```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Mounted at /content/gdrive
```

```
import pandas as pd
import numpy as np
import bz2
import re
import matplotlib.pyplot as plt
import seaborn as sns
from tqdm import tqdm
from sklearn.utils import shuffle
from tqdm import tqdm
from keras.layers import *
from keras.models import Model
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
```

```
data = pd.read_csv('/content/gdrive/MyDrive/IMDB Dataset.csv', encoding="ISO-8859-1")
```

```
data['review'][2]
```

```
'I thought this was a wonderful way to spend time on a too hot summer weekend, sitti
ng in the air conditioned theater and watching a light-hearted comedy. The plot is s
implistic, but the dialogue is witty and the characters are likable (even the well b
read suspected serial killer). While some may be disappointed when they realize this
is not Match Point 2: Risk Addiction, I thought it was proof that Woody Allen is sti
ll fully in control of the style many of us have grown to love.<br /><br />This was
the most I\'d laughed at one of Woody\'s comedies in years (dare I say a decade?). W
hile I\'ve never been impressed with Scarlet Johanson, in this she managed to tone d
own her "sexy" image and jumped right into a average, but spirited young woman.<br /
```

```
# De dimensies van de dataset
data.shape
```

```
(50000, 2)
```

```
# Het aantal records
data.count()
```

```
review       50000
sentiment    50000
dtype: int64
```

```
# Het verwijderen van de stopwoorden
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import string
nltk.download('punkt')
```

```
def verwijderStopwoorden(text):
    stopwoorden = set(stopwords.words('english'))
    tokens = word_tokenize(text.lower())
    result = [x for x in tokens if x not in stopwoorden and not x.startswith('@')]
    seperator = ' '
    return seperator.join(result)

data['review'] = data['review'].map(verwijderStopwoorden)
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
```

```
data['sentiment'][data['sentiment']=='negative'] = 0
data['sentiment'][data['sentiment']=='positive'] = 1
```

```
def clean_html(text):
    clean = re.compile('<.*?>')
    return re.sub(clean,'',text)
```

```
data['review'] = data['review'].apply(clean_html)
data.head()
```

	review	sentiment
0	one reviewers mentioned watching 1 oz episode ...	1
1	wonderful little production . filming techni...	1
2	thought wonderful way spend time hot summer we...	1
3	basically 's family little boy (jake) thinks...	0
4	petter mattei 's `` love time money " visuall...	1

```
data['review'][7]
```

```
'show amazing , fresh & innovative idea 70 's first aired . first 7 8 years brillian
t , things dropped . 1990 , show really funny anymore , 's continued decline complet
e waste time today.   's truly disgraceful far show fallen . writing painfully bad ,
performances almost bad - mildly entertaining respite guest-hosts , show probably wo
uld n't still air . find hard believe creator hand-selected original cast also chose
band hacks followed . one recognize brilliance see fit replace mediocrity ? felt mus
t give 2 stars respect original cast made show huge success . , show awful . ca n't
```

