cleanPosts() Function:

Args:

- Ensure that all input data for the "posts" argument as a single column.

Functionality:

- Disposes of all non-post columns/features of the data (i.e. author, url, score, num_comments, etc.)
- Disposed of all empty rows/posts.
- Disposed of rows that contained [deleted] and [removed] posts.
- All posts are truncated to 512 words (delimited by a space).

Returns:

- Returns a Pandas DataFrame containing the cleaned dataset
- Also returns the number of posts that needed truncation

In [1]:
```
def cleanPosts(posts, newColumnLabel:str):
    posts = posts[posts != '[deleted]'] #Filter out rows where the value is [deleted]
    posts = posts[posts != '[removed]'] #Filter out rows where the value is [removed]
    posts = posts.dropna() #Remove rows where the post is empty

    originalLengths = posts.apply(lambda x: len(x.split())) #A list of the original length of each post.
    truncatedCount = sum(originalLengths > 512) #Calculate the number of posts that will be truncated to 512 wo

    posts = posts.apply(lambda x: ' '.join(x.split()[:512])) #Truncate number of words in the post to 512
    posts = posts.reset_index(drop=True) #Reset the indices by dropping all removed rows from the list

    return pd.DataFrame({newColumnLabel: posts}), truncatedCount
```

Imports

In [2]:
```
import pandas as pd
import os
```

Positive Class CSV Creation

- Outside of the code, converted the original .xlsx source Excel spreadsheet to a utf-8 encoded CSV file to remove right quotation symbols (â€™) from the text.
- 985 of the 13697 PTSD-positive posts were truncated to 512 words (delimited by a space) during this data cleaning. This value is contained in posTruncatedCount.

In [3]:
```
# Read the original CSV file
df = pd.read_csv('raw_datasets/positive/rptsd.csv', delimiter=',')

posts = df.iloc[:, 1] #Only need to work with the posts themselves. Dispose of all other author, url, and other
positive_df, posTruncatedCount = cleanPosts(posts, 'posts')

#Create a new CSV file with the newly organized data
positive_df.to_csv('positive_cleaned.csv', index=False)  #Set index=False to exclude row numbers in the output

numPositive = len(positive_df)
```

Negative Class CSV Creation

- This class was built using a combination of non-PTSD related subreddits (r/fitness, r/teaching, etc.)
- Each subreddit has its own CSV data file.
- The number of positive instances is calculated above, and this value is used to determine how many negative instances should be taken from each subreddit (so that there is roughly equal representation - both between the positive and negative classes as well as equal representation between the various negative subreddits).
- generateFullNegativeClassDatasets should be set to False for the actual research. Setting this to True will simply generate cleaned CSVs for the entire dataset, rather than a random sample.
- The same cleaning procedure was performed on each of these subreddits.
- 676 of the total 13696 negative posts were truncated to 512 words (delimited by a space) during this data cleaning. This value is contained in negTruncatedCount.

In [4]:
```
generateFullNegativeClassDatasets = False #If you set this to True, it will generate a massive cleaned dataset (
negTruncatedCount = 0 #Sums the overall number of negative instances that needed to be truncated across all data

directory = 'raw_datasets/negative' #Directory containing negative CSV files

dfs = [] #List to store DataFrames from each CSV file

listOfNegativeCSVFiles = [f for f in os.listdir(directory) if f.endswith('.csv')] #List of all CSV files
numNegativeCSVFiles = len(listOfNegativeCSVFiles)
numNegPerClass = round(numPositive / numNegativeCSVFiles) #Number of instances to sample from each negative CSV

#Iterate through each CSV file in the directory
for filename in listOfNegativeCSVFiles:
    filepath = os.path.join(directory, filename)

    df = pd.read_csv(filepath, delimiter=',') #Read from the original CSV file and create a DataFrame
    if generateFullNegativeClassDatasets == False:
        df = df.sample(n=numNegPerClass, random_state=42) #Sample only the same number of observations from eac

    posts = df.iloc[:, 3] #Only need to work with the posts themselves. Dispose of all other data columns.
    negative_df, partialNegTruncatedCount = cleanPosts(posts, 'posts')
    negTruncatedCount += partialNegTruncatedCount #Sum up the total number of negative posts that needed trunca

    #Create a new CSV file with the newly organized data
    if generateFullNegativeClassDatasets == True:
        negative_df.to_csv('full_negative_cleaned/' + filename, index=False)
    else:
        negative_df.to_csv('partial_negative_cleaned/' + filename, index=False)

    dfs.append(negative_df) #Append the DataFrame to the list for the total dataset concatenation later.

#Concatenate the DataFrames into one DataFrame
negative_df = pd.concat(dfs, ignore_index=True)

negative_df.reset_index(drop=True, inplace=True) #Reset the index

#Save the resulting DataFrame to a CSV file
negative_df.to_csv('negative_cleaned.csv', index=False)

numNegative = len(negative_df)
```

Class representation is approximately equal.

In [5]:
```
numPositive, numNegative
```

Out[5]:
```
(13697, 13696)
```