In [194]:
```
import json
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import pearsonr
from scipy.stats import spearmanr
```

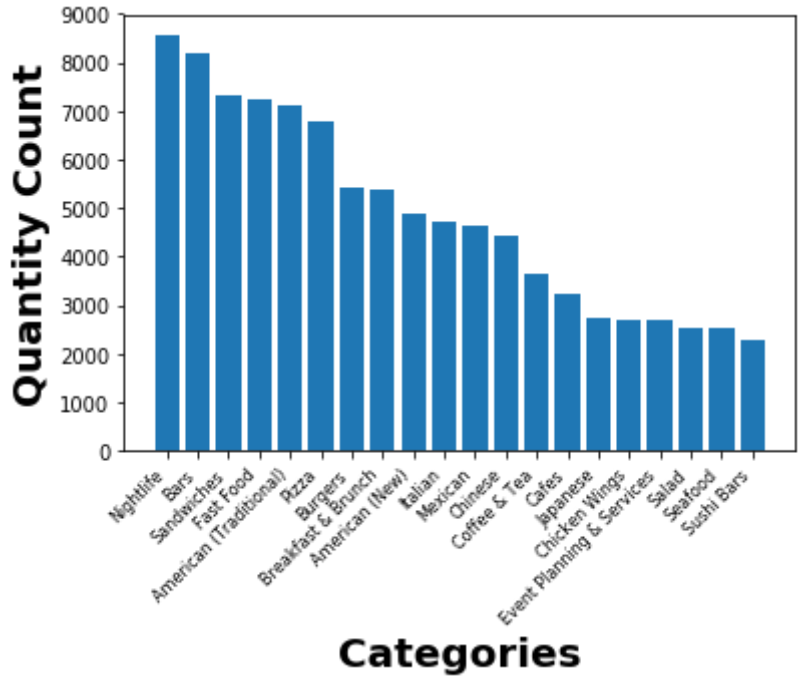
In [39]:
```
# load business.json
filepath = '../../yelp_dataset'

business = []
for l in open(filepath+"/business.json", encoding="utf8").readlines():
    business.append(json.loads(l))
df_business = pd.DataFrame.from_records(business)
```

In [40]:
```
# load data from previous preprocessing/EDA
filepath = '../../'

df_restaurant_tips = pd.read_json(filepath+"restaurant_tips.json", encoding
```

In [41]:
```
# Filter businesses that are only in the 'Restaurant' category
def check_for_rest(row):
    category = row['categories']
    if category:
        tokens = category.split(', ')
        return 'Restaurants' in tokens
    return False

df_business['is_restaurant'] = df_business.apply(check_for_rest, axis=1)

food_businesses = df_business[df_business['is_restaurant'] == True]

# Get the unique IDs for all the businesses that are resturaunts
restaurant_ids  = set(food_businesses['business_id'].unique())

print('Total Unique ID count:',len(restaurant_ids))
```

```
Total Unique ID count: 59371
```

In [42]:
```
categories_series = food_businesses['categories']

cuisine_counts = {}
for _, categories in categories_series.iteritems():
    tokens = categories.split(', ')
    for category in tokens:
        if category == 'Restaurants'or category == 'Food':
            do = 'nothing'
        elif category in cuisine_counts:
            cuisine_counts[category] += 1
        else:
            cuisine_counts[category] = 1

sorted_cuisine_counts = {k: v for k, v in sorted(cuisine_counts.items(), ke

top_20 = dict(list(sorted_cuisine_counts.items())[-20:])

x = list(top_20.keys())
x.reverse()
x = np.array(x)
y = list(top_20.values())
y.reverse()
y = np.array(y)


fig, ax = plt.subplots()
plt.bar(x, y)
plt.xticks(x, x, color='black', rotation=45, fontsize='8', horizontalalignm
plt.xlabel("Categories", fontweight='bold', fontsize='20')
plt.ylabel("Quantity Count", fontweight='bold', fontsize='20')

plt.show()
```



In [43]:
```
# Modifying DF['categories'] to make filtering more efficient
# def split_category(row):
#     return row['categories'].split(',  ')

# food_businesses['categories'] = food_businesses.apply(split_category, axi
```

In [44]:
```
# CUISINES WE WILL EXPLORE:
# Fast Food, American (Traditional), American (New), Italian, Mexican, Chin
def is_fast_food(row):
    category = row['categories']
    if category:
        tokens = category.split(', ')
        return 'Fast Food' in tokens
    return False

def is_american_traditional(row):
    category = row['categories']
    if category:
        tokens = category.split(', ')
        return 'American (Traditional)' in tokens
    return False

def is_american_new(row):
    category = row['categories']
    if category:
        tokens = category.split(', ')
        return 'American (New)' in tokens
    return False

# Fast Food
food_businesses['is_fast_food'] = food_businesses.apply(is_fast_food, axis=
df_fast_food = food_businesses[food_businesses['is_fast_food'] == True]

# American (Traditional)
food_businesses['is_american_t'] = food_businesses.apply(is_american_tradit
df_american_t = food_businesses[food_businesses['is_american_t'] == True]

# American (New)
food_businesses['is_american_n'] = food_businesses.apply(is_american_new, a
df_american_n = food_businesses[food_businesses['is_american_n'] == True]
```

```
c:\users\casey\appdata\local\programs\python\python37\lib\site-packages\i
pykernel_launcher.py:25: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a
-view-versus-a-copy)
c:\users\casey\appdata\local\programs\python\python37\lib\site-packages\i
pykernel_launcher.py:29: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a
-view-versus-a-copy)
c:\users\casey\appdata\local\programs\python\python37\lib\site-packages\i
pykernel_launcher.py:33: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: http://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a
-view-versus-a-copy)
```

In [45]:
```
# get tips for Fast Food mapped to business_id
fast_food_ids = set(df_fast_food.business_id.unique())
american_t_ids = set(df_american_t.business_id.unique())
american_n_ids = set(df_american_n.business_id.unique())

# group tips

cuisine_tips_fast_food = df_restaurant_tips[df_restaurant_tips['business_id
cuisine_tips_american_t = df_restaurant_tips[df_restaurant_tips['business_i
cuisine_tips_american_n = df_restaurant_tips[df_restaurant_tips['business_i
```

In [184]:
```
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer

analyser = SentimentIntensityAnalyzer()

def get_sentiment(sentence):
    dic = analyser.polarity_scores(sentence)
    # dic -> {'neg': 0.778, 'neu': 0.222, 'pos': 0.0, 'compound': -0.5423}
    # for some reason only allows dictionary return type
    return dic

def get_positive_sentiment(row):
    sentence = row['text']
    dic = get_sentiment(sentence)
    return dic['compound'] #score is normalized between 0-1
```

In [185]:
```
# Find sentiment analysis for each tip
cuisine_tips_fast_food['text_sentiment'] = cuisine_tips_fast_food.apply(get
cuisine_tips_american_t['text_sentiment'] = cuisine_tips_american_t.apply(g
cuisine_tips_american_n['text_sentiment'] = cuisine_tips_american_n.apply(g
```

```
/Users/leannahue/Library/Python/3.7/lib/python/site-packages/ipykernel_la
uncher.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a
-view-versus-a-copy)

/Users/leannahue/Library/Python/3.7/lib/python/site-packages/ipykernel_la
uncher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a
-view-versus-a-copy)
  This is separate from the ipykernel package so we can avoid doing impor
ts until
/Users/leannahue/Library/Python/3.7/lib/python/site-packages/ipykernel_la
uncher.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a
-view-versus-a-copy)
  after removing the cwd from sys.path.
```

