dataset_info

August 20, 2024

1 Se på ord i data

[ ]:
```
from ordbilder.annotations import get_annotation_information
from clean_text_data import clean
from pathlib import Path

train_path = Path("../data/transkribus_exports/train_data/train")
s_30_path = Path("../data/transkribus_exports/train_data/side_30")
test_path = Path("../data/transkribus_exports/test_data/2997983/
 ↪Testsett_Samisk_OCR")


# Lag txt-mappe hvis ikke finnes
for path in (train_path, s_30_path, test_path):
    was_unclean = 0
    num_pages = 0
    for alto_dir in path.glob("**/alto"):
        files = list(alto_dir.glob("*.xml"))
        text_dir = alto_dir.parent / "txt"
        text_dir.mkdir(exist_ok=True)
        for xml_file in files:
            num_pages += 1
            page_file = text_dir / f"{xml_file.stem}.txt"
            annotations = get_annotation_information(xml_path=xml_file)
            page_text_pre = "\n".join([e["word"] for e in annotations])
            page_text = clean(page_text_pre)
            if page_text != page_text_pre:
                was_unclean += 1
            with page_file.open("w+") as f:
                f.write(page_text)
    print(f"I {path.name} var det {was_unclean} sider (av {num_pages} totalt)␣
 ↪med uønskede tegn (non-breaking space og em-dash)")
```

```
I train var det 45 sider (av 166 totalt) med uønskede tegn (non-breaking space
og em-dash)
I side_30 var det 0 sider (av 2380 totalt) med uønskede tegn (non-breaking space
og em-dash)
I Testsett_Samisk_OCR var det 5 sider (av 21 totalt) med uønskede tegn (non-
```

breaking space og em-dash)

[ ]:
```
from nb_tokenizer import tokenize
from collections import Counter

def get_tokens(directory: Path) -> Counter[str]:
    tokens = []
    for text_dir in directory.glob("**/txt"):
        text_files = text_dir.glob("*.txt")
        texts = " ".join([e.read_text() for e in text_files])
        tokens += tokenize(texts)
    return Counter(tokens)

train_tokens = get_tokens(train_path)
test_tokens = get_tokens(test_path)
s_30_tokens = get_tokens(s_30_path)
```

[ ]:
```
print(f"""Unike tokens:
    Train:      {'{:_}'.format(len(train_tokens))}
    Side 30:    {'{:_}'.format(len(s_30_tokens))}
    Test:       {'{:_}'.format(len(test_tokens))}
""")
```

```
Unike tokens:
    Train:      21_482
    Side 30:    110_998
    Test:       3_251
```

[ ]:
```
print(f"""Totalt antall tokens:
    Train:      {'{:_}'.format(sum(train_tokens.values()))}
    Side 30:    {'{:_}'.format(sum(s_30_tokens.values()))}
    Test:       {'{:_}'.format(sum(test_tokens.values()))}
""")
```

```
Totalt antall tokens:
    Train:      73_282
    Side 30:    523_382
    Test:       6_598
```

