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

from sklearn.preprocessing import LabelEncoder
```

In [2]:
```
# Read Data
df = pd.read_csv('data/training.1600000.processed.noemoticon.csv', header=None, encoding='latin')
df.columns = ['label', 'id', 'date', 'query', 'user', 'tweet']

# Data reduction
df = df.drop(['id', 'date', 'query', 'user'], axis=1)
```

In [3]:
```
labels_dict = {0:'Negative', 2:'Neutral', 4:'Positive'}

def convert_labels(label):
    return labels_dict[label]

df.label = df.label.apply(lambda x: convert_labels(x))
df
```

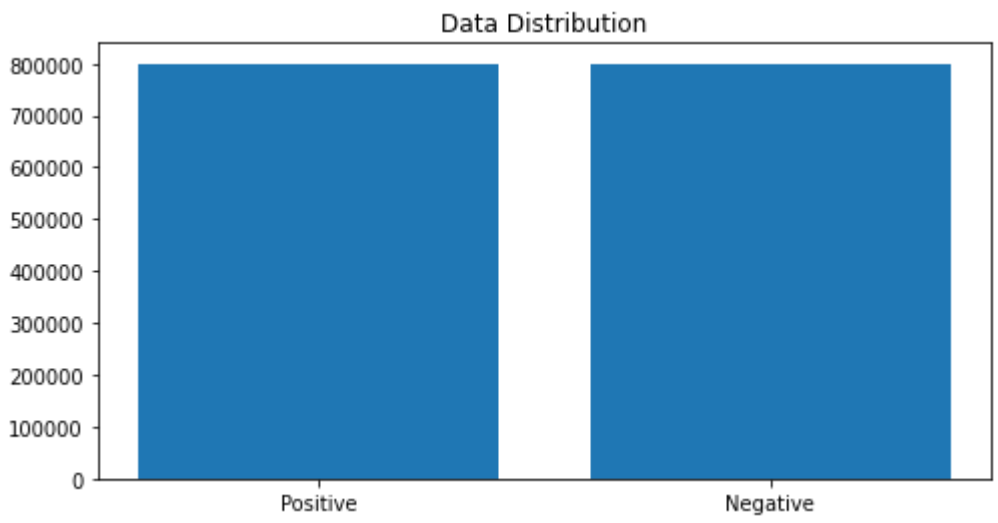
Out[3]:

	label	tweet
0	Negative	@switchfoot http://twitpic.com/2y1zl - Awww, t...
1	Negative	is upset that he can't update his Facebook by ...
2	Negative	@Kenichan I dived many times for the ball. Man...
3	Negative	my whole body feels itchy and like its on fire
4	Negative	@nationwideclass no, it's not behaving at all....
...	...	...
1599995	Positive	Just woke up. Having no school is the best fee...
1599996	Positive	TheWDB.com - Very cool to hear old Walt interv...
1599997	Positive	Are you ready for your MoJo Makeover? Ask me f...
1599998	Positive	Happy 38th Birthday to my boo of alll time!!! ...
1599999	Positive	happy #charitytuesday @theNSPCC @SparksCharity...

1600000 rows × 2 columns

In [4]:
```
instances = df.label.value_counts()

plt.figure(figsize=(8,4))
plt.bar(instances.index, instances.values)
plt.title("Data Distribution")
```

Out[4]:
```
Text(0.5, 1.0, 'Data Distribution')
```



Preprocess

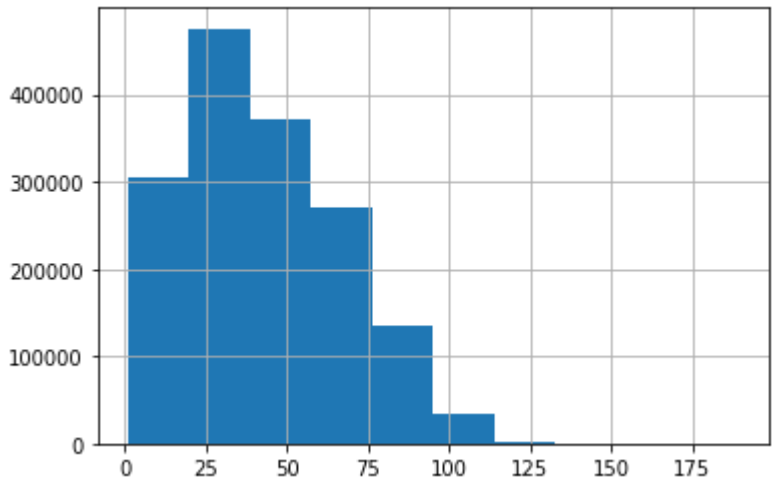
In [5]:
```
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.stem import SnowballStemmer

stop_words = stopwords.words('english')
stemmer = SnowballStemmer('english')

punctuations_and_dummies = "@\S+|https?:\S+|http?:\S|[^A-Za-z0-9]+"

def preprocess(tweet, will_be_stemmed=False):
        tweet = re.sub(punctuations_and_dummies, ' ', str(tweet).lower()).strip()
        tokens = []
        for token in tweet.split():
            if token not in stop_words:
                if will_be_stemmed:
                    tokens.append(stemmer.stem(token))
                else:
                    tokens.append(token)
        return " ".join(tokens)
```

```
df.tweet = df.tweet.apply(lambda tw: preprocess(tw))
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\gokse\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

In [6]:
```
# Remove 0 length tweets
df = df[df.iloc[:,1].astype(str).str.len()!=0]
```

In [7]:
```
tweets_len = [len(x) for x in df['tweet']]
pd.Series(tweets_len).hist()
plt.show()
pd.Series(tweets_len).describe()
```



Out[7]:
```
count    1.592328e+06
mean     4.279740e+01
std      2.415896e+01
min      1.000000e+00
25%      2.300000e+01
50%      3.900000e+01
75%      6.000000e+01
max      1.890000e+02
dtype: float64
```

Number of Letters

In [8]:
```
all_str = ""
for i in df.tweet:
    all_str += i
```

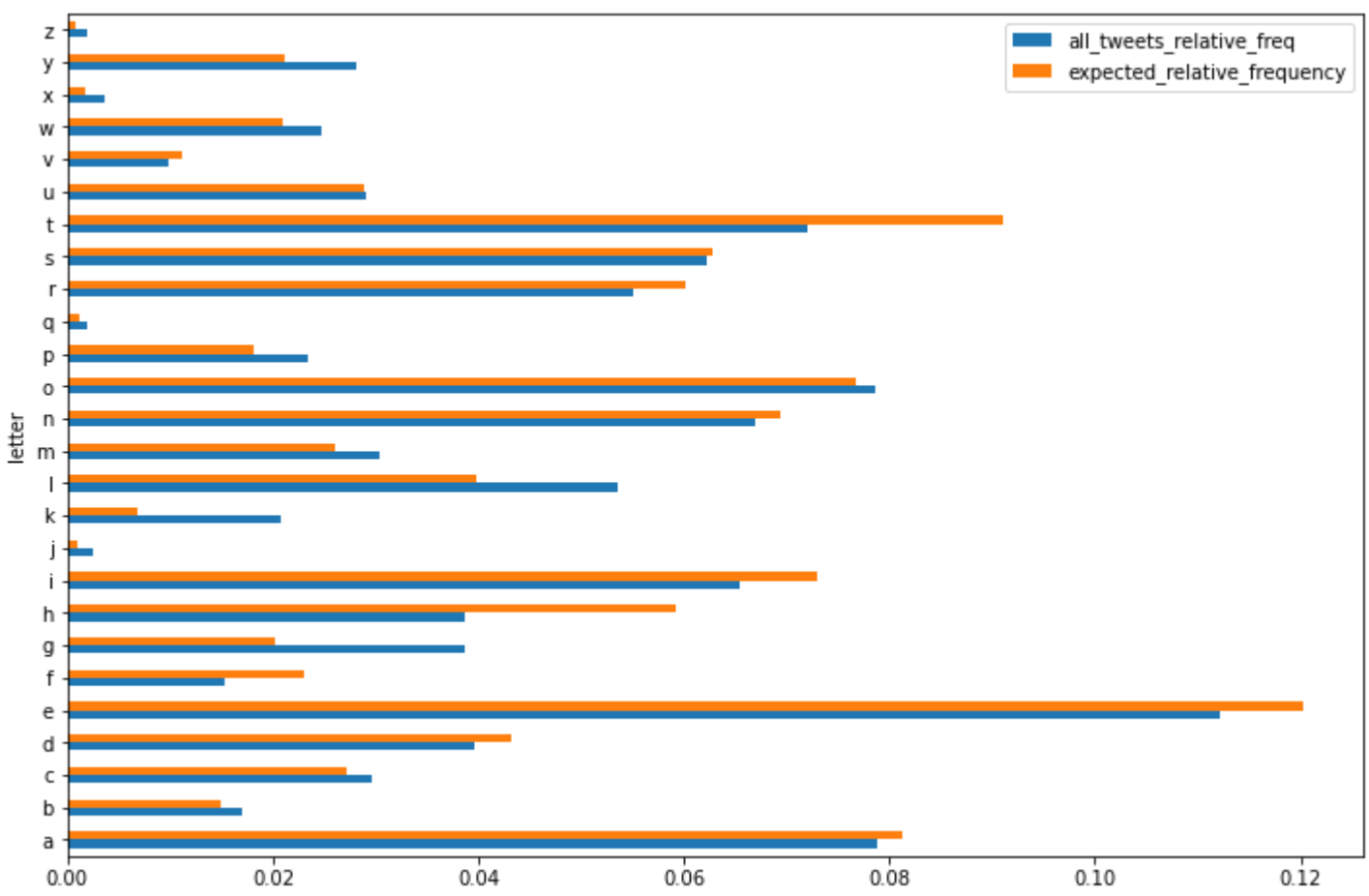
In [9]:
```
from collections import Counter

letter_list = list(all_str)
my_counter = Counter(letter_list)

letter_df = pd.DataFrame.from_dict(my_counter, orient='index').reset_index()
letter_df = letter_df.rename(columns={'index':'letter', 0:'frequency'})
letter_df = letter_df.loc[letter_df['letter'].isin(['a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v'
letter_df['all_tweets_relative_freq']=letter_df['frequency']/letter_df['frequency'].sum()
letter_df = letter_df.sort_values('letter')

english = pd.read_csv('data/letter_frequency_en_US.csv')
english['expected_relative_frequency'] = english['count']/english['count'].sum()
english = english.drop(['count'], axis=1)

letter_df = pd.merge(letter_df, english, on='letter')
letter_df['expected'] = np.round(letter_df['expected_relative_frequency']*letter_df['frequency'].sum(),0)
letter_df = letter_df.reset_index().drop(['index'], axis=1)
letter_df
```

Out[9]:

	letter	frequency	all_tweets_relative_freq	expected_relative_frequency	expected
0	a	4547601	0.078816	0.081238	4687379.0
1	b	975326	0.016904	0.014893	859300.0
2	c	1705409	0.029557	0.027114	1564464.0
3	d	2289515	0.039680	0.043192	2492128.0
4	e	6471295	0.112156	0.120195	6935169.0
5	f	878849	0.015232	0.023039	1329304.0
6	g	2231747	0.038679	0.020257	1168838.0
7	h	2234047	0.038719	0.059215	3416628.0
8	i	3779579	0.065505	0.073054	4215160.0
9	j	143817	0.002493	0.001031	59502.0
10	k	1197291	0.020751	0.006895	397842.0
11	l	3095498	0.053649	0.039785	2295581.0
12	m	1754377	0.030406	0.026116	1506861.0
13	n	3861185	0.066919	0.069478	4008801.0
14	o	4534414	0.078587	0.076812	4431963.0
15	p	1351301	0.023420	0.018189	1049517.0
16	q	115059	0.001994	0.001125	64883.0
17	r	3179237	0.055100	0.060213	3474231.0
18	s	3595565	0.062316	0.062808	3623936.0
19	t	4153946	0.071993	0.090986	5249801.0
20	u	1676743	0.029060	0.028776	1660364.0
21	v	566733	0.009822	0.011075	639015.0
22	w	1422401	0.024652	0.020949	1208717.0
23	x	203131	0.003521	0.001728	99698.0
24	y	1620980	0.028094	0.021135	1219478.0
25	z	114027	0.001976	0.000702	40512.0

In [10]:
```
letter_df.plot(x="letter", y=["all_tweets_relative_freq", "expected_relative_frequency"], kind="barh", figsize=(12,8))
```

Out[10]:
```
<AxesSubplot:ylabel='letter'>
```



Compare the Observed Frequencies with the Expected Frequencies in English

In [12]:
```
letter_df[['frequency', 'expected']].corr()
```

Out[12]:

	frequency	expected
frequency	1.000000	0.967421
expected	0.967421	1.000000

In [13]:
```
df1 = df.copy()

df1['number_of_characters'] = [len(tw) for tw in df1.tweet]
df1
```

Out[13]:

	label	tweet	number_of_characters
0	Negative	awww bummer shoulda got david carr third day	44
1	Negative	upset update facebook texting might cry result...	69
2	Negative	dived many times ball managed save 50 rest go ...	52
3	Negative	whole body feels itchy like fire	32
4	Negative	behaving mad see	16
...	...	...	...
1599995	Positive	woke school best feeling ever	29
1599996	Positive	thewdb com cool hear old walt interviews	40
1599997	Positive	ready mojo makeover ask details	31
1599998	Positive	happy 38th birthday boo alll time tupac amaru ...	52
1599999	Positive	happy charitytuesday thenspcc sparkscharity sp...	57

1592328 rows × 3 columns

In [14]:
```
df1.number_of_characters.max()
```

Out[14]:
```
189
```

In [15]:
```
df1.number_of_characters.min()
```

Out[15]:
```
1
```

In [16]:
```
df1.number_of_characters.mean()
```

Out[16]:
```
42.7974010379771
```

In [17]:
```
df1.number_of_characters.std()
```

Out[17]:
```
24.158961650697616
```

Number of Words

In [18]:
```
df1['number_of_words'] = [len(tw.split()) for tw in df1.tweet]
df1
```

Out[18]:

	label	tweet	number_of_characters	number_of_words
0	Negative	awww bummer shoulda got david carr third day	44	8
1	Negative	upset update facebook texting might cry result...	69	11
2	Negative	dived many times ball managed save 50 rest go ...	52	10
3	Negative	whole body feels itchy like fire	32	6
4	Negative	behaving mad see	16	3
...	...	...	...	...
1599995	Positive	woke school best feeling ever	29	5
1599996	Positive	thewdb com cool hear old walt interviews	40	7
1599997	Positive	ready mojo makeover ask details	31	5
1599998	Positive	happy 38th birthday boo alll time tupac amaru ...	52	9
1599999	Positive	happy charitytuesday thenspcc sparkscharity sp...	57	5

1592328 rows × 4 columns

In [19]:
```
df1.number_of_words.max()
```

Out[19]:
```
50
```

In [20]:
```
df1.number_of_words.min()
```

Out[20]:
```
1
```

In [21]:
```
df1.number_of_words.mean()
```

Out[21]:
```
7.244474128445898
```

In [22]:
```
df1.number_of_words.std()
```

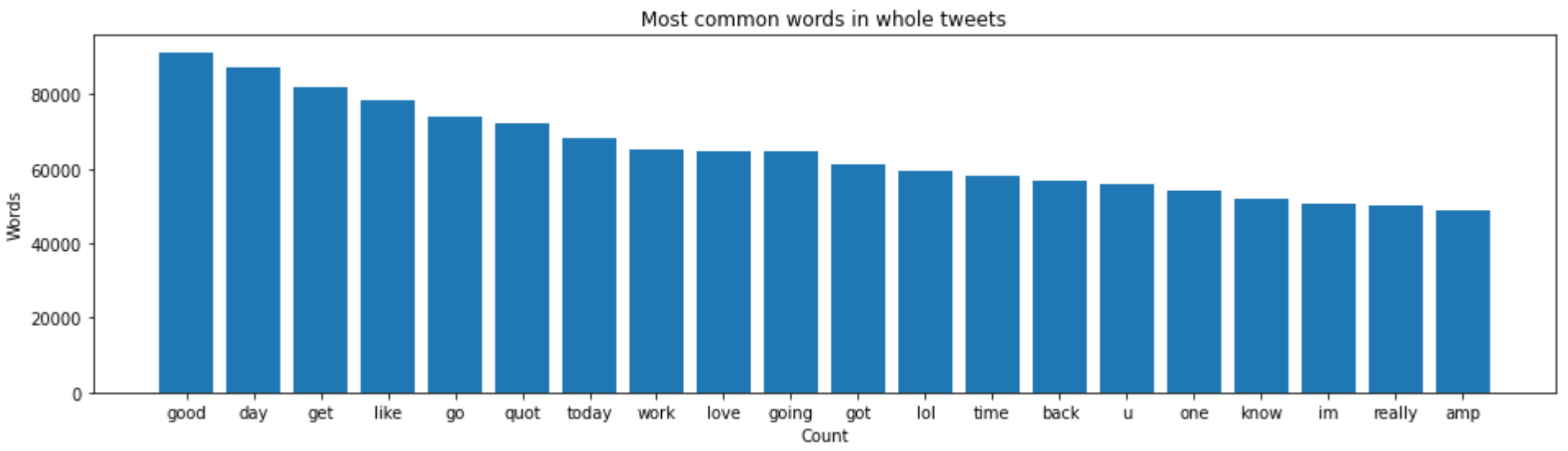
Out[22]:
```
4.030421805719796
```

Positives + Negatives

In [23]:
```
import collections
from wordcloud import WordCloud
```

```
from nltk import word_tokenize, sent_tokenize
from nltk.util import ngrams

all_tweets = ' '.join(df['tweet'].str.lower())

f_words = [word for word in all_tweets.split()]
counted_words = collections.Counter(f_words)

words = []
counts = []
for letter, count in counted_words.most_common(20):
    words.append(letter)
    counts.append(count)

plt.figure(figsize = (16, 4))
plt.title('Most common words in whole tweets')
plt.xlabel('Count')
plt.ylabel('Words')
plt.bar(words, counts)
```

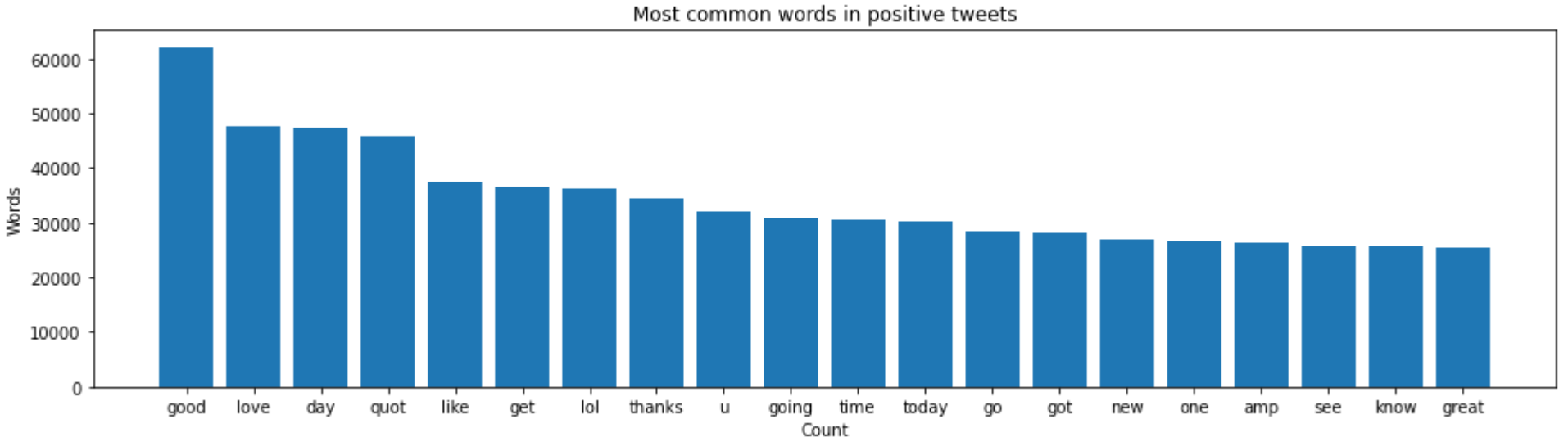
Out[23]: <BarContainer object of 20 artists>



Positives

In [24]:
```
all_tweets = ' '.join(df[df.label == 'Positive'].tweet.str.lower())

f_words = [word for word in all_tweets.split()]
counted_words = collections.Counter(f_words)

words = []
counts = []
for letter, count in counted_words.most_common(20):
    words.append(letter)
    counts.append(count)

plt.figure(figsize = (16, 4))
plt.title('Most common words in positive tweets')
plt.xlabel('Count')
plt.ylabel('Words')
plt.bar(words, counts)
```

Out[24]: <BarContainer object of 20 artists>



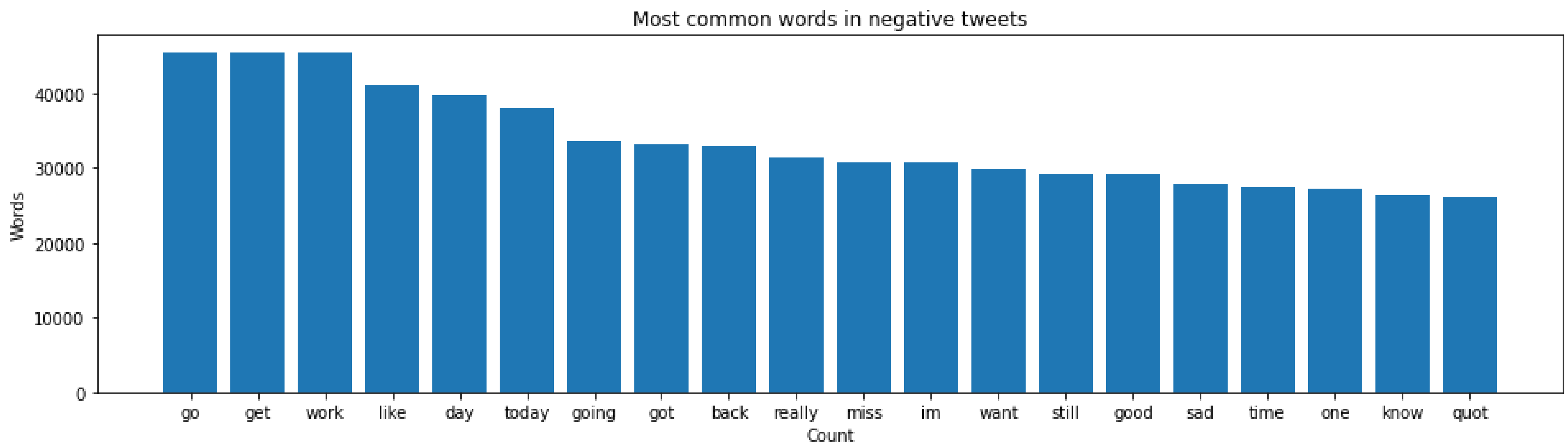
In [25]:
```
plt.figure(figsize = (25, 25))
plt.axis('off')
wordcloud_fig = WordCloud(max_words = 2000 , width = 1600 , height = 800, background_color ='white', min_font_size = 10).generate(" ".join(
plt.imshow(wordcloud_fig, interpolation = 'bilinear')
```

