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']
```

Logistic regression

In [5]:
```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(token_pattern=r'(?u)\b\w\w+\b|!')
tokenizer = vectorizer.build_tokenizer()

print(tokenizer('Nice show!'))
```

```
['Nice', 'show', '!']
```

In [6]:
```
# load GloVe embeddings (downloaded from https://nlp.stanford.edu/projects/glove/)
glove_embeddings = {}
with open("glove.6B.50d.txt", 'r') as f:
    for line in f:
        values = line.split()
        word = values[0]
        vector = np.asarray(values[1:], "float32")
        glove_embeddings[word] = vector
```

In [7]:
```
# define embedding
positive_words = set(['great', 'good', 'better', 'amazing', 'fantastic',
                      'beautiful', 'interesting'])
negative_words = set(['boring', 'disappointment', 'horrible', 'slow',
                      'disrespectful', 'weird', 'bad'])

def embedding(text, embedding_type) -> np.array:
    words = [word.lower() for word in tokenizer(text)]

    if embedding_type == 'hard-coded':
        return np.array([
            sum(1 for word in words if word in positive_words),
            sum(1 for word in words if word in negative_words),
            1 if '!' in words else 0,
            np.log(len(words)),
            1,  # for bias term
        ])

    else:
        # average of GloVe embeddings
        return sum(glove_embeddings[word] for word in words
                   if word in glove_embeddings) \
            / sum(1 for word in words if word in glove_embeddings)
```

In [8]:
```
# embedding_type = 'glove'
embedding_type = 'hard-coded'
```

In [9]:
```
embedding_dimension = len(embedding('Nice show!', embedding_type=embedding_type))
print(f'Embedding dimension: {embedding_dimension}')

training_embeddings = np.stack([
    embedding(review, embedding_type=embedding_type)
    for review in training_reviews
])

test_embeddings = np.stack([
    embedding(review, embedding_type=embedding_type)
    for review in test_reviews
])
```

```
print(training_embeddings.shape, test_embeddings.shape)

if embedding_type == 'hard-coded':
    # normalize
    c = training_embeddings.max(axis=0)
    print(f'Normalize training and test embeddings by {c}')

    training_embeddings /= c
    test_embeddings /= c
```

```
Embedding dimension: 5
(33, 5) (28, 5)
Normalize training and test embeddings by [3.         3.         1.         2.89037176 1.
       ]
```

In [10]:

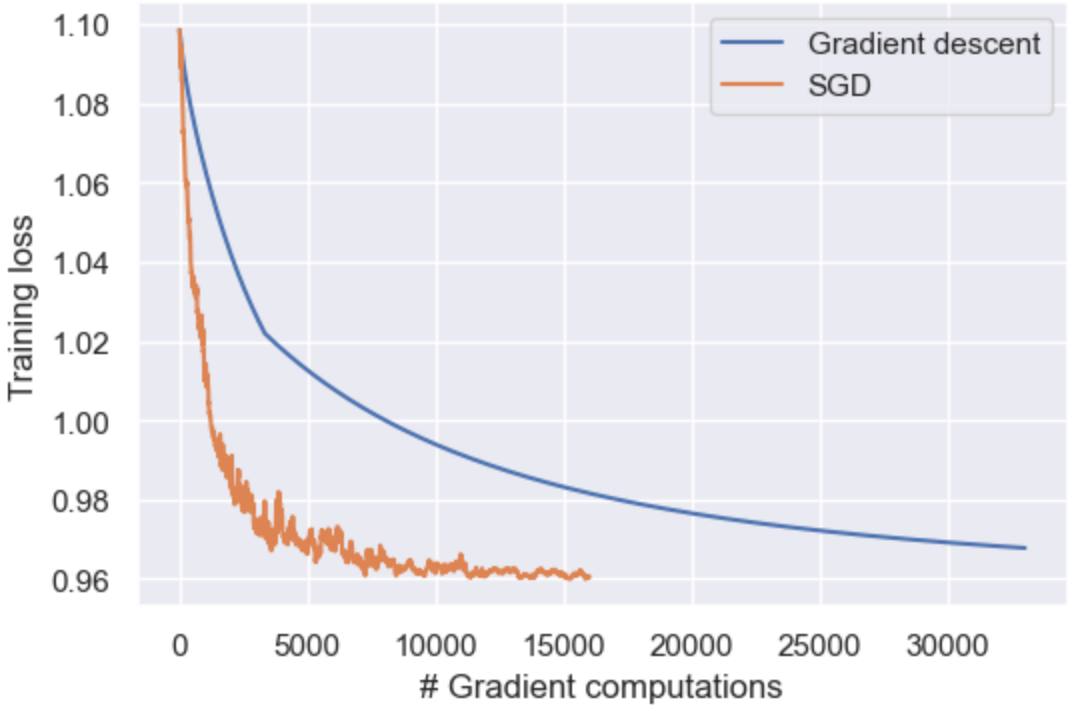
```
from scipy.special import log_softmax

def training_loss(parameters, i=None):
    if i is not None:
        return -log_softmax(np.dot(parameters, training_embeddings[i]))[labels.index(trair
    else:
        return sum(training_loss(parameters, i) for i in range(len(training_reviews))) / l

def gradient(parameters, i=None):
    if i is not None:
        return np.outer(
            np.dot(parameters, training_embeddings[i])
            - np.array([1 if label == training_labels[i] else 0 for label in labels]),
            training_embeddings[i]
        )
    else:
        return sum(gradient(parameters, i) for i in range(len(training_reviews))) / len(tr
```

In [11]:

```
### Gradient descent ###
parameters = np.zeros((3, embedding_dimension))

step = 0.1

losses = [training_loss(parameters)]

for i in range(1000):
    g = gradient(parameters)
    parameters -= step * g
    losses.append(training_loss(parameters))

    if i == 100:
        step /= 2

print(f'Final loss (gradient descent): {losses[-1]}')


### SGD ###
random.seed(0)
parameters = np.zeros((3, embedding_dimension))

step = 0.1
batch_size = 8

sgd_losses = [training_loss(parameters)]

for i in range(2000):
    g = sum(
        gradient(parameters, random.randint(0, len(training_reviews)-1))
```

```
        for _ in range(batch_size)
    ) / batch_size

    parameters -= step * g
    sgd_losses.append(training_loss(parameters))

    if i > 0 and i % 500 == 0:
        step /= 2

print(f'Final loss (SGD): {sgd_losses[-1]}')


plt.plot(range(0, len(losses) * len(training_reviews), len(training_reviews)), losses, lak
plt.plot(range(0, len(sgd_losses) * batch_size, batch_size), sgd_losses, label=f'SGD')
plt.xlabel('# Gradient computations')
plt.ylabel('Training loss')

plt.legend()
plt.show()
```

```
Final loss (gradient descent): 0.9677762153653575
Final loss (SGD): 0.9603657486061161
```



In [12]:

```
if embedding_type == 'hard-coded':
    parameter_table = []

    for i, description in enumerate(['Positive words', 'Negative words', '!', 'Log length'
        row = {
            'Feature': description,
        }
        row.update({
            label: parameters[j, i]
            for j, label in enumerate(labels)
        })
        parameter_table.append(row)

    pd.set_option('display.max_rows', None)
    pd.set_option('display.max_colwidth', None)
    display(pd.DataFrame(parameter_table))
```

	Feature	positive	neutral	negative

	Feature	positive	neutral	negative
0	Positive words	0.419649	0.362162	-0.780547
1	Negative words	-0.507407	0.070439	0.446864
2	!	0.459245	-0.305360	-0.158296
3	Log length	-0.670874	0.027195	0.684209
4	Bias term	0.708486	0.212304	0.054936

In [13]:

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

predictions = []
table = []

for review, label, x in zip(test_reviews, test_labels, test_embeddings):
    prob = softmax((parameters * x).sum(axis=1))
    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: 60.7%
F1 score: [0.66666667 0.5        0.57142857]
```

In [14]:

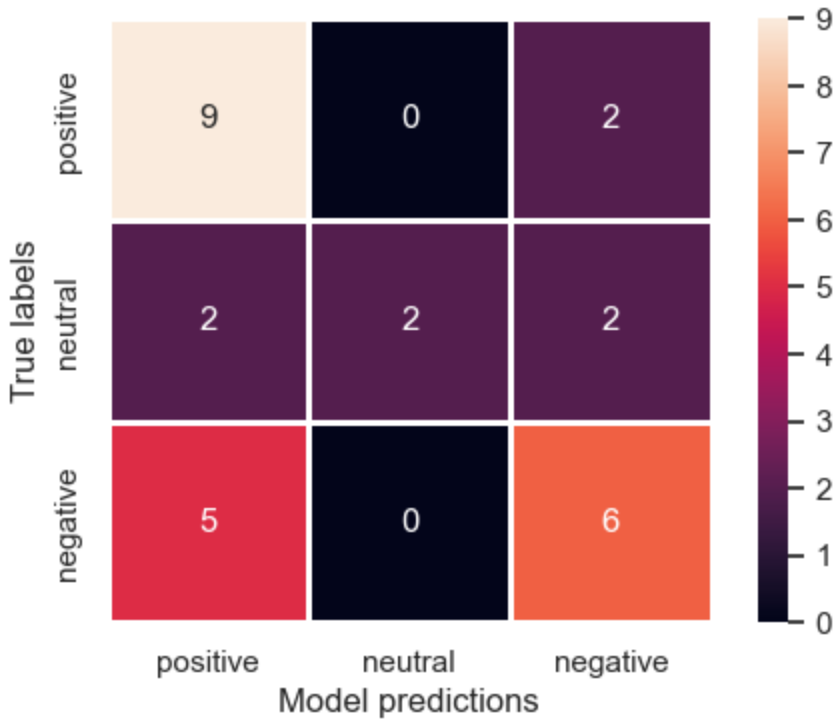
```
pd.set_option('display.max_rows', None)
pd.set_option('display.max_colwidth', None)
display(pd.DataFrame(table))
```

	Review	Label	Prediction	positive %	neutral %	negative %
0	The best show I've seen so far this year!	positive	positive	43.1	21.3	35.6
1	Really enjoyed it	positive	positive	37.5	29.8	32.7
2	Amazing.	positive	positive	51.4	30.7	17.9
3	Why all the hate? I enjoyed it.	positive	negative	31.8	29.9	38.3
4	Beautiful visuals, entertaining, and I believe this show has a lot of potential!	positive	positive	47.4	24.0	28.6
5	A beautiful rendering of Middle Earth's history	positive	positive	36.7	33.7	29.6
6	So far, so good... and there's still hope	positive	positive	34.3	33.9	31.7
7	I'm a fan	positive	positive	47.0	28.6	24.4
8	It works for me	positive	positive	35.1	29.9	35.0

	Review	Label	Prediction	positive %	neutral %	negative %
9	Not the best, but enjoyed every episode. Can't wait to see much much more.	positive	negative	25.4	29.2	45.4
10	Beautiful, flawed, and a wonderful Fall treat	positive	positive	36.7	33.7	29.6
11	Good show with too many subplots	neutral	positive	36.7	33.7	29.6
12	Starts badly, gets better	neutral	positive	40.0	33.4	26.7
13	Good and bad things	neutral	neutral	34.1	34.5	31.3
14	Big and beautiful but can use a little help with its energy.	neutral	negative	31.9	34.0	34.2
15	Pretty but ultimately hollow and lacking in engagement	neutral	negative	29.5	29.7	40.7
16	Beautiful visuals and story overshadowed by unnecessary gore and violence	neutral	neutral	32.6	34.0	33.5
17	Not what you're probably expecting	negative	negative	31.8	29.9	38.3
18	Poor writing; Uninteresting characters, nonsensical actions.	negative	negative	31.8	29.9	38.3
19	Budget was spent on snacks between shots	negative	negative	30.6	29.8	39.6
20	If you ignore the source material, it's still boring and weird	negative	negative	17.8	28.4	53.8
21	Just a bad show	negative	negative	31.7	30.5	37.9
22	It's awful	negative	positive	41.0	29.5	29.5
23	I was hopeful...	negative	positive	41.0	29.5	29.5
24	Painfully mediocre with a few good spots	negative	positive	36.7	33.7	29.6
25	Beautiful to look at... but that’s about it.	negative	positive	34.3	33.9	31.7
26	Underwhelming and disappointing	negative	positive	37.5	29.8	32.7
27	Tolkien is rolling in his grave. No mystery. No inspiration. Wardrobe & acting is pretty bland.	negative	negative	24.9	29.1	46.0

In [15]:

```
mat = confusion_matrix(test_labels, predictions, labels=labels)

sns.heatmap(mat, square=True, annot=True, xticklabels=labels, yticklabels=labels,
            linewidths=1)
plt.xlabel("Model predictions")
plt.ylabel("True labels")
plt.show()
```


positive
neutral
negative
True labels
9
0
2
2
2
2
5
0
6
positive
neutral
negative
Model predictions
9
8
7
6
5
4
3
2
1
0

In [ ]: