SI671 Homework1

Name: Junqi Chen

Uniqname: junqich

UMID: 03846505

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from tqdm import tqdm
```

1. Data Exploration

1.1 Load & transform the Twitter emoji dataset

In [2]:
```
# 1.1a
data = pd.read_csv("./itemsets_data/food_drink_emoji_tweets.txt",
                   sep="delimiter",
                   names=["Tweets"],
                   header=None)
data.head()
```

```
<ipython-input-2-2a84ab53b7d0>:2: ParserWarning: Falling back to the 'python' engine because the 'c' eng
ine does not support regex separators (separators > 1 char and different from '\s+' are interpreted as r
egex); you can avoid this warning by specifying engine='python'.
  data = pd.read_csv("./itemsets_data/food_drink_emoji_tweets.txt",
```

	Tweets
0	RT @CalorieFixess: 🍗🌯🍔🍒 400 Calories https://t...
1	RT @1_F_I_R_S_T: _ 🍈¹⟩ Grow your account fast! ...
2	RT @LegendDeols: 👉👉👉G€T Ready to dance 💃🕺🕺🕺💃💃💃 ...
3	@britch_x Hubby's friend bought us Wendy's-che...
4	RT @DAILYPUPPIES: Workout partner ☕🍌😍 https://...

In [3]:

```python
# 1.1b
emoji_set = {'🍫', '🍡', '🍏', '🍈', '🍯', '🥠', '🍪', '🌰', '🥯', '🍐', '🌯', '🍧', '🥮', '🥚', '🍎',
'🍨', '🥣', '🍊', '🍘', '🍆', '🥓', '🌽', '🥐', '🌮', '🍬', '🍛', '🥘', '🥔', '🍿', '🍉', '🥞', '🍄', '
'🍺', '🦀', '🥦', '🥭', '🍒', '🍤', '🦐', '🥝', '🥙', '🍖', '🥜', '🍥', '🍷', '🍾', '🍲', '🧂', '🥪', '
'🍙', '🍞', '🥃', '🥗', '🍕', '🍔', '🥕', '🧁', '🍋', '🍰', '🥫', '🍗', '🥧', '🥛', '🍮', '🥡', '🍟', '
'🥥', '🍢', '🍌', '🍝', '🍵', '🍼', '🍦', '🍭', '🍑', '🍹', '🍇', '🥨', '🍱', '🦞', '☕', '🌶', '🍻', '
'🥟', '🥖', '🍶', '🍜', '🍓', '🦑', '🍩', '🌭'}

emojis = []
for tweet in data.Tweets:
    emojis.append(set(np.unique([c for c in tweet if c in emoji_set])))

data["Emojis"] = emojis
data.head()
```

	Tweets	Emojis
0	RT @CalorieFixess: 🍗 🌯 🍔 🍒 400 Calories https://t...	{🍔, 🍗, 🌯, 🍒}
1	RT @1_F_I_R_S_T: _ 🍈 ¹⟩ Grow your account fast! ...	{🍇, 🍓, 🍉, 🍍, 🍊, 🍈}
2	RT @LegendDeols: 👉 👉 👉G€T Ready to dance 💃 🕺 🕺 🕺 💃 💃 💃 ...	{🥃, 🍸}
3	@britch_x Hubby's friend bought us Wendy's-che...	{🍟, 🍔}
4	RT @DAILYPUPPIES: Workout partner ☕ 🍌 😍 https://...	{☕, 🍌}

In [4]:

```python
# 1.1c
from sklearn.preprocessing import MultiLabelBinarizer
mlb = MultiLabelBinarizer()
mlb.fit(data['Emojis'])

# (mlb.transform(data['Emojis']), mlb.classes_) # this is the data and classes after transformed
binary_data = pd.DataFrame(mlb.transform(data['Emojis']), columns=mlb.classes_)
binary_data.head()
```

	☕	🌭	🌮	🌯	🌰	🌶	🌽	🍄	🍅	🍆	...	🥭	🥮	🥯	🦀	🦐	🦑	🦞	🧀	🧁	🧂
0	0	0	0	1	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

