Applying BERT Model on the Clift Dataset

The process of applying the BERT (Bidirectional Encoder Representations from Transformers) model to the Clift dataset for sentiment analysis. Sentiment analysis involves determining the sentiment or emotional tone of text data, which can be positive, negative, or neutral.

Data Preparation

We start by importing the necessary libraries and loading the Clift dataset. The dataset contains various columns, but for sentiment analysis, we focus on the text data and sentiment labels.

```
from sklearn.preprocessing import LabelEncoder
import pandas as pd
import nltk
from nltk.corpus import sentiwordnet as swn
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.optimizers import Adam
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize
from textblob import TextBlob
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
import numpy as np
from tensorflow.keras.utils import to_categorical
import seaborn as sns

# Download NLTK resources
nltk.download('punkt')
nltk.download('sentiwordnet')
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('averaged_perceptron_tagger')

# Load the dataset
df = pd.read_excel('/content/Clift Data.xlsx')

x = df.iloc[:, 3].values
y = df.iloc[:, 1].values

x = df.iloc[:, 3].astype(str)
```

Data Preprocessing

Text Preprocessing

To prepare the text data for sentiment analysis, we perform the following steps:

1. **Text Cleaning**: We remove any special characters, digits, and punctuation from the text. Additionally, we tokenize the text.
2. **Stopword Removal**: Common English stopwords are removed to focus on meaningful words.
3. **Lemmatization**: We lemmatize the words to reduce them to their base form.
4. **Sentiment Analysis**: We use TextBlob to perform sentiment analysis and assign sentiment scores and labels (positive, negative, neutral) to each text.

```
# Initialize stopwords, lemmatizer, and POS tagger
stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()

# Initialize TextBlob for sentiment analysis
def analyze_sentiment(text):
    blob = TextBlob(text)
    sentiment_score = blob.sentiment.polarity

    if sentiment_score > 0:
        sentiment_label = "Positive"
    elif sentiment_score < 0:
        sentiment_label = "Negative"
    else:
        sentiment_label = "Neutral"

    return sentiment_score, sentiment_label

def preprocess_text_with_pos(text):
    # Tokenization
    tokens = word_tokenize(text)

    # Removing stopwords and non-alphabetic tokens
    tokens = [token.lower() for token in tokens if isinstance(token, str)
and token.isalpha() and token.lower() not in stop_words]

    # POS tagging
    pos_tags = nltk.pos_tag(tokens)

    # Lemmatization using POS tags
    lemmatized_tokens = []
    for token, pos_tag in pos_tags:
```

```
        pos = 'n'  # Default POS tag for lemmatization is 'n' (noun)
        if pos_tag.startswith('J'):
            pos = 'a'  # Adjective
        elif pos_tag.startswith('V'):
            pos = 'v'  # Verb
        elif pos_tag.startswith('R'):
            pos = 'r'  # Adverb

        lemmatized_token = lemmatizer.lemmatize(token, pos)
        lemmatized_tokens.append(lemmatized_token)

    return lemmatized_tokens
```

Data Transformation

The preprocessed data is then transformed into a DataFrame for further analysis and saved to an Excel file.

```
# Initialize lists to store preprocessed data
preprocessed_data = []

# Process each text
for i, text in enumerate(x):
    # ...

# Create a DataFrame from the preprocessed data
preprocessed_df = pd.DataFrame(preprocessed_data)

# Save the preprocessed DataFrame to an Excel file
preprocessed_df.to_excel('Clift_Preprocessed.xlsx', index=False)

# Display the preprocessed data
print(preprocessed_df)
```

BERT Model Training

We continue by training a BERT-based model on the preprocessed text data.

```
# Convert each element in x to string
x = [str(text) for text in x]

# Tokenize the text using the BERT tokenizer
```

```
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased',
do_lower_case=True)
encoded_inputs = tokenizer(x, padding=True, truncation=True, max_length=20,
return_tensors='tf')

# Convert the labels to categorical format
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(y)
num_classes = len(label_encoder.classes_)
y = to_categorical(y, num_classes)

# Convert TensorFlow tensor to NumPy array
X = encoded_inputs['input_ids'].numpy()

# Reshape X to match the expected shape
X = X.reshape(X.shape[0], -1)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=2)

# Create the neural network model
model = Sequential()
model.add(Dense(units=64, activation='relu', input_dim=X_train.shape[1]))
model.add(Dropout(0.5))
model.add(Dense(units=num_classes, activation='softmax'))

# Compile the model
model.compile(optimizer=Adam(learning_rate=2e-5),
loss='categorical_crossentropy', metrics=['accuracy'])

# Train the model
history = model.fit(X_train, y_train, epochs=30, batch_size=32,
validation_data=(X_test, y_test))
```

Model Evaluation

After training the model, we evaluate its performance using various metrics such as accuracy, precision, recall, F1-score, and confusion matrix.

```
# Evaluate the model
y_pred = model.predict(X_test)
y_pred = y_pred.argmax(axis=1)
y_test = y_test.argmax(axis=1)
```

```
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')
support = len(y_test)

print("Test accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 score:", f1)
print("Support:", support)
```

Conclusion

In this document, we have applied the BERT model to the Clift dataset for sentiment analysis. We performed data preprocessing to clean and transform the text data and then trained a neural network model using BERT embeddings. The model was evaluated for its performance on sentiment classification, and various evaluation metrics were reported. This demonstrates the application of state-of-the-art models like BERT in real-world sentiment analysis tasks. Further fine-tuning and experimentation can lead to improved model performance.