Explaining Drift: Embeddings and Geometry

In [1]:
```
# Reload modules every time before executing the Python code typed
%load_ext autoreload
%autoreload 2

# Import from parent directory
import sys; sys.path.insert(0, '..')

# Configure data storage
from yaml import safe_load
import classes.io
io = classes.io.Io(safe_load(open('../config.yaml', 'r'))['DATA_DIRECTORY'])

# Additional imports
import classes.reduction

import matplotlib.pyplot as plt
import seaborn as sns

from classes.geometry import Geometry
from polylidar.polylidarutil import plot_polygons

from gensim.utils import simple_preprocess
from wordcloud import WordCloud, STOPWORDS
from collections import Counter
```

In [2]:
```
#plt.rcParams["figure.figsize"] = (12,8)
plt.rcParams["figure.figsize"] = (6,4)
```

Source data

- **Amazon Movie Reviews**: **10,000 1-star** and **10,000 5-star** rated texts

In [3]:
```
# load texts
dataset_id = 'amazon-movie-reviews-10000'
texts = io.load_data_pair(dataset_id, io.DATATYPE_TEXT)
```

```
Loaded /home/eml4u/EML4U/data/explanation/data/amazon-movie-reviews-10000/text.pickle
```

Embeddings

- **Doc2Vec embeddings**, 50 dimensions, trained for 50 epochs
- Additional training/optimization possible
- BERT also available

In [4]:
```
# load embeddings
datatype_id = 'doc2vec.dim50-epochs50'
embeddings = io.load_data_pair(dataset_id, io.DATATYPE_EMBEDDINGS, io.DESCRIPTOR_DOC_TO_VEC, 'dim50-epochs50')
```

```
Loaded /home/eml4u/EML4U/data/explanation/data/amazon-movie-reviews-10000/doc2vec.dim50-epochs50.embeddings.pickle
```

Dimension reduction

- Here: **PCA** (faster)
- t-SNE code also available (maybe better)

In [5]:
```
dimension_reduction = classes.reduction.Reduction()
#embeddings_a, embeddings_b = dimension_reduction.pca_dict(embeddings.get_a(), embeddings.get_b())
pca_a, pca_b = dimension_reduction.pca(list(embeddings.get_a().values()), list(embeddings.get_b().values()))
```

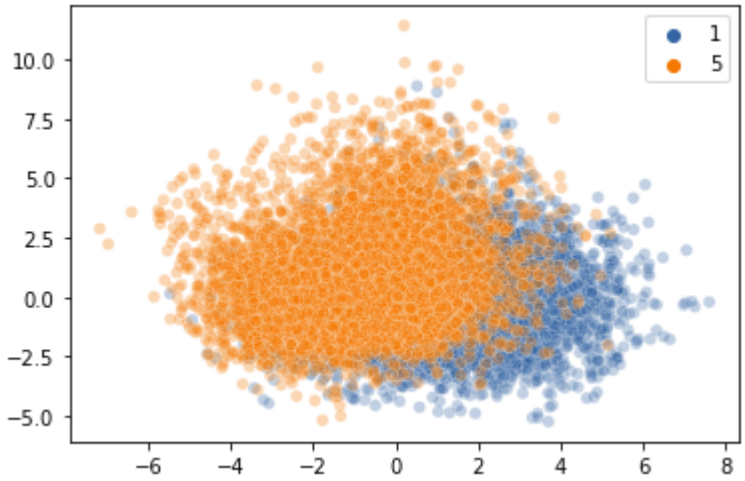
```
PCA seconds: 0.13505211472511292
```

Plot

- Getting an overview

In [6]:
```
def plot(data_a, data_b):
    plot_data = data_a
    sns.scatterplot(data=plot_data, x=plot_data[:,0], y=plot_data[:,1], palette=['#3465A4'], hue=1, alpha = 0.3)
    plot_data = data_b
    sns.scatterplot(data=plot_data, x=plot_data[:,0], y=plot_data[:,1], palette=['#F57900'], hue=5, alpha = 0.3)
```

In [7]:
```
#plot(list(embeddings_a.values()), list(embeddings_b.values()))
plot(pca_a, pca_b)
```



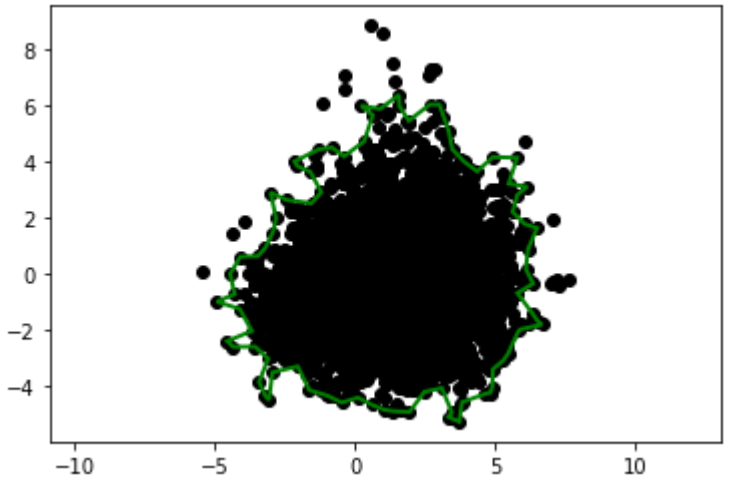
Extract polygons

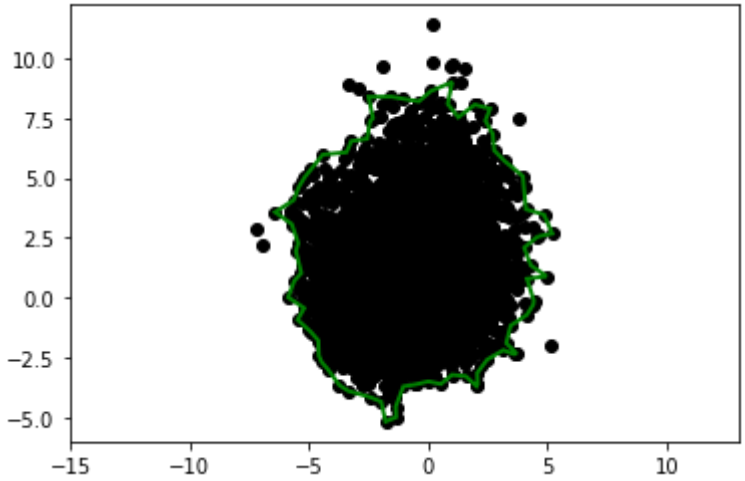
- Shape not exact, can easily be configured (good)

In [8]:
```
def plot_points_polygons(points, polygons):
    fig, ax = plt.subplots(nrows=1, ncols=1)
    ax.scatter(points[:, 0], points[:, 1], c='k')
    plot_polygons(polygons, points, ax)
    plt.axis('equal')
    plt.show()
```

In [9]:
```
geometry = Geometry()
plot_points_polygons(pca_a, geometry.extract_polygons(pca_a))
plot_points_polygons(pca_b, geometry.extract_polygons(pca_b))
```

```
/home/eml4u/.local/lib/python3.8/site-packages/descartes/patch.py:62: ShapelyDeprecationWarning: The array interface is deprecate
d and will no longer work in Shapely 2.0. Convert the '.coords' to a numpy array instead.
  vertices = concatenate([
```





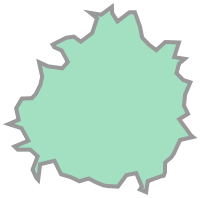
Unique polygon parts

- Non-overlapping embeddings

In [10]:
```
polygon_indexes_a = geometry.extract_polygon_indexes(pca_a)
polygon_indexes_b = geometry.extract_polygon_indexes(pca_b)

polygon_a = geometry.create_polygon(pca_a, polygon_indexes_a[0])
polygon_b = geometry.create_polygon(pca_b, polygon_indexes_b[0])
polygon_a_not_b = polygon_a - polygon_b
polygon_b_not_a = polygon_b - polygon_a
```