```
5 rows × 105 columns
```

```
In [5]:  # 1.1c alternative
         # This is the alternative implementation to feed the input of apriori
         from mlxtend.preprocessing import TransactionEncoder
         te = TransactionEncoder()
         te.fit(data['Emojis'])
         bool_data = pd.DataFrame(te.transform(data['Emojis']), columns=te.columns_)
         bool_data.head()
```

	☕	🌭	🌮	🌯	🌰	🌶	🌽	🍄	🍅	🍆	...	🥭	🥮	🥯	🦀	🦐	🦑	🦞	🧀	
0	False	False	False	True	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	Fa
1	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	Fa
2	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	Fa
3	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	Fa
4	True	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	Fa

```
5 rows × 105 columns
```

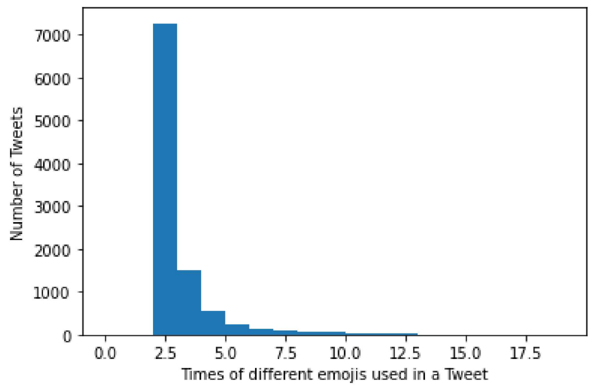
1.2 Exploratory Data Analysis (EDA)

```
In [6]:  print(f"There are {len(mlb.classes_)} different emojis used in the dataset.")

         emoji_count = [len(x) for x in data['Emojis']]
         print(f"There are {np.mean(emoji_count)} emojis used in a Tweet.")
         plt.hist(emoji_count, bins = np.arange(0, 20, 1))
         plt.xlabel("Times of different emojis used in a Tweet")
         plt.ylabel("Number of Tweets")
         plt.show()

         from collections import Counter
         counter = Counter()
         for emoji in data['Emojis']:
             counter.update(emoji)
         print(f"The most popular emojis (with their times appear in the Tweets) are {counter.most_common(5)}.")
```

```
There are 105 different emojis used in the dataset.
There are 2.6302 emojis used in a Tweet.
```



```
The most popular emojis (with their times appear in the Tweets) are [('🍻', 1819), ('🎂', 1486), ('🍔',
1384), ('🍾', 1082), ('🥂', 1031)].
```

2 The Apriori Algorithm

In [7]:
```
# 2a
from mlxtend.frequent_patterns import apriori

def emoji_frequent_itemsets(k, min_support, dataset = binary_data):
    frequent_itemsets = apriori(dataset, min_support = min_support, use_colnames=True)
    frequent_itemsets['length'] = frequent_itemsets['itemsets'].apply(lambda x: len(x))
    return frequent_itemsets[ (frequent_itemsets['length'] == k) &
                    (frequent_itemsets['support'] >= min_support) ].drop(columns=['length'])
```

In [8]:
```
# 2b
emoji_frequent_itemsets(3, 0.007)
```

```
D:\Software\Anaconda3\lib\site-packages\ipykernel\ipkernel.py:283: DeprecationWarning: `should_run_async
` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cel
l` argument and any exception that happen during thetransform in `preprocessing_exc_tuple` in IPython 7.
17 and above.
  and should_run_async(code)
D:\Software\Anaconda3\lib\site-packages\mlxtend\frequent_patterns\fpcommon.py:111: DeprecationWarning: D
ataFrames with non-bool types result in worse computationalperformance and their support might be discon
tinued in the future.Please use a DataFrame with bool type
  warnings.warn(
```

	support	itemsets
155	0.0079	(🍇, 🍊, 🍉)
156	0.0092	(🍟, 🍕, 🍔)
157	0.0070	(🍫, 🍬, 🍭)
158	0.0117	(🍹, 🍷, 🍸)
159	0.0075	(🍷, 🍺, 🍸)
160	0.0076	(🍻, 🍷, 🍸)
161	0.0072	(🥂, 🍷, 🍸)
162	0.0072	(🍹, 🍷, 🍺)
163	0.0077	(🍹, 🍷, 🍻)
164	0.0070	(🥂, 🍹, 🍷)
165	0.0076	(🥂, 🍾, 🍷)
166	0.0076	(🍹, 🍺, 🍸)
167	0.0075	(🍹, 🍻, 🍸)
168	0.0072	(🍹, 🍸, 🍾)
169	0.0079	(🥂, 🍹, 🍸)
170	0.0104	(🥂, 🍾, 🎂)

2.1 Apriori Algorithm under the Hood

In [9]:
```python
# read in the data
freq_2_data = pd.read_csv("./itemsets_data/food_emoji_frequent_2_itemsets.csv",
                                 names=["Emojis"],
                                 header=None)
freq_2_data["itemsets"] = freq_2_data["Emojis"].apply(lambda x: set(x))
freq_2_data.head()
```

