Wordembedding-Lyrik-Visualisierung v2

November 18, 2020

0.1 Wordembedding

[1]:
```
import json
import random
import statistics
from pathlib import Path
from collections import defaultdict
import os
import glob

import syntok.segmenter as segmenter
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from flair.data import Sentence
from flair.embeddings import TransformerWordEmbeddings
from scipy import spatial
from scipy.ndimage.filters import gaussian_filter1d

import ipywidgets
from ipywidgets import IntProgress

import matplotlib.pyplot as plt
```

0.1.1 alle Sätze auswählen die Emotionen beinhalten

[2]:
```
emotion = ['liebreich','angst', 'ängstlich', 'trauer', 'traurig', 'zornig',␣
 ↪'zorn', 'verachtungsvoll', 'verachtung',
           'schuld', 'schuldig', 'liebe',  'geliebt', 'liebevoll', 'stolz',␣
 ↪'scham', 'schämen', 'überrasschung',
           'überrascht', 'sorge', 'sorgenvoll', 'ekel', 'ekeln', 'angeekelt',␣
 ↪'neid', 'neidisch', 'neidvoll',
           'glücklich', 'glück', 'freude', 'freudig', 'freuen', 'erleichterung',␣
 ↪'erleichtert', 'vergnügt', 'vergnügen',
           'zufrieden', 'zufriedenheit', 'verzweiflung', 'verzweifelt',␣
 ↪'verlegenheit', 'verlegen', 'aufregung',
           'aufgeregt', 'aufregen', 'spannung', 'gespannt', 'erregung',␣
 ↪'erregt', 'hoffen', 'hoffnung', 'befriedigt',
           'langweilig', 'langeweile', 'mitgefühl', 'mitfühlen', 'enttäuscht',␣
 ↪'enttäuschung', 'frust', 'frustriert',
           'eifersucht', 'eifersüchtig', 'wut', 'wütend', 'reue', 'schock',␣
 ↪'schockiert', 'zuneigung', 'verärgert',
           'verärgerung', 'erwartungsvoll', 'erwartung', 'vorfreude', 'scheu',␣
 ↪'gelassen', 'gelassenheit', 'mut',
           'mutig', 'neugierde', 'neugierig', 'depression', 'depressiv',␣
 ↪'niedergeschlagenheit', 'niedergeschlagen',
           'lustvoll', 'lust', 'rausch', 'einfühlend', 'einfühlsam',␣
 ↪'euphorisch', 'euphorie', 'dankbarkeit', 'dankbar',
           'hass', 'entsetzt', 'entsetzen', 'demütigung', 'demütig', 'demut',␣
 ↪'interesse', 'interessiert', 'einsamkeit',
           'einsam', 'empörung', 'empört', 'vertrauen', 'qualvoll', 'qual',␣
 ↪'gleichgültigkeit', 'gleichgültig',
           'fröhlichkeit', 'fröhlich', 'schadenfroh', 'schadenfreude',␣
 ↪'schmerz', 'melancholie', 'melancholisch',
           'panik', 'panisch']

sentences = []

inpath = '/home/piah/Dokumente/Uni/Projektarbeit/
 ↪Projektarbeit_LyrikGattungszuweisung/corpus/corpus/gesamt/'

for text in os.listdir(inpath):
    if text.endswith('.txt'):
        f_lemma = []
        result = ''
        t = open(inpath + '/' + text, 'r')
        f = t.read()


for files in f:
    for paragraph in segmenter.process(f):
        for sentence in paragraph:
            tokens = [str(token).strip() for token in sentence]
            for emo in emotion:
                if emo in tokens:
                    index = tokens.index(emo)
                    sentences.append({"sentence": tokens, "index": index,␣
 ↪"source":file, "emotion": emotion})
print(sentence)
```

```
[<Token '\n' : 'In' @ 1757>, <Token ' ' : 'des' @ 1760>, <Token ' ' : 'Kusses' @
1764>, <Token ' ' : 'wildlodernder' @ 1771>, <Token '\n' : 'Flamme' @ 1785>,
<Token ' ' : 'vermählt' @ 1792>, <Token ' ' : 'sich' @ 1801>, <Token '\n' :
```

```
'Alle' @ 1806>, <Token ' ' : 'Süße' @ 1811>, <Token ' ' : 'des' @ 1816>, <Token
' ' : 'Lebens' @ 1820>, <Token '\n' : 'Des' @ 1827>, <Token ' ' : 'Lebens' @
1831>, <Token ' ' : 'und' @ 1838>, <Token ' ' : 'Todes' @ 1842>, <Token '' : '.'
@ 1847>]
```

0.2 Wortvektoren generieren

[3]:

```python
embedding = TransformerWordEmbeddings("redewiedergabe/
 ↪bert-base-historical-german-rw-cased")
```

[5]:

```python
for example in sentences:
    text = " ".join(example["sentence"])
    sentence = Sentence(text, use_tokenizer=False)
    embedding.embed(sentence)

    token = sentence[example["index"]]
    example["vector"] = [float(dim) for dim in token.embedding]
```

0.3 Positive und negative Wortumgebung

[7]:

```python
words = {}

with open("/home/piah/Dokumente/Uni/Projektarbeit/
 ↪Projektarbeit_LyrikGattungszuweisung/scripts/wordembedding/Wortlisten/Positiv.
 ↪txt", "r", encoding="utf-8") as f:
    words["positiv"] = random.sample([f"Die Emotion ist {word}" for word in f.
 ↪read().split("\n")], 500)

with open("/home/piah/Dokumente/Uni/Projektarbeit/
 ↪Projektarbeit_LyrikGattungszuweisung/scripts/wordembedding/Wortlisten/Negativ.
 ↪txt", "r", encoding="utf-8") as f:
    words["negativ"] = random.sample([f"Die Emotion ist {word}" for word in f.
 ↪read().split("\n")], 500)

print(f"Positive Worte: {len(words['positiv'])}")
print(f"Negative Worte: {len(words['negativ'])}")
```

```
Positive Worte: 500
Negative Worte: 500
```

[8]:

```python
positive = []
negative = []

for word in words["positiv"]:
    sentence = Sentence(word, use_tokenizer=False)
    embedding.embed(sentence)
    token = sentence[2]
    positive.append([float(dim) for dim in token.embedding])

for word in words["negativ"]:
    sentence = Sentence(word, use_tokenizer=False)
    embedding.embed(sentence)

    token = sentence[2]
    negative.append([float(dim) for dim in token.embedding])
```

0.4 Ähnlichkeiten

[9]:

```
for sentence in sentences:
    positive_scores = []
    negative_scores = []

    for vector in positive:
        positive_scores.append(1 - spatial.distance.cosine(sentence["vector"],␣
 ↪vector))
    for vector in negative:
        negative_scores.append(1 - spatial.distance.cosine(sentence["vector"],␣
 ↪vector))

    sentence["positive_mean"] = statistics.mean(positive_scores)
    sentence["negative_mean"] = statistics.mean(negative_scores)
    sentence["absolute_difference"] = abs(sentence["positive_mean"] -␣
 ↪sentence["negative_mean"])
```

1 Visualisierung

[5]:

```
from pathlib import Path
import pandas as pd
import numpy as np
import tqdm
import seaborn as sns
from flair.embeddings import TransformerDocumentEmbeddings
from flair.data import Sentence
from sklearn.cluster import KMeans
from sklearn.manifold import TSNE
from sklearn.decomposition import PCA
from sklearn.metrics.cluster import adjusted_rand_score
```

1.0.1 Read corpora

[2]:

```
data = [{"class": "Ballade", "text": file.read_text()} for file in␣
 ↪Path("Balladen").glob("*.txt")]
data.extend([{"class": "Lied", "text": file.read_text()} for file in␣
 ↪Path("Lyrik").glob("*.txt")])
```

1.0.2 Load document embedding

[3]:

```
embedding = TransformerDocumentEmbeddings("redewiedergabe/
 ↪bert-base-historical-german-rw-cased")
```

1.0.3 Get document embeddings

[6]:

```
vectors = []
labels = []

for document in tqdm.tqdm(data):
    sentence = Sentence(document["text"])
    embedding.embed(sentence)
    vectors.append(sentence.embedding.tolist())
    labels.append(document["class"])

vectors = np.array(vectors)
```

```
100%|| 224/224 [07:17<00:00,  1.95s/it]
```

1.0.4 Visualize with PCA

[7]:

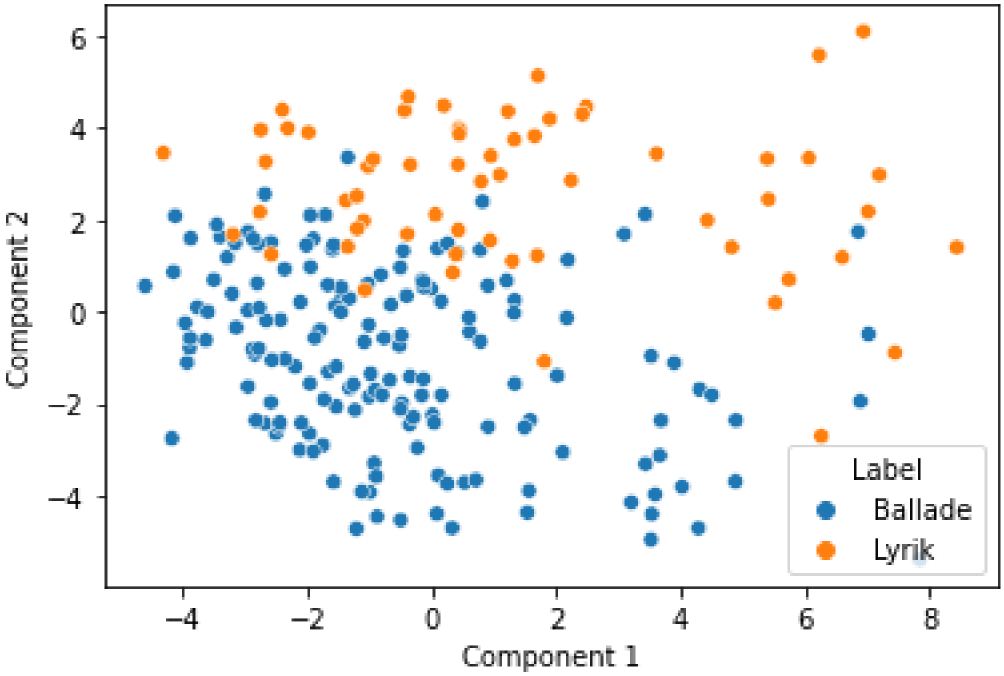
```
pca = PCA(n_components=2)
components = pca.fit_transform(vectors)
```

[18]:

```
df = pd.DataFrame(components)
df["Label"] = labels
df.columns = ["Component 1", "Component 2", "Label"]
sns.scatterplot(x="Component 1", y="Component 2", data=df, hue="Label")
```

[18]:

```
<AxesSubplot:xlabel='Component 1', ylabel='Component 2'>
```

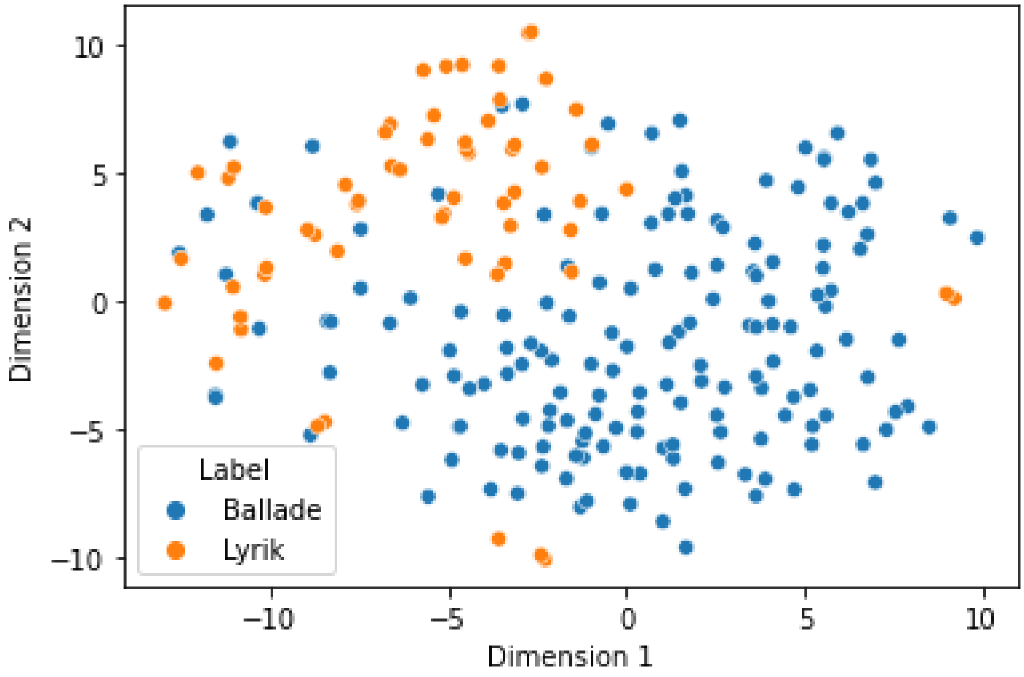


1.0.5 Visualize with t-SNE

[9]:
```
tsne = TSNE(n_components=2, random_state=0)
embedded = tsne.fit_transform(vectors)
```

[19]:
```
df = pd.DataFrame(embedded)
df["Label"] = labels
df.columns = ["Dimension 1", "Dimension 2", "Label"]
sns.scatterplot(x="Dimension 1", y="Dimension 2", data=df, hue="Label")
```

[19]:
```
<AxesSubplot:xlabel='Dimension 1', ylabel='Dimension 2'>
```



1.0.6 k-Means clustering

Original embeddings

```
[11]: kmeans = KMeans(n_clusters=2, random_state=0)
      kmeans.fit(vectors)
```

```
[11]: KMeans(n_clusters=2, random_state=0)
```

```
[12]: adjusted_rand_score(labels, kmeans.labels_)
```

```
[12]: 0.10136794763794267
```

PCA-reduced embeddings

```
[13]: kmeans = KMeans(n_clusters=2, random_state=0)
      kmeans.fit(components)
```

```
[13]: KMeans(n_clusters=2, random_state=0)
```

```
[14]: adjusted_rand_score(labels, kmeans.labels_)
```

```
[14]: 0.07788750992103581
```

t-SNE-reduced embeddings

[15]:

```
kmeans = KMeans(n_clusters=2, random_state=0)
kmeans.fit(embedded)
```

[15]:

```
KMeans(n_clusters=2, random_state=0)
```

[16]:

```
adjusted_rand_score(labels, kmeans.labels_)
```

[16]:

```
0.41730054148976975
```