```
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
```

```
data1 = bz2.BZ2File('/content/gdrive/MyDrive/train.ft.txt.bz2')
```

```
data1 = data1.readlines()
```

```
data = [x.decode('utf-8') for x in data1]
```

```
data = data[:200000]
```

```
train_labels = [0 if x.split(' ')[0] == '__label__1' else 1 for x in data]
train_sentences = [x.split(' ', 1)[1][:-1].lower() for x in data]
```

```
for i in range(len(train_sentences)):
    if 'www.' in train_sentences[i] or 'http:' in train_sentences[i] or 'https:' in train_
        train_sentences[i] = re.sub(r"([^ ]+(?<=\.[a-z]{3}))", "<url>", train_sentences[i]
```

```
data = pd.DataFrame(train_sentences, columns = ['review'])
data.head()
```

| | review |
|---|---|
| **0** | stuning even for the non-gamer: this sound tra... |
| **1** | the best soundtrack ever to anything.: i'm rea... |
| **2** | amazing!: this soundtrack is my favorite music... |
| **3** | excellent soundtrack: i truly like this soundt... |
| **4** | remember, pull your jaw off the floor after he... |

```
data['score'] = train_labels
data.head()
```

| | review | score |
|---|---|---|
| **0** | stuning even for the non-gamer: this sound tra... | 1 |
| **1** | the best soundtrack ever to anything.: i'm rea... | 1 |
| **2** | amazing!: this soundtrack is my favorite music... | 1 |
| **3** | excellent soundtrack: i truly like this soundt... | 1 |
| **4** | remember, pull your jaw off the floor after he... | 1 |

```
# De algemene info over de dataset
data.describe()
```

| | score |
|---|---|
| **count** | 200000.000000 |
| **mean** | 0.505830 |
| **std** | 0.499967 |
| **min** | 0.000000 |
| **25%** | 0.000000 |
| **50%** | 1.000000 |
| **75%** | 1.000000 |
| **max** | 1.000000 |

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

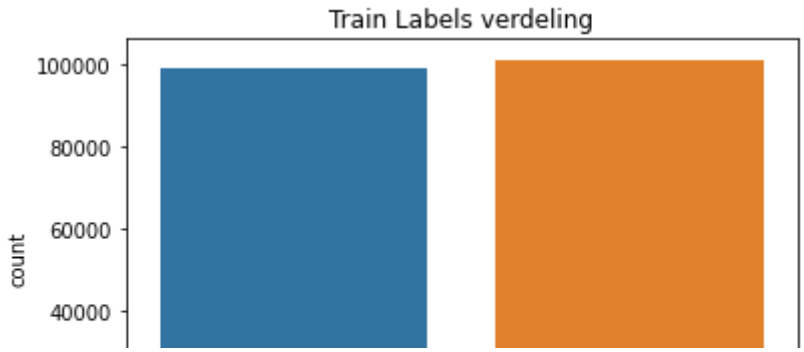
```
(200000, 2)
```

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

```
review    200000
score     200000
dtype: int64
```

```
sns.countplot(train_labels)
plt.title('Train Labels verdeling')
```

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

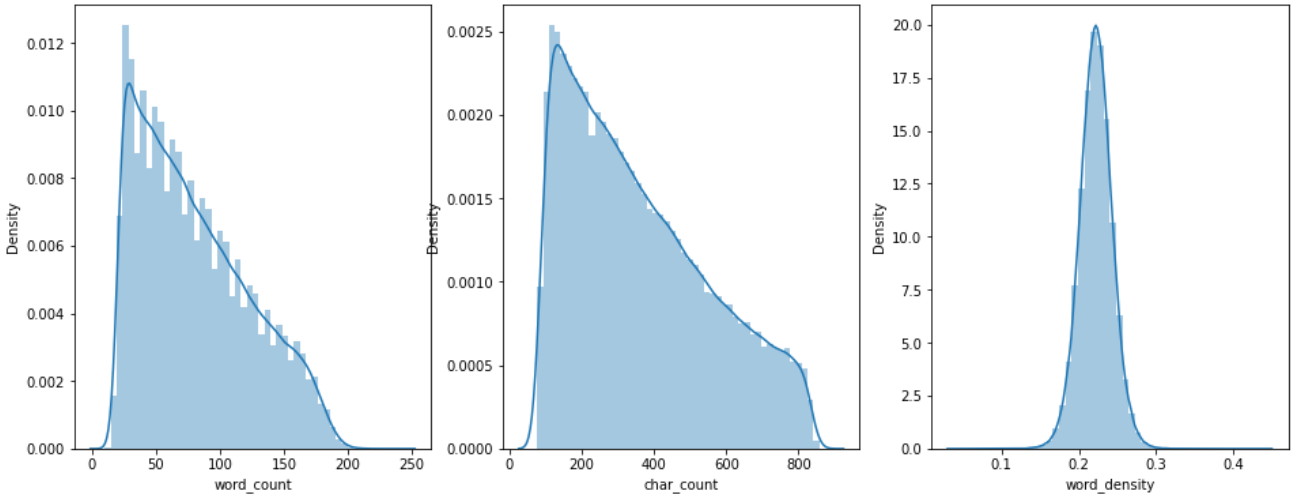
| | review | score | word_count | char_count | word_density |
|---|---|---|---|---|---|
| **0** | stuning even for the non-gamer: this sound tra... | 1 | 80 | 347 | 0.229885 |
| **1** | the best soundtrack ever to anything.: i'm rea... | 1 | 97 | 413 | 0.234300 |
| **2** | amazing!: this soundtrack is my favorite music... | 1 | 129 | 632 | 0.203791 |

```
data.describe()
```