```
D:\Software\Anaconda3\lib\site-packages\ipykernel\ipykernel.py:283: DeprecationWarning: `should_run_async
` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cel
l` argument and any exception that happen during thetransform in `preprocessing_exc_tuple` in IPython 7.
17 and above.
  and should_run_async(code)
```

	Emojis	itemsets
0	🌭🍔	{🌭, 🍔}
1	🌭🍕	{🌭, 🍕}
2	🍔🌮	{🌮, 🍔}
3	🍕🌮	{🌮, 🍕}
4	🍆🍑	{🍆, 🍑}

In [10]:
```python
# 2.1.1a
def generate_candidate_3_itemsets(freq_2_itemsets = list(freq_2_data["itemsets"])):
    all_items = set().union(*freq_2_itemsets)
    candidate_3_itemsets = []
    for itemset in freq_2_itemsets:
        for item in all_items:
            if item not in itemset:
                temp = itemset.copy()
                temp.add(item)
                if temp not in candidate_3_itemsets:
                    candidate_3_itemsets.append(temp)
    return candidate_3_itemsets

# generate_candidate_3_itemsets()
len(generate_candidate_3_itemsets())
```

```
D:\Software\Anaconda3\lib\site-packages\ipykernel\ipykernel.py:283: DeprecationWarning: `should_run_async
` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cel
l` argument and any exception that happen during thetransform in `preprocessing_exc_tuple` in IPython 7.
17 and above.
  and should_run_async(code)
```

```
2168
```

```
In [11]: # 2.1.1b
         import itertools
         def get_all_subsets(set, n):
             return list(itertools.combinations(set, n))

         def prune_candidate_3_itemsets(candidate_3_itemsets = generate_candidate_3_itemsets(),
                                        freq_2_itemsets = list(freq_2_data["itemsets"])):
             pruned_3_itemsets = []
             for itemset in candidate_3_itemsets:
                 subsets = get_all_subsets(itemset, 2)
                 is_candidate = True
                 for subset in subsets:
                     if set(subset) not in freq_2_itemsets:
                         is_candidate = False
                 if is_candidate:
                     pruned_3_itemsets.append(itemset)
             return pruned_3_itemsets

         # prune_candidate_3_itemsets()
         len(prune_candidate_3_itemsets())
```

```
D:\Software\Anaconda3\lib\site-packages\ipykernel\ipkernel.py:283: DeprecationWarning: `should_run_async
` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cel
l` argument and any exception that happen during thetransform in `preprocessing_exc_tuple` in IPython 7.
17 and above.
  and should_run_async(code)


88
```

2.1.2 Database Scan

```
In [12]:  # 2.1.2a
          def calculate_frequent_itemsets(min_support,
                                          candidate_itemsets = prune_candidate_3_itemsets(),
                                          dataset = binary_data):
              n = len(dataset)
              supports = []
              for itemset in tqdm(candidate_itemsets):
                  count = 0
                  for i in range(n):
                      count += np.prod([dataset[emoji][i] for emoji in itemset])
                  supports.append(count / n)

              frequent_itemsets = pd.DataFrame({
                  "support": supports,
                  "itemsets": candidate_itemsets,
              })
              return frequent_itemsets[frequent_itemsets["support"] > min_support]

          calculate_frequent_itemsets(0.001)
```

```
D:\Software\Anaconda3\lib\site-packages\ipykernel\ipkernel.py:283: DeprecationWarning: `should_run_async
` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cel
l` argument and any exception that happen during thetransform in `preprocessing_exc_tuple` in IPython 7.
17 and above.
  and should_run_async(code)
100%|██████████████████████████████████████████████████████████████████| 88/88 [00:11<00:00,
7.59it/s]
```

	support	itemsets
0	0.0040	{🌭, 🍕, 🍔}
1	0.0040	{🌮, 🍔, 🍕}
2	0.0045	{🍇, 🍓, 🍉}
3	0.0079	{🍇, 🍊, 🍉}
4	0.0061	{🍇, 🥝, 🍉}
...	...	...
83	0.0058	{🍰, 🎂, 🍫}
84	0.0029	{🎂, 🍬, 🍭}
85	0.0045	{🍰, 🍬, 🍭}
86	0.0032	{🍰, 🎂, 🍬}
87	0.0035	{🍰, 🎂, 🍭}

88 rows × 2 columns

3. Evaluating Frequent Itemsets

