**Detecting depression on social media**

**Motivation**

With its 230 million regular users, Twitter has become such a broad stream of personal expression that researchers are beginning to use it as a tool to dig into public health problems.

- excerpt from *Time*, 2014.

The goal of this experiment is to perform sentiment analysis on random tweets and facebook posts and detect signs of depression in these texts. The task is classfication of normal and depressive tweets, where depressive tweets are defined as tweets that contain depression-related keywords.

The code for this experiment is available in this [notebook](https://drive.google.com/drive/folders/1OqFH8LZ6NYB0xgyQ_c2Ld3Fahr9-PlxS?usp=drive_link).

**Dataset**

**Sentiment140**: the **Sentiment140** dataset containing 1,6 million tweets from the Twitter API with the 6 following attributes: *target*, *id*, *date*, *flag*, *user*, *text*. For the classification task, I took a sample of 8,000 random tweets.

Public tweets scraped using the Twitter API: Since there is no readily available public dataset on depression, I used a Twitter scrape tool called **Twint** to collect 2,345 tweets that contain the depression-related keywords.

For the collection of Facebook posts, A Python scraper library facebook\_scraper

Is used in scraping data for depression related post from facebook , approximately 15,000 data points were gathered.

The three datasets are labelled respectively ( 0 denotes normal tweets, 1 denotes depressive tweets) and shuffle-merged into one big dataset containing 25858 data with 2 attributes: *text* and *label*.

**Preprocessing**

To prepare the data for training, I remove bad symbols, stop words, punctuations, and expand contractions from the tweets. The tweets are then tokenized . A wordcloud image of both depressive and non depressive post are generated ,

5000 datapoints are then split into 80% training and 20% testing.

**Training**

**Hyper-parameters**

Number of unique words in the vocabulary:

MAX\_NUM\_WORDS = 18611

Maximum sentence length:

MAX\_SEQ\_LENGTH = 33

Embedding size: EMBEDDING\_DIM = 66

**Models**

**LSTM Model**

embedding\_vector\_features=sent\_length\*2

lstm\_model=Sequential()

lstm\_model.add(Embedding(voc\_size,embedding\_vector\_features,input\_length=sent\_length))

lstm\_model.add((LSTM(100)))

lstm\_model.add(Dense(1,activation='sigmoid'))

lstm\_model.compile(loss='binary\_crossentropy',optimizer='adam',metrics=['accuracy'])

Model: "sequential\_2"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

embedding\_2 (Embedding) (None, 33, 66) 1228326

lstm\_2 (LSTM) (None, 100) 66800

dense\_2 (Dense) (None, 1) 101

=================================================================

Total params: 1295227 (4.94 MB)

Trainable params: 1295227 (4.94 MB)

Non-trainable params: 0 (0.00 Byte)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### RNN Model

rnn\_model = Sequential()

rnn\_model.add(Embedding(voc\_size, embedding\_vector\_features, input\_length=sent\_length))

rnn\_model.add(SimpleRNN(100))

rnn\_model.add(Dense(1, activation='sigmoid'))

rnn\_model.compile(loss='binary\_crossentropy', optimizer='adam', metrics=['accuracy'])

print(rnn\_model.summary())

Model: "sequential\_3"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

embedding\_3 (Embedding) (None, 33, 66) 1228326

simple\_rnn (SimpleRNN) (None, 100) 16700

dense\_3 (Dense) (None, 1) 101

=================================================================

Total params: 1245127 (4.75 MB)

Trainable params: 1245127 (4.75 MB)

Non-trainable params: 0 (0.00 Byte)

**BiLSTM**

bilstm\_model = Sequential()

bilstm\_model.add(Embedding(voc\_size, embedding\_vector\_features, input\_length=sent\_length))

bilstm\_model.add(Bidirectional(LSTM(100)))

bilstm\_model.add(Dense(1, activation='sigmoid'))

bilstm\_model.compile(loss='binary\_crossentropy', optimizer='adam', metrics=['accuracy'])

Model: "sequential\_4"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

embedding\_4 (Embedding) (None, 33, 66) 1228326

bidirectional (Bidirection (None, 200) 133600

al)

dense\_4 (Dense) (None, 1) 201

=================================================================

Total params: 1362127 (5.20 MB)

Trainable params: 1362127 (5.20 MB)

Non-trainable params: 0 (0.00 Byte)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

None

**BERT**

**from sentence\_transformers import SentenceTransformer**

**bert\_model = SentenceTransformer('distilbert-base-nli-mean-tokens')**

**embeddings = bert\_model.encode(tweets, show\_progress\_bar=True)**

**(5000, 768)**

**classifier = Sequential()**

**classifier.add (layers.Dense(256, activation='relu', input\_shape=(768,)))**

**classifier.add (layers.Dense(1, activation='sigmoid'))**

**classifier.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])**

**hist = classifier.fit (X\_train, y\_train, epochs=100, batch\_size=16, validation\_data=(X\_test, y\_test))**

**Results**

Below are the averaged results of different models used for this task.

| **Models** | **Accuracy** | **F1** | **Precision** | **Recall** |
| --- | --- | --- | --- | --- |
| LSTM | 0.79 | 0.79 | 0.79 | 0.79 |
| RNN | 0.78 | 0.78 | 0.78 | 0.78 |
| BiLSTM | 0.78 | 0.78 | 0.78 | 0.78 |
| BERT(Tranformer) | **0.95** | **0.95** | **0.95** | **0.95** |