APPENDIX

READ DATASET

```
1. tweet_comments = pd.read_csv("tweet.csv")
2. tweet_comments.shape
3. tweet_comments.head(5)
```

CHECKING MISS VALUES

```
4. tweet_comments.isnull().values.any()
```

DATASET VISUALIZATION

```
5. import seaborn as sns
6. sns.countplot(x='sentiment', data = tweet_comments)
```

PREPROCESSING

```
7. tweet_comments["text"][5]
8.
9. TAG_RE = re.compile(r'<[^>]+>')
10.
11. def remove_tags(text):
12.
13.     return TAG_RE.sub('', text)
14.
15. import nltk
16. nltk.download('stopwords')
17.
18. def preprocess_text(sen):
19.     '''Cleans text data up, leaving only 2 or more char long non-
    stepwords composed of A-Z & a-z only
20.     in lowercase'''
21.
22.     sentence = sen.lower()
23.
24.     # Remove html tags
25.     sentence = remove_tags(sentence)
26.
27.     # Remove punctuations and numbers
```

```
    sentence = re.sub('[^a-zA-Z]', ' ', sentence)

    # Single character removal
    sentence = re.sub(r"\s+[a-zA-Z]\s+", ' ', sentence)

    # Remove multiple spaces
    sentence = re.sub(r'\s+', ' ', sentence)

    # Remove Stopwords
    pattern = re.compile(r'\b(' + r'|'.join(stopwords.words('english')
) + r')\b\s*')
    sentence = pattern.sub('', sentence)

    return sentence
```

CONCAT PREPROCESSING TEXT FUNCTION ON TWEETS

```
X = []
sentences = list(tweet_comments['text'])
for sen in sentences:
    X.append(preprocess_text(sen))
```

SPLITING THE DATASET

```
# Converting sentiment labels to 0 & 1

y = tweet_comments['sentiment']

y = np.array(list(map(lambda x: 1 if x=="positive" else 0, y)))

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.
30, random_state=20)
```

WORD EMBEDDING GLOVE

```
#tokenizing the data
word_tokenizer = Tokenizer()
word_tokenizer.fit_on_texts(X_train)

X_train = word_tokenizer.texts_to_sequences(X_train)
X_test = word_tokenizer.texts_to_sequences(X_test)
# adding 1 to store dimension for words for which no pretrained word emb
ed exist
```

```
58.
59. vocab_length = len(word_tokenizer.word_index) + 1
60.
61. vocab_length
62.
63. # padding all review to fixd length 100
64.
65. maxlen = 100
66.
67. X_train = pad_sequences(X_train, padding='post', maxlen=maxlen)
68. X_test = pad_sequences(X_test, padding='post', maxlen=maxlen)
69.
70. # Load GloVe embeding and create embed dictionary
71.
72. from numpy import asarray
73. from numpy import zeros
74.
75. embeddings_dictionary = dict()
76. glove_file = open('a2_glove.6B.100d.txt', encoding="utf8")
77.
78. for line in glove_file:
79.   records = line.split()
80.   word = records[0]
81.   vector_dimensions = asarray(records[1:], dtype='float32')
82.   embeddings_dictionary [word] = vector_dimensions
83. glove_file.close()
84.
85. max(len(data) for data in tweet_comments['text'])
86.
87. # create embedding matrix 100 column
88.
89. embedding_matrix = zeros ((vocab_length, 100))
90. for word, index in word_tokenizer.word_index.items():
91.   embedding_vector = embeddings_dictionary.get(word)
92.   if embedding_vector is not None:
93.     embedding_matrix[index] = embedding_vector
94.
95. embedding_matrix.shape
```

CONVOLUTIONAL 1D

```
#convolutional 1D
from keras.layers import Conv1D

# neural net architec

cnn_model = Sequential()

embedding_layer = Embedding(vocab_length, 100, weights=[embedding_matrix], input_length=maxlen , trainable = False)
cnn_model.add(embedding_layer)

cnn_model.add(Conv1D(128, 5, activation='relu'))
cnn_model.add(GlobalMaxPooling1D())
cnn_model.add(Dense(1, activation='sigmoid'))

#model compiling

cnn_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['acc'])
print(cnn_model.summary())

#model training

cnn_model_history = cnn_model.fit(X_train, y_train, batch_size=128, epochs=20, verbose=1, validation_split=0.2)

#prediction on test

score = cnn_model.evaluate(X_test, y_test, verbose=1)

#model perfo
print("Test Accuracy:", score[1])

#model visuali

import matplotlib.pyplot as plt

plt.plot(cnn_model_history.history['acc'])
plt.plot(cnn_model_history.history['val_acc'])

plt.title('model accuracy')
plt.ylabel('accuracy')
```

```
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc = 'upper left')
plt.show()

plt.plot(cnn_model_history.history['loss'])
plt.plot(cnn_model_history.history['val_loss'])

plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train','test'], loc = 'upper left')
plt.show()
```

LSTM

```
from keras.layers import LSTM

# Neural Network architecture

lstm_model = Sequential()
embedding_layer = Embedding(vocab_length, 100, weights=[embedding_matrix], input_length=maxlen , trainable=False)

lstm_model.add(embedding_layer)
lstm_model.add(Bidirectional(LSTM(128)))
lstm_model.add(Dense(1, activation='sigmoid'))

# Model compiling

lstm_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['acc'])
print(lstm_model.summary())

# Model Training

lstm_model_history = lstm_model.fit(X_train, y_train, batch_size=128, epochs=20, verbose=1, validation_split=0.2)

# Predictions on the Test Set
```

```
168.
169.score = lstm_model.evaluate(X_test, y_test, verbose=1)
170.
171.# Model Performance
172.
173.print("Test Accuracy:", score[1])
174.
175.# Model Performance Charts
176.
177.import matplotlib.pyplot as plt
178.
179.plt.plot(lstm_model_history.history['acc'])
180.plt.plot(lstm_model_history.history['val_acc'])
181.
182.plt.title('model accuracy')
183.plt.ylabel('accuracy')
184.plt.xlabel('epoch')
185.plt.legend(['train','test'], loc='upper left')
186.plt.show()
187.
188.plt.plot(lstm_model_history.history['loss'])
189.plt.plot(lstm_model_history.history['val_loss'])
190.
191.plt.title('model loss')
192.plt.ylabel('loss')
193.plt.xlabel('epoch')
194.plt.legend(['train','test'], loc='upper left')
195.plt.show()
```

PAPER NAME

TA-19.K1.0045.docx

WORD COUNT

7809 Words

CHARACTER COUNT

40835 Characters

PAGE COUNT

31 Pages

FILE SIZE

59.1KB

SUBMISSION DATE

Jan 10, 2023 2:49 PM GMT+7

REPORT DATE

Jan 10, 2023 2:50 PM GMT+7

17% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

- 12% Internet database
- Crossref database
- 14% Submitted Works database
- 9% Publications database
- Crossref Posted Content database

Excluded from Similarity Report

- Bibliographic material
- Cited material
- Manually excluded text blocks
- Quoted material
- Small Matches (Less then 10 words)