Import Packages

In [1]:
```
import pandas as pd
import plotly.express as px
import seaborn as sns
from plotly import tools
from wordcloud import WordCloud,STOPWORDS
import plotly.graph_objects as go
from plotly.offline import iplot
from collections import defaultdict
from textblob import TextBlob
from tqdm import tqdm
tqdm.pandas()
```

Import Data

In [2]:
```
games = pd.read_pickle('games_v2.pkl')
games['review_date_pd'] = pd.to_datetime(games['review_date'])
games['review_date_pd'] = games['review_date_pd'].dt.to_period('Y')
games['review_date_pd'] = games['review_date_pd'].astype(str)
games['review_date_pd'] = games['review_date_pd'].astype(int)
```

Data Info

In [3]:
```
games.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1780154 entries, 0 to 1780267
Data columns (total 19 columns):
 #   Column             Dtype 
---  ------             ----- 
 0   marketplace        object
 1   customer_id        int64 
 2   review_id          object
 3   product_id         object
 4   product_parent     int64 
 5   product_title      object
 6   product_category   object
 7   star_rating        int64 
 8   helpful_votes      int64 
 9   total_votes        int64 
 10  vine               object
 11  verified_purchase  object
 12  review_headline    object
 13  review_body        object
 14  review_date        object
 15  review_full        object
 16  Sentiment_target   object
 17  review_clean       object
 18  review_date_pd     int64 
dtypes: int64(6), object(13)
memory usage: 271.6+ MB
```

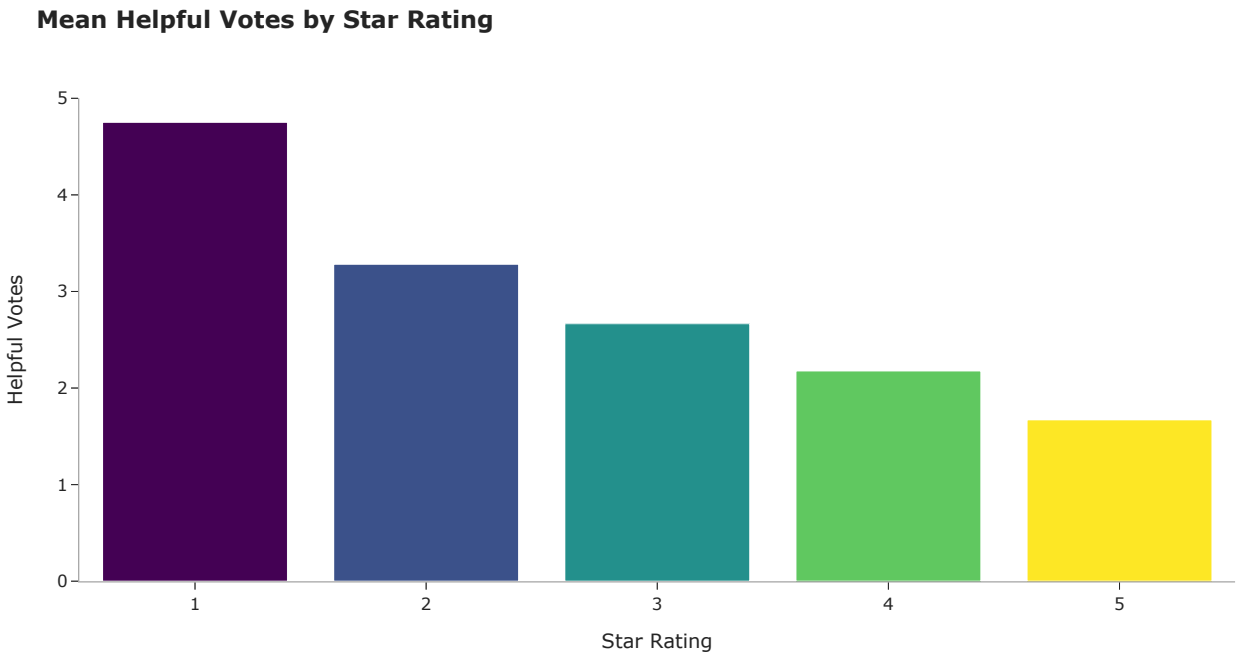
Type *Markdown* and LaTeX: α^2

Mean Helpful Votes by Star Rating

In [4]:
```
games.groupby('star_rating')['helpful_votes'].mean()
```

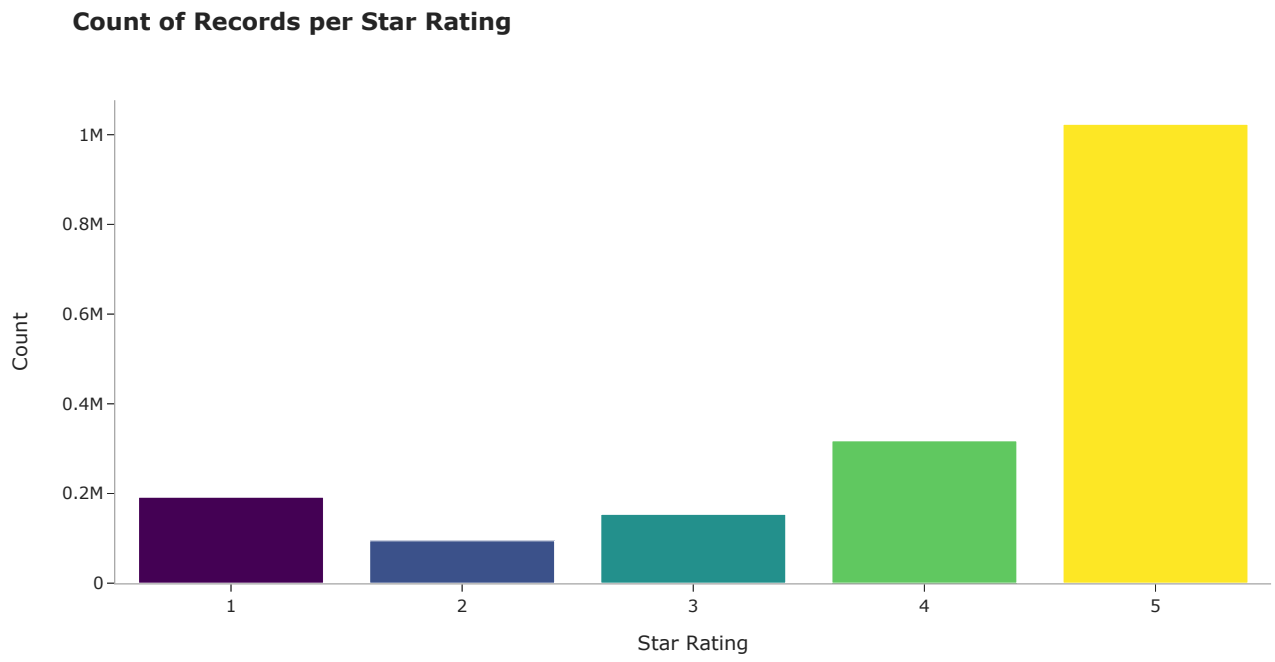
Out[4]:
```
star_rating
1    4.751093
2    3.281358
3    2.666104
4    2.177043
5    1.670852
Name: helpful_votes, dtype: float64
```

In [5]:
```
mean_helpful_votes_by_star_ratings_df = pd.DataFrame(games.groupby('star_rating')['helpful_votes'].mean()).reset_index()

helpful_votes_by_star_rating_mean_fig = px.bar(
    mean_helpful_votes_by_star_ratings_df,
    x='star_rating',
    y='helpful_votes',
    color='star_rating',
    labels = {
        'Sentiment_target': 'Sentiment Target',
        'helpful_votes':'Helpful Votes',
        'star_rating':'Star Rating'
    },
    title = '<b>Mean Helpful Votes by Star Rating</b>',
    template='simple_white',
)

helpful_votes_by_star_rating_mean_fig.update_traces(marker_coloraxis=None)

helpful_votes_by_star_rating_mean_fig.show()

helpful_votes_by_star_rating_mean_fig.write_image("helpful_votes_by_star_rating_mean_fig.jpeg")
```



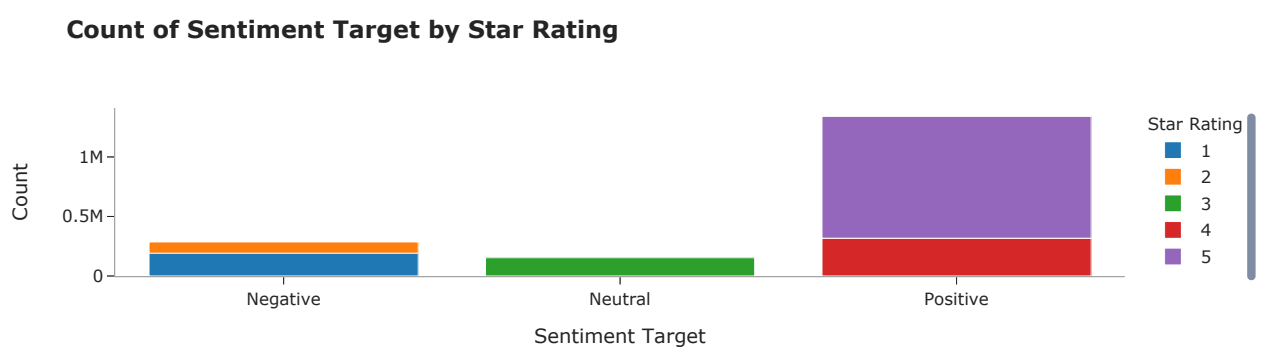
Get Count of Records by Star Ratings

In [6]:
```
star_rating_counts_fig = px.bar(
    games.star_rating.value_counts().reset_index(),
    y='star_rating',
    x='index',
    color='index',
    labels = {
        'index': 'Star Rating',
        'star_rating':'Count',
    },
    title = '<b>Count of Records per Star Rating</b>',
    template='simple_white',

)

star_rating_counts_fig.update_traces(marker_coloraxis=None)

star_rating_counts_fig.show()

star_rating_counts_fig.write_image("star_rating_counts_fig.jpeg")
```



Sentiment Star Rating

In [7]:
```
sentiment_star_counts = pd.DataFrame(games.groupby(['Sentiment_target', 'star_rating']).size().reset_index())
sentiment_star_counts.rename(columns = {0:'Count'}, inplace=True)

sentiment_star_counts['star_rating'] = sentiment_star_counts['star_rating'].astype(str)

sentiment_star_counts_fig = px.bar(
    sentiment_star_counts,
    x='Sentiment_target',
    y='Count',
    color='star_rating',
    labels = {
        'Sentiment_target': 'Sentiment Target',
        'star_rating':'Star Rating'
    },
    title = '<b>Count of Sentiment Target by Star Rating</b>',
    template='simple_white',
    width=900,
    height=300,
)

sentiment_star_counts_fig.show()


sentiment_star_counts_fig.write_image("sentiment_star_counts_fig.jpeg")
```

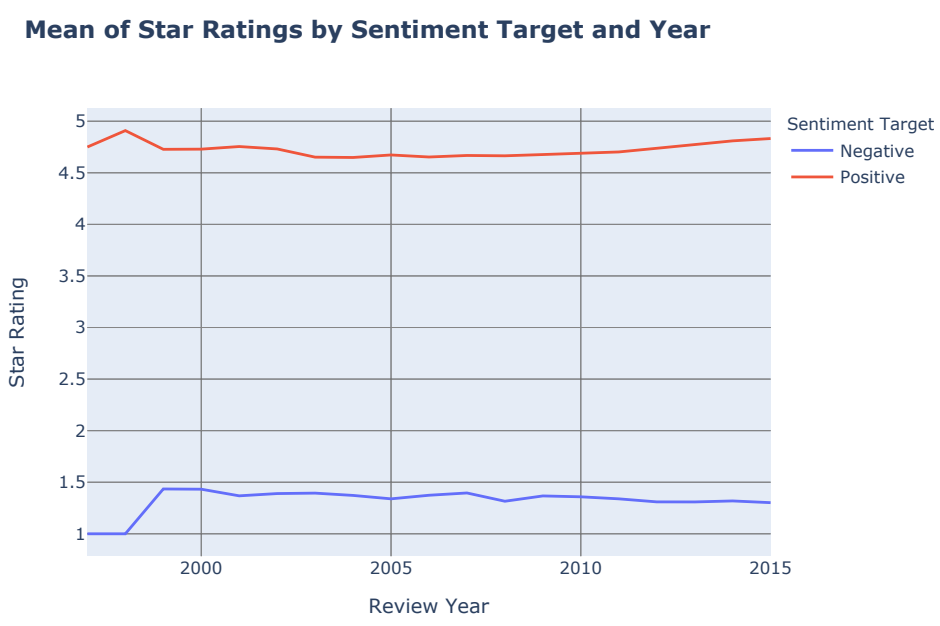


Plot Data by Date

In [8]:
```
date_target_star_mean = pd.DataFrame(
    games.groupby(
        [
            'Sentiment_target',
            'review_date_pd',
        ]
    )['star_rating'].mean().reset_index()
)
```

Mean of Star Ratings by Sentiment Target and Year

In [9]:
```
date_target_star_mean = date_target_star_mean[
    ~date_target_star_mean['Sentiment_target'].str.contains('eu')
]
date_target_star_mean_fig = px.line(
    date_target_star_mean,
    x='review_date_pd',
    y='star_rating',
    color='Sentiment_target',
    labels = {
        'Sentiment_target': 'Sentiment Target',
        'star_rating':'Star Rating',
        'review_date_pd':"Review Year"
    },
    title = '<b>Mean of Star Ratings by Sentiment Target and Year</b>',
    width=700,
    height=500,
)

date_target_star_mean_fig.show()
```



Barchart Plots

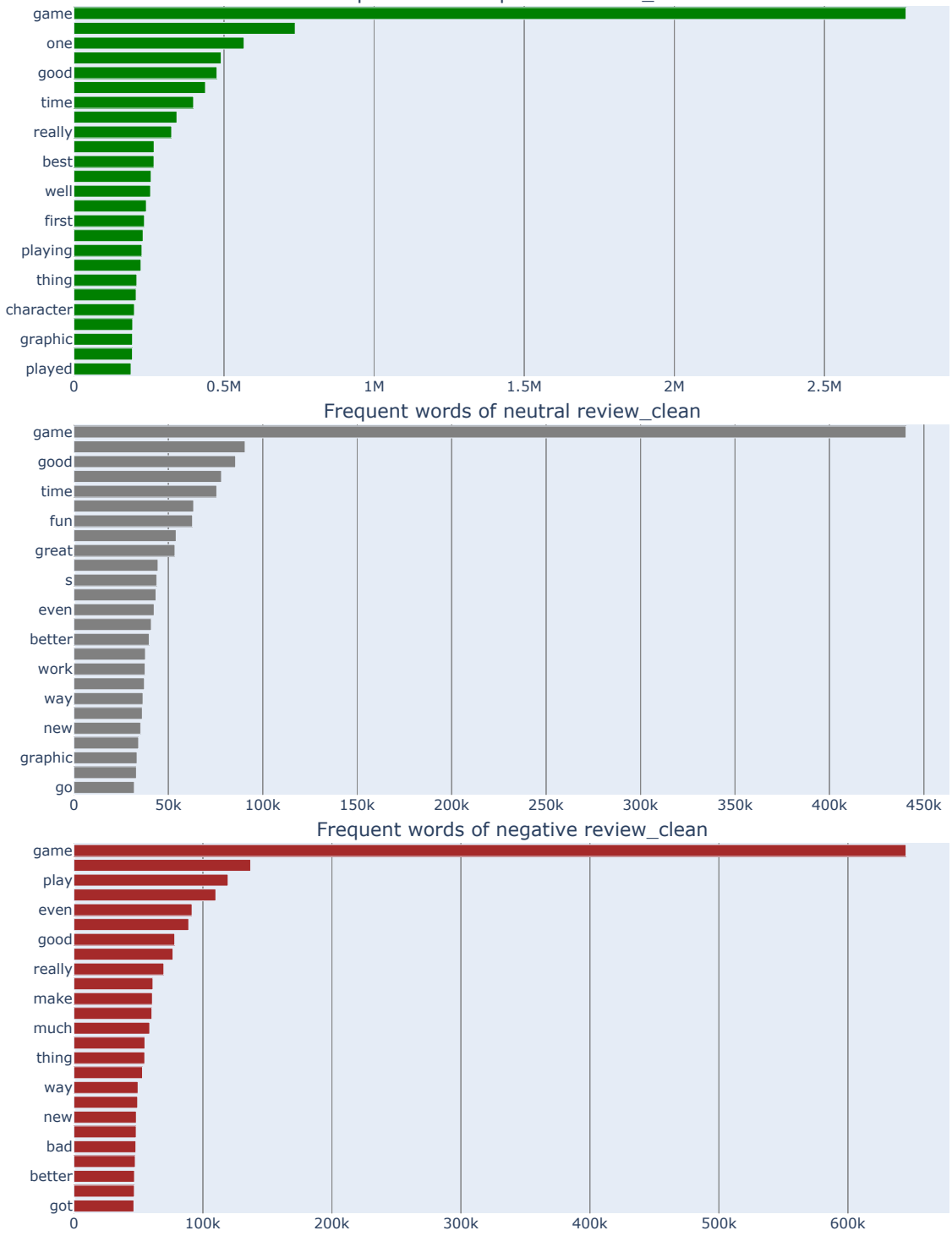
In [10]:
```
# Get Text Blobs
games['polarity'] = games['review_clean'].progress_map(lambda text: TextBlob(text).sentiment.polarity)
games['review_len'] = games['review_clean'].astype(str).progress_apply(len)
games['word_count'] = games['review_clean'].progress_apply(lambda x: len(str(x).split()))

#Filtering data
review_pos = games[games["Sentiment_target"]=='Positive'].dropna()
review_neu = games[games["Sentiment_target"]=='Neutral'].dropna()
review_neg = games[games["Sentiment_target"]=='Negative'].dropna()

## custom function for ngram generation ##
def generate_ngrams(text, n_gram=1):
    token = [token for token in text.lower().split(" ") if token != "" if token not in STOPWORDS]
    ngrams = zip(*[token[i:] for i in range(n_gram)])
    return [" ".join(ngram) for ngram in ngrams]

## custom function for horizontal bar chart ##
def horizontal_bar_chart(df, color):
    trace = go.Bar(
        y=df["word"].values[::-1],
        x=df["wordcount"].values[::-1],
        showlegend=False,
        orientation = 'h',
        marker=dict(
            color=color,
        ),
    )
    return trace

# FUNCTION TO MUNGE DATA
def return_sorted_df_by_word_count_and_word(df, NUM_SENT=1):
    freq_dict = defaultdict(int)
    for sent in df["review_clean"]:
        for word in generate_ngrams(sent, NUM_SENT):
            freq_dict[word] += 1
    fd_sorted = pd.DataFrame(sorted(freq_dict.items(), key=lambda x: x[1])[::-1])
    fd_sorted.dropna(inplace=True)
    fd_sorted.columns = ["word", "wordcount"]
    fd_sorted['word'] = fd_sorted['word'].str.replace('[^a-zA-Z ]+', '', regex=True).str.strip()
    fd_sorted['str_len'] = fd_sorted['word'].str.split().apply(len)
    fd_sorted = fd_sorted[
        (fd_sorted['str_len'] == NUM_SENT)
    ]
    return fd_sorted
```

```
100%|██████████| 1780154/1780154 [12:32<00:00, 2366.79it/s]
100%|██████████| 1780154/1780154 [00:01<00:00, 1008742.48it/s]
100%|██████████| 1780154/1780154 [00:10<00:00, 176230.22it/s]
```

Single Word Plots

In [11]:
```
SINGLE_NUM_SENT = 1

# For Positive Reivews Get the bar chart from negative review_clean ##
fd_sorted = return_sorted_df_by_word_count_and_word(review_pos, SINGLE_NUM_SENT)
trace0 = horizontal_bar_chart(fd_sorted.head(25), 'green')

# Repeat for Neutral Reviews
fd_sorted = return_sorted_df_by_word_count_and_word(review_neu, SINGLE_NUM_SENT)
trace1 = horizontal_bar_chart(fd_sorted.head(25), 'grey')

# Repeat for Negative Reviews
fd_sorted = return_sorted_df_by_word_count_and_word(review_neg, SINGLE_NUM_SENT)
trace2 = horizontal_bar_chart(fd_sorted.head(25), 'brown')

# Creating Subplots
fig = tools.make_subplots(
    rows=3, cols=1, vertical_spacing=0.04,
    subplot_titles=[
        "Frequent words of positive review_clean",
        "Frequent words of neutral review_clean",
        "Frequent words of negative review_clean"
    ]
)

fig.append_trace(trace0, 1, 1)
fig.append_trace(trace1, 2, 1)
fig.append_trace(trace2, 3, 1)

fig['layout'].update(height=1200, width=900, paper_bgcolor='rgb(233,233,233)', title="Word Count Plots")

iplot(fig, filename='word-plots')
```

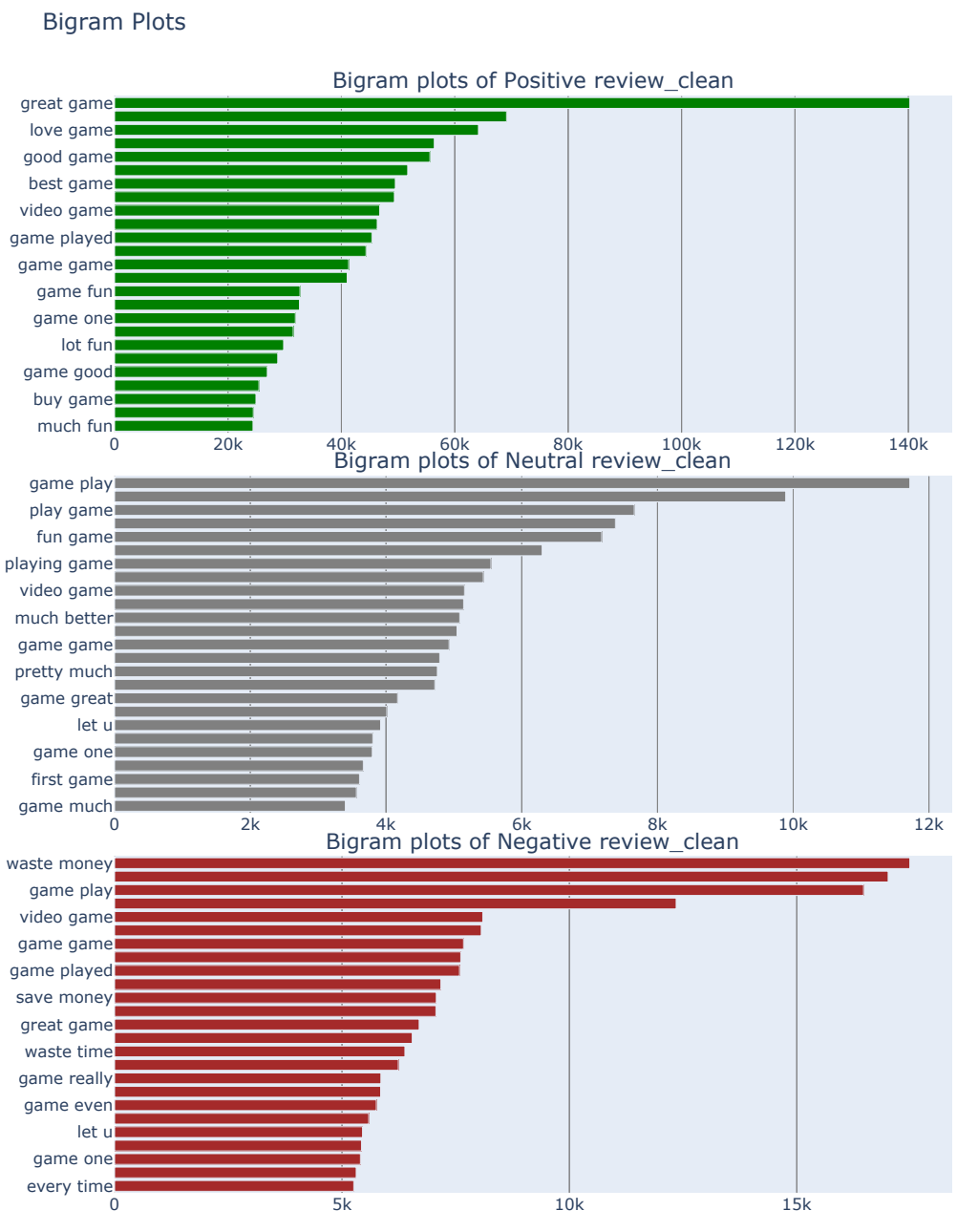
```
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/plotly/tools.py:461: DeprecationWarning:

plotly.tools.make_subplots is deprecated, please use plotly.subplots.make_subplots instead
```



