In [3]:
```
from pathlib import Path

all_txt_files =[]
for file in Path("/Users/stineschou/Desktop/TF_IDF_ekstern").rglob("*.txt"):
     all_txt_files.append(file.parent / file.name)
# counts the length of the list
n_files = len(all_txt_files)
print(n_files)
```

```
4
```

In [4]:
```
all_txt_files.sort()
all_txt_files[0]
```

Out[4]:
```
PosixPath('/Users/stineschou/Desktop/TF_IDF_ekstern/Interview_FVM.txt')
```

In [5]:
```
all_docs = []
for txt_file in all_txt_files:
    with open(txt_file) as f:
        txt_file_as_string = f.read()
    all_docs.append(txt_file_as_string)
```

In [6]:
```
#import the TfidfVectorizer from Scikit-Learn.
from sklearn.feature_extraction.text import TfidfVectorizer

#implementer denne liste af stopwords: stopord.txt

#f = open('/Users/stineschou/Desktop/stopord.txt','r')
with open('/Users/stineschou/Desktop/stopord.txt') as f:
    lines = f.read().splitlines()
```

In [7]:
```
vectorizer = TfidfVectorizer(max_df=.65, min_df=1, stop_words=lines, use_idf=True, norm=None)
transformed_documents = vectorizer.fit_transform(all_docs)
```

```
/anaconda3/lib/python3.7/site-packages/sklearn/feature_extraction/text.py:391: UserWarning: Your stop_words may be inconsistent with your preprocessing. Tokenizing the stop words generated tokens ['bl', 'ca', 'eks', 'pga'] not in stop
_words.
  'stop_words.' % sorted(inconsistent))
```

In [8]:
```
transformed_documents_as_array = transformed_documents.toarray()
# use this line of code to verify that the numpy array represents the same number of documents that we have in the file list
len(transformed_documents_as_array)
```

Out[8]:
```
4
```

In [9]:
```
import pandas as pd

# make the output folder if it doesn't already exist
Path("./tf_idf_output").mkdir(parents=True, exist_ok=True)

# construct a list of output file paths using the previous list of text files the relative path for tf_idf_output
output_filenames = [str(txt_file).replace(".txt", ".csv").replace("txt/", "tf_idf_output/") for txt_file in all_txt_files]

# loop each item in transformed_documents_as_array, using enumerate to keep track of the current position
for counter, doc in enumerate(transformed_documents_as_array):
    # construct a dataframe
    tf_idf_tuples = list(zip(vectorizer.get_feature_names(), doc))
    one_doc_as_df = pd.DataFrame.from_records(tf_idf_tuples, columns=['term', 'score']).sort_values(by='score', ascending=False).reset_index(drop=True)

    # output to a csv using the enumerated value for the filename
    one_doc_as_df.to_csv(output_filenames[counter])
```

In [ ]:

In [ ]: