Sentiment analysis and Topic modeling on Tweets

Team members

1. Prabhath Reddy
2. Vinit Kanani
3. Reshma Chowdary Bobba
4. Sathvick Reddy Narahari

Dataset

We are using Sentiment140 as a dataset. This dataset consists of tweets which are collected using twitter API, they labeled a tweet as positive if it has :) emoticon in it and negative if it has :(emoticon in it. The length of the training dataset is 16,00,000. The dataset consists of 6 columns, 0 - polarity of the tweet, 1 - id of the tweet, 2 - date of the tweet, 3 - the query, 4 - the user that tweeted, 5 - the text of the tweet. The polarity of the tweet column has three values they are 0 = neutral, 2 = negative and 4 = positive . The data is available in the form of google drive link at http://help.sentiment140.com/for-students/ .

Description of the problem

We would like to perform sentiment analysis on the dataset . The data has a column called polarity which tells us if the tweet is neutral, negative or positive. We would like to see if the models we create to perform sentiment analysis are aligning with their assumption that tweets which have :) emoticon contain positive sentiment and tweets which have :(emoticon contain negative sentiment.We would like to perform topic modeling on all the tweets, which will help us in detecting the topics present in the tweets. This will help us to organize and also summarize such a large tweets dataset.

Potential Methods

We are planning to use supervised techniques for sentiment analysis of the tweets and unsupervised techniques for topic modeling. We are planning to use supervised techniques, Naive Bayes classifier and Decision Trees which will be trained on the labeled data to predict the sentiment of the test data. We are using Latent Dirichlet Allocation (LDA) an unsupervised technique to detect topics present in the tweets.

In [1]:
```
from wordcloud import WordCloud
from textblob import TextBlob
from sklearn.metrics import confusion_matrix
from nltk.sentiment.vader import SentimentIntensityAnalyzer
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer

import nltk
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import plotly.express as px
import re
```

In [2]:
```
nltk.download('vader_lexicon')
nltk.download('punkt')
nltk.download('wordnet')
nltk.download('omw-1.4')
nltk.download('stopwords')
pd.set_option('display.max_colwidth', None)
```

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     /Users/vinitkanani/nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]     /Users/vinitkanani/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     /Users/vinitkanani/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to
[nltk_data]     /Users/vinitkanani/nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/vinitkanani/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

In [3]:
```
# import data
data = pd.read_csv('data/training.csv', encoding="ISO-8859-1", header=None)
data.columns = ['sentiment', 'id', 'date', 'query', 'user', 'tweet']
```

In [4]:
```
data.head()

print("Size of the dataset", data.shape)
```

```
Size of the dataset (1600000, 6)
```

In [5]:
```
# Missing Values
print("Missing Values \n\n", data.isnull().sum())
```

```
Missing Values

 sentiment    0
id           0
date         0
query        0
user         0
tweet        0
dtype: int64
```

1. Data Preprocessing

In [6]:

```
print("Number of http links", data['tweet'].str.count('http').sum())
data['tweet'] = data['tweet'].str.replace(r'http\S+|www.\S+', '', case=False, regex=True

print("Number of @ mentions", data['tweet'].str.count('@').sum())
data['tweet'] = data['tweet'].str.replace(r'@\S+', '', case=False, regex=True)

print("Number of # mentions", data['tweet'].str.count('#').sum())
data['tweet'] = data['tweet'].str.replace(r'#\S+', '', case=False, regex=True)

print("Number of RT", data['tweet'].str.count('RT').sum())
data['tweet'] = data['tweet'].str.replace(r'RT', '', case=False, regex=True)
```

```
Number of http links 71635
Number of @ mentions 798628
Number of # mentions 45133
Number of RT 0
```

In [7]:

```
stop_words = set(stopwords.words('english'))
stop_words.add('quot')
stop_words.add('amp')

lemma = WordNetLemmatizer()

def clean_text(text):
    text = str(text).lower()
    text = re.sub(r'http\S+', ' ', text)
    text = re.sub('[^a-zA-Z]', ' ', text)
    text = word_tokenize(text)
    text = [item for item in text if item not in stop_words]
    text = [lemma.lemmatize(w) for w in text]
    text = [i for i in text if len(i) > 2]
    text = ' '.join(text)
    return text
```

In [9]:

```
data['clean_tweet'] = data['tweet'].apply(clean_text)
print(data.head(2))
```

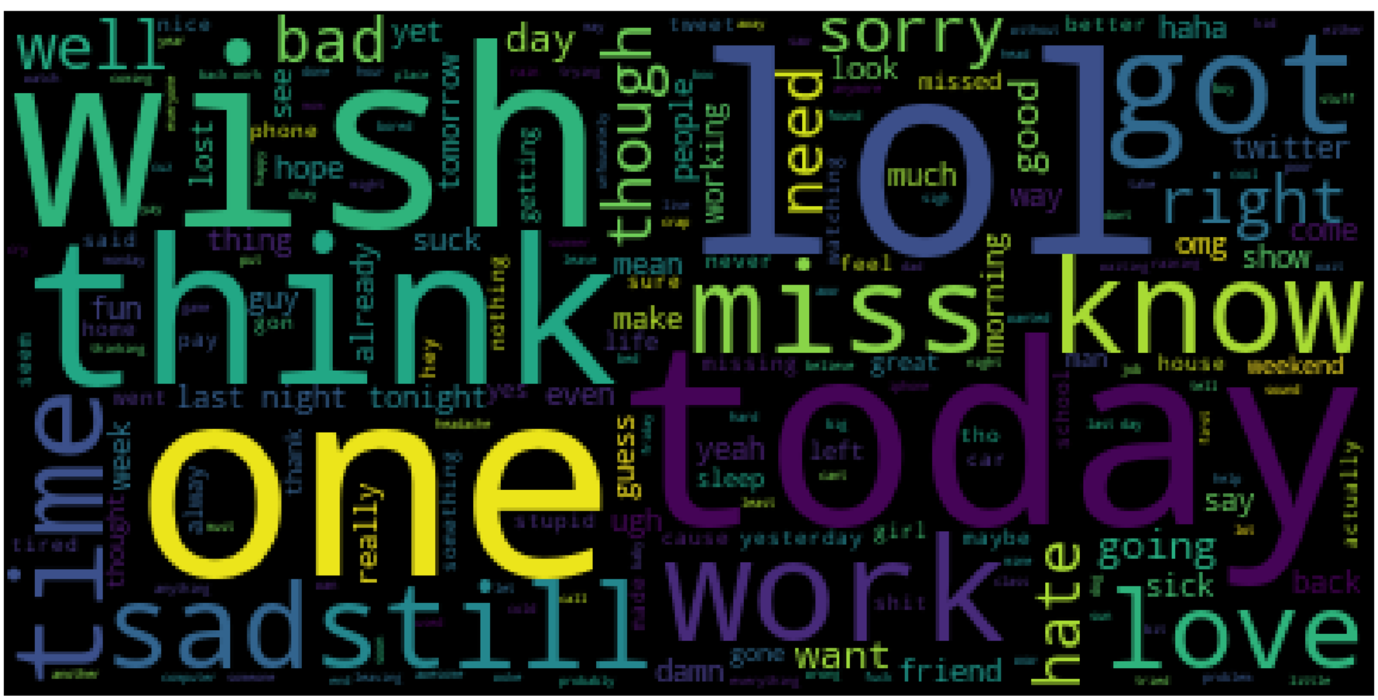
```
   sentiment          id                          date     query  \
0          0  1467810369  Mon Apr 06 22:19:45 PDT 2009  NO_QUERY
1          0  1467810672  Mon Apr 06 22:19:49 PDT 2009  NO_QUERY

             user  \
0  _TheSpecialOne_
1    scotthamilton


                   tweet  \
0                                  - Awww, that's a bummer.  You shoulda got David Carr
of Third Day to do it. ;D
1  is upset that he can't update his Facebook by texting it... and might cry as a result
  School today also. Blah!

                                                            clean_tweet
0                                awww bummer shoulda got david carr third day
1  upset update facebook texting might cry result school today also blah
```

2. Wordclouds

In [10]:
```python
def get_word_cloud(text):
    plt.figure(figsize=(20,20))
    word_cloud = WordCloud().generate(text)
    plt.imshow(word_cloud)
    plt.axis("off")
    plt.show()
```

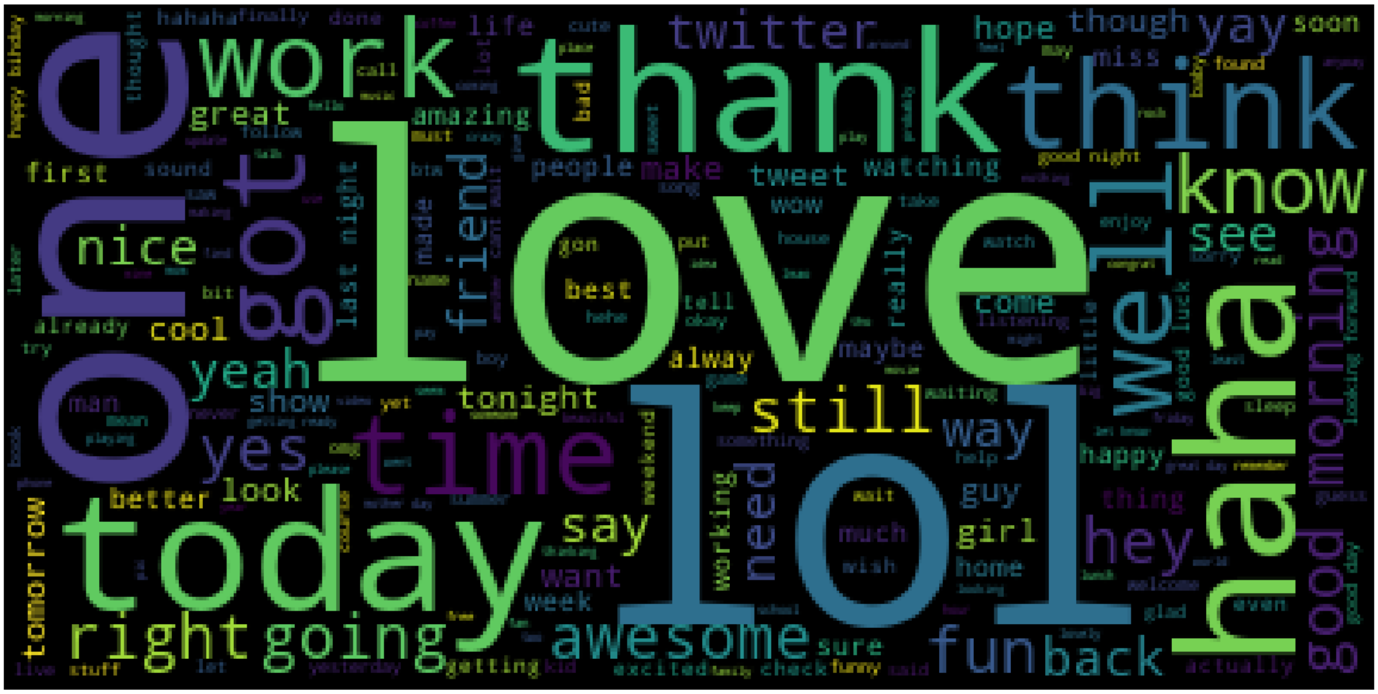
2.1 WordCloud of tweets with negative sentiment

In [11]:
```python
get_word_cloud(data[data['sentiment'] == 0]['clean_tweet'].str.cat(sep=' '))
```



2.2 WordCloud of tweets with positive sentiment

In [12]:
```python
get_word_cloud(data[data['sentiment'] == 4]['clean_tweet'].str.cat(sep=' '))
```

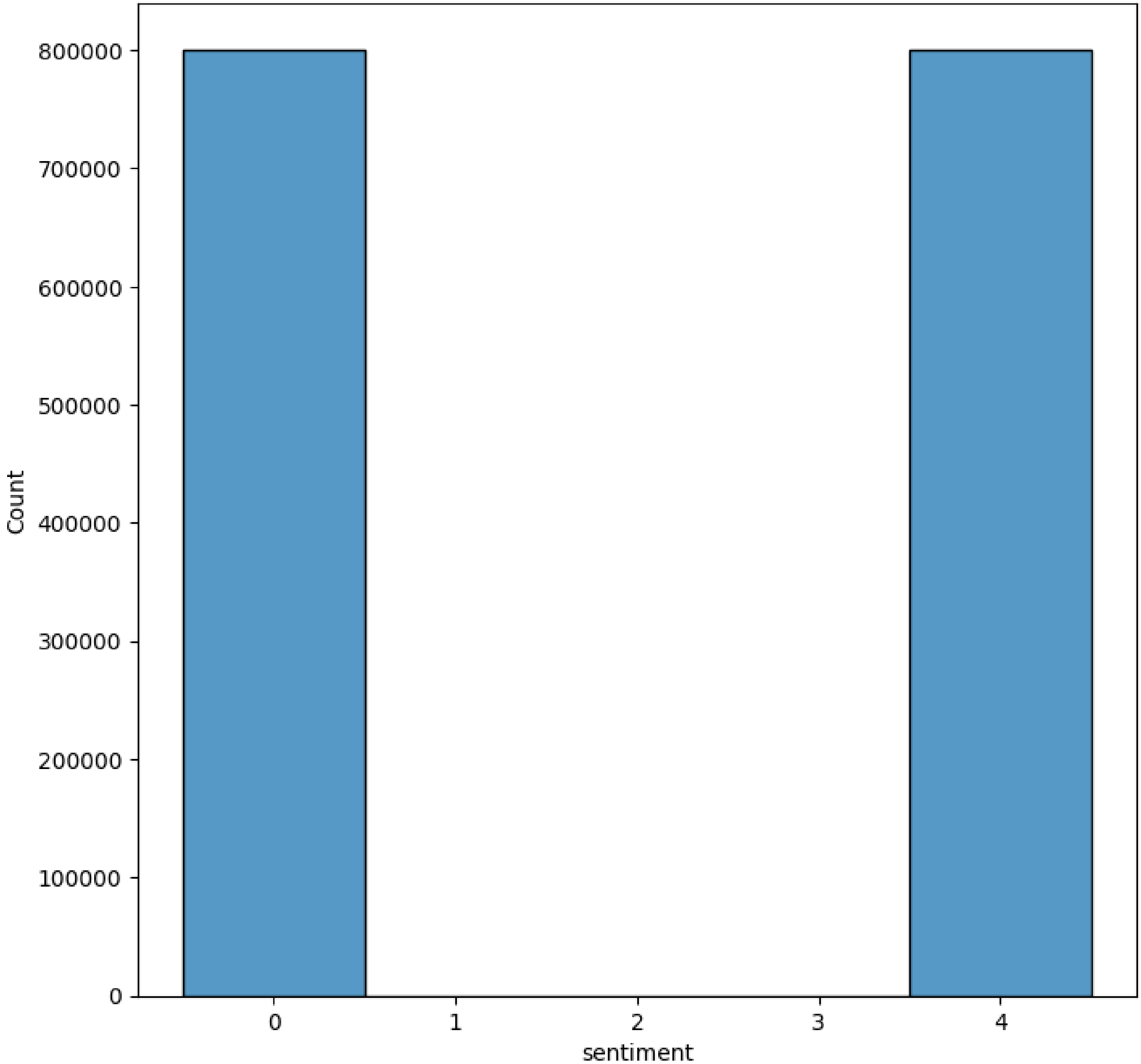


3. Count of tweets for each sentiment

In [13]:

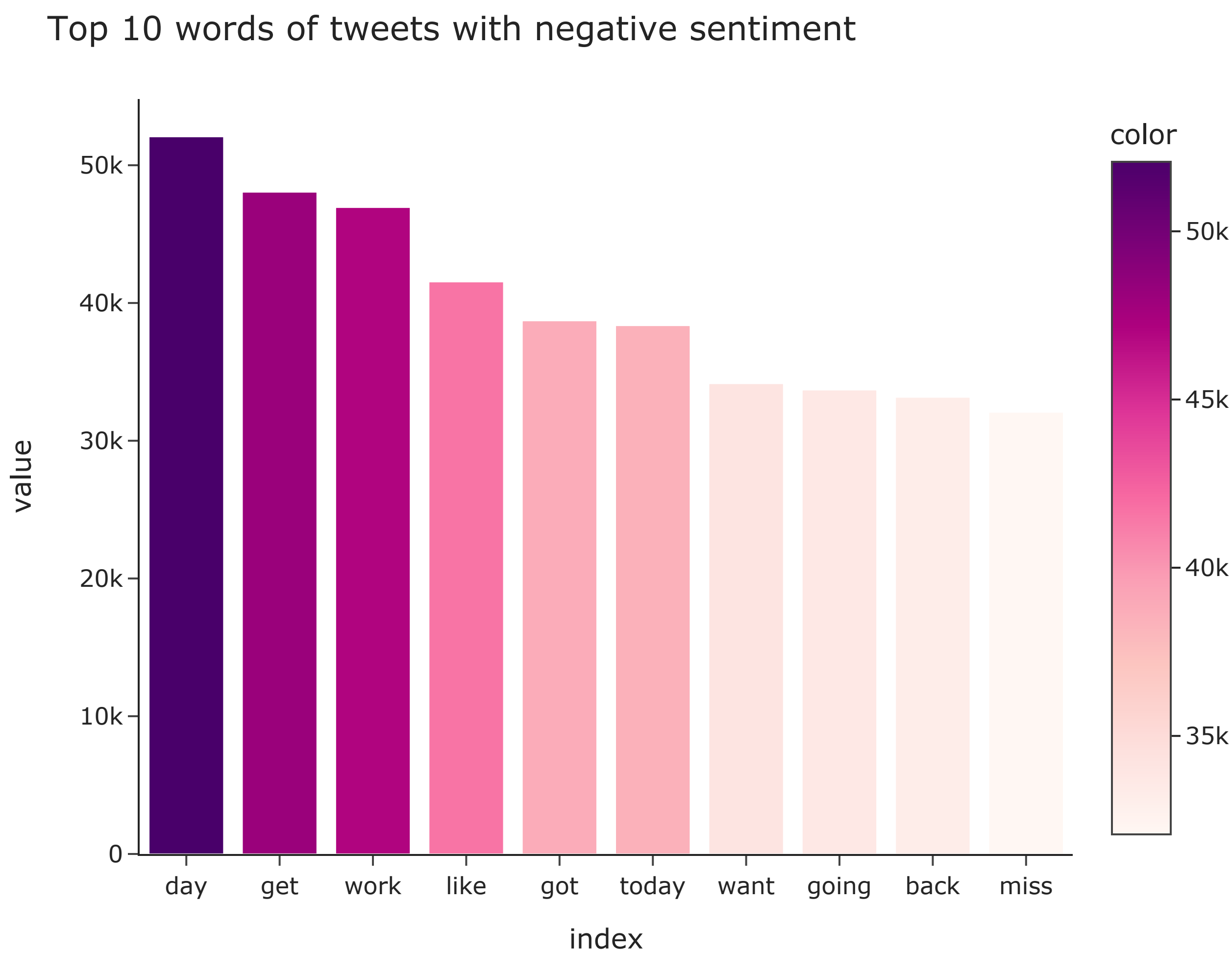
```python
plt.figure(figsize=(8,8))
sns.histplot(data=data['tweet'], x=data['sentiment'], discrete="True")
```

Out[13]: <AxesSubplot: xlabel='sentiment', ylabel='Count'>



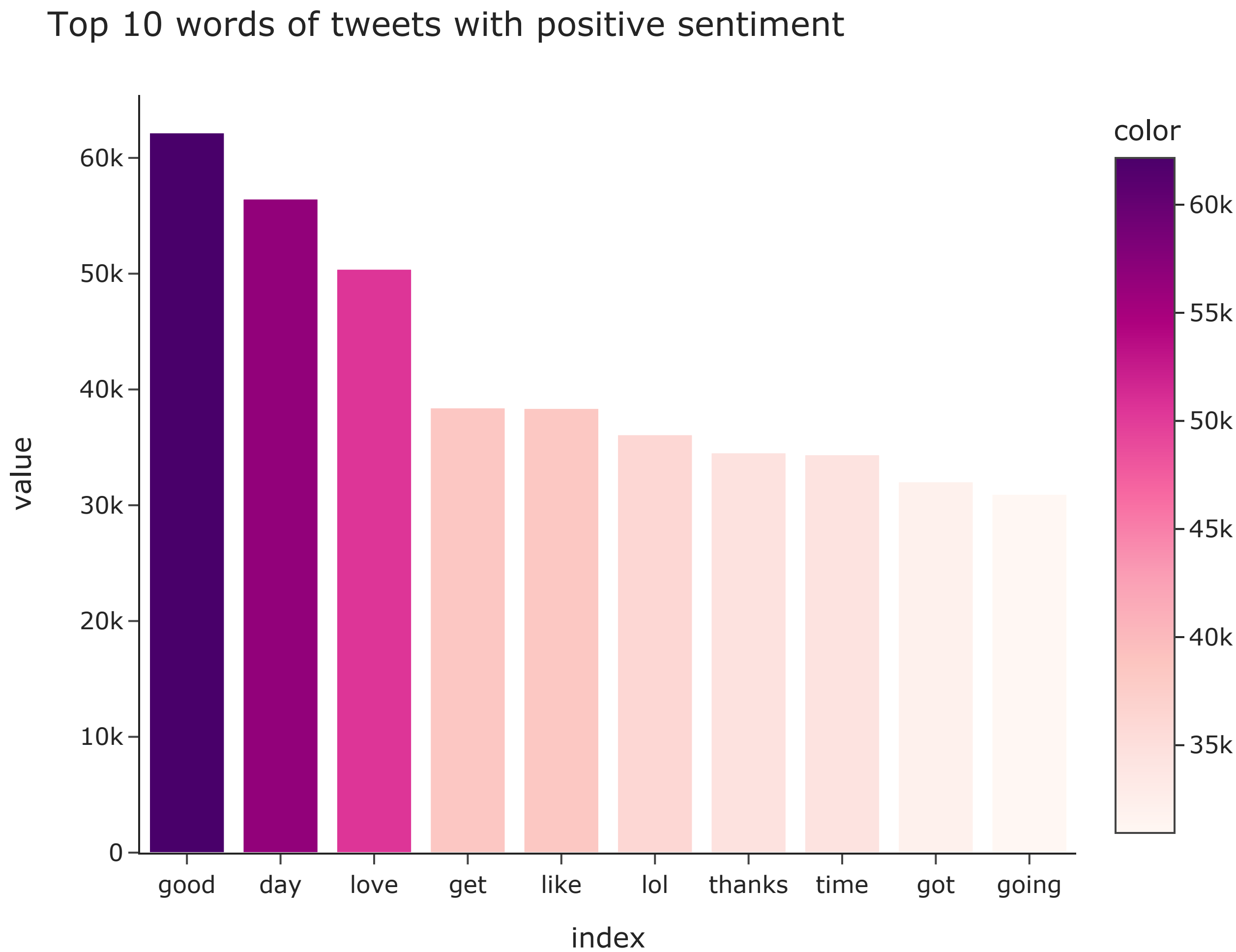
4. Top 10 frequent words of tweets with negative sentiment

In [14]:
```
top10_word = data.clean_tweet[data.sentiment == 0].str.split(expand=True).stack().value_

fig = px.bar(top10_word, color=top10_word.values, color_continuous_scale=px.colors.seque
    top10_word.values])
fig.update_traces(hovertemplate='Count: %{customdata[0]}')
fig.update_layout(title=f"Top 10 words of tweets with negative sentiment",
                  template='simple_white', hovermode='x unified')
fig.show()
```



5. Top 10 frequent words of tweets with positive sentiment

In [15]:
```
top10_word = data.clean_tweet[data.sentiment == 4].str.split(expand=True).stack().value_


fig = px.bar(top10_word, color=top10_word.values, color_continuous_scale=px.colors.seque
    top10_word.values])
fig.update_traces(hovertemplate='Count: %{customdata[0]}')
fig.update_layout(title=f"Top 10 words of tweets with positive sentiment",
                  template='simple_white', hovermode='x unified')
fig.show()
```


Top 10 words of tweets with positive sentiment
value
60k
50k
40k
30k
20k
10k
0
good
day
love
get
like
lol
thanks
time
got
going
index
color
60k
55k
50k
45k
40k
35k