```
In [22]:
# 3a
from mlxtend.frequent_patterns import association_rules

frequent_itemsets = apriori(binary_data, min_support = 0.005, use_colnames=True)
associa_df = association_rules(frequent_itemsets, metric="lift", min_threshold=3)
associa_df
```

```
D:\Software\Anaconda3\lib\site-packages\ipykernel\ipkernel.py:283: DeprecationWarning: `should_run_async
` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cel
l` argument and any exception that happen during thetransform in `preprocessing_exc_tuple` in IPython 7.
17 and above.
  and should_run_async(code)
D:\Software\Anaconda3\lib\site-packages\mlxtend\frequent_patterns\fpcommon.py:111: DeprecationWarning: D
ataFrames with non-bool types result in worse computationalperformance and their support might be discon
tinued in the future.Please use a DataFrame with bool type
  warnings.warn(
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(🍩)	(☕)	0.0312	0.0589	0.0084	0.269231	4.570981	0.006562	1.287821
1	(☕)	(🍩)	0.0589	0.0312	0.0084	0.142615	4.570981	0.006562	1.129947
2	(🌭)	(🍔)	0.0193	0.1384	0.0113	0.585492	4.230435	0.008629	2.078610
3	(🍔)	(🌭)	0.1384	0.0193	0.0113	0.081647	4.230435	0.008629	1.067890
4	(🌭)	(🍕)	0.0193	0.0441	0.0050	0.259067	5.874543	0.004149	1.290131
...	...	...	...	...	...	...	...	...	...
507	(🍺, 🍸)	(🍹, 🍻)	0.0127	0.0162	0.0051	0.401575	24.788568	0.004894	1.643982
508	(🍹)	(🍻, 🍺, 🍸)	0.0615	0.0069	0.0051	0.082927	12.018381	0.004676	1.082902
509	(🍻)	(🍹, 🍺, 🍸)	0.1819	0.0076	0.0051	0.028037	3.689129	0.003718	1.021027
510	(🍺)	(🍹, 🍻, 🍸)	0.0799	0.0075	0.0051	0.063830	8.510638	0.004501	1.060170
511	(🍸)	(🍹, 🍻, 🍺)	0.0518	0.0065	0.0051	0.098456	15.147015	0.004763	1.101998

```
512 rows × 9 columns
```

```
In [24]:
# 3b
import math

def mi(antecedent_support, consequent_support, support):
    px1y1 = support
    px1 = antecedent_support
    px0 = 1 - px1
    py1 = consequent_support
    py0 = 1 - py1
    px1y0 = px1 - px1y1
    px0y1 = py1 - px1y1
    px0y0 = 1 - px1y1 - px1y0 - px0y1

    return (px1y1 * math.log2(px1y1 / (px1 * py1))) + (px1y0 * math.log2(px1y0 / (px1 * py0))) + \
           (px0y1 * math.log2(px0y1 / (px0 * py1))) + (px0y0 * math.log2(px0y0 / (px0 * py0)))
```

```
D:\Software\Anaconda3\lib\site-packages\ipykernel\ipkernel.py:283: DeprecationWarning: `should_run_async
` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cel
l` argument and any exception that happen during thetransform in `preprocessing_exc_tuple` in IPython 7.
17 and above.
  and should_run_async(code)
```

```
In [25]:
# 3c
associa_df["mutual information"] = [
    mi(associa_df["antecedent support"][i], associa_df["consequent support"][i], associa_df["support"][i])
    for i in range(len(associa_df))
]
associa_df
```

```
D:\Software\Anaconda3\lib\site-packages\ipykernel\ipkernel.py:283: DeprecationWarning: `should_run_async
` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cel
l` argument and any exception that happen during thetransform in `preprocessing_exc_tuple` in IPython 7.
17 and above.
  and should_run_async(code)
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(🍩)	(☕)	0.0312	0.0589	0.0084	0.269231	4.570981	0.006562	1.287821
1	(☕)	(🍩)	0.0589	0.0312	0.0084	0.142615	4.570981	0.006562	1.129947
2	(🌭)	(🍔)	0.0193	0.1384	0.0113	0.585492	4.230435	0.008629	2.078610
3	(🍔)	(🌭)	0.1384	0.0193	0.0113	0.081647	4.230435	0.008629	1.067890
4	(🌭)	(🍕)	0.0193	0.0441	0.0050	0.259067	5.874543	0.004149	1.290131
...	...	...	...	...	...	...	...	...	...
507	(🍺, 🍸)	(🍹, 🍻)	0.0127	0.0162	0.0051	0.401575	24.788568	0.004894	1.643982
508	(🍹)	(🍻, 🍺, 🍸)	0.0615	0.0069	0.0051	0.082927	12.018381	0.004676	1.082902
509	(🍻)	(🍹, 🍺, 🍸)	0.1819	0.0076	0.0051	0.028037	3.689129	0.003718	1.021027
510	(🍺)	(🍹, 🍻, 🍸)	0.0799	0.0075	0.0051	0.063830	8.510638	0.004501	1.060170
511	(🍸)	(🍹, 🍻, 🍺)	0.0518	0.0065	0.0051	0.098456	15.147015	0.004763	1.101998