Bigram Plot

In [12]:
```
BIGRAM_NUM_SENT = 2

fd_sorted = return_sorted_df_by_word_count_and_word(review_pos, BIGRAM_NUM_SENT)
trace0 = horizontal_bar_chart(fd_sorted.head(25), 'green')

fd_sorted = return_sorted_df_by_word_count_and_word(review_neu, BIGRAM_NUM_SENT)
trace1 = horizontal_bar_chart(fd_sorted.head(25), 'grey')

fd_sorted = return_sorted_df_by_word_count_and_word(review_neg, BIGRAM_NUM_SENT)
trace2 = horizontal_bar_chart(fd_sorted.head(25), 'brown')

fig = tools.make_subplots(
    rows=3, cols=1,
    vertical_spacing=0.04, horizontal_spacing=0.25,
    subplot_titles=[
        "Bigram plots of Positive review_clean",
        "Bigram plots of Neutral review_clean",
        "Bigram plots of Negative review_clean"
    ]
)

fig.append_trace(trace0, 1, 1)
fig.append_trace(trace1, 2, 1)
fig.append_trace(trace2, 3, 1)


fig['layout'].update(height=1000, width=800, paper_bgcolor='rgb(233,233,233)', title="Bigram Plots")
iplot(fig, filename='word-plots')
```

```
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/plotly/tools.py:461: DeprecationWarning:

plotly.tools.make_subplots is deprecated, please use plotly.subplots.make_subplots instead
```



Trigram Plots

In [13]:
```
TRIGRAM_NUM_SENT = 3

fd_sorted = return_sorted_df_by_word_count_and_word(review_pos, TRIGRAM_NUM_SENT)
trace0 = horizontal_bar_chart(fd_sorted.head(25), 'green')

fd_sorted = return_sorted_df_by_word_count_and_word(review_neu, TRIGRAM_NUM_SENT)
trace1 = horizontal_bar_chart(fd_sorted.head(25), 'grey')

fd_sorted = return_sorted_df_by_word_count_and_word(review_neg, TRIGRAM_NUM_SENT)
trace2 = horizontal_bar_chart(fd_sorted.head(25), 'brown')


fig = tools.make_subplots(
    rows=3, cols=1,
    vertical_spacing=0.04, horizontal_spacing=0.05,
    subplot_titles=[
        "Tri-gram plots of Positive review_clean",
        "Tri-gram plots of Neutral review_clean",
        "Tri-gram plots of Negative review_clean"
    ]
)

fig.append_trace(trace0, 1, 1)
fig.append_trace(trace1, 2, 1)
fig.append_trace(trace2, 3, 1)

fig['layout'].update(height=1200, width=1200, paper_bgcolor='rgb(233,233,233)', title="Trigram Count Plots")
iplot(fig, filename='word-plots')
```

```
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/plotly/tools.py:461: DeprecationWarning:

plotly.tools.make_subplots is deprecated, please use plotly.subplots.make_subplots instead
```