Text Classification Using Transformer Networks (BERT)

Some initialization:

```
import random
import torch
import numpy as np
import pandas as pd
from tqdm.notebook import tqdm

# enable tqdm in pandas
tqdm.pandas()

# set to True to use the gpu (if there is one available)
use_gpu = True

# select device
device = torch.device('cuda' if use_gpu and torch.cuda.is_available()
else 'cpu')
print(f'device: {device.type}')

# random seed
seed = 1122

# set random seed
if seed is not None:
    print(f'random seed: {seed}')
    random.seed(seed)
    np.random.seed(seed)
    torch.manual_seed(seed)
```

```
device: cuda
random seed: 1122
```

Read the train/dev/test datasets and create a HuggingFace `Dataset` object:

```
def read_data(filename):
    # read csv file
    df = pd.read_csv(filename, header=None)
    # add column names
    df.columns = ['label', 'title', 'description']
    # make labels zero-based
    df['label'] -= 1
    # concatenate title and description, and remove backslashes
    df['text'] = df['title'] + " " + df['description']
```

```
    df['text'] = df['text'].str.replace('\\', ' ', regex=False)
    return df

labels = open('agnews_classes.txt').read().splitlines()
train_df =
read_data('https://raw.githubusercontent.com/mhjabreel/CharCnn_Keras/
refs/heads/master/data/ag_news_csv/train.csv')
test_df =
read_data('https://raw.githubusercontent.com/mhjabreel/CharCnn_Keras/
refs/heads/master/data/ag_news_csv/test.csv')
train_df
```

```
{"type":"dataframe","variable_name":"train_df"}
```

```
from sklearn.model_selection import train_test_split

train_df, eval_df = train_test_split(train_df, train_size=0.9)
train_df.reset_index(inplace=True, drop=True)
eval_df.reset_index(inplace=True, drop=True)

print(f'train rows: {len(train_df.index):,}')
print(f'eval rows: {len(eval_df.index):,}')
print(f'test rows: {len(test_df.index):,}')
```

```
train rows: 108,000
eval rows: 12,000
test rows: 7,600
```

```
!pip install datasets
```

```
Requirement already satisfied: datasets in
/usr/local/lib/python3.10/dist-packages (3.1.0)
Requirement already satisfied: filelock in
/usr/local/lib/python3.10/dist-packages (from datasets) (3.16.1)
Requirement already satisfied: numpy>=1.17 in
/usr/local/lib/python3.10/dist-packages (from datasets) (1.26.4)
Requirement already satisfied: pyarrow>=15.0.0 in
/usr/local/lib/python3.10/dist-packages (from datasets) (18.0.0)
Requirement already satisfied: dill<0.3.9,>=0.3.0 in
/usr/local/lib/python3.10/dist-packages (from datasets) (0.3.8)
Requirement already satisfied: pandas in
/usr/local/lib/python3.10/dist-packages (from datasets) (2.1.4)
Requirement already satisfied: requests>=2.32.2 in
/usr/local/lib/python3.10/dist-packages (from datasets) (2.32.3)
Requirement already satisfied: tqdm>=4.66.3 in
/usr/local/lib/python3.10/dist-packages (from datasets) (4.66.5)
Requirement already satisfied: xxhash in
/usr/local/lib/python3.10/dist-packages (from datasets) (3.5.0)
Requirement already satisfied: multiprocess<0.70.17 in
/usr/local/lib/python3.10/dist-packages (from datasets) (0.70.16)
Requirement already satisfied: fsspec<=2024.9.0,>=2023.1.0 in
```

```
/usr/local/lib/python3.10/dist-packages (from
fsspec[http]<=2024.9.0,>=2023.1.0->datasets) (2024.6.1)
Requirement already satisfied: aiohttp in
/usr/local/lib/python3.10/dist-packages (from datasets) (3.10.5)
Requirement already satisfied: huggingface-hub>=0.23.0 in
/usr/local/lib/python3.10/dist-packages (from datasets) (0.24.7)
Requirement already satisfied: packaging in
/usr/local/lib/python3.10/dist-packages (from datasets) (24.1)
Requirement already satisfied: pyyaml>=5.1 in
/usr/local/lib/python3.10/dist-packages (from datasets) (6.0.2)
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->datasets)
(2.4.0)
Requirement already satisfied: aiosignal>=1.1.2 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->datasets)
(1.3.1)
Requirement already satisfied: attrs>=17.3.0 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->datasets)
(24.2.0)
Requirement already satisfied: frozenlist>=1.1.1 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->datasets)
(1.4.1)
Requirement already satisfied: multidict<7.0,>=4.5 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->datasets)
(6.1.0)
Requirement already satisfied: yarl<2.0,>=1.0 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->datasets)
(1.11.1)
Requirement already satisfied: async-timeout<5.0,>=4.0 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->datasets)
(4.0.3)
Requirement already satisfied: typing-extensions>=3.7.4.3 in
/usr/local/lib/python3.10/dist-packages (from huggingface-hub>=0.23.0-
>datasets) (4.12.2)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests>=2.32.2-
>datasets) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.10/dist-packages (from requests>=2.32.2-
>datasets) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests>=2.32.2-
>datasets) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests>=2.32.2-
>datasets) (2024.8.30)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.10/dist-packages (from pandas->datasets)
(2.8.2)
```

```
Requirement already satisfied: pytz>=2020.1 in
/usr/local/lib/python3.10/dist-packages (from pandas->datasets)
(2024.2)
Requirement already satisfied: tzdata>=2022.1 in
/usr/local/lib/python3.10/dist-packages (from pandas->datasets)
(2024.1)
Requirement already satisfied: six>=1.5 in
/usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2-
>pandas->datasets) (1.16.0)
```

```
from datasets import Dataset, DatasetDict

ds = DatasetDict()
ds['train'] = Dataset.from_pandas(train_df)
ds['validation'] = Dataset.from_pandas(eval_df)
ds['test'] = Dataset.from_pandas(test_df)
ds
```

```
DatasetDict({
    train: Dataset({
        features: ['label', 'title', 'description', 'text'],
        num_rows: 108000
    })
    validation: Dataset({
        features: ['label', 'title', 'description', 'text'],
        num_rows: 12000
    })
    test: Dataset({
        features: ['label', 'title', 'description', 'text'],
        num_rows: 7600
    })
})
```

