In [1]:
```
import random
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from tqdm import tqdm

sns.set()
plt.rcParams['figure.dpi'] = 100
```

Collection of movie reviews

In [2]:
```
training_reviews_with_labels = [
    ("Watched the whole thing: surprisingly good!", 'positive'),
    ("Beautifully, thoughtfully made", 'positive'),
    ("A good end to a good season", 'positive'),
    ("Better than expected", 'positive'),
    ("Amazing achievement!!", 'positive'),
    ("It's fantastic", 'positive'),
    ("Fully prepared to hate this... completely surprised!", 'positive'),
    ("The right direction", 'positive'),
    ("A visual and storytelling masterpiece.", 'positive'),
    ("Definitely should see if you're a fan of the genre", 'positive'),
    ("Amazing cinematography, most of the storylines are good", 'positive'),
    ("A great series you can actually watch with your family.", 'positive'),
    ("Pretty good start", 'positive'),

    ("Beautiful to watch, a few interesting characters, storyline is good until it isn't",
    ("So much potential, but less realization", 'neutral'),
    ("Missed the mark", 'neutral'),
    ("Has potential, but also flaws", 'neutral'),
    ("Amazing looks but lacking a clear Jacksoneque vision", 'neutral'),
    ("Slow; Slow; Slow", 'neutral'),
    ("It's fine - as in OK - as in mediocre", 'neutral'),
    ("Beautiful but", 'neutral'),
    ("Great CGI but lacks an interesting plot", 'neutral'),

    ("More boring than logic homework", 'negative'),
    ("A major disappointment", 'negative'),
    ("Horrible writing, slow plot, and disrespectful", 'negative'),
    ("An ok fantasy story that has little to do with Tolkien", 'negative'),
    ("Boring, even for generic fantasy", 'negative'),
    ("So many problems", 'negative'),
    ("What is this, an Anti-FanFic or something? Please read the books!", 'negative'),
    ("Just bad all around", 'negative'),
    ("I mourn for what this could have been", 'negative'),
    ("Short version, skip it. You'll know you did right when you don't hear people talking
    ("Could have been so much more", 'negative'),
]
```

In [3]:
```
test_reviews_with_labels = [
    ("The best show I've seen so far this year!", 'positive'),
    ("Really enjoyed it", 'positive'),
    ("Amazing.", 'positive'),
    ("Why all the hate? I enjoyed it.", 'positive'),
    ("Beautiful visuals, entertaining, and I believe this show has a lot of potential!",
    ("A beautiful rendering of Middle Earth's history", 'positive'),
    ("So far, so good... and there's still hope", 'positive'),
    ("I'm a fan", 'positive'),
```

```
    ("It works for me", 'positive'),
    ("Not the best, but enjoyed every episode. Can't wait to see much much more.", 'positi
    ("Beautiful, flawed, and a wonderful Fall treat", 'positive'),

    ("Good show with too many subplots", 'neutral'),
    ("Starts badly, gets better", 'neutral'),
    ("Good and bad things", 'neutral'),
    ("Big and beautiful but can use a little help with its energy.", 'neutral'),
    ("Pretty but ultimately hollow and lacking in engagement", 'neutral'),
    ("Beautiful visuals and story overshadowed by unnecessary gore and violence", 'neutral

    ("Not what you're probably expecting", 'negative'),
    ("Poor writing; Uninteresting characters, nonsensical actions.", 'negative'),
    ("Budget was spent on snacks between shots", 'negative'),
    ("If you ignore the source material, it's still boring and weird", 'negative'),
    ("Just a bad show", 'negative'),
    ("It's awful", 'negative'),
    ("I was hopeful...", 'negative'),
    ("Painfully mediocre with a few good spots", 'negative'),
    ("Beautiful to look at... but that's about it.", 'negative'),
    ("Underwhelming and disappointing", 'negative'),
    ("Tolkien is rolling in his grave. No mystery. No inspiration. Wardrobe & acting is pr
]
```

In [4]:

```
labels = ['positive', 'neutral', 'negative']

training_reviews = [review for review, label in training_reviews_with_labels]
training_labels = np.array([label for review, label in training_reviews_with_labels])

test_reviews = [review for review, label in test_reviews_with_labels]
test_labels = np.array([label for review, label in test_reviews_with_labels])

print('Labels:')
print(labels)
print()

print(f'There are {len(training_reviews)} training reviews')
print('Training labels:')
print(training_labels)
print()

print(f'There are {len(test_reviews)} test reviews')
print('Test labels:')
print(test_labels)
```

```
Labels:
['positive', 'neutral', 'negative']

There are 33 training reviews
Training labels:
['positive' 'positive' 'positive' 'positive' 'positive' 'positive'
 'positive' 'positive' 'positive' 'positive' 'positive' 'positive'
 'positive' 'neutral' 'neutral' 'neutral' 'neutral' 'neutral' 'neutral'
 'neutral' 'neutral' 'neutral' 'negative' 'negative' 'negative' 'negative'
 'negative' 'negative' 'negative' 'negative' 'negative' 'negative'
 'negative']

There are 28 test reviews
Test labels:
['positive' 'positive' 'positive' 'positive' 'positive' 'positive'
 'positive' 'positive' 'positive' 'positive' 'positive' 'neutral'
 'neutral' 'neutral' 'neutral' 'neutral' 'neutral' 'negative' 'negative'
 'negative' 'negative' 'negative' 'negative' 'negative' 'negative'
 'negative' 'negative' 'negative']
```

Classification with a fine-tuned BERT model

In [5]:
```
import transformers
transformers.utils.logging.set_verbosity_error()
```

In [6]:
```
# create training dataset
from datasets import Dataset

training_dataset = Dataset.from_dict({
    'text': training_reviews,
    'label': [labels.index(label) for label in training_labels],
})

print(training_dataset)
```

```
Dataset({
    features: ['text', 'label'],
    num_rows: 33
})
```

In [7]:
```
from transformers import AutoModelForSequenceClassification, TrainingArguments, Trainer, A

model_name = 'bert-base-uncased'
# model_name = 'roberta-base'
tokenizer = AutoTokenizer.from_pretrained(model_name)
```

```
2022-12-15 02:17:46.371606: I tensorflow/core/platform/cpu_feature_guard.cc:193] This Tens
orFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the fol
lowing CPU instructions in performance-critical operations:  AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flag
s.
```

In [8]:
```
for review in training_reviews[:5]:
    print()
    print(review)
    print(tokenizer.tokenize(review))
```

```
Watched the whole thing: surprisingly good!
['watched', 'the', 'whole', 'thing', ':', 'surprisingly', 'good', '!']

Beautifully, thoughtfully made
['beautifully', ',', 'thoughtfully', 'made']

A good end to a good season
['a', 'good', 'end', 'to', 'a', 'good', 'season']

Better than expected
['better', 'than', 'expected']

Amazing achievement!!
['amazing', 'achievement', '!', '!']
```

In [9]:
```
tokenizer.tokenize('tagliatelle')
```

Out[9]:
```
['tag', '##lia', '##tell', '##e']
```

In [10]:
```
def preprocess_function(examples):
    return tokenizer(examples["text"], truncation=True)
```

```
tokenized_training_dataset = training_dataset.map(preprocess_function, batched=True, remov

print(tokenized_training_dataset)
```

```
huggingface/tokenizers: The current process just got forked, after parallelism has already
been used. Disabling parallelism to avoid deadlocks...
To disable this warning, you can either:
        - Avoid using `tokenizers` before the fork if possible
  0%|          | 0/1 [00:00<?, ?ba/s]
        - Explicitly set the environment variable TOKENIZERS_PARALLELISM=(true | false)
Dataset({
    features: ['label', 'input_ids', 'token_type_ids', 'attention_mask'],
    num_rows: 33
})
```

In [11]:

```
%load_ext tensorboard
%tensorboard --logdir results
```

```
huggingface/tokenizers: The current process just got forked, after parallelism has already
been used. Disabling parallelism to avoid deadlocks...
To disable this warning, you can either:
        - Avoid using `tokenizers` before the fork if possible
        - Explicitly set the environment variable TOKENIZERS_PARALLELISM=(true | false)
```

In [12]:

```
transformers.set_seed(42)

model = AutoModelForSequenceClassification.from_pretrained(model_name, num_labels=3, outpu

print('Number of parameters:', sum(p.numel() for p in model.parameters()))
```

```
Number of parameters: 109484547
```