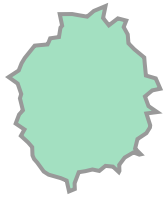
In [11]:
```
polygon_a
```

Out[11]:



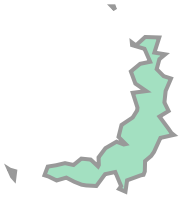
In [12]:
```
polygon_b
```

Out[12]:



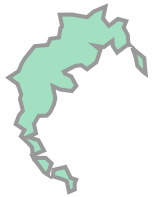
In [13]:
```
polygon_a_not_b
```

Out[13]:



In [14]:
```
polygon_b_not_a
```

Out[14]:



Get points inside unique polygons

- Get reviews with non-overlapping embeddings

In [15]:
```
indexes_only_a = geometry.get_indexes_of_points_in_polygon(pca_a, list(embeddings.get_a().keys()), polygon_a_not_b)
indexes_only_b = geometry.get_indexes_of_points_in_polygon(pca_b, list(embeddings.get_b().keys()), polygon_b_not_a)
```

Get words (tokens)

- Use available keys to get texts related to embeddings
- Min length: 2 chars, max 15 chars
- Remove stopwords
- Count words

In [16]:
```
tokens_a = []
for index in indexes_only_a:
    tokens_a += simple_preprocess(texts.get_a()[index], deacc=False, min_len=2, max_len=15)
tokens_b = []
for index in indexes_only_b:
    tokens_b += simple_preprocess(texts.get_b()[index], deacc=False, min_len=2, max_len=15)

stopwords = set(STOPWORDS)
stopwords.add('br')
tokens_a = [w for w in tokens_a if w not in stopwords]
tokens_b = [w for w in tokens_b if w not in stopwords]

counts_a = Counter(tokens_a)
counts_b = Counter(tokens_b)
```

Wordcloud

- Visualize results (color has no meaning)
- Bad: e.g. 'movie' in both clouds

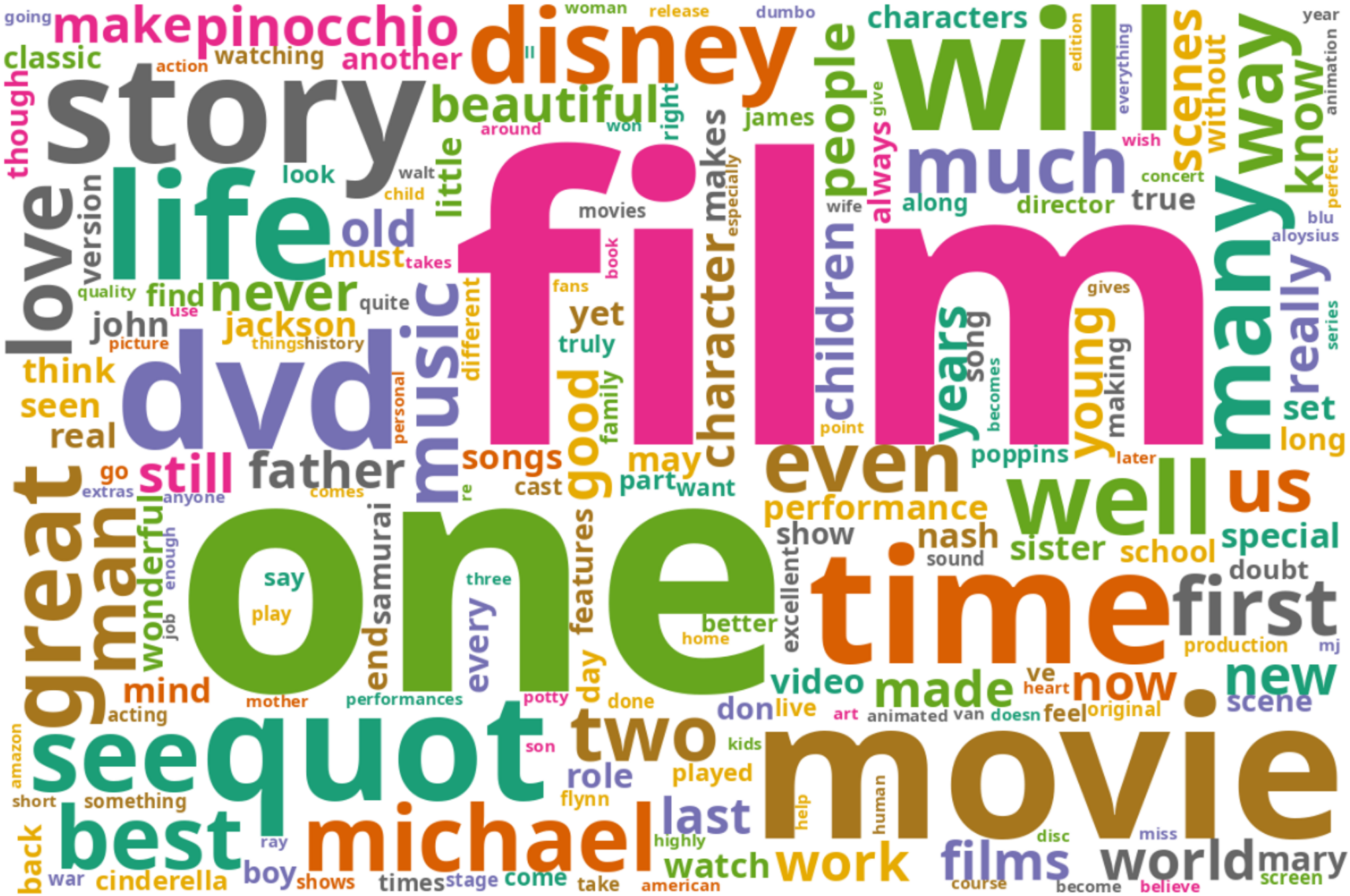
In [17]:
```
font_path='/usr/share/fonts/truetype/noto/NotoSans-Bold.ttf' #  fc-list | grep 'NotoSans-Bold'
wordcloud_a_counts = WordCloud(background_color="white", font_path=font_path, colormap='Dark2', width=1200, height=800).generate_
wordcloud_b_counts = WordCloud(background_color="white", font_path=font_path, colormap='Dark2', width=1200, height=800).generate_
```

