

## CS 8920 – Independent Study Project Report

K M Tawsik Jawad

U01034189

**Domain:** Finding out commonly used word sequences in the reddit posts of patients suffering from ADHD. The 2<sup>nd</sup> dataset is collected from the channels like r/anxiety, r/depression etc. from reddit posts. Same technique in patients suffering from different mental health issues in the 2<sup>nd</sup> dataset was applied to find out the most common word sequences in both of them. Performing word tokenization and padding, one hot encoding to feed into LSTM to predict next words from commonly occurring sequences of words.

**Data Pre-Processing:** Since only the posts are required for analysis, other columns of the dataset were dropped along with null columns. Then necessary pre-processing was performed as follows:

```
In [1]: import pandas as pd

newDf = pd.read_csv('ADHD.csv')

newDf.head()
```

C:\Users\Asus\Anaconda3\lib\site-packages\pandas\compat\\_optional.py:138: UserWarning: Pandas requires version '2.7.0' or of 'numexpr' (version '2.6.8' currently installed).  
warnings.warn(msg, UserWarning)  
C:\Users\Asus\Anaconda3\lib\site-packages\IPython\core\interactiveshell.py:3267: DtypeWarning: Columns (2,6) have mixed tpecify dtype option on import or set low\_memory=False.  
exec(code\_obj, self.user\_global\_ns, self.user\_ns)

Out[1]:

	title	selftext	score	id	url	num_comments	created_utc	created_
0	Android app to strengthen attention/focus	Hey /r/ADHD,\n\nCheck out my simple Android ap...	6	k348a	https://www.reddit.com/r/ADHD/comments/k348a/a...	7.0	1315014480.0	20
1	Does anyone here have experience with Imipramine?	My doctor has suggested it for ADD with anxiet...	4	k3gdz	https://www.reddit.com/r/ADHD/comments/k3gdz/d...	1.0	1315051084.0	20
2	New study shows that for people with ADHD, \n"...		NaN	k4q79	http://www.utexas.edu/features/2011/08/29/adhd...	11.0	1315174374.0	20
3	What does the ADHD test look like?	I'm 21 and didn't know about ADHD until recent...	6	k5fvd	https://www.reddit.com/r/ADHD/comments/k5fvd/w...	10.0	1315239683.0	20
4	Are you guys good with maps and directions?	It seems like one thing I've always been reall...	0	k6efi	https://www.reddit.com/r/ADHD/comments/k6efi/a...	0.0	1315317191.0	20

```
In [3]: newDf = newDf.drop(['selftext', 'url', 'num_comments', 'created_utc', 'score', 'id', 'created_datetime'], axis='columns')
newDf
```

Out[3]:

	title
4815	My doctor lost his licence a few days from nex...
78504	Besides not being neurotypical, is anyone else...
56091	Casinos, yes or no?
77259	Anybody here has had success with the Getting ...
332202	frustration
...	...
296548	Hardcore Avoidance Recently- anyone else?
38891	ADD and coffee?
73617	Drugs made my life much better than i thought ...
125490	Ritalin euphoria?
34990	I know this is ridiculous but can someone plea...

6721 rows × 1 columns

Next, the unnecessary urls are removed from the corpus. The entire corpus is converted to lowercase and removal of extra white spaces was done for easier analysis.

```
In [128]: newDf['title'] = newDf['title'].str.lower()
newDf['title']
```

Out[128]:

43598	when you feel a new hyper fixation coming... (...
11093	i can't seem to keep friends for very long and...
10600	trouble processing auditory
11294	what's your favorite adhd podcast?
8913	energy? what's that?
...	...
23356	constant cold hands and feet?
7450	my angel dog
33704	jumbly brain and expressing ideas/thoughts
4599	how i am gamifying my life to increase product...
29538	2:34am and i've just finished this! i meant to...

Name: title, Length: 888, dtype: object

```
In [129]: def remove_punctuation(text):
no_punctuation = "".join([word for word in text if word not in PUNCTUATION])
return no_punctuation

newDf['title'] = newDf['title'].apply(remove_punctuation)

newDf['title'].head()
```

Out[129]:

43598	when you feel a new hyper fixation coming post...
11093	i can't seem to keep friends for very long and...
10600	trouble processing auditory
11294	what's your favorite adhd podcast
8913	energy what's that

Name: title, dtype: object

```
In [130]: def remove_extra_white_spaces(text):
single_char_pattern = r'\s+[a-zA-Z]\s+'
without_sc = re.sub(pattern=single_char_pattern, repl=" ", string = text)
return without_sc

newDf['title'] = newDf['title'].apply(remove_extra_white_spaces)
```

Next, we performed word lemmatization which basically reduces a word to its root form. The reason for choosing lemmatization in stead of stemming is to keep a better root mapping of the word. If the word extracted by stemming from 'better' is 'bet', for lemmatization it would be 'good'.

```
In [133]: from nltk.stem import WordNetLemmatizer
def lemmatize_text(text):
    lemmatizer = WordNetLemmatizer()
    tokens = word_tokenize(text)
    for i in range(len(tokens)):
        lemma_word = lemmatizer.lemmatize(tokens[i])
        tokens[i] = lemma_word
    return " ".join(tokens)

newDf['title'] = newDf['title'].apply(lemmatize_text)
newDf['title'].head()

Out[133]: 43598          feel new hyper fixation coming post
          11093      ' seem keep friend long ' thing make different...
          10600          trouble processing auditory
          11294          ' favorite adhd podcast
          8913          energy '
Name: title, dtype: object
```

Moving on to the last part of the pre-processing is the generation of N-Grams. Experiments were performed with unigrams, bigrams, trigrams. Here the most commonly used bigrams in the entire corpus is shown:

```
def generate_N_grams(text, ngram=1):
    text = re.sub(r'[^a-zA-Z0-9\s]', ' ', text)
    tokens = [token for token in text.split(" ") if token != ""]
    output = list(ngrams(tokens, ngram))
    return output

In [21]: lst = []
for X in newDf['title']:
    y = generate_N_grams(X, 2)
    lst.append(y)

def flatten(input):
    new_list = []
    for i in input:
        for j in i:
            new_list.append(j)
    return new_list

lst1 = flatten(lst)

finalList1 = []

from collections import Counter
c = Counter(lst1)
M = c.most_common(30)

for key, val in M:
    finalList1.append(key)

print(finalList1)

[('anyone', 'else'), ('feel', 'like'), ('diagnosed', 'adhd'), ('adhd', 'med'), ('adderall', 'xr'), ('adhd', 'medication'), ('rst', 'time'), ('people', 'adhd'), ('need', 'advice'), ('need', 'help'), ('adult', 'adhd'), ('side', 'effect'), ('dont', 'krw'), ('got', 'diagnosed'), ('think', 'adhd'), ('recently', 'diagnosed'), ('year', 'old'), ('anybody', 'else'), ('taking', 'adderall'), ('adhd', 'symptom'), ('like', 'im'), ('please', 'help'), ('someone', 'adhd'), ('take', 'med'), ('long', 'term'), ('ing', 'diagnosed'), ('finally', 'got'), ('first', 'day'), ('adhd', 'diagnosis'), ('else', 'feel')]
```

Next, we perform the similar pre-processing steps of data cleaning in the 2<sup>nd</sup> dataset of mental health issues and generate N-Grams to find out what common N-Grams occur in the 2<sup>nd</sup> dataset.

```
In [15]: import pandas as pd

df = pd.read_csv('anxiety_2018_features_tfidf_256.csv')

df.head()
```

Out[15]:

	subreddit	author	date	post	automated_readability_index	coleman_liaw_index	flesch_kincaid_grade_level	flesch_reading_ease	gulepease
0	anxiety	Watch_Me_Get_	2018/01/01	Does anyone else like taking long walks while ...	5.007692	8.594298	3.283462	83.099038	85.9
1	anxiety	Imherefor games	2018/01/01	Meditation is making me anxious I do a couple ...	-0.716269	1.456650	1.640910	95.231515	96.9
2	anxiety	ABrokenBeing	2018/01/01	Rant about anxiety meds - I've been self medic...	5.952747	7.543635	6.078791	76.416209	87.1
3	anxiety	Sleekdiamond41	2018/01/01	Some advice for my friend please? I dated this...	6.582939	6.555885	6.839898	77.193316	85.2
4	anxiety	coffeegreentea	2018/01/01	Feeling like I'm back to where I started.	5.459158	6.349314	6.922802	72.442031	87.7

```
df['post'] = df['post'].apply(remove_punctuation)
df['post'] = df['post'].apply(remove_extra_white_spaces)
df['post'] = df['post'].apply(remove_stopwords)
df['post'] = df['post'].apply(drop_duplicates)
df['post'] = df['post'].apply(lemmatize_text)

lst = []
for x in df['post']:
    y = generate_N_grams(X,2)
    lst.append(y)

def flatten(input):
    new_list = []
    for i in input:
        for j in i:
            new_list.append(j)
    return new_list

lst1 = flatten(lst)

finalList2 = []

from collections import Counter
c = Counter(lst1)
M = c.most_common(30)

for key,val in M:
    finalList2.append(key)

print(finalList2)

[('feel', 'like'), ('panic', 'attack'), ('dont', 'know'), ('anyone', 'else'), ('dont', 'want'), ('like', 'im'), ('im', 'goin
g'), ('anxiety', 'attack'), ('even', 'though'), ('social', 'anxiety'), ('first', 'time'), ('year', 'ago'), ('make', 'feel'),
('year', 'old'), ('felt', 'like'), ('every', 'time'), ('feeling', 'like'), ('gon', 'na'), ('im', 'sure'), ('anxiety', 'depressi
on'), ('really', 'had'), ('im', 'scared'), ('every', 'day'), ('month', 'ago'), ('side', 'effect'), ('high', 'school'), ('last',
```

Finally, we use the set operator in the two lists of that contain the most common sequences to find out exactly which word sequences are there in each of the dataset:

```
In [25]: Y = set(finalList2) & set(finalList1)
print(Y)

{('like', 'im'), ('dont', 'know'), ('side', 'effect'), ('feel', 'like'), ('year', 'old'), ('first', 'time'), ('anyone', 'else')}
```

Next, we analyze the n-gram sequences using trigrams. First lets see what sequences are most common using trigrams in the ADHD dataset:

```
In [26]: lst = []
for x in newDf['title']:
    y = generate_N_grams(x,3)
    lst.append(y)

def flatten(input):
    new_list = []
    for i in input:
        for j in i:
            new_list.append(j)
    return new_list

lst1 = flatten(lst)

finalList1 = []

from collections import Counter
c = Counter(lst1)
M = c.most_common(30)

for key,val in M:
    finalList1.append(key)

print(finalList1)

[('feel', 'like', 'im'), ('anyone', 'else', 'feel'), ('ever', 'feel', 'like'), ('anyone', 'else', 'experience'), ('recently', 'diagnosed', 'adhd'), ('else', 'feel', 'like'), ('anyone', 'else', 'find'), ('got', 'diagnosed', 'adhd'), ('rejection', 'sensitive', 'dysphoria'), ('anyone', 'else', 'issue'), ('anyone', 'else', 'struggle'), ('kick', 'start', 'sunday'), ('anyone', 'else', 'problem'), ('anyone', 'else', 'get'), ('med', 'first', 'time'), ('make', 'feel', 'like'), ('think', 'might', 'adhd'), ('anyone', 'else', 'trouble'), ('anyone', 'else', 'hate'), ('anyone', 'else', 'really'), ('finally', 'got', 'diagnosed'), ('get', 'stuff', 'done'), ('short', 'term', 'memory'), ('started', 'taking', 'adderall'), ('could', 'use', 'advice'), ('feel', 'like', 'failure'), ('diagnosed', 'adult', 'adhd'), ('get', 'thing', 'done'), ('taking', 'adhd', 'medication'), ('might', 'adhd', 'im')]
```

Next, we do the similar approach in the 2<sup>nd</sup> dataset of mental health:

```

for x in df['post']:
    y = generate_N_grams(X,3)
    lst.append(y)

def flatten(input):
    new_list = []
    for i in input:
        for j in i:
            new_list.append(j)
    return new_list

lst1 = flatten(lst)

finalList2 = []

from collections import Counter
c = Counter(lst1)
M = c.most_common(30)

for key,val in M:
    finalList2.append(key)
|
print(finalList2)

[('feel', 'like', 'im'), ('anyone', 'else', 'experience'), ('anyone', 'else', 'get'), ('anyone', 'else', 'feel'), ('make', 'feel', 'like'), ('anxiety', 'panic', 'attack'), ('im', 'gon', 'na'), ('like', 'im', 'going'), ('long', 'story', 'short'), ('dont', 'know', 'im'), ('first', 'panic', 'attack'), ('dont', 'even', 'know'), ('feel', 'like', 'cant'), ('feel', 'like', 'going'), ('anxiety', 'feel', 'like'), ('dont', 'really', 'know'), ('feel', 'like', 'anxiety'), ('feel', 'like', 'ive'), ('always', 'feel', 'like'), ('feel', 'like', 'need'), ('generalized', 'anxiety', 'disorder'), ('cant', 'stop', 'thinking'), ('get', 'panic', 'attack'), ('every', 'single', 'day'), ('anyone', 'else', 'experienced'), ('feel', 'like', 'everyone'), ('first', 'time', 'postin g'), ('even', 'though', 'know'), ('feel', 'like', 'dont'), ('feeling', 'like', 'im')]

```

And find out the common trigrams in the two datasets in the following:

```

In [28]: Y = set(finalList2) & set(finalList1)
print(Y)

{('make', 'feel', 'like'), ('anyone', 'else', 'feel'), ('anyone', 'else', 'get'), ('feel', 'like', 'im'), ('anyone', 'else', 'experience')}

```

Let's analyze the common sequences in bigrams and trigrams putting the results together in the following two snapshots. First one is the bigram and second is trigram:

```

In [25]: Y = set(finalList2) & set(finalList1)
print(Y)

{('like', 'im'), ('dont', 'know'), ('side', 'effect'), ('feel', 'like'), ('year', 'old'), ('first', 'time'), ('anyone', 'else')}

```

```

In [28]: Y = set(finalList2) & set(finalList1)
print(Y)

{('make', 'feel', 'like'), ('anyone', 'else', 'feel'), ('anyone', 'else', 'get'), ('feel', 'like', 'im'), ('anyone', 'else', 'experience')}

```

Although the bigram produces a greater number of common sequences than trigrams, the bigrams don't really put too much meaning in the sequences. If we look at trigrams, they are shown to provide more meaning with more words. So, intuition about the next words in the sentence can be better developed using tri-grams rather than bigrams for this case.

Next, women suffering from ADHD also put up posts in reddit. Analyzing this “adhdwomen.csv” dataset after performing the similar pre-processing as the previous datasets, they were prepared to be used for a word prediction task. Here it can be seen in the picture below how word tokenization is performed on the dataset to assign an id to each word in the corpus.

