AGNews Classification

```
!pip install datasets==1.6.1
```

```
pip install transformers
```

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
import numpy as np
from datasets import load_dataset
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.preprocessing.text import Tokenizer
from sklearn.model_selection import train_test_split
import os
import zipfile
import tensorflow as tf
from tensorflow.keras.models import Model, Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense, Dropout, GlobalMaxPooli
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping
from tensorflow.keras.utils import plot_model
from tensorflow.keras.callbacks import EarlyStopping
```

```
vocab_size = 20000
maxlen = 200

(train), (test) =  load_dataset('ag_news', split=['train', 'test'])
```

```
WARNING:datasets.builder:Using custom data configuration default
WARNING:datasets.builder:Reusing dataset ag_news (/root/.cache/huggingface/dat
The cache for model files in Transformers v4.22.0 has been updated. Migrating
Moving 0 files to the new cache system
```

0it [00:04, ?it/s]

```
training_labels = []
testing_labels = []

for l in train['label']:
   training_labels.append(l)
```

```
for l in test['label']:
   testing_labels.append(l)

training_labels_final = np.array(training_labels)
testing_labels_final = np.array(testing_labels)
```

```
padding_type='post'
truncation_type='post'
max_length = 100
oov_tok = '<OOV>'
tokenizer = Tokenizer(num_words = vocab_size, oov_token=oov_tok)
tokenizer.fit_on_texts(train['text'])
X_train_sequences = tokenizer.texts_to_sequences(train['text'])
X_train_padded = pad_sequences(X_train_sequences, maxlen=max_length, truncating='po
X_test_sequences = tokenizer.texts_to_sequences(test['text'])
X_test_padded = pad_sequences(X_test_sequences,maxlen=max_length)
```

```
X_train_padded, X_val_padded, y_train, y_val = train_test_split(X_train_padded, tra
```

```
!wget --no-check-certificate \
    http://nlp.stanford.edu/data/glove.6B.zip \
    -O /tmp/glove.6B.zip
```

```
word_index = tokenizer.word_index
```

```
with zipfile.ZipFile('/tmp/glove.6B.zip', 'r') as zip_ref:
    zip_ref.extractall('/tmp/glove')
```

```
embeddings_index = {}
f = open('/tmp/glove/glove.6B.100d.txt')
for line in f:
    values = line.split()
    word = values[0]
    coefs = np.asarray(values[1:], dtype='float32')
    embeddings_index[word] = coefs
f.close()
```

```
embedding_matrix = np.zeros((len(word_index) + 1, max_length))
for word, i in word_index.items():
    embedding_vector = embeddings_index.get(word)
    if embedding_vector is not None:
        embedding_matrix[i] = embedding_vector
```

```
embedding_layer = Embedding(len(word_index) + 1,
                             max_length,
                             weights=[embedding_matrix],
                             input_length=max_length,
                             trainable=False)
```

```
model = Sequential([
    embedding_layer,
    Bidirectional(LSTM(150, return_sequences=True)),
    (Dropout(0.2)),
    Bidirectional(LSTM(150)),
    Dense(6, activation='relu'),
    (Dropout(0.2)),
    Dense(6, activation='relu'),
   Dense(4, activation='softmax')
])
```

```
model.compile(loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              optimizer='adam',
              metrics=['accuracy'])
```

```
callbacks = [
            EarlyStopping(patience = 10)
            ]
num_epochs = 7
history = model.fit(X_train_padded, y_train, epochs=num_epochs, validation_data=(X_
```

```
 1/7
local/lib/python3.7/dist-packages/tensorflow/python/util/dispatch.py:1082: Use
urn dispatch_target(*args, **kwargs)
3375 [==============================] - 107s 29ms/step - loss: 0.5362 - accura
 2/7
3375 [==============================] - 97s 29ms/step - loss: 0.3511 - accurac
 3/7
3375 [==============================] - 95s 28ms/step - loss: 0.3105 - accurac
 4/7
3375 [==============================] - 96s 28ms/step - loss: 0.2795 - accurac
 5/7
3375 [==============================] - 95s 28ms/step - loss: 0.2561 - accurac
 6/7
3375 [==============================] - 93s 28ms/step - loss: 0.2364 - accurac
 7/7
3375 [==============================] - 92s 27ms/step - loss: 0.2189 - accurac
```

```
test_loss, test_acc = model.evaluate(X_test_padded, testing_labels_final)
print("Test accuracy:",test_acc)
```

```
238/238 [==============================] - 4s 12ms/step - loss: 0.2519 - accur
Test accuracy: 0.9203947186470032
```