In [13]:
```
print(model)
```

```
BertForSequenceClassification(
  (bert): BertModel(
    (embeddings): BertEmbeddings(
      (word_embeddings): Embedding(30522, 768, padding_idx=0)
      (position_embeddings): Embedding(512, 768)
      (token_type_embeddings): Embedding(2, 768)
      (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (encoder): BertEncoder(
      (layer): ModuleList(
        (0): BertLayer(
          (attention): BertAttention(
            (self): BertSelfAttention(
              (query): Linear(in_features=768, out_features=768, bias=True)
              (key): Linear(in_features=768, out_features=768, bias=True)
              (value): Linear(in_features=768, out_features=768, bias=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
            (output): BertSelfOutput(
              (dense): Linear(in_features=768, out_features=768, bias=True)
              (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
          )
          (intermediate): BertIntermediate(
            (dense): Linear(in_features=768, out_features=3072, bias=True)
          )
          (output): BertOutput(
            (dense): Linear(in_features=3072, out_features=768, bias=True)
            (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
        (1): BertLayer(
          (attention): BertAttention(
            (self): BertSelfAttention(
              (query): Linear(in_features=768, out_features=768, bias=True)
              (key): Linear(in_features=768, out_features=768, bias=True)
              (value): Linear(in_features=768, out_features=768, bias=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
            (output): BertSelfOutput(
              (dense): Linear(in_features=768, out_features=768, bias=True)
              (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
          )
          (intermediate): BertIntermediate(
            (dense): Linear(in_features=768, out_features=3072, bias=True)
          )
          (output): BertOutput(
            (dense): Linear(in_features=3072, out_features=768, bias=True)
            (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
        (2): BertLayer(
          (attention): BertAttention(
            (self): BertSelfAttention(
              (query): Linear(in_features=768, out_features=768, bias=True)
              (key): Linear(in_features=768, out_features=768, bias=True)
              (value): Linear(in_features=768, out_features=768, bias=True)
```

```
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (output): BertSelfOutput(
      (dense): Linear(in_features=768, out_features=768, bias=True)
      (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
  )
  (intermediate): BertIntermediate(
    (dense): Linear(in_features=768, out_features=3072, bias=True)
  )
  (output): BertOutput(
    (dense): Linear(in_features=3072, out_features=768, bias=True)
    (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
    (dropout): Dropout(p=0.1, inplace=False)
  )
)
(3): BertLayer(
  (attention): BertAttention(
    (self): BertSelfAttention(
      (query): Linear(in_features=768, out_features=768, bias=True)
      (key): Linear(in_features=768, out_features=768, bias=True)
      (value): Linear(in_features=768, out_features=768, bias=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (output): BertSelfOutput(
      (dense): Linear(in_features=768, out_features=768, bias=True)
      (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
  )
  (intermediate): BertIntermediate(
    (dense): Linear(in_features=768, out_features=3072, bias=True)
  )
  (output): BertOutput(
    (dense): Linear(in_features=3072, out_features=768, bias=True)
    (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
    (dropout): Dropout(p=0.1, inplace=False)
  )
)
(4): BertLayer(
  (attention): BertAttention(
    (self): BertSelfAttention(
      (query): Linear(in_features=768, out_features=768, bias=True)
      (key): Linear(in_features=768, out_features=768, bias=True)
      (value): Linear(in_features=768, out_features=768, bias=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (output): BertSelfOutput(
      (dense): Linear(in_features=768, out_features=768, bias=True)
      (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
  )
  (intermediate): BertIntermediate(
    (dense): Linear(in_features=768, out_features=3072, bias=True)
  )
  (output): BertOutput(
    (dense): Linear(in_features=3072, out_features=768, bias=True)
    (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
    (dropout): Dropout(p=0.1, inplace=False)
  )
)
(5): BertLayer(
  (attention): BertAttention(
    (self): BertSelfAttention(
```

```
        (query): Linear(in_features=768, out_features=768, bias=True)
        (key): Linear(in_features=768, out_features=768, bias=True)
        (value): Linear(in_features=768, out_features=768, bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (output): BertSelfOutput(
        (dense): Linear(in_features=768, out_features=768, bias=True)
        (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
    )
    (intermediate): BertIntermediate(
      (dense): Linear(in_features=768, out_features=3072, bias=True)
    )
    (output): BertOutput(
      (dense): Linear(in_features=3072, out_features=768, bias=True)
      (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
  )
  (6): BertLayer(
    (attention): BertAttention(
      (self): BertSelfAttention(
        (query): Linear(in_features=768, out_features=768, bias=True)
        (key): Linear(in_features=768, out_features=768, bias=True)
        (value): Linear(in_features=768, out_features=768, bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (output): BertSelfOutput(
        (dense): Linear(in_features=768, out_features=768, bias=True)
        (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
    )
    (intermediate): BertIntermediate(
      (dense): Linear(in_features=768, out_features=3072, bias=True)
    )
    (output): BertOutput(
      (dense): Linear(in_features=3072, out_features=768, bias=True)
      (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
  )
  (7): BertLayer(
    (attention): BertAttention(
      (self): BertSelfAttention(
        (query): Linear(in_features=768, out_features=768, bias=True)
        (key): Linear(in_features=768, out_features=768, bias=True)
        (value): Linear(in_features=768, out_features=768, bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (output): BertSelfOutput(
        (dense): Linear(in_features=768, out_features=768, bias=True)
        (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
    )
    (intermediate): BertIntermediate(
      (dense): Linear(in_features=768, out_features=3072, bias=True)
    )
    (output): BertOutput(
      (dense): Linear(in_features=3072, out_features=768, bias=True)
      (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
  )
```

```
      (8): BertLayer(
        (attention): BertAttention(
          (self): BertSelfAttention(
            (query): Linear(in_features=768, out_features=768, bias=True)
            (key): Linear(in_features=768, out_features=768, bias=True)
            (value): Linear(in_features=768, out_features=768, bias=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
          (output): BertSelfOutput(
            (dense): Linear(in_features=768, out_features=768, bias=True)
            (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
        (intermediate): BertIntermediate(
          (dense): Linear(in_features=768, out_features=3072, bias=True)
        )
        (output): BertOutput(
          (dense): Linear(in_features=3072, out_features=768, bias=True)
          (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
          (dropout): Dropout(p=0.1, inplace=False)
        )
      )
      (9): BertLayer(
        (attention): BertAttention(
          (self): BertSelfAttention(
            (query): Linear(in_features=768, out_features=768, bias=True)
            (key): Linear(in_features=768, out_features=768, bias=True)
            (value): Linear(in_features=768, out_features=768, bias=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
          (output): BertSelfOutput(
            (dense): Linear(in_features=768, out_features=768, bias=True)
            (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
        (intermediate): BertIntermediate(
          (dense): Linear(in_features=768, out_features=3072, bias=True)
        )
        (output): BertOutput(
          (dense): Linear(in_features=3072, out_features=768, bias=True)
          (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
          (dropout): Dropout(p=0.1, inplace=False)
        )
      )
      (10): BertLayer(
        (attention): BertAttention(
          (self): BertSelfAttention(
            (query): Linear(in_features=768, out_features=768, bias=True)
            (key): Linear(in_features=768, out_features=768, bias=True)
            (value): Linear(in_features=768, out_features=768, bias=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
          (output): BertSelfOutput(
            (dense): Linear(in_features=768, out_features=768, bias=True)
            (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
        (intermediate): BertIntermediate(
          (dense): Linear(in_features=768, out_features=3072, bias=True)
        )
        (output): BertOutput(
          (dense): Linear(in_features=3072, out_features=768, bias=True)
          (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
```

```
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
        (11): BertLayer(
          (attention): BertAttention(
            (self): BertSelfAttention(
              (query): Linear(in_features=768, out_features=768, bias=True)
              (key): Linear(in_features=768, out_features=768, bias=True)
              (value): Linear(in_features=768, out_features=768, bias=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
            (output): BertSelfOutput(
              (dense): Linear(in_features=768, out_features=768, bias=True)
              (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
          )
          (intermediate): BertIntermediate(
            (dense): Linear(in_features=768, out_features=3072, bias=True)
          )
          (output): BertOutput(
            (dense): Linear(in_features=3072, out_features=768, bias=True)
            (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
      )
    )
    (pooler): BertPooler(
      (dense): Linear(in_features=768, out_features=768, bias=True)
      (activation): Tanh()
    )
  )
  (dropout): Dropout(p=0.1, inplace=False)
  (classifier): Linear(in_features=768, out_features=3, bias=True)
)
```

In [14]:

```
training_args = TrainingArguments(
    output_dir="./results",
    logging_dir="./results",
    learning_rate=5e-5,
    per_device_train_batch_size=8,
    num_train_epochs=20,
    weight_decay=0.01,
    logging_steps=5,
)

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_training_dataset,
    tokenizer=tokenizer,
)

trainer.train()
```

```
/usr/local/lib/python3.9/site-packages/transformers/optimization.py:306: FutureWarning: Th
is implementation of AdamW is deprecated and will be removed in a future version. Use theP
yTorch implementation torch.optim.AdamW instead, or set `no_deprecation_warning=True` to d
isable this warning
  warnings.warn(
***** Running training *****
  Num examples = 33
  Num Epochs = 20
  Instantaneous batch size per device = 8
```

```
  Total train batch size (w. parallel, distributed & accumulation) = 8
  Gradient Accumulation steps = 1
  Total optimization steps = 100
{'loss': 1.1186, 'learning_rate': 4.75e-05, 'epoch': 1.0}
{'loss': 1.0344, 'learning_rate': 4.5e-05, 'epoch': 2.0}
{'loss': 0.8069, 'learning_rate': 4.25e-05, 'epoch': 3.0}
{'loss': 0.6417, 'learning_rate': 4e-05, 'epoch': 4.0}
{'loss': 0.4622, 'learning_rate': 3.7500000000000003e-05, 'epoch': 5.0}
{'loss': 0.2898, 'learning_rate': 3.5e-05, 'epoch': 6.0}
{'loss': 0.1892, 'learning_rate': 3.2500000000000004e-05, 'epoch': 7.0}
{'loss': 0.104, 'learning_rate': 3e-05, 'epoch': 8.0}
{'loss': 0.0743, 'learning_rate': 2.7500000000000004e-05, 'epoch': 9.0}
{'loss': 0.0336, 'learning_rate': 2.5e-05, 'epoch': 10.0}
{'loss': 0.0216, 'learning_rate': 2.25e-05, 'epoch': 11.0}
{'loss': 0.0153, 'learning_rate': 2e-05, 'epoch': 12.0}
{'loss': 0.0123, 'learning_rate': 1.75e-05, 'epoch': 13.0}
{'loss': 0.0099, 'learning_rate': 1.5e-05, 'epoch': 14.0}
{'loss': 0.0085, 'learning_rate': 1.25e-05, 'epoch': 15.0}
{'loss': 0.0075, 'learning_rate': 1e-05, 'epoch': 16.0}
{'loss': 0.0071, 'learning_rate': 7.5e-06, 'epoch': 17.0}
{'loss': 0.0062, 'learning_rate': 5e-06, 'epoch': 18.0}
{'loss': 0.0058, 'learning_rate': 2.5e-06, 'epoch': 19.0}

Training completed. Do not forget to share your model on huggingface.co/models =)


{'loss': 0.0056, 'learning_rate': 0.0, 'epoch': 20.0}
{'train_runtime': 154.1382, 'train_samples_per_second': 4.282, 'train_steps_per_second':
0.649, 'train_loss': 0.24271499956026674, 'epoch': 20.0}
```

