```python
# Imports
import pickle
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
import matplotlib.image as mpimg
import seaborn as sns
import nltk
from nltk.corpus import stopwords
import string
import re
from nltk import word_tokenize, FreqDist
from sklearn.feature_extraction.text import CountVectorizer

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

from keras.models import load_model

##############################################
#  Data Cleaning and Visualization Functions
##############################################

# Cleans Titles for Games
def game_cleaner(title):
    cleaned_game = title.lower()
    cleaned_game = cleaned_game.replace("starcraft2co-op","
starcraft2")
    cleaned_game = cleaned_game.replace("starcraft2brutal+6","
starcraft2")
    cleaned_game = cleaned_game.replace("warcraft3reforged","
warcraft3")
    cleaned_game = cleaned_game.replace("lowkovszelda","zelda")
    cleaned_game = cleaned_game.replace("lowkoplays...starcraft","
starcraft")
    cleaned_game = cleaned_game.replace("dotaunderlords","underlords
")
    cleaned_game = cleaned_game.replace("worldofwarcraftrp","
worldofwarcraft")
    cleaned_game = cleaned_game.replace("lowkoplaysworldofwarcraft",
"worldofwarcraft")
    cleaned_game = cleaned_game.replace("lowkovsabzu","sabzu")
    cleaned_game = cleaned_game.replace("lowkovsaplaguetale","
aplaguetale")
```

```
    cleaned_game = cleaned_game.replace("lowkovssekiro","sekiro")
    cleaned_game = cleaned_game.replace("hearthstonebattlegrounds","
hearthstone")
    return cleaned_game

# Preprocessor to help tokenize data
def splitter(input_message):
    message = input_message
    container = []
    # These stop words don't add much to the analysis.  Kept other
common stop words
    special_stops = ['of','OF','is','the','THE','by','BY','it','in',
'on','and','but','being','an',
                     'for','to','they','any','from','then','some','
you','your','their','as','about',
                     'out','with','his','hers','he','she','at','go',
'be']

    # Identifies special numbers that we want to keep from the data
    # Usually involves computer parts (memory, GPU), numbers found
in video game titles, or matchups (1 vs 1, 2 vs 2)
    comp_numbers = [2077,32,64,128,256,512,1024,1080,3080,60,144,
2018,2019,2020,2021,2022,100,200,1,2,3,4]

    # specific player cleanup
    message = message.replace("dongraegu","drg")
    message = message.replace("dong rae gu",'drg')
    message = message.replace("rattata",'vanya')
    message = message.replace("liquid","")
    message = message.replace("dark templar","darktemplar")
    message = message.replace("dark shrine","darkshrine")

    # Obtains the first word before the column which is usually a
video game title
    splitted = message.split(":")
    index = 0
    if ":" in message:
        container.append(game_cleaner("".join(message.split(":")[0].
split())))
        index = 1

    # Starts overall split of the data
    for word in message.split():
        if (index == 1) | (":" in word):
            index = 2
            continue
        # Removes punctuations and special characters
        all_upper = word.strip("!?()#&*,")
        # Removes possessives
        all_upper = all_upper.replace("'s","").replace("'d","").
replace("'","")

        # Removes numbers not in our accepted numbers list
        if all_upper.isdigit():
            if (int(all_upper) in comp_numbers):
                pass
            else:
                continue



        # Keeping words that are fully capitalized, otherwise, set
word to lowercase
        if all_upper == all_upper.upper():
            container.append(all_upper)
        else:
            # Catches stop words
            if all_upper.lower() in special_stops:
                continue
            else:
                container.append(all_upper.lower())
    joined = " ".join(container)
    return joined


# Pulls out special words from predefined list
def get_from_list(message, special_list):
    container = []
    splitted = splitter(message)
    for word in splitted.split():
        if word.lower() in special_list:
            container.append(word.lower())
    return container

# Pulls out specific race or country info for players
def get_player_info(message,special_list,special_df, info=None):
    info_df = special_df.copy()
    info_df['tag'] = info_df['tag'].apply(lambda x: x.lower())
    container = []
    players = []
    splitted = splitter(message)
    if (info == 'race') | (info == 'country'):
        for word in splitted.split():
            if word.lower() in special_list:
                row = special_df.loc[info_df['tag'] == word.lower()]
                value = row[info].values[0]
                if word.lower() not in players:
                    container.append(value)
                    players.append(word.lower())
    # handles special cases
    if len(container)==3:
        container.pop(0)
    return container

# Extracts desired information from video stats
# Had to remove dislike count as they were removed from YouTube
recently
def get_stats(video):
    vid_id = video['id']
    title = video['snippet']['title']
    date = video['snippet']['publishedAt']
    duration = video['contentDetails']['duration'] #example
PT1H1M31S
    thumbnail = video['snippet']['thumbnails']['medium']['url']
    tags = video['snippet']['tags']
    views = video['statistics']['viewCount']
    likes = video['statistics']['likeCount']
    return {'vid_id':vid_id,'title':title,'date':date,'duration':
duration,
            'thumbnail':thumbnail,'tags':tags,'views':views,'likes':
likes}

# Convert duration to minutes
def dur_to_min(time):

    # Using regex to parse time
    nph = '(?=\d{1,2}H)(\d{1,2})'
    npm = '(?=\d{1,2}M)(\d{1,2})'
    nps = '(?=\d{1,2}S)(\d{1,2})'
    ph = re.compile(nph)
    pm = re.compile(npm)
    ps = re.compile(nps)
    h = ph.findall(time)
    m = pm.findall(time)
    s = ps.findall(time)

    # Converts to float
    if h:
        h = float(h[0])
    else:
        h = 0.0
    if m:
        m = float(m[0])
    else:
        m = 0.0
    if s:
        s = float(s[0])
    else:
        s = 0.0

    # Calculates total minutes
    minutes = (h*60)+m+(s/60.0)
    return round(minutes,2)


##############################################
#  Function for Dashboard and User Input
##############################################

# Returns DataFrame of the user's input
def preprocessor(user_input):
    grammed_df = pickle.load(open(r"Data\grammed_df.pickle","rb"))
    user_input_split = splitter(user_input).split()

    # Containers
    avg_word_views = []
    count_word_views = []
    not_found = []

    # Find average views and number of videos, 0 if none
    for word in user_input_split:
        try:
            avg_word_views.append(round(grammed_df.loc[grammed_df[
word]==1].views.mean()))
            count_word_views.append(grammed_df.loc[grammed_df[word
]==1].views.count())
        except:
            avg_word_views.append(0)
            count_word_views.append(0)
            not_found.append(word)

    # Creating Dataframe for the words with their average views and
number of videos
    words_df = pd.DataFrame(data=[user_input_split])
    words_df = words_df.T
    words_df['avg_views'] = avg_word_views
    words_df['num_videos'] = count_word_views
    words_df.columns = ['words','avg_views','num_videos']
    return words_df, not_found

# Plots a horizontal bar plot of user's input
def visualizer(user_input):
    words_df, not_found = preprocessor(user_input)
    words_df.sort_values(by='avg_views',ascending=False,inplace=True
)
    fig,ax = plt.subplots(figsize=(6,6))
    sns.barplot(y=words_df['words'],x=words_df['avg_views'],palette=
'mako',ax=ax)
    ax.set_title('Average Views By Word')
    ax.set_xlabel('Average Views')
    ax.set_ylabel('Word')
    return fig

# Returns the predicted number of views
def view_predict(user_input):
    tokenizer = pickle.load(open(r'Data\tfidf_fit.pickle','rb'))
    model = load_model(r'Data\model_mae_iqr.h5')
    transformed = tokenizer.transform([user_input])
    prediction = model.predict(transformed.todense())
    return round(prediction[0][0])
```