Tokenize the texts:

```
from transformers import AutoTokenizer

transformer_name = 'bert-base-cased'
tokenizer = AutoTokenizer.from_pretrained(transformer_name)
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/
_token.py:89: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your
settings tab (https://huggingface.co/settings/tokens), set it as
secret in your Google Colab and restart your session.
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to
access public models or datasets.
  warnings.warn(
```

{"model_id":"227b7fd833304070a099bc19545a9f12","version_major":2,"version_minor":0}

{"model_id":"aebd88ff59314e0dbfa370d23888fe5a","version_major":2,"version_minor":0}

{"model_id":"f68a633147f14662a74d7e96a2a5ad1f","version_major":2,"version_minor":0}

{"model_id":"16d73a4fe916499ca0ad92e9a4c3902e","version_major":2,"version_minor":0}

```
/usr/local/lib/python3.10/dist-packages/transformers/
tokenization_utils_base.py:1601: FutureWarning:
`clean_up_tokenization_spaces` was not set. It will be set to `True`
by default. This behavior will be depracted in transformers v4.45, and
will be then set to `False` by default. For more details check this
issue: https://github.com/huggingface/transformers/issues/31884
  warnings.warn(
```

```
def tokenize(examples):
    return tokenizer(examples['text'], truncation=True)

train_ds = ds['train'].map(
    tokenize, batched=True,
    remove_columns=['title', 'description', 'text'],
)
eval_ds = ds['validation'].map(
    tokenize,
    batched=True,
    remove_columns=['title', 'description', 'text'],
)
train_ds.to_pandas()
```

{"model_id":"6c88a355c894439ea99abbfad81293a5","version_major":2,"version_minor":0}

{"model_id":"be77d96a71f84a9990ed258c08052759","version_major":2,"version_minor":0}

{"type":"dataframe"}

Create the transformer model:

```
from torch import nn
from transformers.modeling_outputs import SequenceClassifierOutput
from transformers.models.bert.modeling_bert import BertModel,
BertPreTrainedModel

#
https://github.com/huggingface/transformers/blob/65659a29cf5a079842e61
```

```
a63d57fa24474288998/src/transformers/models/bert/
modeling_bert.py#L1486

class BertForSequenceClassification(BertPreTrainedModel):
    def __init__(self, config):
        super().__init__(config)
        self.num_labels = config.num_labels
        self.bert = BertModel(config)
        self.dropout = nn.Dropout(config.hidden_dropout_prob)
        self.classifier = nn.Linear(config.hidden_size,
config.num_labels)
        self.init_weights()

    def forward(self, input_ids=None, attention_mask=None,
token_type_ids=None, labels=None, **kwargs):
        outputs = self.bert(
            input_ids,
            attention_mask=attention_mask,
            token_type_ids=token_type_ids,
            **kwargs,
        )
        cls_outputs = outputs.last_hidden_state[:, 0, :]
        cls_outputs = self.dropout(cls_outputs)
        logits = self.classifier(cls_outputs)
        loss = None
        if labels is not None:
            loss_fn = nn.CrossEntropyLoss()
            loss = loss_fn(logits, labels)
        return SequenceClassifierOutput(
            loss=loss,
            logits=logits,
            hidden_states=outputs.hidden_states,
            attentions=outputs.attentions,
        )

from transformers import AutoConfig

config = AutoConfig.from_pretrained(
    transformer_name,
    num_labels=len(labels),
)

model = (
    BertForSequenceClassification
    .from_pretrained(transformer_name, config=config)
)

{"model_id":"a62030245b9e4bbdb188449ee93897c9","version_major":2,"vers
ion_minor":0}
```

```
Some weights of BertForSequenceClassification were not initialized
from the model checkpoint at bert-base-cased and are newly
initialized: ['classifier.bias', 'classifier.weight']
You should probably TRAIN this model on a down-stream task to be able
to use it for predictions and inference.
```

Create the trainer object and train:

```
from transformers import TrainingArguments

num_epochs = 2
batch_size = 24
weight_decay = 0.01
model_name = f'{transformer_name}-sequence-classification'

training_args = TrainingArguments(
    output_dir=model_name,
    log_level='error',
    num_train_epochs=num_epochs,
    per_device_train_batch_size=batch_size,
    per_device_eval_batch_size=batch_size,
    evaluation_strategy='epoch',
    weight_decay=weight_decay,
)
```

```
/usr/local/lib/python3.10/dist-packages/transformers/
training_args.py:1525: FutureWarning: `evaluation_strategy` is
deprecated and will be removed in version 4.46 of  Transformers. Use
`eval_strategy` instead
  warnings.warn(
```

```
from sklearn.metrics import accuracy_score

def compute_metrics(eval_pred):
    y_true = eval_pred.label_ids
    y_pred = np.argmax(eval_pred.predictions, axis=-1)
    return {'accuracy': accuracy_score(y_true, y_pred)}

from transformers import Trainer

trainer = Trainer(
    model=model,
    args=training_args,
    compute_metrics=compute_metrics,
    train_dataset=train_ds,
    eval_dataset=eval_ds,
    tokenizer=tokenizer,
)

!pip install wandb
```

```
Collecting wandb
  Downloading wandb-0.18.7-py3-none-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (9.7 kB)
Requirement already satisfied: click!=8.0.0,>=7.1 in
/usr/local/lib/python3.10/dist-packages (from wandb) (8.1.7)
Collecting docker-pycreds>=0.4.0 (from wandb)
  Downloading docker_pycreds-0.4.0-py2.py3-none-any.whl.metadata (1.8
kB)
Collecting gitpython!=3.1.29,>=1.0.0 (from wandb)
  Downloading GitPython-3.1.43-py3-none-any.whl.metadata (13 kB)
Requirement already satisfied: platformdirs in
/usr/local/lib/python3.10/dist-packages (from wandb) (4.3.6)
Requirement already satisfied: protobuf!=4.21.0,!=5.28.0,<6,>=3.19.0
in /usr/local/lib/python3.10/dist-packages (from wandb) (3.20.3)
Requirement already satisfied: psutil>=5.0.0 in
/usr/local/lib/python3.10/dist-packages (from wandb) (5.9.5)
Requirement already satisfied: pyyaml in
/usr/local/lib/python3.10/dist-packages (from wandb) (6.0.2)
Requirement already satisfied: requests<3,>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from wandb) (2.32.3)
Collecting sentry-sdk>=2.0.0 (from wandb)
  Downloading sentry_sdk-2.18.0-py2.py3-none-any.whl.metadata (9.9 kB)
Collecting setproctitle (from wandb)
  Downloading setproctitle-1.3.4-cp310-cp310-
manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.manylinux
2014_x86_64.whl.metadata (10 kB)
Requirement already satisfied: setuptools in
/usr/local/lib/python3.10/dist-packages (from wandb) (71.0.4)
Requirement already satisfied: typing-extensions<5,>=4.4 in
/usr/local/lib/python3.10/dist-packages (from wandb) (4.12.2)
Requirement already satisfied: six>=1.4.0 in
/usr/local/lib/python3.10/dist-packages (from docker-pycreds>=0.4.0-
>wandb) (1.16.0)
Collecting gitdb<5,>=4.0.1 (from gitpython!=3.1.29,>=1.0.0->wandb)
  Downloading gitdb-4.0.11-py3-none-any.whl.metadata (1.2 kB)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.0.0-
>wandb) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.0.0-
>wandb) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.0.0-
>wandb) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.0.0-
>wandb) (2024.8.30)
Collecting smmap<6,>=3.0.1 (from gitdb<5,>=4.0.1->gitpython!
=3.1.29,>=1.0.0->wandb)
  Downloading smmap-5.0.1-py3-none-any.whl.metadata (4.3 kB)
```

```
Downloading wandb-0.18.7-py3-none-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl (16.1 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 16.1/16.1 MB 98.8 MB/s eta
0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 207.3/207.3 kB 20.4 MB/s eta
0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 317.5/317.5 kB 19.4 MB/s eta
0:00:00
anylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.manylinux2
014_x86_64.whl (30 kB)
Downloading gitdb-4.0.11-py3-none-any.whl (62 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 62.7/62.7 kB 5.8 MB/s eta
0:00:00
map-5.0.1-py3-none-any.whl (24 kB)
Installing collected packages: smmap, setproctitle, sentry-sdk,
docker-pycreds, gitdb, gitpython, wandb
Successfully installed docker-pycreds-0.4.0 gitdb-4.0.11 gitpython-
3.1.43 sentry-sdk-2.18.0 setproctitle-1.3.4 smmap-5.0.1 wandb-0.18.7

trainer.train()

<IPython.core.display.HTML object>

TrainOutput(global_step=9000, training_loss=0.17164631907145184,
metrics={'train_runtime': 5223.2333, 'train_samples_per_second':
41.354, 'train_steps_per_second': 1.723, 'total_flos':
1.3075366022140032e+16, 'train_loss': 0.17164631907145184, 'epoch':
2.0})
```