| | score | word_count | char_count | word_density |
|---|---|---|---|---|
| **count** | 200000.000000 | 200000.000000 | 200000.000000 | 200000.000000 |
| **mean** | 0.505830 | 80.092535 | 361.827595 | 0.222160 |
| **std** | 0.499967 | 43.251276 | 197.531975 | 0.021086 |
| **min** | 0.000000 | 10.000000 | 78.000000 | 0.035294 |
| **25%** | 0.000000 | 44.000000 | 194.000000 | 0.208556 |
| **50%** | 1.000000 | 72.000000 | 323.000000 | 0.222222 |
| **75%** | 1.000000 | 111.000000 | 500.000000 | 0.235669 |
| **max** | 1.000000 | 241.000000 | 873.000000 | 0.444444 |

```
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)
```




```
# 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]
    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['review'].head()
```

```
0    stuning even non-gamer : sound track beautiful...
1    best soundtrack ever anything . : 'm reading l...
```

```
2    amazing ! : soundtrack favorite music time , h...
3    excellent soundtrack : truly like soundtrack e...
4    remember , pull jaw floor hearing : 've played...
Name: review, dtype: object
```

```
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__)

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
```

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()
```

| | review | score | numberOfWords |
|---|---|---|---|
| **0** | stuning even non-gamer : sound track beautiful... | 1 | 51 |
| **1** | best soundtrack ever anything . : 'm reading l... | 1 | 58 |
| **2** | amazing ! : soundtrack favorite music time , h... | 1 | 108 |
| **3** | excellent soundtrack : truly like soundtrack e... | 1 | 101 |
| **4** | remember , pull jaw floor hearing : 've played... | 1 | 68 |

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

```
count    200000.000000
mean         53.914610
std          28.933542
min           7.000000
25%          30.000000
50%          48.000000
75%          74.000000
max         433.000000
Name: numberOfWords, dtype: float64
```

```
# Een training dataset opzetten
from sklearn.model_selection import train_test_split
X = dataRF.drop(['score','numberOfWords'],axis=1)
y = dataRF['score']
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)




def initial_model2():
    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='categorical_crossentropy',
                  optimizer= 'adam',
                  metrics=['accuracy'])

    return model
```

```
    return model


#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


#Het model initialiseren
def initial_model4():
    model = Sequential()

    model.add(Embedding(num_tokens, embedding_dim,
                        embeddings_initializer=keras.initializers.Constant(embedding_matri
    model.add(Bidirectional(LSTM(16, dropout=0.2, recurrent_dropout=0.2)))
    model.add(Dense(16, 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_2"
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_2 (Embedding)      (None, None, 100)         2000200
_________________________________________________________________
dropout_1 (Dropout)          (None, None, 100)         0
_________________________________________________________________
conv1d_2 (Conv1D)            (None, None, 16)          3216
_________________________________________________________________
global_max_pooling1d_2 (Glob (None, 16)                0
_________________________________________________________________
dropout_2 (Dropout)          (None, 16)                0
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 34
=================================================================
Total params: 2,003,450
Trainable params: 3,250
Non-trainable params: 2,000,200
_________________________________________________________________
Epoch 1/30
1094/1094 [==============================] - 10s 9ms/step - loss: 0.6226 - accuracy
Epoch 2/30
1094/1094 [==============================] - 9s 8ms/step - loss: 0.4823 - accuracy
Epoch 3/30
1094/1094 [==============================] - 9s 8ms/step - loss: 0.4683 - accuracy
Epoch 4/30
1094/1094 [==============================] - 9s 9ms/step - loss: 0.4590 - accuracy
Epoch 5/30
1094/1094 [==============================] - 10s 9ms/step - loss: 0.4577 - accuracy
Epoch 6/30
1094/1094 [==============================] - 10s 9ms/step - loss: 0.4505 - accuracy
Epoch 7/30
1094/1094 [==============================] - 10s 9ms/step - loss: 0.4517 - accuracy
Epoch 8/30
1094/1094 [==============================] - 10s 9ms/step - loss: 0.4448 - accuracy
Epoch 9/30
1094/1094 [==============================] - 9s 9ms/step - loss: 0.4461 - accuracy
Epoch 10/30
1094/1094 [==============================] - 9s 8ms/step - loss: 0.4452 - accuracy
Epoch 11/30
1094/1094 [==============================] - 9s 8ms/step - loss: 0.4423 - accuracy
Epoch 12/30
1094/1094 [==============================] - 9s 8ms/step - loss: 0.4426 - accuracy
Epoch 13/30
1094/1094 [==============================] - 9s 8ms/step - loss: 0.4383 - accuracy
Epoch 14/30
1094/1094 [==============================] - 9s 8ms/step - loss: 0.4432 - accuracy
Epoch 15/30
1094/1094 [==============================] - 9s 8ms/step - loss: 0.4407 - accuracy
Epoch 16/30
1094/1094 [==============================] - 9s 8ms/step - loss: 0.4403 - accuracy
Epoch 17/30
1094/1094 [==============================] - 9s 8ms/step - loss: 0.4383 - accuracy
Epoch 18/30
1094/1094 [==============================] - 9s 8ms/step - loss: 0.4387 - accuracy
Epoch 19/30
1094/1094 [==============================] - 9s 8ms/step - loss: 0.4389 - 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:   0.83
Nauwkeurigheid test dataset:   0.82
```