KaggleCompetition

December 31, 2023

1 Kaggle Competition

1.1 Data Preprocessing

[ ]:
```
import pandas as pd
import numpy as np
```

1.1.1 Read Files

[ ]:
```
data_id = pd.read_csv('data_identification.csv')
data_id.shape
```

[ ]:
```
(1867535, 2)
```

[ ]:
```
emotion = pd.read_csv('emotion.csv')
emotion.shape
```

[ ]:
```
(1455563, 2)
```

[ ]:
```
import json
```

[ ]:
```
tweets = []

with open('tweets_DM.json', 'r') as file:
    for line in file:
        tweet = json.loads(line)
        tweets.append(tweet)
```

[ ]:
```
tweets_df = pd.DataFrame(tweets)
source_data = pd.json_normalize(tweets_df['_source'])
tweets_df = tweets_df.drop('_source', axis=1).join(source_data)
tweets_df.rename(columns={'tweet.tweet_id': 'tweet_id', 'tweet.hashtags':␣
↪'hashtags'}, inplace=True)
tweets_df
```

[ ]:
```
          _score          _index           _crawldate   _type  \
0            391  hashtag_tweets  2015-05-23 11:42:47  tweets
1            433  hashtag_tweets  2016-01-28 04:52:09  tweets
```

```
2           232  hashtag_tweets  2017-12-25 04:39:20  tweets
3           376  hashtag_tweets  2016-01-24 23:53:05  tweets
4           989  hashtag_tweets  2016-01-08 17:18:59  tweets
…           …              …                    …     …
1867530     827  hashtag_tweets  2015-05-12 12:51:52  tweets
1867531     368  hashtag_tweets  2017-10-02 17:54:04  tweets
1867532     498  hashtag_tweets  2016-10-10 11:04:32  tweets
1867533     840  hashtag_tweets  2016-09-02 14:25:06  tweets
1867534     360  hashtag_tweets  2016-11-16 01:40:07  tweets

                                hashtags  tweet_id  \
0                              [Snapchat]  0x376b20
1           [freepress, TrumpLegacy, CNN]  0x2d5350
2                             [bibleverse]  0x28b412
3                                       []  0x1cd5b0
4                                       []  0x2de201
…                                      …       …
1867530  [mixedfeeling, butimTHATperson]  0x316b80
1867531                                 []  0x29d0cb
1867532                                 []  0x2a6a4f
1867533                                 []  0x24faed
1867534                      [Sundayvibes]  0x34be8c

                                                tweet.text
0        People who post "add me on #Snapchat" must be …
1        @brianklaas As we see, Trump is dangerous to #…
2        Confident of your obedience, I write to you, k…
3                      Now ISSA is stalking Tasha    <LH>
4        "Trust is not the same as faith. A friend is s…
…                                                      …
1867530  When you buy the last 2 tickets remaining for …
1867531  I swear all this hard work gone pay off one da…
1867532  @Parcel2Go no card left when I wasn't in so I …
1867533  Ah, corporate life, where you can date <LH> us…
1867534             Blessed to be living #Sundayvibes <LH>

[1867535 rows x 7 columns]
```

1.1.2 Clean the Data Set

[ ]:
```
tweets_df['_index'].unique()
```

[ ]:
```
array(['hashtag_tweets'], dtype=object)
```

[ ]:
```
tweets_df['_type'].unique()
```

[ ]:
```
array(['tweets'], dtype=object)
```

since the _index, _type, and the _crawldate columns does not provide useful information for understanding the emotion of the text, we can drop them

[ ]:
```
tweets_df = tweets_df.drop(columns=['_index', '_crawldate', '_type', '_score'])
tweets_df
```

[ ]:
```
                                hashtags  tweet_id  \
0                              [Snapchat]  0x376b20
1          [freepress, TrumpLegacy, CNN]  0x2d5350
2                             [bibleverse]  0x28b412
3                                       []  0x1cd5b0
4                                       []  0x2de201
…                                      …       …
1867530  [mixedfeeling, butimTHATperson]  0x316b80
1867531                                 []  0x29d0cb
1867532                                 []  0x2a6a4f
1867533                                 []  0x24faed
1867534                      [Sundayvibes]  0x34be8c

                                                tweet.text
0        People who post "add me on #Snapchat" must be …
1        @brianklaas As we see, Trump is dangerous to #…
2        Confident of your obedience, I write to you, k…
3                      Now ISSA is stalking Tasha    <LH>
4        "Trust is not the same as faith. A friend is s…
…                                                      …
1867530  When you buy the last 2 tickets remaining for …
1867531  I swear all this hard work gone pay off one da…
1867532  @Parcel2Go no card left when I wasn't in so I …
1867533  Ah, corporate life, where you can date <LH> us…
1867534             Blessed to be living #Sundayvibes <LH>

[1867535 rows x 3 columns]
```

then combine the labels of emotion and tweet identification to the data frame

[ ]:
```
merged_df = pd.merge(tweets_df, emotion, on='tweet_id', how='left')
merged_df = pd.merge(merged_df, data_id, on='tweet_id', how='left')
merged_df
```

[ ]:
```
                                hashtags  tweet_id  \
0                              [Snapchat]  0x376b20
1          [freepress, TrumpLegacy, CNN]  0x2d5350
2                             [bibleverse]  0x28b412
3                                       []  0x1cd5b0
4                                       []  0x2de201
…                                      …       …
1867530  [mixedfeeling, butimTHATperson]  0x316b80
```

```
1867531                                 []  0x29d0cb
1867532                                 []  0x2a6a4f
1867533                                 []  0x24faed
1867534                     [Sundayvibes]  0x34be8c

                                                tweet.text       emotion  \
0        People who post "add me on #Snapchat" must be …  anticipation
1        @brianklaas As we see, Trump is dangerous to #…       sadness
2        Confident of your obedience, I write to you, k…           NaN
3                      Now ISSA is stalking Tasha    <LH>          fear
4        "Trust is not the same as faith. A friend is s…           NaN
…                                                      …           …
1867530  When you buy the last 2 tickets remaining for …           NaN
1867531  I swear all this hard work gone pay off one da…           NaN
1867532  @Parcel2Go no card left when I wasn't in so I …           NaN
1867533  Ah, corporate life, where you can date <LH> us…           joy
1867534              Blessed to be living #Sundayvibes <LH>           joy

        identification
0                train
1                train
2                 test
3                train
4                 test
…                …
1867530           test
1867531           test
1867532           test
1867533          train
1867534          train

[1867535 rows x 5 columns]
```

[ ]:
```
print(merged_df.isnull().sum())

# the testing emotions are not given
```

```
hashtags              0
tweet_id              0
tweet.text            0
emotion          411972
identification         0
dtype: int64
```

to process the text part, some first-step cleaning was done to the data including removing html tags, mentions (any @_____), hashtag in the text, urls, characters and numbers

[ ]:
```python
import re

def clean_text(text):
    text = re.sub(r'<[^>]+>', '', text)  # Remove HTML tags
    text = re.sub(r'@\w+', '', text)  # Remove mentions
    text = text.replace('#', '') # Remove hashtags
    text = re.sub(r'http\S+', '', text)  # Remove URLs
    text = re.sub(r'[^A-Za-z0-9\s]', '', text)  # Remove special chars
    text = re.sub(r'\d+', '', text) # Remove numbers
    text = text.strip()
    text = text.lower()
    return text
```

[ ]:
```python
merged_df['text'] = merged_df['tweet.text'].apply(clean_text)
```

[ ]:
```python
tweets_cleaned = merged_df.drop('tweet.text', axis=1)
tweets_cleaned
```

[ ]:
```
                                hashtags  tweet_id       emotion  \
0                              [Snapchat]  0x376b20  anticipation
1           [freepress, TrumpLegacy, CNN]  0x2d5350       sadness
2                             [bibleverse]  0x28b412           NaN
3                                       []  0x1cd5b0          fear
4                                       []  0x2de201           NaN
…                                      …       …            …
1867530  [mixedfeeling, butimTHATperson]  0x316b80           NaN
1867531                                 []  0x29d0cb           NaN
1867532                                 []  0x2a6a4f           NaN
1867533                                 []  0x24faed           joy
1867534                     [Sundayvibes]  0x34be8c           joy

        identification                                               text
0                train  people who post add me on snapchat must be deh…
1                train  as we see trump is dangerous to freepress arou…
2                 test  confident of your obedience i write to you kno…
3                train                          now issa is stalking tasha
4                 test  trust is not the same as faith a friend is som…
…                …                                                    …
1867530           test  when you buy the last  tickets remaining for a…
1867531           test    i swear all this hard work gone pay off one day
1867532           test  no card left when i wasnt in so i have no idea…
1867533          train  ah corporate life where you can date  using ju…
1867534          train                   blessed to be living sundayvibes

[1867535 rows x 5 columns]
```

[ ]:
```
for i in range(10):
    print(f"Tweet {i+1}: {merged_df['tweet.text'].iloc[i]}\n")
    print(f"Tweet {i+1}: {tweets_cleaned['text'].iloc[i]}\n")
```

```
Tweet 1: People who post "add me on #Snapchat" must be dehydrated. Cuz man…
that's <LH>

Tweet 1: people who post add me on snapchat must be dehydrated cuz man thats

Tweet 2: @brianklaas As we see, Trump is dangerous to #freepress around the
world. What a <LH> <LH> #TrumpLegacy.  #CNN

Tweet 2: as we see trump is dangerous to freepress around the world what a
trumplegacy  cnn

Tweet 3: Confident of your obedience, I write to you, knowing that you will do
even more than I ask. (Philemon 1:21) 3/4 #bibleverse <LH> <LH>

Tweet 3: confident of your obedience i write to you knowing that you will do
even more than i ask philemon  bibleverse

Tweet 4: Now ISSA is stalking Tasha  <LH>

Tweet 4: now issa is stalking tasha

Tweet 5: "Trust is not the same as faith. A friend is someone you trust. Putting
faith in anyone is a mistake." ~ Christopher Hitchens <LH> <LH>

Tweet 5: trust is not the same as faith a friend is someone you trust putting
faith in anyone is a mistake  christopher hitchens

Tweet 6: @RISKshow @TheKevinAllison Thx for the BEST TIME tonight. What stories!
Heartbreakingly <LH> #authentic #LaughOutLoud good!!

Tweet 6: thx for the best time tonight what stories heartbreakingly  authentic
laughoutloud good

Tweet 7: Still waiting on those supplies Liscus. <LH>

Tweet 7: still waiting on those supplies liscus

Tweet 8: Love knows no gender.  <LH>

Tweet 8: love knows no gender

Tweet 9: @DStvNgCare @DStvNg More highlights are being shown than actual sports!
Who watches triathlon highlights anyway? <LH> #LeagueCup
```

```
Tweet 9: more highlights are being shown than actual sports who watches
triathlon highlights anyway  leaguecup

Tweet 10: When do you have enough ? When are you satisfied ? Is you goal really
all about money ?  #materialism #money #possessions <LH>

Tweet 10: when do you have enough  when are you satisfied  is you goal really
all about money   materialism money possessions
```

1.2 Exploratory Data Analysis (EDA)

this part will include some data visualizations and frequency analysis of the text data

[ ]:
```python
import seaborn as sns
import matplotlib.pyplot as plt
import nltk
```

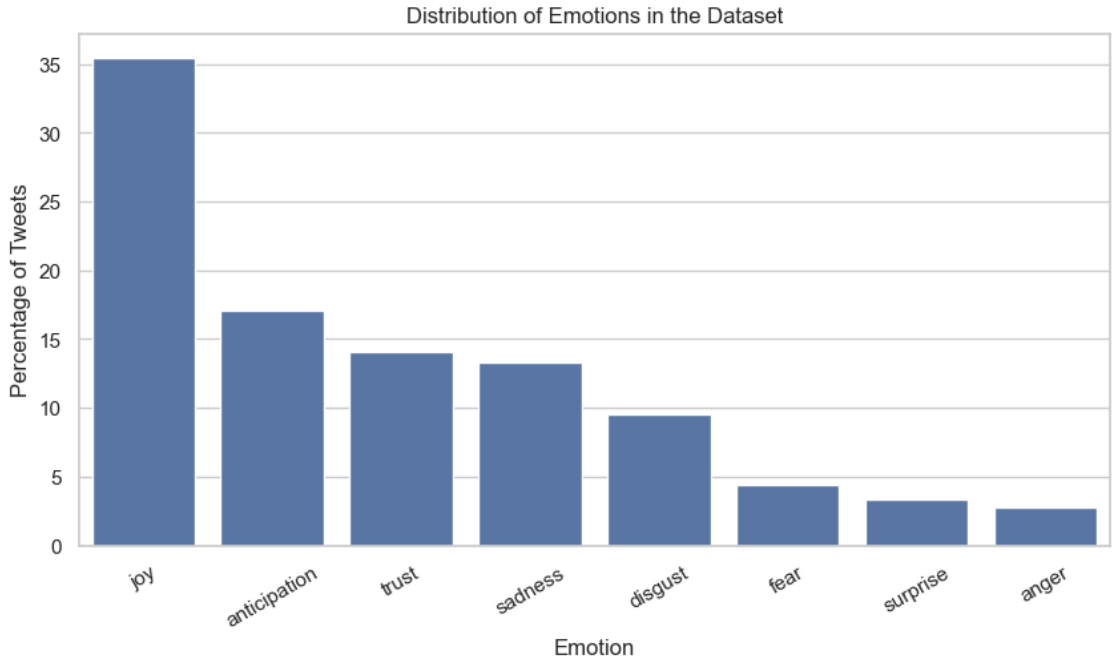
1.2.1 Basic Structure

[ ]:
```python
tweets_cleaned.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1867535 entries, 0 to 1867534
Data columns (total 5 columns):
 #   Column          Dtype
---  ------          -----
 0   hashtags        object
 1   tweet_id        object
 2   emotion         object
 3   identification  object
 4   text            object
dtypes: object(5)
memory usage: 71.2+ MB
```

1.2.2 Distribution of Emotions

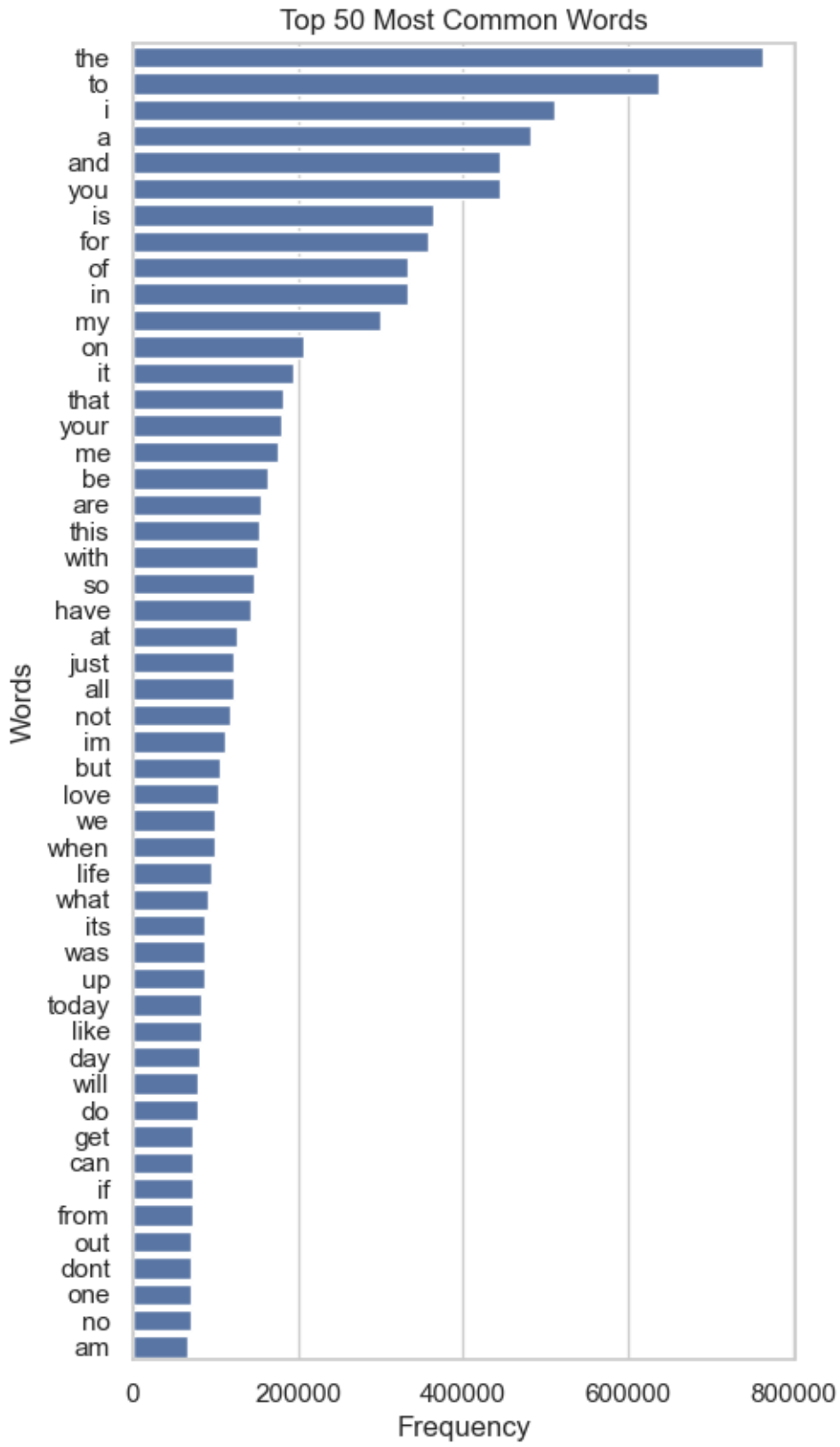
[ ]:
```python
emotion_counts = tweets_cleaned['emotion'].value_counts(normalize=True) * 100
sns.set(style="whitegrid")
plt.figure(figsize=(10, 5))
sns.barplot(x=emotion_counts.index, y=emotion_counts.values)
plt.title('Distribution of Emotions in the Dataset')
plt.ylabel('Percentage of Tweets')
plt.xlabel('Emotion')
plt.xticks(rotation=30)
plt.show()
```



- distribution of the data set is inbalanced can lead to
  - model bias: the model may become biased towards the more frequent classes. This is because there is more data for the model to learn from for these classes, leading to better performance on them compared to the less frequent classes
  - poor generalization: may not generalize well to new, unseen data, especially for the underrepresented classes

1.2.3 Word Frequency Analysis

[ ]:
```
from collections import Counter

topNwords = 50
words = ' '.join(tweets_cleaned['text']).split()
word_freq = Counter(words)
most_common_words = word_freq.most_common(topNwords)

plt.figure(figsize=(5, 10))
sns.barplot(x=[val[1] for val in most_common_words], y=[val[0] for val in␣
 ↪most_common_words])
plt.title(f'Top {topNwords} Most Common Words')
plt.xlabel('Frequency')
plt.ylabel('Words')
plt.show()
```


Top 50 Most Common Words
Words
the
to
i
a
and
you
is
for
of
in
my
on
it
that
your
me
be
are
this
with
so
have
at
just
all
not
im
but
love
we
when
life
what
its
was
up
today
like
day
will
do
get
can
if
from
out
dont
one
no
am
0
200000
400000
600000
800000
Frequency

- it is discovered that these top words are not very benefitial for emotion classification
- decided to remove stop words first

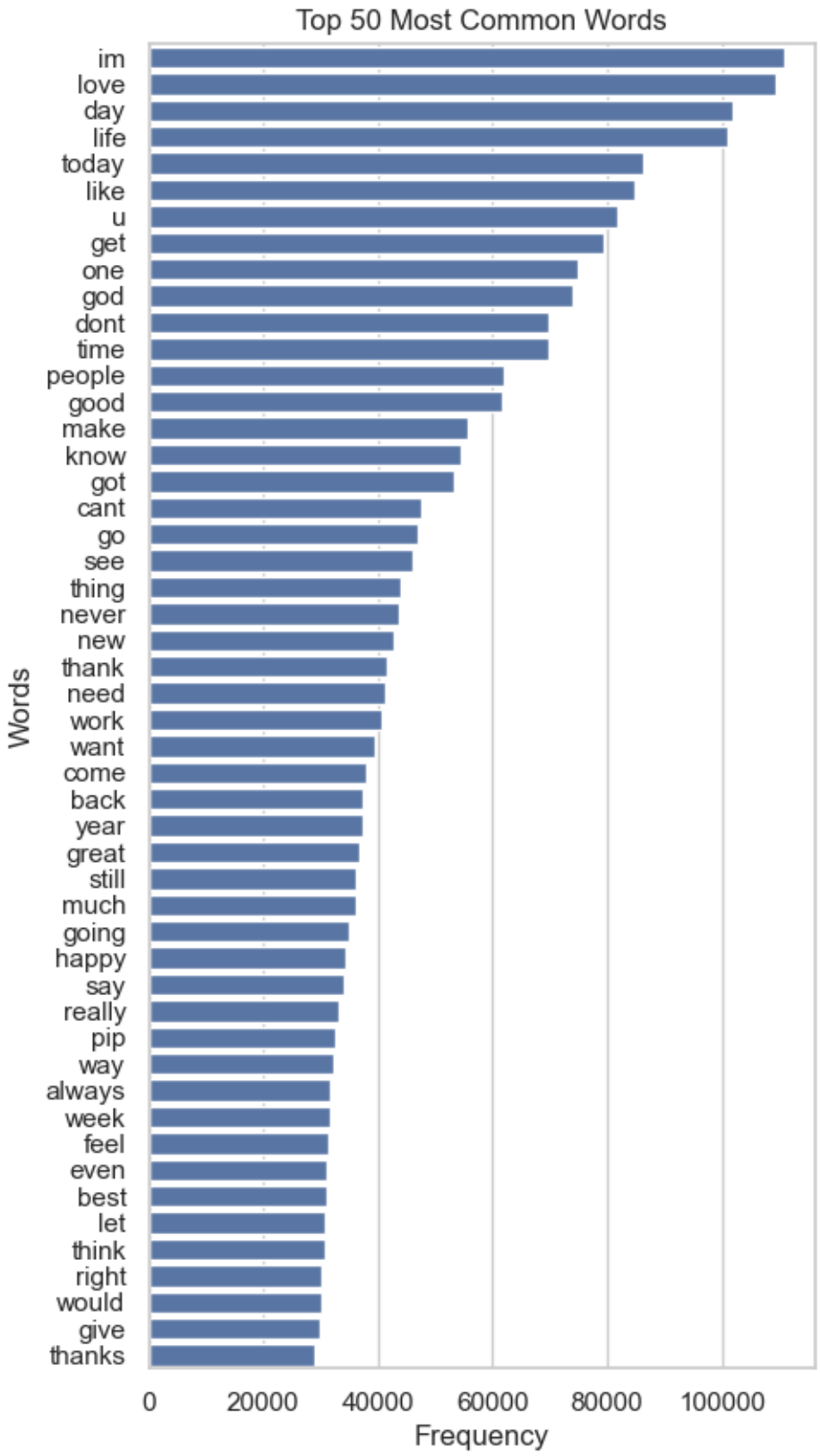
[ ]:
```
from nltk.corpus import stopwords

nltk.download('stopwords')
stop = set(stopwords.words('english'))

tweets_cleaned['text'] = tweets_cleaned['text'].apply(lambda x: ' '.join([word␣
 ↪for word in x.split() if word not in (stop)]))
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/vvnliu/nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
```

[ ]:
```
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize

nltk.download('wordnet')
nltk.download('punkt')

lemmatizer = WordNetLemmatizer()

def lemmatize_text(text):
    return ' '.join([lemmatizer.lemmatize(word) for word in␣
 ↪word_tokenize(text)])

tweets_cleaned['text'] = tweets_cleaned['text'].apply(lemmatize_text)
```

```
[nltk_data] Downloading package wordnet to /Users/vvnliu/nltk_data…
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package punkt to /Users/vvnliu/nltk_data…
[nltk_data]   Package punkt is already up-to-date!
```

[ ]:
```
topNwords = 50
words = ' '.join(tweets_cleaned['text']).split()
word_freq = Counter(words)
most_common_words = word_freq.most_common(topNwords)

plt.figure(figsize=(5, 10))
sns.barplot(x=[val[1] for val in most_common_words], y=[val[0] for val in␣
 ↪most_common_words])
plt.title(f'Top {topNwords} Most Common Words')
plt.xlabel('Frequency')
plt.ylabel('Words')
plt.show()
```


Top 50 Most Common Words
Words
im
love
day
life
today
like
u
get
one
god
dont
time
people
good
make
know
got
cant
go
see
thing
never
new
thank
need
work
want
come
back
year
great
still
much
going
happy
say
really
pip
way
always
week
feel
even
best
let
think
right
would
give
thanks
0
20000
40000
60000
80000
100000
Frequency

- high frequency words now turned into more meaningful words

[ ]:
```
unique_emotions = tweets_cleaned['emotion'].unique()
emotion_specific_words = {}

for emotion in unique_emotions:
    specific_tweets = tweets_cleaned[tweets_cleaned['emotion'] ==␣
 ↪emotion]['text']
    word_freq = Counter(' '.join(specific_tweets).split())
    emotion_specific_words[emotion] = word_freq.most_common(5)

for emotion, common_words in emotion_specific_words.items():
    print(f"Emotion: {emotion}, Common words: {common_words}")
```

```
Emotion: anticipation, Common words: [('life', 32647), ('god', 29806), ('dream',
16431), ('u', 14663), ('come', 12479)]
Emotion: sadness, Common words: [('im', 10928), ('like', 10198), ('sad', 8758),
('please', 8698), ('dont', 8486)]
Emotion: nan, Common words: []
Emotion: fear, Common words: [('im', 5028), ('issa', 3402), ('lawrence', 3278),
('like', 3160), ('get', 2880)]
Emotion: joy, Common words: [('love', 58525), ('today', 36218), ('day', 33856),
('pip', 32376), ('life', 29614)]
Emotion: anger, Common words: [('im', 3282), ('get', 2819), ('dont', 2097),
('like', 1964), ('people', 1865)]
Emotion: trust, Common words: [('day', 16214), ('today', 13754), ('im', 13419),
('life', 12256), ('love', 9308)]
Emotion: disgust, Common words: [('like', 8583), ('u', 6879), ('get', 6470),
('im', 6455), ('people', 6428)]
Emotion: surprise, Common words: [('im', 2872), ('like', 2577), ('get', 1999),
('dont', 1980), ('people', 1738)]
```

1.3 Model Training

[ ]:
```
train_data = tweets_cleaned[tweets_cleaned['identification'] == 'train']
test_data = tweets_cleaned[tweets_cleaned['identification'] == 'test']
```

[ ]:
```
train_data
```

[ ]:

	hashtags	tweet_id	emotion	identification	\
0	[Snapchat]	0x376b20	anticipation	train	
1	[freepress, TrumpLegacy, CNN]	0x2d5350	sadness	train	
3	[]	0x1cd5b0	fear	train	
5	[authentic, LaughOutLoud]	0x1d755c	joy	train	
6	[]	0x2c91a8	anticipation	train	
…	…	…	…	…	

```
1867526              [NoWonder, Happy]  0x321566           joy          train
1867527                             []  0x38959e           joy          train
1867528                    [blessyou]  0x2cbca6           joy          train
1867533                             []  0x24faed           joy          train
1867534                   [Sundayvibes]  0x34be8c           joy          train

                                                      text
0        people post add snapchat must dehydrated cuz m…
1        see trump dangerous freepress around world tru…
3                                     issa stalking tasha
5        thx best time tonight story heartbreakingly au…
6                             still waiting supply liscus
…                                                      …
1867526                  im happy nowonder name show happy
1867527  every circumtance id like thankful almighty je…
1867528  there currently two girl walking around librar…
1867533  ah corporate life date using relative anachron…
1867534                          blessed living sundayvibes

[1455563 rows x 5 columns]
```

[ ]:
```
test_data
```

[ ]:
```
                                hashtags  tweet_id emotion identification  \
2                             [bibleverse]  0x28b412     NaN           test
4                                       []  0x2de201     NaN           test
9        [materialism, money, possessions]  0x218443     NaN           test
30                    [GodsPlan, GodsWork]  0x2939d5     NaN           test
33                                      []  0x26289a     NaN           test
…                                      …       …     …             …
1867525                                 []  0x2913b4     NaN           test
1867529                                 []  0x2a980e     NaN           test
1867530    [mixedfeeling, butimTHATperson]  0x316b80     NaN           test
1867531                                 []  0x29d0cb     NaN           test
1867532                                 []  0x2a6a4f     NaN           test

                                                      text
2        confident obedience write knowing even ask phi…
4        trust faith friend someone trust putting faith…
9        enough satisfied goal really money materialism…
30                    god woke chase day godsplan godswork
33                            tough time turn symbol hope
…                                                      …
1867525  message ye heard beginning love one another jo…
1867529  lad hath five barley loaf two small fish among…
1867530  buy last ticket remaining show sell mixedfeeli…
1867531                   swear hard work gone pay one day
```

```
1867532                    card left wasnt idea get parcel

[411972 rows x 5 columns]
```

[ ]:
```
train_data = train_data.drop('identification', axis=1)
test_data = test_data.drop('identification', axis=1)
```

[ ]:
```
X_data = train_data.drop('emotion', axis=1)
y_data = train_data['emotion']
X_valid = test_data.drop('emotion', axis=1)
```

[ ]:
```
y_submit = pd.DataFrame({
    'id': X_valid['tweet_id'],
    'emotion': test_data['emotion']
})

y_submit
```

[ ]:
```
                id emotion
2         0x28b412     NaN
4         0x2de201     NaN
9         0x218443     NaN
30        0x2939d5     NaN
33        0x26289a     NaN
…              …     …
1867525   0x2913b4     NaN
1867529   0x2a980e     NaN
1867530   0x316b80     NaN
1867531   0x29d0cb     NaN
1867532   0x2a6a4f     NaN

[411972 rows x 2 columns]
```

change labels to one-hot encoding

[ ]:
```
from sklearn.preprocessing import LabelEncoder
import keras
from keras.utils import to_categorical

label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y_data)
y_one_hot = to_categorical(y_encoded)
```

```
/Users/vvnliu/miniconda3/envs/tensorflow/lib/python3.10/site-
packages/h5py/__init__.py:36: UserWarning: h5py is running against HDF5 1.14.3
when it was built against 1.14.2, this may cause problems
  _warn(("h5py is running against HDF5 {0} when it was built against {1}, "
```

```
[ ]: from sklearn.model_selection import train_test_split

     # Assuming 'data' is your padded sequences and 'your_labels' are your labels
     X_train, X_test, y_train, y_test = train_test_split(X_data, y_one_hot,␣
      ↪test_size=0.2, random_state=42)
```

```
[ ]: from gensim.models import KeyedVectors

     # Load the model (this will take some time)
     google_w2v_model = KeyedVectors.
      ↪load_word2vec_format('GoogleNews-vectors-negative300.bin.gz', binary=True)
```

tokenize the word vecotors and make them into sequences based on the Google model

```
[ ]: from keras.preprocessing.text import Tokenizer
     from keras.preprocessing.sequence import pad_sequences

     tokenizer = Tokenizer()
     tokenizer.fit_on_texts(X_train['text'])
     sequences = tokenizer.texts_to_sequences(X_train['text'])
```

```
[ ]: tokenizer2 = Tokenizer()
     tokenizer2.fit_on_texts(X_train['text'])
     sequences2 = tokenizer2.texts_to_sequences(X_test['text'])
```

```
[ ]: tokenizer3 = Tokenizer()
     tokenizer3.fit_on_texts(X_valid['text'])
     sequences3 = tokenizer3.texts_to_sequences(X_valid['text'])
```

padded the sequences to have uniform length, the max length is set to 50

i've plot out the distribution of the length of the sequences and found the max of the training set was around 36. however, the training result did not perform well so i tried some other numbers with 50 obtaining the best result.

```
[ ]: # Pad sequences to ensure uniform length
     max_length = 50
     X_train_padded = pad_sequences(sequences, maxlen=max_length)
     X_test_padded = pad_sequences(sequences2, maxlen=max_length)
     X_valid_padded = pad_sequences(sequences3, maxlen=max_length)
```

```
[ ]: vocab_size = len(tokenizer.word_index) + 1
     embedding_matrix = np.zeros((vocab_size, 300))  # 300 is the dimensionality of␣
      ↪GoogleNews vectors

     for word, i in tokenizer.word_index.items():
         if word in google_w2v_model:
             embedding_matrix[i] = google_w2v_model[word]
```

[ ]:
```
from keras.models import Sequential
from keras.layers import Embedding, LSTM, Dense

model = Sequential()
model.add(Embedding(vocab_size, 300,
                    weights=[embedding_matrix],
                    input_length=max_length,
                    trainable=False))
model.add(LSTM(100))
model.add(Dense(8, activation='softmax'))

model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

