```
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/Tweets.csv', encoding="ISO-8859-1")
```

```
data = data.drop(["airline_sentiment_gold", "negativereason_gold", "tweet_coord", "tweet_l
```

```
data = data[data['airline_sentiment_confidence'] >= 0.5][['airline_sentiment', 'text']]
```

```
data.head()
```

	airline_sentiment	text
0	neutral	@VirginAmerica What @dhepburn said.
2	neutral	@VirginAmerica I didn't today... Must mean I n...
3	negative	@VirginAmerica it's really aggressive to blast...
4	negative	@VirginAmerica and it's a really big bad thing...
5	negative	@VirginAmerica seriously would pay $30 a fligh...

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

```
(14404, 2)
```

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

```
airline_sentiment    14404
text                 14404
dtype: int64
```

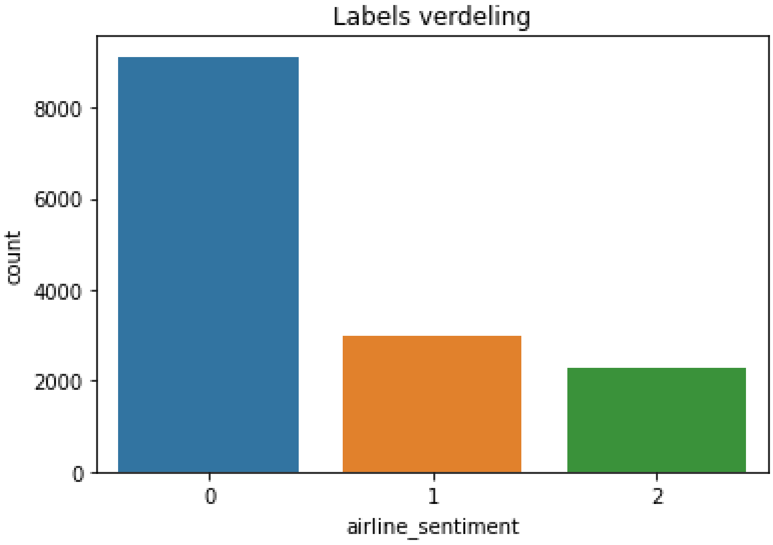
```
# Het verwijderen van de stopwoorden
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import string
nltk.download('punkt')

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

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

```
data['airline_sentiment'][data['airline_sentiment']=='negative'] = 0
data['airline_sentiment'][data['airline_sentiment']=='neutral'] = 1
data['airline_sentiment'][data['airline_sentiment']=='positive'] = 2
```

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

```
data.head()
```

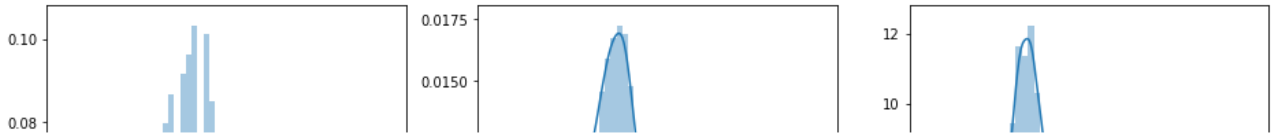
	airline_sentiment	text	word_count	char_count	word_density
0	1	virginamerica dhepburn said .	6	28	0.206897
2	1	virginamerica n't today ... must mean need tak...	12	53	0.222222
3	0	virginamerica 's really aggressive blast obnox...	18	94	0.189474
		virginamerica 's really big			

```
data.describe()
```

	word_count	char_count	word_density
count	14404.000000	14404.000000	14404.000000
mean	14.647390	67.829631	0.216095
std	5.289233	24.349557	0.036742
min	2.000000	8.000000	0.077778
25%	11.000000	50.000000	0.191011
50%	15.000000	71.000000	0.212766
75%	18.000000	86.000000	0.236842
max	40.000000	208.000000	0.519231

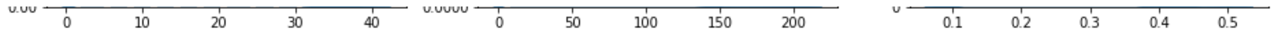
```
fig, ax = plt.subplots(1, 3, figsize=(16, 6))
dp=sns.distplot(data['word_count'],ax=ax[0])
dp=sns.distplot(data['char_count'],ax=ax[1])
dp=sns.distplot(data['word_density'],ax=ax[2])
plt.show()
```

```
/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)
```



```
data['text'].head()
```

```
0                          virginamerica dhepburn said .
2    virginamerica n't today ... must mean need tak...
3    virginamerica 's really aggressive blast obnox...
4                 virginamerica 's really big bad thing
5    virginamerica seriously would pay $ 30 flight ...
Name: text, 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
```

```
    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.text.str.split().apply(len)
dataRF.head()
```

	airline_sentiment	text	numberOfWords
0	1	virginamerica dhepburn said .	4
2	1	virginamerica n't today ... must mean need tak...	11
3	0	virginamerica 's really aggressive blast obnox...	17
4	0	virginamerica 's really big bad thing	6
5	0	virginamerica seriously would pay $ 30 flight ...	17

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

```
    count    14404.000000
    mean        13.516523
    std          5.242989
    min          1.000000
    25%         10.000000
    50%         14.000000
    75%         17.000000
    max         39.000000
    Name: numberOfWords, dtype: float64
```

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

```
y = dataRF['airline_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 = 3
#De methode utils.to_categorical zet vectors om in binaire matrices
#De scores (0, 1of 2) worden omgezet naar een binaire matrix. Het aantal klassen is
#drie omdat er drie opties zijn: positief, neutraal 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 = 3
#Het model initialiseren
```

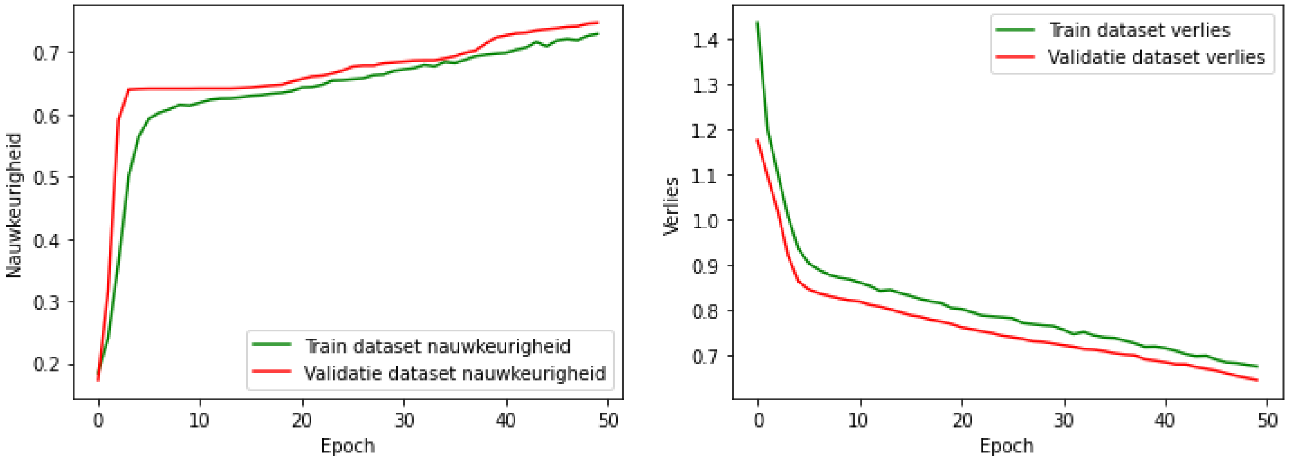
```
def initial_model():
    model = Sequential()

    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 17/50
79/79 [==============================] - 1s 11ms/step - loss: 0.8217 - accuracy: 0
Epoch 18/50
79/79 [==============================] - 1s 11ms/step - loss: 0.8196 - accuracy: 0
Epoch 19/50
79/79 [==============================] - 1s 11ms/step - loss: 0.8145 - accuracy: 0
Epoch 20/50
79/79 [==============================] - 1s 11ms/step - loss: 0.8016 - accuracy: 0

Epoch 21/50
```

```
79/79 [==============================] - 1s 11ms/step - loss: 0.8089 - accuracy: 0
Epoch 22/50
79/79 [==============================] - 1s 11ms/step - loss: 0.7925 - accuracy: 0
Epoch 23/50
79/79 [==============================] - 1s 11ms/step - loss: 0.7848 - accuracy: 0
Epoch 24/50
79/79 [==============================] - 1s 11ms/step - loss: 0.7947 - accuracy: 0
Epoch 25/50
79/79 [==============================] - 1s 11ms/step - loss: 0.7888 - accuracy: 0
Epoch 26/50
79/79 [==============================] - 1s 11ms/step - loss: 0.7781 - accuracy: 0
Epoch 27/50
79/79 [==============================] - 1s 11ms/step - loss: 0.7692 - accuracy: 0
Epoch 28/50
79/79 [==============================] - 1s 11ms/step - loss: 0.7818 - accuracy: 0
Epoch 29/50
79/79 [==============================] - 1s 11ms/step - loss: 0.7684 - accuracy: 0
Epoch 30/50
79/79 [==============================] - 1s 11ms/step - loss: 0.7663 - accuracy: 0
Epoch 31/50
79/79 [==============================] - 1s 10ms/step - loss: 0.7533 - accuracy: 0
Epoch 32/50
79/79 [==============================] - 1s 11ms/step - loss: 0.7490 - accuracy: 0
Epoch 33/50
79/79 [==============================] - 1s 11ms/step - loss: 0.7483 - accuracy: 0
Epoch 34/50
79/79 [==============================] - 1s 11ms/step - loss: 0.7378 - accuracy: 0
Epoch 35/50
79/79 [==============================] - 1s 11ms/step - loss: 0.7497 - accuracy: 0
Epoch 36/50
79/79 [==============================] - 1s 11ms/step - loss: 0.7370 - accuracy: 0
Epoch 37/50
79/79 [==============================] - 1s 11ms/step - loss: 0.7250 - accuracy: 0
Epoch 38/50
79/79 [==============================] - 1s 11ms/step - loss: 0.7126 - accuracy: 0
Epoch 39/50
79/79 [==============================] - 1s 11ms/step - loss: 0.7227 - accuracy: 0
Epoch 40/50
79/79 [==============================] - 1s 11ms/step - loss: 0.7256 - accuracy: 0
Epoch 41/50
79/79 [==============================] - 1s 11ms/step - loss: 0.7180 - accuracy: 0
Epoch 42/50
79/79 [==============================] - 1s 11ms/step - loss: 0.7140 - accuracy: 0
Epoch 43/50
79/79 [==============================] - 1s 10ms/step - loss: 0.6926 - accuracy: 0
Epoch 44/50
79/79 [==============================] - 1s 10ms/step - loss: 0.6970 - accuracy: 0
Epoch 45/50
79/79 [==============================] - 1s 11ms/step - loss: 0.7017 - accuracy: 0
```

```
#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:   0.76
Nauwkeurigheid test dataset:   0.75
```



✓ 1 s voltooid om 19:27