Evaluate on the test partition:

```
test_ds = ds['test'].map(
    tokenize,
    batched=True,
    remove_columns=['title', 'description', 'text'],
)
test_ds.to_pandas()

{"model_id":"6c402e8a301740c8851df5fa5ccdb56f","version_major":2,"vers
ion_minor":0}

{"summary":"{\n  \"name\": \"test_ds\",\n  \"rows\": 7600,\n
\"fields\": [\n    {\n      \"column\": \"label\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
1,\n        \"min\": 0,\n        \"max\": 3,\n
\"num_unique_values\": 4,\n        \"samples\": [\n          3,\n
0,\n          2\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"input_ids\",\n      \"properties\": {\n        \"dtype\":
\"object\",\n        \"semantic_type\": \"\",\n
```

```
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"token_type_ids\",\n      \"properties\": {\n        \"dtype\":
\"object\",\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"attention_mask\",\n      \"properties\": {\n        \"dtype\":
\"object\",\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\n}","type":"dataframe"}
```

```
output = trainer.predict(test_ds)
output
```

```
<IPython.core.display.HTML object>
```

```
PredictionOutput(predictions=array([[ 0.2702788 , -3.8723545 ,
4.8265123 , -1.4792093 ],
       [-0.5341982 , -3.978437  , -2.7466586 ,  6.3919373 ],
       [-0.40933844, -4.1844025 , -1.9446223 ,  5.871319  ],
       ...,
       [-1.4534968 ,  7.8491173 , -2.5994985 , -3.259976  ],
       [-0.27771565, -3.440511  ,  5.8607273 , -2.4316626 ],
       [-2.995202  , -4.3541536 ,  4.739249  ,  1.7062145 ]],
      dtype=float32), label_ids=array([2, 3, 3, ..., 1, 2, 2]),
metrics={'test_loss': 0.18299901485443115, 'test_accuracy':
0.9468421052631579, 'test_runtime': 51.0118,
'test_samples_per_second': 148.985, 'test_steps_per_second': 6.214})
```

```
from sklearn.metrics import classification_report

y_true = output.label_ids
y_pred = np.argmax(output.predictions, axis=-1)
target_names = labels
print(classification_report(y_true, y_pred,
target_names=target_names))
```

```
              precision    recall  f1-score   support

       World       0.96      0.95      0.96      1900
      Sports       0.99      0.99      0.99      1900
    Business       0.93      0.91      0.92      1900
    Sci/Tech       0.91      0.94      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
```

Observamos que los resultados son muy buenos, pues se obtiene un 95% de exactitud y métricas muy buenas para cada clase: F1-scores superiores a 90%, lo cual indica un buen desempeño en la precisión y recall de cada clase.

Estructura del Pipeline

1. **Configuración Inicial**:
   - Se configura el dispositivo para trabajar con GPU o CPU, dependiendo de lo que esté disponible.
   - Se establece una semilla aleatoria para que los resultados puedan repetirse.
2. **Preparación de los Datos**:
   - Los datos se cargan desde archivos CSV y se limpian para eliminar caracteres no deseados.
   - Se combinan las columnas de título y descripción en un solo texto para procesarlo de manera más eficiente.
   - Se dividen los datos en conjuntos de entrenamiento, validación y prueba.
3. **Creación de Datasets**:
   - Los datos procesados se convierten en estructuras especiales llamadas `Dataset` y `DatasetDict`, que son útiles para trabajar con modelos de `Hugging Face`.
4. **Tokenización**:
   - Se utiliza un modelo preentrenado para convertir los textos en números que el modelo pueda entender.
   - Esta tokenización se aplica a los textos de entrenamiento, validación y prueba.
5. **Definición del Modelo**:
   - Se utiliza un modelo BERT preentrenado que se adapta para la tarea de clasificación añadiendo una capa que predice las etiquetas.
6. **Configuración del Entrenamiento**:
   - Se eligen parámetros como el número de épocas, el tamaño de los lotes y cómo evaluar el desempeño del modelo durante el entrenamiento.
7. **Entrenamiento**:
   - El modelo se entrena con los datos de entrenamiento y se evalúa periódicamente con los datos de validación para ajustar su rendimiento.
8. **Evaluación Final**:
   - El modelo entrenado se prueba con el conjunto de datos de prueba para medir su precisión, recall y otras métricas importantes.
   - Se generan predicciones para analizar cómo clasifica los textos.
9. **Resultados y Análisis**:
   - Se preparan reportes con las métricas de desempeño del modelo, mostrando su precisión en cada categoría y en general.