```
512 rows × 10 columns
```

4. Itemset Similarity

4.1 Jaccard Similarity

```
In [15]:
# 4.1a
def jaccard_similarity(A, B):
    # A, B are two sets
    # assuming one of those is non-empty
    if (len(A) == 0 and len(B) == 0):
        print("Not Right Format")
        return None
    return len(A.intersection(B)) / len(A.union(B))
```

```
D:\Software\Anaconda3\lib\site-packages\ipykernel\ipkernel.py:283: DeprecationWarning: `should_run_async
` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cel
l` argument and any exception that happen during thetransform in `preprocessing_exc_tuple` in IPython 7.
17 and above.
  and should_run_async(code)
```

In [19]:

```python
# 4.1b
def get_top_similar_tweet(top_n = 5, tweet_id = 0, dataset = data):
    # top_n is the number of most similar tweets return
    # tweet_id is the target tweet id
    target = dataset["Emojis"][tweet_id]
    dataset["Similarity"] = [jaccard_similarity(target, itemset) for itemset in dataset["Emojis"]]
    return dataset.sort_values(by=["Similarity"], ascending=False).head(top_n + 1)

# The first line is the target tweet
# The second line is the most similar tweet in term of emojis
get_top_similar_tweet()
```

```
D:\Software\Anaconda3\lib\site-packages\ipykernel\ipkernel.py:283: DeprecationWarning: `should_run_async
` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cel
l` argument and any exception that happen during thetransform in `preprocessing_exc_tuple` in IPython 7.
17 and above.
  and should_run_async(code)
```

	Tweets	Emojis	Similarity
0	RT @CalorieFixess: 🍗 🌯 🍔 🍒 400 Calories https://t...	{🍔, 🍗, 🌯, 🍒}	1.00
6800	RT @levelscafeabuja: Chow! 🤩 💦 🍗 🌯 🍔 #LevelsCafeAb...	{🍔, 🌯, 🍗}	0.75
5334	RT @thatssochioma: You don’ t want to miss this...	{🍔, 🍗}	0.50
3466	@jewishmuseummd @americanart I'm definitely in...	{🍗, 🍒}	0.50
7877	@tafarireid07 Did you say bbq? 🔥 🍔 🍗 🚙	{🍔, 🍗}	0.50
7692	RT @WVUfootball: We have some new digs! 🍴 🍗 🍔 #H...	{🍔, 🍗}	0.50

According to the result above, we can see that the most similar tweet (the second line) shares the most common contents with the target tweet (the first line) in terms of emojis. For example, take the first line of tweet as target (as shown above), we can see that the most similar tweet have 3 same emojis used as the target ({🍔, 🌯, 🍗}).

As a result, the Jaccard similarity function is a simple but powerful indicator when measuring the similarity between two entries.

In [ ]: