Data Pre-processing

In [123…

```
from __future__ import print_function
import nltk, re, pickle, os
import pandas as pd
import numpy as np
from textblob import TextBlob
from nltk.tokenize import sent_tokenize, word_tokenize, wordpunct_tokenize, MWETokenize
from nltk.stem import porter, WordNetLemmatizer
from nltk.corpus import stopwords
from nltk.util import ngrams
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.decomposition import LatentDirichletAllocation,  TruncatedSVD
from collections import Counter
from operator import itemgetter
import heapq
import urllib
import urllib.parse
import spacy
nlp = spacy.load('en_core_web_sm')
from urllib.request import urlopen
from lxml import etree
from googleapiclient.discovery import build
from youtube_transcript_api import YouTubeTranscriptApi
import requests
import json
import lxml
import cufflinks as cf
cf.go_offline()
cf.set_config_file(offline=False, world_readable=True)
from plotly.offline import init_notebook_mode, iplot
init_notebook_mode(connected=True)
import plotly.graph_objs as go
import plotly.offline as offline
offline.init_notebook_mode()
from plotly import tools
import plotly.tools as tls
init_notebook_mode(connected=True)
import ast
import plotly.express as px
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
from wordcloud import WordCloud
```

In [6]:

```
ted_main = pd.read_csv('ted_main.csv')
ted_trans = pd.read_csv('transcripts.csv')
ted_all = pd.merge(ted_trans,right=ted_main,on='url')
ted_all.head(5)
```

Out[6]:

	transcript	url	comments	description	duration	eve

	transcript	url	comments	description	duration	eve
0	Good morning. How are you? (Laughter)It's been ...	https://www.ted.com/talks/ken_robinson_says_sc...	4553	Sir Ken Robinson makes an entertaining and pro...	1164	TED20
1	Thank you so much, Chris. And it's truly a gre...	https://www.ted.com/talks/al_gore_on_averting_...	265	With the same humor and humanity he exuded in ...	977	TED20
2	(Music: "The Sound of Silence," Simon & Garfun...	https://www.ted.com/talks/david_pogue_says_sim...	124	New York Times columnist David Pogue takes aim...	1286	TED20
3	If you're here today — and I'm very happy that...	https://www.ted.com/talks/majora_carter_s_tale...	200	In an emotionally charged talk, MacArthur-winn...	1116	TED20
4	About 10 years ago, I took on the task to teac...	https://www.ted.com/talks/hans_rosling_shows_t...	593	You've never seen data presented like this. Wi...	1190	TED20

In [7]:
```
with open('ted_all.pkl', 'wb') as picklefile:
    pickle.dump(ted_all, picklefile)
```

In [8]:
```
ted_all['id'] = ted_all.index
```

Extracting Video Ids from Youtube of Ted Talks

In [34]:
```
# function to get video id and url from youtube from a given title of a ted talk
def getData(base_url, page_url, query):
    page = urlopen(page_url+urllib.parse.quote(query))
#     print(page.url)
    video_ids = re.findall(r"watch\?v=(\S{11})", page.read().decode())
    if video_ids:
        link = "/watch?v="+video_ids[0]
        video_link = urlopen(base_url+link)
        params = {"format": "json", "url": base_url+link}
        url = "https://www.youtube.com/oembed"
        query_string = urllib.parse.urlencode(params)
```

```
        url = url + "?" + query_string
        with urlopen(url) as response:
            response_text = response.read()
            data = json.loads(response_text.decode())
            query1 = query.lower()
            query2 = data['title'].lower()
            if query1 not in query2:
                return None
    else:
        return None, None
    return video_ids[0]
```

In [37]:

```
base_url = "https://www.youtube.com"
page_url = 'https://www.youtube.com/c/TED/search?query='
data = pd.DataFrame(columns=['title', 'video_id'])
i = 0
for t in ted_all['title']:
    if i%200==0:
        print(i)
    vid = getData(base_url, page_url, t)
    i += 1
    if vid != None:
        data = data.append({'title': t, 'video_id': vid}, ignore_index=True)
```

```
0
200
400
600
800
1000
1200
1400
1600
1800
2000
2200
2400
```

In [38]:

```
df_all = pd.merge(data,right=ted_all,on='title')
with open('df_all.pkl', 'wb') as picklefile:
    pickle.dump(df_all, picklefile)
df_all
```

Out[38]:

	title	video_id	transcript	url	comments
0	Do schools kill creativity?	iG9CE55wbtY	Good morning. How are you? (Laughter)It's been ...	https://www.ted.com/talks/ken_robinson_says_sc...	4553
1	Averting the climate crisis	rDiGYuQicpA	Thank you so much, Chris. And it's truly a gre...	https://www.ted.com/talks/al_gore_on_averting_...	265

	title	video_id	transcript	url	comments
2	Simplicity sells	NEjZt0y6OOw	(Music: "The Sound of Silence," Simon & Garfun...	https://www.ted.com/talks/david_pogue_says_sim...	124
3	Greening the ghetto	gQ-cZRmHfs4	If you're here today — and I'm very happy that...	https://www.ted.com/talks/majora_carter_s_tale...	200
4	The best stats you've ever seen	hVimVzgtD6w	About 10 years ago, I took on the task to teac...	https://www.ted.com/talks/hans_rosling_shows_t...	593
...	...	...	...	...	...
1977	What we're missing in the debate about immigra...	2wu28tI9VkM	So, Ma was trying to explain something to me a...	https://www.ted.com/talks/duarte_geraldino_wha...	17
1978	The most Martian place on Earth	lWnr-99DNeE	This is a picture of a sunset on Mars taken by...	https://www.ted.com/talks/armando_azua_bustos_...	6
1979	What intelligent machines can learn from a sch...	0bRocfcPhHU	In my early days as a graduate student, I went...	https://www.ted.com/talks/radhika_nagpal_what_...	10
1980	A black man goes undercover in the alt-right	OqUaEJLfrLo	I took a cell phone and accidentally made myse...	https://www.ted.com/talks/theo_e_j_wilson_a_bl...	32
1981	How a video game might help us build better ci...	qYUmI5kGsYk	We humans are becoming an urban species, so ci...	https://www.ted.com/talks/karoliina_korppoo_ho...	8

1982 rows × 19 columns

Data Cleaning

Non-speech sounds, events

- While looking at initial n-grams, I notice that we get a lot of "thank you applause". So, I started looking at all the non-word behavior that is transcribed (see below). Luckily, they put all of the speaker's parenthetical comments, and some titles of songs(?) played into brackets [ ] and all of the audience sounds, viedos, music, etc in parenthases.
- So, it is safe to first go and take out everything that is in parentheses before we even tokenize so that we can just look at speech.
- It would be interesting to collect these and keep a count in the main matrix, especially for things like 'laughter' or applause or multimedia (present/not present) in making recommendations or calculating the popularity of a talk.

Examples of parentheticals (non-speech sounds)

(Applause)(Applause ends)(Pre-recorded applause)(Pre-recorded applause and cheering)(Audience cheers)(Laughter)(Shouting)(Mock sob)(Breathes in)(Baby cooing)(Video)(Singing)(Heroic music)(Loud music)(Music)(Music ends)(Plays notes)(Sighs)(Clears throat)(Whispering)

Four important steps for cleaning the text and getting it into a format that we can analyze:

- tokenize
- lemmatize/stemmatize
- remove stop words/punctuation
- vectorize

In [39]:
```
def clean_text(text):
    lemmizer = WordNetLemmatizer()
    stop = stopwords.words('english')
    stop += ['.', ',',':','...','!"','?"', "'", '"',' - ',' — ',"'","'.",'!', ';','♫♫'
             '.\'"','[',']','—',".\'", 'ok','okay','yeah','ya','stuff', ' 000 ',' em ',
             ' oh ','thank','thanks','la','was','wa','?','like','go',' le ',' ca ',' I
    cleaned_text = []
    for post in text:
        cleaned_words = []
        clean_parens = re.sub(r'\([^)]*\)', ' ', post)
        for word  in wordpunct_tokenize(clean_parens):
            if word.lower() not in stop:
                low_word = lemmizer.lemmatize(word)
                if low_word.lower() not in stop:
                    cleaned_words.append(low_word.lower())
        cleaned_text.append(' '.join(cleaned_words))
    return cleaned_text
```

In [40]:
```
df_all['transcript'] = clean_text(df_all['transcript'])
```

In [41]:
```
with open('cleaned_talks.pkl', 'wb') as picklefile:
    pickle.dump(df_all, picklefile)
```

In [47]:
```
df_all['transcript'][0][:300]
```

Out[47]:
```
'good morning great blown away whole thing fact leaving three theme running conference r
elevant want talk one extraordinary evidence human creativity presentation people variet
y range second put u place idea going happen term future idea may play interest educatio
n actually find everybody interest ed'
```

Vectorization

Vectorization is the important step of turning our words into numbers. This function takes each word in each document and counts the number of times the word appears. You end up with each word as your columns and each row is a document (talk), so the data is the frequency of each word in each document, we call this a sparse matrix.

In [48]:
```
c_vectorizer = CountVectorizer(ngram_range=(1,3),
                               stop_words='english',
                               max_df = 0.6,
                               max_features=10000)
t_vectorizer = TfidfVectorizer(ngram_range=(1, 3),
                                     stop_words='english',
                                     token_pattern="\\b[a-z][a-z]+\\b",
                                     lowercase=True,
                                     max_df = 0.6)
c_vectorizer.fit(df_all['transcript'])
c_x = c_vectorizer.transform(df_all['transcript'])
t_vectorizer.fit(df_all['transcript'])
t_x = t_vectorizer.transform(df_all['transcript'])
```

Vectorization will come in use with Topic Modelling.

In [49]:
```
with open('cleaned_talks.pkl', 'rb') as picklefile:
    df = pickle.load(picklefile)
```

In [50]:
```
len(df)
```

Out[50]:
```
1982
```

Exploratory Data Analysis

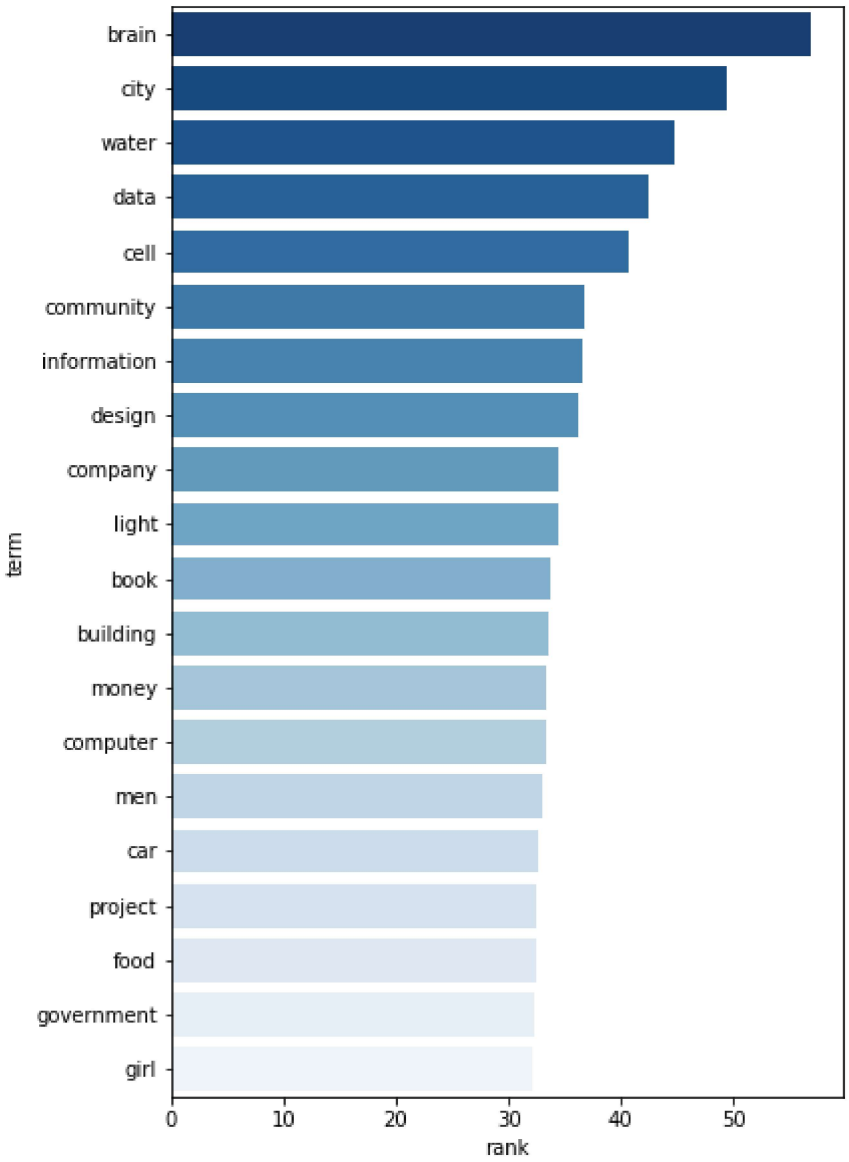
In [53]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(stop_words="english",
                         use_idf=True,
                         ngram_range=(1,1),
                         min_df = 0.05,
                         max_df = 0.3)
tfidf = vectorizer.fit_transform(df['transcript'])
```

In [54]:
```
import seaborn as sns
import matplotlib.pyplot as plt
def rank_words(terms, feature_matrix):
    sums = feature_matrix.sum(axis=0)
    data = []
    for col, term in enumerate(terms):
        data.append((term, sums[0,col]))
    ranked = pd.DataFrame(data, columns=['term','rank']).sort_values('rank', ascending=
    return ranked

ranked = rank_words(terms=vectorizer.get_feature_names(), feature_matrix=tfidf)

fig, ax = plt.subplots(figsize=(6,10), ncols=1, nrows=1)
sns.barplot(x='rank',y='term',data=ranked[:20], palette='Blues_r', ax=ax)
```

Out[54]:
```
<AxesSubplot:xlabel='rank', ylabel='term'>
```



- These are the most important words in the transcripts.

In [55]:
```
# Let's visualize a word cloud with the frequencies obtained by idf transformation
dic = {ranked.loc[i,'term'].upper(): ranked.loc[i,'rank'] for i in range(0,len(ranked))
```

```
wordcloud = WordCloud(background_color='white',
                      max_words=100,
                      colormap='Reds').generate_from_frequencies(dic)
fig = plt.figure(1,figsize=(12,15))
plt.imshow(wordcloud,interpolation="bilinear")
plt.axis('off')
plt.show()
```

In [56]:

```
def sentimental(sent):
    sid_obj = SentimentIntensityAnalyzer()
    sentiment_dict = sid_obj.polarity_scores(sent)
    if sentiment_dict['compound'] >= 0.05:
        return "pos"
    elif sentiment_dict['compound'] <= -0.05:
        return "neg"
    else:
        return "neu"
```

In [58]:

```
for i,sent in enumerate(df['transcript']):
    if sentimental(sent) == 'pos':
        df.at[i, 'sentiment'] = 'positive'
    elif sentimental(sent) == 'neg':
        df.at[i, 'sentiment'] = 'negative'
    else:`
        df.at[i, 'sentiment'] = 'neutral'
```

In [102…