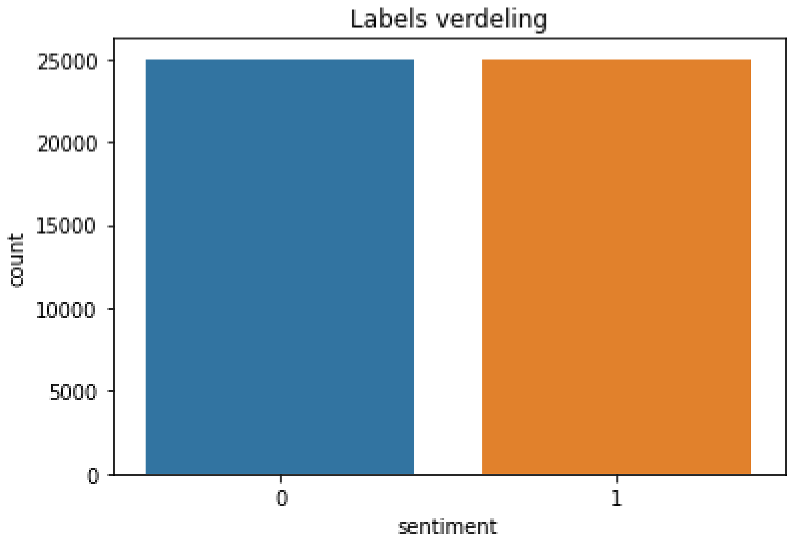
```
data.describe()
```

	review	sentiment
count	50000	50000

Dubbelklik (of druk op Enter) om te bewerken

```
sns.countplot(data['sentiment'])
plt.title('Labels verdeling')
```

```
    /usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pas
      FutureWarning
    Text(0.5, 1.0, 'Labels verdeling')
```



```
data['word_count'] = data['review'].apply(lambda x : len(x.split()))
data['char_count'] = data['review'].apply(lambda x : len(x.replace(" ","")))
data['word_density'] = data['word_count'] / (data['char_count'] + 1)
```

```
data.head()
```

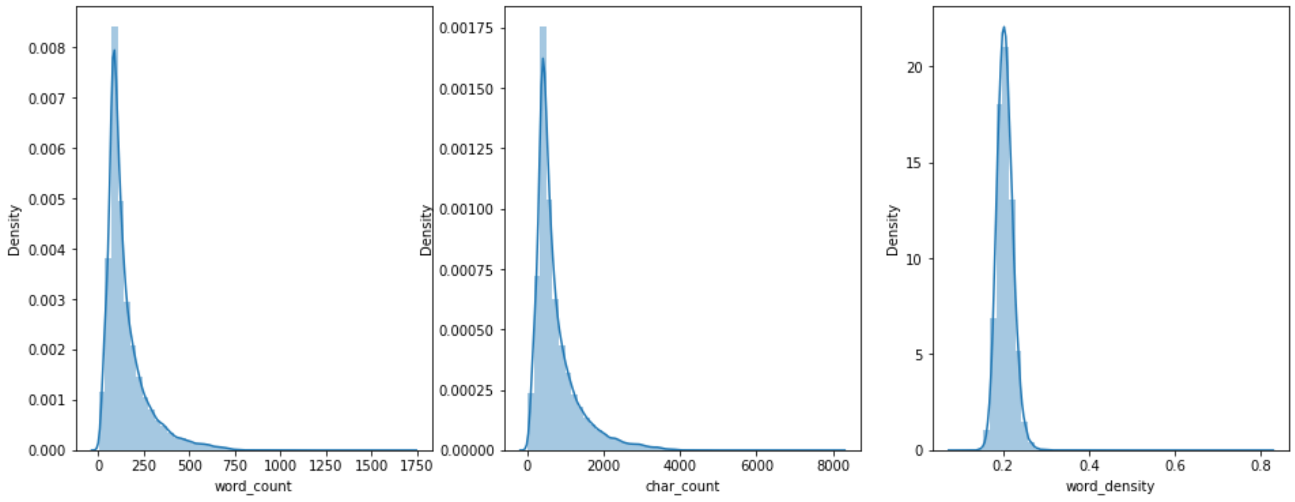
	review	sentiment	word_count	char_count	word_density
0	one reviewers mentioned watching 1 oz episode ...	1	213	1029	0.206796
1	wonderful little production . filming techni...	1	105	601	0.174419
2	thought wonderful way spend time hot summer we...	1	109	528	0.206049

```
data.describe()
```

	word_count	char_count	word_density
count	50000.000000	50000.000000	50000.000000
mean	152.845840	753.534140	0.205559
std	115.158279	583.175244	0.019474
min	5.000000	18.000000	0.080537
25%	82.000000	393.000000	0.192580

```
fig, ax = plt.subplots(1, 3, figsize=(16, 6))
dp=sns.distplot(data['word_count'],ax=ax[0])
dp=sns.distplot(data['char_count'],ax=ax[1])
dp=sns.distplot(data['word_density'],ax=ax[2])
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2557: FutureWarning:
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2557: FutureWarning:
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2557: FutureWarning:
  warnings.warn(msg, FutureWarning)
```



```
dataRF = data
```

```
dataRF = dataRF.drop(['word_count', 'char_count','word_density'], axis=1)
```

```
try:
  %tensorflow_version 2.x
except Exception:
  pass
```

```
import tensorflow as tf
import tensorflow as tf
from tensorflow import keras

print(tf.__version__)

import numpy as np
import matplotlib.pyplot as plt
import sklearn as sk
import pandas as pd

seed = 2020
np.random.seed(seed)

import sklearn as sk
from sklearn.model_selection import train_test_split

from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Flatten, Embedding, Conv1D,  MaxPoolin
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.constraints import max_norm
from tensorflow.keras.models import load_model


import nltk
```

```
     2.4.1
```

```
def plot_history(history):
  plt.figure(figsize = (12,4))
  plt.subplot(1,2,1)

  plt.xlabel('Epoch')
  plt.ylabel('Nauwkeurigheid')
  plt.plot(history.epoch, np.array(history.history['accuracy']),'g-',
           label='Train dataset nauwkeurigheid')
  plt.plot(history.epoch, np.array(history.history['val_accuracy']),'r-',
           label = 'Validatie dataset nauwkeurigheid')
  plt.legend()

  plt.subplot(1,2,2)
  plt.xlabel('Epoch')
  plt.ylabel('Verlies')
  plt.plot(history.epoch, np.array(history.history['loss']),'g-',
           label='Train dataset verlies')
  plt.plot(history.epoch, np.array(history.history['val_loss']),'r-',
           label = 'Validatie dataset verlies')
  plt.legend()
```

```
# De lengte van de woorden in een nieuwe kolom
dataRF['numberOfWords'] = dataRF.review.str.split().apply(len)
dataRF.head()
```

	review	sentiment	numberOfWords
0	one reviewers mentioned watching 1 oz episode ...	1	213
1	wonderful little production . filming techni...	1	105
2	thought wonderful way spend time hot summer we...	1	109
3	basically 's family little boy (jake) thinks...	0	85
4	petter mattei 's `` love time money '' visuall...	1	151

```
dataRF['numberOfWords'].describe()
```

```
count    50000.000000
mean       152.845840
std        115.158279
min          5.000000
25%         82.000000
50%        114.000000
75%        186.000000
max       1705.000000
Name: numberOfWords, dtype: float64
```

```
# Een training dataset opzetten
from sklearn.model_selection import train_test_split
X = dataRF.drop(['sentiment','numberOfWords'],axis=1)
y = dataRF['sentiment']
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.30)

X_train = np.asarray(X_train)
X_test = np.asarray(X_test)
```

```
num_classes = 2
#De methode utils.to_categorical zet vectors om in binaire matrices
#De scores (0 of 1) worden omgezet naar een binaire matrix. Het aantal klassen is
#twee omdat er twee opties zijn: positief of negatief
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)
```

```
from tensorflow.keras.layers.experimental.preprocessing import TextVectorization
#TextVectorization zet een lijst van strings om in een lijst van tokens
vectorizer = TextVectorization(max_tokens=20000, output_sequence_length=20)
text_ds = tf.data.Dataset.from_tensor_slices(X_train).batch(128)
vectorizer.adapt(text_ds)
```

```
voc = vectorizer.get_vocabulary()
word_index = dict(zip(voc, range(len(voc))))
```

```
#Glove staat voor Global Vector
glove_file = '/content/gdrive/My Drive/glove.6B.100d.txt'

#Glove-files bevatten woord vectors. De file die hier gebruikt wordt, bevat 400.000 vector
embeddings_index = {}
```

```
with open(glove_file) as f:
    for line in f:
      values = line.split()
      woord = values[0]
      coefs = np.asarray(values[1:], dtype='float32')
      embeddings_index[woord] = coefs


num_tokens = len(voc) + 2
embedding_dim = 100
missed_words = []

# Een embedding matrix aanmaken
embedding_matrix = np.zeros((num_tokens, embedding_dim))
for word, i in word_index.items():
    embedding_vector = embeddings_index.get(word)
    if embedding_vector is not None:
        embedding_matrix[i] = embedding_vector
    else:
        missed_words.append(word)




num_classes = 2
#Het model initialiseren
def initial_model():
    model = Sequential()

    model.add(Embedding(num_tokens, embedding_dim, embeddings_initializer=keras.initialize
    model.add(Conv1D(16,activation='relu',kernel_size=3))
    model.add(MaxPooling1D(3))
    model.add(Dropout(0.2))

    model.add(Conv1D(16,activation='relu',kernel_size=3))
    model.add(Dropout(0.2))

    model.add(Conv1D(16,activation='relu',kernel_size=3))
    model.add(GlobalMaxPooling1D())
    model.add(Dense(16, activation='relu', kernel_initializer='he_uniform'))
    model.add(Dropout(0.2))
    model.add(Dense(num_classes, activation='softmax'))

#Categorical Crosssentropy berekent het cross-entropie verlies tussen de labels en de voor
#Dit is een optimalisator die het Adam-algoritme implementeert.
    model.compile(loss='categorical_crossentropy',
                  optimizer= tf.keras.optimizers.Adam(learning_rate = 0.0001),
                  metrics=['accuracy'])
    return model


#Het model verwacht een array, dus dit wordt hier omgezet
X_train_final = vectorizer(np.array([s for s in X_train])).numpy()
X_test_final = vectorizer(np.array([s for s in X_test])).numpy()

y_train_final = np.array(y_train)
```

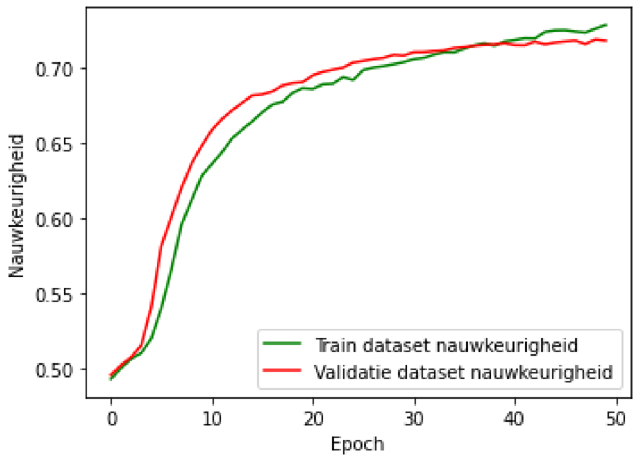
```
y_test_final = np.array(y_test)


model_1 = initial_model()
model_1.summary()
batch_size = 128
epochs = 50

history_1 = model_1.fit(X_train_final, y_train_final,
                    batch_size=batch_size,
                    epochs=epochs,
                    verbose=1,
                    validation_data=(X_test_final, y_test_final)
                    )
```

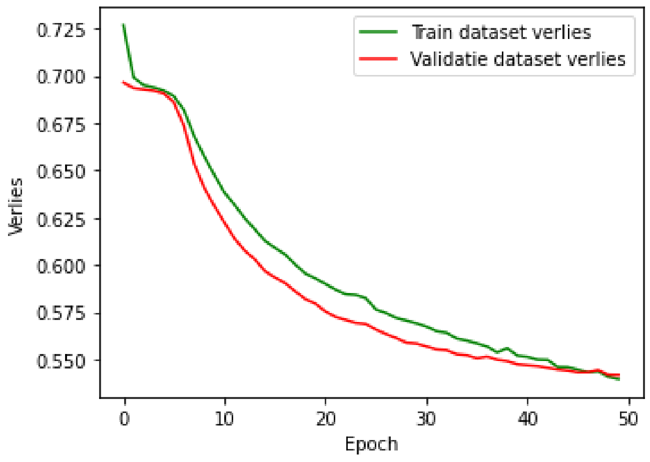
```
Epoch 16/50
274/274 [==============================] - 3s 10ms/step - loss: 0.6103 - accuracy:
Epoch 17/50
274/274 [==============================] - 3s 10ms/step - loss: 0.6075 - accuracy:
Epoch 18/50
274/274 [==============================] - 3s 10ms/step - loss: 0.6026 - accuracy:
Epoch 19/50
274/274 [==============================] - 3s 10ms/step - loss: 0.5975 - accuracy:
Epoch 20/50
274/274 [==============================] - 3s 10ms/step - loss: 0.5915 - accuracy:
Epoch 21/50
274/274 [==============================] - 3s 10ms/step - loss: 0.5897 - accuracy:
Epoch 22/50
274/274 [==============================] - 3s 10ms/step - loss: 0.5896 - accuracy:
Epoch 23/50
274/274 [==============================] - 3s 10ms/step - loss: 0.5851 - accuracy:
Epoch 24/50
274/274 [==============================] - 3s 10ms/step - loss: 0.5847 - accuracy:
Epoch 25/50
274/274 [==============================] - 3s 10ms/step - loss: 0.5803 - accuracy:
Epoch 26/50
274/274 [==============================] - 3s 10ms/step - loss: 0.5794 - accuracy:
Epoch 27/50
274/274 [==============================] - 3s 10ms/step - loss: 0.5748 - accuracy:
Epoch 28/50
274/274 [==============================] - 3s 10ms/step - loss: 0.5729 - accuracy:
Epoch 29/50
274/274 [==============================] - 3s 10ms/step - loss: 0.5722 - accuracy:
Epoch 30/50
274/274 [==============================] - 3s 10ms/step - loss: 0.5730 - accuracy:
Epoch 31/50
274/274 [==============================] - 3s 10ms/step - loss: 0.5668 - accuracy:
Epoch 32/50
274/274 [==============================] - 3s 10ms/step - loss: 0.5653 - accuracy:
Epoch 33/50
274/274 [==============================] - 3s 10ms/step - loss: 0.5643 - accuracy:
Epoch 34/50
274/274 [==============================] - 3s 10ms/step - loss: 0.5630 - accuracy:
Epoch 35/50
274/274 [==============================] - 3s 10ms/step - loss: 0.5617 - accuracy:
Epoch 36/50
274/274 [==============================] - 3s 10ms/step - loss: 0.5607 - accuracy:
Epoch 37/50
```

```
274/274 [==============================] - 3s 10ms/step - loss: 0.5527 - accuracy:
Epoch 38/50
274/274 [==============================] - 3s 10ms/step - loss: 0.5522 - accuracy:
Epoch 39/50
274/274 [==============================] - 3s 10ms/step - loss: 0.5537 - accuracy:
Epoch 40/50
274/274 [==============================] - 3s 10ms/step - loss: 0.5526 - accuracy:
Epoch 41/50
274/274 [==============================] - 3s 10ms/step - loss: 0.5521 - accuracy:
Epoch 42/50
274/274 [==============================] - 3s 10ms/step - loss: 0.5515 - accuracy:
Epoch 43/50
274/274 [==============================] - 3s 10ms/step - loss: 0.5506 - accuracy:
Epoch 44/50
274/274 [==============================] - 3s 10ms/step - loss: 0.5447 - accuracy:
Epoch 45/50
```

```
#De resultaten visualiseren
[train_loss, train_accuracy] = model_1.evaluate(X_train_final, y_train_final, verbose=0)
print("Nauwkeurigheid training dataset:{:7.2f}".format(train_accuracy))
[val_loss, val_accuracy] = model_1.evaluate(X_test_final, y_test_final, verbose=0)
print("Nauwkeurigheid test dataset:{:7.2f}".format(val_accuracy))
plot_history(history_1)
```

```
Nauwkeurigheid training dataset:   0.75
Nauwkeurigheid test dataset:   0.72
```


Nauwkeurigheid
Epoch
Train dataset nauwkeurigheid
Validatie dataset nauwkeurigheid


Verlies
Epoch
Train dataset verlies
Validatie dataset verlies