**NAME**: SHIVAPRAKASH BALASUBRAMANIAN **EMAIL ADDRESS**: sbalas22@ncsu.edu

# CSC 791: Natural Language Processing
# P1: Sentiment Analysis

## PROJECT DESCRIPTION

This assignment is to learn how to use word vectors to classify sentences based on the sentiment they express. Specifically, this assignment emphasizes long sentences, which are often challenging for existing sentiment analysis tools.

Select 2 word embeddings (vectors/encoders) model in your baseline model. As a baseline model, you can use a word-embedding of your choice (or Tf-IDF) to vectorize text and a classification approach to classify each sentence based on its overall sentiment. You need to take the following steps to get a baseline model.

- Tokenize each sentence in word tokens
- Compute vectors for each word token in a sentence and average these word vectors to get a vector for the sentence
- Train a classifier to classify each long sentence into 0, 1, 2

**BASELINE MODEL 1:**

- Preprocessing: Stopword Removal from Spacy list of Stop Words
- Tokenization: Spacy Tokeneizer to get the base words
- Vectorizer: TF-IDF vectorizer
- Classifier: Linear SVM (LinearSVC)

```
[[  8  41  33]
 [  4 203  96]
 [ 13 108 177]]
              precision    recall  f1-score   support

           0       0.32      0.10      0.15        82
           1       0.58      0.67      0.62       303
           2       0.58      0.59      0.59       298

    accuracy                           0.57       683
   macro avg       0.49      0.45      0.45       683
weighted avg       0.55      0.57      0.55       683

Accuracy:  0.568081991215227
```

**BASELINE MODEL 2:**

- Preprocessing: Stopword Removal from Spacy list of Stop Words
- Tokenization: TensorFlow Universal Sentence Encoder
- Vectorizer: TensorFlow Universal Sentence Vector
- Classifier: Linear SVM (LinearSVC)

```
[[ 13  46  23]
 [ 18 212  73]
 [ 15  93 190]]
              precision    recall  f1-score   support

           0       0.28      0.16      0.20        82
           1       0.60      0.70      0.65       303
           2       0.66      0.64      0.65       298

    accuracy                           0.61       683
   macro avg       0.52      0.50      0.50       683
weighted avg       0.59      0.61      0.60       683

Accuracy: 0.6076134699853587
```

**PROPOSED APPROACH:**

Spacy is an open-source Python framework for NLP. Spacy comes with a pre-trained word2vec model with a vocabulary of more than a million words. Instead of using the tokens directly, the new approach introduced is averaging out the vectors from individual tokens generated by the Spacy tokenizer. This probably captures the essence (sentiment) of long sentences better than individually chunking the words. The process is as below:

- Preprocessing: Stopword Removal from Spacy list of Stop Words
- Tokenization: Spacy Tokeneizer to get the base words
- Vectorizer: Spacy Vectorizer along with averaging
- Classifier: Linear SVM (LinearSVC)

```
[114] # For Spacy word2vec

      def createVector(sent):
        vect = nlp(sent['sentence'])
        avg = np.zeros(300)
        # print(vect)

        # Calculate average vector
        for token in vect:
          avg += token.vector
        vect_avg = avg/len(vect)

        return vect_avg.tolist()
```

```
[115] # Apply Transformation
      df_train['vector'] = df_train.apply(lambda i : createVector(i), axis=1)
      df_test['vector']  = df_test.apply(lambda i  : createVector(i), axis=1)
```

```
[[  7  45  30]
 [ 21 207  75]
 [ 11  87 200]]
              precision    recall  f1-score   support

           0       0.18      0.09      0.12        82
           1       0.61      0.68      0.64       303
           2       0.66      0.67      0.66       298

    accuracy                           0.61       683
   macro avg       0.48      0.48      0.47       683
weighted avg       0.58      0.61      0.59       683

Accuracy: 0.6061493411420205
```

**PERFORMANCE ANALYSIS:**

| Metric | Accuracy | Weighted F1 Score |
|---|---|---|
| Baseline Model 1<br>Basic TF-IDF | 0.57 | 0.55 |
| Baseline Model 2<br>TensorFlow Universal Sentence Encoder | 0.61 | 0.59 |
| Proposed Approach<br>Spacy with Averaging | 0.61 | 0.61 |