In [1]:

```python
import numpy as np
import pandas as pd
```

In [2]:

```python
''' Globals '''
MAXBODYSIZE = 500
MAXHEADSIZE = 50
EMBEDDINGDIM = 300
Stances = {'agree', 'disgree', 'discuss', 'unrelated'}
```

In [3]:

```python
''' Load data sets '''
trainBodiesDF = pd.read_csv('./DefaultFiles/train_bodies.csv')
trainHeadDF = pd.read_csv('./DefaultFiles/train_stances.csv')
testBodiesDF = pd.read_csv('./DefaultFiles/test_bodies.csv')
testHeadDF = pd.read_csv('./DefaultFiles/test_stances_unlabeled.csv')
```

In [4]:

```python
'''
    Cleaning
    - drop heads with no reference body
    - drop null heads
    - reset indexes to accomodate change
'''
totalTrain = pd.merge(trainBodiesDF, trainHeadDF, on='Body ID')
trainBodiesDF = totalTrain.groupby('Body ID').first()[['articleBody']]
trainHeadDF = totalTrain[['Body ID','Headline','Stance']]
trainHeadDF = trainHeadDF.dropna()
trainBodiesDF.reset_index(inplace=True)
trainHeadDF.reset_index(inplace=True)
print(trainBodiesDF.head(3))
print(trainHeadDF.head(3))
print(testBodiesDF.head(3))
print(testHeadDF.head(3))
```

```
   Body ID                                        articleBody
0        0  A small meteorite crashed into a wooded area i...
1        4  Last week we hinted at what was to come as Ebo...
2        5  (NEWSER) – Wonder how long a Quarter Pounder w...
   index  Body ID                                           Headline  \
0      0        0  Soldier shot, Parliament locked down after gun...
1      1        0  Tourist dubbed ‘Spider Man’ after spider burro...
2      2        0  Luke Somers 'killed in failed rescue attempt i...

      Stance
0  unrelated
1  unrelated
2  unrelated
   Body ID                                        articleBody
0        1  Al-Sisi has denied Israeli reports stating tha...
1        2  A bereaved Afghan mother took revenge on the T...
2        3  CNBC is reporting Tesla has chosen Nevada as t...
                                            Headline  Body ID
0  Ferguson riots: Pregnant woman loses eye after...     2008
1  Crazy Conservatives Are Sure a Gitmo Detainee ...     1550
2  A Russian Guy Says His Justin Bieber Ringtone ...        2
```

In [5]:

```python
'''
    Load Pretrained Word2Vec by Google
    Word2Vec is a shallow neural network ot produce word embeddings
    The primary goal is vectorize the linguistic context of the word
    You can download from here:
    https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit
'''
from gensim.models.keyedvectors import KeyedVectors
from gensim.models import Word2Vec

word2Vec = KeyedVectors.load_word2vec_format('GensimVectors/GoogleNews-vectors-negative300.bin', binary=True)
```

In [6]:

```
'''
    For downloading for nltk
    import
    on first time download the following packages

    nltk.download()
    select d
    download packages ['punkt', wordnet', 'stopwords']
'''
import nltk
```

In [7]:

```
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer, WordNetLemmatizer
from nltk.corpus import wordnet, stopwords
import re

'''
    Processing text
    1. Split into words i.e [[word,word],[word,word,word]]
    2. Stem - chop of ends
    3. Lemmatise - remove inflection endings and return to base citionary
    4. remove stopwards
    5. only take words containing only letters and contained in Word2Vec vocab
'''
def process( text):
    out = []
    lemmatizer = WordNetLemmatizer()
    stemmer = PorterStemmer()
    stop_words = set(stopwords.words("english"))
    outout = []
    for word in word_tokenize(text):
        word = word.strip().lower()
        word = stemmer.stem(word)
        word = lemmatizer.lemmatize(word, wordnet.VERB)
        # major speed gain only testing for letters
        word = word.replace("n't", 'not')
        word = word.replace("'m", 'am')
        word = word.replace("'ve'", 'have')
        word = word.replace("'d", 'would')
        word = word.replace("'ll", "will")
        if word != '' and word.isalpha() and word in word2Vec:
            out.append(word.lower())
    return out
```

In [8]:

```
'''
    Loop through all four data frames and process the text
    ~ Will take approximately 2 minutes
'''
for index, row in trainBodiesDF.iterrows():
    trainBodiesDF.iat[index, trainBodiesDF.columns.get_loc("articleBody")] = " ".join(process(row['articleBody'])
)
for index, row in trainHeadDF.iterrows():
    trainHeadDF.iat[index, trainHeadDF.columns.get_loc("Headline")] = " ".join(process(row['Headline']))
for index, row in testBodiesDF.iterrows():
    testBodiesDF.iat[index, testBodiesDF.columns.get_loc("articleBody")] = " ".join(process(row['articleBody']))
for index, row in testHeadDF.iterrows():
    testHeadDF.iat[index, testHeadDF.columns.get_loc("Headline")] = " ".join(process(row['Headline']))
```

In [9]:

```
print(trainBodiesDF.head(3))
print(trainHeadDF.head(3))
print(testBodiesDF.head(3))
print(testHeadDF.head(3))
```

```
   Body ID                                        articleBody
0        0  small crash into wood area in capit overnight ...
1        4  last week we hint at what wa come as ebola fea...
2        5  newser wonder how long quarter pounder with ca...
   index  Body ID                                        Headline  \
0      0        0  soldier shoot parliament lock down after erupt...
1      1        0  tourist dub spider man after spider burrow und...
2      2        0                     luke in fail attempt in yemen

      Stance
0  unrelated
1  unrelated
2  unrelated
   Body ID                                        articleBody
0        1  ha report state that he offer extend the gaza ...
1        2  afghan mother take on the taliban after watch ...
2        3  cnbc be report tesla ha choose nevada as the s...
                                            Headline  Body ID
0  ferguson riot pregnant woman lose eye after co...     2008
1                           be sure gitmo kill foley     1550
2  russian guy say hi justin bieber rington save ...        2
```

In [10]:

```
''' Save a checkpoint '''
trainBodiesDF.to_csv('ProcessedTrainBodies.csv',index=False)
trainHeadDF.to_csv('ProcessedTrainHead.csv',index=False)
testBodiesDF.to_csv('ProcessedTestBodies.csv',index=False)
testHeadDF.to_csv('ProcessedTestHead.csv',index=False)
print(trainBodiesDF.shape, trainHeadDF.shape, testBodiesDF.shape, testHeadDF.shape, )
```

```
(1683, 2) (49972, 4) (904, 2) (25413, 2)
```

In [11]:

```
'''
    Create and train tokenizer
    Tokenizer is utilised to create numerical representations of the data
'''
from keras.preprocessing.text import Tokenizer
from keras.utils.np_utils import to_categorical
from keras.preprocessing.sequence import pad_sequences


totalText = []
for index, row in trainBodiesDF.iterrows():
    totalText.append(row['articleBody'])
for index, row in trainHeadDF.iterrows():
    totalText.append(row['Headline'])
for index, row in testBodiesDF.iterrows():
    totalText.append(row['articleBody'])
for index, row in testHeadDF.iterrows():
    totalText.append(row['Headline'])

tokenizer = Tokenizer()
tokenizer.fit_on_texts(totalText)
wordIndexs = tokenizer.word_index
vocabSize = tokenizer.word_counts
print('Vocab Size: ',len(wordIndexs))
```

```
Using TensorFlow backend.

Vocab Size:  9309
```

In [12]:

```
'''
    utilise tokenizer and save word representations
'''
wordIndexsdf = pd.DataFrame.from_dict(wordIndexs, orient='index')
wordIndexsdf.to_csv('wordIndexs.csv',index=False)
wordIndexsdf.head(5)
```

Out[12]:

|  | 0 |
|---|---|
| **the** | 1 |
| **be** | 2 |
| **in** | 3 |
| **on** | 4 |
| **for** | 5 |

In [13]:

```
embeddingVector = {}
for word, index in wordIndexs.items():
    if word != '':
        embeddingVector[index] = word2Vec[word]
embeddingdf = pd.DataFrame.from_dict(embeddingVector, orient='index')
embeddingdf.to_csv('embeddingVectors.csv',index=False)
embeddingdf.head(5)
```

Out[13]:

|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | ... | 290 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 0.080078 | 0.104980 | 0.049805 | 0.053467 | -0.067383 | -0.120605 | 0.035156 | -0.118652 | 0.043945 | 0.030151 | ... | -0.071289 |
| **2** | -0.228516 | -0.088379 | 0.127930 | 0.150391 | -0.073242 | 0.086426 | 0.063965 | 0.096680 | 0.058350 | 0.143555 | ... | -0.109863 |
| **3** | 0.070312 | 0.086914 | 0.087891 | 0.062500 | 0.069336 | -0.108887 | -0.081543 | -0.154297 | 0.020752 | 0.131836 | ... | -0.168945 |
| **4** | 0.026733 | -0.090820 | 0.027832 | 0.204102 | 0.006226 | -0.090332 | 0.022583 | -0.161133 | 0.132812 | 0.061035 | ... | 0.026855 |
| **5** | -0.011780 | -0.047363 | 0.044678 | 0.063477 | -0.018188 | -0.063965 | -0.001312 | -0.072266 | 0.064453 | 0.086426 | ... | -0.022583 |

5 rows × 300 columns

In [14]:

```
embeddingMatrix = embeddingdf.to_numpy()
embeddingMatrix[5:]
```

Out[14]:

```
array([[-0.01574707, -0.02832031,  0.08349609, ...,  0.00686646,
         0.06103516, -0.1484375 ],
       [ 0.08447266, -0.00035286,  0.05322266, ...,  0.01708984,
         0.06079102, -0.10888672],
       [-0.03613281, -0.12109375,  0.13378906, ..., -0.08642578,
         0.14355469,  0.02734375],
       ...,
       [ 0.20605469, -0.29882812,  0.06298828, ...,  0.13671875,
        -0.17675781, -0.11523438],
       [ 0.00595093,  0.00102997, -0.19921875, ..., -0.3046875 ,
         0.05151367, -0.17382812],
       [-0.01330566,  0.0088501 ,  0.01184082, ..., -0.07373047,
        -0.08056641,  0.0703125 ]])
```

In [22]:

```
'''
    Loaded Function
    Purposes
    - Change pandas dataframe to trainable / testable numpy data
    - texts to sequences - convert words into their appropriate numerical representation
    - pad_sequences - convert all vectors into desired length (increase / decrease size)
    - for train data - convert stances into numerical representation
'''
def CreateNetworkData(bodydf, headdf, stance):
    heads = []
    bodies = []
    stances = []
    stancesLookup = {'unrelated': 0 , 'agree':1, 'disagree':2, 'discuss':3}
    for index, row in headdf.iterrows():
        # don't drop rows in test
        if not stance:
            if pd.isna(row['Headline']):
                heads.append([])
            else:
                heads.append(row['Headline'].split(" "))
            try:
                bodies.append(bodydf.loc[bodydf['Body ID'] == int(row['Body ID'])].iloc[0]['articleBody'][0].spli
t(" "))
            except Exception:
                print(bodydf.loc[bodydf['Body ID'] == int(row['Body ID'])].iloc[0]['articleBody'])
            if stance:
                stances.append(stancesLookup[row['Stance'].strip()])
        else:
            if not pd.isna(row['Headline']):
                heads.append(row['Headline'].split(" "))
                try:
                    bodies.append(bodydf.loc[bodydf['Body ID'] == int(row['Body ID'])].iloc[0]['articleBody'][0].
split(" "))
                except Exception:
                    print(row['Body ID'])
                    bodies.append([])
                if stance:
                    stances.append(stancesLookup[row['Stance'].strip()])
    heads = tokenizer.texts_to_sequences(heads)
    bodies = tokenizer.texts_to_sequences(bodies)
    heads = pad_sequences(heads,maxlen = MAXHEADSIZE,padding = 'post')
    bodies = pad_sequences(bodies,maxlen = MAXBODYSIZE,padding = 'post')
    if stance:
        stances = to_categorical(stances, num_classes=4)
    return heads,bodies,stances
```

In [23]:

```
'''
    Create data structures for lstm nework
'''
trainHeads,trainBodies,trainStances = CreateNetworkData(trainBodiesDF, trainHeadDF, True)
```

```
624
```

In [24]:

```
import keras

from keras.layers.embeddings import Embedding
from keras.models import Sequential, Model
from keras.layers import Dense, Dropout, Embedding, Input
from keras.layers.wrappers import Bidirectional
from keras.layers.recurrent import LSTM
from keras.layers import concatenate
from keras.preprocessing import sequence
```

In [25]:

```
'''
    Bidirectional LSTM used
    inputs are concatenated and feed into a two layer dense network with dropout
    Please refer to report for further information about method
'''

InputHead = Input(shape=(MAXHEADSIZE,), dtype='int32', name='InputHead')
InputBody = Input(shape=(MAXBODYSIZE,), dtype='int32', name='InputBody')
Embeddings = Embedding(len(wordIndexs), EMBEDDINGDIM, weights=[embeddingMatrix],trainable=False)
EmbedHead = Embeddings(InputHead)
EmbedBody = Embeddings(InputBody)

LSTMHead = Bidirectional(LSTM(64,dropout=0.2, recurrent_dropout=0.2, name='LSTMHead'))(EmbedHead)
LSTMBody = Bidirectional(LSTM(64,dropout=0.2, recurrent_dropout=0.2, name='LSTMBody'))(EmbedBody)

Concat = concatenate([LSTMHead,LSTMBody])

DenseLayer = Dense(128,activation='relu')(Concat)
DenseLayer = Dropout(0.4)(DenseLayer)
DenseLayer = Dense(4,activation='softmax')(DenseLayer)
LSTMNetwork = Model(inputs=[InputHead,InputBody], outputs=[DenseLayer])
LSTMNetwork.compile(optimizer='adam', loss='categorical_crossentropy',metrics=['acc'])
print(LSTMNetwork.summary())
```

```
WARNING:tensorflow:From /home/leo/.local/lib/python3.7/site-packages/tensorflow/python/framework/op_
def_library.py:263: colocate_with (from tensorflow.python.framework.ops) is deprecated and will be r
emoved in a future version.
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From /home/leo/.local/lib/python3.7/site-packages/keras/backend/tensorflow_backen
d.py:3445: calling dropout (from tensorflow.python.ops.nn_ops) with keep_prob is deprecated and will
be removed in a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - keep_prob`.
____________________________________________________________________________________________________
Layer (type)                     Output Shape          Param #     Connected to
====================================================================================================
InputHead (InputLayer)           (None, 50)            0
____________________________________________________________________________________________________
InputBody (InputLayer)           (None, 500)           0
____________________________________________________________________________________________________
embedding_1 (Embedding)          multiple              2792700     InputHead[0][0]
                                                                   InputBody[0][0]
____________________________________________________________________________________________________
bidirectional_1 (Bidirectional) (None, 128)            186880      embedding_1[0][0]
____________________________________________________________________________________________________
bidirectional_2 (Bidirectional) (None, 128)            186880      embedding_1[1][0]
____________________________________________________________________________________________________
concatenate_1 (Concatenate)      (None, 256)           0           bidirectional_1[0][0]
                                                                   bidirectional_2[0][0]
____________________________________________________________________________________________________
dense_1 (Dense)                  (None, 128)           32896       concatenate_1[0][0]
____________________________________________________________________________________________________
dropout_1 (Dropout)              (None, 128)           0           dense_1[0][0]
____________________________________________________________________________________________________
dense_2 (Dense)                  (None, 4)             516         dropout_1[0][0]
====================================================================================================
Total params: 3,199,872
Trainable params: 407,172
Non-trainable params: 2,792,700
____________________________________________________________________________________________________
None
```

In [26]:

```
''' Train the model ~ takes roughly 10 hours '''
for i in range(10):
    LSTMNetwork.fit([trainHeads, trainBodies],[trainStances], epochs=4, batch_size=128,verbose = True)
```

```
WARNING:tensorflow:From /home/leo/.local/lib/python3.7/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 futur
e version.
Instructions for updating:
Use tf.cast instead.
Epoch 1/4
49972/49972 [==============================] - 1132s 23ms/step - loss: 0.7947 - acc: 0.7295
Epoch 2/4
49972/49972 [==============================] - 1125s 23ms/step - loss: 0.7481 - acc: 0.7312
Epoch 3/4
49972/49972 [==============================] - 1126s 23ms/step - loss: 0.7273 - acc: 0.7314
Epoch 4/4
```

```
49972/49972 [==============================] - 1128s 23ms/step - loss: 0.7054 - acc: 0.7324
Epoch 1/4
49972/49972 [==============================] - 1127s 23ms/step - loss: 0.6860 - acc: 0.7360
Epoch 2/4
49972/49972 [==============================] - 1129s 23ms/step - loss: 0.6680 - acc: 0.7388
Epoch 3/4
49972/49972 [==============================] - 1128s 23ms/step - loss: 0.6541 - acc: 0.7422
Epoch 4/4
49972/49972 [==============================] - 1128s 23ms/step - loss: 0.6419 - acc: 0.7437
Epoch 1/4
49972/49972 [==============================] - 1129s 23ms/step - loss: 0.6305 - acc: 0.7460
Epoch 2/4
49972/49972 [==============================] - 1126s 23ms/step - loss: 0.6241 - acc: 0.7482
Epoch 3/4
49972/49972 [==============================] - 1129s 23ms/step - loss: 0.6144 - acc: 0.7496
Epoch 4/4
49972/49972 [==============================] - 1128s 23ms/step - loss: 0.6052 - acc: 0.7514
Epoch 1/4
49972/49972 [==============================] - 1129s 23ms/step - loss: 0.5975 - acc: 0.7534
Epoch 2/4
49972/49972 [==============================] - 1129s 23ms/step - loss: 0.5911 - acc: 0.7546
Epoch 3/4
49972/49972 [==============================] - 1129s 23ms/step - loss: 0.5860 - acc: 0.7566
Epoch 4/4
49972/49972 [==============================] - 1130s 23ms/step - loss: 0.5787 - acc: 0.7579
Epoch 1/4
49972/49972 [==============================] - 1130s 23ms/step - loss: 0.5745 - acc: 0.7584
Epoch 2/4
49972/49972 [==============================] - 1129s 23ms/step - loss: 0.5684 - acc: 0.7607
Epoch 3/4
49972/49972 [==============================] - 1131s 23ms/step - loss: 0.5600 - acc: 0.7633
Epoch 4/4
49972/49972 [==============================] - 1129s 23ms/step - loss: 0.5575 - acc: 0.7649
Epoch 1/4
49972/49972 [==============================] - 1134s 23ms/step - loss: 0.5503 - acc: 0.7653
Epoch 2/4
49972/49972 [==============================] - 1128s 23ms/step - loss: 0.5467 - acc: 0.7675
Epoch 3/4
49972/49972 [==============================] - 1129s 23ms/step - loss: 0.5442 - acc: 0.7661
Epoch 4/4
49972/49972 [==============================] - 1129s 23ms/step - loss: 0.5397 - acc: 0.7671
Epoch 1/4
49972/49972 [==============================] - 1129s 23ms/step - loss: 0.5373 - acc: 0.7692
Epoch 2/4
49972/49972 [==============================] - 1129s 23ms/step - loss: 0.5298 - acc: 0.7705
Epoch 3/4
49972/49972 [==============================] - 1130s 23ms/step - loss: 0.5284 - acc: 0.7714
Epoch 4/4
49972/49972 [==============================] - 1131s 23ms/step - loss: 0.5276 - acc: 0.7727
Epoch 1/4
49972/49972 [==============================] - 1130s 23ms/step - loss: 0.5226 - acc: 0.7716
Epoch 2/4
49972/49972 [==============================] - 1134s 23ms/step - loss: 0.5203 - acc: 0.7740
Epoch 3/4
49972/49972 [==============================] - 1141s 23ms/step - loss: 0.5151 - acc: 0.7766
Epoch 4/4
49972/49972 [==============================] - 1130s 23ms/step - loss: 0.5140 - acc: 0.7765
Epoch 1/4
49972/49972 [==============================] - 1133s 23ms/step - loss: 0.5085 - acc: 0.7755
Epoch 2/4
49972/49972 [==============================] - 1131s 23ms/step - loss: 0.5086 - acc: 0.7759
Epoch 3/4
49972/49972 [==============================] - 1131s 23ms/step - loss: 0.5060 - acc: 0.7779
Epoch 4/4
49972/49972 [==============================] - 1131s 23ms/step - loss: 0.5040 - acc: 0.7788
Epoch 1/4
49972/49972 [==============================] - 1131s 23ms/step - loss: 0.5005 - acc: 0.7800
Epoch 2/4
49972/49972 [==============================] - 1131s 23ms/step - loss: 0.4985 - acc: 0.7798
Epoch 3/4
49972/49972 [==============================] - 1132s 23ms/step - loss: 0.4978 - acc: 0.7788
Epoch 4/4
49972/49972 [==============================] - 1131s 23ms/step - loss: 0.4942 - acc: 0.7807
```

In [27]:

```
''' Save all your hard work '''
LSTMNetwork.save('finalModel.5h')
```

In [28]:

```
from keras.models import load_model

LSTMNetwork = load_model('finalModel.5h')
```

In [29]:

```
''' Create test data appropriate for model '''
testHeads,testBodies,out = CreateNetworkData(testBodiesDF, testHeadDF, False)
```

In [30]:

```
''' Predict the test data '''
predictions = LSTMNetwork.predict([testHeads, testBodies])
```

In [31]:

```
'''
    Convert predictions into csv approrpiate for evaluating
    - take argmax of predictions to determine classifcaiton
    - map these back to the appropriate stance in word
'''
testStancesDf = pd.read_csv('./DefaultFiles/test_stances_unlabeled.csv')
reverseMap = np.vectorize(lambda label: { 0:'unrelated', 1:'agree', 2:'disagree', 3:'discuss'}[label])
testPredsFinal = np.column_stack((testStancesDf, reverseMap(np.argmax(predictions,axis=1))))
testPredsFinal
```

Out[31]:

```
array([['Ferguson riots: Pregnant woman loses eye after cops fire BEAN BAG round through car window'
,
        2008, 'unrelated'],
       ['Crazy Conservatives Are Sure a Gitmo Detainee Killed James Foley',
        1550, 'unrelated'],
       ['A Russian Guy Says His Justin Bieber Ringtone Saved Him From A Bear Attack',
        2, 'unrelated'],
       ...,
       ['The success of the Affordable Care Act is a hugely inconvenient truth for its opponents',
        2584, 'unrelated'],
       ['The success of the Affordable Care Act is a hugely inconvenient truth for its opponents',
        2585, 'unrelated'],
       ['The success of the Affordable Care Act is a hugely inconvenient truth for its opponents',
        2586, 'unrelated']], dtype=object)
```

In [32]:

```
''' One liner to save dataframe appropriately '''
pd.DataFrame(testPredsFinal, columns=['Headline', 'Body ID', 'Stance']).to_csv('testPredictions.csv', index=False
)
```

In [33]:

```
'''
    Print out the confusion matrix of the predictions and evaluate score
'''

%run -i scorer.py DefaultFiles/competition_test_stances.csv testPredictions.csv
```

```
CONFUSION MATRIX:
-------------------------------------------------------------
|           |   agree   | disagree  |  discuss  | unrelated |
-------------------------------------------------------------
|   agree   |    29     |     2     |    35     |   1837    |
-------------------------------------------------------------
| disagree  |     2     |     0     |     5     |    690    |
-------------------------------------------------------------
|  discuss  |    23     |     1     |    175    |   4265    |
-------------------------------------------------------------
| unrelated |    107    |     1     |    379    |   17862   |
-------------------------------------------------------------
ACCURACY: 0.711

MAX  - the best possible score (100% accuracy)
NULL - score as if all predicted stances were unrelated
TEST - score based on the provided predictions

||    MAX    ||    NULL   ||    TEST   ||
|| 11651.25  ||  4587.25  ||  4686.5   ||
```