Wie funktioniert unser Programm

Jonathan Gärtner Antonios Schinarakis

- Importieren benötigter Module (Code der uns viel Arbeit abnimmt)
- Laden des von uns trainierten KNN (künstliches neuronales Netz. Später mehr)

```
In [1]: from tensorflow.keras.models import load_model#Importing needed Modules
        import pickle
        import nltk
        import numpy as np

        model = load_model("punctation_lib/punctator")
        model.load_weights("punctation_lib/punctator.h5")

        model.summary()
```

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_input (InputLayer)      [(None, 40)]              0
_________________________________________________________________
lstm_embedding (Embedding)   (None, 40, 128)           7040
_________________________________________________________________
lstm (LSTM)                  (None, 128)               131584
_________________________________________________________________
lstm_predictions (Dense)     (None, 40)                5160
=================================================================
Total params: 143,784
Trainable params: 143,784
Non-trainable params: 0
_________________________________________________________________
```

↑ So sieht der Aufbau unseres KNN aus.
Machen wir uns mal einen Beispiel Satz:
Diesen Tokenisieren wir dann, dass heißt wir zerlegen ihn in Wortstücke

```
In [2]: satz =
 "Das ist ein Beispiel Satz hier kommt noch ein Satz schauen wir mal,ob die" +
"Punkte erkannt werden"
```

```
        satz = nltk.word_tokenize(satz, "german")
        print("So sieht der Satz tokenisiert aus: " + str(satz))
```

```
So sieht der Satz tokenisiert aus: ['Das', 'ist', 'ein', 'Beispiel', '
    Satz', 'hier', 'kommt', 'noch', 'ein', 'Satz', 'schauen', 'wir', '
    mal', 'ob', 'die', 'Punkte', 'erkannt', 'werden']
```

Jetzt kommt ein sehr wichtiger Punkt wir wandeln den Satz erst in seine Satzbausteine (Adjektiv, Verb, ...) um. Das nennt man taggen. Dann wandeln wir die Ausgabe in Zahlen um damit unser Modell damit rechnen kann

```
In [29]: ausgabe = satz#Für später um die Punkte einzusetzen
         with open('ClassifierBasedGermanTagger/germanTagger.pickle', 'rb') as f:
         #Das Modul zum "taggen"
                 tagger = pickle.load(f)
         satz = tagger.tag(satz)
         print("Die Satzbausteine: " + str(satz))
         tag_set = ['PPER', 'APPRART', 'PWS', 'NE', 'PRELS', 'KOKOM', 'PIAT', 'CARD',
                 'VMINF', 'PIS', 'XY', 'PTKANT','PTKNEG', 'APPR', 'ADV', 'KON', 'VMFIN',
                 'APZR', 'ADJD', 'PDS', 'VVFIN', 'PRF', 'VAINF', 'ADJA', '$.','TRUNC', 'VVPP',
                 'PDAT','ART', 'NN', 'PPOSAT', 'VVINF', '$(', 'VAPP', '$,', 'PWAV',
                 'KOUS', 'KOUI','FM', 'VVIZU', 'VVIMP', 'VAFIN', 'PTKZU', 'PTKVZ',
                 'PROAV', 'VAIMP', 'NNE', 'PWAT', 'APPO', 'ITJ',
                    'PRELAT', 'VMPP', 'PPOSS', 'PTKA', 'NULL']
#Das sind alle Textbausteine die Abkürzungen findet man auf der Website der Uni Stuttgart
         tag_sen = []
         for sen_tag in satz:
             if sen_tag[1] in tag_set:
                 tag_sen.append(tag_set.index(sen_tag[1]))#Umwandlung in Nummern
         print()#Leerzeile für Formatierung
         print("Die resultierenden Nummern: " + str(tag_sen))
```

```
Die Satzbausteine: [('Das', 'ART'), ('ist', 'VAFIN'), ('ein', 'ART'),
    ('Beispiel', 'NN'), ('Satz', 'NN'), ('hier', 'ADV'), ('kommt', '
    VVFIN'), ('noch', 'ADV'), ('ein', 'ART'), ('Satz', 'NN'), ('schauen
    ', 'VVINF'), ('wir', 'PPER'), ('mal', 'ADV'), ('ob', 'KOUS'), ('die
    ', 'ART'), ('Punkte', 'NN'), ('erkannt', 'VVPP'), ('werden', 'VAINF
    ')]

Die resultierenden Nummern: [28, 41, 28, 29, 29, 14, 20, 14, 28, 29,
    31, 0, 14, 36, 28, 29, 26, 22]
```

Da unser Satz weniger als 40 Wörter hat müssen wir diesen array nur noch so ergänzen das er 40 Elemente hat:(Wer wissen will wie es mit mehr als 40 Wörtern funktioniert schaut sich bitte unsere github Seite an)

```
In [30]: while len(tag_sen) % 40 != 0:  # Bis es keinen Rest beim durch 40 teilen gibt
             tag_sen.append(54) #Equavavilent zu dem NULL tag
```

```
        print("Länge des Arrays: " + str(len(tag_sen)))
        print()
        print("Der ganze Array: " + str(tag_sen))
```

Lange des Arrays: 40

Der ganze Array: [28, 41, 28, 29, 29, 14, 20, 14, 28, 29, 31, 0, 14, 36, 28, 29, 26, 22, 54, 54, 54, 54, 54, 54, 54, 54, 54, 54, 54, 54, 54, 54, 54, 54, 54, 54, 54, 54, 54, 54, 54, 54, 54, 54, 54]

Das können wir jetzt einfach in unser KNN eingeben da nimmt uns tensorflow die ganze Arbeit ab. Wir müssen nur den Satz (tag_sen) zweimal nehemen, weil unser KNN das vom Training "gewöhnt" ist. Um zu verstehen wie unser KNN aufgebaut ist lesen Sie bitte das entsprechende Plakat

```
In [31]: model_prepared = np.array([tag_sen, tag_sen])
         vec = model.predict(model_prepared)[0]#In vec ist die Ausgabe des KNN gespeichert
         print("Die Ausgabe des KNN: " + str(vec))
```

```
Die Ausgabe des KNN: [0.15349647 0.19199911 0.23071742 0.18247968
 0.18456912 0.8439578
 0.1459465  0.10394245 0.10377932 0.12534562 0.757808   0.1857554
 0.26473033 0.3548991  0.25904894 0.35849626 0.40727315 0.4375044
 0.407189   0.33716125 0.44110548 0.23494288 0.1601592  0.24967259
 0.3936718  0.3363497  0.3676647  0.3098882  0.3000732  0.4100883
 0.32170045 0.41351205 0.32559984 0.3054048  0.5426147  0.3909998
 0.4238664  0.2235547  0.22792968 0.01355103]
```

Jetzt werden wir überall wo mehr als 50% an Wahrscheinlichkeit sind einen Punkt setzten

```
In [32]: for iter in range(0, len(vec)):
             if vec[iter] >= 0.5 and tag_sen[iter] != 54:
                 ausgabe[iter - 1] = ausgabe[iter- 1] + "."
                 ausgabe[iter] = ausgabe[iter][0].upper() + ausgabe[iter][1:]
         ausgabe[-1] = ausgabe[-1] + "."
         print("So sieht es als array aus: ", str(ausgabe))
         ausgabe = ' '.join(ausgabe)
         print()
         print("Und so als normaler Satz: ", str(ausgabe))
```

```
So sieht es als array aus:  ['Das', 'ist', 'ein', 'Beispiel', 'Satz.',
 'Hier', 'kommt', 'noch', 'ein', 'Satz.', 'Schauen', 'wir', 'mal', '
 ob', 'die', 'Punkte', 'erkannt', 'werden.']

Und so als normaler Satz:  Das ist ein Beispiel Satz. Hier kommt noch
 ein Satz. Schauen wir mal ob die Punkte erkannt werden.
```

Wenn Sie jetzt aufgepasst haben ist ihnen sicher aufgefallen das wir die Satzanfänge und nicht die Satzenden aus dem KNN bekommen haben. Tatsächlich liefen die KNNs die Satzanfänge gefunden haben einfach besser, als die anderen. Das heißt für unser LSTM kann besser Satzanfänge, als Satzenden finden