```
import transformers
from transformers import BertModel, BertTokenizer, AdamW,
get_linear_schedule_with_warmup
import torch

import numpy as np
import pandas as pd
import seaborn as sns
from pylab import rcParams
import matplotlib.pyplot as plt
from matplotlib import rc
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report
from collections import defaultdict
from textwrap import wrap

from torch import nn, optim
from torch.utils.data import Dataset, DataLoader
import torch.nn.functional as F

%matplotlib inline

device = torch.device("cuda:0" if torch.cuda.is_available() else
"cpu")
device
```

```
device(type='cpu')
```

```
df = pd.read_csv("/20191002-items.csv")
df.head()
```

```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 10942,\n  \"fields\":
[\n    {\n      \"column\": \"itemId\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 209056794,\n        \"min\":
6068,\n        \"max\": 724216967,\n        \"num_unique_values\":
4422,\n        \"samples\": [\n          82256,\n          511518109,\
n          362125827\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"category\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 5,\n        \"samples\":
[\n          \"beli-laptop\",\n          \"shop-televisi-digital\",\n
\"beli-smart-tv\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"name\",\n      \"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 4286,\n        \"samples\": [\n
\"Bestrunner 64 MB USB 2.0 Logam Lipat Putar Flash Pena Stik Memori
Drive Disk U Perak\",\n          \"WD My Passport New Design
2TB/2.5Inch/USB3.0 - Kuning+Free Pouch+Pen+USB 8GB\",\n
\"Seagate New Backup Plus Portable Hardisk Eksternal 4TB USB3.0 +
Pouch + Pen\"\n        ],\n        \"semantic_type\": \"\",\n
```

\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"brandName\",\n   \"properties\": {\n   \"dtype\":
\"category\",\n   \"num_unique_values\": 234,\n
\"samples\": [\n   \"Beauties\",\n   \"JBS\",\n
\"Big House\"\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"url\",\n   \"properties\": {\n   \"dtype\": \"category\",\n
\"num_unique_values\": 4422,\n   \"samples\": [\n
\"https://www.lazada.co.id/products/kingston-dtig3-16gb-usb20-putih-
biru-i82256-s94970.html?search=1\",\n
\"https://www.lazada.co.id/products/vr24-power-usb-flashdisk-voice-
recorder-portable-digital-i511518109-s683272253.html?search=1\",\n
\"https://www.lazada.co.id/products/adata-flashdrives-uv150-flashdisk-
usb-31-super-speed-16-gb-black-i362125827-s380791162.html?search=1\"\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"price\",\n   \"properties\":
{\n   \"dtype\": \"number\",\n   \"std\": 6681452,\n
\"min\": 1000,\n   \"max\": 275000000,\n
\"num_unique_values\": 1861,\n   \"samples\": [\n
2550000,\n   389500,\n   1648000\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"averageRating\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
1,\n   \"min\": 1,\n   \"max\": 5,\n
\"num_unique_values\": 5,\n   \"samples\": [\n   3,\n
1,\n   5\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"totalReviews\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 260,\n   \"min\": 1,\n
\"max\": 9631,\n   \"num_unique_values\": 217,\n
\"samples\": [\n   201,\n   114,\n   79\
n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"retrievedDate\",\n   \"properties\": {\n   \"dtype\":
\"object\",\n   \"num_unique_values\": 1,\n   \"samples\":
[\n   \"2019-10-02\"\n   ],\n   \"semantic_type\":
\"\",\n   \"description\": \"\"\n   }\n   }\n]\
n}","type":"dataframe","variable_name":"df"}

```
df.shape
```

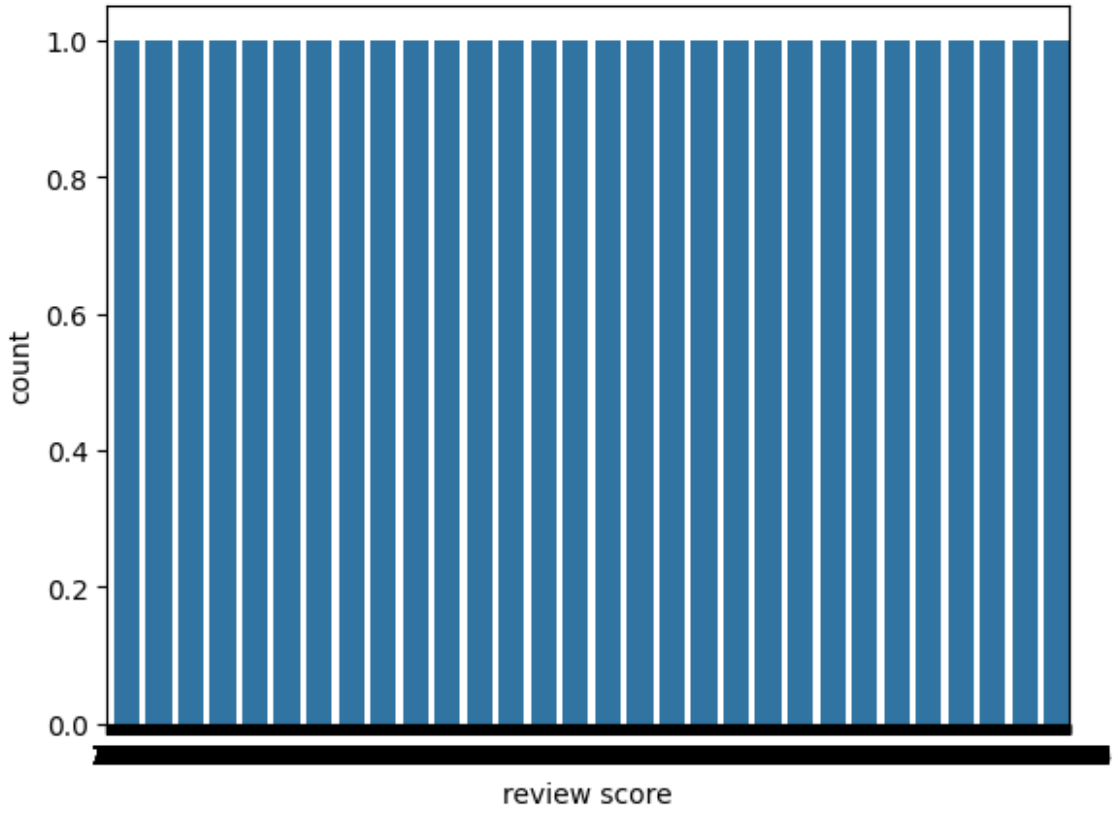
```
(10942, 9)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10942 entries, 0 to 10941
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
```

```
 0   itemId          10942 non-null  int64
 1   category        10942 non-null  object
 2   name            10942 non-null  object
 3   brandName       10938 non-null  object
 4   url             10942 non-null  object
 5   price           10942 non-null  int64
 6   averageRating   10942 non-null  int64
 7   totalReviews    10942 non-null  int64
 8   retrievedDate   10942 non-null  object
dtypes: int64(4), object(5)
memory usage: 769.5+ KB
```

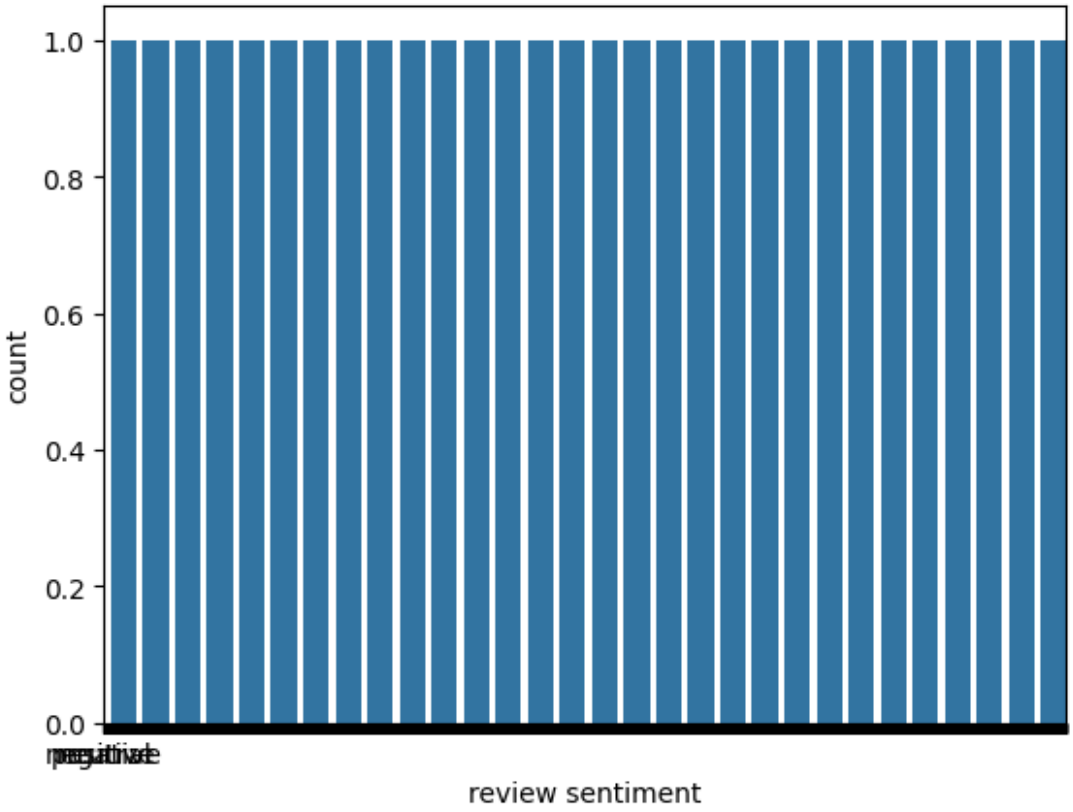
```
sns.countplot(df.averageRating)
plt.xlabel('review score');
```




```
def to_sentiment(rating):
    try:
        rating = int(rating)
        if rating <= 2:
            return 0
        elif rating == 3:
            return 1
        else:
            return 2
```

```
    except ValueError:
        return None  # or another default value

df['sentiment'] = df['averageRating'].apply(to_sentiment)

class_names = ['negative', 'neutral', 'positive']

ax = sns.countplot(df.sentiment)
plt.xlabel('review sentiment')
ax.set_xticklabels(class_names);
```

```
<ipython-input-66-15c40b0fa175>:3: UserWarning: set_ticklabels()
should only be used with a fixed number of ticks, i.e. after
set_ticks() or using a FixedLocator.
  ax.set_xticklabels(class_names);
```




```
PRE_TRAINED_MODEL_NAME = 'bert-base-cased'

tokenizer = BertTokenizer.from_pretrained(PRE_TRAINED_MODEL_NAME)
```

```
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/
_auth.py:94: 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":"52c4d0aeb4fd4ee8b519b0bae8bba9ca","version_major":2,"vers
ion_minor":0}

{"model_id":"0320c637ef4d4af88e1ec3ff8d876b2c","version_major":2,"vers
ion_minor":0}

{"model_id":"a4971a7cc326414594fc9ea1fda8c3ae","version_major":2,"vers
ion_minor":0}

{"model_id":"b7ec8e4cfac14239b146c6cf1391fb1f","version_major":2,"vers
ion_minor":0}
```


```
sample_txt = "This is an example sentence for tokenization."
encoding = tokenizer.encode_plus(
  sample_txt,
  max_length=32,
  add_special_tokens=True, # Add '[CLS]' and '[SEP]'
  return_token_type_ids=False,
  pad_to_max_length=True,
  return_attention_mask=True,  #RETURNS 0 FOR PADDINGS
  return_tensors='pt',  # Return PyTorch tensors
)

encoding.keys()
```

```
Truncation was not explicitly activated but `max_length` is provided a
specific value, please use `truncation=True` to explicitly truncate
examples to max length. Defaulting to 'longest_first' truncation
strategy. If you encode pairs of sequences (GLUE-style) with the
tokenizer you can select this strategy more precisely by providing a
specific strategy to `truncation`.
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_util
s_base.py:2681: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=True`
or `padding='longest'` to pad to the longest sequence in the batch, or
use `padding='max_length'` to pad to a max length. In this case, you
can give a specific length with `max_length` (e.g. `max_length=45`) or
leave max_length to None to pad to the maximal input size of the model
(e.g. 512 for Bert).
  warnings.warn(

dict_keys(['input_ids', 'attention_mask'])
```

```
print(len(encoding['input_ids'][0]))
encoding['input_ids'][0]
```

```
32
```

```
tensor([  101,  2023,  2003,  2019,  2742,  6251,  2005, 19204,  3989,
1012,
          102,     0,     0,     0,     0,     0,     0,     0,     0,
0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,
0,
            0,     0])
```

```
print(len(encoding['attention_mask'][0]))
encoding['attention_mask']
```

```
32
```

```
tensor([[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0]])
```

```
tokenizer.convert_ids_to_tokens(encoding['input_ids'][0])
```

```
['[CLS]',
 'this',
 'is',
 'an',
 'example',
 'sentence',
 'for',
 'token',
 '##ization',
 '.',
 '[SEP]',
 '[PAD]',
 '[PAD]',
 '[PAD]',
 '[PAD]',
 '[PAD]',
 '[PAD]',
 '[PAD]',
 '[PAD]',
 '[PAD]',
 '[PAD]',
 '[PAD]',
 '[PAD]',
 '[PAD]',
 '[PAD]',
 '[PAD]',
 '[PAD]',
 '[PAD]',
 '[PAD]',
 '[PAD]',
 '[PAD]',
 '[PAD]',
 '[PAD]',
```

```
 '[PAD]',
 '[PAD]']

token_lens = []

for txt in df.brandName:
    if isinstance(txt, str):  # Process only valid strings
        tokens = tokenizer.encode(txt, max_length=512,
truncation=True)
        token_lens.append(len(tokens))
    else:
        token_lens.append(0)  # Assign length 0 for missing values

print(token_lens)
```

```
[5, 5, 4, 3, 4, 4, 4, 3, 3, 4, 3, 4, 4, 4, 4, 4, 4, 4, 3, 4, 5, 3, 3,
4, 4, 4, 5, 4, 4, 4, 4, 4, 4, 4, 5, 5, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4,
5, 4, 3, 5, 3, 4, 4, 4, 4, 4, 4, 4, 4, 3, 5, 4, 5, 4, 4, 4, 4, 4, 4,
4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 3, 4, 4, 3, 3, 4, 5, 4, 4, 4,
4, 4, 3, 3, 4, 4, 5, 4, 4, 4, 4, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4, 4, 4, 3, 3, 4, 4, 5, 4, 4, 4, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 3, 5,
5, 4, 4, 3, 5, 4, 5, 5, 4, 3, 4, 4, 4, 5, 4, 3, 4, 3, 4, 4, 4, 3, 4,
4, 4, 4, 4, 4, 4, 4, 4, 5, 4, 3, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4, 4, 4, 4, 3, 4, 4, 5, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 5, 5, 4, 4, 4, 4, 4, 5,
3, 4, 4, 4, 3, 5, 5, 5, 4, 5, 5, 5, 5, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4,
4, 4, 5, 4, 5, 5, 5, 3, 5, 4, 4, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4,
4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 3, 3, 5, 5, 3, 5, 4, 5, 5, 4, 5, 4, 5,
5, 4, 0, 5, 4, 4, 3, 4, 4, 4, 5, 4, 5, 4, 4, 4, 4, 4, 4, 4, 4, 5, 3,
5, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4, 4, 4, 4, 5, 5, 5, 5, 5, 5, 5,
4, 5, 4, 5, 5, 5, 5, 5, 4, 4, 4, 4, 4, 5, 5, 4, 4, 4, 4, 5, 4, 5, 5,
4, 4, 4, 4, 4, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 3, 3, 4, 4, 5, 5, 5,
5, 5, 5, 5, 5, 3, 4, 4, 4, 4, 4, 3, 3, 3, 4, 4, 4, 4, 4, 4, 5, 5, 4,
7, 5, 5, 5, 4, 4, 4, 4, 4, 4, 4, 5, 4, 3, 3, 3, 3, 4, 4, 5, 5, 3, 5,
4, 5, 5, 4, 5, 5, 5, 5, 4, 3, 5, 4, 4, 3, 4, 3, 4, 5, 4, 4, 4, 4, 4,
5, 4, 4, 3, 4, 4, 4, 5, 4, 4, 4, 5, 3, 5, 3, 4, 5, 4, 4, 4, 5, 5, 5,
5, 4, 4, 4, 5, 5, 4, 5, 4, 4, 4, 4, 4, 4, 4, 4, 5, 4, 5, 5, 5, 4, 5,
5, 4, 4, 4, 4, 5, 3, 5, 5, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4, 4, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 5, 4, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 3, 4, 4, 4, 5, 4, 5, 4, 4, 5, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 5, 5, 5, 3, 3, 4, 4, 4, 5, 5, 5, 5,
4, 4, 4, 4, 5, 5, 4, 4, 4, 5, 4, 5, 4, 5, 5, 4, 5, 4, 5, 4, 5, 5, 4,
5, 5, 3, 5, 4, 4, 5, 5, 3, 4, 4, 4, 4, 5, 5, 4, 5, 5, 4, 4, 4, 4, 5,
4, 4, 4, 4, 4, 4, 4, 4, 4, 6, 3, 3, 3, 4, 4, 5, 4, 5, 4, 5, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 5, 4, 4,
3, 3, 3, 3, 5, 4, 4, 4, 4, 4, 4, 4, 5, 5, 5, 5, 3, 4, 4, 5, 5, 4, 4, 5,
5, 4, 4, 5, 4, 4, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 5, 4, 5,
4, 3, 5, 4, 5, 5, 5, 5, 3, 4, 5, 5, 5, 4, 3, 5, 5, 4, 4, 4, 4, 3, 3,
```

3, 4, 4, 3, 3, 5, 4, 4, 4, 4, 4, 4, 3, 4, 5, 5, 5, 3, 4, 4, 4, 4, 5,
5, 4, 3, 4, 4, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5,
4, 4, 3, 5, 5, 4, 4, 4, 4, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 3, 3, 4, 5,
4, 4, 5, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 5, 4, 4, 4, 5, 4, 4,
4, 4, 4, 4, 4, 3, 4, 4, 4, 4, 3, 4, 5, 5, 5, 4, 4, 4, 4, 4, 4, 5, 4,
4, 4, 4, 5, 4, 4, 5, 4, 3, 3, 4, 4, 4, 4, 4, 3, 4, 5, 5, 4, 4, 5, 4,
5, 4, 5, 5, 5, 5, 5, 5, 4, 3, 5, 4, 4, 4, 4, 5, 4, 4, 4, 4, 4, 5, 4,
4, 4, 4, 3, 5, 5, 5, 5, 3, 3, 4, 4, 5, 4, 3, 5, 5, 4, 4, 4, 3, 4, 5,
5, 5, 5, 4, 4, 5, 5, 4, 5, 4, 5, 4, 4, 5, 5, 5, 5, 5, 4, 5, 5, 5, 5,
3, 4, 3, 4, 4, 5, 5, 5, 3, 3, 5, 4, 4, 4, 4, 4, 4, 4, 4, 3, 3, 3, 3,
4, 5, 4, 5, 4, 4, 4, 4, 4, 4, 4, 4, 5, 4, 4, 5, 5, 4, 4, 4, 4, 5, 3,
5, 3, 4, 4, 5, 5, 4, 4, 5, 5, 4, 4, 4, 4, 4, 3, 5, 5, 4, 4, 5, 5, 4,
5, 4, 5, 4, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 4, 4, 3, 4, 4, 4, 4, 4, 3,
3, 4, 3, 4, 5, 4, 3, 5, 3, 5, 5, 5, 5, 4, 4, 4, 4, 4, 4, 3, 5, 4, 4,
5, 5, 5, 5, 4, 4, 5, 4, 5, 4, 4, 4, 3, 5, 4, 3, 3, 5, 5, 3, 3, 3, 3,
4, 4, 4, 4, 4, 4, 3, 4, 3, 4, 3, 4, 4, 4, 4, 3, 3, 3, 4, 4, 4, 4, 4,
4, 3, 4, 5, 3, 4, 4, 4, 3, 3, 3, 4, 4, 4, 4, 3, 3, 3, 3, 3, 4, 3, 4,
4, 4, 4, 4, 8, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 4, 5, 4, 4, 3, 4, 5,
4, 4, 4, 3, 4, 4, 4, 4, 3, 3, 3, 3, 4, 3, 5, 5, 5, 5, 4, 4, 3, 5, 5,
3, 5, 4, 4, 4, 5, 4, 0, 5, 5, 4, 4, 4, 4, 4, 4, 5, 4, 4, 4, 3, 5, 3,
5, 4, 5, 3, 4, 4, 4, 5, 4, 4, 5, 4, 4, 4, 4, 4, 4, 3, 4, 4, 4, 3, 3,
4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 4, 4,
4, 4, 5, 3, 3, 5, 4, 4, 5, 5, 3, 4, 4, 4, 5, 5, 4, 3, 7, 3, 5, 4, 5,
5, 4, 4, 4, 4, 4, 3, 5, 5, 4, 4, 4, 5, 5, 5, 5, 3, 4, 4, 4, 4, 4, 5,
4, 4, 5, 4, 5, 5, 5, 4, 4, 4, 4, 4, 4, 5, 4, 4, 4, 5, 3, 3, 4, 5, 5,
4, 5, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4, 5, 4, 4, 4, 4, 4, 3, 4, 3, 4, 4,
4, 3, 4, 5, 5, 5, 5, 5, 5, 4, 4, 4, 4, 4, 4, 5, 7, 5, 3, 3, 5, 4, 4,
4, 4, 4, 3, 3, 4, 4, 4, 4, 4, 4, 4, 3, 4, 3, 3, 4, 4, 5, 3, 4, 3, 3, 4,
4, 4, 5, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 3, 4, 4, 4, 4, 4, 5, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 5, 4, 3, 4, 4, 4, 3, 4, 5,
5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 3, 4, 3, 4, 4, 5, 3, 4,
5, 4, 4, 4, 3, 5, 5, 5, 5, 5, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4, 4,
4, 3, 4, 5, 5, 5, 5, 5, 4, 4, 4, 4, 5, 4, 4, 3, 3, 3, 4, 4, 4, 4, 3,
3, 4, 4, 4, 4, 4, 4, 4, 5, 4, 4, 4, 4, 4, 3, 3, 3, 4, 5, 5, 5, 5, 5, 5,
5, 5, 5, 5, 5, 4, 5, 5, 4, 5, 9, 9, 9, 4, 4, 5, 5, 5, 4, 4, 4, 4, 3,
5, 4, 4, 5, 5, 4, 4, 4, 3, 4, 3, 4, 3, 3, 3, 3, 4, 4, 4, 3, 3, 4, 5,
3, 3, 4, 4, 4, 4, 4, 4, 4, 5, 4, 4, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4,
4, 4, 4, 4, 3, 4, 3, 4, 4, 4, 4, 4, 4, 3, 3, 3, 4, 5, 3, 5, 4, 4, 3,
3, 5, 4, 4, 4, 4, 4, 5, 4, 4, 5, 5, 5, 4, 4, 4, 5, 5, 4, 4, 4, 4, 4,
4, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 5, 4, 4, 4, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 6, 4, 4, 4, 4, 5, 4, 4, 5, 4, 5, 4, 4,
4, 4, 5, 4, 5, 5, 5, 5, 4, 5, 4, 4, 4, 4, 3, 5, 5, 5, 5, 4, 4, 4, 4,
4, 5, 3, 3, 5, 3, 3, 3, 3, 4, 5, 5, 5, 4, 3, 4, 6, 3, 3, 5, 4, 3, 3,
3, 4, 3, 3, 3, 3, 5, 4, 6, 5, 7, 4, 4, 4, 4, 5, 5, 4, 4, 4, 5, 4, 4,
4, 4, 4, 4, 4, 3, 4, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4, 4, 4, 5, 4,
4, 5, 4, 4, 5, 5, 5, 5, 5, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4, 5, 4, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3,
4, 4, 4, 3, 4, 4, 3, 4, 5, 5, 3, 3, 4, 4, 4, 5, 5, 5, 3, 3, 3, 4, 3,

4, 4, 4, 5, 4, 4, 4, 4, 4, 4, 4, 4, 5, 3, 4, 4, 3, 4, 4, 4, 4, 4, 3,
5, 4, 4, 5, 4, 5, 4, 5, 5, 4, 4, 5, 5, 4, 5, 5, 5, 4, 4, 4, 4, 4, 4,
4, 3, 4, 3, 4, 3, 3, 4, 4, 4, 4, 4, 4, 3, 4, 4, 4, 4, 4, 4, 3, 4, 4,
5, 5, 4, 5, 3, 4, 4, 4, 4, 4, 4, 4, 4, 3, 3, 4, 4, 4, 3, 4, 4, 4, 5,
4, 4, 3, 4, 4, 4, 4, 3, 5, 5, 3, 4, 5, 4, 6, 4, 4, 4, 4, 4, 5, 5, 5,
5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4, 4, 5, 5, 4, 4, 3, 4, 4, 4, 4, 4,
4, 3, 4, 3, 4, 4, 4, 4, 4, 4, 4, 3, 5, 4, 4, 6, 4, 4, 3, 3, 4, 5, 5,
4, 4, 4, 3, 3, 4, 3, 4, 3, 3, 3, 3, 3, 4, 4, 4, 3, 4, 4, 4, 4, 3, 4,
5, 3, 3, 3, 4, 4, 4, 4, 3, 4, 3, 3, 5, 4, 4, 4, 4, 4, 4, 4, 4, 5, 4,
4, 4, 4, 4, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 3, 5, 5, 5, 5, 5,
5, 5, 3, 5, 5, 3, 3, 3, 5, 5, 4, 4, 4, 5, 3, 4, 4, 4, 4, 4, 4, 4, 3,
4, 5, 4, 4, 4, 5, 4, 4, 4, 4, 4, 3, 4, 4, 4, 4, 4, 4, 4, 3, 4, 4, 3,
4, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 4, 4, 4, 5, 4,
4, 5, 5, 3, 5, 5, 4, 4, 4, 3, 4, 4, 3, 4, 3, 3, 4, 4, 4, 4, 4, 4, 4,
4, 3, 5, 4, 4, 4, 5, 4, 3, 3, 3, 4, 3, 4, 5, 5, 4, 4, 3, 4, 4, 3, 4,
4, 5, 3, 3, 5, 5, 4, 3, 3, 4, 5, 3, 3, 3, 3, 4, 5, 3, 5, 3, 5, 4, 4,
4, 3, 3, 4, 4, 4, 3, 4, 4, 4, 4, 3, 3, 3, 4, 5, 4, 4, 4, 4, 4, 4, 4,
4, 4, 4, 3, 4, 4, 4, 4, 5, 5, 4, 5, 4, 4, 4, 3, 4, 4, 4, 4, 4, 5, 5,
5, 5, 5, 4, 4, 4, 5, 5, 5, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 5,
5, 5, 3, 4, 4, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 3, 4, 3, 5, 5,
4, 4, 3, 3, 4, 4, 4, 5, 4, 3, 3, 3, 3, 4, 4, 5, 4, 4, 4, 4, 4, 4, 3,
3, 3, 3, 4, 4, 4, 4, 3, 4, 4, 5, 4, 4, 5, 4, 4, 4, 4, 3, 4, 4, 4, 4,
3, 4, 4, 4, 3, 4, 5, 5, 4, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4, 3, 4, 4, 4, 4, 3, 4, 5, 4, 4, 4, 4, 4, 4, 3, 4, 4, 4, 4, 4, 4, 4,
4, 4, 3, 4, 6, 4, 3, 4, 4, 3, 4, 4, 3, 4, 4, 4, 4, 4, 4, 3, 3, 5, 3,
3, 3, 3, 4, 3, 5, 4, 3, 5, 4, 3, 3, 3, 4, 3, 4, 4, 4, 3, 4, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 4, 4, 0, 5, 4, 3, 5, 3, 3, 4, 4, 4, 4, 4, 3, 4,
4, 5, 0, 3, 3, 3, 5, 4, 3, 4, 3, 3, 4, 6, 3, 3, 4, 6, 5, 4, 4, 5, 4,
4, 4, 4, 4, 3, 4, 4, 4, 4, 4, 4, 4, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
3, 4, 3, 3, 3, 3, 3, 4, 4, 3, 4, 4, 3, 3, 3, 5, 3, 3, 3, 4, 3, 3, 3,
4, 4, 3, 4, 3, 4, 4, 3, 5, 3, 3, 3, 3, 4, 4, 5, 5, 5, 5, 3, 3, 4, 5,
3, 4, 3, 4, 4, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 4, 4, 4, 4,
4, 4, 5, 3, 3, 4, 5, 5, 5, 5, 4, 4, 5, 3, 4, 4, 4, 4, 3, 5, 5, 4, 3,
5, 5, 5, 5, 5, 4, 3, 3, 4, 4, 4, 4, 5, 5, 4, 4, 5, 4, 4, 4, 5, 4, 4,
3, 4, 3, 4, 3, 3, 3, 4, 4, 3, 5, 5, 3, 3, 3, 3, 4, 5, 5, 5, 6, 6, 5,
3, 3, 4, 5, 4, 6, 6, 4, 5, 5, 3, 5, 4, 5, 4, 3, 4, 4, 4, 5, 4, 4, 4,
4, 4, 4, 5, 4, 3, 4, 5, 4, 4, 4, 4, 3, 4, 4, 4, 4, 5, 5, 4, 4, 4, 4,
4, 4, 4, 4, 5, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 4, 4, 4, 4, 4, 4, 5,
5, 5, 5, 5, 5, 5, 4, 4, 4, 4, 4, 4, 4, 3, 4, 4, 4, 4, 5, 5, 5, 4, 4,
4, 3, 4, 4, 4, 3, 4, 5, 4, 3, 4, 4, 3, 4, 4, 4, 4, 5, 4, 4, 5, 4, 5,
4, 4, 4, 3, 4, 4, 4, 4, 4, 6, 4, 3, 4, 4, 4, 4, 4, 4, 3, 4, 4, 4, 3,
3, 3, 4, 5, 4, 4, 4, 4, 4, 3, 3, 3, 4, 5, 3, 3, 3, 3, 4, 5, 4, 4, 5,
5, 5, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4, 4,
12, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 3, 3, 7, 3, 3,
3, 7, 7, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4, 4, 4, 4, 4, 4, 4, 3, 4, 3,
3, 4, 4, 5, 4, 4, 3, 4, 4, 5, 5, 5, 4, 4, 3, 5, 5, 5, 3, 3, 4, 3, 4,
4, 4, 4, 4, 4, 3, 4, 4, 4, 4, 4, 3, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4, 3, 3, 3, 3, 3, 4, 4, 5, 5, 5, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5,

4, 3, 5, 3, 4, 4, 4, 4, 4, 5, 5, 5, 5, 5, 5, 5, 5, 5, 4, 4, 4, 5, 4,
3, 4, 3, 4, 4, 4, 5, 4, 4, 4, 3, 4, 4, 3, 4, 4, 4, 4, 4, 5, 4, 4, 3,
3, 4, 3, 4, 4, 4, 4, 4, 4, 5, 4, 3, 3, 3, 4, 4, 4, 4, 4, 5, 4, 4, 3,
3, 4, 4, 5, 4, 3, 3, 3, 4, 4, 3, 3, 4, 4, 4, 4, 3, 4, 4, 5, 5, 4, 6,
3, 4, 4, 3, 4, 4, 4, 4, 3, 5, 4, 4, 4, 5, 3, 3, 3, 3, 4, 4, 4, 4, 4,
4, 4, 4, 4, 5, 5, 5, 5, 4, 3, 3, 5, 4, 4, 3, 3, 4, 4, 4, 4, 5, 4, 4,
6, 4, 4, 4, 4, 4, 3, 4, 4, 3, 4, 4, 4, 3, 4, 4, 3, 5, 4, 3, 4, 4, 4,
4, 4, 4, 4, 4, 4, 3, 4, 3, 3, 5, 3, 4, 4, 4, 4, 4, 4, 4, 4, 5, 4, 4,
3, 3, 4, 4, 4, 4, 5, 4, 4, 4, 3, 4, 4, 5, 3, 4, 4, 5, 4, 4, 4, 4, 4,
3, 3, 3, 4, 4, 5, 5, 3, 3, 4, 4, 5, 5, 5, 5, 4, 4, 4, 4, 3, 4, 4, 5,
4, 4, 4, 6, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 4, 3, 4, 3, 3, 3,
4, 3, 5, 4, 4, 5, 3, 3, 5, 4, 4, 4, 4, 4, 4, 3, 3, 3, 3, 4, 4, 4, 3,
3, 3, 4, 3, 3, 4, 3, 4, 3, 5, 4, 3, 4, 3, 5, 3, 3, 5, 4, 4, 4, 4, 5,
4, 4, 3, 4, 3, 4, 5, 6, 3, 7, 4, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4,
6, 4, 5, 4, 4, 4, 4, 3, 3, 4, 4, 4, 4, 4, 4, 3, 4, 4, 4, 4, 4, 4, 4,
4, 3, 4, 4, 3, 4, 5, 4, 4, 4, 5, 5, 5, 4, 3, 4, 4, 3, 4, 5, 3, 3, 4,
4, 4, 4, 4, 4, 4, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 3, 4, 4, 4, 3, 4,
3, 4, 4, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 8, 3, 3, 3, 4,
4, 3, 4, 4, 3, 4, 5, 4, 4, 5, 3, 4, 4, 4, 4, 3, 3, 4, 4, 4, 4, 6, 4,
3, 3, 3, 4, 4, 4, 4, 4, 5, 4, 4, 4, 4, 4, 4, 3, 4, 4, 4, 5, 4, 4, 4,
4, 4, 4, 3, 4, 4, 5, 5, 5, 4, 4, 4, 3, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4,
4, 4, 4, 4, 5, 4, 4, 3, 4, 3, 4, 5, 4, 4, 4, 4, 5, 5, 4, 4, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4, 3, 3, 3,
4, 4, 3, 3, 4, 4, 4, 4, 4, 5, 5, 4, 3, 4, 6, 4, 4, 6, 4, 4, 4, 7, 4,
4, 6, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 4, 4, 4, 4, 4, 4, 5, 5, 4, 4, 4,
4, 4, 4, 3, 4, 4, 4, 4, 3, 4, 3, 4, 3, 3, 3, 3, 3, 4, 4, 5, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4, 4, 4,
4, 5, 4, 4, 4, 4, 4, 3, 4, 4, 4, 4, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4, 5, 4, 3, 5, 3, 4, 4, 4, 3, 5, 5, 5, 4, 4, 3, 3, 3, 3, 3, 3, 4, 3,
4, 3, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 3, 4, 4, 4, 5, 5, 4,
4, 5, 5, 5, 4, 4, 4, 5, 5, 5, 5, 3, 5, 4, 4, 4, 4, 5, 5, 5, 5, 5, 5,
5, 5, 5, 5, 5, 5, 4, 4, 5, 5, 5, 4, 4, 5, 4, 5, 5, 5, 5, 4, 5, 5, 5,
5, 3, 4, 3, 4, 3, 3, 4, 4, 5, 4, 3, 5, 5, 5, 5, 4, 4, 5, 3, 4, 4, 4,
5, 4, 4, 4, 4, 4, 4, 5, 5, 4, 4, 5, 5, 5, 5, 4, 5, 5, 5, 5, 4, 5, 4,
4, 4, 4, 5, 5, 5, 5, 5, 5, 5, 3, 4, 5, 4, 4, 4, 5, 4, 5, 4, 4, 4, 4,
4, 4, 4, 3, 4, 3, 4, 5, 4, 4, 3, 3, 4, 5, 4, 5, 3, 4, 4, 4, 4, 4, 4,
4, 3, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4, 5, 4, 4, 5, 5, 5, 5, 5, 5,
5, 5, 5, 5, 5, 5, 4, 5, 4, 4, 4, 4, 4, 4, 4, 4, 5, 5, 4, 4, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 5, 5, 4, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4, 5, 5, 6, 4, 4, 4, 4, 4, 4, 4, 4, 6, 4, 4, 4, 4, 3, 4, 4, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4, 5, 4,
5, 5, 5, 4, 4, 4, 4, 6, 4, 4, 4, 3, 3, 3, 4, 4, 5, 5, 5, 4, 5, 5, 5,
4, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4, 4, 6, 3, 3, 4,
7, 3, 4, 4, 3, 4, 5, 5, 3, 3, 4, 4, 4, 4, 4, 3, 3, 4, 4, 3, 4, 4, 3,
4, 4, 5, 4, 4, 5, 4, 4, 4, 5, 4, 3, 4, 4, 4, 4, 4, 5, 5, 5, 5, 4, 4,
4, 4, 4, 4, 5, 5, 3, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4, 4, 4, 3, 4, 4, 5, 5, 4, 4,
4, 4, 4, 4, 4, 5, 4, 5, 4, 5, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 3, 3,
4, 4, 4, 3, 4, 4, 5, 4, 4, 4, 4, 4, 4, 5, 3, 5, 4, 4, 4, 5, 3, 4, 4,

4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5,
4, 4, 4, 5, 4, 4, 4, 4, 5, 5, 4, 4, 5, 5, 4, 4, 5, 4, 5, 5, 5, 4, 4,
5, 4, 5, 5, 4, 4, 4, 4, 4, 4, 4, 3, 3, 4, 4, 3, 3, 4, 4, 4, 5, 4, 5,
5, 5, 5, 5, 5, 4, 4, 4, 5, 4, 4, 5, 5, 5, 4, 4, 4, 4, 4, 4, 5, 4, 4,
5, 5, 5, 5, 5, 4, 5, 4, 4, 4, 4, 4, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3,
3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 3, 4,
4, 4, 4, 5, 4, 4, 4, 4, 4, 6, 4, 4, 4, 4, 4, 4, 4, 4, 5, 4, 5, 5, 4,
3, 4, 3, 4, 4, 4, 5, 3, 4, 4, 4, 4, 5, 4, 4, 4, 4, 4, 3, 5, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 5, 6, 4, 4, 5, 4, 4, 5, 3, 5, 3, 3, 4, 4, 4, 4,
3, 4, 5, 4, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4, 5, 3, 4, 4, 4, 5, 3, 3, 4, 4, 4,
4, 4, 4, 4, 4, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4, 4, 4, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 3, 3, 4, 4, 4, 3, 4, 4, 5, 4, 3, 4, 4,
4, 4, 4, 4, 4, 4, 5, 4, 4, 4, 4, 4, 4, 3, 4, 4, 4, 4, 4, 4, 3, 3, 4,
4, 4, 4, 4, 4, 4, 5, 4, 4, 4, 3, 4, 4, 4, 4, 4, 4, 3, 4, 3, 3, 3, 4,
4, 4, 4, 4, 4, 3, 4, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 3, 4, 4, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 3, 4, 4, 3, 4, 4, 3, 4, 4, 3, 4, 4, 3, 4, 4, 4,
4, 4, 3, 4, 4, 4, 4, 3, 4, 4, 4, 4, 4, 4, 3, 3, 3, 3, 3, 4, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 3, 3, 3,
3, 4, 3, 4, 4, 4, 3, 4, 4, 4, 3, 3, 4, 4, 4, 3, 4, 4, 5, 4, 4, 4, 4,
3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4, 4, 4, 4, 4, 4,
3, 3, 4, 4, 4, 4, 4, 3, 4, 4, 4, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4, 3, 4, 4, 4, 3, 5, 5, 3, 4, 4, 4, 4, 4, 4, 3, 4, 3, 3, 3, 3, 4, 3,
3, 3, 4, 4, 4, 3, 5, 5, 4, 4, 4, 4, 4, 4, 4, 3, 4, 3, 4, 4, 4, 4, 3,
4, 4, 4, 5, 4, 4, 4, 4, 4, 4, 3, 4, 4, 4, 5, 4, 5, 5, 5, 5, 5, 3, 3,
3, 3, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4, 4, 4, 4, 3, 4,
4, 4, 4, 4, 4, 3, 4, 3, 3, 4, 3, 4, 4, 3, 3, 4, 4, 4, 4, 3, 4, 3, 4,
3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 3, 5, 5, 5, 6, 6, 3, 4, 4, 5, 3, 4, 4,
4, 4, 3, 4, 4, 4, 4, 4, 5, 5, 5, 5, 5, 5, 4, 3, 4, 4, 3, 4, 4, 4, 4,
4, 4, 4, 3, 3, 4, 4, 4, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4, 3, 4, 3, 4, 4, 4, 4,
4, 4, 4, 4, 4, 4, 3, 4, 4, 4, 5, 4, 3, 4, 4, 4, 4, 3, 3, 3, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 3, 3, 4, 3, 3, 4, 4, 4, 4, 4, 3, 4, 4, 3, 3, 3,
3, 4, 3, 3, 4, 5, 3, 4, 3, 4, 4, 4, 3, 3, 3, 4, 3, 4, 4, 4, 3, 3, 3,
3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4, 4, 4, 4,
4, 4, 7, 6, 4, 4, 4, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 3, 3, 3, 4,
3, 4, 4, 3, 3, 4, 5, 4, 4, 3, 3, 3, 4, 4, 3, 3, 4, 4, 4, 3, 4, 4, 3,
3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 6, 6, 3, 4, 6, 4, 4, 4, 4, 4, 3, 3, 3,
4, 7, 3, 3, 3, 4, 4, 4, 4, 3, 4, 3, 4, 4, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4,
5, 5, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 5, 4, 4, 5, 5, 4, 3, 4, 4,
3, 3, 4, 3, 3, 4, 4, 5, 4, 4, 4, 4, 4, 4, 4, 3, 3, 4, 4, 3, 4, 4, 4,
3, 3, 4, 3, 4, 4, 4, 4, 4, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4, 4,
3, 4, 5, 3, 4, 4, 4, 4, 5, 4, 3, 3, 4, 4, 3, 4, 5, 4, 3, 4, 4, 4, 4, 4,
4, 4, 3, 4, 4, 4, 4, 3, 5, 4, 4, 5, 3, 4, 4, 4, 3, 4, 4, 4, 3, 4, 4, 4,
3, 5, 3, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 3, 3, 3, 3, 5, 4, 4, 5, 4, 3, 4, 4, 4, 4,
4, 4, 5, 4, 4, 4, 4, 4, 5, 3, 4, 4, 5, 5, 5, 4, 4, 5, 4, 5, 3, 4, 4, 4,
4, 4, 4, 3, 5, 4, 4, 4, 4, 3, 4, 4, 4, 4, 3, 3, 4, 4, 4, 4, 4, 4, 4, 5,

5, 5, 4, 4, 4, 4, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4, 3, 3, 3, 4,
4, 4, 4, 4, 4, 4, 3, 3, 5, 4, 3, 3, 3, 5, 4, 3, 4, 3, 4, 4, 4, 4, 4,
4, 4, 4, 3, 4, 3, 3, 3, 4, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3,
4, 4, 3, 4, 4, 4, 4, 3, 4, 3, 4, 4, 3, 4, 5, 4, 3, 4, 4, 4, 5, 4, 4,
4, 3, 5, 4, 4, 4, 3, 4, 3, 4, 4, 4, 4, 5, 4, 4, 4, 3, 4, 4, 3, 3, 3,
5, 3, 3, 5, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 3, 4, 3, 3, 4, 4, 4,
4, 3, 3, 3, 4, 4, 4, 4, 4, 3, 5, 3, 4, 4, 3, 4, 4, 4, 3, 3, 3, 3, 3,
4, 3, 4, 4, 4, 4, 8, 4, 3, 3, 4, 3, 4, 4, 3, 3, 5, 4, 4, 4, 4, 4, 3,
3, 4, 3, 4, 4, 4, 3, 5, 4, 4, 4, 4, 4, 4, 3, 5, 3, 4, 4, 4, 4, 4, 4,
3, 3, 4, 4, 4, 4, 4, 3, 4, 4, 4, 5, 4, 4, 4, 4, 4, 4, 4, 3, 3, 4, 4,
5, 4, 4, 3, 4, 3, 3, 3, 4, 5, 5, 5, 4, 3, 4, 3, 3, 4, 3, 4, 3, 4, 4,
4, 4, 4, 3, 4, 4, 4, 4, 4, 5, 4, 4, 4, 5, 5, 5, 4, 3, 3, 4, 3, 3, 3,
3, 3, 4, 3, 4, 7, 5, 4, 3, 4, 5, 3, 5, 4, 4, 4, 4, 4, 4, 3, 4, 4, 4,
4, 3, 3, 4, 4, 3, 3, 3, 4, 4, 5, 4, 4, 4, 4, 5, 4, 3, 4, 4, 4, 4, 3,
5, 5, 4, 4, 4, 4, 3, 4, 3, 4, 3, 3, 4, 4, 4, 4, 4, 4, 3, 4, 3, 4, 5,
3, 4, 4, 4, 3, 3, 4, 4, 4, 5, 3, 5, 5, 3, 4, 6, 4, 4, 4, 5, 5, 5, 5,
4, 4, 3, 4, 4, 4, 4, 4, 4, 3, 4, 3, 5, 4, 6, 5, 4, 3, 4, 3, 3, 3, 4,
4, 3, 4, 5, 3, 3, 4, 3, 4, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4,
4, 4, 5, 5, 5, 5, 3, 3, 3, 4, 4, 5, 3, 4, 4, 4, 4, 4, 4, 4, 3, 4, 4,
4, 3, 4, 3, 4, 4, 4, 4, 4, 3, 3, 4, 5, 5, 4, 4, 3, 3, 4, 4, 3, 3, 3,
3, 4, 5, 3, 5, 3, 5, 4, 4, 3, 3, 4, 4, 4, 3, 3, 3, 5, 4, 4, 4, 4, 4,
4, 3, 4, 4, 4, 5, 5, 4, 3, 4, 5, 5, 5, 5, 5, 5, 3, 3, 3, 4, 4, 3, 3,
3, 4, 3, 3, 4, 4, 4, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4, 4,
4, 4, 4, 4, 4, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 6, 4, 3, 4, 3, 3, 4, 4,
3, 5, 3, 3, 4, 4, 4, 4, 4, 4, 3, 3, 4, 4, 3, 3, 4, 3, 4, 6, 4, 6, 4,
4, 3, 4, 4, 4, 4, 3, 4, 3, 4, 4, 3, 3, 4, 4, 4, 4, 4, 5, 3, 4, 4, 4,
4, 4, 4, 4, 4, 3, 5, 5, 5, 6, 6, 3, 3, 4, 4, 5, 3, 4, 4, 4, 3, 4, 4,
5, 5, 4, 4, 4, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 5, 5, 5, 5, 5, 4,
4, 4, 4, 5, 4, 3, 5, 4, 4, 3, 4, 4, 4, 4, 4, 4, 4, 3, 3, 4, 3, 3, 4,
4, 4, 4, 4, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4,
3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 5, 4, 4, 3,
4, 4, 5, 3, 4, 4, 4, 4, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 4, 4,
4, 5, 4, 4, 4, 3, 4, 4, 4, 5, 4, 3, 4, 4, 5, 4, 4, 5, 4, 4, 4, 4, 3,
3, 3, 3, 3, 3, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 6, 3, 3, 4, 4, 4,
3, 3, 3, 4, 4, 4, 4, 3, 3, 4, 3, 4, 4, 3, 5, 4, 4, 4, 4, 4, 4, 3, 5,
4, 4, 4, 4, 4, 4, 3, 3, 3, 3, 4, 3, 3, 3, 4, 5, 3, 3, 4, 4, 5, 3, 6,
3, 3, 4, 6, 4, 4, 3, 3, 3, 4, 4, 3, 4, 3, 4, 3, 3, 4, 4, 3, 3, 4, 3,
4, 3, 4, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 4, 3, 3, 3, 4, 3, 4, 5, 3,
4, 6, 4, 3, 4, 4, 4, 3, 4, 5, 5, 5, 4, 4, 4, 3, 3, 4, 4, 4, 4, 4, 4,
4, 5, 5, 4, 4, 4, 4, 4, 4, 4, 3, 3, 4, 4, 4, 7, 6, 4, 4, 4, 4, 3, 3,
3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 3, 3, 3, 3, 3, 3, 3, 3, 4, 3, 4,
3, 4, 3, 3, 4, 4, 5, 4, 4, 4, 5, 4, 5, 4, 4, 5, 5, 5, 5, 3, 3, 3, 4,
4, 3, 3, 4, 4, 4, 5, 4, 3, 4, 4, 4, 4, 4, 4, 3, 4, 5, 4, 3, 3, 4, 4,
4, 4, 4, 4, 4, 3, 4, 4, 4, 4, 4, 3, 4, 4, 4, 4, 4, 4, 6, 4, 4, 4, 4,
6, 3, 4, 4, 3, 4, 4, 6, 4, 4, 4, 4, 4, 4, 4, 4, 3, 3, 3, 4, 7, 3, 4, 5,
3, 3, 4, 4, 4, 3, 4, 3, 4, 4, 3, 4, 4, 4, 3, 5, 4, 3, 4, 4, 4, 3, 4,
4, 4, 5, 3, 3, 3, 4, 4, 3, 3, 5, 5, 5, 4, 4, 4, 5, 5, 4, 4, 4, 4, 4,
4, 4, 4, 5, 5, 4, 3, 3, 4, 4, 4, 4, 4, 3, 5, 3, 3, 4, 4, 5, 5, 4, 3,
4, 4, 4, 3, 3, 4, 3, 4, 4, 4, 4, 4, 3, 4, 5, 3, 4, 4, 5, 4, 4, 4, 5,

5, 4, 3, 4, 4, 4, 4, 4, 4, 5, 3, 5, 3, 4, 4, 3, 5, 5, 4, 4, 4, 4, 4,
4, 3, 4, 3, 3, 4, 3, 3, 4, 3, 5, 4, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4, 4, 4, 4, 3, 3, 4, 5, 4, 4, 4, 4, 3, 4, 4, 4, 4, 4, 4, 3, 5, 5, 4,
4, 3, 5, 4, 5, 5, 4, 3, 4, 4, 4, 5, 4, 3, 4, 3, 4, 4, 3, 4, 5, 4, 3,
4, 3, 4, 4, 4, 4, 4, 4, 3, 4, 5, 5, 4, 4, 4, 4, 4, 3, 4, 4, 4, 4, 5,
4, 4, 3, 5, 5, 5, 5, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 5, 5, 3,
5, 4, 4, 5, 4, 3, 4, 4, 4, 5, 3, 5, 4, 5, 5, 4, 5, 5, 4, 4, 3, 5, 5,
3, 5, 4, 3, 4, 4, 4, 4, 4, 3, 5, 5, 5, 4, 4, 5, 4, 4, 4, 4, 4, 5, 4,
4, 4, 4, 4, 4, 4, 3, 4, 4, 4, 4, 3, 3, 3, 4, 4, 5, 5, 5, 5, 5, 5, 5,
5, 3, 4, 4, 4, 4, 3, 3, 3, 4, 4, 4, 5, 5, 7, 5, 5, 5, 4, 4, 4, 4, 4,
5, 4, 3, 3, 3, 3, 4, 4, 3, 5, 4, 5, 5, 4, 5, 5, 5, 5, 4, 3, 5, 4, 3,
3, 4, 4, 4, 4, 4, 4, 5, 4, 4, 3, 4, 4, 5, 3, 5, 3, 4, 5, 4, 4, 5, 5,
5, 4, 4, 4, 5, 4, 5, 4, 4, 4, 4, 4, 4, 5, 4, 4, 3, 5, 5, 4, 4, 4, 4,
4, 4, 4, 4, 4, 4, 3, 4, 4, 4, 4, 4, 4, 3, 5, 4, 4, 4, 5, 3, 4, 4, 5,
4, 5, 4, 4, 5, 4, 4, 4, 5, 5, 5, 3, 3, 4, 4, 4, 5, 5, 4, 4, 4, 4, 5,
4, 4, 4, 5, 5, 5, 5, 4, 5, 5, 5, 5, 5, 5, 3, 4, 4, 4, 4, 4, 4, 4, 4,
6, 3, 3, 3, 4, 4, 5, 4, 4, 4, 4, 4, 4, 3, 4, 3, 3, 3, 5, 5, 5, 5, 3,
5, 5, 4, 4, 5, 4, 4, 4, 4, 4, 4, 4, 4, 5, 5, 4, 5, 4, 3, 5, 4, 5, 5,
5, 3, 5, 5, 5, 5, 4, 4, 3, 3, 3, 3, 5, 4, 4, 4, 4, 3, 4, 5, 5, 5, 3,
5, 5, 4, 4, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 4, 3, 4, 3, 3, 3, 4,
4, 3, 3, 4, 5, 5, 5, 4, 3, 4, 4, 4, 5, 4, 4, 4, 4, 4, 4, 3, 4, 4, 3,
4, 5, 5, 5, 4, 4, 5, 4, 4, 4, 4, 5, 4, 3, 4, 3, 4, 4, 4, 5, 5, 5, 4,
3, 4, 4, 4, 5, 4, 3, 5, 5, 4, 4, 5, 4, 3, 5, 5, 4, 4, 4, 3, 4, 5, 5,
5, 5, 4, 4, 5, 5, 4, 5, 5, 4, 4, 5, 5, 4, 5, 5, 5, 5, 3, 4, 3, 3, 3,
3, 3, 3, 5, 5, 4, 4, 4, 4, 4, 4, 3, 3, 4, 4, 5, 4, 4, 5, 4, 5, 5, 4,
5, 5, 4, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 4, 4, 3, 4, 4, 4, 3, 3, 4, 3,
5, 4, 3, 5, 4, 3, 5, 5, 5, 5, 5, 4, 4, 4, 4, 3, 4, 3, 3, 3, 3, 3, 3,
4, 4, 4, 4, 4, 4, 3, 4, 3, 4, 3, 4, 4, 4, 4, 3, 3, 3, 4, 4, 4, 4, 4,
4, 3, 4, 5, 3, 4, 4, 3, 4, 4, 4, 3, 3, 3, 3, 3, 4, 3, 4, 4, 4, 4, 8,
3, 4, 4, 4, 3, 5, 4, 4, 4, 3, 3, 3, 4, 3, 5, 5, 5, 5, 4, 4, 3, 5, 5,
3, 5, 4, 4, 5, 4, 4, 4, 4, 4, 4, 4, 4, 3, 5, 3, 5, 5, 3, 4, 4, 5, 5,
4, 4, 3, 3, 3, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 4, 3, 3, 5, 4, 3, 4, 4, 4, 5, 5, 4, 3, 3, 5, 4,
5, 5, 4, 4, 4, 3, 5, 5, 4, 4, 5, 5, 5, 5, 3, 4, 4, 4, 4, 4, 5, 4, 5,
4, 5, 5, 5, 4, 4, 4, 4, 4, 4, 5, 4, 4, 4, 3, 3, 4, 4, 4, 4, 4, 3, 5,
4, 3, 3, 4, 5, 5, 4, 4, 4, 4, 5, 5, 3, 4, 4, 3, 3, 4, 4, 4, 4, 4, 4,
3, 4, 3, 3, 4, 5, 3, 4, 3, 3, 4, 4, 4, 5, 3, 4, 4, 4, 4, 4, 4, 4, 5,
3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4, 3, 4, 4, 4, 3,
4, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 3, 4, 4, 3, 4, 4, 3, 5,
5, 5, 5, 5, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4, 5, 5, 5, 5, 4, 4, 4, 4, 3,
3, 4, 4, 4, 4, 3, 3, 4, 4, 4, 5, 4, 4, 4, 4, 4, 3, 3, 4, 5, 5, 5, 5,
5, 5, 5, 5, 5, 5, 4, 5, 4, 5, 9, 9, 9, 5, 5, 5, 4, 3, 4, 5, 4, 4, 4,
3, 3, 3, 3, 3, 4, 4, 4, 3, 3, 4, 3, 3, 4, 4, 4, 4, 4, 3, 3, 3, 3, 4,
3, 4, 4, 4, 4, 4, 3, 3, 4, 3, 3, 5, 4, 4, 4, 4, 4, 5, 4, 4, 5, 4, 4,
4, 4, 4, 4, 4, 5, 5, 4, 4, 4, 4, 4, 4, 6, 4, 4, 4, 5, 4, 4, 4, 4, 5,
4, 5, 5, 5, 5, 4, 4, 4, 3, 5, 5, 5, 4, 4, 3, 5, 3, 3, 4, 5, 5, 5, 4,
3, 6, 3, 3, 5, 4, 3, 3, 3, 4, 3, 3, 3, 3, 4, 5, 7, 4, 4, 4, 5, 5, 4,
5, 4, 4, 4, 4, 4, 4, 3, 4, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4, 4, 5,
4, 4, 5, 4, 4, 5, 5, 5, 5, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,

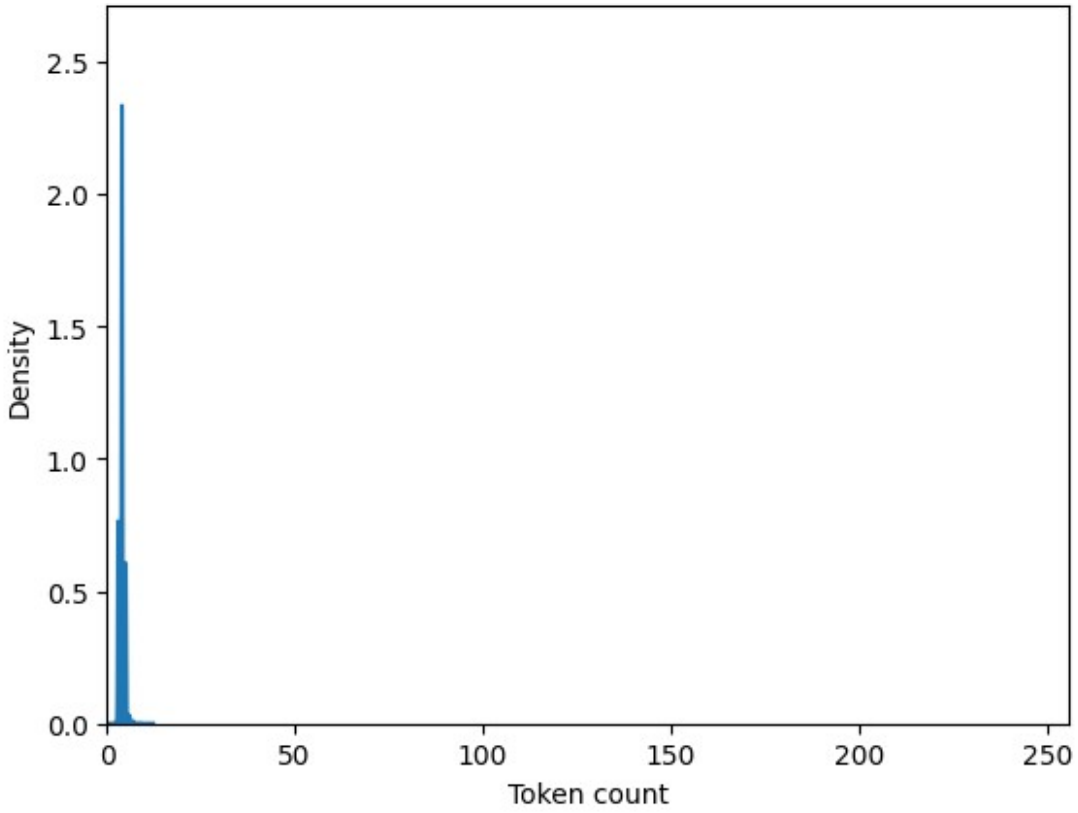
4, 5, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4, 4, 4, 3,
4, 4, 3, 3, 3, 4, 4, 5, 5, 5, 3, 3, 3, 3, 4, 4, 5, 4, 4, 4, 4, 4, 5,
3, 4, 3, 4, 4, 4, 4, 3, 4, 4, 5, 4, 4, 5, 5, 4, 5, 5, 4, 5, 5, 5, 4,
4, 4, 4, 4, 4, 4, 3, 4, 3, 4, 3, 3, 4, 4, 4, 4, 4, 4, 3, 4, 4, 4, 4,
3, 4, 4, 5, 5, 3, 4, 4, 4, 4, 4, 4, 4, 3, 3, 4, 4, 4, 3, 4, 4, 5, 4,
4, 3, 4, 4, 4, 4, 3, 5, 5, 3, 4, 5, 6, 4, 4, 4, 4, 4, 5, 5, 5, 5, 4,
4, 4, 4, 4, 4, 4, 3, 4, 4, 4, 3, 4, 4, 4, 4, 4, 4, 3, 4, 3, 4, 4, 4,
3, 5, 4, 4, 6, 4, 4, 3, 3, 4, 5, 4, 4, 4, 3, 3, 4, 3, 3, 3, 3, 3, 3,
4, 4, 3, 4, 4, 4, 4, 3, 4, 5, 3, 3, 3, 4, 4, 3, 4, 3, 3, 5, 4, 4, 4,
4, 4, 4, 4, 4, 5, 4, 4, 4, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 3,
5, 5, 5, 5, 5, 5, 3, 5, 5, 3, 3, 3, 5, 5, 4, 4, 4, 5, 3, 4, 4, 4, 4,
3, 5, 4, 4, 4, 4, 4, 3, 4, 4, 4, 4, 4, 4, 3, 4, 3, 4, 4, 3, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 4, 4, 5, 5, 3, 5, 4, 4,
4, 3, 4, 4, 3, 4, 3, 3, 4, 4, 4, 4, 4, 4, 3, 5, 4, 4, 5, 4, 3, 3, 3,
3, 4, 5, 5, 4, 4, 3, 4, 4, 3, 4, 4, 3, 3, 3, 3, 4, 5, 3, 3, 3, 3, 4,
5, 3, 5, 3, 5, 4, 4, 3, 3, 4, 4, 4, 3, 4, 4, 4, 4, 3, 3, 3, 5, 4, 4,
4, 4, 4, 4, 4, 4, 4, 3, 4, 4, 4, 4, 5, 5, 4, 5, 4, 4, 3, 4, 4, 4, 4,
4, 5, 5, 5, 5, 5, 4, 4, 4, 5, 5, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4,
4, 5, 5, 5, 3, 4, 4, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 3, 5, 5,
4, 3, 3, 4, 4, 5, 4, 3, 3, 3, 4, 5, 4, 4, 4, 4, 4, 4, 3, 3, 3, 3, 4,
4, 4, 4, 3, 4, 5, 4, 4, 4, 4, 3, 4, 4, 4, 3, 4, 4, 3, 4, 4, 4, 4, 4,
4, 4, 4, 4, 4, 4, 3, 4, 4, 4, 3, 4, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4, 4, 4, 4, 4, 4, 6, 4, 3, 4, 4, 3, 4, 4, 3, 4, 4, 4, 4, 3, 3, 5, 3,
3, 3, 3, 4, 3, 5, 4, 3, 5, 3, 3, 3, 4, 3, 4, 4, 4, 3, 4, 4, 4, 4, 4,
4, 4, 4, 5, 4, 3, 5, 3, 3, 4, 4, 4, 3, 4, 5, 3, 3, 3, 5, 4, 3, 4, 3,
3, 6, 3, 3, 4, 6, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4, 4, 4, 4, 4, 4, 3, 3, 3, 3,
3, 3, 3, 3, 3, 3, 3, 3, 3, 4, 3, 3, 3, 3, 3, 4, 4, 3, 4, 4, 3, 3, 3,
5, 3, 3, 3, 4, 3, 3, 3, 3, 3, 4, 4, 3, 5, 3, 3, 3, 3, 4, 4, 5, 5, 5,
3, 3, 4, 5, 3, 4, 3, 4, 4, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 4,
4, 4, 4, 5, 4, 5, 5, 5, 5, 4, 4, 5, 4, 4, 4, 5, 4, 3, 5, 4, 3, 3, 4,
4, 4, 5, 5, 4, 4, 5, 4, 4, 5, 4, 4, 3, 4, 3, 4, 3, 3, 3, 4, 4, 3, 5,
5, 3, 3, 3, 3, 4, 5, 5, 5, 6, 6, 3, 3, 4, 5, 4, 6, 6, 5, 5, 3, 5, 4,
5, 4, 3, 4, 4, 4, 5, 4, 4, 5, 3, 4, 5, 4, 4, 4, 4, 3, 4, 4, 4, 4, 5,
5, 4, 4, 4, 4, 4, 4, 4, 5, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5,
5, 5, 5, 5, 5, 5, 4, 4, 4, 4, 4, 4, 3, 4, 4, 5, 5, 5, 4, 4, 4, 3, 4,
4, 4, 5, 4, 3, 4, 4, 3, 4, 4, 5, 4, 4, 5, 4, 4, 4, 4, 3, 4, 4, 4, 4,
6, 3, 4, 4, 4, 4, 4, 3, 4, 4, 3, 3, 3, 4, 5, 4, 4, 4, 4, 4, 3, 3, 3,
4, 5, 3, 3, 3, 3, 4, 5, 4, 4, 5, 5, 5, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 4, 3, 4, 4, 12, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4, 4, 4, 3, 3, 3, 7, 3, 3, 3, 7, 7, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4, 4, 4, 4, 4,
4, 3, 4, 3, 3, 4, 4, 5, 4, 4, 3, 4, 4, 5, 5, 4, 3, 5, 5, 5, 3, 3, 4,
3, 4, 4, 4, 4, 4, 4, 3, 4, 3, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 3, 3, 3,
3, 4, 4, 5, 5, 5, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 4, 3, 5, 3, 4,
4, 4, 4, 5, 5, 5, 5, 5, 5, 5, 4, 4, 4, 5, 4, 3, 4, 3, 4, 4, 4, 5, 4,
4, 3, 4, 4, 5, 4, 4, 3, 3, 4, 3, 4, 4, 4, 4, 4, 4, 5, 3, 3, 3, 4, 4,
4, 5, 4, 4, 3, 3, 4, 4, 5, 4, 3, 3, 3, 4, 4, 3, 3, 4, 4, 4, 4, 3, 4,
4, 4, 6, 3, 4, 4, 3, 4, 4, 4, 4, 3, 5, 4, 4, 4, 3, 3, 3, 3, 4, 4, 4,
4, 4, 4, 4, 4, 4, 5, 4, 3, 3, 4, 4, 3, 3, 5, 4, 4, 6, 4, 4, 4, 4, 4,

3, 4, 3, 4, 4, 4, 3, 4, 4, 3, 5, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4,
3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 5, 4, 3, 3, 4, 4, 4, 4, 4, 4, 3, 4,
3, 4, 4, 5, 4, 4, 4, 4, 4, 3, 3, 3, 4, 4, 5, 5, 3, 3, 4, 5, 5, 4, 4,
4, 4, 3, 4, 5, 4, 4, 4, 6, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 3, 3,
3, 4, 3, 5, 4, 4, 5, 3, 3, 5, 4, 4, 4, 4, 4, 4, 3, 3, 3, 3, 4, 3, 3,
4, 3, 3, 4, 4, 3, 4, 3, 4, 3, 5, 3, 3, 5, 4, 4, 4, 5, 4, 3, 4, 3, 4,
6, 3, 7, 4, 3, 3, 4, 4, 4, 4, 4, 4, 3, 4, 6, 4, 5, 4, 4, 3, 3, 4, 4,
4, 4, 4, 4, 3, 4, 4, 4, 4, 4, 3, 4, 4, 3, 4, 5, 4, 4, 5, 5, 5, 4, 3,
4, 4, 3, 4, 4, 4, 4, 4, 4, 3, 3, 4, 4, 4, 4, 4, 4, 3, 3, 4, 4, 4, 3,
4, 3, 4, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 8, 3, 3, 3, 4, 3,
4, 4, 3, 4, 5, 4, 4, 5, 3, 4, 4, 4, 4, 3, 3, 4, 4, 6, 4, 3, 3, 3, 4,
4, 4, 4, 5, 4, 4, 3, 4, 4, 4, 4, 4, 4, 4, 3, 4, 4, 5, 5, 5, 4, 4, 4,
3, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 4, 4, 3, 4, 3, 4, 5,
4, 4, 4, 4, 5, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4, 4, 4, 3, 4, 3, 3, 4, 4, 3, 3, 4, 4, 4, 4, 4, 5, 5, 4, 3, 6, 4, 4,
6, 4, 4, 4, 7, 4, 4, 6, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5,
5, 4, 4, 4, 4, 4, 4, 4, 3, 4, 4, 4, 3, 3, 4, 3, 3, 3, 3, 3, 4, 4, 5, 4,
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4, 4,
4, 4, 5, 4, 4, 4, 4, 4, 3, 4, 4, 4, 4, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4, 4, 5, 4, 3, 5, 3, 4, 4, 4, 3, 5, 5, 5, 4, 3, 3, 3, 3, 3, 3, 4, 3,
4, 3, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 3, 4, 4, 4, 5, 5, 4, 4,
5, 5, 5, 4, 4, 4, 5, 3, 5, 4, 4, 4, 4, 5, 5, 5, 5, 5, 5, 5, 5, 4, 4,
5, 5, 5, 5, 4, 5, 5, 5, 5, 4, 5, 5, 5, 5, 3, 3, 4, 3, 3, 4, 4, 5, 4,
3, 5, 5, 5, 5, 3, 4, 4, 4, 5, 4, 4, 4, 4, 4, 4, 5, 4, 4, 5, 5, 4, 5,
5, 5, 5, 4, 5, 4, 4, 4, 4, 5, 5, 5, 5, 5, 5, 3, 4, 5, 4, 4, 4, 5, 4,
5, 4, 4, 4, 4, 4, 3, 4, 3, 4, 5, 4, 4, 3, 3, 4, 4, 5, 3, 4, 4, 4, 4,
3, 4, 3, 4, 4, 4, 4, 4, 4, 3, 4, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 4, 5,
4, 4, 4, 4, 4, 4, 4, 4, 5, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 4,
4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 5, 6, 4, 4, 4, 4, 4, 4, 6, 3, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4, 5,
4, 4, 4, 6, 4, 3, 3, 3, 4, 4, 5, 5, 5, 4, 5, 5, 5, 4, 5, 4, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 6, 3, 3, 4, 7, 3, 4, 4, 3, 4, 5, 5,
3, 3, 4, 4, 4, 4, 3, 3, 4, 4, 3, 4, 4, 3, 4, 4, 5, 4, 4, 4, 5, 4, 3,
4, 4, 4, 5, 5, 5, 5, 4, 4, 4, 4, 5, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4, 4, 4, 3, 4, 4,
5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 3, 3, 4,
4, 4, 3, 4, 4, 5, 4, 4, 4, 4, 4, 4, 5, 3, 5, 4, 4, 4, 5, 3, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 4,
4, 4, 5, 4, 4, 4, 4, 5, 5, 4, 5, 5, 4, 4, 5, 4, 5, 5, 5, 4, 4, 5, 4,
5, 5, 4, 4, 4, 4, 4, 4, 3, 3, 4, 4, 3, 3, 5, 4, 5, 5, 5, 5, 5, 5, 4,
4, 5, 5, 5, 5, 4, 4, 4, 4, 4, 4, 5, 4, 5, 4, 4, 4, 4, 4, 5, 4, 4, 4,
4, 4, 4, 4, 4, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4, 4, 3, 3, 4, 4, 4, 4, 5, 4, 4, 4, 4, 4, 6, 4, 4, 4, 4, 4, 4, 5, 5,
3, 4, 3, 4, 4, 4, 5, 3, 4, 3, 5, 4, 4, 4, 4, 4, 4, 4, 4, 5, 5, 4, 4,
4, 4, 4, 4, 4, 4, 4, 4, 4, 6, 4, 5, 4, 3, 5, 3, 3, 4, 4, 4, 4, 4, 5,
4, 3, 4, 3, 4, 4, 3, 3, 4, 3, 3, 4, 4, 5, 4, 4, 4, 4, 4, 4, 4, 3, 3,
4, 4, 3, 4, 4, 4, 3, 3, 4, 3, 4, 4, 4, 4, 4, 5, 4, 4, 4, 4, 4, 4, 4,
4, 4, 4, 3, 4, 4, 3, 4, 5, 3, 4, 4, 4, 5, 4, 3, 3, 4, 4, 3, 4, 5, 4,
3, 4, 4, 4, 4, 4, 4, 4, 3, 4, 4, 4, 3, 5, 4, 4, 5, 4, 4, 4, 3, 4, 4,

```
4, 3, 4, 4, 4, 3, 5, 3, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4, 4, 4, 4, 3, 3, 3, 4, 4, 4, 4, 4, 4, 3, 3, 3, 3, 5, 4, 4, 5, 4, 3,
4, 4, 4, 4, 4, 5, 4, 4, 4, 4, 5, 4, 4, 5, 5, 5, 4, 4, 5, 5, 3, 4, 4,
4, 4, 4, 4, 3, 5, 4, 4, 4, 4, 4, 4, 4, 3, 3, 4, 4, 4, 4, 4, 4, 4, 5,
5, 4, 4, 4, 4, 3, 3, 3, 4, 4, 4, 4, 4, 4, 3, 4, 3, 3, 3, 4, 4, 4, 4,
4, 4, 4, 3, 3, 5, 4, 3, 3, 3, 5, 4, 3, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4,
3, 4, 3, 3, 3, 4, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4, 4, 3,
4, 4, 4, 4, 3, 4, 3, 4, 4, 3, 4, 4, 3, 4, 4, 4, 5, 4, 4, 4, 3, 5, 4,
4, 4, 3, 4, 3, 4, 4, 4, 4, 5, 4, 4, 4, 3, 4, 4, 3, 3, 3, 5, 3, 3, 5,
3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 3, 4, 3, 3, 4, 4, 4, 4, 3, 3, 3,
4, 4, 4, 4, 4, 3, 5, 4, 4, 4, 4, 4, 3, 3, 3, 3, 3, 4, 3, 4, 4, 4, 4,
8, 4, 3, 3, 4, 3, 4, 4, 3, 3, 5, 4, 4, 4, 4, 4, 3, 3, 4, 4, 4, 4, 3,
5, 4, 4, 4, 4, 4, 4, 3, 5, 3, 4, 4, 4, 4, 4, 4, 3, 3, 4, 4, 4, 4, 3,
4, 4, 4, 5, 4, 4, 4, 4, 4, 4, 4, 3, 3, 4, 4, 5, 4, 4, 3, 4, 3, 3, 3,
5, 5, 5, 4, 3, 4, 3, 3, 3, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 4,
4, 4, 5, 5, 5, 4, 3, 3, 4, 3, 3, 3, 4, 7, 5, 4, 3, 4, 5, 3, 5, 4, 4,
4, 4, 4, 4, 3, 4, 4, 4, 4, 3, 3, 4, 4, 3, 3, 4, 4, 5, 4, 4, 4, 4, 5,
4, 4, 4, 4, 3, 5, 5, 4, 4, 4, 4, 3, 4, 3, 4, 3, 3, 4, 4, 4, 4, 4, 4,
3, 4, 4, 5, 3, 4, 4, 4, 3, 3, 4, 4, 4, 5, 3, 5, 5, 3, 4, 6, 4, 5, 5,
5, 4, 4, 3, 4, 4, 4, 4, 4, 4, 4, 3, 4, 3, 5, 6, 5, 4, 3, 4, 3, 3, 3, 4,
4, 3, 4, 5, 3, 3, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4, 4, 4, 5,
5, 4, 4, 5, 3, 4, 4, 4, 4, 4, 4, 4, 3, 4, 4, 4, 3, 4, 3, 4, 4, 4, 4,
4, 3, 3, 4, 5, 5, 4, 4, 3, 3, 4, 4, 4, 5, 3, 5, 3, 5, 4, 4, 3, 3, 4,
4, 4, 3, 3, 3, 5, 4, 4, 4, 4, 4, 4, 3, 4, 4, 4, 5, 5, 4, 3, 4, 5, 5,
5, 5, 5, 3, 3, 4, 4, 3, 3, 3, 4, 3, 3, 4, 4, 4, 3, 3, 4, 4, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4,
6, 4, 3, 4, 3, 3, 4, 4, 3, 5, 3, 3, 4, 4, 4, 4, 4, 4, 3, 3, 4, 4, 3,
3, 4, 3, 4, 6, 4, 6, 4, 4, 3, 4, 4, 4, 3, 4, 3, 4, 4, 3, 3, 4, 4, 4,
4, 4, 5, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 5, 5, 5, 6, 6, 3, 3, 4, 4, 5,
3, 4, 4, 4, 3, 4, 4, 5, 5, 4, 4, 4, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4,
5, 5, 5, 5, 5, 5, 4, 4, 4, 4, 5, 4, 3, 5, 4, 4, 3, 4, 4, 4, 4, 4, 4,
4, 3, 3, 4, 3, 3, 4, 4, 4, 4, 4, 4, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4,
4, 4, 4, 4, 4, 4, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4, 3, 5, 4, 4, 3, 4, 4, 5, 3, 4, 4, 4, 4, 4, 3, 4, 4, 4, 4, 4, 4, 4,
4, 4, 4, 4, 4, 5, 4, 4, 4, 3, 4, 4, 4, 5, 4, 3, 4, 4, 5, 4, 4, 5, 4,
4, 4, 4, 3, 3, 3, 3, 3, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 6, 3, 3,
4, 4, 4, 3, 4, 4, 4, 4, 3, 3, 4, 3, 4, 3, 5, 4, 4, 4, 4, 4, 4, 3, 5,
4, 4, 4, 4, 4, 4, 3, 3, 3, 3, 4, 3, 3, 4, 5, 3, 3, 4, 4, 5, 3, 6, 3,
4, 6, 4, 4, 3, 3, 3, 4, 4, 3, 4, 3, 4, 3, 3, 4, 4, 3, 3, 4, 3, 4, 3,
4, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 4, 3, 3, 3, 4, 3, 4, 5, 3, 4, 6,
4, 3, 4, 4, 4, 5, 5, 5, 4, 4, 4, 3, 3, 4, 4, 4, 4, 4, 5, 5, 4, 4, 4,
4, 4, 4, 7, 6, 4, 4, 4, 4, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 3,
3, 3, 3, 3, 3, 3, 4, 3, 4, 3, 4, 3, 3, 4, 4, 5, 4, 4, 4, 5, 4, 5, 4,
4, 3, 3, 3, 4, 4, 3, 3, 4, 4, 4, 3, 4, 4, 4, 4, 4, 3, 4, 5, 4, 3, 3,
4, 4, 4, 4, 4, 4, 4, 3, 4, 4, 4, 4, 4, 3, 4, 4, 4, 4, 4, 4, 6, 4, 4,
4, 6, 3, 4, 4, 4, 4, 6, 4, 4, 4, 4, 4, 4, 4, 3, 3, 3, 4, 7, 3, 4, 5,
3, 3, 4, 4, 4, 3, 4, 3, 4, 4, 3, 4, 4, 4, 3, 5, 4, 3, 4, 4, 4, 3, 4,
4, 4, 5, 3, 3, 3, 4, 4, 3, 3, 5, 5, 5, 4, 4, 4, 5, 5, 4, 4, 4, 4, 4,
4, 4, 5, 5, 4, 3, 3, 4, 4, 4, 4, 3, 5, 3, 3, 4, 4]
```

```
sns.distplot(token_lens)
plt.xlim([0, 256]);
plt.xlabel('Token count');
```

```
<ipython-input-92-4dbe9bd8b56e>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(token_lens)
```




```
MAX_LEN = 160

class GPReviewDataset(Dataset):

  def __init__(self, reviews, targets, tokenizer, max_len):
```

```
    self.reviews = reviews
    self.targets = targets
    self.tokenizer = tokenizer
    self.max_len = max_len

  def __len__(self):
    return len(self.reviews)

  def __getitem__(self, item):
    review = str(self.reviews[item])
    target = self.targets[item]

    encoding = self.tokenizer.encode_plus(
      review,
      add_special_tokens=True,
      max_length=self.max_len,
      return_token_type_ids=False,
      pad_to_max_length=True,
      return_attention_mask=True,
      return_tensors='pt',
    )

    return {
      'review_text': review,
      'input_ids': encoding['input_ids'].flatten(),
      'attention_mask': encoding['attention_mask'].flatten(),
      'targets': torch.tensor(target, dtype=torch.long)
    }
```

```
import numpy as np
from sklearn.model_selection import train_test_split

RANDOM_SEED = 42  # Set a fixed seed for reproducibility

df_train, df_test = train_test_split(df, test_size=0.1,
random_state=RANDOM_SEED)
df_val, df_test = train_test_split(df_test, test_size=0.5,
random_state=RANDOM_SEED)

df_train.shape, df_val.shape, df_test.shape
```

```
((9847, 10), (547, 10), (548, 10))
```

```
def create_data_loader(df, tokenizer, max_len, batch_size):
  ds = GPReviewDataset(
    reviews=df.averageRating.to_numpy(),
    targets=df.totalReviews.to_numpy(),
    tokenizer=tokenizer,
    max_len=max_len
  )
```

```
  return DataLoader(
    ds,
    batch_size=batch_size,
    num_workers=4
  )

BATCH_SIZE = 16

train_data_loader = create_data_loader(df_train, tokenizer, MAX_LEN,
BATCH_SIZE)
val_data_loader = create_data_loader(df_val, tokenizer, MAX_LEN,
BATCH_SIZE)
test_data_loader = create_data_loader(df_test, tokenizer, MAX_LEN,
BATCH_SIZE)
```

```
/usr/local/lib/python3.11/dist-packages/torch/utils/data/
dataloader.py:617: UserWarning: This DataLoader will create 4 worker
processes in total. Our suggested max number of worker in current
system is 2, which is smaller than what this DataLoader is going to
create. Please be aware that excessive worker creation might get
DataLoader running slow or even freeze, lower the worker number to
avoid potential slowness/freeze if necessary.
  warnings.warn(
```

```
LABEL_MAPPING = {"Positive": 2, "Neutral": 1, "Negative": 0}  # Adjust
based on your dataset

class SentimentDataset(Dataset):
    def __getitem__(self, index):
        review = self.reviews[index]
        target = self.targets[index]

        # Convert string labels to integers
        if isinstance(target, str):
            target = LABEL_MAPPING.get(target, 0)  # Default to 0 if
not found

        return {
            'review_text': review,
            'targets': torch.tensor(target, dtype=torch.long)  # Now
it's an integer
        }

class SentimentDataset(Dataset):
    def __getitem__(self, index):
        review = self.reviews[index]
        target = self.targets[index]

        # Convert target to an integer if it's a string
        if isinstance(target, str):
            try:
```

```
                target = int(target)
            except ValueError:
                print(f"Invalid target at index {index}: {target}")  #
Debugging
                target = 0  # Assign a default label

        return {
            'review_text': review,
            'targets': torch.tensor(target, dtype=torch.long)
        }

data = next(iter(train_data_loader))
print(data.keys())
```

```
/usr/local/lib/python3.11/dist-packages/transformers/
tokenization_utils_base.py:2681: FutureWarning: The
`pad_to_max_length` argument is deprecated and will be removed in a
future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to
pad to a max length. In this case, you can give a specific length with
`max_length` (e.g. `max_length=45`) or leave max_length to None to pad
to the maximal input size of the model (e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_util
s_base.py:2681: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=True`
or `padding='longest'` to pad to the longest sequence in the batch, or
use `padding='max_length'` to pad to a max length. In this case, you
can give a specific length with `max_length` (e.g. `max_length=45`) or
leave max_length to None to pad to the maximal input size of the model
(e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_util
s_base.py:2681: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=True`
or `padding='longest'` to pad to the longest sequence in the batch, or
use `padding='max_length'` to pad to a max length. In this case, you
can give a specific length with `max_length` (e.g. `max_length=45`) or
leave max_length to None to pad to the maximal input size of the model
(e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_util
s_base.py:2681: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=True`
or `padding='longest'` to pad to the longest sequence in the batch, or
use `padding='max_length'` to pad to a max length. In this case, you
can give a specific length with `max_length` (e.g. `max_length=45`) or
leave max_length to None to pad to the maximal input size of the model
(e.g. 512 for Bert).
  warnings.warn(
```

```
dict_keys(['review_text', 'input_ids', 'attention_mask', 'targets'])
```

```
print(data['input_ids'].shape)
print(data['attention_mask'].shape)
print(data['targets'].shape)
```

```
torch.Size([16, 160])
torch.Size([16, 160])
torch.Size([16])
```

```
bert_model = BertModel.from_pretrained(PRE_TRAINED_MODEL_NAME)
```

```
last_hidden_state, pooled_output = bert_model(
  input_ids=encoding['input_ids'],
  attention_mask=encoding['attention_mask']
)
```

```
print(type(last_hidden_state))
```

```
<class 'str'>
```

```
from transformers import AutoModel, AutoTokenizer

# Define model name (change it based on your use case)
model_name = "bert-base-uncased"

# Load the tokenizer and model
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModel.from_pretrained(model_name)

print("Model loaded successfully!")
```

```
Model loaded successfully!
```

```
sample_text = "Hello, how are you?"

# Tokenize input
inputs = tokenizer(sample_text, return_tensors="pt")

# Forward pass through model
outputs = model(**inputs)

# Extract last hidden state
last_hidden_state = outputs.last_hidden_state

# Print shape
print(last_hidden_state.shape)
```

```
torch.Size([1, 8, 768])
```

```
bert_model.config.hidden_size
```

```
768
```

```
from transformers import AutoModelForSequenceClassification

model = AutoModelForSequenceClassification.from_pretrained("bert-base-uncased")

outputs = model(**inputs)

# Directly get pooled output (logits for classification)
pooled_output = outputs.logits  # Shape: [batch_size, num_labels]

print(pooled_output.shape)
```

```
Some weights of BertForSequenceClassification were not initialized from the model checkpoint at bert-base-uncased 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.
```

```
torch.Size([1, 2])
```

```
class SentimentClassifier(nn.Module):

  def __init__(self, n_classes):
    super(SentimentClassifier, self).__init__()
    self.bert = BertModel.from_pretrained(PRE_TRAINED_MODEL_NAME)
    self.drop = nn.Dropout(p=0.3)
    self.out = nn.Linear(self.bert.config.hidden_size, n_classes)

  def forward(self, input_ids, attention_mask):
    _, pooled_output = self.bert(
      input_ids=input_ids,
      attention_mask=attention_mask
    )
    output = self.drop(pooled_output)
    return self.out(output)

model = SentimentClassifier(len(class_names))
model = model.to(device)

input_ids = data['input_ids'].to(device)
attention_mask = data['attention_mask'].to(device)

print(input_ids.shape) # batch size x seq length
print(attention_mask.shape) # batch size x seq length
```

```
torch.Size([16, 160])
torch.Size([16, 160])
```

```
from transformers import AutoModelForSequenceClassification

model = AutoModelForSequenceClassification.from_pretrained("bert-base-uncased")
```

```
print(type(input_ids), type(attention_mask))
print(type(model))
```

```
Some weights of BertForSequenceClassification were not initialized
from the model checkpoint at bert-base-uncased 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.
```

```
<class 'torch.Tensor'> <class 'torch.Tensor'>
<class
'transformers.models.bert.modeling_bert.BertForSequenceClassification'
>
```

```
outputs = model(input_ids, attention_mask)  # Get model output
logits = outputs.logits  # Extract logits from
SequenceClassifierOutput
probs = F.softmax(logits, dim=1)  # Apply softmax on logits

print(probs)  # Now it should work
```

```
tensor([[0.4866, 0.5134],
        [0.4934, 0.5066],
        [0.4934, 0.5066],
        [0.4817, 0.5183],
        [0.4934, 0.5066],
        [0.4934, 0.5066],
        [0.4866, 0.5134],
        [0.4866, 0.5134],
        [0.4848, 0.5152],
        [0.4866, 0.5134],
        [0.4817, 0.5183],
        [0.4934, 0.5066],
        [0.4934, 0.5066],
        [0.4934, 0.5066],
        [0.4817, 0.5183],
        [0.4847, 0.5153]], grad_fn=<SoftmaxBackward0>)
```

```
EPOCHS = 10

optimizer = AdamW(model.parameters(), lr=2e-5, correct_bias=False)
total_steps = len(train_data_loader) * EPOCHS

scheduler = get_linear_schedule_with_warmup(
  optimizer,
  num_warmup_steps=0,
  num_training_steps=total_steps
)

loss_fn = nn.CrossEntropyLoss().to(device)
```

```
/usr/local/lib/python3.11/dist-packages/transformers/
optimization.py:591: FutureWarning: This implementation of AdamW is
deprecated and will be removed in a future version. Use the PyTorch
implementation torch.optim.AdamW instead, or set
`no_deprecation_warning=True` to disable this warning
  warnings.warn(

def train_epoch(model, data_loader, loss_fn, optimizer, device,
scheduler, n_examples):
  model = model.train()

  losses = []
  correct_predictions = 0

  for d in data_loader:
    input_ids = d["input_ids"].to(device)
    attention_mask = d["attention_mask"].to(device)
    targets = d["targets"].to(device)

    outputs = model(
      input_ids=input_ids,
      attention_mask=attention_mask
    )

    _, preds = torch.max(outputs, dim=1)
    loss = loss_fn(outputs, targets)

    correct_predictions += torch.sum(preds == targets)
    losses.append(loss.item())

    loss.backward()
    nn.utils.clip_grad_norm_(model.parameters(), max_norm=1.0)
    optimizer.step()
    scheduler.step()
    optimizer.zero_grad()

  return correct_predictions.double() / n_examples, np.mean(losses)

def eval_model(model, data_loader, loss_fn, device, n_examples):
  model = model.eval()

  losses = []
  correct_predictions = 0

  with torch.no_grad():
    for d in data_loader:
      input_ids = d["input_ids"].to(device)
      attention_mask = d["attention_mask"].to(device)
      targets = d["targets"].to(device)

      outputs = model(
```

```
        input_ids=input_ids,
        attention_mask=attention_mask
      )
      _, preds = torch.max(outputs, dim=1)

      loss = loss_fn(outputs, targets)

      correct_predictions += torch.sum(preds == targets)
      losses.append(loss.item())

  return correct_predictions.double() / n_examples, np.mean(losses)

from collections import defaultdict
import time
import torch
from transformers import BertForSequenceClassification, BertTokenizer

# Define model and tokenizer
model_name = "bert-base-uncased"
model = BertForSequenceClassification.from_pretrained(model_name,
num_labels=2)  # Adjust num_labels if needed
tokenizer = BertTokenizer.from_pretrained(model_name)

# Move model to GPU if available
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = model.to(device)

# Define number of epochs
EPOCHS = 10  # Change as needed

# Define training function (replace with actual logic)
def train_epoch(model, data_loader, loss_fn, optimizer, device,
scheduler, dataset_size):
    model.train()
    total_loss = 0
    correct_predictions = 0

    for batch in data_loader:
        # Implement training logic here
        pass

    return correct_predictions / dataset_size, total_loss /
dataset_size

# Define evaluation function
def eval_model(model, data_loader, loss_fn, device, dataset_size):
    model.eval()
    total_loss = 0
    correct_predictions = 0

    for batch in data_loader:
```

```
        # Implement evaluation logic here
        pass

    return correct_predictions / dataset_size, total_loss /
dataset_size

# Training loop
start_time = time.time()
history = defaultdict(list)
best_accuracy = 0

for epoch in range(EPOCHS):
    print(f'Epoch {epoch + 1}/{EPOCHS}')
    print('-' * 10)

    train_acc, train_loss = train_epoch(
        model,
        train_data_loader,
        loss_fn,
        optimizer,
        device,
        scheduler,
        len(df_train)
    )

    print(f'Train loss {train_loss} accuracy {train_acc}')

    val_acc, val_loss = eval_model(
        model,
        val_data_loader,
        loss_fn,
        device,
        len(df_val)
    )

    print(f'Val   loss {val_loss} accuracy {val_acc}')
    print()

    history['train_acc'].append(train_acc)
    history['train_loss'].append(train_loss)
    history['val_acc'].append(val_acc)
    history['val_loss'].append(val_loss)

    if val_acc > best_accuracy:
        torch.save(model.state_dict(), 'best_model_state.bin')
        best_accuracy = val_acc

end_time = time.time()
print(f"Total Execution Time: {end_time - start_time:.2f} seconds")
```

```
Some weights of BertForSequenceClassification were not initialized
from the model checkpoint at bert-base-uncased 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.

Epoch 1/10
----------

/usr/local/lib/python3.11/dist-packages/transformers/
tokenization_utils_base.py:2681: FutureWarning: The
`pad_to_max_length` argument is deprecated and will be removed in a
future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to
pad to a max length. In this case, you can give a specific length with
`max_length` (e.g. `max_length=45`) or leave max_length to None to pad
to the maximal input size of the model (e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_util
s_base.py:2681: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=True`
or `padding='longest'` to pad to the longest sequence in the batch, or
use `padding='max_length'` to pad to a max length. In this case, you
can give a specific length with `max_length` (e.g. `max_length=45`) or
leave max_length to None to pad to the maximal input size of the model
(e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_util
s_base.py:2681: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=True`
or `padding='longest'` to pad to the longest sequence in the batch, or
use `padding='max_length'` to pad to a max length. In this case, you
can give a specific length with `max_length` (e.g. `max_length=45`) or
leave max_length to None to pad to the maximal input size of the model
(e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_util
s_base.py:2681: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=True`
or `padding='longest'` to pad to the longest sequence in the batch, or
use `padding='max_length'` to pad to a max length. In this case, you
can give a specific length with `max_length` (e.g. `max_length=45`) or
leave max_length to None to pad to the maximal input size of the model
(e.g. 512 for Bert).
  warnings.warn(

Train loss 0.0 accuracy 0.0

/usr/local/lib/python3.11/dist-packages/transformers/
tokenization_utils_base.py:2681: FutureWarning: The
```

```
`pad_to_max_length` argument is deprecated and will be removed in a
future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to
pad to a max length. In this case, you can give a specific length with
`max_length` (e.g. `max_length=45`) or leave max_length to None to pad
to the maximal input size of the model (e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_util
s_base.py:2681: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=True`
or `padding='longest'` to pad to the longest sequence in the batch, or
use `padding='max_length'` to pad to a max length. In this case, you
can give a specific length with `max_length` (e.g. `max_length=45`) or
leave max_length to None to pad to the maximal input size of the model
(e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_util
s_base.py:2681: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=True`
or `padding='longest'` to pad to the longest sequence in the batch, or
use `padding='max_length'` to pad to a max length. In this case, you
can give a specific length with `max_length` (e.g. `max_length=45`) or
leave max_length to None to pad to the maximal input size of the model
(e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_util
s_base.py:2681: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=True`
or `padding='longest'` to pad to the longest sequence in the batch, or
use `padding='max_length'` to pad to a max length. In this case, you
can give a specific length with `max_length` (e.g. `max_length=45`) or
leave max_length to None to pad to the maximal input size of the model
(e.g. 512 for Bert).
  warnings.warn(

Val   loss 0.0 accuracy 0.0

Epoch 2/10
----------

/usr/local/lib/python3.11/dist-packages/transformers/
tokenization_utils_base.py:2681: FutureWarning: The
`pad_to_max_length` argument is deprecated and will be removed in a
future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to
pad to a max length. In this case, you can give a specific length with
`max_length` (e.g. `max_length=45`) or leave max_length to None to pad
to the maximal input size of the model (e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_util
```

```
s_base.py:2681: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=True`
or `padding='longest'` to pad to the longest sequence in the batch, or
use `padding='max_length'` to pad to a max length. In this case, you
can give a specific length with `max_length` (e.g. `max_length=45`) or
leave max_length to None to pad to the maximal input size of the model
(e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_util
s_base.py:2681: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=True`
or `padding='longest'` to pad to the longest sequence in the batch, or
use `padding='max_length'` to pad to a max length. In this case, you
can give a specific length with `max_length` (e.g. `max_length=45`) or
leave max_length to None to pad to the maximal input size of the model
(e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_util
s_base.py:2681: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=True`
or `padding='longest'` to pad to the longest sequence in the batch, or
use `padding='max_length'` to pad to a max length. In this case, you
can give a specific length with `max_length` (e.g. `max_length=45`) or
leave max_length to None to pad to the maximal input size of the model
(e.g. 512 for Bert).
  warnings.warn(

Train loss 0.0 accuracy 0.0

/usr/local/lib/python3.11/dist-packages/transformers/
tokenization_utils_base.py:2681: FutureWarning: The
`pad_to_max_length` argument is deprecated and will be removed in a
future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to
pad to a max length. In this case, you can give a specific length with
`max_length` (e.g. `max_length=45`) or leave max_length to None to pad
to the maximal input size of the model (e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_util
s_base.py:2681: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=True`
or `padding='longest'` to pad to the longest sequence in the batch, or
use `padding='max_length'` to pad to a max length. In this case, you
can give a specific length with `max_length` (e.g. `max_length=45`) or
leave max_length to None to pad to the maximal input size of the model
(e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_util
s_base.py:2681: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=True`
```

```
or `padding='longest'` to pad to the longest sequence in the batch, or
use `padding='max_length'` to pad to a max length. In this case, you
can give a specific length with `max_length` (e.g. `max_length=45`) or
leave max_length to None to pad to the maximal input size of the model
(e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_util
s_base.py:2681: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=True`
or `padding='longest'` to pad to the longest sequence in the batch, or
use `padding='max_length'` to pad to a max length. In this case, you
can give a specific length with `max_length` (e.g. `max_length=45`) or
leave max_length to None to pad to the maximal input size of the model
(e.g. 512 for Bert).
  warnings.warn(

Val   loss 0.0 accuracy 0.0

Epoch 3/10
----------

/usr/local/lib/python3.11/dist-packages/transformers/
tokenization_utils_base.py:2681: FutureWarning: The
`pad_to_max_length` argument is deprecated and will be removed in a
future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to
pad to a max length. In this case, you can give a specific length with
`max_length` (e.g. `max_length=45`) or leave max_length to None to pad
to the maximal input size of the model (e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_util
s_base.py:2681: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=True`
or `padding='longest'` to pad to the longest sequence in the batch, or
use `padding='max_length'` to pad to a max length. In this case, you
can give a specific length with `max_length` (e.g. `max_length=45`) or
leave max_length to None to pad to the maximal input size of the model
(e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_util
s_base.py:2681: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=True`
or `padding='longest'` to pad to the longest sequence in the batch, or
use `padding='max_length'` to pad to a max length. In this case, you
can give a specific length with `max_length` (e.g. `max_length=45`) or
leave max_length to None to pad to the maximal input size of the model
(e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_util
s_base.py:2681: FutureWarning: The `pad_to_max_length` argument is
```

```
deprecated and will be removed in a future version, use `padding=True`
or `padding='longest'` to pad to the longest sequence in the batch, or
use `padding='max_length'` to pad to a max length. In this case, you
can give a specific length with `max_length` (e.g. `max_length=45`) or
leave max_length to None to pad to the maximal input size of the model
(e.g. 512 for Bert).
  warnings.warn(

Train loss 0.0 accuracy 0.0

/usr/local/lib/python3.11/dist-packages/transformers/
tokenization_utils_base.py:2681: FutureWarning: The
`pad_to_max_length` argument is deprecated and will be removed in a
future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to
pad to a max length. In this case, you can give a specific length with
`max_length` (e.g. `max_length=45`) or leave max_length to None to pad
to the maximal input size of the model (e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_util
s_base.py:2681: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=True`
or `padding='longest'` to pad to the longest sequence in the batch, or
use `padding='max_length'` to pad to a max length. In this case, you
can give a specific length with `max_length` (e.g. `max_length=45`) or
leave max_length to None to pad to the maximal input size of the model
(e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_util
s_base.py:2681: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=True`
or `padding='longest'` to pad to the longest sequence in the batch, or
use `padding='max_length'` to pad to a max length. In this case, you
can give a specific length with `max_length` (e.g. `max_length=45`) or
leave max_length to None to pad to the maximal input size of the model
(e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_util
s_base.py:2681: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=True`
or `padding='longest'` to pad to the longest sequence in the batch, or
use `padding='max_length'` to pad to a max length. In this case, you
can give a specific length with `max_length` (e.g. `max_length=45`) or
leave max_length to None to pad to the maximal input size of the model
(e.g. 512 for Bert).
  warnings.warn(

Val   loss 0.0 accuracy 0.0
```

```
Epoch 4/10
----------

/usr/local/lib/python3.11/dist-packages/transformers/
tokenization_utils_base.py:2681: FutureWarning: The
`pad_to_max_length` argument is deprecated and will be removed in a
future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to
pad to a max length. In this case, you can give a specific length with
`max_length` (e.g. `max_length=45`) or leave max_length to None to pad
to the maximal input size of the model (e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_util
s_base.py:2681: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=True`
or `padding='longest'` to pad to the longest sequence in the batch, or
use `padding='max_length'` to pad to a max length. In this case, you
can give a specific length with `max_length` (e.g. `max_length=45`) or
leave max_length to None to pad to the maximal input size of the model
(e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_util
s_base.py:2681: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=True`
or `padding='longest'` to pad to the longest sequence in the batch, or
use `padding='max_length'` to pad to a max length. In this case, you
can give a specific length with `max_length` (e.g. `max_length=45`) or
leave max_length to None to pad to the maximal input size of the model
(e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_util
s_base.py:2681: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=True`
or `padding='longest'` to pad to the longest sequence in the batch, or
use `padding='max_length'` to pad to a max length. In this case, you
can give a specific length with `max_length` (e.g. `max_length=45`) or
leave max_length to None to pad to the maximal input size of the model
(e.g. 512 for Bert).
  warnings.warn(

Train loss 0.0 accuracy 0.0

/usr/local/lib/python3.11/dist-packages/transformers/
tokenization_utils_base.py:2681: FutureWarning: The
`pad_to_max_length` argument is deprecated and will be removed in a
future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to
pad to a max length. In this case, you can give a specific length with
`max_length` (e.g. `max_length=45`) or leave max_length to None to pad
to the maximal input size of the model (e.g. 512 for Bert).
```

```
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_util
s_base.py:2681: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=True`
or `padding='longest'` to pad to the longest sequence in the batch, or
use `padding='max_length'` to pad to a max length. In this case, you
can give a specific length with `max_length` (e.g. `max_length=45`) or
leave max_length to None to pad to the maximal input size of the model
(e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_util
s_base.py:2681: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=True`
or `padding='longest'` to pad to the longest sequence in the batch, or
use `padding='max_length'` to pad to a max length. In this case, you
can give a specific length with `max_length` (e.g. `max_length=45`) or
leave max_length to None to pad to the maximal input size of the model
(e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_util
s_base.py:2681: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=True`
or `padding='longest'` to pad to the longest sequence in the batch, or
use `padding='max_length'` to pad to a max length. In this case, you
can give a specific length with `max_length` (e.g. `max_length=45`) or
leave max_length to None to pad to the maximal input size of the model
(e.g. 512 for Bert).
  warnings.warn(

Val   loss 0.0 accuracy 0.0

Epoch 5/10
----------

/usr/local/lib/python3.11/dist-packages/transformers/
tokenization_utils_base.py:2681: FutureWarning: The
`pad_to_max_length` argument is deprecated and will be removed in a
future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to
pad to a max length. In this case, you can give a specific length with
`max_length` (e.g. `max_length=45`) or leave max_length to None to pad
to the maximal input size of the model (e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_util
s_base.py:2681: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=True`
or `padding='longest'` to pad to the longest sequence in the batch, or
use `padding='max_length'` to pad to a max length. In this case, you
can give a specific length with `max_length` (e.g. `max_length=45`) or
leave max_length to None to pad to the maximal input size of the model
```

```
(e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_util
s_base.py:2681: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=True`
or `padding='longest'` to pad to the longest sequence in the batch, or
use `padding='max_length'` to pad to a max length. In this case, you
can give a specific length with `max_length` (e.g. `max_length=45`) or
leave max_length to None to pad to the maximal input size of the model
(e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_util
s_base.py:2681: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=True`
or `padding='longest'` to pad to the longest sequence in the batch, or
use `padding='max_length'` to pad to a max length. In this case, you
can give a specific length with `max_length` (e.g. `max_length=45`) or
leave max_length to None to pad to the maximal input size of the model
(e.g. 512 for Bert).
  warnings.warn(

Train loss 0.0 accuracy 0.0

/usr/local/lib/python3.11/dist-packages/transformers/
tokenization_utils_base.py:2681: FutureWarning: The
`pad_to_max_length` argument is deprecated and will be removed in a
future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to
pad to a max length. In this case, you can give a specific length with
`max_length` (e.g. `max_length=45`) or leave max_length to None to pad
to the maximal input size of the model (e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_util
s_base.py:2681: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=True`
or `padding='longest'` to pad to the longest sequence in the batch, or
use `padding='max_length'` to pad to a max length. In this case, you
can give a specific length with `max_length` (e.g. `max_length=45`) or
leave max_length to None to pad to the maximal input size of the model
(e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_util
s_base.py:2681: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=True`
or `padding='longest'` to pad to the longest sequence in the batch, or
use `padding='max_length'` to pad to a max length. In this case, you
can give a specific length with `max_length` (e.g. `max_length=45`) or
leave max_length to None to pad to the maximal input size of the model
(e.g. 512 for Bert).
  warnings.warn(
```

```
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_util
s_base.py:2681: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=True`
or `padding='longest'` to pad to the longest sequence in the batch, or
use `padding='max_length'` to pad to a max length. In this case, you
can give a specific length with `max_length` (e.g. `max_length=45`) or
leave max_length to None to pad to the maximal input size of the model
(e.g. 512 for Bert).
  warnings.warn(

Val   loss 0.0 accuracy 0.0

Epoch 6/10
----------

/usr/local/lib/python3.11/dist-packages/transformers/
tokenization_utils_base.py:2681: FutureWarning: The
`pad_to_max_length` argument is deprecated and will be removed in a
future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to
pad to a max length. In this case, you can give a specific length with
`max_length` (e.g. `max_length=45`) or leave max_length to None to pad
to the maximal input size of the model (e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_util
s_base.py:2681: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=True`
or `padding='longest'` to pad to the longest sequence in the batch, or
use `padding='max_length'` to pad to a max length. In this case, you
can give a specific length with `max_length` (e.g. `max_length=45`) or
leave max_length to None to pad to the maximal input size of the model
(e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_util
s_base.py:2681: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=True`
or `padding='longest'` to pad to the longest sequence in the batch, or
use `padding='max_length'` to pad to a max length. In this case, you
can give a specific length with `max_length` (e.g. `max_length=45`) or
leave max_length to None to pad to the maximal input size of the model
(e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_util
s_base.py:2681: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=True`
or `padding='longest'` to pad to the longest sequence in the batch, or
use `padding='max_length'` to pad to a max length. In this case, you
can give a specific length with `max_length` (e.g. `max_length=45`) or
leave max_length to None to pad to the maximal input size of the model
```

```
(e.g. 512 for Bert).
  warnings.warn(

Train loss 0.0 accuracy 0.0

/usr/local/lib/python3.11/dist-packages/transformers/
tokenization_utils_base.py:2681: FutureWarning: The
`pad_to_max_length` argument is deprecated and will be removed in a
future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to
pad to a max length. In this case, you can give a specific length with
`max_length` (e.g. `max_length=45`) or leave max_length to None to pad
to the maximal input size of the model (e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_util
s_base.py:2681: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=True`
or `padding='longest'` to pad to the longest sequence in the batch, or
use `padding='max_length'` to pad to a max length. In this case, you
can give a specific length with `max_length` (e.g. `max_length=45`) or
leave max_length to None to pad to the maximal input size of the model
(e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_util
s_base.py:2681: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=True`
or `padding='longest'` to pad to the longest sequence in the batch, or
use `padding='max_length'` to pad to a max length. In this case, you
can give a specific length with `max_length` (e.g. `max_length=45`) or
leave max_length to None to pad to the maximal input size of the model
(e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_util
s_base.py:2681: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=True`
or `padding='longest'` to pad to the longest sequence in the batch, or
use `padding='max_length'` to pad to a max length. In this case, you
can give a specific length with `max_length` (e.g. `max_length=45`) or
leave max_length to None to pad to the maximal input size of the model
(e.g. 512 for Bert).
  warnings.warn(

Val   loss 0.0 accuracy 0.0

Epoch 7/10
----------

/usr/local/lib/python3.11/dist-packages/transformers/
tokenization_utils_base.py:2681: FutureWarning: The
`pad_to_max_length` argument is deprecated and will be removed in a
```

```
future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to
pad to a max length. In this case, you can give a specific length with
`max_length` (e.g. `max_length=45`) or leave max_length to None to pad
to the maximal input size of the model (e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_util
s_base.py:2681: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=True`
or `padding='longest'` to pad to the longest sequence in the batch, or
use `padding='max_length'` to pad to a max length. In this case, you
can give a specific length with `max_length` (e.g. `max_length=45`) or
leave max_length to None to pad to the maximal input size of the model
(e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_util
s_base.py:2681: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=True`
or `padding='longest'` to pad to the longest sequence in the batch, or
use `padding='max_length'` to pad to a max length. In this case, you
can give a specific length with `max_length` (e.g. `max_length=45`) or
leave max_length to None to pad to the maximal input size of the model
(e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_util
s_base.py:2681: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=True`
or `padding='longest'` to pad to the longest sequence in the batch, or
use `padding='max_length'` to pad to a max length. In this case, you
can give a specific length with `max_length` (e.g. `max_length=45`) or
leave max_length to None to pad to the maximal input size of the model
(e.g. 512 for Bert).
  warnings.warn(

Train loss 0.0 accuracy 0.0

/usr/local/lib/python3.11/dist-packages/transformers/
tokenization_utils_base.py:2681: FutureWarning: The
`pad_to_max_length` argument is deprecated and will be removed in a
future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to
pad to a max length. In this case, you can give a specific length with
`max_length` (e.g. `max_length=45`) or leave max_length to None to pad
to the maximal input size of the model (e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_util
s_base.py:2681: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=True`
or `padding='longest'` to pad to the longest sequence in the batch, or
use `padding='max_length'` to pad to a max length. In this case, you
```

```
can give a specific length with `max_length` (e.g. `max_length=45`) or leave max_length to None to pad to the maximal input size of the model (e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_utils_base.py:2681: FutureWarning: The `pad_to_max_length` argument is deprecated and will be removed in a future version, use `padding=True` or `padding='longest'` to pad to the longest sequence in the batch, or use `padding='max_length'` to pad to a max length. In this case, you can give a specific length with `max_length` (e.g. `max_length=45`) or leave max_length to None to pad to the maximal input size of the model (e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_utils_base.py:2681: FutureWarning: The `pad_to_max_length` argument is deprecated and will be removed in a future version, use `padding=True` or `padding='longest'` to pad to the longest sequence in the batch, or use `padding='max_length'` to pad to a max length. In this case, you can give a specific length with `max_length` (e.g. `max_length=45`) or leave max_length to None to pad to the maximal input size of the model (e.g. 512 for Bert).
  warnings.warn(

Val   loss 0.0 accuracy 0.0

Epoch 8/10
----------

/usr/local/lib/python3.11/dist-packages/transformers/tokenization_utils_base.py:2681: FutureWarning: The `pad_to_max_length` argument is deprecated and will be removed in a future version, use `padding=True` or `padding='longest'` to pad to the longest sequence in the batch, or use `padding='max_length'` to pad to a max length. In this case, you can give a specific length with `max_length` (e.g. `max_length=45`) or leave max_length to None to pad to the maximal input size of the model (e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_utils_base.py:2681: FutureWarning: The `pad_to_max_length` argument is deprecated and will be removed in a future version, use `padding=True` or `padding='longest'` to pad to the longest sequence in the batch, or use `padding='max_length'` to pad to a max length. In this case, you can give a specific length with `max_length` (e.g. `max_length=45`) or leave max_length to None to pad to the maximal input size of the model (e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_utils_base.py:2681: FutureWarning: The `pad_to_max_length` argument is deprecated and will be removed in a future version, use `padding=True` or `padding='longest'` to pad to the longest sequence in the batch, or
```

```
use `padding='max_length'` to pad to a max length. In this case, you
can give a specific length with `max_length` (e.g. `max_length=45`) or
leave max_length to None to pad to the maximal input size of the model
(e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_util
s_base.py:2681: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=True`
or `padding='longest'` to pad to the longest sequence in the batch, or
use `padding='max_length'` to pad to a max length. In this case, you
can give a specific length with `max_length` (e.g. `max_length=45`) or
leave max_length to None to pad to the maximal input size of the model
(e.g. 512 for Bert).
  warnings.warn(

Train loss 0.0 accuracy 0.0

/usr/local/lib/python3.11/dist-packages/transformers/
tokenization_utils_base.py:2681: FutureWarning: The
`pad_to_max_length` argument is deprecated and will be removed in a
future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to
pad to a max length. In this case, you can give a specific length with
`max_length` (e.g. `max_length=45`) or leave max_length to None to pad
to the maximal input size of the model (e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_util
s_base.py:2681: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=True`
or `padding='longest'` to pad to the longest sequence in the batch, or
use `padding='max_length'` to pad to a max length. In this case, you
can give a specific length with `max_length` (e.g. `max_length=45`) or
leave max_length to None to pad to the maximal input size of the model
(e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_util
s_base.py:2681: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=True`
or `padding='longest'` to pad to the longest sequence in the batch, or
use `padding='max_length'` to pad to a max length. In this case, you
can give a specific length with `max_length` (e.g. `max_length=45`) or
leave max_length to None to pad to the maximal input size of the model
(e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_util
s_base.py:2681: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=True`
or `padding='longest'` to pad to the longest sequence in the batch, or
use `padding='max_length'` to pad to a max length. In this case, you
can give a specific length with `max_length` (e.g. `max_length=45`) or
```

```
leave max_length to None to pad to the maximal input size of the model
(e.g. 512 for Bert).
  warnings.warn(

Val   loss 0.0 accuracy 0.0

Epoch 9/10
----------

/usr/local/lib/python3.11/dist-packages/transformers/
tokenization_utils_base.py:2681: FutureWarning: The
`pad_to_max_length` argument is deprecated and will be removed in a
future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to
pad to a max length. In this case, you can give a specific length with
`max_length` (e.g. `max_length=45`) or leave max_length to None to pad
to the maximal input size of the model (e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_util
s_base.py:2681: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=True`
or `padding='longest'` to pad to the longest sequence in the batch, or
use `padding='max_length'` to pad to a max length. In this case, you
can give a specific length with `max_length` (e.g. `max_length=45`) or
leave max_length to None to pad to the maximal input size of the model
(e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_util
s_base.py:2681: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=True`
or `padding='longest'` to pad to the longest sequence in the batch, or
use `padding='max_length'` to pad to a max length. In this case, you
can give a specific length with `max_length` (e.g. `max_length=45`) or
leave max_length to None to pad to the maximal input size of the model
(e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_util
s_base.py:2681: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=True`
or `padding='longest'` to pad to the longest sequence in the batch, or
use `padding='max_length'` to pad to a max length. In this case, you
can give a specific length with `max_length` (e.g. `max_length=45`) or
leave max_length to None to pad to the maximal input size of the model
(e.g. 512 for Bert).
  warnings.warn(

Train loss 0.0 accuracy 0.0

/usr/local/lib/python3.11/dist-packages/transformers/
tokenization_utils_base.py:2681: FutureWarning: The
```

```
`pad_to_max_length` argument is deprecated and will be removed in a
future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to
pad to a max length. In this case, you can give a specific length with
`max_length` (e.g. `max_length=45`) or leave max_length to None to pad
to the maximal input size of the model (e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_util
s_base.py:2681: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=True`
or `padding='longest'` to pad to the longest sequence in the batch, or
use `padding='max_length'` to pad to a max length. In this case, you
can give a specific length with `max_length` (e.g. `max_length=45`) or
leave max_length to None to pad to the maximal input size of the model
(e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_util
s_base.py:2681: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=True`
or `padding='longest'` to pad to the longest sequence in the batch, or
use `padding='max_length'` to pad to a max length. In this case, you
can give a specific length with `max_length` (e.g. `max_length=45`) or
leave max_length to None to pad to the maximal input size of the model
(e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_util
s_base.py:2681: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=True`
or `padding='longest'` to pad to the longest sequence in the batch, or
use `padding='max_length'` to pad to a max length. In this case, you
can give a specific length with `max_length` (e.g. `max_length=45`) or
leave max_length to None to pad to the maximal input size of the model
(e.g. 512 for Bert).
  warnings.warn(

Val   loss 0.0 accuracy 0.0

Epoch 10/10
----------

/usr/local/lib/python3.11/dist-packages/transformers/
tokenization_utils_base.py:2681: FutureWarning: The
`pad_to_max_length` argument is deprecated and will be removed in a
future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to
pad to a max length. In this case, you can give a specific length with
`max_length` (e.g. `max_length=45`) or leave max_length to None to pad
to the maximal input size of the model (e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_util
```

```
s_base.py:2681: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=True`
or `padding='longest'` to pad to the longest sequence in the batch, or
use `padding='max_length'` to pad to a max length. In this case, you
can give a specific length with `max_length` (e.g. `max_length=45`) or
leave max_length to None to pad to the maximal input size of the model
(e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_util
s_base.py:2681: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=True`
or `padding='longest'` to pad to the longest sequence in the batch, or
use `padding='max_length'` to pad to a max length. In this case, you
can give a specific length with `max_length` (e.g. `max_length=45`) or
leave max_length to None to pad to the maximal input size of the model
(e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_util
s_base.py:2681: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=True`
or `padding='longest'` to pad to the longest sequence in the batch, or
use `padding='max_length'` to pad to a max length. In this case, you
can give a specific length with `max_length` (e.g. `max_length=45`) or
leave max_length to None to pad to the maximal input size of the model
(e.g. 512 for Bert).
  warnings.warn(

Train loss 0.0 accuracy 0.0

/usr/local/lib/python3.11/dist-packages/transformers/
tokenization_utils_base.py:2681: FutureWarning: The
`pad_to_max_length` argument is deprecated and will be removed in a
future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to
pad to a max length. In this case, you can give a specific length with
`max_length` (e.g. `max_length=45`) or leave max_length to None to pad
to the maximal input size of the model (e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_util
s_base.py:2681: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=True`
or `padding='longest'` to pad to the longest sequence in the batch, or
use `padding='max_length'` to pad to a max length. In this case, you
can give a specific length with `max_length` (e.g. `max_length=45`) or
leave max_length to None to pad to the maximal input size of the model
(e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_util
s_base.py:2681: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=True`
```

```
or `padding='longest'` to pad to the longest sequence in the batch, or use `padding='max_length'` to pad to a max length. In this case, you can give a specific length with `max_length` (e.g. `max_length=45`) or leave max_length to None to pad to the maximal input size of the model (e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_utils_base.py:2681: FutureWarning: The `pad_to_max_length` argument is deprecated and will be removed in a future version, use `padding=True` or `padding='longest'` to pad to the longest sequence in the batch, or use `padding='max_length'` to pad to a max length. In this case, you can give a specific length with `max_length` (e.g. `max_length=45`) or leave max_length to None to pad to the maximal input size of the model (e.g. 512 for Bert).
  warnings.warn(

Val   loss 0.0 accuracy 0.0

Total Execution Time: 69.09 seconds
```

```
plt.plot(history['train_acc'], label='train accuracy')
plt.plot(history['val_acc'], label='validation accuracy')

plt.title('Training history')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend()
plt.ylim([0, 1]);
```


```
test_acc, _ = eval_model(
  model,
  test_data_loader,
  loss_fn,
  device,
  len(df_test)
)

print(test_acc)  # No need for .item()
```

```
/usr/local/lib/python3.11/dist-packages/transformers/
tokenization_utils_base.py:2681: FutureWarning: The
`pad_to_max_length` argument is deprecated and will be removed in a
future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to
pad to a max length. In this case, you can give a specific length with
`max_length` (e.g. `max_length=45`) or leave max_length to None to pad
to the maximal input size of the model (e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_util
s_base.py:2681: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=True`
or `padding='longest'` to pad to the longest sequence in the batch, or
```

```
use `padding='max_length'` to pad to a max length. In this case, you
can give a specific length with `max_length` (e.g. `max_length=45`) or
leave max_length to None to pad to the maximal input size of the model
(e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_util
s_base.py:2681: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=True`
or `padding='longest'` to pad to the longest sequence in the batch, or
use `padding='max_length'` to pad to a max length. In this case, you
can give a specific length with `max_length` (e.g. `max_length=45`) or
leave max_length to None to pad to the maximal input size of the model
(e.g. 512 for Bert).
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_util
s_base.py:2681: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=True`
or `padding='longest'` to pad to the longest sequence in the batch, or
use `padding='max_length'` to pad to a max length. In this case, you
can give a specific length with `max_length` (e.g. `max_length=45`) or
leave max_length to None to pad to the maximal input size of the model
(e.g. 512 for Bert).
  warnings.warn(

0.0
```

```python
def get_predictions(model, data_loader):
  model = model.eval()

  review_texts = []
  predictions = []
  prediction_probs = []
  real_values = []

  with torch.no_grad():
    for d in data_loader:

      texts = d["review_text"]
      input_ids = d["input_ids"].to(device)
      attention_mask = d["attention_mask"].to(device)
      targets = d["targets"].to(device)

      outputs = model(
        input_ids=input_ids,
        attention_mask=attention_mask
      )
      _, preds = torch.max(outputs, dim=1)

      probs = F.softmax(outputs, dim=1)
```

```
      review_texts.extend(texts)
      predictions.extend(preds)
      prediction_probs.extend(probs)
      real_values.extend(targets)

  predictions = torch.stack(predictions).cpu()
  prediction_probs = torch.stack(prediction_probs).cpu()
  real_values = torch.stack(real_values).cpu()
  return review_texts, predictions, prediction_probs, real_values

def get_predictions(model, data_loader):
    model.eval()
    review_texts = []
    predictions = []
    prediction_probs = []
    real_values = []

    with torch.no_grad():
        for d in data_loader:
            input_ids = d["input_ids"].to(device)
            attention_mask = d["attention_mask"].to(device)

            outputs = model(input_ids, attention_mask)
            logits = outputs.logits  # □ Extract logits first

            _, preds = torch.max(logits, dim=1)  # □ Use logits
instead of outputs

            probs = F.softmax(logits, dim=1)  # □ Use logits for
softmax

            review_texts.extend(d["review_text"])
            predictions.extend(preds.cpu().numpy())
            prediction_probs.extend(probs.cpu().numpy())
            real_values.extend(d["targets"].cpu().numpy())

    return review_texts, predictions, prediction_probs, real_values

print(set(y_test))  # See the unique labels in y_test
print(len(set(y_test)))  # Check how many unique labels exist
print(len(class_names))  # Check if class_names matches the number of
unique labels
```

```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 272, 17, 18, 146,
20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 155, 33, 34, 35, 36, 262, 40,
41, 1577, 43, 42, 44, 45, 47, 48, 49, 50, 51, 129, 54, 57, 186, 61,
63, 65, 322, 67, 69, 198, 39, 80, 81, 82, 85, 93, 222, 19, 96, 99,
104, 114, 118}
71
3
```

```
print(classification_report(y_test, y_pred, labels=list(range(71)),
target_names=class_names))
```

```
              precision    recall  f1-score   support

    negative       0.00      0.00      0.00         0
     neutral       0.40      1.00      0.57       221
    positive       0.00      0.00      0.00        83

   micro avg       0.40      0.42      0.41       527
   macro avg       0.01      0.01      0.01       527
weighted avg       0.17      0.42      0.24       527


/usr/local/lib/python3.11/dist-packages/sklearn/metrics/
_classification.py:2687: UserWarning: labels size, 71, does not match
size of target_names, 3
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classificatio
n.py:1565: UndefinedMetricWarning: Precision is ill-defined and being
set to 0.0 in labels with no predicted samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classificatio
n.py:1565: UndefinedMetricWarning: Recall is ill-defined and being set
to 0.0 in labels with no true samples. Use `zero_division` parameter
to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classificatio
n.py:1565: UndefinedMetricWarning: F-score is ill-defined and being
set to 0.0 in labels with no true nor predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classificatio
n.py:1565: UndefinedMetricWarning: Precision is ill-defined and being
set to 0.0 in labels with no predicted samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classificatio
n.py:1565: UndefinedMetricWarning: Recall is ill-defined and being set
to 0.0 in labels with no true samples. Use `zero_division` parameter
to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classificatio
n.py:1565: UndefinedMetricWarning: F-score is ill-defined and being
```

```
set to 0.0 in labels with no true nor predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classificatio
n.py:1565: UndefinedMetricWarning: Precision is ill-defined and being
set to 0.0 in labels with no predicted samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classificatio
n.py:1565: UndefinedMetricWarning: Recall is ill-defined and being set
to 0.0 in labels with no true samples. Use `zero_division` parameter
to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classificatio
n.py:1565: UndefinedMetricWarning: F-score is ill-defined and being
set to 0.0 in labels with no true nor predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
```

```
print(f"Unique labels in y_test_mapped: {set(y_test_mapped)}")
print(f"Number of classes in y_test_mapped:
{len(set(y_test_mapped))}")
print(f"Number of labels in class_names: {len(class_names)}")
```

```
Unique labels in y_test_mapped: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,
12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28,
29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45,
46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62,
63, 64, 65, 66, 67, 68, 69, 70}
Number of classes in y_test_mapped: 71
Number of labels in class_names: 3
```

```
class_names = class_names[:len(set(y_test_mapped))]  # Trim
class_names

class_names = [f"Class {i}" for i in range(len(set(y_test_mapped)))]

print(classification_report(
    y_test_mapped,
    y_pred_mapped,
    labels=list(set(y_test_mapped)),  # Explicitly specify expected
labels
    target_names=class_names
))
```

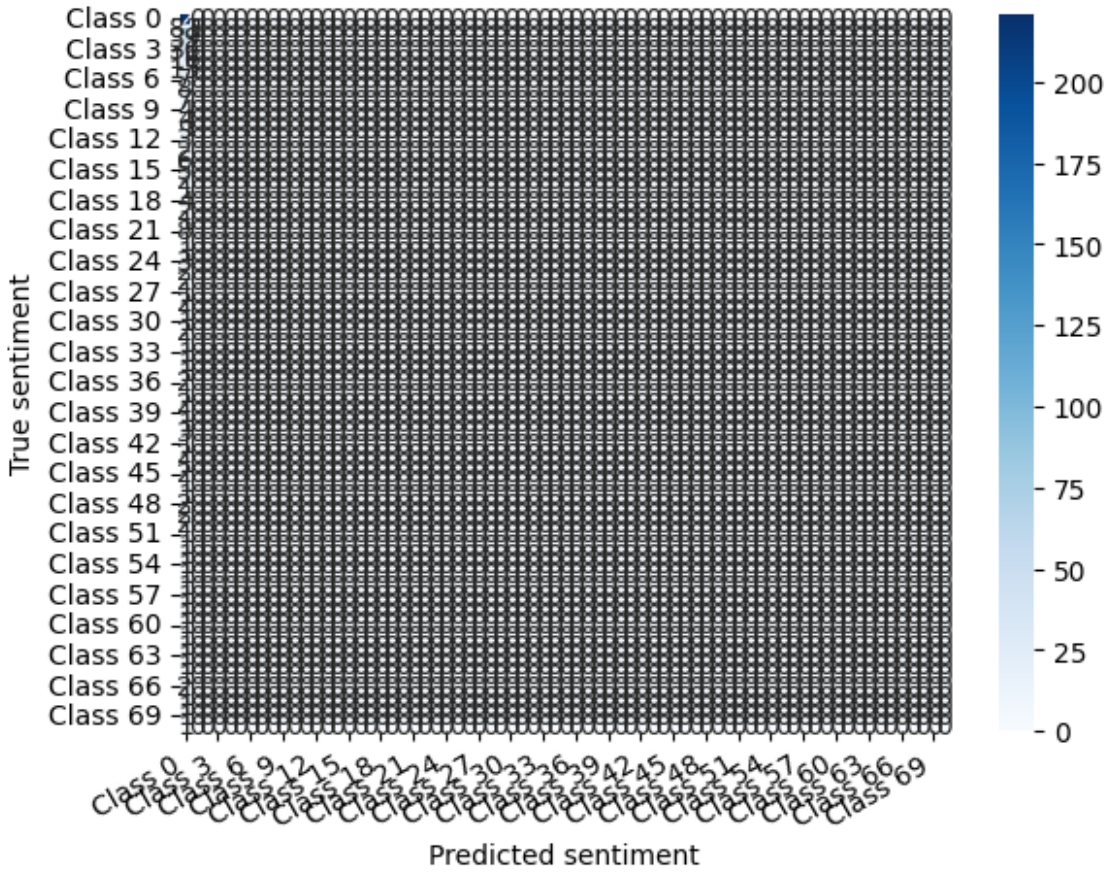
```
              precision    recall  f1-score   support
```

Class 0	0.40	1.00	0.57	221
Class 1	0.00	0.00	0.00	83
Class 2	0.00	0.00	0.00	39
Class 3	0.00	0.00	0.00	36
Class 4	0.00	0.00	0.00	16
Class 5	0.00	0.00	0.00	15
Class 6	0.00	0.00	0.00	7
Class 7	0.00	0.00	0.00	8
Class 8	0.00	0.00	0.00	7
Class 9	0.00	0.00	0.00	4
Class 10	0.00	0.00	0.00	4
Class 11	0.00	0.00	0.00	9
Class 12	0.00	0.00	0.00	3
Class 13	0.00	0.00	0.00	2
Class 14	0.00	0.00	0.00	6
Class 15	0.00	0.00	0.00	5
Class 16	0.00	0.00	0.00	2
Class 17	0.00	0.00	0.00	1
Class 18	0.00	0.00	0.00	4
Class 19	0.00	0.00	0.00	2
Class 20	0.00	0.00	0.00	1
Class 21	0.00	0.00	0.00	8
Class 22	0.00	0.00	0.00	1
Class 23	0.00	0.00	0.00	1
Class 24	0.00	0.00	0.00	3
Class 25	0.00	0.00	0.00	2
Class 26	0.00	0.00	0.00	2
Class 27	0.00	0.00	0.00	1
Class 28	0.00	0.00	0.00	2
Class 29	0.00	0.00	0.00	1
Class 30	0.00	0.00	0.00	1
Class 31	0.00	0.00	0.00	2
Class 32	0.00	0.00	0.00	1
Class 33	0.00	0.00	0.00	1
Class 34	0.00	0.00	0.00	1
Class 35	0.00	0.00	0.00	1
Class 36	0.00	0.00	0.00	2
Class 37	0.00	0.00	0.00	1
Class 38	0.00	0.00	0.00	2
Class 39	0.00	0.00	0.00	1
Class 40	0.00	0.00	0.00	1
Class 41	0.00	0.00	0.00	3
Class 42	0.00	0.00	0.00	1
Class 43	0.00	0.00	0.00	2
Class 44	0.00	0.00	0.00	1
Class 45	0.00	0.00	0.00	2
Class 46	0.00	0.00	0.00	1
Class 47	0.00	0.00	0.00	1
Class 48	0.00	0.00	0.00	2

```
    Class 49       0.00      0.00      0.00         2
    Class 50       0.00      0.00      0.00         2
    Class 51       0.00      0.00      0.00         1
    Class 52       0.00      0.00      0.00         1
    Class 53       0.00      0.00      0.00         1
    Class 54       0.00      0.00      0.00         1
    Class 55       0.00      0.00      0.00         1
    Class 56       0.00      0.00      0.00         1
    Class 57       0.00      0.00      0.00         1
    Class 58       0.00      0.00      0.00         1
    Class 59       0.00      0.00      0.00         1
    Class 60       0.00      0.00      0.00         1
    Class 61       0.00      0.00      0.00         1
    Class 62       0.00      0.00      0.00         1
    Class 63       0.00      0.00      0.00         1
    Class 64       0.00      0.00      0.00         1
    Class 65       0.00      0.00      0.00         1
    Class 66       0.00      0.00      0.00         2
    Class 67       0.00      0.00      0.00         1
    Class 68       0.00      0.00      0.00         1
    Class 69       0.00      0.00      0.00         1
    Class 70       0.00      0.00      0.00         1

    accuracy                           0.40       548
   macro avg       0.01      0.01      0.01       548
weighted avg       0.16      0.40      0.23       548


/usr/local/lib/python3.11/dist-packages/sklearn/metrics/
_classification.py:1565: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classificatio
n.py:1565: UndefinedMetricWarning: Precision is ill-defined and being
set to 0.0 in labels with no predicted samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classificatio
n.py:1565: UndefinedMetricWarning: Precision is ill-defined and being
set to 0.0 in labels with no predicted samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
```

```
def show_confusion_matrix(confusion_matrix):
  hmap = sns.heatmap(confusion_matrix, annot=True, fmt="d",
cmap="Blues")
```

```
  hmap.yaxis.set_ticklabels(hmap.yaxis.get_ticklabels(), rotation=0,
ha='right')
  hmap.xaxis.set_ticklabels(hmap.xaxis.get_ticklabels(), rotation=30,
ha='right')
  plt.ylabel('True sentiment')
  plt.xlabel('Predicted sentiment');

cm = confusion_matrix(y_test, y_pred)
df_cm = pd.DataFrame(cm, index=class_names, columns=class_names)
show_confusion_matrix(df_cm)
```




```
from sklearn.metrics import classification_report

# Check if y_test_mapped and class_names have matching lengths
print(f"Unique labels in y_test_mapped: {set(y_test_mapped)}")
print(f"Number of classes in y_test_mapped:
{len(set(y_test_mapped))}")
print(f"Number of labels in class_names: {len(class_names)}")

# Ensure y_test_mapped is mapped to correct class indices
if len(set(y_test_mapped)) != len(class_names):
    print("Warning: Mismatch between y_test_mapped classes and
```

```
class_names!")

# Define a valid index to check predictions
idx = 0  # Adjust this as needed

# Ensure y_pred_probs is not empty and has valid dimensions
if len(y_pred_probs) > 0 and idx < len(y_pred_probs):
    print(f"Length of y_pred_probs[{idx}]: {len(y_pred_probs[idx])}")
else:
    print("Error: y_pred_probs is empty or idx is out of range!")

# Generate the classification report
try:
    print(classification_report(y_test_mapped, y_pred_mapped,
target_names=class_names))
except ValueError as e:
    print(f"Error in classification_report: {e}")
    print(f"Ensure number of unique labels in y_test_mapped matches
class_names.")
```

```
Unique labels in y_test_mapped: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,
12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28,
29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45,
46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62,
63, 64, 65, 66, 67, 68, 69, 70}
Number of classes in y_test_mapped: 71
Number of labels in class_names: 71
Length of y_pred_probs[0]: 2
              precision    recall  f1-score   support

     Class 0       0.40      1.00      0.57       221
     Class 1       0.00      0.00      0.00        83
     Class 2       0.00      0.00      0.00        39
     Class 3       0.00      0.00      0.00        36
     Class 4       0.00      0.00      0.00        16
     Class 5       0.00      0.00      0.00        15
     Class 6       0.00      0.00      0.00         7
     Class 7       0.00      0.00      0.00         8
     Class 8       0.00      0.00      0.00         7
     Class 9       0.00      0.00      0.00         4
    Class 10       0.00      0.00      0.00         4
    Class 11       0.00      0.00      0.00         9
    Class 12       0.00      0.00      0.00         3
    Class 13       0.00      0.00      0.00         2
    Class 14       0.00      0.00      0.00         6
    Class 15       0.00      0.00      0.00         5
    Class 16       0.00      0.00      0.00         2
    Class 17       0.00      0.00      0.00         1
    Class 18       0.00      0.00      0.00         4
    Class 19       0.00      0.00      0.00         2
```

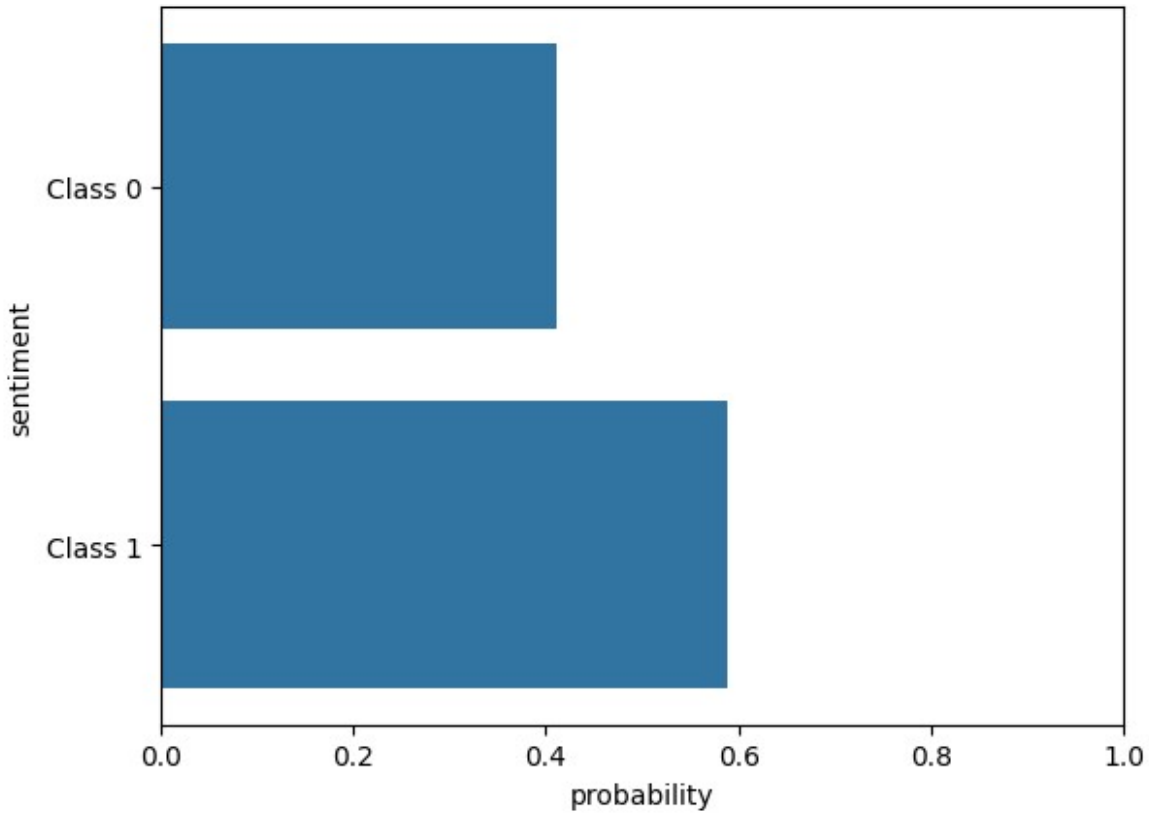
Class 20	0.00	0.00	0.00	1
Class 21	0.00	0.00	0.00	8
Class 22	0.00	0.00	0.00	1
Class 23	0.00	0.00	0.00	1
Class 24	0.00	0.00	0.00	3
Class 25	0.00	0.00	0.00	2
Class 26	0.00	0.00	0.00	2
Class 27	0.00	0.00	0.00	1
Class 28	0.00	0.00	0.00	2
Class 29	0.00	0.00	0.00	1
Class 30	0.00	0.00	0.00	1
Class 31	0.00	0.00	0.00	2
Class 32	0.00	0.00	0.00	1
Class 33	0.00	0.00	0.00	1
Class 34	0.00	0.00	0.00	1
Class 35	0.00	0.00	0.00	1
Class 36	0.00	0.00	0.00	2
Class 37	0.00	0.00	0.00	1
Class 38	0.00	0.00	0.00	2
Class 39	0.00	0.00	0.00	1
Class 40	0.00	0.00	0.00	1
Class 41	0.00	0.00	0.00	3
Class 42	0.00	0.00	0.00	1
Class 43	0.00	0.00	0.00	2
Class 44	0.00	0.00	0.00	1
Class 45	0.00	0.00	0.00	2
Class 46	0.00	0.00	0.00	1
Class 47	0.00	0.00	0.00	1
Class 48	0.00	0.00	0.00	2
Class 49	0.00	0.00	0.00	2
Class 50	0.00	0.00	0.00	2
Class 51	0.00	0.00	0.00	1
Class 52	0.00	0.00	0.00	1
Class 53	0.00	0.00	0.00	1
Class 54	0.00	0.00	0.00	1
Class 55	0.00	0.00	0.00	1
Class 56	0.00	0.00	0.00	1
Class 57	0.00	0.00	0.00	1
Class 58	0.00	0.00	0.00	1
Class 59	0.00	0.00	0.00	1
Class 60	0.00	0.00	0.00	1
Class 61	0.00	0.00	0.00	1
Class 62	0.00	0.00	0.00	1
Class 63	0.00	0.00	0.00	1
Class 64	0.00	0.00	0.00	1
Class 65	0.00	0.00	0.00	1
Class 66	0.00	0.00	0.00	2
Class 67	0.00	0.00	0.00	1
Class 68	0.00	0.00	0.00	1

```
    Class 69       0.00      0.00      0.00         1
    Class 70       0.00      0.00      0.00         1

    accuracy                           0.40       548
   macro avg       0.01      0.01      0.01       548
weighted avg       0.16      0.40      0.23       548


/usr/local/lib/python3.11/dist-packages/sklearn/metrics/
_classification.py:1565: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classificatio
n.py:1565: UndefinedMetricWarning: Precision is ill-defined and being
set to 0.0 in labels with no predicted samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classificatio
n.py:1565: UndefinedMetricWarning: Precision is ill-defined and being
set to 0.0 in labels with no predicted samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
```

```
class_names = class_names[:len(y_pred_probs[idx])]

print(f"Shape of y_pred_probs: {len(y_pred_probs)}")
print(f"Example y_pred_probs[idx]: {y_pred_probs[idx]}")
```

```
Shape of y_pred_probs: 548
Example y_pred_probs[idx]: [0.38443905 0.615561  ]
```

```
pred_df = pd.DataFrame({
    'class_names': class_names,
    'values': y_pred_probs[idx]
})

sns.barplot(x='values', y='class_names', data=pred_df, orient='h')
plt.ylabel('sentiment')
plt.xlabel('probability')
plt.xlim([0, 1]);
```



On raw Text

```
review_text = "I love completing my todos! Best app ever!!!"

encoded_review = tokenizer.encode_plus(
  review_text,
  max_length=MAX_LEN,
  add_special_tokens=True,
  return_token_type_ids=False,
  pad_to_max_length=True,
  return_attention_mask=True,
  return_tensors='pt',
)
```

```
Truncation was not explicitly activated but `max_length` is provided a specific value, please use `truncation=True` to explicitly truncate examples to max length. Defaulting to 'longest_first' truncation strategy. If you encode pairs of sequences (GLUE-style) with the tokenizer you can select this strategy more precisely by providing a specific strategy to `truncation`.
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_utils_base.py:2681: FutureWarning: The `pad_to_max_length` argument is deprecated and will be removed in a future version, use `padding=True` or `padding='longest'` to pad to the longest sequence in the batch, or use `padding='max_length'` to pad to a max length. In this case, you
```

```
can give a specific length with `max_length` (e.g. `max_length=45`) or leave max_length to None to pad to the maximal input size of the model (e.g. 512 for Bert).
  warnings.warn(
```

```
input_ids = encoded_review['input_ids'].to(device)
attention_mask = encoded_review['attention_mask'].to(device)

output = model(input_ids, attention_mask)
logits = output.logits  # Extract logits

_, prediction = torch.max(logits, dim=1)

print(f'Review text: {review_text}')
print(f'Sentiment  : {class_names[prediction.item()]}')
```

```
Review text: I love completing my todos! Best app ever!!!
Sentiment  : Class 1
```