```
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 wee
     kend, sitting in the air conditioned theater and watching a light-hearted
     comedy. The plot is simplistic, but the dialogue is witty and the charact
     ers are likable (even the well bread suspected serial killer). While some
```

```
# 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')
```

Opgeslagen. ×

```
    stopwoorden = set(stopwords.words('english'))
    tokens = word_tokenize(text.lower())
    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 yea
rs brilliant , things dropped . 1990 , show really funny anymore , 's con
tinued decline complete waste time today.   's truly disgraceful far show
```

```
data.describe()
```

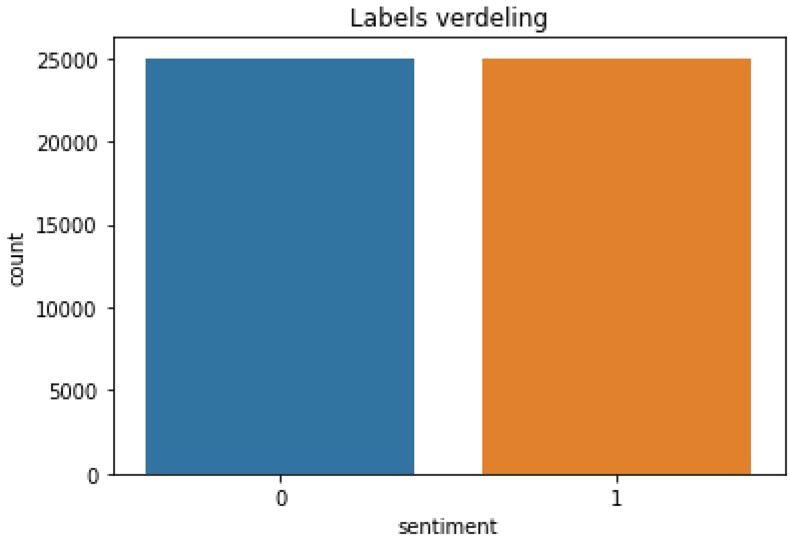
	review	sentiment
count	50000	50000
unique	49578	2
top	loved today 's show ! ! ! variety solely cooki...	1
freq	5	25000

Opgeslagen. ✕

verken

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

```
    /usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWa
      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
	thought wonderful way				

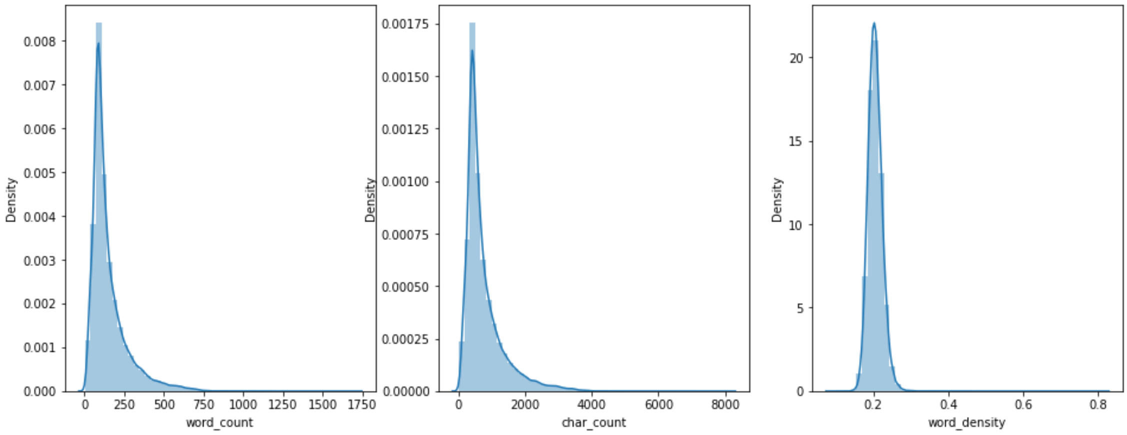
```
data.describe()
```

Opgeslagen. ×

	word_count	char_count	word_density
count	50000.000000	50000.000000	50000.000000

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

print(tf.__version__)
```

Opgeslagen. ×

```
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
```

VERBORGEN UITVOER WEERGEVEN

```
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()
```

Opgeslagen. ×

```
                                           review  sentiment  numberOfWords
0  one reviewers mentioned watching 1 oz episode ...          1            213
```

```
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()
                                             dtype='float32')
                                           s
```

Opgeslagen. ×

```
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)




#Het model initialiseren
def initial_model3():
    model = Sequential()

    model.add(Embedding(num_tokens, embedding_dim,
                        embeddings_initializer=keras.initializers.Constant(embedding_matri
    model.add(LSTM(2))

    model.compile(loss='binary_crossentropy',
                  optimizer= 'adam',
                  metrics=['accuracy'])

    return model


def initial_model4():
    model = Sequential()

    model.add(Embedding(num_tokens, embedding_dim,
                        embeddings_initializer=keras.initializers.Constant(embedding_matri
    model.add(Dropout(0.20))
    model.add(Conv1D(16,activation='relu', kernel_size=2))
    model.add(GlobalMaxPooling1D())
    model.add(Dropout(0.20))
    model.add(Dense(num_classes, activation='sigmoid'))

    model.compile(loss='binary_crossentropy',
                  optimizer= 'adam',
                  metrics=['accuracy'])

    return model


def initial_model2():
    model = Sequential()

                  embedding_dim,
                  nitializer=keras.initializers.Constant(embedding_matri
    model.add(Bidirectional(LSTM(16, dropout=0.2, recurrent_dropout=0.2)))
```

Opgeslagen. ✕

```
    model.add(Dense(32, activation='relu'))
    model.add(Dropout(0.50))
    model.add(Dense(2, activation='softmax'))

    model.compile(loss='categorical_crossentropy',
                  optimizer= 'adam',
                  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_model4()
model_1.summary()
batch_size = 128
epochs = 30

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)
                    )
```

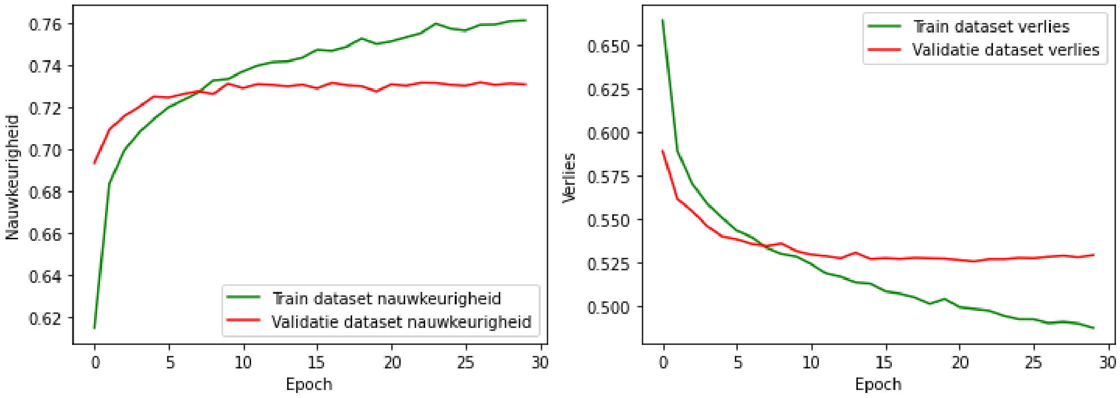
```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, None, 100)         2000200
_________________________________________________________________
dropout_1 (Dropout)          (None, None, 100)         0
_________________________________________________________________
conv1d_1 (Conv1D)            (None, None, 16)          3216
_________________________________________________________________
global_max_pooling1d_1 (Glob (None, 16)                0
_________________________________________________________________
dropout_2 (Dropout)          (None, 16)                0
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 34
=================================================================
Total params: 2,003,450
Trainable params: 3,250
Non-trainable params: 2,000,200
_________________________________________________________________
Epoch 1/30
274/274 [==============================] - 3s 10ms/step - loss: 0.7905 - accuracy:
Epoch 2/30
                              =========] - 2s 9ms/step - loss: 0.6281 - accuracy:

                              =========] - 2s 9ms/step - loss: 0.5982 - accuracy:
```

Opgeslagen.

```
Epoch 4/30
274/274 [==============================] - 2s 9ms/step - loss: 0.5825 - accuracy:
Epoch 5/30
274/274 [==============================] - 2s 9ms/step - loss: 0.5784 - accuracy:
Epoch 6/30
274/274 [==============================] - 2s 9ms/step - loss: 0.5726 - accuracy:
Epoch 7/30
274/274 [==============================] - 2s 9ms/step - loss: 0.5683 - accuracy:
Epoch 8/30
274/274 [==============================] - 2s 9ms/step - loss: 0.5659 - accuracy:
Epoch 9/30
274/274 [==============================] - 2s 9ms/step - loss: 0.5578 - accuracy:
Epoch 10/30
274/274 [==============================] - 2s 9ms/step - loss: 0.5586 - accuracy:
Epoch 11/30
274/274 [==============================] - 2s 9ms/step - loss: 0.5562 - accuracy:
Epoch 12/30
274/274 [==============================] - 2s 9ms/step - loss: 0.5508 - accuracy:
Epoch 13/30
274/274 [==============================] - 2s 9ms/step - loss: 0.5517 - accuracy:
Epoch 14/30
274/274 [==============================] - 2s 9ms/step - loss: 0.5469 - accuracy:
Epoch 15/30
274/274 [==============================] - 2s 9ms/step - loss: 0.5510 - accuracy:
Epoch 16/30
274/274 [==============================] - 2s 9ms/step - loss: 0.5436 - accuracy:
Epoch 17/30
274/274 [==============================] - 2s 9ms/step - loss: 0.5461 - accuracy:
Epoch 18/30
274/274 [==============================] - 2s 9ms/step - loss: 0.5404 - accuracy:
Epoch 19/30
274/274 [==============================] - 2s 9ms/step - loss: 0.5461 - accuracy:
```

```
#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:   0.79
Nauwkeurigheid test dataset:   0.73
```



Opgeslagen. ×