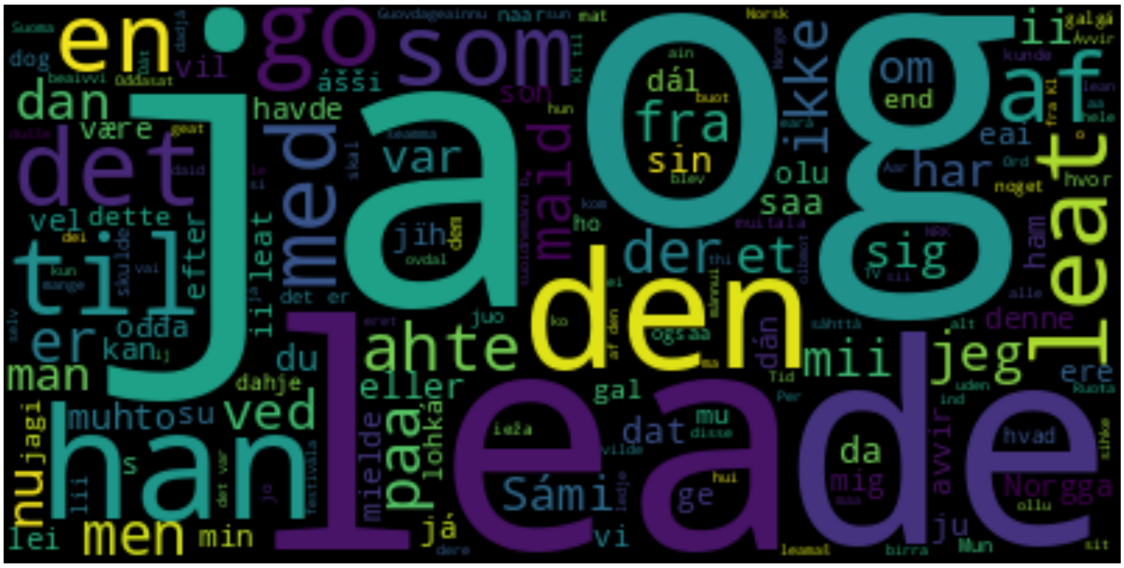
[ ]:
```
from typing import Iterable, Any

def percent_overlap(container1: Iterable[Any], container2: Iterable[Any]) ->␣
 ↪float:
    intersection_len = len(set(container1).intersection(set(container2)))
    intersection_part = intersection_len / len(container1)
    return round(intersection_part * 100, 2)
```

[ ]:
```
print(f"""Tokenoverlapp

    Train og test:
        {percent_overlap(test_tokens, train_tokens)}% av tokensa i test finnes␣
 ↪i train
        {percent_overlap(train_tokens, test_tokens)}% av tokensa i train finnes␣
 ↪i test

    Train og side 30:
        {percent_overlap(train_tokens, s_30_tokens)}% av tokensa i train finnes␣
 ↪i side 30
        {percent_overlap(s_30_tokens, train_tokens)}% av tokensa i side 30␣
 ↪finnes i train

    Test og side 30:
        {percent_overlap(test_tokens, s_30_tokens)}% av tokensa i test finnes i␣
 ↪side 30
        {percent_overlap(s_30_tokens, test_tokens)}% av tokensa i side 30␣
 ↪finnes i test

""")
```

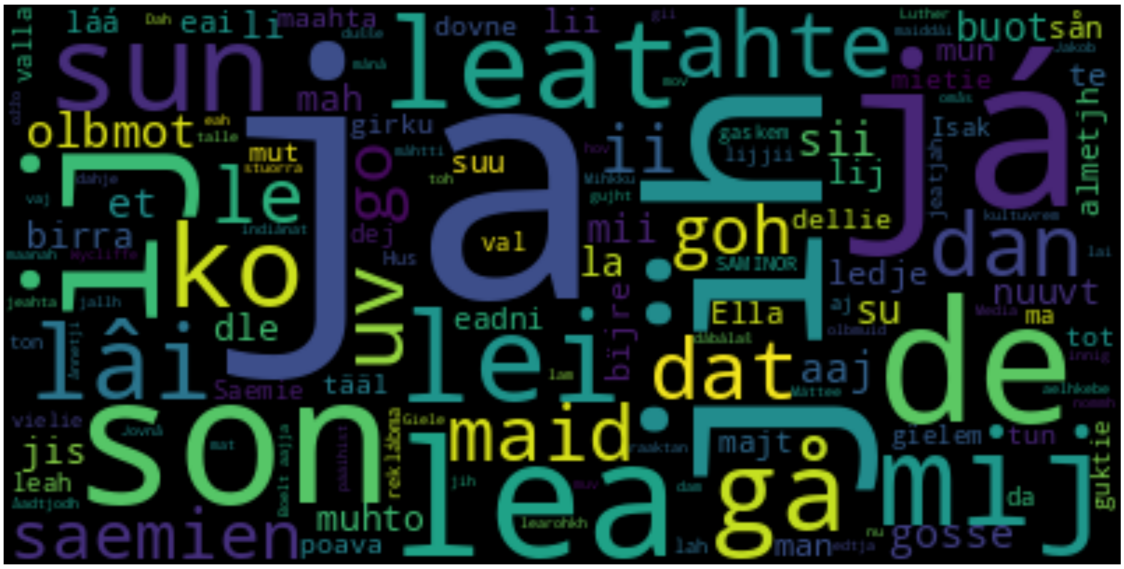
```
Tokenoverlapp

    Train og test:
        29.96% av tokensa i test finnes i train
        4.53% av tokensa i train finnes i test

    Train og side 30:
        39.71% av tokensa i train finnes i side 30
        7.68% av tokensa i side 30 finnes i train

    Test og side 30:
        49.34% av tokensa i test finnes i side 30
        1.45% av tokensa i side 30 finnes i test
```

[ ]:
```
import matplotlib.pyplot as plt
from wordcloud import WordCloud

def get_text(directory: Path) -> str:
    text =""
    for text_dir in directory.glob("**/txt"):
        text_files = text_dir.glob("*.txt")
```

```
        texts = " ".join([e.read_text() for e in text_files])
        text += texts
    return text

train_text = get_text(train_path)
train_cloud = WordCloud().generate(train_text)
plt.figure(figsize=(60,20))
plt.imshow(train_cloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```



[ ]:
```
test_text = get_text(test_path)
test_cloud = WordCloud().generate(test_text)
plt.figure(figsize=(60,20))
plt.imshow(test_cloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```



[ ]:
```
s_30_text = get_text(s_30_path)
s_30_cloud = WordCloud().generate(s_30_text)
plt.figure(figsize=(60,20))
plt.imshow(s_30_cloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

2 Se på bokstaver og tegn i data

[ ]:
```
train_chars = Counter(train_text)
s_30_chars = Counter(s_30_text)
test_chars = Counter(test_text)


print(f"""Tegnoverlapp

    Train og test:
        {percent_overlap(test_chars, train_chars)}% av tegna i test finnes i␣
 ↪train
        {percent_overlap(train_chars, test_chars)}% av tegna i train finnes i␣
 ↪test

    Train og side 30:
        {percent_overlap(train_chars, s_30_chars)}% av tegna i train finnes i␣
 ↪side 30
        {percent_overlap(s_30_chars, train_chars)}% av tegna i side 30 finnes i␣
 ↪train

    Test og side 30:
        {percent_overlap(test_chars, s_30_chars)}% av tegna i test finnes i␣
 ↪side 30
        {percent_overlap(s_30_chars, test_chars)}% av tegna i side 30 finnes i␣
 ↪test

""")
```

```
Tegnoverlapp

    Train og test:
        100.0% av tegna i test finnes i train
        62.26% av tegna i train finnes i test

    Train og side 30:
        86.16% av tegna i train finnes i side 30
        97.86% av tegna i side 30 finnes i train

    Test og side 30:
        97.98% av tegna i test finnes i side 30
        69.29% av tegna i side 30 finnes i test
```

2.1 Forskjeller i tegn mellom settene

[ ]:
```
print(f"Bare i side 30: {set(s_30_chars) - (set(train_chars).
  ↪union(set(test_chars)))}")
```

```
Bare i side 30: {'ǩ', ' ', 'ğ'}
```

[ ]:
```
print(f"Bare i train {set(train_chars) - set(s_30_chars).
  ↪union(set(test_chars))}")
```

```
Bare i train {'€', ' ', ' ', 'ç', 'ú', '"', ' ', 'û', 'í', 'Þ', 'þ', ' ', '√',
'_', 'ô', ' ', '©', ' ', ' ', '½'}
```