In [48]:
```
#seperate into meal times
breakfast_times = set([5,6,7,8,9,10])
lunch_times = set([11,12,13,14,15,16])
dinner_time = set([17,18,19,20,21,22])

#fast food tips
fast_food_tips_breakfast = cuisine_tips_fast_food[cuisine_tips_fast_food['l
fast_food_tips_lunch = cuisine_tips_fast_food[cuisine_tips_fast_food['local
fast_food_tips_dinner = cuisine_tips_fast_food[cuisine_tips_fast_food['loca

#american traditional tips
american_t_tips_breakfast = cuisine_tips_american_t[cuisine_tips_american_t
american_t_tips_lunch = cuisine_tips_american_t[cuisine_tips_american_t['lo
american_t_tips_dinner = cuisine_tips_american_t[cuisine_tips_american_t['l

#american new tips
american_n_tips_breakfast = cuisine_tips_american_n[cuisine_tips_american_n
american_n_tips_lunch = cuisine_tips_american_n[cuisine_tips_american_n['lo
american_n_tips_dinner = cuisine_tips_american_n[cuisine_tips_american_n['l
```

In [50]:
```
def find_avg_tip_sentiment(row, df_tips):
    business_id = row['business_id']
    tips_with_this_business = df_tips[df_tips['business_id'] == business_id
    return tips_with_this_business['text_sentiment'].mean()
```

In [103]:
```
# Find average sentiment for each business separated by meal times

# Fast food businesses
fast_food_businesses_breakfast = pd.DataFrame({'business_id': list(fast_foo
fast_food_businesses_breakfast['avg_tip_sentiment'] = \
    fast_food_businesses_breakfast.apply(lambda row: find_avg_tip_sentiment

fast_food_businesses_lunch = pd.DataFrame({'business_id': list(fast_food_ti
fast_food_businesses_lunch['avg_tip_sentiment'] = \
    fast_food_businesses_lunch.apply(lambda row: find_avg_tip_sentiment(row

fast_food_businesses_dinner = pd.DataFrame({'business_id': list(fast_food_t
fast_food_businesses_dinner['avg_tip_sentiment'] = \
    fast_food_businesses_dinner.apply(lambda row: find_avg_tip_sentiment(ro

# American traditional businesses
american_t_businesses_breakfast = pd.DataFrame({'business_id': list(america
american_t_businesses_breakfast['avg_tip_sentiment'] = \
    american_t_businesses_breakfast.apply(lambda row: find_avg_tip_sentimen

american_t_businesses_lunch = pd.DataFrame({'business_id': list(american_t_
american_t_businesses_lunch['avg_tip_sentiment'] = \
    american_t_businesses_lunch.apply(lambda row: find_avg_tip_sentiment(ro

american_t_businesses_dinner = pd.DataFrame({'business_id': list(american_t
american_t_businesses_dinner['avg_tip_sentiment'] = \
    american_t_businesses_dinner.apply(lambda row: find_avg_tip_sentiment(r

# American new businesses
american_n_businesses_breakfast = pd.DataFrame({'business_id': list(america
american_n_businesses_breakfast['avg_tip_sentiment'] = \
    american_n_businesses_breakfast.apply(lambda row: find_avg_tip_sentimen

american_n_businesses_lunch = pd.DataFrame({'business_id': list(american_n_
american_n_businesses_lunch['avg_tip_sentiment'] = \
    american_n_businesses_lunch.apply(lambda row: find_avg_tip_sentiment(ro

american_n_businesses_dinner = pd.DataFrame({'business_id': list(american_n
american_n_businesses_dinner['avg_tip_sentiment'] = \
    american_n_businesses_dinner.apply(lambda row: find_avg_tip_sentiment(r
```

In [52]:
```
# Find average sentiment for each meal time and cuisine

#fast food tips
fast_food_breakfast_avg_sentiment = fast_food_tips_breakfast['text_sentimen
fast_food_lunch_avg_sentiment = fast_food_tips_lunch['text_sentiment'].mean
fast_food_dinner_avg_sentiment = fast_food_tips_dinner['text_sentiment'].me
print('Fast Food Average Sentiment')
print('Breakfast', fast_food_breakfast_avg_sentiment)
print('Lunch', fast_food_lunch_avg_sentiment)
print('Dinner', fast_food_dinner_avg_sentiment, '\n')

#american traditional tips
american_t_breakfast_avg_sentiment = american_t_tips_breakfast['text_sentim
american_t_lunch_avg_sentiment = american_t_tips_lunch['text_sentiment'].me
american_t_dinner_avg_sentiment = american_t_tips_dinner['text_sentiment'].
print('American(Traditional) Average Sentiment')
print('Breakfast', american_t_breakfast_avg_sentiment)
print('Lunch', american_t_lunch_avg_sentiment)
print('Dinner', american_t_dinner_avg_sentiment, '\n')

#american new tips
american_n_breakfast_avg_sentiment = american_n_tips_breakfast['text_sentim
american_n_lunch_avg_sentiment = american_n_tips_lunch['text_sentiment'].me
american_n_dinner_avg_sentiment = american_n_tips_dinner['text_sentiment'].
print('American (New) Average Sentiment')
print('Breakfast', american_n_breakfast_avg_sentiment)
print('Lunch', american_n_lunch_avg_sentiment)
print('Dinner', american_n_dinner_avg_sentiment)
```

```
Fast Food Average Sentiment
Breakfast 0.2213416063426405
Lunch 0.2443311774930272
Dinner 0.23666857515244616

American(Traditional) Average Sentiment
Breakfast 0.2885007468259895
Lunch 0.2877473850913377
Dinner 0.2789951294708015

American (New) Average Sentiment
Breakfast 0.28987180729563233
Lunch 0.29474464157522307
Dinner 0.2924089559172432
```

In [66]:
```
# find averge sentiment per business id based on tips for each meal time

# FAST FOOD
fast_food_breakfast_ids = fast_food_tips_breakfast['business_id'].unique()

fast_food_breakfast_ids_to_sentiment = {}
for ID in fast_food_breakfast_ids:
    temp = fast_food_tips_breakfast[fast_food_tips_breakfast['business_id']
    fast_food_breakfast_ids_to_sentiment[ID] = temp['text_sentiment'].mean(


fast_food_lunch_ids = fast_food_tips_lunch['business_id'].unique()

fast_food_lunch_ids_to_sentiment = {}
for ID in fast_food_lunch_ids:
    temp = fast_food_tips_lunch[fast_food_tips_lunch['business_id'] == ID]
    fast_food_lunch_ids_to_sentiment[ID] = temp['text_sentiment'].mean()


fast_food_dinner_ids = fast_food_tips_dinner['business_id'].unique()

fast_food_dinner_ids_to_sentiment = {}
for ID in fast_food_dinner_ids:
    temp = fast_food_tips_dinner[fast_food_tips_dinner['business_id'] == ID
    fast_food_dinner_ids_to_sentiment[ID] = temp['text_sentiment'].mean()
```

In [65]:
```
# AMERICAN TRADITIONAL
american_t_breakfast_ids = american_t_tips_breakfast['business_id'].unique(

american_t_breakfast_ids_to_sentiment = {}
for ID in american_t_tips_breakfast:
    temp = american_t_tips_breakfast[american_t_tips_breakfast['business_id
    american_t_breakfast_ids_to_sentiment[ID] = temp['text_sentiment'].mean


american_t_lunch_ids = american_t_tips_lunch['business_id'].unique()

american_t_lunch_ids_to_sentiment = {}
for ID in american_t_tips_lunch:
    temp = american_t_tips_lunch[american_t_tips_lunch['business_id'] == ID
    american_t_lunch_ids_to_sentiment[ID] = temp['text_sentiment'].mean()


american_t_dinner_ids = american_t_tips_dinner['business_id'].unique()

american_t_dinner_ids_to_sentiment = {}
for ID in american_t_tips_dinner:
    temp = american_t_tips_dinner[american_t_tips_dinner['business_id'] ==
    american_t_dinner_ids_to_sentiment[ID] = temp['text_sentiment'].mean()
```

In [64]:
```
# AMERICAN NEW
american_n_breakfast_ids = american_n_tips_breakfast['business_id'].unique(

american_n_breakfast_ids_to_sentiment = {}
for ID in american_n_tips_breakfast:
    temp = american_n_tips_breakfast[american_n_tips_breakfast['business_id
    american_n_breakfast_ids_to_sentiment[ID] = temp['text_sentiment'].mean


american_n_lunch_ids = american_n_tips_lunch['business_id'].unique()

american_n_lunch_ids_to_sentiment = {}
for ID in american_n_tips_lunch:
    temp = american_n_tips_lunch[american_n_tips_lunch['business_id'] == ID
    american_n_lunch_ids_to_sentiment[ID] = temp['text_sentiment'].mean()


american_n_dinner_ids = american_n_tips_dinner['business_id'].unique()

american_n_dinner_ids_to_sentiment = {}
for ID in american_n_tips_dinner:
    temp = american_n_tips_dinner[american_n_tips_dinner['business_id'] ==
    american_n_dinner_ids_to_sentiment[ID] = temp['text_sentiment'].mean()
```

In [169]:
```
# Romantic, Intimate, Hipster, Classy
# Create Dataset for breakfast first
count = 0
count_romantic = 0
def get_features(row):
    try:
        attributes = list(row['attributes'])[0]
        business_id = list(row['business_id'])[0]

        ambience = attributes['Ambience'].replace("\'", "\"").lower()
        ambience = json.loads(ambience)

        romantic = bool_to_bit(ambience['romantic'])
        hipster = bool_to_bit(ambience['hipster'])
        classy = bool_to_bit(ambience['classy'])
        casual = bool_to_bit(ambience['casual'])

        return [romantic, hipster, classy, casual]

    except:
        return None

def bool_to_bit(x):
    return int(x == True)
```

In [ ]:
```
# GET FEATURE FOR FAST_FOOD_BREAKFAST
nan_value = float("NaN")
fast_food_businesses_breakfast['Romantic'] = nan_value
fast_food_businesses_breakfast['Hipster'] = nan_value
fast_food_businesses_breakfast['Classy'] = nan_value
fast_food_businesses_breakfast['Casual'] = nan_value
for ID in fast_food_breakfast_ids:
    row = df_fast_food.loc[df_fast_food['business_id'] == ID]
    features = get_features(row)

    if features != None:
        fast_food_businesses_breakfast.loc[fast_food_businesses_breakfast['
        fast_food_businesses_breakfast.loc[fast_food_businesses_breakfast['
        fast_food_businesses_breakfast.loc[fast_food_businesses_breakfast['
        fast_food_businesses_breakfast.loc[fast_food_businesses_breakfast['

# GET FEATURE FOR FAST_FOOD_LUNCH
nan_value = float("NaN")
fast_food_businesses_lunch['Romantic'] = nan_value
fast_food_businesses_lunch['Hipster'] = nan_value
fast_food_businesses_lunch['Classy'] = nan_value
fast_food_businesses_lunch['Casual'] = nan_value
for ID in fast_food_lunch_ids:
    row = df_fast_food.loc[df_fast_food['business_id'] == ID]
    features = get_features(row)

    if features != None:
        fast_food_businesses_lunch.loc[fast_food_businesses_lunch['business
        fast_food_businesses_lunch.loc[fast_food_businesses_lunch['business
        fast_food_businesses_lunch.loc[fast_food_businesses_lunch['business
        fast_food_businesses_lunch.loc[fast_food_businesses_lunch['business

# GET FEATURE FOR FAST_FOOD_DINNER
nan_value = float("NaN")
fast_food_businesses_dinner['Romantic'] = nan_value
fast_food_businesses_dinner['Hipster'] = nan_value
fast_food_businesses_dinner['Classy'] = nan_value
fast_food_businesses_dinner['Casual'] = nan_value
for ID in fast_food_dinner_ids:
    row = df_fast_food.loc[df_fast_food['business_id'] == ID]
    features = get_features(row)

    if features != None:
        fast_food_businesses_dinner.loc[fast_food_businesses_dinner['busine
        fast_food_businesses_dinner.loc[fast_food_businesses_dinner['busine
        fast_food_businesses_dinner.loc[fast_food_businesses_dinner['busine
        fast_food_businesses_dinner.loc[fast_food_businesses_dinner['busine
```

