In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import tools # self-defined functions
import warnings,sys
if not sys.warnoptions:
    warnings.simplefilter("ignore")
```

In [3]:
```
# Load data
load_file = '../datasets/reddit_submissions.json'
someposts = pd.read_json(load_file , lines=True)
someposts.index = someposts['id']
someposts.head()
```

Out[3]:

	author	created_utc	id	num_comments	selftext	subreddit	subreddit_nam
id							
76r64	Crito	1223866465	76r64	0		ptsd	
7goht	Crito	1228150531	7goht	0		ptsd	
7guki	socialcelebs	1228219003	7guki	0		mentalhealth	r/m
7gxll	[deleted]	1228244460	7gxll	1		EatingDisorders	r/Eatir
7gxm3	[deleted]	1228244538	7gxm3	1		EatingDisorders	r/Eatir

Preprocesses

record_process

Preprocesses the raw data, with the following guidelines:

- Exclude rows from the training set where `selftext` is a blank string, or has the values of either `"[removed]"` or `"[deleted]"`.
- Exclude rows with less than `5` comments.
- Only use the `title` and `selftext` fields as a source of features.
- Make a decision on how to handle subreddit categories with fewer than 1000 examples that simply merge them into /**one category**/, because the sum amount of rare categories is about 1142, a small amount.

In [4]:
```
subreddit_mappings, someposts = tools.record_process(someposts)
someposts.head()
```

```
There are 236742 records after processing
The sum of rare categories is 1142
```

Out[4]:

	title	selftext	target
id			
96l9t	Coping with panic/anxiety attacks. You tips?	Following on from the Onion article, and some ...	6
96zvm	Nothing much to keep me going (reintroduced)	About 4 months ago I posted something with bas...	6
972xv	Would it be a good idea to pool our resources ...	Hi all,\n\nI had an idea last night and I was ...	6
976on	This is my declaration of Interdependence	I am a fraud. I have spent a lifetime distanci...	6
977ls	I'm considering submitting myself to a psychia...	I won't go too into the details. Suffice to sa...	8

In [5]:
```
print('the integer to subreddit:')
subreddit_mappings
```

Out[5]:
```
{0: 'Anger',
 1: 'BPD',
 2: 'BipolarReddit',
 3: 'EatingDisorders',
 4: 'MMFB',
 5: 'StopSelfHarm',
 6: 'SuicideWatch',
 7: 'alcoholism',
 8: 'depression',
 9: 'dpdr',
 10: 'getting_over_it',
 11: 'mentalhealth',
 12: 'others',
 13: 'ptsd',
 14: 'rapecounseling',
 15: 'schizophrenia',
 16: 'socialanxiety'}
```

In [23]:
```
someposts['target'].value_counts()
```

Out[23]:
```
8     91554
6     54735
1     16850
2     14157
16    12618
15    10242
11     7554
4      6569
7      4931
13     4391
14     2911
9      2654
5      2044
10     1985
3      1218
0      1187
12     1142
Name: target, dtype: int64
```

text_process

- I process at the word level.
- I not only remove **punctuation**, but also do **stemming**. The rate of amount of unique word with/without stemming = 0.6, which may effect on results.
- I do not remove stop words in order to capture contextual features, but we can look back in the later iteration to remove the stop words since the classification is topic-based and I want to save time in the training by dealing with less words.
- I combine the title info with the selftext content by adding **End** tokens in text process in the future step

In [6]:
```
vocabulary_size = 5000
index_to_word, data  = tools.textprocess(someposts[['title','selftext']].
```

```
Found 82477 unique words tokens.
Using vocabulary size 5000.
The least frequent word in our vocabulary is 'alzheim' and appeared 18
1 times.

Example sentence: Following on from the Onion article, and some sugges
tions that a discussion would be good, can anyone share their tips for
dealing with this?

Example sentence after processing: ['follow', 'on', 'from', 'the', 'UN
KNOWN_TOKEN', 'articl', 'and', 'some', 'suggest', 'that', 'a', 'discus
s', 'would', 'be', 'good', 'can', 'anyon', 'share', 'their', 'tip', 'f
or', 'deal', 'with', 'thi']

Example input sentence: [618, 28, 70, 3, 4999, 1564, 2, 87, 665, 9, 4,
928, 69, 20, 115, 29, 116, 451, 196, 1048, 15, 267, 19, 18]
```

In [15]:
```
with open('ModelTraining/index_to_word.csv','w') as f:
    f.write(str(index_to_word))
```

partition_dataset

Partitions the model-ready data into train, validation, and test sets. Since we have 240k records(a relative large set), I picks train/validation/test ratio **80%, 10%, 10%**. My training process will not use test set for unleaky info.

In [16]:
```
# partitions the model-ready data into train, validation, and test sets.
print('There are {} records after processing'.format(len(someposts)))
X_train, X_test, X_val, y_train, y_test, y_val = tools.partition_dataset(
print('There are {},{},{} records for train, validation, and test sets'.f
```

```
There are 236742 records after processing
There are 189393,23675,23674 records for train, validation, and test s
ets
```

Model

Possible models:

- Bag of Words/Bigrams + LR/SVM
- Average Embedding + LR
- LDA
- Tree Kernels
- @RNN: I try **word-based 1-layer LSTM** as baseline
- @CNN: I secondly try **character-level CNN**
- @RCNN: I finally will try the advanced RCNN

I first try the simplest word-based LSTM, then try character-level CNN and finally try advanced word-based RCNN if possible.

In [17]:
```
from tensorflow import keras
import tensorflow as tf

from keras.models import Sequential
from keras.layers import Dense, Embedding, LSTM

from sklearn.metrics import classification_report
```

```
Using TensorFlow backend.
```

Baseline: word-based one-layer LSTM

In [18]:
```
# Cut texts after this number of words
max_len = 300
X_train = keras.preprocessing.sequence.pad_sequences(X_train, maxlen=max_
X_val = keras.preprocessing.sequence.pad_sequences(X_val, maxlen=max_len)
X_test = keras.preprocessing.sequence.pad_sequences(X_test, maxlen=max_le

print(X_train.shape)
```

```
(189393, 300)
```

In [19]:
```
# embedding and train
embedding_dimension = 16
n_classes = len(subreddit_mappings)

model = Sequential()
model.add(Embedding(vocabulary_size, embedding_dimension, input_length=ma
model.add(LSTM(128, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(n_classes, activation='softmax'))
model.summary()

# complie
model.compile('adam', 'sparse_categorical_crossentropy', metrics=['accura
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 300, 16)           80000
_________________________________________________________________
lstm_1 (LSTM)                (None, 128)               74240
_________________________________________________________________
dense_1 (Dense)              (None, 17)                2193
=================================================================
Total params: 156,433
Trainable params: 156,433
Non-trainable params: 0
_________________________________________________________________
```

In [22]:

```
# prepare for training
early_stopping = keras.callbacks.EarlyStopping(monitor='acc',
                                               min_delta=0.0001,
                                               patience=1,
                                               verbose=1)

checkpoint = keras.callbacks.ModelCheckpoint('ModelTraining/lstm_1st.hdf5
                                           verbose=1,
                                           save_best_only=True)
# training
history = model.fit(X_train, y_train,
                    batch_size = 32,
                    epochs=5,
                    validation_data=(X_val, y_val),
                    callbacks=[checkpoint, early_stopping])
```

```
Train on 189393 samples, validate on 23675 samples
Epoch 1/5
189393/189393 [==============================] - 2503s 13ms/step - los
s: 1.6889 - acc: 0.4617 - val_loss: 1.3811 - val_acc: 0.5653

Epoch 00001: val_loss improved from inf to 1.38109, saving model to ls
tm_1st.hdf5
Epoch 2/5
189393/189393 [==============================] - 2413s 13ms/step - los
s: 1.3874 - acc: 0.5602 - val_loss: 1.2269 - val_acc: 0.6086

Epoch 00002: val_loss improved from 1.38109 to 1.22686, saving model t
o lstm_1st.hdf5
Epoch 3/5
189393/189393 [==============================] - 3350s 18ms/step - los
s: 1.2123 - acc: 0.6096 - val_loss: 1.1656 - val_acc: 0.6239

Epoch 00003: val_loss improved from 1.22686 to 1.16564, saving model t
o lstm_1st.hdf5
Epoch 4/5
189393/189393 [==============================] - 3233s 17ms/step - los
s: 1.1484 - acc: 0.6259 - val_loss: 1.1239 - val_acc: 0.6324

Epoch 00004: val_loss improved from 1.16564 to 1.12395, saving model t
o lstm_1st.hdf5
Epoch 5/5
189393/189393 [==============================] - 2517s 13ms/step - los
s: 1.1076 - acc: 0.6361 - val_loss: 1.1040 - val_acc: 0.6381

Epoch 00005: val_loss improved from 1.12395 to 1.10398, saving model t
o lstm_1st.hdf5
```

Evaluation and save

Using **f1-score** to capture precision and recall, this model is good to classify 3-'EatingDisorders' with 0.80 and 7-'alcoholism' with 0.81

In [28]:
```
# predict and evaluate
results = model.predict(X_test)
predictions = results.argmax(axis = 1)
print(classification_report(y_test, predictions))
```

```
             precision    recall  f1-score   support

          0       0.59      0.31      0.40       127
          1       0.72      0.56      0.63      1731
          2       0.74      0.61      0.67      1423
          3       0.83      0.77      0.80       123
          4       0.80      0.19      0.30       657
          5       0.57      0.42      0.49       193
          6       0.61      0.68      0.64      5483
          7       0.82      0.79      0.81       502
          8       0.61      0.74      0.67      9109
          9       0.86      0.57      0.68       272
         10       0.00      0.00      0.00       205
         11       0.31      0.04      0.07       737
         12       0.00      0.00      0.00       114
         13       0.76      0.58      0.66       476
         14       0.65      0.64      0.65       276
         15       0.61      0.62      0.61      1001
         16       0.71      0.62      0.66      1245

avg / total       0.63      0.64      0.62     23674
```

In [26]:
```
#Save partly trained model
model.save('ModelTraining/partly_trained_lstm_0613.h5')
```