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

newDf = pd.read_csv('adhdwomen.csv')

newDf.head()
```

Out[2]:

	title	selftext	score	id	url	num_comments	created_utc	created_datetime
0	Adult Women Are the New Face of ADHD	NaN	3	29ka8	http://www.thedailybeast.com/articles/2014/06/...	0	1.404224e+09	2014-07-01 14:07:46
1	Why Women Hide Their ADHD Symptoms	NaN	3	2ip2ra	https://euromd.com/9-diseases-and-conditions/1...	0	1.412801e+09	2014-10-08 20:48:14
2	Adult ADHD and Burnout: Success or Failure?	NaN	1	2q6jdk	http://rethinkadhd.wordpress.com/2014/12/23/ad...	0	1.419349e+09	2014-12-23 15:34:03
3	How Am I And My ADHD Still Alive?	NaN	2	2sc7fa	http://blogs.psychcentral.com/adhd-man/2015/01...	0	1.421192e+09	2015-01-13 23:35:23
4	I'd like to see this subreddit grow!	Hello, I'm a working, married, mother of 3. [...]	1	3296xx	https://www.reddit.com/r/adhdwomen/comments/32...	1	1.428778e+09	2015-04-11 18:43:49

```
In [29]: import pandas as pd
import os
import numpy as np

import tensorflow as tf
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.layers import Embedding, LSTM, Dense, Bidirectional
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import Adam

tokenizer = Tokenizer(oov_token='<ooov>')
tokenizer.fit_on_texts(newDf['title'])
total_words = len(tokenizer.word_index) + 1

print("Total number of words: ", total_words)
print("Word: ID")
print("adhd: ", tokenizer.word_index['adhd'])
print("anybody: ", tokenizer.word_index['anybody'])

Total number of words: 1942
Word: ID
adhd: 2
anybody: 158
```

Next, I convert the texts in the corpus into number sequences. Then, normalize the dataset by taking the max length sentence and padding the rest of the sentences with the exact same length in the following way:

```
In [16]: input_sequences = []
for line in newOf['title']:
    token_list = tokenizer.texts_to_sequences([line])[0]
    #print(token_list)

    for i in range(1, len(token_list)):
        n_gram_sequence = token_list[:i+1]
        input_sequences.append(n_gram_sequence)

# print(input_sequences)
print("Total input sequences: ", len(input_sequences))
print(input_sequences[1])

Total input sequences: 3999
[[88, 61]]
```

```
In [17]: max_sequence_len = max([len(x) for x in input_sequences])
input_sequences = np.array(pad_sequences(input_sequences, maxlen=max_sequence_len, padding='pre'))
input_sequences[1]
```

```
Out[17]: array([ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  
                 0,  0,  0,  0,  0,  0,  0,  0,  0, 88, 61])
```

Next, we split the dependent and independent variables. Since we are formulating a word prediction problem, the final word id according to the input sequences is taken as the dependent variable and the ids before that are the independent variables. Analyzing the x and y it can be clear:

```
In [18]: xs = input_sequences[:, :-1]
print(xs[1])
```

```
[ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0 88]
```

```
In [19]: labels = input_sequences[:, -1]
          print(labels[1])
```

61

```
In [20]: ys = tf.keras.utils.to_categorical(labels, num_classes=total_words)
```

Next, I perform one hot encoding on the labels variable to prepare the dependent set to be fed to the LSTM network. Since I wanted to perform the analysis on a random sample of the dataset and LSTM training is a convoluted network, iteration number was kept to 20 where in each step loss value and accuracy metric is seen. For updating weight networks, adam optimizer was used for its good performance in deep learning NLP tasks.



```
In [21]: model = Sequential()
model.add(Embedding(total_words, 100, input_length=max_sequence_len-1))
model.add(Bidirectional(LSTM(150)))
model.add(Dense(total_words, activation='softmax'))
adam = Adam(lr=0.01)
model.compile(loss='categorical_crossentropy', optimizer=adam, metrics=['accuracy'])
history = model.fit(xs, ys, epochs=20, verbose=1)
#print model.summary()
print(model)
```

WARNING:tensorflow:From C:\Users\Asus\Anaconda3\lib\site-packages\tensorflow\python\ops\resource\_variable\_ops.py:435: colocate\_with (from tensorflow.python.framework.ops) is deprecated and will be removed in a future version.  
Instructions for updating:  
Colocations handled automatically by placer.  
WARNING:tensorflow:From C:\Users\Asus\Anaconda3\lib\site-packages\tensorflow\python\ops\math\_ops.py:3066: to\_int32 (from tensorflow.python.ops.math\_ops) is deprecated and will be removed in a future version.  
Instructions for updating:  
Use tf.cast instead.

Epoch 1/20  
3999/3999 [=====] - 11s 3ms/sample - loss: 7.5079 - acc: 0.0240  
Epoch 2/20  
3999/3999 [=====] - 9s 2ms/sample - loss: 6.7427 - acc: 0.0413  
Epoch 3/20  
3999/3999 [=====] - 10s 2ms/sample - loss: 5.9676 - acc: 0.0618  
Epoch 4/20  
3999/3999 [=====] - 10s 2ms/sample - loss: 4.6403 - acc: 0.1398  
Epoch 5/20  
3999/3999 [=====] - 11s 3ms/sample - loss: 3.2007 - acc: 0.3273  
Epoch 6/20  
3999/3999 [=====] - 11s 3ms/sample - loss: 1.8161 - acc: 0.5934  
Epoch 7/20  
3999/3999 [=====] - 10s 3ms/sample - loss: 1.0191 - acc: 0.7704  
Epoch 8/20  
3999/3999 [=====] - 9s 2ms/sample - loss: 0.6806 - acc: 0.8510  
Epoch 9/20  
3999/3999 [=====] - 9s 2ms/sample - loss: 0.5389 - acc: 0.8775  
Epoch 10/20

Finally, I performed the text prediction with predicting next 20 words after training the sequences with LSTM. For the input, I used one of the common trigrams “anyone else experience” which was found in both the trigrams of ADHD dataset and the mental health dataset. The result is as follows:

```
In [34]: seed_text = "anyone else experience"
next_words = 20

for _ in range(next_words):
    token_list = tokenizer.texts_to_sequences([seed_text])[0]
    token_list = pad_sequences([token_list], maxlen=max_sequence_len-1, padding='pre')
    predicted = model.predict_classes(token_list, verbose=0)
    output_word = ""
    for word, index in tokenizer.word_index.items():
        if index == predicted:
            output_word = word
            break
    seed_text += " " + output_word
print(seed_text)
```

anyone else experience overwhelm avalanche feel defeated general like suicided would share something helpful im overwhelmed try  
ing organize prioritize figure daily task night

```
In [35]: seed_text = "feel like im"
next_words = 20

for _ in range(next_words):
    token_list = tokenizer.texts_to_sequences([seed_text])[0]
    token_list = pad_sequences([token_list], maxlen=max_sequence_len-1, padding='pre')
    predicted = model.predict_classes(token_list, verbose=0)
    output_word = ""
    for word, index in tokenizer.word_index.items():
        if index == predicted:
            output_word = word
            break
    seed_text += " " + output_word
print(seed_text)
```

feel like im really productive day better failure planner planner love swear erin condren seems month start missing deadline re  
latable read sentence immediately

```
In [36]: seed_text = "medication"
next_words = 20

for _ in range(next_words):
    token_list = tokenizer.texts_to_sequences([seed_text])[0]
    token_list = pad_sequences([token_list], maxlen=max_sequence_len-1, padding='pre')
    predicted = model.predict_classes(token_list, verbose=0)
    output_word = ""
    for word, index in tokenizer.word_index.items():
        if index == predicted:
            output_word = word
            break
    seed_text += " " + output_word
print(seed_text)

medication question x adderall xr adderall xr surrounding adhd thanks say say care deep hurt thought night able able relate lol
```

**Conclusion:** So, analyzing the output sentences it can be inferred that the women suffering from ADHD have a hard time adjusting to the daily functions and many of them have higher degree of depressive symptoms. It can also be inferred that many of them are trying to organize their day-to-day activities, to focus more on their priorities and make a productive outcome keeping the ADHD symptoms aside. As usual with any kind of sickness, they are hoping to get help from people suffering from similar symptoms. Lastly, it can be observed that Adderall is used commonly when medication is provided as input where some common side effects of the medicine is seen.