Lab 2 Homework Report

Student ID: r13922135

Student Name: Yun-Han Wu / 吳昀翰

Methodology

- Preprocessing
  - Use emoji library to transform emojis to description text.
    - This way the meaning of emoji is preserved compared to just removing the emojis.
  - Replace abbreviations (Using online available resources, reference in the code.)
  - Remove stop words using built-in functionality in NTLK package.
  - Remove mentions, URLs and punctuation.

```
def clean_tweet_text(df, col):
        # Convert emojis to word
        df[col] = df[col].apply(emoji.demojize)

        # Replace underscores with spaces
        df[col] = df[col].str.replace('_', ' ')

        # Convert chat words to original text
        df[col] = df[col].apply(replace_chat_words)

        # Convert to lowercase
        df[col] = df[col].str.lower()

        # Convert abbreviations to word
        df[col] = df[col].apply(replace_abbreviations)

        # Remove the patterns
        for pattern in patterns:
                df[col] = df[col].str.replace(pattern, '', regex=True)

        # Remove stopwords
        stop_words = set(nltk.corpus.stopwords.words('english'))
        df[col] = df[col].apply(lambda x: ' '.join([word for word in x.split() if word not in stop_words]))
```

  - I have also tried to implement a new attribute using the emojis. (but fail to integrate it into the model)

- Embedding
  - I use tokenizer from Keras for embedding the text appended with hashtags.

```
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

train_df = pd.read_pickle("data/train_df.pkl")
test_df = pd.read_pickle("data/test_df.pkl")

# Tokenize text
max_words = 20000  # Maximum vocabulary size
max_len = 100  # Maximum sequence length

tokenizer = Tokenizer(num_words=max_words)
tokenizer.fit_on_texts(train_df["combined_text"])

train_sequences = tokenizer.texts_to_sequences(train_df["combined_text"])
train_padded = pad_sequences(train_sequences, maxlen=max_len, padding="post")

test_sequences = tokenizer.texts_to_sequences(test_df["combined_text"])
test_padded = pad_sequences(test_sequences, maxlen=max_len, padding="post")
```

  - I use one-hot encoding to deal with labels. (As in the lab)
- Model
  - I use a neural network to learn the pattern.

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense, Dropout, GlobalMaxPool1D

# Model parameters
embedding_dim = 128

# Build the model
model = Sequential([
        Embedding(input_dim=max_words, output_dim=embedding_dim, input_length=max_len),
        LSTM(128, return_sequences=True),
        GlobalMaxPool1D(),
        Dense(128, activation="relu"),
        Dropout(0.5),
        Dense(train_emotions.shape[1], activation="softmax")
])

# Compile the model
model.compile(optimizer="adam", loss="categorical_crossentropy", metrics=["accuracy"])
model.summary()
```

  - After some testing, I set it to run 5 epochs.
    (Tested with 10 epochs, result shows overfitting)

```
history = model.fit(
        train_padded, train_emotions,
        epochs=5,
        batch_size=32,
        validation_split=0.2
)
```

Result


```
Epoch 1/5
36390/36390 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 418s 11ms/step - accuracy: 0.5075 - loss: 1.3598 - val_accuracy: 0.5633 - val_loss: 1.1992
Epoch 2/5
36390/36390 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 440s 11ms/step - accuracy: 0.5743 - loss: 1.1775 - val_accuracy: 0.5677 - val_loss: 1.1839
Epoch 3/5
36390/36390 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 437s 11ms/step - accuracy: 0.5939 - loss: 1.1248 - val_accuracy: 0.5709 - val_loss: 1.1810
Epoch 4/5
36390/36390 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 440s 11ms/step - accuracy: 0.6094 - loss: 1.0821 - val_accuracy: 0.5702 - val_loss: 1.1947
Epoch 5/5
36390/36390 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 407s 11ms/step - accuracy: 0.6259 - loss: 1.0385 - val_accuracy: 0.5701 - val_loss: 1.2022
```

The model does not perform very well on classifying the tweets. It is probably because due to the approach I implemented, it has limited ability to learn the context.

Possible Solutions

Compared to the basic neural networks I implemented, more advanced approaches like transformers (Bert, GPT, etc.) can be used to learn contextual meaning more effectively.

More preprocessing techniques can also be used to better represent the original data.