In [18]:
```
plt.imshow(wordcloud_a_counts)
plt.axis("off")
#plt.savefig('2021-10-27-wordcloud_a_counts.png', dpi=300, bbox_inches='tight')
```

Out[18]:
```
(-0.5, 1199.5, 799.5, -0.5)
```

In [19]:
```
plt.imshow(wordcloud_b_counts)
plt.axis("off")
#plt.savefig('2021-10-27-wordcloud_b_counts.png', dpi=300, bbox_inches='tight')
```

Out[19]: (-0.5, 1199.5, 799.5, -0.5)



Wordcloud with unique words

- Only words which appear in only 1-star (or 5-star) documents

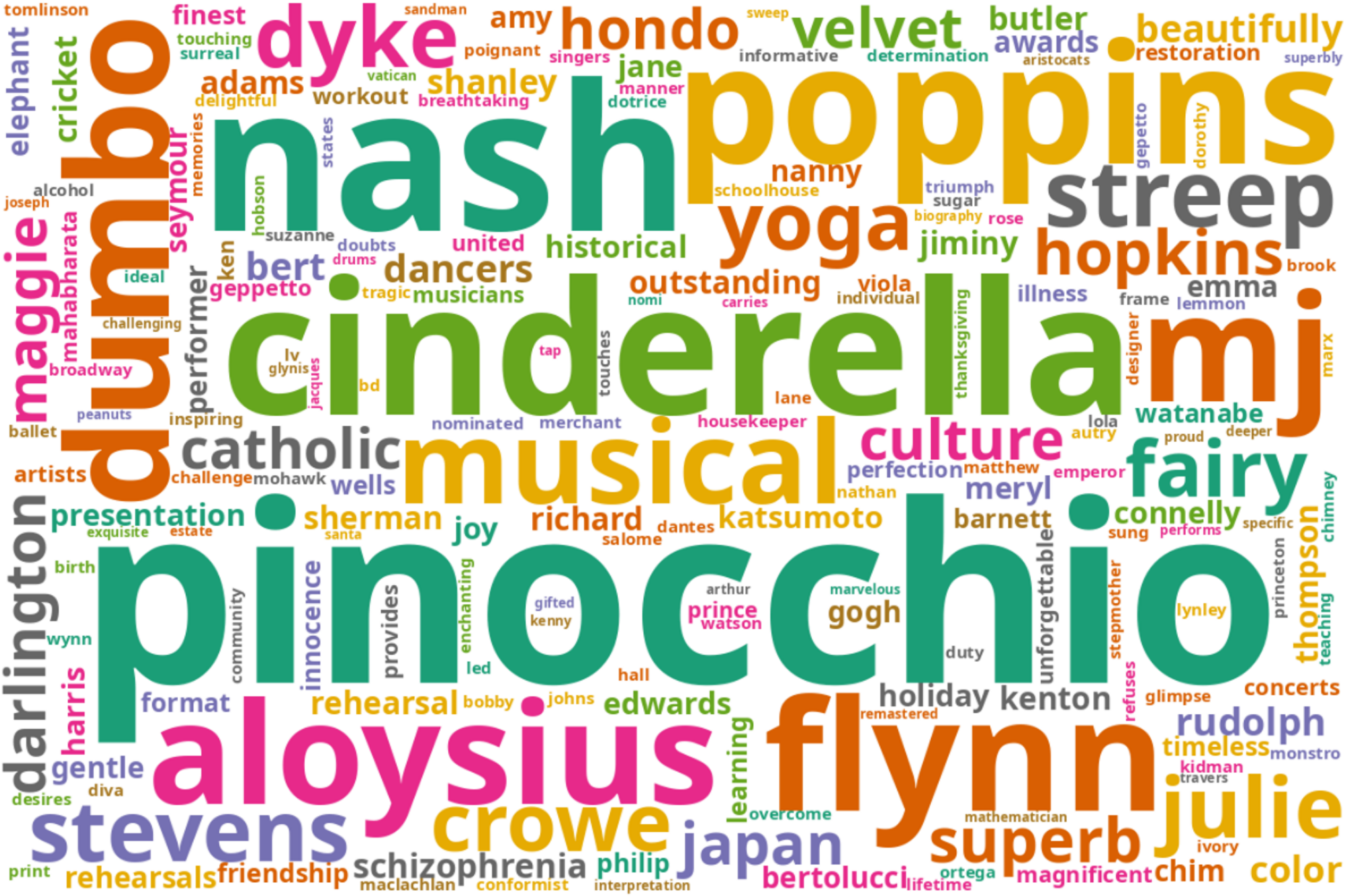
In [20]:
```
tokens_a_set = set(tokens_a)
tokens_b_set = set(tokens_b)
tokens_a2 = [x for x in tokens_a if x not in tokens_b_set]
tokens_b2 = [x for x in tokens_b if x not in tokens_a_set]

counts_a = Counter(tokens_a2)
counts_b = Counter(tokens_b2)

wordcloud_a_counts = WordCloud(background_color="white", font_path=font_path, colormap='Dark2', width=1200, height=800).generate_
wordcloud_b_counts = WordCloud(background_color="white", font_path=font_path, colormap='Dark2', width=1200, height=800).generate_
```

In [21]:
```
plt.imshow(wordcloud_a_counts)
plt.axis("off")
#plt.savefig('2021-10-27-wordcloud_a_counts2.png', dpi=300, bbox_inches='tight')
```

Out[21]:
```
(-0.5, 1199.5, 799.5, -0.5)
```

In [22]:
```
plt.imshow(wordcloud_b_counts)
plt.axis("off")
#plt.savefig('2021-10-27-wordcloud_b_counts2.png', dpi=300, bbox_inches='tight')
```

Out[22]: (-0.5, 1199.5, 799.5, -0.5)



Credits: Data Science Group (DICE) at Paderborn University, Adrian Wilke, 2021-20-27 This work has been supported by the German Federal Ministry of

Education and Research (BMBF) within the project EML4U under the grant no 01IS19080B.