Introduction to Natural Language Processing (NLP)

23CSAI05H

ID	Name
211392	Ahmed Sameh
211896	Abdulrahman Ayman
212071	Sara Anabtawi
206879	Ezz Yasser

Introduction:

In this project, we used the AG's news topic classification dataset to categorize news articles based on their descriptions using transformers in order to determine their category from the dataset's four categories, which are World, Sports, Business, and Sci/Tech [1]. From model 1 to model 4, we employed **DistilBERT** transformer, while in model 5 we employed **BERT** transformer.

Model 1:

```python
# Define the clean_text function to preprocess the text data
def clean_text(text):
    stemmer = PorterStemmer()  # Initialize the PorterStemmer object
    text = str(text).lower()  # Convert text to lowercase
    text = re.sub('\d+', '', text)  # Remove all numbers from the text
    text = re.sub('\[.*?\]', '', text)  # Remove HTML tags from the text
    text = re.sub('https?://\S+|www\.\S+', '', text)  # Remove URLs from the text
    text = re.sub(r'[%s]' % re.escape(string.punctuation), '', text)  # Remove all punctuation marks from the text
    # Tokenize text
    tokens = nltk.word_tokenize(text)
    # Stemming and stopwords removal
    stemmed_tokens = [stemmer.stem(word) for word in tokens if word not in stop_words]
    return ' '.join(stemmed_tokens)
```

The code in the image above defines a function called pure_text that is used to preprocess text data. Initially, it lowercases text and removes punctuation, HTML elements, numbers, and URLs. After that, the ending words are removed, and the text is converted into words. The word stem is then applied with a Stemmer to reduce the words to their base forms. The result is a labeled sequence of cleaned root words that is ready for further natural language processing. It does not contain stop words or extra letters.

```python
# Encode the labels
label_encoder = LabelEncoder()
train['Class Index'] = label_encoder.fit_transform(train['Class Index'])
test['Class Index'] = label_encoder.transform(test['Class Index'])
```

We used LabelEncoder to convert non-numeric labels to numeric format. The sensor is first installed in the "Class Index" column of the training data (train ['Class Index']), where it learns unique identifiers. It then converts the identifiers in both the training and test datasets to numeric values.

```
# Initialize the DistilBERT tokenizer
tokenizer = DistilBertTokenizer.from_pretrained('distilbert-base-uncased')
```

In the DistilBERT model, we used the tokenizer from the pre-trained "distillbert-base-uncased" configuration. This tokenizer prepares text data for processing by converting it to a format suitable for the model, focusing on lowercase letters.

```
# Define a function to encode texts into a format suitable for the model training
def encode_texts(tokenizer, texts, max_len=256):
    input_ids = []
    attention_masks = []
    for text in texts:
        encoded = tokenizer.encode_plus(
            text,
            add_special_tokens=True,
            max_length=max_len,
            truncation=True,
            padding='max_length',
            return_attention_mask=True,
            return_tensors='tf',
        )
        # Ensure the tensors are squeezed to remove unnecessary dimensions
        input_ids.append(tf.squeeze(encoded['input_ids']))
        attention_masks.append(tf.squeeze(encoded['attention_mask']))
    return np.array(input_ids), np.array(attention_masks)
```

In the code image above, we initialized the encode_texts function, which converts a list of strings into a format that can be used to train a model using the tokenizer. It processes each text by encoding it to contain special identifiers, trimming or padding it to a specified maximum length, and creating attention masks. These attention masks help the model focus on relevant parts of the input sequence. The function returns input identifiers and attention masks suitable for model training using TensorFlow.

```
# Encode the text data
X_train_ids, X_train_masks = encode_texts(tokenizer, train['Description'].values, max_len=512)
X_test_ids, X_test_masks = encode_texts(tokenizer, test['Description'].values, max_len=512)
```

In the code image above, we use the encode_texts function to process the text descriptions in the "Description" column of the test and train datasets. This description specifies a maximum sequence length of 512 and makes it a suitable format for model training. This function produces two variables: X_train_ids and X_train_masks for the training data, and X_test_ids and X_test_masks for the test data. These variables contain labeled text data and the corresponding mask, which is important for model training and evaluation.

```
# Convert labels to one-hot encoding
y_train = to_categorical(train['Class Index'].values)
y_test = to_categorical(test['Class Index'].values)
```

In the code image above, we converted the test and train datasets to single-bit encoding sequences using ' to_categorical function using the "index" column class. This function converts integer class labels to an array of binary classes. The variables y_train and y_test store the unique encoding labels for the training and test sets.

```
model = TFDistilBertForSequenceClassification.from_pretrained('distilbert-base-uncased', num_labels=len(label_encoder.classes_))
```

In the code image above, we use the DistilBERT model for sequence classification from the Hugging Face transformer library using the pre-trained "distillbert-base-uncased" . Configure the model to process the number of unique labels specified by the label_encoder.classes_ length, which can generate the correct number of class probabilities for classification .

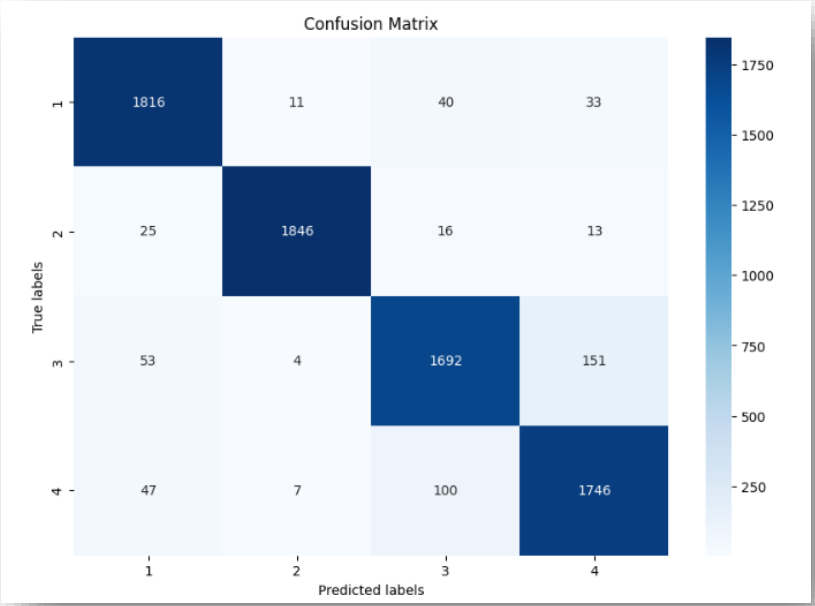
```
# Prepare the model for training
optimizer = tf.keras.optimizers.Adam(learning_rate=2e-5)
loss = tf.keras.losses.CategoricalCrossentropy(from_logits=True)
model.compile(optimizer=optimizer, loss=loss, metrics=['accuracy'])
```

In the code image above, we configured the DistilBERT model for training by setting the Adam optimizer with a learning rate of 2×10^-5, using categorical crossentropy for the loss function, and monitoring accuracy as the performance metric.

```
# Training the model
model.fit([X_train_ids, X_train_masks], y_train, epochs=1, batch_size=16,
          validation_data=([X_test_ids, X_test_masks], y_test))
```

In the code image above, we train a DistilBERT model on the training data. We use input and masking as functions and one-hot encoding class labels as targets. The model was trained in one epoch with a batch size of 16.

Confusion matrix for model 1:



Classification report for model 1:

	precision	recall	f1-score	support
1	0.94	0.96	0.95	1900
2	0.99	0.97	0.98	1900
3	0.92	0.89	0.90	1900
4	0.90	0.92	0.91	1900
accuracy			0.93	7600
macro avg	0.93	0.93	0.93	7600
weighted avg	0.93	0.93	0.93	7600

Training and validation accuracy for model 1:

```
7500/7500 [==============================] - 6706s 891ms/step - loss: 0.2344 - accuracy: 0.9204 - val
_loss: 0.1920 - val_accuracy: 0.9342
```

Prediction based on description in model 1:

```
#the numeric labels refers to -------> 1-World, 2-Sports, 3-Business, 4-Sci/Tech
would buy AT amp;T Wireless Services Inc. #39;s (AWE.
Predicted Category: 3
True Category: 3

Description: MEXICO CITY - Sebastien Bourdais took his first Champ Car World Series title, beating
teammate Bruno Junqueira with a flag-to-flag win Sunday in the Mexican Grand Prix.
Predicted Category: 2
True Category: 2

Description: Petter Solberg demonstrated his winning potential aboard his Subaru Impreza WRC2004 t
oday to take three stage wins and end Leg one in second position overall.
Predicted Category: 2
True Category: 2

Description: Companies across the country are offering yoga and meditation classes to help employe
es relax, reduce stress and recharge.
Predicted Category: 3
True Category: 4
```

Model 2:

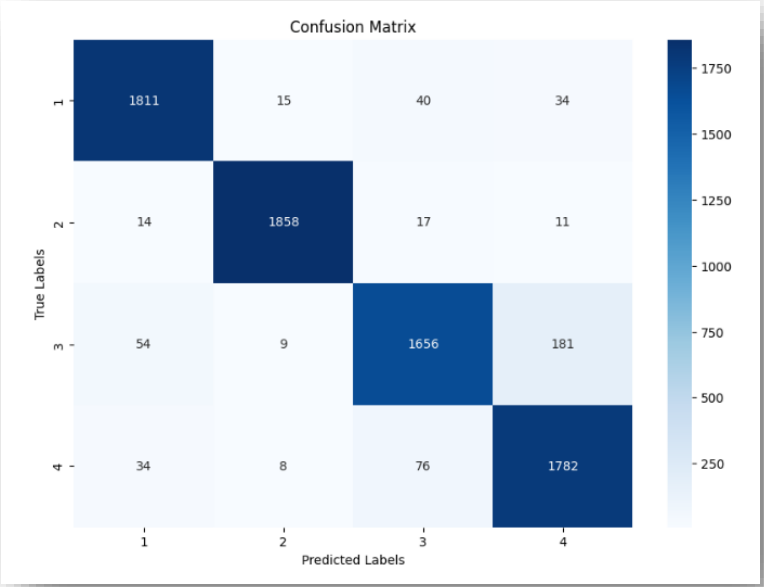
We Modified the Learning Rate from 2e-5 into 5e-5:

```
# Prepare the model for training
optimizer = tf.keras.optimizers.Adam(learning_rate=5e-5)
loss = tf.keras.losses.CategoricalCrossentropy(from_logits=True)
model.compile(optimizer=optimizer, loss=loss, metrics=['accuracy'])
```

Then We Modified the Batch Size of model 1 which was 16 to 28 in order to try different variations and compare their evaluation metrics:

```
# Training the model
model.fit([X_train_ids, X_train_masks], y_train, epochs=1, batch_size=28,
          validation_data=([X_test_ids, X_test_masks], y_test))
```

Confusion Matrix of Model 2:



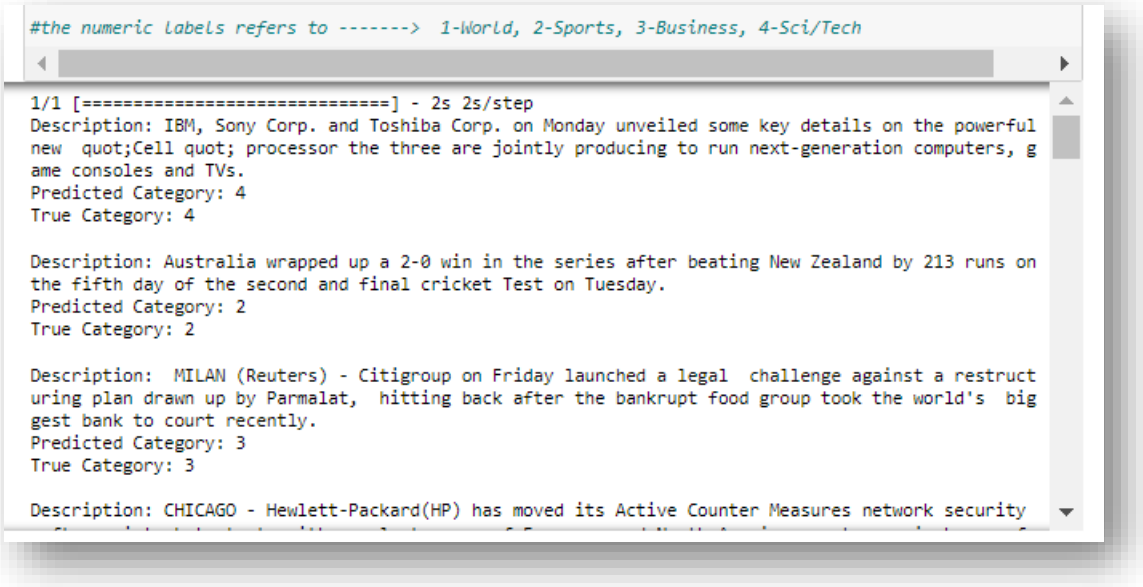
Classification Report of Model 2

	precision	recall	f1-score	support
1	0.95	0.95	0.95	1900
2	0.98	0.98	0.98	1900
3	0.93	0.87	0.90	1900
4	0.89	0.94	0.91	1900
accuracy			0.94	7600
macro avg	0.94	0.94	0.94	7600
weighted avg	0.94	0.94	0.94	7600

The Training Results of Model 2 after 1 Epoch

```
4286/4286 [==============================] - 6598s 2s/step - loss: 0.2337 - accuracy: 0.9198 - val_lo
ss: 0.1981 - val_accuracy: 0.9351
```

A Sample of Model 2 Predictions:



```
#the numeric labels refers to -------> 1-World, 2-Sports, 3-Business, 4-Sci/Tech

1/1 [==============================] - 2s 2s/step
Description: IBM, Sony Corp. and Toshiba Corp. on Monday unveiled some key details on the powerful
new  quot;Cell quot; processor the three are jointly producing to run next-generation computers, g
ame consoles and TVs.
Predicted Category: 4
True Category: 4

Description: Australia wrapped up a 2-0 win in the series after beating New Zealand by 213 runs on
the fifth day of the second and final cricket Test on Tuesday.
Predicted Category: 2
True Category: 2

Description:  MILAN (Reuters) - Citigroup on Friday launched a legal  challenge against a restruct
uring plan drawn up by Parmalat,  hitting back after the bankrupt food group took the world's  big
gest bank to court recently.
Predicted Category: 3
True Category: 3

Description: CHICAGO - Hewlett-Packard(HP) has moved its Active Counter Measures network security
```

Model 3:

We Modified here the maximum length of the token sequences (max_len) to 256 instead of 512 in both:

```
# Encode the text data
X_train_ids, X_train_masks = encode_texts(tokenizer, train['Description'].values, max_len=256)
X_test_ids, X_test_masks = encode_texts(tokenizer, test['Description'].values, max_len=256)
```

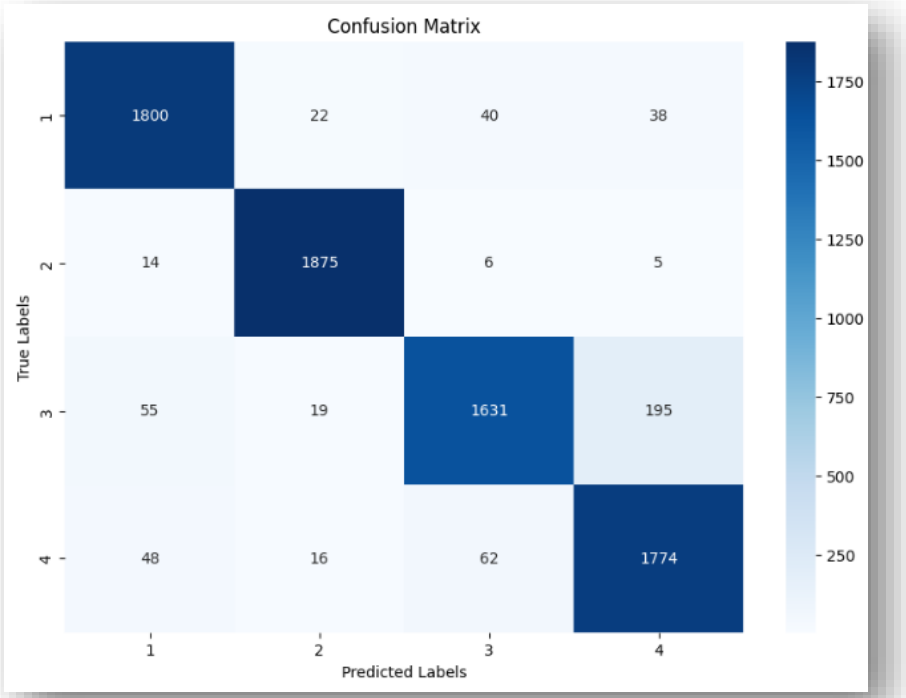
Then we Modified the Learning Rate to 1e-5:

```
# Prepare the model for training
optimizer = tf.keras.optimizers.Adam(learning_rate=1e-5)
loss = tf.keras.losses.CategoricalCrossentropy(from_logits=True)
model.compile(optimizer=optimizer, loss=loss, metrics=['accuracy'])
```

Then We Modified the Batch Size to 32 in order to try different variations and compare their evaluation metrics:

```
# Training the model
model.fit([X_train_ids, X_train_masks], y_train, epochs=1, batch_size=32,
          validation_data=([X_test_ids, X_test_masks], y_test))
```

Confusion Matrix of Model 3:



Classification Report of Model 3

	precision	recall	f1-score	support
1	0.94	0.95	0.94	1900
2	0.97	0.99	0.98	1900
3	0.94	0.86	0.90	1900
4	0.88	0.93	0.91	1900
accuracy			0.93	7600
macro avg	0.93	0.93	0.93	7600
weighted avg	0.93	0.93	0.93	7600

The Training Results of Model 3 after 1 Epoch:

```
3750/3750 [==============================] - 2989s 791ms/step - loss: 0.2576 - accuracy: 0.9135 - val
_loss: 0.1970 - val_accuracy: 0.9316
```

A Sample of the Predictions of Model 3

```
#the numeric labels refers to ------->  1-World, 2-Sports, 3-Business, 4-Sci/Tech
```

```
1/1 [==============================] - 0s 335ms/step
Description:  NEW YORK (Reuters) - Erubiel Durazo's three-run homer in  the second inning helped t
he Oakland Athletics remain top of  the American League (AL) West with a 9-4 win over the reeling
Baltimore Orioles Thursday.
Predicted Category: 2
True Category: 2

Description: United Airlines, trying to further pare costs so it can emerge from bankruptcy, said
Thursday it is seeking about \$725 million in annual savings through proposed pay
Predicted Category: 3
True Category: 3

Description: Microsoft has warned of seven newly found flaws in its software that could allow an a
ttacker to steal data and take over a personal computer running the Windows operating system.
Predicted Category: 4
True Category: 4
```

Model 4:

We Modified here the maximum length of the token sequences (max_len) to 128 instead of 512 in both:

```
# Encode the text data
X_train_ids, X_train_masks = encode_texts(tokenizer, train['Description'].values, max_len=128)
X_test_ids, X_test_masks = encode_texts(tokenizer, test['Description'].values, max_len=128)
```

Then we Modified the Learning Rate to 3e-5:

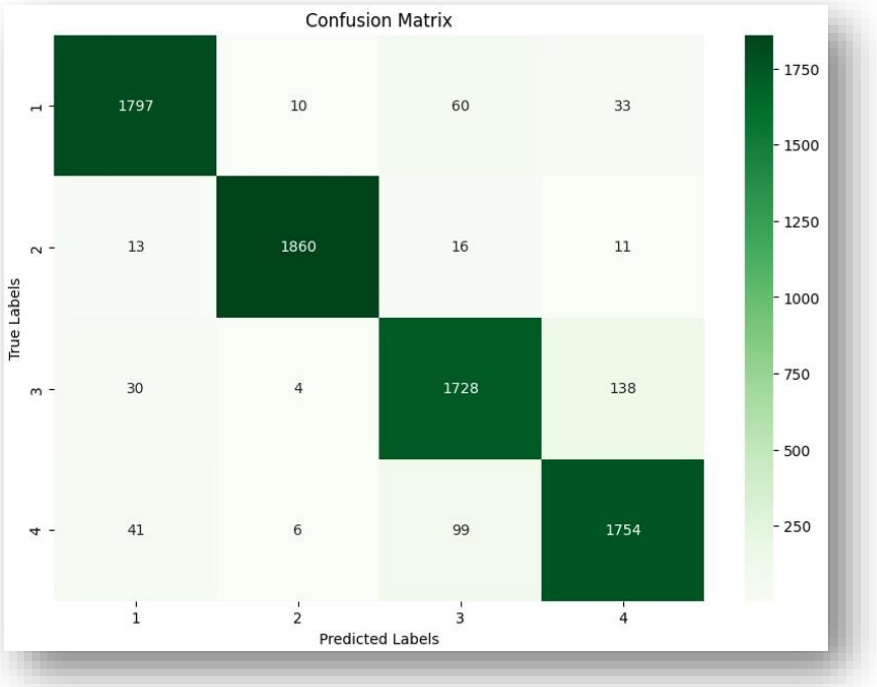
```
# Prepare the model for training
optimizer = tf.keras.optimizers.Adam(learning_rate=3e-5)
loss = tf.keras.losses.CategoricalCrossentropy(from_logits=True)
model.compile(optimizer=optimizer, loss=loss, metrics=['accuracy'])
```

Then We Modified the Batch Size to 64 to try different variations and compare their evaluation metrics as well as the epochs with 3. The Training Results of Model 4 after 3 Epoch:

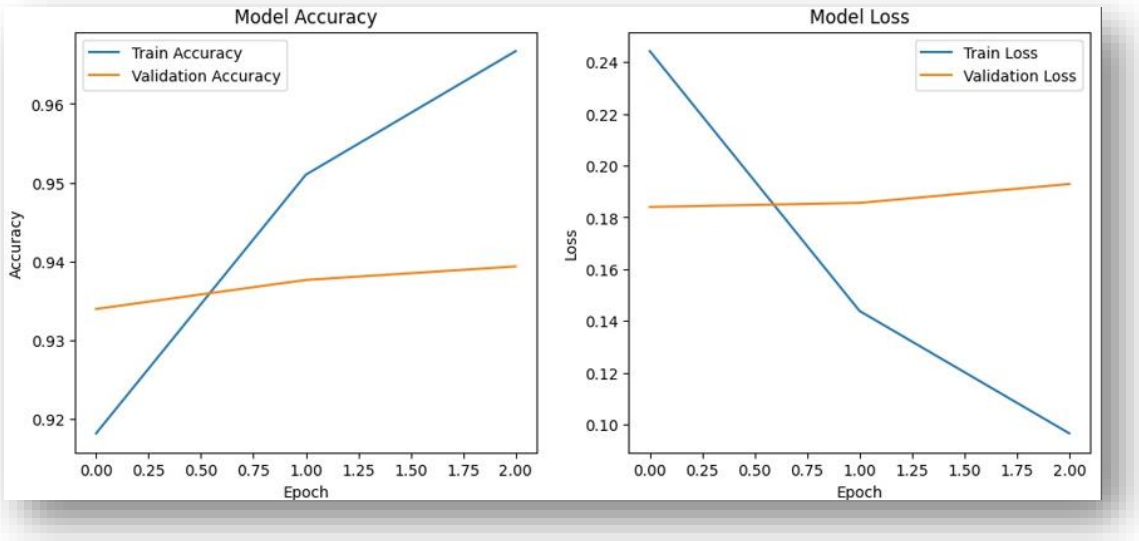
```
# Training the model
history = model.fit(
    [X_train_ids, X_train_masks],
    y_train,
    epochs=3,
    batch_size=64,
    validation_data=([X_test_ids, X_test_masks], y_test),
    callbacks=[tensorboard_callback]  # Add the TensorBoard callback here
)
```

```
Epoch 1/3
1875/1875 [==============================] - 1447s 761ms/step - loss: 0.2442 - accuracy: 0.9181 - val_loss: 0.1840 - val_accuracy: 0.9339
Epoch 2/3
1875/1875 [==============================] - 1422s 759ms/step - loss: 0.1438 - accuracy: 0.9510 - val_loss: 0.1856 - val_accuracy: 0.9376
Epoch 3/3
1875/1875 [==============================] - 1412s 753ms/step - loss: 0.0966 - accuracy: 0.9667 - val_loss: 0.1929 - val_accuracy: 0.9393
```

Confusion Matrix Of model 4:



Learning curve of model 4:



Classification report of model 4:

```
Classification Report:
              precision    recall  f1-score   support

           1       0.96      0.95      0.95      1900
           2       0.99      0.98      0.98      1900
           3       0.91      0.91      0.91      1900
           4       0.91      0.92      0.91      1900

    accuracy                           0.94      7600
   macro avg       0.94      0.94      0.94      7600
weighted avg       0.94      0.94      0.94      7600
```

Predictions of model 4:

```
#the numeric labels refers to -------> 1-World, 2-Sports, 3-Business, 4-Sci/Tech
Predicted Category: 1
True Category: 1

Description: The Browns started the season on a good note for the first time since 1994, and the w
in buoys the teams hopes for the near future.
Predicted Category: 2
True Category: 2

Description: Intel is preparing a marketing strategy that will brand desktop PCs with a similar la
bel that made its Centrino notebook technology a household name, according to sources familiar wit
h the company's plans.
Predicted Category: 4
True Category: 3

Description: A car bomb exploded outside the Education Ministry in central Baghdad Tuesday, killin
g at least six people and wounding about eight, the Interior Ministry said.
Predicted Category: 1
True Category: 1
```

```
238/238 [==============================] - 32s 133ms/step
Number of correct predictions: 7139
Number of incorrect predictions: 461
```

Model 5:

Display the First few rows of the training dataset:

```
# Display the first few rows of the training dataset
df_train.head()
```

	Class Index	Title	Description
0	3	Wall St. Bears Claw Back Into the Black (Reuters)	Reuters - Short-sellers, Wall Street's dwindli...
1	3	Carlyle Looks Toward Commercial Aerospace (Reu...	Reuters - Private investment firm Carlyle Grou...
2	3	Oil and Economy Cloud Stocks' Outlook (Reuters)	Reuters - Soaring crude prices plus worries\ab...
3	3	Iraq Halts Oil Exports from Main Southern Pipe...	Reuters - Authorities have halted oil export\f...
4	3	Oil prices soar to all-time record, posing new...	AFP - Tearaway world oil prices, toppling reco...

Define a dictionary mapping Numerical Labels to text categories:

```
[ ] # Define a dictionary mapping numerical labels to text categories
    TEXT_LABELS = {1: "World", 2: "Sports", 3: "Business", 4: "Sci/Tech"}
```

Define a Function to Combine the title and Description into a single column:

```
[ ] # Define a function to combine title and description into a single text column
    def combine_title_and_description(df):
        # Returns a dataset with the title and description fields combined
        df['text'] = df[['Title', 'Description']].agg('. '.join, axis=1)
        df = df.drop(['Title', 'Description'], axis=1)
        return df
```

Apply the Function to Training and Testing Dataset:

```
[ ] # Apply the function to training and test datasets
    df_train = combine_title_and_description(df_train)
    df_test = combine_title_and_description(df_test)
    df_train.head()
```

	Class Index	text
0	3	Wall St. Bears Claw Back Into the Black (Reute...
1	3	Carlyle Looks Toward Commercial Aerospace (Reu...
2	3	Oil and Economy Cloud Stocks' Outlook (Reuters...
3	3	Iraq Halts Oil Exports from Main Southern Pipe...
4	3	Oil prices soar to all-time record, posing new...

Display the Head of the Test Data frame:

```
[ ] df_test.head()
```

	Class Index	text
0	3	Fears for T N pension after talks. Unions repr...
1	4	The Race is On: Second Private Team Sets Launc...
2	4	Ky. Company Wins Grant to Study Peptides (AP)....
3	4	Prediction Unit Helps Forecast Wildfires (AP)....
4	4	Calif. Aims to Limit Farm-Related Smog (AP). A...

Print the Shape of the training and testing datasets:

```
[ ] # Print the shape of the training and test datasets
    print('Shape of the training data: ', df_train.shape)
    print('Shape of the test data: ', df_test.shape)

    Shape of the training data:  (120000, 2)
    Shape of the test data:  (7600, 2)
```

Display the ‘Text’ Column of the Training Dataset:

```
[ ]  # Display the 'text' column of the training dataset
     df_train['text']
```

```
0         Wall St. Bears Claw Back Into the Black (Reute...
1         Carlyle Looks Toward Commercial Aerospace (Reu...
2         Oil and Economy Cloud Stocks' Outlook (Reuters...
3         Iraq Halts Oil Exports from Main Southern Pipe...
4         Oil prices soar to all-time record, posing new...
                              ...
119995    Pakistan's Musharraf Says Won't Quit as Army C...
119996    Renteria signing a top-shelf deal. Red Sox gen...
119997    Saban not going to Dolphins yet. The Miami Dol...
119998    Today's NFL games. PITTSBURGH at NY GIANTS Tim...
119999    Nets get Carter from Raptors. INDIANAPOLIS -- ...
Name: text, Length: 120000, dtype: object
```

```
[ ]  #  setup and remove stopwords for text processing
     import nltk
     from nltk.corpus import stopwords
     from nltk.tokenize import word_tokenize
     import multiprocessing as mp

     nltk.download('stopwords')
     stop_words = set(stopwords.words('english'))

     def remove_stopwords(text):
         words = text.split()
         words = [word for word in words if word.lower() not in stopwords.words('english')]
         return " ".join(words)
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\Administrator\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

In the code above in the image, the remove_stopwords function takes the input of words and extracts only the English stop words. Break the text into words, analyze the endings using the NLTK list of English endings, and then join the remaining words into strings. This helps reduce noise and improve the quality of text data for further analysis or processing.

Apply The Function to Remove Stop words:

```
[ ]  # Apply the function to remove stopwords from the 'text' column in train dataset
     df_train['clean_text'] = df_train['text'].apply(remove_stopwords)

[ ]  # Apply the function to remove stopwords from the 'text' column in test dataset
     df_test['clean_text']=df_test['text'].apply(remove_stopwords)
```

```
[ ]  X_train = df_train['text']

     # Initialize LabelEncoder and encode labels into one-hot vectors for classification
     le = LabelEncoder()
     y_train = df_train['Class Index']
     y_train = le.fit_transform(y_train)
     y_train = to_categorical(y_train, num_classes=4)


     X_test = df_test['text']
     y_test = df_test['Class Index']
     y_test = le.transform(y_test)
     y_test = to_categorical(y_test, num_classes=4)
```

This code in the image above uses LabelEncoder to encode point labels in numeric format and to_categorical to convert to a single encoding set for the training and test datasets.

```
[ ]  # Define a text input layer for a TensorFlow model
     text_input = tf.keras.layers.Input(shape=(), dtype=tf.string)
     # Define a preprocessor layer from TensorFlow Hub for BERT model
     preprocessor = hub.KerasLayer(
         "https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3")
     encoder_inputs = preprocessor(text_input)
     # Define a BERT encoder layer, set as trainable
     encoder = hub.KerasLayer(
         "https://tfhub.dev/tensorflow/bert_en_uncased_L-12_H-768_A-12/4",
         trainable=True)
     outputs = encoder(encoder_inputs)
     # Extract pooled and sequence outputs from BERT
     pooled_output = outputs["pooled_output"]      # [batch_size, 768].
     sequence_output = outputs["sequence_output"]  # [batch_size, seq_length, 768].
```

This code above in the image defines a text input layer for the TensorFlow model, specifying the input image as an empty tensor and the data type as a string. We built a front-end layer using TensorFlow Hub for BERT models that process text input. The newly defined BERT encoder layer, provided by TensorFlow Hub, is configured to be trained. The code extracts the hierarchical output of the BERT model, which can be used as part of further operations such as classification or regression.

```
[ ]  hub_inputs = preprocessor(['ID for each word, with zero padding at the end.'])
     {key: value[0, :25].numpy() for key, value in hub_inputs.items()}

     {'input_word_ids': array([  101,  8909,  2005,  2169,  2773,  1010,  2007,  5717, 11687,
              4667,  2012,  1996,  2203,  1012,   102,     0,     0,     0,
                 0,     0,     0,     0,     0,     0,     0]),
      'input_mask': array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0,
             0, 0, 0]),
      'input_type_ids': array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
             0, 0, 0])}
```

The code in the Image above uses a preprocessing layer to process the input text and extract the first 25 elements of each processed feature. A dictionary understanding is created to store the truncated function values as NumPy arrays.

```
[ ]  # Process the inputs through BERT to get outputs
     result = encoder(
         inputs=hub_inputs,
         training=False,
     )

     print("Pooled output shape:", result['pooled_output'].shape)
     print("Sequence output shape:", result['sequence_output'].shape)

     Pooled output shape: (1, 768)
     Sequence output shape: (1, 128, 768)
```

The code in the image above processes the inputs through the BERT encoder to obtain outputs. It utilizes the previously defined encoder layer and passes the hub_inputs dictionary containing preprocessed inputs.

```
# Configuration for model training
epochs = 3
batch_size = 32
eval_batch_size = 32
# Calculate the number of steps per epoch and warmup steps
train_data_size = df_train.shape[0]
steps_per_epoch = int(train_data_size / batch_size)
num_train_steps = steps_per_epoch * epochs
num_warmup_steps = int(0.1*num_train_steps)
# Initialize learning rate and setup optimizer with warmup phase
init_lr = 3e-5
optimizer = optimization.create_optimizer(init_lr=init_lr,
                                          num_train_steps=num_train_steps,
                                          num_warmup_steps=num_warmup_steps,
                                          optimizer_type='adamw')
```

This code above in the image prepares for training by calculating the number of training phases and warm-up phases based on the dataset, periods and set size, which is 3. It resets the learning and directs the warm-up phase of the optimizer using. the AdamW optimizer in the optimization module. This setting helps to optimize the learning process by gradually increasing the learning during the warm-up phase before stabilizing it during the main training phase.

```
[ ]  # Function to build the TensorFlow model with BERT encoder
     def build_model(num_classes, optimizer, max_len=512):
         # Define the input layer for text data
         text_input = tf.keras.layers.Input(shape=(), dtype=tf.string, name='text')
          # Add a preprocessing layer to handle text transformation for BERT
         preprocessing_layer = hub.KerasLayer("https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3", name='preprocessing')
         encoder_inputs = preprocessing_layer(text_input)
          # Add the BERT encoder layer from TensorFlow Hub
         encoder = hub.KerasLayer("https://tfhub.dev/tensorflow/bert_en_uncased_L-12_H-768_A-12/4", trainable=True, name='BERT_encoder')
         outputs = encoder(encoder_inputs)
         # Extract pooled output for classification tasks
         net = outputs['pooled_output']
         net = tf.keras.layers.Dropout(0.1)(net)
         net = tf.keras.layers.Dense(64, activation='relu')(net)
         net = tf.keras.layers.Dropout(0.1)(net)
         out = tf.keras.layers.Dense(num_classes, activation='softmax', name='classifier')(net)

         # Create and compile the model
         model = tf.keras.models.Model(text_input, out)
         model.compile(optimizer, loss='categorical_crossentropy', metrics=['accuracy'])

         return model
```

The code in the image above This function creates a TensorFlow model with a BERT encoder for text classification tasks. It requires parameters including the number of classes, the optimizer, and an optional maximum sequence length. It defines a text input layer, adds a preprocessing layer to convert text for BERT, and includes the TensorFlow Hub BERT encoder. It then separates the combined output for classification, applies final control, and adds density layers for classification. Finally, it creates and builds the model with the specified optimizer, loss function and metric before returning it.

```
[ ]  # Build and summarize the model
     model = build_model(num_classes=4, optimizer=optimizer)
```

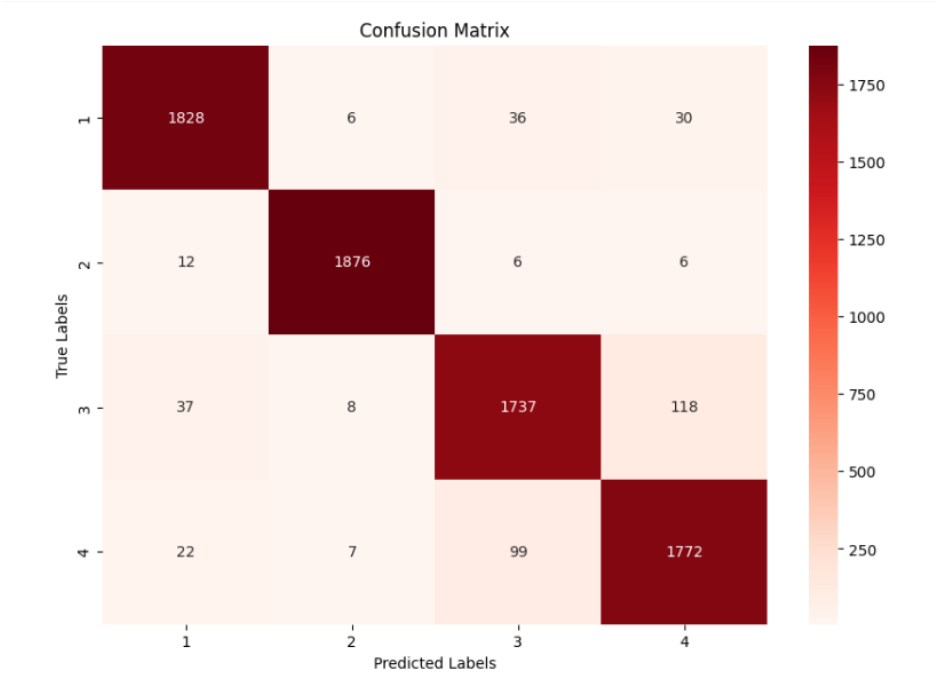
```
model.summary()
```

```
Model: "model"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 text (InputLayer)              [(None,)]            0           []

 preprocessing (KerasLayer)     {'input_word_ids':   0           ['text[0][0]']
                                (None, 128),
                                 'input_mask': (Non
                                e, 128),
                                 'input_type_ids':
                                (None, 128)}

 BERT_encoder (KerasLayer)      {'encoder_outputs':  109482241   ['preprocessing[0][0]',
                                 [(None, 128, 768),               'preprocessing[0][1]',
                                 (None, 128, 768),                'preprocessing[0][2]']
                                 (None, 128, 768),
                                 (None, 128, 768),
                                 (None, 128, 768),
                                 (None, 128, 768),
                                 (None, 128, 768),
                                 (None, 128, 768),
                                 (None, 128, 768),
                                 (None, 128, 768),
                                 (None, 128, 768),
                                 (None, 128, 768)],
                                 'default': (None,
                                768),
                                 'pooled_output': (
                                None, 768),
                                 'sequence_output':
                                 (None, 128, 768)}

 dropout (Dropout)              (None, 768)          0           ['BERT_encoder[0][13]']

 dense (Dense)                  (None, 64)           49216       ['dropout[0][0]']

 dropout_1 (Dropout)            (None, 64)           0           ['dense[0][0]']

 classifier (Dense)             (None, 4)            260         ['dropout_1[0][0]']

==================================================================================================
Total params: 109,531,717
Trainable params: 109,531,716
Non-trainable params: 1
```

Training and validation accuracy for model 5:

```
Epoch 1/3
3750/3750 [==============================] - ETA: 0s - loss: 0.2886 - accuracy: 0.8999
Epoch 1: val_accuracy improved from -inf to 0.94276, saving model to model.h5
3750/3750 [==============================] - 41663s 11s/step - loss: 0.2886 - accuracy: 0.8999 - val_loss: 0.1732 - val_accurac
y: 0.9428
Epoch 2/3
3750/3750 [==============================] - ETA: 0s - loss: 0.1355 - accuracy: 0.9570
Epoch 2: val_accuracy improved from 0.94276 to 0.94895, saving model to model.h5
3750/3750 [==============================] - 43062s 11s/step - loss: 0.1355 - accuracy: 0.9570 - val_loss: 0.1624 - val_accurac
y: 0.9489
Epoch 3/3
3750/3750 [==============================] - ETA: 0s - loss: 0.0838 - accuracy: 0.9746
Epoch 3: val_accuracy improved from 0.94895 to 0.94908, saving model to model.h5
3750/3750 [==============================] - 43624s 12s/step - loss: 0.0838 - accuracy: 0.9746 - val_loss: 0.2000 - val_accurac
y: 0.9491
```

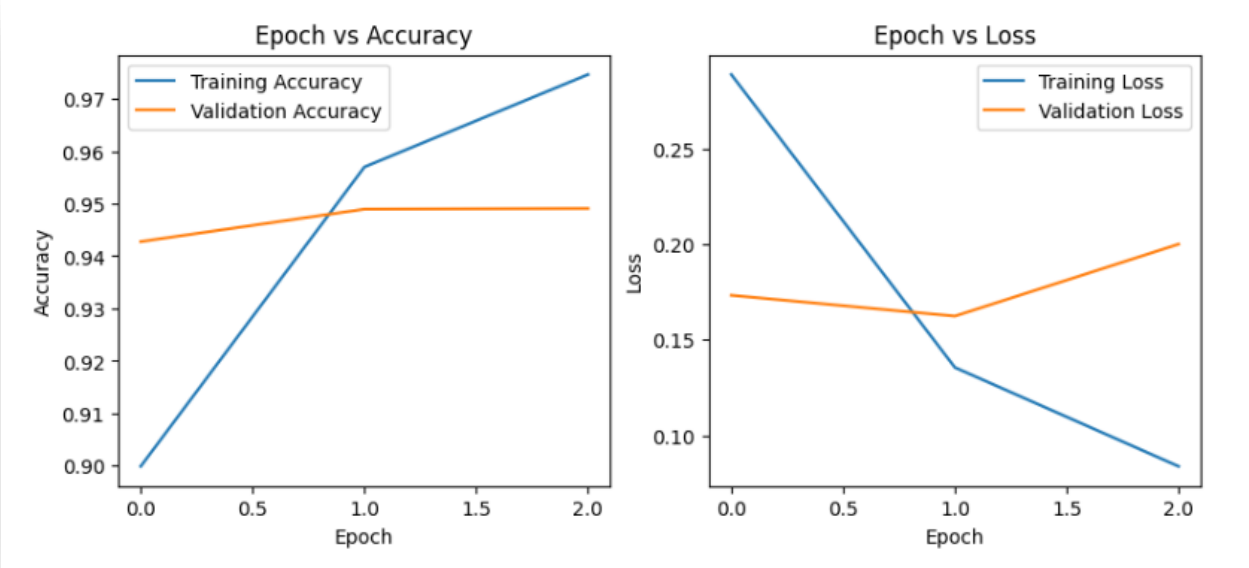
Confusion matrix for model 5:



Classification report for model 5:

```
Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.96      0.96      1900
           1       0.99      0.99      0.99      1900
           2       0.92      0.91      0.92      1900
           3       0.92      0.93      0.93      1900

    accuracy                           0.95      7600
   macro avg       0.95      0.95      0.95      7600
weighted avg       0.95      0.95      0.95      7600
```

Learning curves for model 5:



```python
import tensorflow as tf

# Predict on the entire test dataset
test_pred_logits = model.predict(df_test['text'])
test_pred = tf.argmax(test_pred_logits, axis=1)

# Extract the true labels for the test dataset
y_test_all = le.transform(df_test['Class Index'])
y_test_all_cat = tf.keras.utils.to_categorical(y_test_all, num_classes=4)
test_true_labels = tf.argmax(y_test_all_cat, axis=1)

# Convert predictions and true labels to category names
predictions = le.inverse_transform(test_pred.numpy())
true_labels = le.inverse_transform(test_true_labels.numpy())

# Count correct and wrong predictions
correct_count = np.sum(test_pred == test_true_labels)
wrong_count = len(df_test) - correct_count

# Print the number of correct and incorrect predictions
print(f"Number of correct predictions: {correct_count}")
print(f"Number of incorrect predictions: {wrong_count}")
```

The code in the image above predicts labels for the entire test dataset and calculates the number of correct and incorrect predictions. It prints these counts and optionally provides details of each prediction.

Predictions for model 5:

```
238/238 [==============================] - 982s 4s/step
Number of correct predictions: 7213
Number of incorrect predictions: 387
```

```
Text: Fears for T N pension after talks. Unions representing workers at Turner   Newall say they are 'disappointed' after talks with stricken parent firm Federal Mogul....
Predicted Category: 3
True Category: 3

Text: The Race is On: Second Private Team Sets Launch Date for Human Spaceflight (SPACE.com). SPACE.com - TORONTO, Canada -- A second\team of rocketeers competing for the  #36;10 million Ansari X Prize, a c...
Predicted Category: 4
True Category: 4

Text: Ky. Company Wins Grant to Study Peptides (AP). AP - A company founded by a chemistry researcher at the University of Louisville won a grant to develop a method of producing better peptides, which are ...
Predicted Category: 4
True Category: 4

Text: Prediction Unit Helps Forecast Wildfires (AP). AP - It's barely dawn when Mike Fitzpatrick starts his shift with a blur of colorful maps, figures and endless charts, but already he knows what the day ...
Predicted Category: 4
True Category: 4

Text: Calif. Aims to Limit Farm-Related Smog (AP). AP - Southern California's smog-fighting agency went after emissions of the bovine variety Friday, adopting the nation's first rules to reduce air pollutio...
Predicted Category: 4
True Category: 4

Text: Open Letter Against British Copyright Indoctrination in Schools. The British Department for Education and Skills (DfES) recently launched a "Music Manifesto" campaign, with the ostensible intention of...
Predicted Category: 4
True Category: 4

Text: Loosing the War on Terrorism. \\"Sven Jaschan, self-confessed author of the Netsky and Sasser viruses, is\responsible for 70 percent of virus infections in 2004, according to a six-month\virus roundup...
Predicted Category: 4
True Category: 4

Text: FOAFKey: FOAF, PGP, Key Distribution, and Bloom Filters. \\FOAF/LOAF  and bloom filters have a lot of interesting properties for social\network and whitelist distribution.\\I think we can go one level...
Predicted Category: 4
True Category: 4

Text: E-mail scam targets police chief. Wiltshire Police warns about "phishing" after its fraud squad chief was targeted....
Predicted Category: 4
True Category: 4

Text: Card fraud unit nets 36,000 cards. In its first two years, the UK's dedicated card fraud unit, has recovered 36,000 stolen cards and 171 arrests - and estimates it saved 65m....
Predicted Category: 4
True Category: 4

Text: Group to Propose New High-Speed Wireless Format.  LOS ANGELES (Reuters) - A group of technology companies  including Texas Instruments Inc. &lt;TXN.N&gt;, STMicroelectronics  &lt;STM.PA&gt; and Broadc...
Predicted Category: 4
True Category: 4
```

Trail 1(Albert model):

Encoding:

```
# Initialize the ALBERT tokenizer
tokenizer = AlbertTokenizer.from_pretrained('albert-base-v2')
```

```
Downloading:   0%|          | 0.00/742k [00:00<?, ?B/s]

Downloading:   0%|          | 0.00/25.0 [00:00<?, ?B/s]

Downloading:   0%|          | 0.00/684 [00:00<?, ?B/s]
```

```
# Define a function to encode texts into a format suitable for the model training
def encode_texts(tokenizer, texts, max_len=256):
    input_ids = []
    attention_masks = []
    for text in texts:
        encoded = tokenizer.encode_plus(
            text,
            add_special_tokens=True,
            max_length=max_len,
            truncation=True,
            padding='max_length',
            return_attention_mask=True,
            return_tensors='tf',
        )
        # Ensure the tensors are squeezed to remove unnecessary dimensions
        input_ids.append(tf.squeeze(encoded['input_ids']))
        attention_masks.append(tf.squeeze(encoded['attention_mask']))
    return np.array(input_ids), np.array(attention_masks)
```

```
# Encode the text data
X_train_ids, X_train_masks = encode_texts(tokenizer, train['Description'].values, max_len=512)
X_test_ids, X_test_masks = encode_texts(tokenizer, test['Description'].values, max_len=512)
```

Preparing the model:

```
# Load the ALBERT model
model = TFAlbertForSequenceClassification.from_pretrained('albert-base-v2', num_labels=len(label_encoder.classes_))
```

```
Downloading:   0%|          | 0.00/60.1M [00:00<?, ?B/s]
```

```
# Prepare the model for training
optimizer = tf.keras.optimizers.Adam(learning_rate=2e-5)
loss = tf.keras.losses.CategoricalCrossentropy(from_logits=True)
model.compile(optimizer=optimizer, loss=loss, metrics=['accuracy'])
```

Fitting the model:

```
# Training the model
model.fit([X_train_ids, X_train_masks], y_train, epochs=1, batch_size=12,
          validation_data=([X_test_ids, X_test_masks], y_test))
```

Trail 2 (albert model):

Encoding:

```
# Initialize the ALBERT tokenizer
tokenizer = AlbertTokenizer.from_pretrained('albert-base-v2')
```

```
Downloading:   0%|          | 0.00/742k [00:00<?, ?B/s]

Downloading:   0%|          | 0.00/25.0 [00:00<?, ?B/s]

Downloading:   0%|          | 0.00/684 [00:00<?, ?B/s]
```

```
# Define a function to encode texts into a format suitable for the model training
def encode_texts(tokenizer, texts, max_len=256):
    input_ids = []
    attention_masks = []
    for text in texts:
        encoded = tokenizer.encode_plus(
            text,
            add_special_tokens=True,
            max_length=max_len,
            truncation=True,
            padding='max_length',
            return_attention_mask=True,
            return_tensors='tf',
        )
        # Ensure the tensors are squeezed to remove unnecessary dimensions
        input_ids.append(tf.squeeze(encoded['input_ids']))
        attention_masks.append(tf.squeeze(encoded['attention_mask']))
    return np.array(input_ids), np.array(attention_masks)
```

```
# Encode the text data
X_train_ids, X_train_masks = encode_texts(tokenizer, train['Description'].values, max_len=512)
X_test_ids, X_test_masks = encode_texts(tokenizer, test['Description'].values, max_len=512)
```

Preparing the model:

```
]: # Load the DistilBERT model
   model = TFDistilBertForSequenceClassification.from_pretrained('distilbert-base-uncased', num_labels=len(label_encoder.classes_))

   Downloading:   0%|          | 0.00/347M [00:00<?, ?B/s]

]: # Prepare the model for training
   optimizer = tf.keras.optimizers.Adam(learning_rate=5e-5)
   loss = tf.keras.losses.CategoricalCrossentropy(from_logits=True)
   model.compile(optimizer=optimizer, loss=loss, metrics=['accuracy'])
```

Fitting the model:

```
# Training the model
model.fit([X_train_ids, X_train_masks], y_train, epochs=1, batch_size=12,
          validation_data=([X_test_ids, X_test_masks], y_test))
```

Trail 3(albert model):

Encoding:

```
# Initialize the ALBERT tokenizer
tokenizer = AlbertTokenizer.from_pretrained('albert-base-v2')
```

```
Downloading:   0%|          | 0.00/742k [00:00<?, ?B/s]

Downloading:   0%|          | 0.00/25.0 [00:00<?, ?B/s]

Downloading:   0%|          | 0.00/684 [00:00<?, ?B/s]
```

```
# Define a function to encode texts into a format suitable for the model training
def encode_texts(tokenizer, texts, max_len=256):
    input_ids = []
    attention_masks = []
    for text in texts:
        encoded = tokenizer.encode_plus(
            text,
            add_special_tokens=True,
            max_length=max_len,
            truncation=True,
            padding='max_length',
            return_attention_mask=True,
            return_tensors='tf',
        )
        # Ensure the tensors are squeezed to remove unnecessary dimensions
        input_ids.append(tf.squeeze(encoded['input_ids']))
        attention_masks.append(tf.squeeze(encoded['attention_mask']))
    return np.array(input_ids), np.array(attention_masks)
```

```
# Encode the text data
X_train_ids, X_train_masks = encode_texts(tokenizer, train['Description'].values, max_len=512)
X_test_ids, X_test_masks = encode_texts(tokenizer, test['Description'].values, max_len=512)
```

Preparing the model:

```
# Load the ALBERT model
model = TFAlbertForSequenceClassification.from_pretrained('albert-base-v2', num_labels=len(label_encoder.classes_))
```

```
# Prepare the model for training
optimizer = tf.keras.optimizers.Adam(learning_rate=2e-5)
loss = tf.keras.losses.CategoricalCrossentropy(from_logits=True)
model.compile(optimizer=optimizer, loss=loss, metrics=['accuracy'])
```

Fitting the model:

```
# Training the model
model.fit([X_train_ids, X_train_masks], y_train, epochs=1, batch_size=12,
          validation_data=([X_test_ids, X_test_masks], y_test))
```

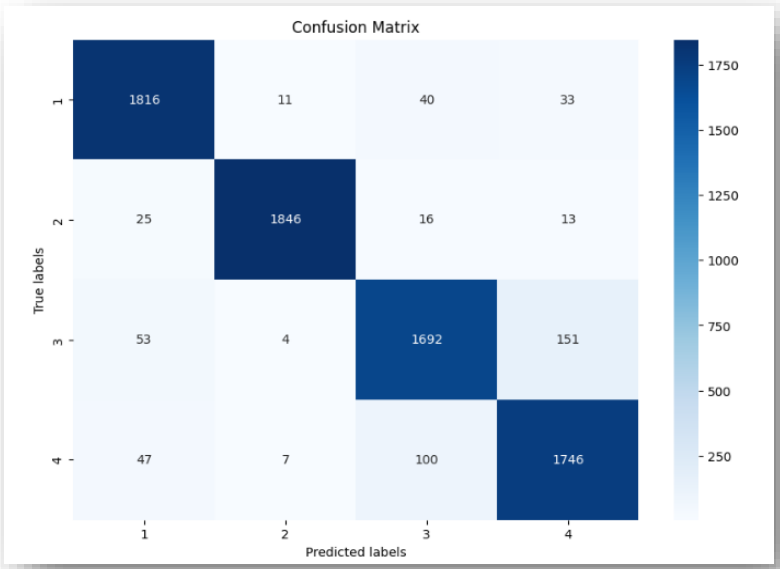
Comparison:

Based On Confusion Matrix:

Classes: 1-World, 2-Sports, 3-Business, 4-Sci/Tech

The total of each class is **1900**.

Model 1:



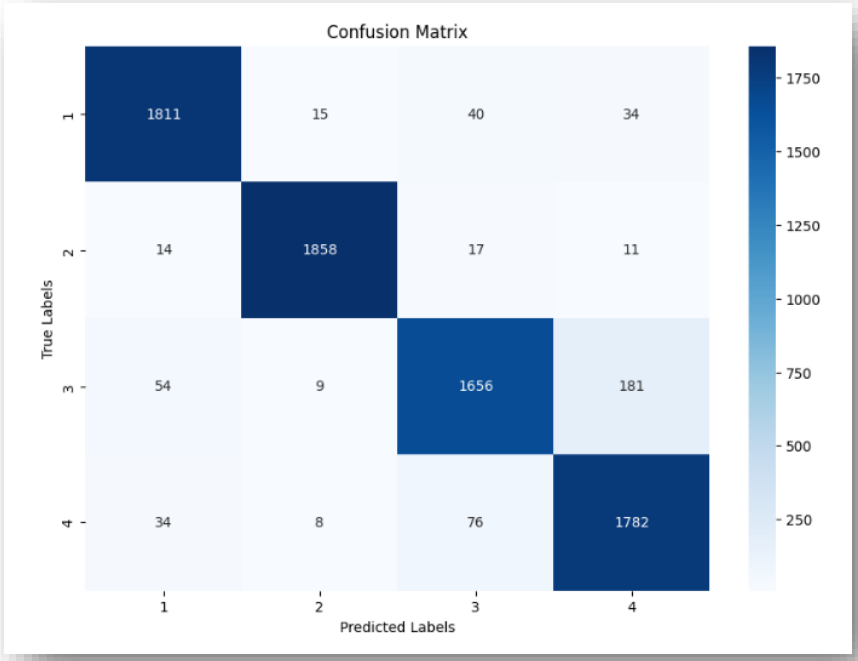
1:1816

2:1846

3:1692

4:1746

Model 2:



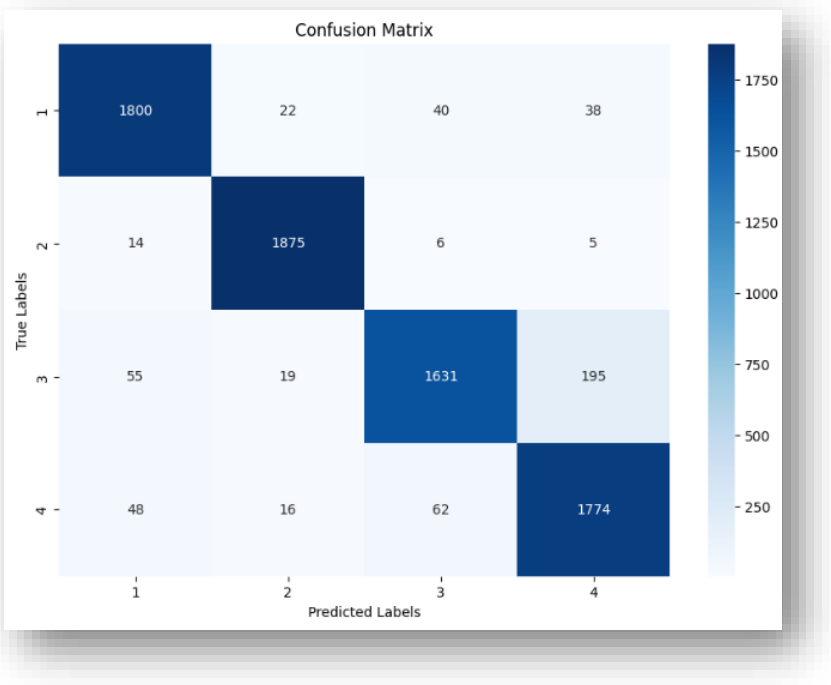
1: 1811

2: 1858

3: 1656

4:1782

Model 3:



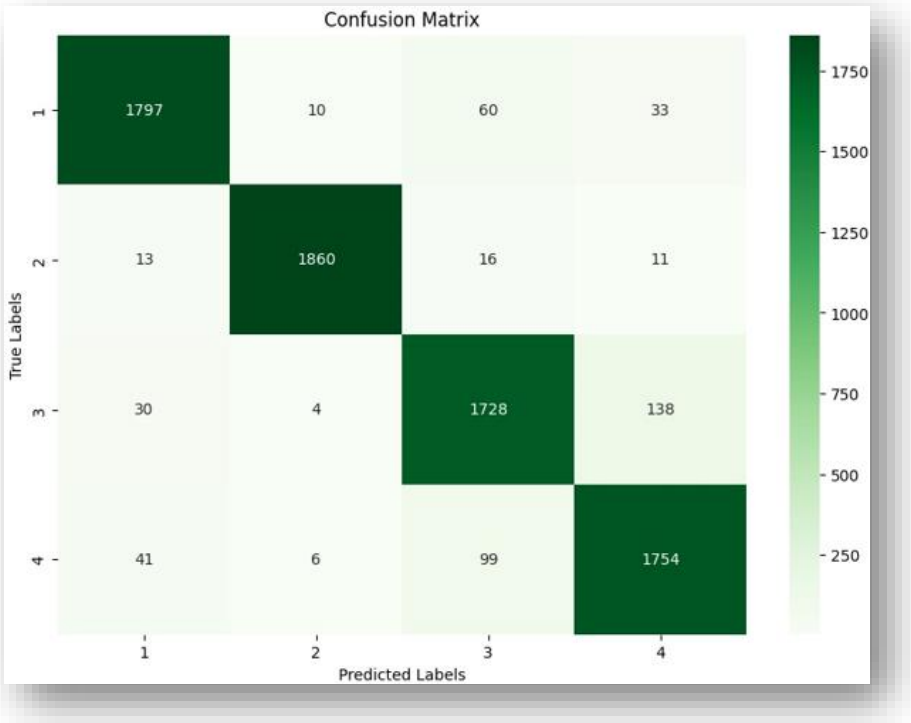
1: 1800

2: 1875

3:1631

4:1774

Model 4:



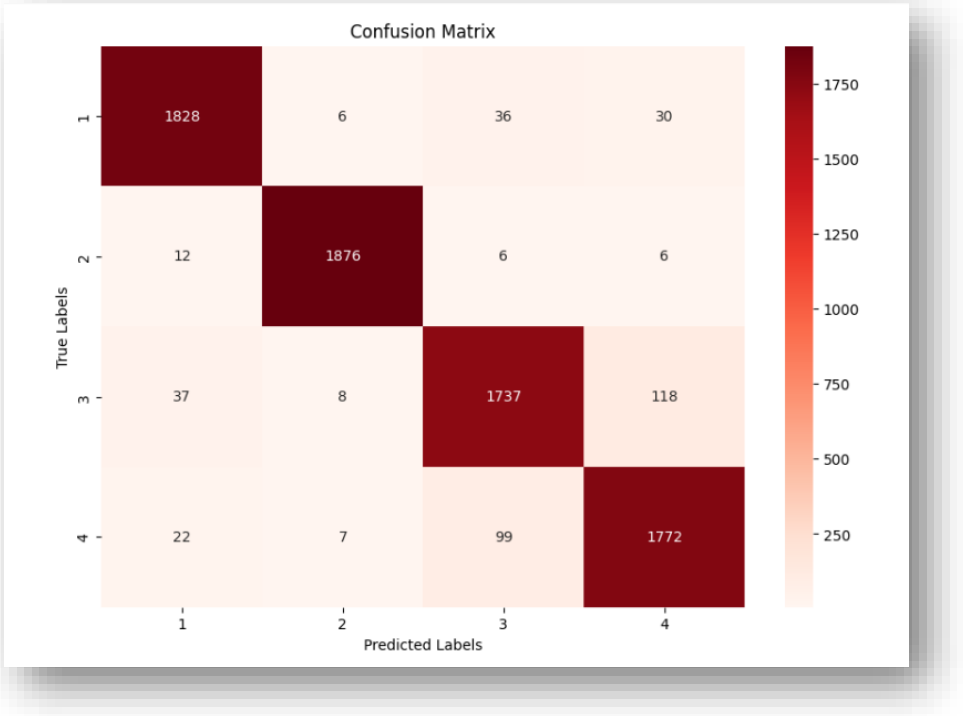
1: 1797

2: 1860

3:1728

4:1754

Model5:



1: 1828

2: 1876

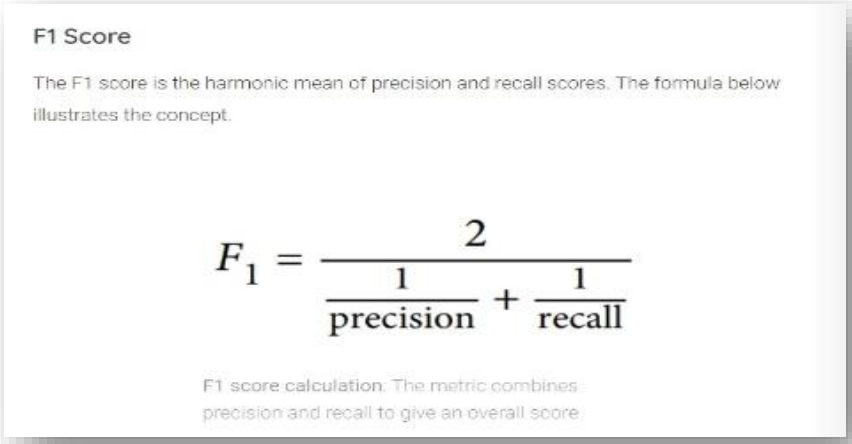
3:1737

4: 1772

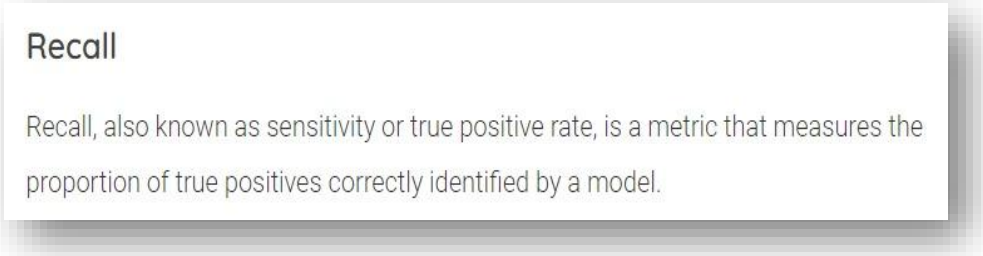
The Models Ranking from Highest to Lowest Based on the Confusion Matrices:

- Model 5
- Model 4
- Model 2
- Model 1
- Model 3

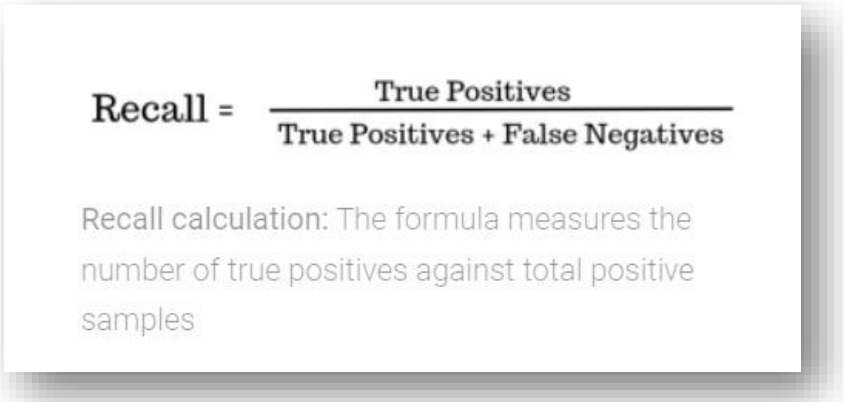
Classification report comparison:



[2]



[2]



[2]

Precision

Precision is a metric that measures the proportion of true positives (correct predictions) against the model's total positive predictions. The formula below summarizes the concept.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

Precision calculation: The formula measures the number of correct predictions relative to the number of total positive predictions

[2]

What is Accuracy?

Accuracy measures how often a model predicts the outcome correctly relative to the total number of predictions. The metric has widespread use for measuring model performance in computer vision tasks, including classification, object detection, and segmentation.

$$\text{Accuracy} = \frac{\text{Correct Prediction}}{\text{All Predictions}}$$

Accuracy calculation: The formula is the ratio of correct predictions to the total number of predictions

[2]

Model 5:

```
Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.96      0.96      1900
           1       0.99      0.99      0.99      1900
           2       0.92      0.91      0.92      1900
           3       0.92      0.93      0.93      1900

    accuracy                           0.95      7600
   macro avg       0.95      0.95      0.95      7600
weighted avg       0.95      0.95      0.95      7600
```

Model 4:

```
Classification Report:
              precision    recall  f1-score   support

           1       0.96      0.95      0.95      1900
           2       0.99      0.98      0.98      1900
           3       0.91      0.91      0.91      1900
           4       0.91      0.92      0.91      1900

    accuracy                           0.94      7600
   macro avg       0.94      0.94      0.94      7600
weighted avg       0.94      0.94      0.94      7600
```

Model 3:

```
Classification Report:
              precision    recall  f1-score   support

           1       0.94      0.95      0.94      1900
           2       0.97      0.99      0.98      1900
           3       0.94      0.86      0.90      1900
           4       0.88      0.93      0.91      1900

    accuracy                           0.93      7600
   macro avg       0.93      0.93      0.93      7600
weighted avg       0.93      0.93      0.93      7600
```

Model 2:

Classification Report:

	precision	recall	f1-score	support
1	0.95	0.95	0.95	1900
2	0.98	0.98	0.98	1900
3	0.93	0.87	0.90	1900
4	0.89	0.94	0.91	1900
accuracy			0.94	7600
macro avg	0.94	0.94	0.94	7600
weighted avg	0.94	0.94	0.94	7600

Model 1:

	precision	recall	f1-score	support
1	0.94	0.96	0.95	1900
2	0.99	0.97	0.98	1900
3	0.92	0.89	0.90	1900
4	0.90	0.92	0.91	1900
accuracy			0.93	7600
macro avg	0.93	0.93	0.93	7600
weighted avg	0.93	0.93	0.93	7600

The Models Ranking from Highest to Lowest Based on the based on the accuracy, macro-average precision, recall, and F1-score from the Classification Reports:

- Model 5
- Model 2 & Model 4
- Model 1 & Model 3

Conclusion:

In the beginning we used the **Bert** model, but it was time consuming and required high computational resources, so we went for the **Albert** model(trials) it was lightweight compared to Bert, but it was not enough due to the training time. In the end, we settled on **DistilBERT** because we were capable of training it on Google Colab taking a much lesser time than the previous models. Between the models **DistilBERT** and **Bert**, the **Bert** model scored higher in terms of accuracy, confusion matrix and classification report.

References:

[1] Aman Anand Rai. "AG News Classification Dataset." Kaggle. Available online: https://www.kaggle.com/datasets/amananandrai/ag-news-classification-dataset. Accessed on May 15, 2024.

[2] Viso.ai. "Precision-Recall in Computer Vision." Available online: https://viso.ai/computer-vision/precision-recall/. Accessed on [May 15, 2024.]