```
df.head()
```

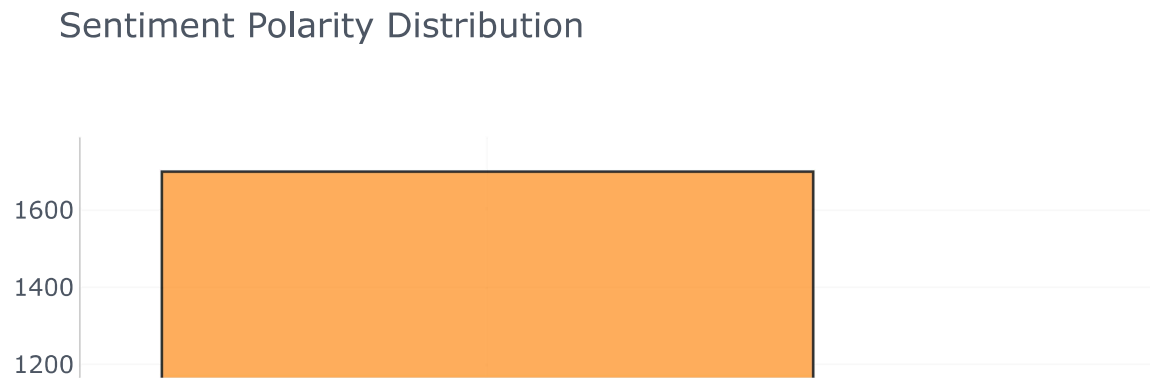
Out[102…

	title	video_id	transcript	url	comments	de

	title	video_id	transcript	url	comments	de
0	Do schools kill creativity?	iG9CE55wbtY	good morning great blown away whole thing fact...	https://www.ted.com/talks/ken_robinson_says_sc...	4553	er
1	Averting the climate crisis	rDiGYuQicpA	much chris truly great honor opportunity come ...	https://www.ted.com/talks/al_gore_on_averting_...	265	h I
2	Simplicity sells	NEjZt0y6OOw	hello voice mail old friend called tech suppor...	https://www.ted.com/talks/david_pogue_says_sim...	124	Pc
3	Greening the ghetto	gQ-cZRmHfs4	today happy heard sustainable development save...	https://www.ted.com/talks/majora_carter_s_tale...	200	er M
4	The best stats you've ever seen	hVimVzgtD6w	10 year ago took task teach global development...	https://www.ted.com/talks/hans_rosling_shows_t...	593	r

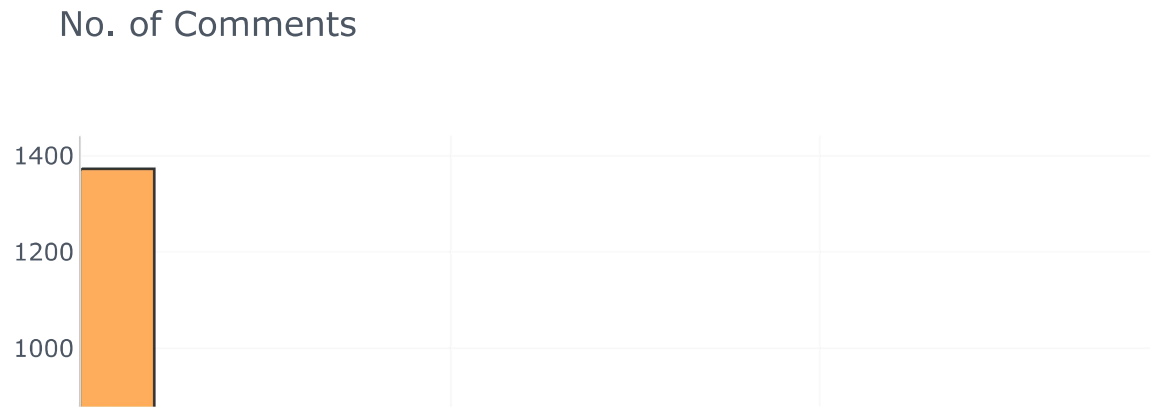
In [66]:

```
df['sentiment'].iplot(
    kind='hist',
    bins=50,
    xTitle='polarity',
    linecolor='black',
    yTitle='count',
    title='Sentiment Polarity Distribution')
```

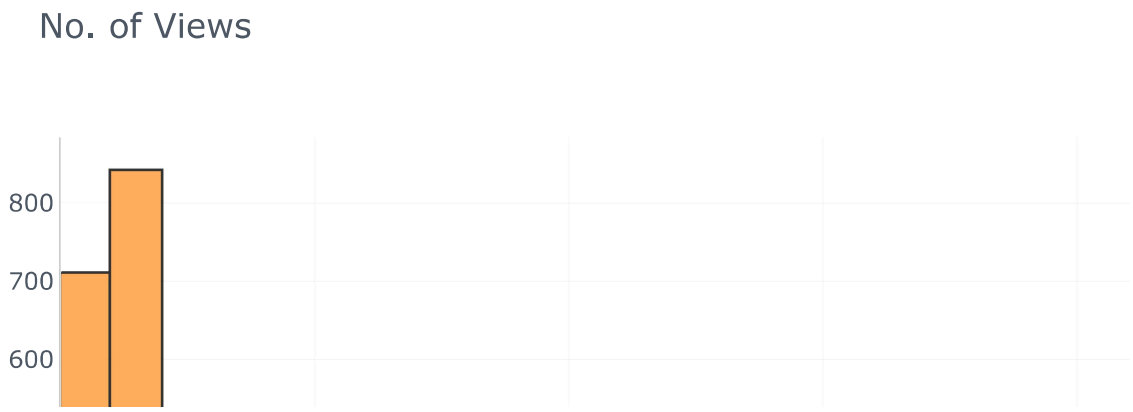


In [69]:

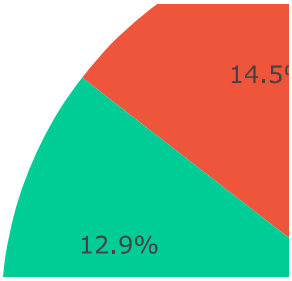
```
df['comments'].iplot(
    kind='hist',
    bins=50,
    xTitle='comments',
    linecolor='black',
    yTitle='count',
    title='No. of Comments')
```



In [70]:
```
df['views'].iplot(
    kind='hist',
    bins=50,
    xTitle='views',
    linecolor='black',
    yTitle='count',
    title='No. of Views')
```



In [129…
```
dic = {'Expression':[], 'Count': []}
for i in df['ratings']:
    i = ast.literal_eval(i)
    for j in i:
        dic['Expression'].append(j['name'])
        dic['Count'].append(j['count'])
fig = px.pie(dic, names='Expression', values='Count')
fig.show()
```



- This pie chart shows different expressions found in the ted talk with respect to their percentage and count.

In [120…

```
dic = {}
for i in df['tags']:
    i = ast.literal_eval(i)
    for j in i:
        if j in dic:
            dic[j] += 1
        else:
            dic[j] = 1
keys = [*dic.keys()]
values = [*dic.values()]
dic = {'Tags':keys, 'Count':values}
fig = px.bar(dic, x='Tags', y='Count')
fig.show()
```

- Plot of Different tags that the ted talks are labelled with their count.

In [132…

```
def senti(sent):
    sid_obj = SentimentIntensityAnalyzer()
    sentiment_dict = sid_obj.polarity_scores(sent)
    if sentiment_dict['compound'] >= 0.05:
        return 1
    elif sentiment_dict['compound'] <= -0.05:
        return -1
    else:
        return 0
```

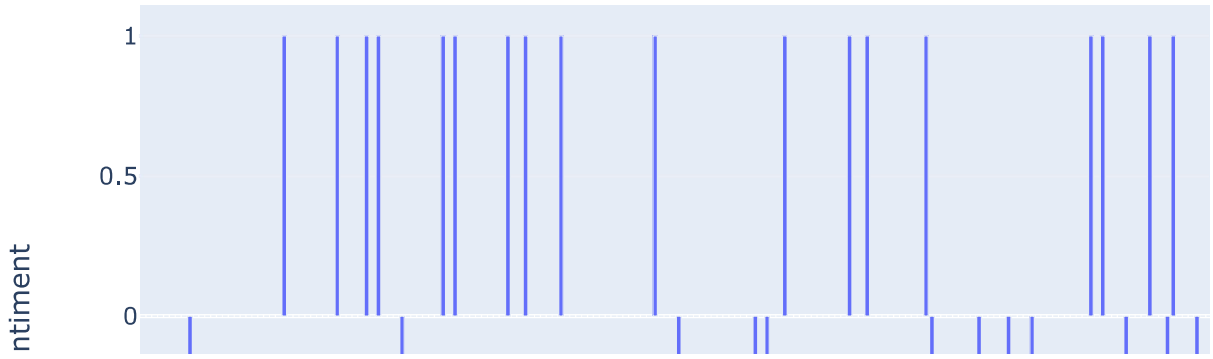
In [143…

```
dic = {'Expression':[], 'Sentiment':[]}
for i in df['ratings']:
    i = ast.literal_eval(i)
    for j in i:
        dic['Expression'].append(j['name'])
dic['Expression'] = np.unique(dic['Expression'])
for i in dic['Expression']:
    dic['Sentiment'].append(senti(i))
fig = px.bar(dic, x='Expression', y='Sentiment')
fig.show()
```

- Plot of Expression vs its Sentiment. Here Positive means 1, Neutral means 0 and Negative means -1.

In [164…

```
dic = {'Tags':[], 'Sentiment':[]}
for i in df['tags']:
    i = ast.literal_eval(i)
    for j in i:
        dic['Tags'].append(j)
dic['Tags'] = np.unique(dic['Tags'])
for i in dic['Tags']:
    dic['Sentiment'].append(senti(i))
fig = px.bar(dic, x='Tags', y='Sentiment')
fig.show()
```



- Plot of Tags vs Sentiment.