In [ ]:
```
# GET FEATURES FOR AMERICAN TRADITIONAL BREAKFAST
nan_value = float("NaN")
american_t_businesses_breakfast['Romantic'] = nan_value
american_t_businesses_breakfast['Hipster'] = nan_value
american_t_businesses_breakfast['Classy'] = nan_value
american_t_businesses_breakfast['Casual'] = nan_value
for ID in american_t_breakfast_ids:
    row = df_american_t.loc[df_american_t['business_id'] == ID]
    features = get_features(row)
    if features != None:
        american_t_businesses_breakfast.loc[american_t_businesses_breakfast
        american_t_businesses_breakfast.loc[american_t_businesses_breakfast
        american_t_businesses_breakfast.loc[american_t_businesses_breakfast
        american_t_businesses_breakfast.loc[american_t_businesses_breakfast

# GET FEATURES FOR AMERICAN TRADITIONAL LUNCH
nan_value = float("NaN")
american_t_businesses_lunch['Romantic'] = nan_value
american_t_businesses_lunch['Hipster'] = nan_value
american_t_businesses_lunch['Classy'] = nan_value
american_t_businesses_lunch['Casual'] = nan_value
for ID in american_t_lunch_ids:
    row = df_american_t.loc[df_american_t['business_id'] == ID]
    features = get_features(row)
    if features != None:
        american_t_businesses_lunch.loc[american_t_businesses_lunch['busine
        american_t_businesses_lunch.loc[american_t_businesses_lunch['busine
        american_t_businesses_lunch.loc[american_t_businesses_lunch['busine
        american_t_businesses_lunch.loc[american_t_businesses_lunch['busine

# GET FEATURE FOR AMERICAN TRADITIONAL DINNER
nan_value = float("NaN")
american_t_businesses_dinner['Romantic'] = nan_value
american_t_businesses_dinner['Hipster'] = nan_value
american_t_businesses_dinner['Classy'] = nan_value
american_t_businesses_dinner['Casual'] = nan_value
for ID in american_t_dinner_ids:
    row = df_american_t.loc[df_american_t['business_id'] == ID]
    features = get_features(row)

    if features != None:
        american_t_businesses_dinner.loc[american_t_businesses_dinner['busi
        american_t_businesses_dinner.loc[american_t_businesses_dinner['busi
        american_t_businesses_dinner.loc[american_t_businesses_dinner['busi
        american_t_businesses_dinner.loc[american_t_businesses_dinner['busi
```

In [ ]:
```
ET FEATURES FOR AMERICAN NEW BREAKFAST
_value = float("NaN")
rican_n_businesses_breakfast['Romantic'] = nan_value
rican_n_businesses_breakfast['Hipster'] = nan_value
rican_n_businesses_breakfast['Classy'] = nan_value
rican_n_businesses_breakfast['Casual'] = nan_value
 ID in american_n_breakfast_ids:
 row = df_american_n.loc[df_american_n['business_id'] == ID]
 features = get_features(row)
 if features != None:
     american_n_businesses_breakfast.loc[american_n_businesses_breakfast['bu
     american_n_businesses_breakfast.loc[american_n_businesses_breakfast['bu
     american_n_businesses_breakfast.loc[american_n_businesses_breakfast['bu
     american_n_businesses_breakfast.loc[american_n_businesses_breakfast['bu

ET FEATURES FOR AMERICAN NEW LUNCH
_value = float("NaN")
rican_n_businesses_lunch['Romantic'] = nan_value
rican_n_businesses_lunch['Hipster'] = nan_value
rican_n_businesses_lunch['Classy'] = nan_value
rican_n_businesses_lunch['Casual'] = nan_value
 ID in american_n_lunch_ids:
 row = df_american_n.loc[df_american_n['business_id'] == ID]
 features = get_features(row)
 if features != None:
     american_n_businesses_lunch.loc[american_n_businesses_lunch['business_i
     american_n_businesses_lunch.loc[american_n_businesses_lunch['business_i
     american_n_businesses_lunch.loc[american_n_businesses_lunch['business_i
     american_n_businesses_lunch.loc[american_n_businesses_lunch['business_i

ET FEATURES FOR AMERICAN NEW DINNER
_value = float("NaN")
rican_n_businesses_dinner['Romantic'] = nan_value
rican_n_businesses_dinner['Hipster'] = nan_value
rican_n_businesses_dinner['Classy'] = nan_value
rican_n_businesses_dinner['Casual'] = nan_value
 ID in american_n_dinner_ids:
 row = df_american_n.loc[df_american_n['business_id'] == ID]
 features = get_features(row)
 if features != None:
     american_n_businesses_dinner.loc[american_n_businesses_dinner['business
     american_n_businesses_dinner.loc[american_n_businesses_dinner['business
     american_n_businesses_dinner.loc[american_n_businesses_dinner['business
     american_n_businesses_dinner.loc[american_n_businesses_dinner['business
```