```
2023-12-31 02:13:33.121171: I metal_plugin/src/device/metal_device.cc:1154]
Metal device set to: Apple M2
2023-12-31 02:13:33.121260: I metal_plugin/src/device/metal_device.cc:296]
systemMemory: 16.00 GB
2023-12-31 02:13:33.121273: I metal_plugin/src/device/metal_device.cc:313]
maxCacheSize: 5.33 GB
2023-12-31 02:13:33.121747: I
tensorflow/core/common_runtime/pluggable_device/pluggable_device_factory.cc:306]
Could not identify NUMA node of platform GPU ID 0, defaulting to 0. Your kernel
may not have been built with NUMA support.
2023-12-31 02:13:33.122275: I
tensorflow/core/common_runtime/pluggable_device/pluggable_device_factory.cc:272]
Created TensorFlow device (/job:localhost/replica:0/task:0/device:GPU:0 with 0
MB memory) -> physical PluggableDevice (device: 0, name: METAL, pci bus id:
<undefined>)
```

[ ]:
```
num_epochs = 10
batch_size = 25

model.fit(X_train_padded, y_train, epochs=num_epochs, batch_size=batch_size,␣
 ↪validation_split=0.1)
```

```
Epoch 1/10

2023-12-31 02:13:35.086088: I
tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:117]
Plugin optimizer for device_type GPU is enabled.

41921/41921 [==============================] - 667s 16ms/step - loss: 1.3717 -
accuracy: 0.5005 - val_loss: 1.3307 - val_accuracy: 0.5162
Epoch 2/10
41921/41921 [==============================] - 658s 16ms/step - loss: 1.3001 -
accuracy: 0.5282 - val_loss: 1.3145 - val_accuracy: 0.5237
Epoch 3/10
```

```
41921/41921 [==============================] - 660s 16ms/step - loss: 1.2725 -
accuracy: 0.5387 - val_loss: 1.3094 - val_accuracy: 0.5257
Epoch 4/10
41921/41921 [==============================] - 657s 16ms/step - loss: 1.2536 -
accuracy: 0.5454 - val_loss: 1.3092 - val_accuracy: 0.5265
Epoch 5/10
41921/41921 [==============================] - 658s 16ms/step - loss: 1.2389 -
accuracy: 0.5510 - val_loss: 1.3144 - val_accuracy: 0.5256
Epoch 6/10
41921/41921 [==============================] - 654s 16ms/step - loss: 1.2265 -
accuracy: 0.5554 - val_loss: 1.3166 - val_accuracy: 0.5248
Epoch 7/10
41921/41921 [==============================] - 644s 15ms/step - loss: 1.2161 -
accuracy: 0.5594 - val_loss: 1.3214 - val_accuracy: 0.5259
Epoch 8/10
41921/41921 [==============================] - 648s 15ms/step - loss: 1.2067 -
accuracy: 0.5625 - val_loss: 1.3232 - val_accuracy: 0.5241
Epoch 9/10
41921/41921 [==============================] - 646s 15ms/step - loss: 1.1984 -
accuracy: 0.5658 - val_loss: 1.3294 - val_accuracy: 0.5230
Epoch 10/10
41921/41921 [==============================] - 648s 15ms/step - loss: 1.1910 -
accuracy: 0.5683 - val_loss: 1.3375 - val_accuracy: 0.5217
```

[ ]:
```
<keras.src.callbacks.History at 0x61634a0b0>
```

[ ]:
```
loss, accuracy = model.evaluate(X_test_padded, y_test)
print(f'Test Accuracy: {accuracy*100:.2f}%')
```

```
9098/9098 [==============================] - 89s 10ms/step - loss: 1.3356 -
accuracy: 0.5217
Test Accuracy: 52.17%
```

[ ]:
```
predictions = model.predict(X_valid_padded)
```

```
12875/12875 [==============================] - 56s 4ms/step
```

[ ]:
```
predictions
```

[ ]:
```
array([[0.0313473 , 0.13781048, 0.14356543, …, 0.20847818, 0.00858407,
        0.0355196 ],
       [0.01138101, 0.04839285, 0.07681911, …, 0.2734959 , 0.08086084,
        0.04380331],
       [0.00664917, 0.1746202 , 0.0410641 , …, 0.10083697, 0.03518936,
        0.13484192],
       …,
       [0.00288988, 0.30807176, 0.01014552, …, 0.00520783, 0.01221734,
        0.40342933],
       [0.0077603 , 0.05795806, 0.01774925, …, 0.04405299, 0.01190932,
```

```
        0.1304694 ],
       [0.00503109, 0.33500794, 0.14155719, …, 0.11989159, 0.04174703,
        0.09989318]], dtype=float32)
```

[ ]:
```
class_indices = np.argmax(predictions, axis=1)
original_classes = label_encoder.inverse_transform(class_indices)
```

[ ]:
```
y_submit['emotion'] = original_classes
```

[ ]:
```
y_submit
```

[ ]:
```
                id       emotion
2         0x28b412           joy
4         0x2de201           joy
9         0x218443           joy
30        0x2939d5       sadness
33        0x26289a          fear
…              …             …
1867525   0x2913b4           joy
1867529   0x2a980e           joy
1867530   0x316b80         trust
1867531   0x29d0cb           joy
1867532   0x2a6a4f  anticipation

[411972 rows x 2 columns]
```

[ ]:
```
y_submit.to_csv('submission.csv', index=False)
```