Out[14]:

```
TrainOutput(global_step=100, training_loss=0.24271499956026674, metrics={'train_runtime':
154.1382, 'train_samples_per_second': 4.282, 'train_steps_per_second': 0.649, 'train_los
s': 0.24271499956026674, 'epoch': 20.0})
```

In [15]:

```python
from scipy.special import softmax
from sklearn.metrics import confusion_matrix, accuracy_score, f1_score
from IPython.display import display

model.eval()

predictions = []
table = []

for review, label in zip(test_reviews, test_labels):
    inputs = tokenizer.encode(review, return_tensors='pt').to(model.device)
    prob = softmax(model(inputs).logits.detach().squeeze())
    prediction = labels[prob.argmax()]

    row = {
        'Review': review,
        'Label': label,
        'Prediction': prediction,
    }
    for label, p in zip(labels, prob):
        row[f'{label} %'] = f'{p * 100:.1f}'

    predictions.append(prediction)
    table.append(row)

print(f'Accuracy: {accuracy_score(test_labels, predictions) * 100:.1f}%')
print(f'F1 score: {f1_score(test_labels, predictions, labels=labels, average=None)}')
```

```
Accuracy: 75.0%
F1 score: [0.88       0.54545455 0.7       ]
```

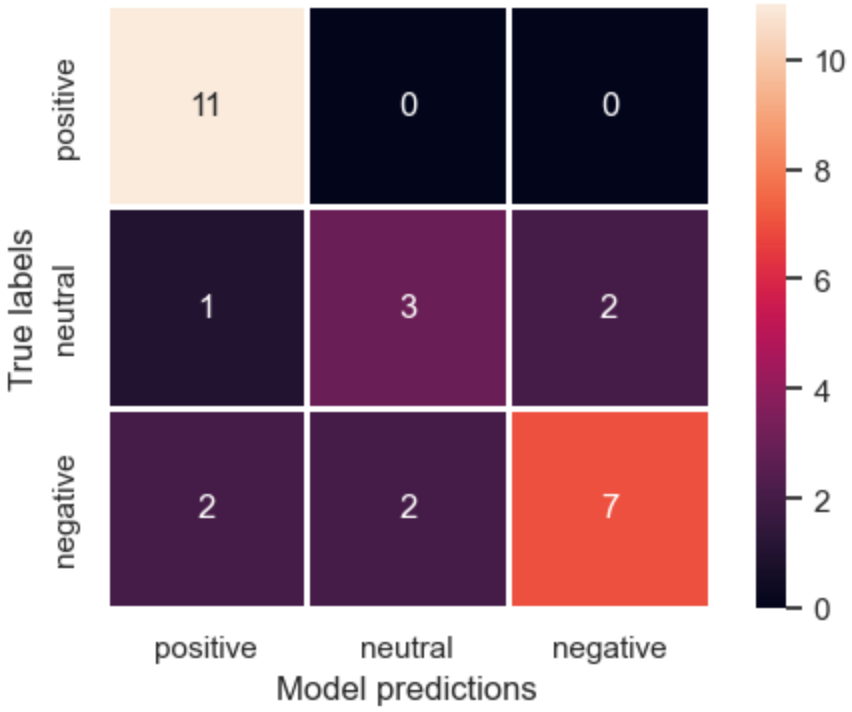
In [16]:
```
pd.set_option('display.max_rows', None)
pd.set_option('display.max_colwidth', None)
display(pd.DataFrame(table))
```

	Review	Label	Prediction	positive %	neutral %	negative %
0	The best show I've seen so far this year!	positive	positive	99.4	0.2	0.4
1	Really enjoyed it	positive	positive	99.5	0.2	0.3
2	Amazing.	positive	positive	99.5	0.3	0.2
3	Why all the hate? I enjoyed it.	positive	positive	84.0	0.5	15.5
4	Beautiful visuals, entertaining, and I believe this show has a lot of potential!	positive	positive	99.5	0.3	0.3
5	A beautiful rendering of Middle Earth's history	positive	positive	99.3	0.2	0.5
6	So far, so good... and there's still hope	positive	positive	99.4	0.3	0.3
7	I'm a fan	positive	positive	99.3	0.3	0.4
8	It works for me	positive	positive	98.2	0.5	1.3
9	Not the best, but enjoyed every episode. Can't wait to see much much more.	positive	positive	96.5	2.9	0.6
10	Beautiful, flawed, and a wonderful Fall treat	positive	positive	94.4	2.1	3.5
11	Good show with too many subplots	neutral	negative	25.7	3.3	71.0
12	Starts badly, gets better	neutral	neutral	0.4	99.3	0.4
13	Good and bad things	neutral	negative	1.4	0.4	98.3
14	Big and beautiful but can use a little help with its energy.	neutral	neutral	0.3	99.6	0.1
15	Pretty but ultimately hollow and lacking in engagement	neutral	neutral	0.2	99.6	0.2
16	Beautiful visuals and story overshadowed by unnecessary gore and violence	neutral	positive	80.8	18.0	1.2
17	Not what you're probably expecting	negative	negative	7.8	0.8	91.4
18	Poor writing; Uninteresting characters, nonsensical actions.	negative	negative	0.7	4.4	94.9
19	Budget was spent on snacks between shots	negative	neutral	13.1	84.7	2.2
20	If you ignore the source material, it's still boring and weird	negative	negative	0.5	0.3	99.2
21	Just a bad show	negative	negative	0.6	0.3	99.1
22	It's awful	negative	negative	29.2	0.7	70.1
23	I was hopeful...	negative	positive	99.3	0.5	0.3
24	Painfully mediocre with a few good spots	negative	neutral	0.4	97.7	1.9
25	Beautiful to look at... but that's about it.	negative	positive	68.2	31.2	0.6
26	Underwhelming and disappointing	negative	negative	0.8	0.5	98.7
27	Tolkien is rolling in his grave. No mystery. No inspiration. Wardrobe & acting is pretty bland.	negative	negative	20.9	6.8	72.3

In [17]:
```
mat = confusion_matrix(test_labels, predictions, labels=labels)

sns.heatmap(mat, square=True, annot=True, xticklabels=labels, yticklabels=labels, linewidt
plt.xlabel("Model predictions")
```

```
plt.ylabel("True labels")
plt.show()
```



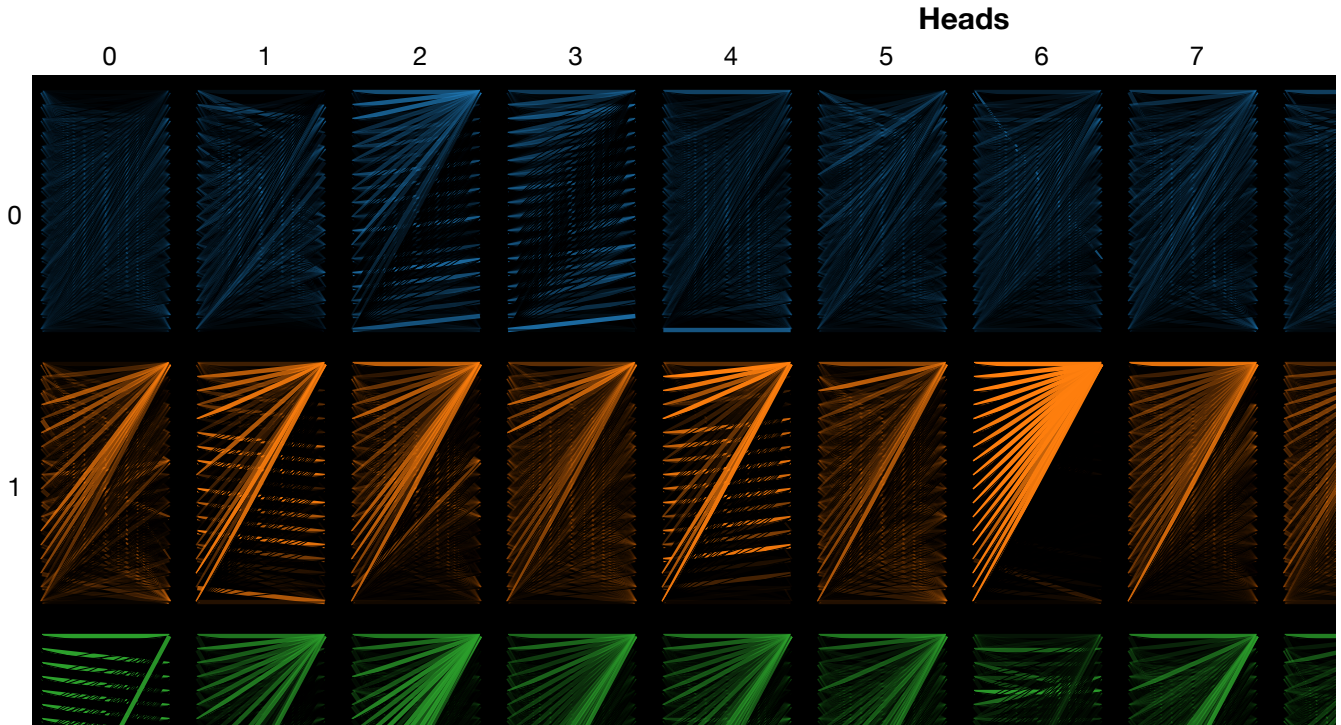
In [18]:

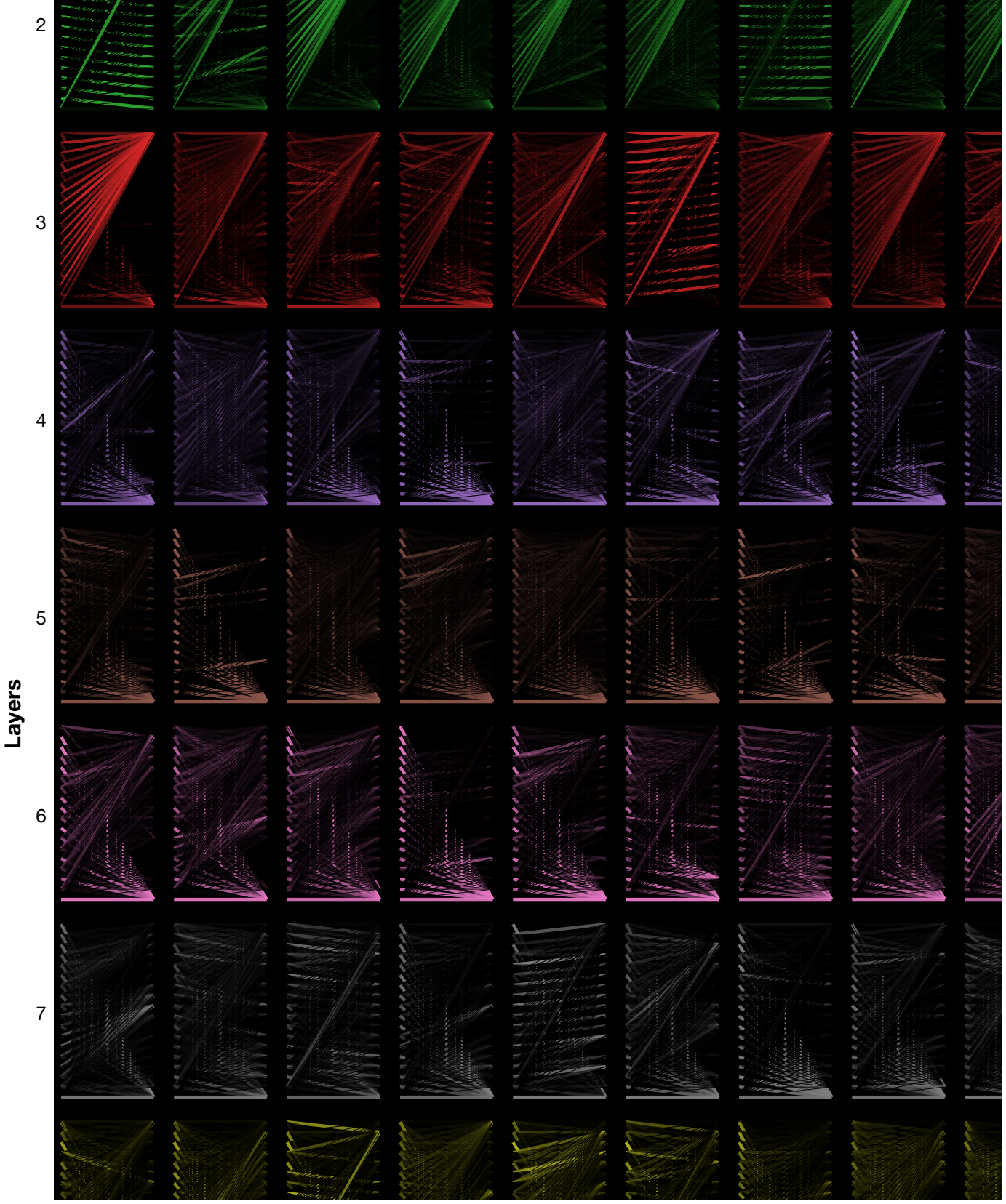
```
input_text = "Beautiful visuals, entertaining, and I believe this show has a lot of potent
# input_text = "It works for me"
# input_text = "Big and beautiful but can use a little help with its energy."
# input_text = "Why all the hate? I enjoyed it."

inputs = tokenizer.encode(input_text, return_tensors='pt').to(model.device)
outputs = model(inputs)
attention = outputs.attentions
tokens = tokenizer.convert_ids_to_tokens(inputs[0])  # convert input ids to token strings
```

In [19]:

```
from bertviz import model_view, head_view
model_view(attention, tokens)
```




Layers
2
3
4
5
6
7