ELMO embedding in classifying Engageable and Non-engageable messages

In [1]:

```
import tensorflow_hub as hub
import tensorflow as tf
```

In [2]:

```
elmo = hub.Module("https://tfhub.dev/google/elmo/2", trainable=True)
```

In [3]:

```
import keras
from keras import backend as K
from keras.models import Model
from keras.layers import Dense, Embedding, Input
from keras.layers import LSTM, Bidirectional, GlobalMaxPool1D, Dropout
from keras.preprocessing import text, sequence
#from keras.callbacks import EarlyStopping, ModelCheckpoint
```

```
Using TensorFlow backend.
```

In [4]:

```
# Have taken only 10000 data points as elmo takes alot of time to exec
import pickle

# load elmo_train_new
pickle_in = open("train.pickle", "rb")
train = pickle.load(pickle_in)

# load elmo_train_new
pickle_in = open("test.pickle", "rb")
test = pickle.load(pickle_in)
```

In [10]:

```
list_sentences_train=train['CleanText']
list_sentences_test=test['CleanText']
y_train=train['Engage']
y_test=test['Engage']

def ELMoEmbedding(x):
    return elmo(tf.squeeze(tf.cast(x,tf.string)),signature="default",a
```

In [12]:
```
import keras_metrics

def build_model():
    input_text=Input(shape=(1,),dtype="string")
    embedding=Lambda(ELMoEmbedding,output_shape=(1024,))(input_text)
    dense=Dense(256,activation='relu',kernel_regularizer=keras.regular
    pred=Dense(1,activation='sigmoid')(dense)
    model=Model(inputs=[input_text],outputs=pred)
    model.compile(loss='binary_crossentropy', optimizer='adam', metric

    return model

model_elmo=build_model()
```

In [13]:
```
model_elmo.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_4 (InputLayer)         (None, 1)                 0         
_________________________________________________________________
lambda_4 (Lambda)            (None, 1024)              0         
_________________________________________________________________
dense_6 (Dense)              (None, 256)               262400    
_________________________________________________________________
dense_7 (Dense)              (None, 1)                 257       
=================================================================
Total params: 262,657
Trainable params: 262,657
Non-trainable params: 0
_________________________________________________________________
```

In [14]:
```
with tf.Session() as session:
    K.set_session(session)
    session.run(tf.global_variables_initializer())
    session.run(tf.tables_initializer())
    history=model_elmo.fit(list_sentences_train, y_train, validation_da
```

```
Train on 8000 samples, validate on 2000 samples
Epoch 1/5
8000/8000 [==============================] - 2545s 318ms/step - loss
: 0.8254 - acc: 0.7268 - precision: 0.7091 - recall: 0.8012 - val_lo
ss: 0.6370 - val_acc: 0.7760 - val_precision: 0.7427 - val_recall: 0
.8852
Epoch 2/5
8000/8000 [==============================] - 2550s 319ms/step - loss
: 0.7316 - acc: 0.7468 - precision: 0.7326 - recall: 0.8048 - val_lo
ss: 0.5896 - val_acc: 0.7755 - val_precision: 0.7452 - val_recall: 0
.8777
Epoch 3/5
8000/8000 [==============================] - 2548s 319ms/step - loss
: 0.6122 - acc: 0.7756 - precision: 0.7572 - recall: 0.8345 - val_lo
ss: 0.5543 - val_acc: 0.7815 - val_precision: 0.7520 - val_recall: 0
.8786
Epoch 4/5
8000/8000 [==============================] - 2556s 319ms/step - loss
: 0.6919 - acc: 0.7509 - precision: 0.7360 - recall: 0.8094 - val_lo
ss: 0.5646 - val_acc: 0.7630 - val_precision: 0.7108 - val_recall: 0
.9341
Epoch 5/5
8000/8000 [==============================] - 2520s 315ms/step - loss
: 0.5527 - acc: 0.7811 - precision: 0.7637 - recall: 0.8361 - val_lo
ss: 0.6241 - val_acc: 0.7665 - val_precision: 0.7865 - val_recall: 0
.7695
```

In [1]:
```
from prettytable import PrettyTable
x = PrettyTable()

x.field_names = ["Test Accuracy","Precision","Recall",'F1 Score']
x.add_row([76.65,78.65,76.95,77.79])

print('\tEmbeddings from Language Models ')
print(x)
```

```
        Embeddings from Language Models
+---------------+-----------+--------+----------+
| Test Accuracy | Precision | Recall | F1 Score |
+---------------+-----------+--------+----------+
|     76.65     |   78.65   | 76.95  |  77.79   |
+---------------+-----------+--------+----------+
```

Conclusion:

1) As the dataset count is low and the sentences in the reviews are **Polysemy** (Polysemy wherein a word could have multiple meanings or senses), we use ELMO embedding which overcomes the problem of Polysemy.

2) I have used only 10000 dataset, as ELMO embedding is computationally expensive and also used only 5 epochs because of the computation cost.

3) More dataset and higher number of epochs may increase the model performance.