Summary

For my project, I wanted to see how the sentiment or language surrounding gaming has changed in 20 years. My research question was, how has vocabulary used frequently with gaming evolved after two decades in media? The dataset I used was the NPR Media Dialog Data. I used the A-Priori algorithm to find the frequent words used in the same sentences as “gaming.” The intended result was the most frequent words in 1999-2009 and the most frequent words from 2010-2019. The significance of the result was that it allowed for an easier comparison of the language surrounding gaming between the years.

Introduction

National Public Radio (NPR) is a media organization, primarily working in the radio industry. The data set contains transcripts of most, if not all, of their aired episodes between 8 Jan 1999 and 12 Oct 2019. From the dataset, I will be looking into the “episodes” and “utterances” CVSs. Episodes contains the fields for episode identifiers, program names, titles, and the air dates. Utterances contains the fields for episode identifiers, episode order, speaker name, and utterance (transcript). For this project, the transcript is what will be used to find the frequent words used together with “gaming.” Episode identifiers from the Utterance table will be used to get the air dates from the Episode table to filter out episodes not from 1999 or 2019. Background information necessary to understand A-Priori would be the meaning of support, confidence, and interest. Also understanding association rules and monotonicity.

I wanted to know how the language surrounding gaming has changed in twenty years. As a kid I remember hearing on tv that video games were a waste of time and were for “losers.” I also remember being told that they make people violent, so early in my life, there was a negative light on gaming. Now that I am older, gaming is more popular than ever, and I feel that there has been a shift in how it is viewed in media. This research helps share my interests with others because it could show how sentiment and perception around gaming has evolved over time. Also, broader than that, I think cultural shifts in general are very fascinating to learn about.

How has vocabulary used frequently with gaming evolved after two decades in media? NPR is a media organisation with a primary focus on the latest news in the USA. The organisation has existed for decades, meaning their data set contains a large and diverse amount of content from news reports and interviews. They have existed through shifts in language and culture which is why their data set is relevant to my research question. The A-Priori algorithm is used for finding frequent item sets. The project will need to look through the NPR transcripts to find frequent words used alongside the word “gaming.” This makes the algorithm perfect for helping to answer the research question.

Experimental Design and Methods

The A-Priori algorithm was used to identify frequent item sets, in this case frequent words used alongside “gaming.” The algorithm is made up of two passes. The first pass involves recording the frequency of each unique item individual item in a basket. The second pass involves the pruning of items whose frequencies are below a certain support threshold. Then generate item sets of all pairs from the frequent items and calculate their frequencies. In my case, pass 2 will be repeated however the repeated pass with have pairs of size 3. The key idea of this algorithm is monotonicity, “If a set I of items is frequent, then so is every subset of I.” This means that frequent item set information can be compacted resulting in a reduction of memory usage from the algorithm.

Method

1. Download the episode and utterance CSVs.
2. Parse the data into a better format for Dask.
3. Load the parsed data into Dask data frames.
4. Create a data frame from merging the episode and utterance data frames. The data frame needs to have the fields episode id, episode order, episode date, and utterance.
5. Filter out utterances from the transcript that do not contain the word “gaming.”
6. Filter out irrelevant utterances from the “gaming” sentences e.g. “the.”
7. Split into two tables. One table for episodes that released between 1999-2009, and another for those the released between 2010-2019.
8. A-Priori:
    a) Generate a frequency table of all the unique words.
    b) Prune words that are below the support threshold.
    c) Generate a 2-itemset frequency table.
    d) Prune sets that are below the support threshold.

[Perform A-Priori on the 1999-2009 table and record the results. Do the same with the 2010-019 table.]
9. Compare the results.

A potential limitation was that the word “gaming” would not appear often enough in the transcripts, and that the words are just not frequent enough in the “gaming” sentences.

The code in *figure 1* below is used to implement steps 1-3 of my method, importing the datasets into Python as Dask data frames.

```
# Set CSVs paths
episodes_csv_path = '/content/drive/MyDrive/Colab Notebooks/DATA301 Project Files/episodes.csv'
utterances_csv_path = '/content/drive/MyDrive/Colab Notebooks/DATA301 Project Files/utterances.csv'

episodes_df = df.read_csv(episodes_csv_path)
utterances_df = df.read_csv(utterances_csv_path)
```

Figure 1: Importing the CSVs as Data Frames.

Figure 2 presents the code that merges the two data frames from *Figure 1* on the id column and episode column. The resulting data frame has episode and episode_order as the primary key. It also removes the unnecessary columns.

```
# Keep only necessary columns
episodes_df_cols_to_keep = ['id', 'episode_date']
utterances_df_cols_to_keep = ['episode', 'episode_order', 'utterance']
merged_df_cols_to_keep = ['episode', 'episode_order', 'episode_date', 'utterance']

# Merge data frames
merged_df = df.merge(episodes_df[episodes_df_cols_to_keep], utterances_df[utterances_df_cols_to_keep],
                     left_on='id', right_on='episode')[merged_df_cols_to_keep]
```

Figure 2: Data Frame Merging Code.

Figure 3 shows step 5 of the method. It filters out rows from the transcript that do not contain the word “gaming” in their utterance.

```
# Filter out rows that do not contain 'gaming'
gaming_df = merged_df[merged_df['utterance'].str.contains('gaming')]
```

Figure 3: Data Frame Created by Filtering Out Utterances Not Containing “Gaming”.

In *Figure 4*, the Natural Language Toolkit (NLTK) library is imported to remove irrelevant words from the utterances. Custom words were also added to support the NLTK stop words. This is one part of step 6.

```
# Import library to be used for filtering out irrelevant words
import string
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

nltk.download('stopwords')
nltk.download('punkt')

# Custom stop words
custom_stopwords = {
    'i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', 'your',
    'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', 'her', 'hers',
    'herself', 'it', 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves',
    'what', 'which', 'who', 'whom', 'this', 'that', 'these', 'those', 'am', 'is', 'are',
    'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does',
    'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until',
    'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into',
    'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down',
    'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here',
    'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more',
    'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so',
    'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', 'should', 'now',
    'gaming', 'game', 'games', 'know', 'like', 'thats', 'people', 'think', 'dont', 'also',
    'theyre', 'tribes', 'get', 'say', 'says', 'theres', 'us', 'get', 'lot', 'years', 'video'}

stop_words = set(stopwords.words('english')).union(custom_stopwords)
```

Figure 4: Importing of Relevant Modules to be used for Cleaning.

Figure 5 is made up of three functions. These have the purpose of cleaning the utterances including making the text lowercase (useful of matching), removing punctuation, and removing stop words.

```
# Converts strings to lowercase
def make_lower_case(utterance):
  return utterance.lower()

# Remove punctuation marks from strings
def remove_punctuation(utterance):
  return utterance.translate(str.maketrans('', '', string.punctuation))

# Remove stopwords from strings
def remove_stopwords(utterance):
  utterance_tokens = word_tokenize(utterance)
  filtered_utterance = [word for word in utterance_tokens if word not in stop_words]
  return ' '.join(filtered_utterance)
```

Figure 5: Utterance Cleaning Functions.

The final part of step 6 is just applying the methods shown in *Figure 5.*

```
# Apply above functions on the utterance column to clean utterances
gaming_df['utterance'] = gaming_df['utterance'].apply(make_lower_case, meta=('string'))
gaming_df['utterance'] = gaming_df['utterance'].apply(remove_punctuation, meta=('string'))
gaming_df['utterance'] = gaming_df['utterance'].apply(remove_stopwords, meta=('string'))
```

Figure 6: Application of The Cleaning Functions.

Figure 7 is simply splitting the current data frame into two. One for episodes from 1999-2009, and the other for 2010-2019. It makes use of the Pandas library to utilise the to_datetime function which converts a string to a datetime data type for easier data comparisons. This was step 7.

```
# Need library to convert single date to datetime data type
import pandas as pd

# Dataframe of episodes between 1999 and 2009 (inclusive)
gaming_1999_to_2009_df = gaming_df.loc[(gaming_df['episode_date'] >= pd.to_datetime('1999-01-01')) & (gaming_df['episode_date'] <= pd.to_datetime('2009-12-31'))].compute()

# Dataframe of episodes between 2010 and 2019 (inclusive)
gaming_2010_to_2019_df = gaming_df.loc[(gaming_df['episode_date'] >= pd.to_datetime('2010-01-01')) & (gaming_df['episode_date'] <= pd.to_datetime('2019-12-31'))].compute()
```

Figure 7: Data Frame Splitting.