Out[25]: <matplotlib.image.AxesImage at 0x24db9ae5940>

Negatives

In [26]:
```
all_tweets = ' '.join(df[df.label == 'Negative'].tweet.str.lower())

f_words = [word for word in all_tweets.split()]
counted_words = collections.Counter(f_words)

words = []
counts = []
for letter, count in counted_words.most_common(20):
    words.append(letter)
    counts.append(count)

plt.figure(figsize = (16, 4))
plt.title('Most common words in negative tweets')
plt.xlabel('Count')
plt.ylabel('Words')
plt.bar(words, counts)
```

Out[26]:
```
<BarContainer object of 20 artists>
```



In [27]:
```
from wordcloud import WordCloud

plt.figure(figsize = (25, 25))
plt.axis('off')
wordcloud_fig = WordCloud(max_words = 2000 , width = 1600 , height = 800, background_color ='white', min_font_size = 10).generate(" ".join(
plt.imshow(wordcloud_fig, interpolation = 'bilinear')
```

Out[27]:
```
<matplotlib.image.AxesImage at 0x24dac1656a0>
```

Training Data and Test Data Splitting

In [28]:
```python
from sklearn.model_selection import train_test_split

train_data, test_data = train_test_split(df, test_size=0.2, random_state=7)
print('Training Data', len(train_data), 'Test Data', len(test_data))

train_data.head(10)
```

```
Training Data 1273862 Test Data 318466
```

Out[28]:

	label	tweet
927561	Positive	thank
810082	Positive	shakin money tree headed back car business
1153331	Positive	tasmania okay chat ya later
988182	Positive	yeah 100 tab storm towelled raiders awesome
543405	Negative	happy mari asleep gotta start laundry b4 wakes...
22855	Negative	wanting go city raining gotta homework 2
483675	Negative	sitting home bored x
667231	Negative	miss amy cards
1370823	Positive	dinner hubby two us 10 coupo
94439	Negative	know prob running around

In [29]:
```python
test_data.head(10)
```

Out[29]:

	label	tweet
1414233	Positive	nice job thanks
181653	Negative	way wanna shower n get pretty
788422	Negative	woke early think anyone completely bored
76646	Negative	lucky u scheduled today happy camper lol
972366	Positive	way cute curt hope school raised tons money lo...
797869	Negative	stupid braces making lips dry
1129919	Positive	good miss
1036078	Positive	past days sucked hopefully weekend happier
550532	Negative	marcus doubt megan fox good 2 years might get ...
1365388	Positive	going school contempt fash exams going conquer...

Tokenization

In [30]:
```python
from keras.preprocessing.text import Tokenizer

tokenizer = Tokenizer()
tokenizer.fit_on_texts(train_data.tweet)
word_index = tokenizer.word_index
vocab_size = len(tokenizer.word_index) + 1
print("Vocabulary Size :", vocab_size)
```

```
Using TensorFlow backend.
Vocabulary Size : 290458
```

GLOVE Embedding

In [31]:
```
MODELS_PATH = 'models'
EMBEDDING_DIMENSION = 300
```

In [32]:
```
import tensorflow as tf

BATCH_SIZE = 1024
EPOCHS = 10
LR = 1e-3

embeddings_index = {}

glove_file = open('glove/glove.6B.300d.txt', encoding='utf8')
for line in glove_file:
    values = line.split()
    word = value = values[0]
    coefficients = np.asarray(values[1:], dtype='float32')
    embeddings_index[word] = coefficients
glove_file.close()

print('%s word vectors.' % len(embeddings_index))


embedding_matrix = np.zeros((vocab_size, EMBEDDING_DIMENSION))
for word, i in word_index.items():
    embedding_vector = embeddings_index.get(word)
    if embedding_vector is not None:
        embedding_matrix[i] = embedding_vector

embedding_layer = tf.keras.layers.Embedding(vocab_size, EMBEDDING_DIMENSION, weights=[embedding_matrix], input_length=30, trainable=False)
```

```
400000 word vectors.
```

In [ ]:

In [ ]: