KAGGLE CHALLENGE: REAL OR NOT? NLP WITH DISASTER TWEETS

Machine Learning with TensorFlow SoSe2020

Ina Schicke, Moritz Larsen

THE CHALLENGE

- goal: build a machine learning model to predict wether a tweet is about a real disaster or not
- dataset: ~11,000 handclassified Tweets
  - training set: 7,613 tweets
    - variables: id, keyword, location, text, target
  - test set: 3,263 tweets
    - variables: id, keyword, location, text

THE DATA

- location

  4435 different locations

  33% missing data in train and test set

- keyword

  222 different keywords

  0.8% missing data in train and test set

- text

  contains the different tweets

  text cleaning was required

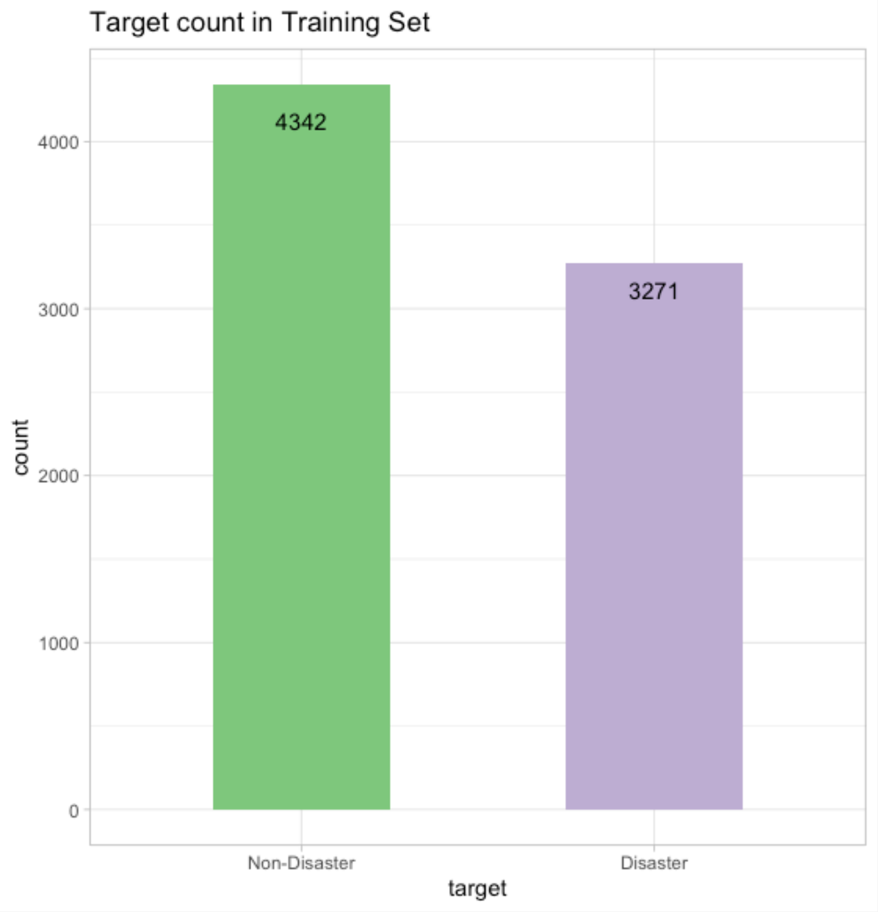
- target

  categorical variable

  either 1 = disaster or 0 = no disaster

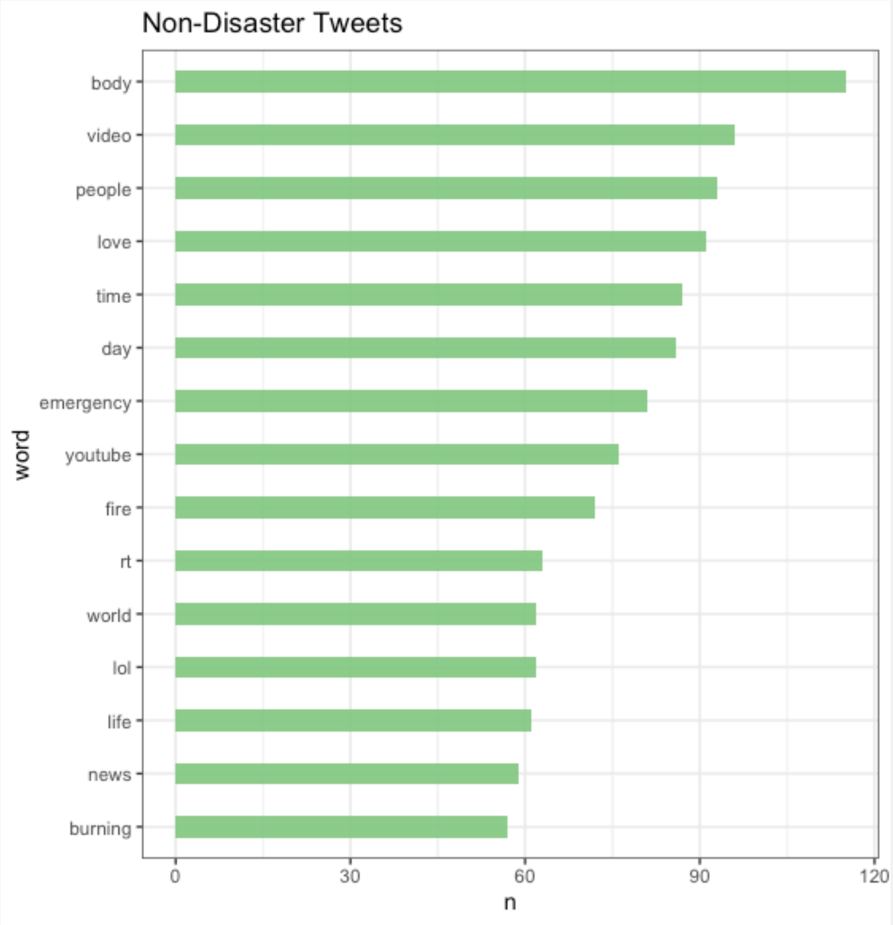
THE DATA

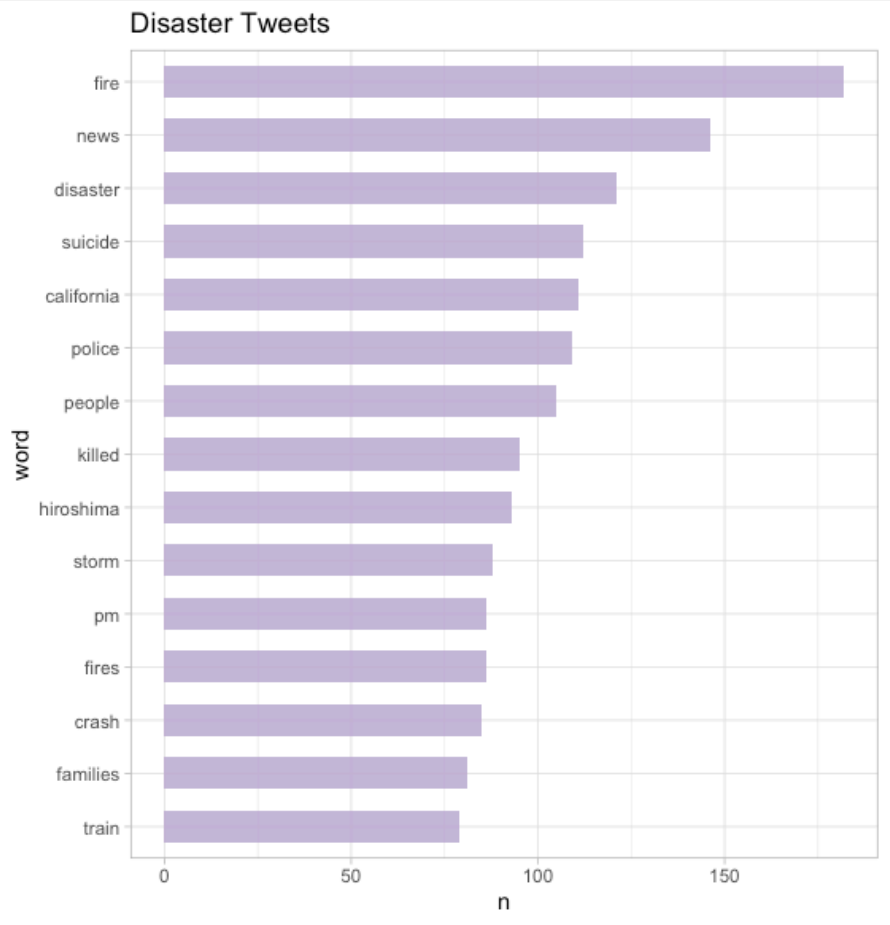
target distribution in the training set



THE DATA

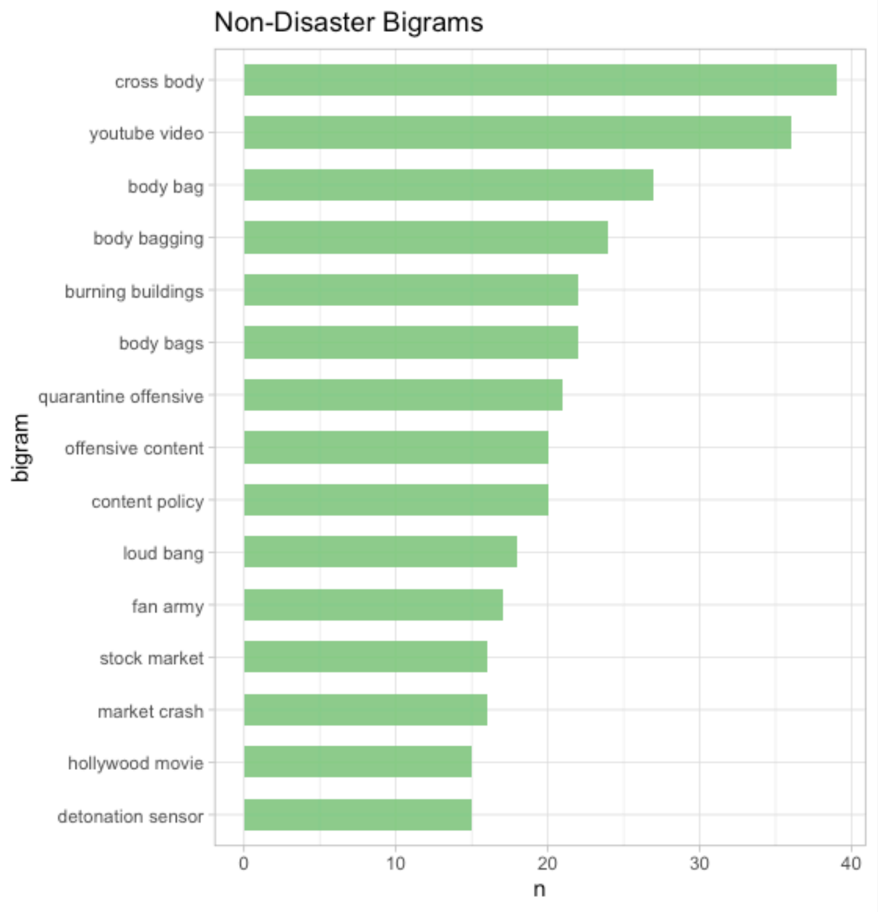
most common **words** in the training set

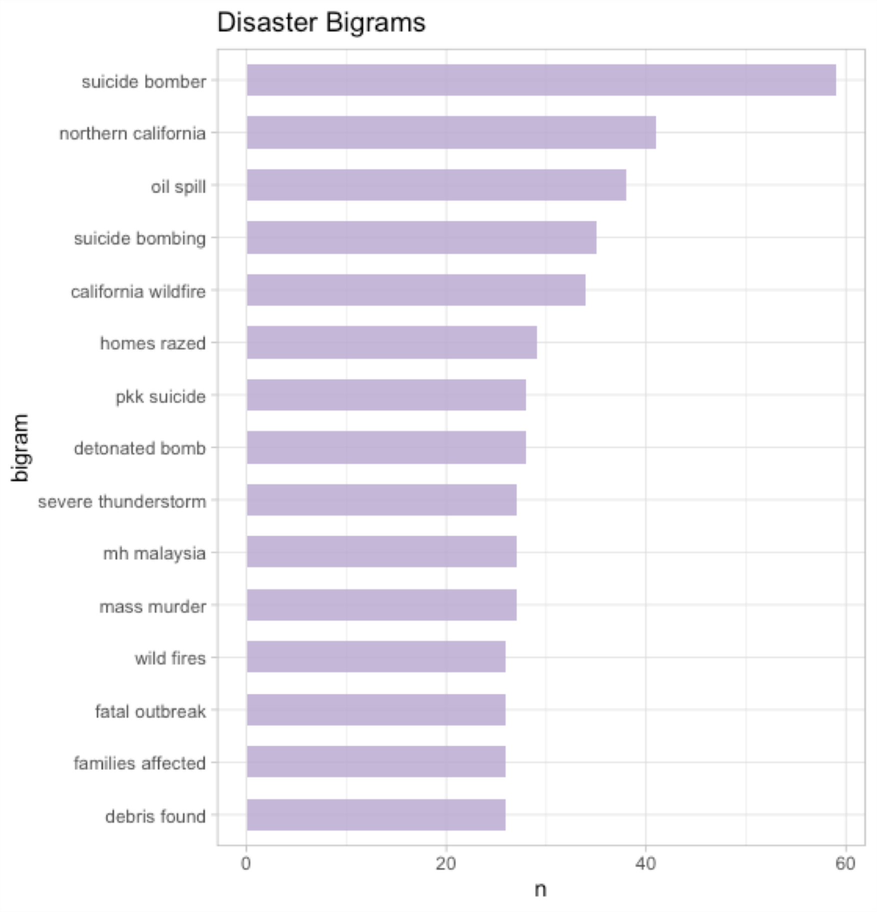




THE DATA

most common **bigrams** in the training set





TEXT CLEANING PROCESS

- expand contractions
- remove punctuation
- lowercase
- remove urls and digits
- remove stopwords
- reduced dataset to text and target

THE MODEL

environment:

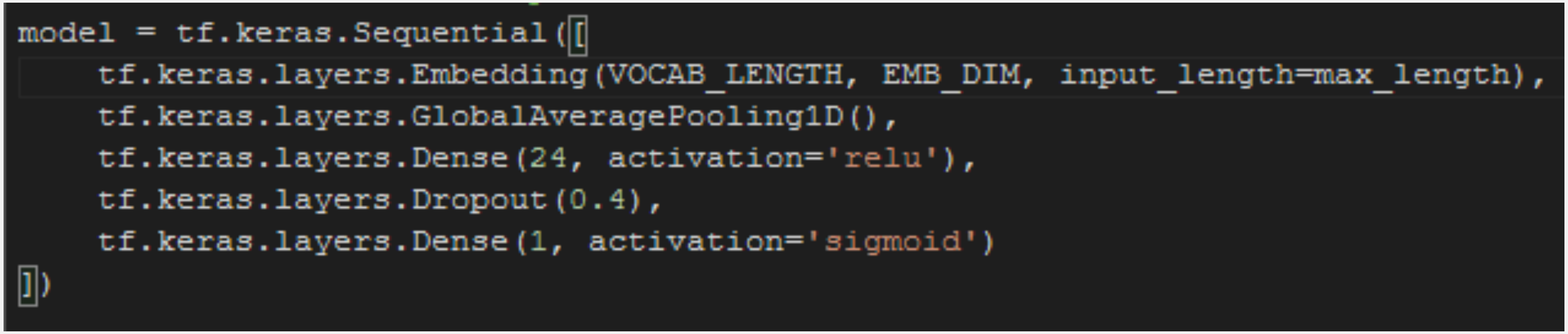
- Tensorflow in Google Colabs

data preparation steps:

- tokenization with BERT Full Tokenizer
- split into randomized training and test set 80/20
- padded tweets to equal length

THE MODEL

model architecture:



```
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(VOCAB_LENGTH, EMB_DIM, input_length=max_length),
    tf.keras.layers.GlobalAveragePooling1D(),
    tf.keras.layers.Dense(24, activation='relu'),
    tf.keras.layers.Dropout(0.4),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```

hyperparameters:

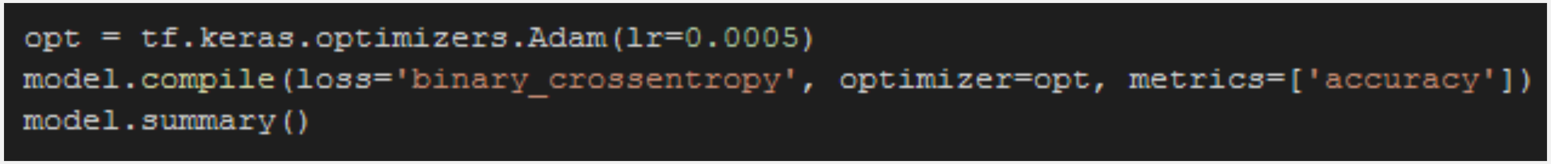
VOCAB_LENGTH = len(tokenizer.vocab) ~30,000

EMB_DIM = 16

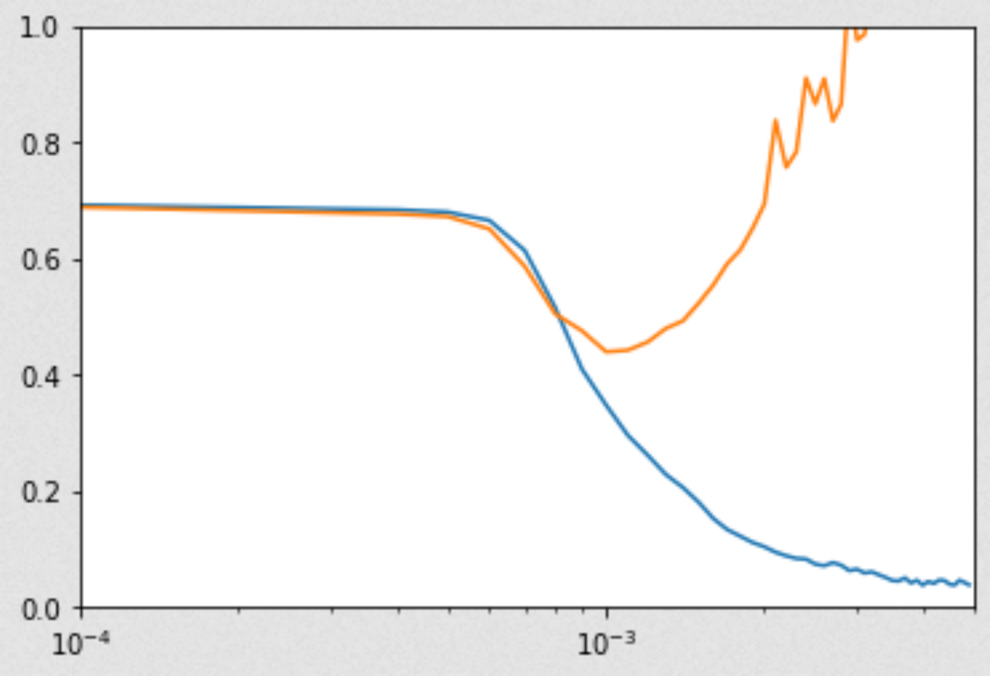
max_length = 128

THE MODEL

learning rate:

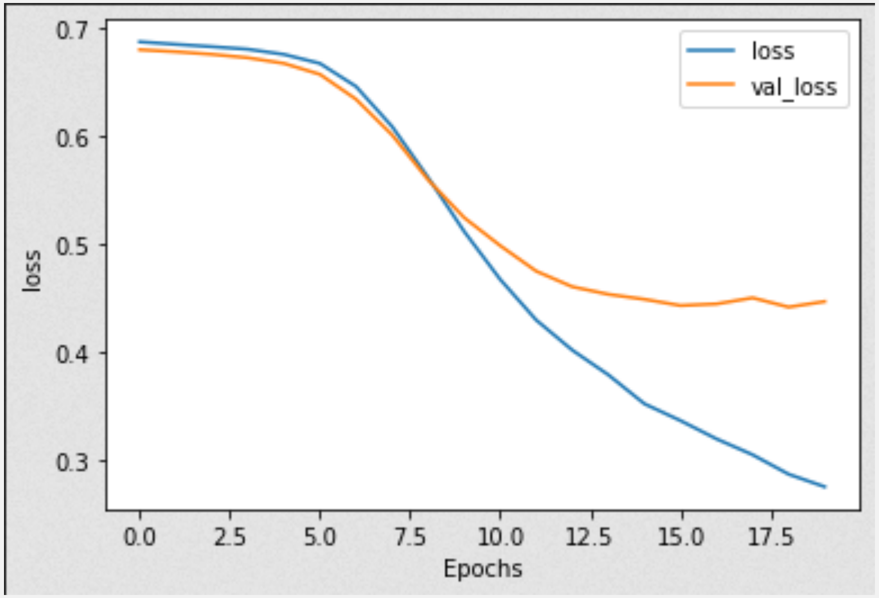


```
opt = tf.keras.optimizers.Adam(lr=0.0005)
model.compile(loss='binary_crossentropy', optimizer=opt, metrics=['accuracy'])
model.summary()
```

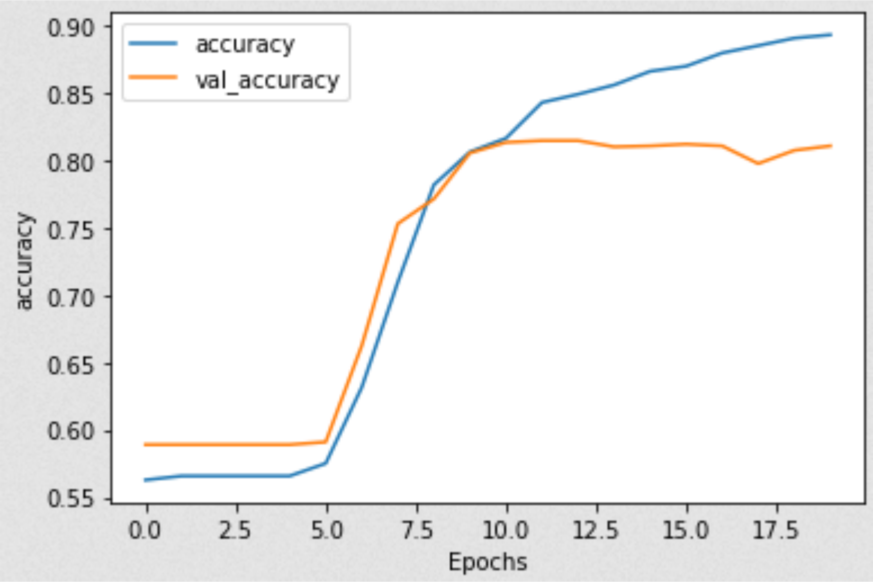


THE MODEL

loss



accuracy



after 15 epochs:

loss: 0.35
val_loss: 0.45

acc: 0.87
acc_val: 0.81

DIFFICULTIES

- hard to get an overview over text data and text cleaning
  - ➢ preformatted list of stopwords

- heavy overfitting in the beginning
  - ➢ dropout layer

- no improvement of val_acc (~80%)
  - even with LSTM, CNN