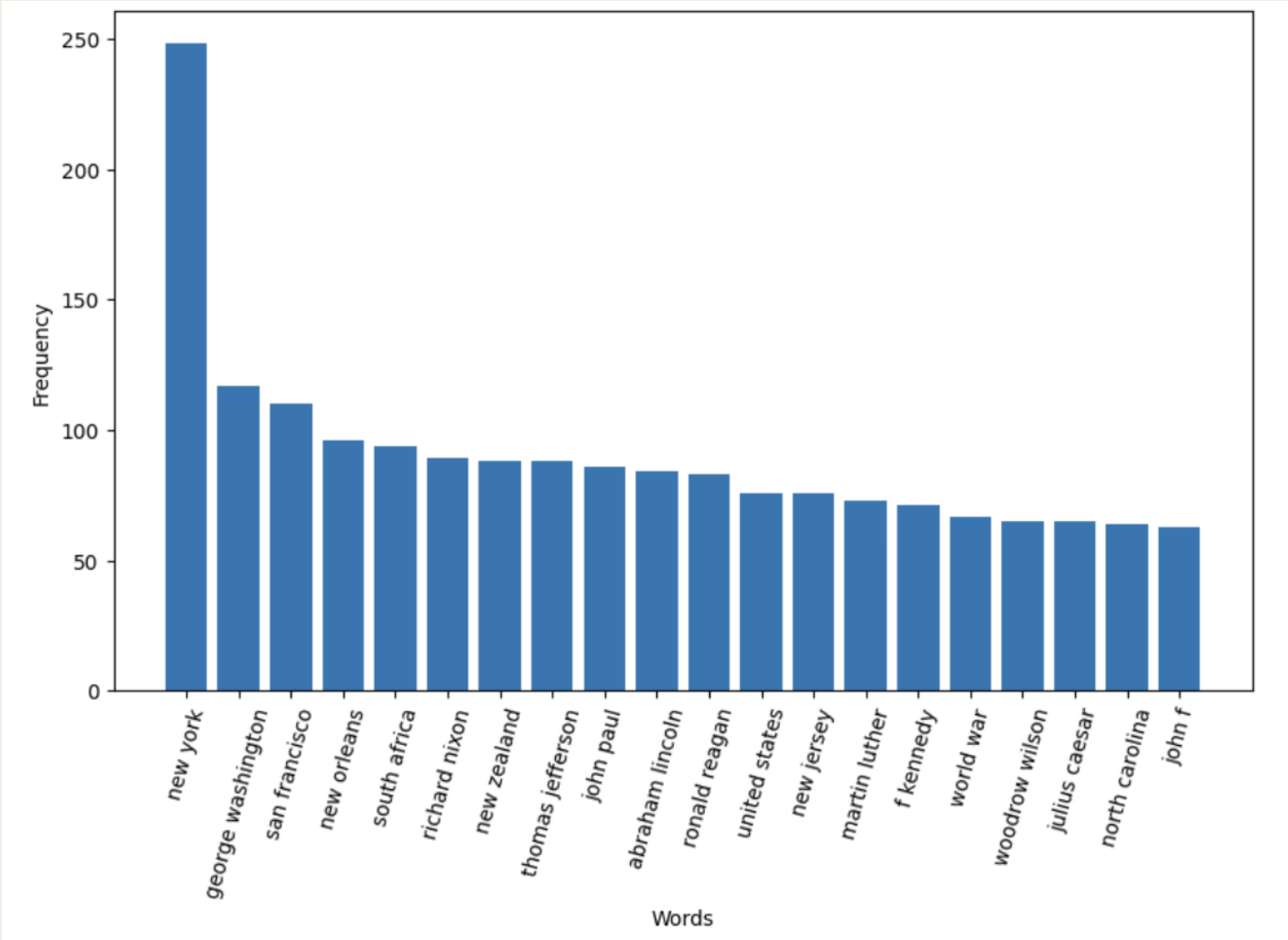
The top common words in the Answers is with the highest frequency is also 'New York'; a total of 9 cities, states and countries are found in the common words of Answers. Additionally, 11 of the top common words in Answers are names with historical importance.

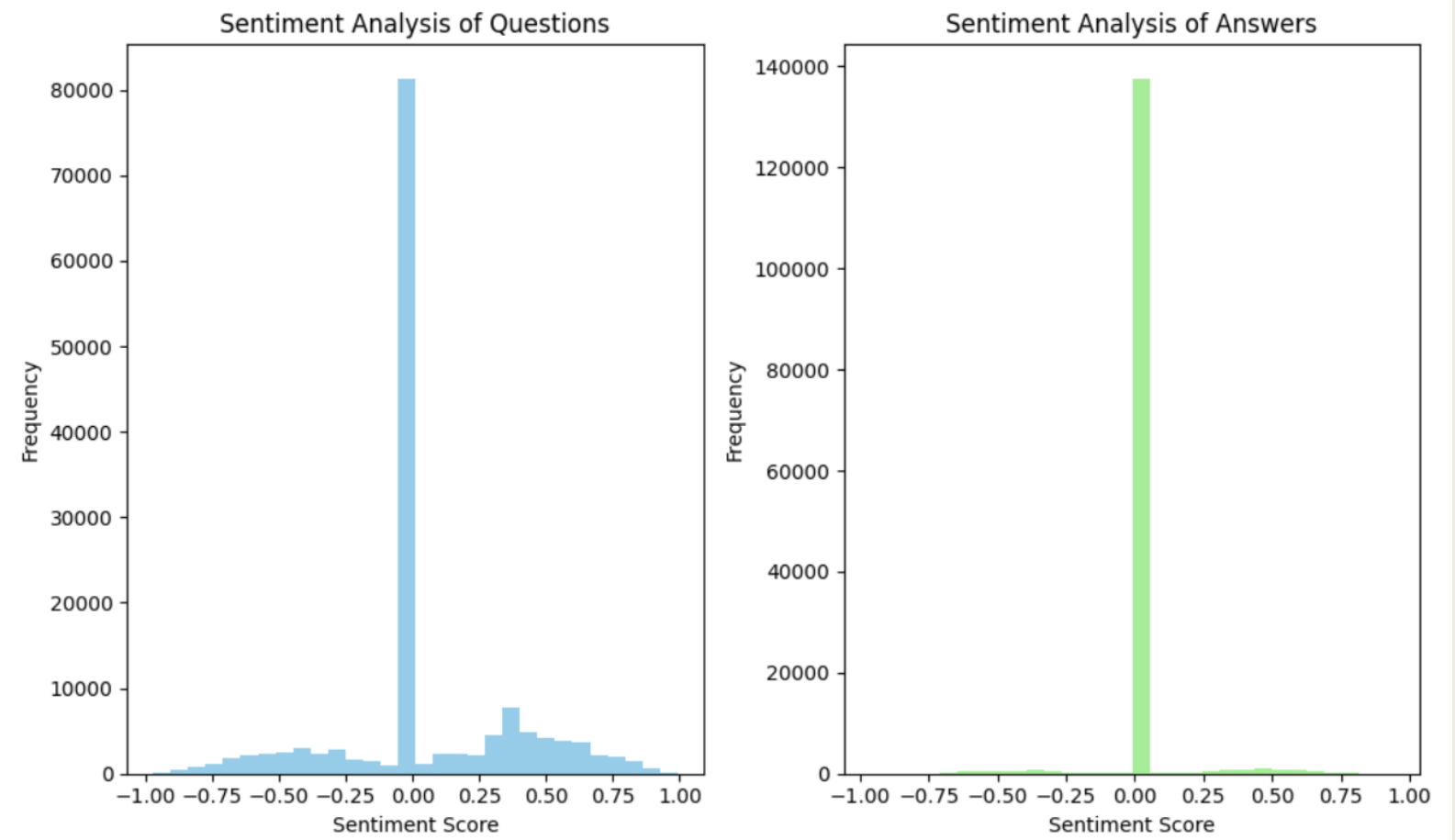
```
bigrams = ngrams(all_ans, 2)
bigram_freq = Counter(bigrams)

most_common_bigrams = bigram_freq.most_common(20)

bigrams, freqs = zip(*most_common_bigrams)
bigrams = [' '.join(bigram) for bigram in bigrams]
plt.figure(figsize = (10,6))
plt.bar(bigrams, freqs)
plt.xticks(rotation = 75)
plt.xlabel('Words')
plt.ylabel('Frequency')
plt.show()
```

Same process was completed for ‘question’



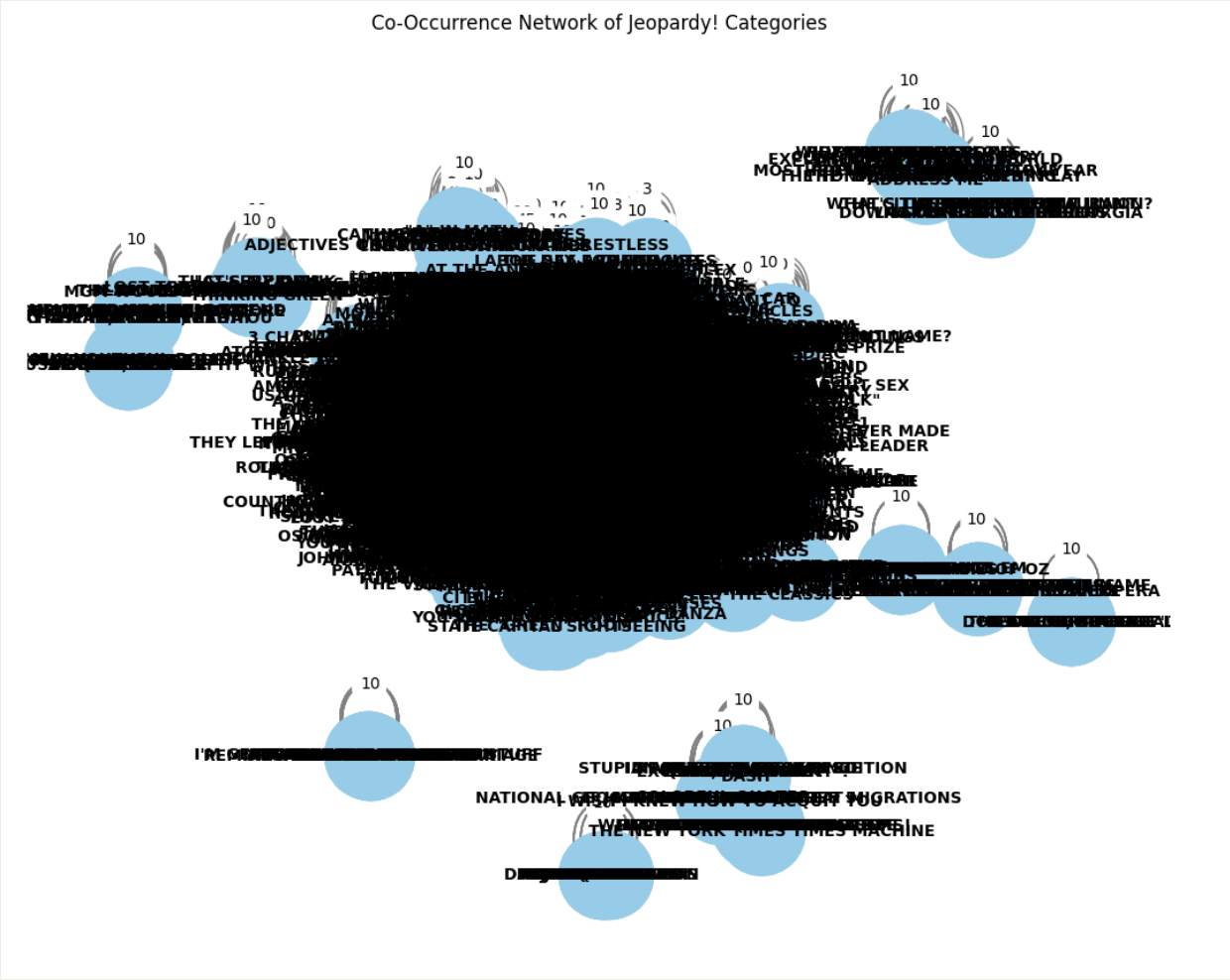


Sentiment Analysis

Determine the sentiment of questions and answers: Both the Questions and Answers are generally around 0, which suggests the text is neutral. Almost 140,000 of answers are neutral (0) possibly because they tend to be shorter. Around 80,000 Questions were at the sentiment score of 0, with a wider range (compared to answers) of positive and negative sentiment score. They might have more emotional language; examples of questions with positive and negative sentiment were printed for context.

Network Analysis

- Creating a Co-occurence Network
- Nodes represent categories, and edges represent the co-occurrence between categories, with edge labels showing the weights (co-occurrence counts).

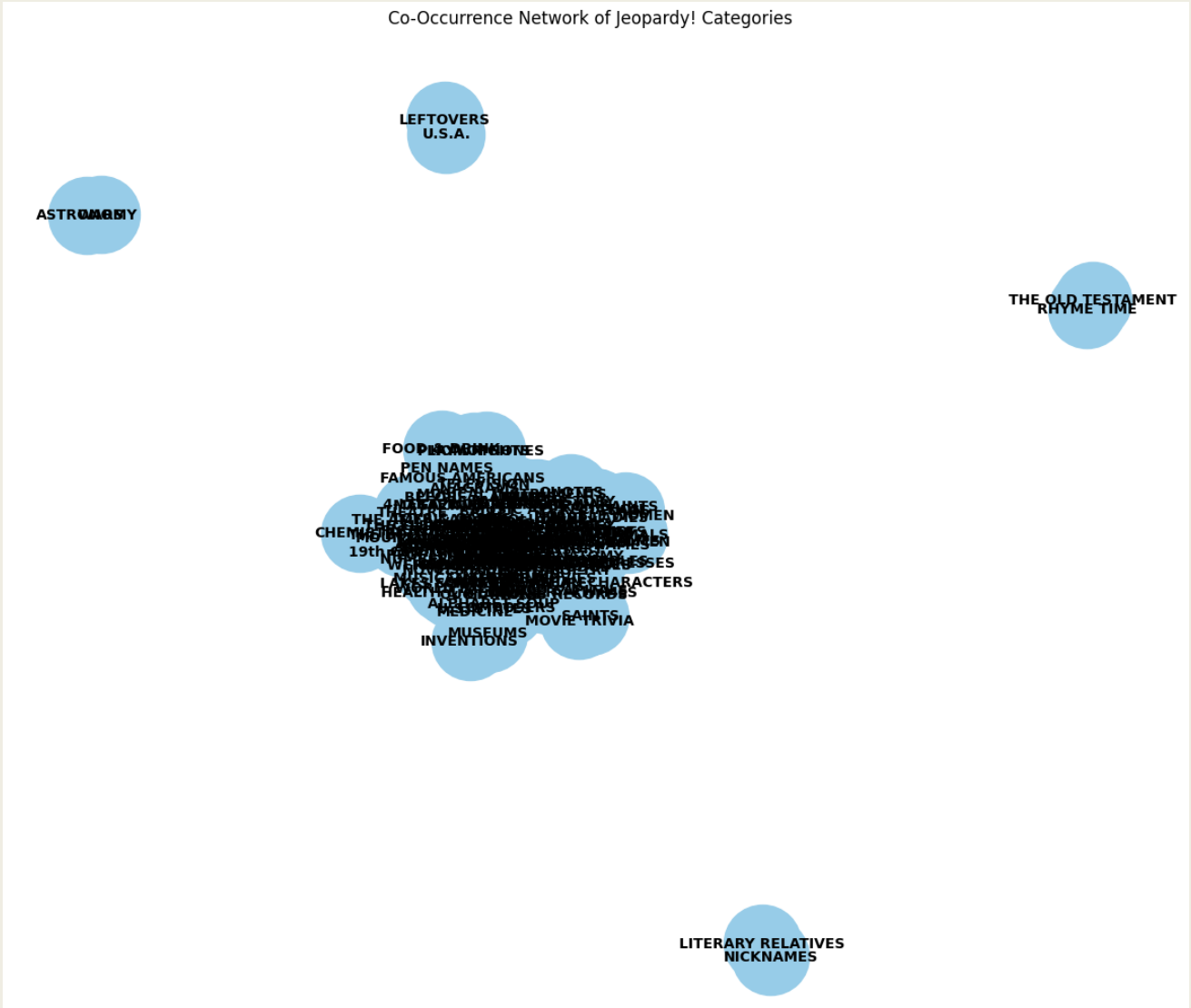


With 19,779 unique categories, there are 182,598 connections between these categories. Each category is connected to approximately 18 other categories which suggests that while there is a significant amount of interconnection, it's not overly dense. The low density of 0.000933 indicates that the network is not heavily interconnected.

Network Analysis

- Get the most frequent co-occurrence pairs
- Only category pairs that co-occur more than twice are retained in the network.
- Looking at the pairs of Jeopardy categories, we can see that U.S. Geography and World History are the most common co-occurring categories. Additionally, we observe that "History" and "SPORTS" are frequently seen among other category pairs. Based on this analysis, if you are preparing for Jeopardy!, you should focus heavily on both "SPORTS" and "History."

```
Most Frequent Co-Occurrence Pairs (Top 10):
U.S. GEOGRAPHY - WORLD HISTORY: 6
ANIMALS - HISTORY: 6
AUTHORS - HISTORY: 6
GEOGRAPHY - SPORTS: 5
MUSIC - SPORTS: 5
BIRDS - WORLD GEOGRAPHY: 5
FOOD - SPORTS: 5
LITERATURE - WORLD HISTORY: 5
SCIENCE - WORLD HISTORY: 5
FASHION - SPORTS: 5
```



Predictive Modeling

- In attempt to predict category based on the body of text, we used 3 different models: Logistic Regression, Random Forest, KNeighbors and added ngrams to each to assess their accuracy.
- We found that the Logistic Regression model (with no ngrams) is the best classifier among the ones we've tried since it has the highest accuracy, precision, F1-score.

```
X = filtered_df['Cleaned_Question']
y = filtered_df[' Category']

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a pipeline with TF-IDF
pipeline = Pipeline([
    ('tfidf', TfidfVectorizer(stop_words='english')),
    ('clf', LogisticRegression(max_iter=1000)),
])

# Train the pipeline
pipeline.fit(X_train, y_train)

# Predict the answers for the test set
y_pred = pipeline.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy * 100:.2f}%')
print(classification_report(y_test, y_pred))
```

```
Accuracy: 58.28%
                  precision    recall  f1-score   support

AMERICAN HISTORY       0.61      0.43      0.50        47
  BEFORE & AFTER       0.52      0.74      0.61        81
 BODIES OF WATER       0.87      0.92      0.90        52
         HISTORY       0.31      0.19      0.23        43
      LITERATURE       0.77      0.71      0.74        68
       POTPOURRI       0.43      0.18      0.26        65
         SCIENCE       0.56      0.84      0.67        69
   WORLD HISTORY       0.44      0.46      0.45        52

        accuracy                           0.58       477
       macro avg       0.56      0.56      0.55       477
    weighted avg       0.57      0.58      0.56       477
```

Conclusion

- **Change over time** Between 1995-2000 some categories were asked in much higher frequencies; however, they average out between 2005-2012. By Normalizing frequencies we were able to identify that 'before and after' was a new category in late 1990's.
- **Text Processing** In the word cloud of Questions, 'one' and 'name' were visually the largest. Bigram of the questions indicated that 'New York' is the top common phrase. There is a clear pattern of questions asking contestants to identify something similar ie 'also called', questions relating to names, and historical or political context. Similarly, in the Answers wordcloud,'John' and 'new' were the largest words. The top common words in the Answers is with the highest frequency is also 'New York'. 11 of the top common words in Answers are names with historical importance. For both trigrams provided more context some interesting distinctions.
- **Sentiment Analysis** Both the Questions and Answers are generally around 0, which suggests the text is neutral. Questions had a wider range of positive and negative sentiment score.
- **Network Analysis** Looking at the pairs of Jeopardy categories, we can see that U.S. Geography and World History are the most common co-occurring categories. Additionally, we observe that "History" and "SPORTS" are frequently seen among other category pairs. Based on this analysis, if you are preparing for Jeopardy!, you should focus heavily on both "SPORTS" and "History."
- **Predictive Modeling** In attempt to predict category based on the body of text, we used 3 different models and added ngrams to each to assess their accuracy. We found that the Logistic Regression model (with no ngrams) is the best classifier among the ones we've tried since it has the highest accuracy, precision, F1-score.

Further Enhancement

- Our original model includes all data. However, we came upon these problems that prevented us to use the full dataset and include more features:
  - long run time
  - notebook crashing before the code finish running
- With more time we would have liked to advance our network analysis and construct a network to determine which Jeopardy category is worth more. Additionally, we were unable to conduct temporal Analysis: based on the air data, analyze how categories, questions, or answers evolve over time.
- If we have more time, we want to try modeling with more of our dataset, potentially using methods such as LDA to reduce the feature space and include additional feature engineering steps like lemmatization, text length, and more.

Full code can be found on

- Full code can be found on https://colab.research.google.com/drive/1jSCXR1Ib3m-ZV8gre8GQA5P56Ix2bQgm?usp=sharing