Figure 8 below is the main part of the project. This is step 8, where the A-Priori algorithm is implemented.

```
# The A-Priori algorithm
def a_priori(utterance_bag, support=40):
  # Create frequency dictionary of unique words
  step_1 = dict(utterance_bag.map(lambda x: x.split(" ")[:-1]).flatten().frequencies().filter(lambda x: x[1] > support).compute())

  # Pairing words
  def find_pairs(text_line):
    pairs = []
    basket = text_line.split(" ")[:-1]
    for i in range(len(basket)):
      for j in range(i + 1, len(basket)):
        if basket[i] in step_1 and basket[j] in step_1 and basket[i] != basket[j]:
          pairs.append(tuple(sorted((basket[i], basket[j]))))
    return pairs

  # Find frequency of pairs
  step_2_frequencies = dict(utterance_bag.map(find_pairs).flatten().frequencies())

  step_2_prep = [(word_pair[0], word_pair[1], frequency) for word_pair, frequency in step_2_frequencies.items()]

  step_2 = df.from_pandas(pd.DataFrame(step_2_prep, columns=['First Word', 'Second Word', 'Frequency']), npartitions=1)
  return step_2
```

Figure 8: A-Priori Algorithm Implementation.

Results

Unfortunately, I was not able to answer my research question, how has vocabulary used frequently with gaming evolved after two decades in media? My dataset had insufficient data, in retrospect, I see that it was perhaps not relevant enough to the project. As seen in *figure 9-10*, the resulting data has nothing conclusive or to make note of. In the future, for better results, a more relevant data set should be used.

Frequent Words Pairs Used Alongside 'Gaming', 1999-2009

	First Word	Second Word	Frequency
0	one	really	20
1	industry	one	10
2	industry	would	12
3	industry	really	18
4	one	would	19
5	really	would	22

Figure 9: Frequent Word Pairs Used Alongside ‘Gaming’, 1999-2009.

Frequent Words Pairs Used Alongside 'Gaming', 2010-2019

	First Word	Second Word	Frequency
0	going	one	8
1	new	online	12
2	going	new	6
3	going	online	7
4	one	really	16
5	one	online	6
6	online	really	7
7	new	really	5
8	really	well	8
9	one	well	10
10	online	well	3
11	new	one	5
12	going	well	11
13	going	really	5
14	new	well	5

Figure 10: Frequent Word Pairs Used Alongside ‘Gaming’, 2010-2019.

Conclusion

I was sadly unable to answer my research question. The dataset I was using simply did not have sufficient information for what the project was aiming to accomplish. The NPR data set was just not relevant enough to the project to be able to answer the research question.

Well, I suppose that an implication is that perhaps the conversation around gaming is not popular enough in news media. Another implication is that, as mentioned above, the NPR data set was simply not relevant enough to the 'gaming' conversation.

In the future, I would like to use a more relevant dataset for the project. Perhaps then I could answer my research question. Also, I want to learn more about sentiment analysis. This is something I came across while trying to research this project. I found that my project is basically trying to perform a sentiment analysis.

Critique of Design and Project

A part of my design that I feel could have worked better was the filtering of stop words. Step 6 of my method involved the use of the NLTK library. I feel this way because the library was under used in my implementation. I have learnt that you can get specific with how the stop words are derived by using a training set. At the time, I did not really understand how this worked or how I would code this.

Something I would like to change is to take advantage of this feature because, as seen above in *figure 4*, I had to use a lot of custom stop words. It would be nice to not have to write down all the irrelevant words I can think of that aren't part of the NLTK stop words set. This would save me time, but also reveal a more relevant result, cleaner than the current implementation.

Reflection

Something I learnt during my time working on the project is how valuable parallel coding is. It allows us to create more efficient code, saving us time and allowing us perform work on large data sets. I found the algorithms we were taught in the course interesting, with Basket Analysis being the most interesting in my opinion. I discovered something called sentiment analysis which is the analysis of text in the pursuit of trying to find the tone and emotion behind it. This is a fascinating concept that I plan on exploring further.

NPR Media Dialog Data Recommender Page:
https://cseweb.ucsd.edu/~jmcauley/datasets.html#interview

NPR Media Dialog Data Kraggle Download Page:
https://www.kaggle.com/datasets/shuyangli94/interview-npr-media-dialog-transcripts

MMDS chapter 6 - Frequent Item Sets:
http://www.mmds.org/mmds/v2.1/ch06-assocrules.pdf

A-Priori algorithm explanation:
https://www.javatpoint.com/apriori-algorithm

DATA301 Lab 03 Collab Notebook:
https://colab.research.google.com/drive/1StvvadXos6PT4S4PKwq-N-wfpedeFwV9