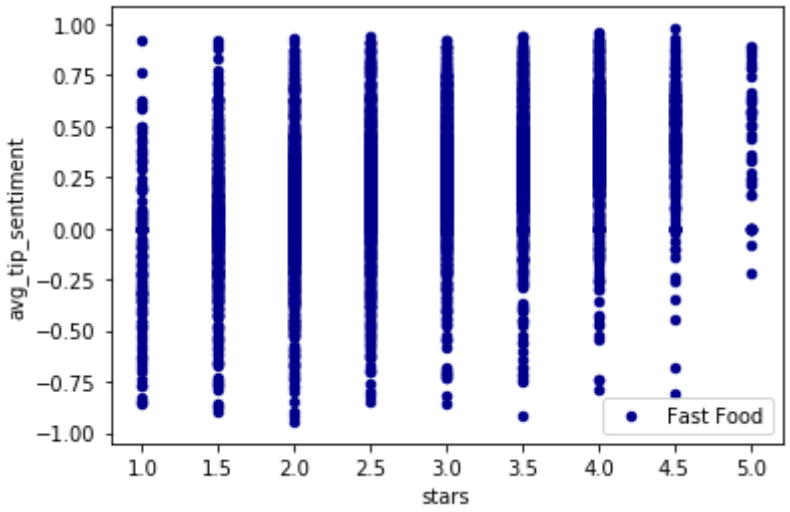
In [170]:
```
print(american_t_businesses_dinner['Classy'].value_counts())
print(american_t_businesses_dinner['Romantic'].value_counts())
print(american_t_businesses_dinner['Hipster'].value_counts())
print(american_t_businesses_dinner['Casual'].value_counts())
```

```
0.0    4975
1.0     125
Name: Classy, dtype: int64
0.0    5046
1.0      54
Name: Romantic, dtype: int64
0.0    5010
1.0      90
Name: Hipster, dtype: int64
1.0    3426
0.0    1674
Name: Casual, dtype: int64
```

In [188]:
```
def find_star_rating(row, businesses):
    business_id = row['business_id']
    business_info = businesses[businesses['business_id'] == business_id]
    return list(business_info['stars'])[0]
```

In [189]:
```
fast_food_businesses_star_sentiment = pd.DataFrame({'business_id': list(cui
fast_food_businesses_star_sentiment['avg_tip_sentiment'] = \
    fast_food_businesses_star_sentiment.apply(lambda row: find_avg_tip_sent
fast_food_businesses_star_sentiment['stars'] = \
    fast_food_businesses_star_sentiment.apply(lambda row: find_star_rating(
```

In [190]:
```
ax1 = fast_food_businesses_star_sentiment.plot.scatter(x='stars', \
    y='avg_tip_sentiment', c='DarkBlue', label='Fast Food')
```



In [195]:
```
pearson_corr, _ = pearsonr(fast_food_businesses_star_sentiment['avg_tip_sen
print('Pearson', pearson_corr)

spearman_corr, _ = spearmanr(fast_food_businesses_star_sentiment['avg_tip_s
print('Spearman', spearman_corr)
```

```
Pearson 0.391347743246476
Spearman 0.4101010110870563
```

In [ ]:
```

```