## EDA

Student: Femke Bakker (13264443)

- Goal of the research: classify documents using LLMs
- The data consists of documents and their labels (classes). Besides the text from the pdfs there are no other features that will be used for the classification
- The EDA is performed on 20% of the dataset, because the dataset is very large (33000 docs).
- First de data is loaded in and some cleaning of the tokens is done.
- Missing values, documents where no text was extracted from the PDFs, are already removed.

Link to github: https://github.com/Amsterdam-Internships/document-classification-using-large-language-models/tree/main

```python
import sys
sys.path.append('../scripts/')

import pandas as pd
from load_data import load_txt_files
import nltk
from nltk.tokenize import word_tokenize
import matplotlib.pyplot as plt

txtfile_paths = pd.read_csv("../data/txtfile_paths.csv")
all_txt_df = load_txt_files(txtfile_paths, ['complete'])
df = all_txt_df.copy()

df['tokens'] = df['text'].apply(word_tokenize)

def calculate_length(token_list):
    return len(token_list)

df['token_length'] = df['tokens'].apply(calculate_length)
SAVED_DF = df.copy()

import numpy as np

missing_text = df[df['text'].str.len() < 5]
print(f"There are {len(missing_text)} documents that have less than 5 
characters")

print(f"Removing the documents that could not be extracted or have 
less than 5 characters, leaves us with {len(df)-len(missing_text)} 
(out of {len(df)})")
df = df[df['text'].str.len() > 5]
```

```
There are 11 documents that have less than 5 characters
Removing the documents that could not be extracted or have less than 5
characters, leaves us with 33117 (out of 33128)
```

```python
from nltk.corpus import stopwords
import string
import nltk
nltk.download('stopwords')


# remove stopwords
top_df = df.copy()

def remove_stopwords(tokens):
    stop_words = set(stopwords.words('dutch'))
    tokens_without_stopwords = [word for word in tokens if
word.lower() not in stop_words]
    tokens_without_punctuation = [word for word in
tokens_without_stopwords if word not in string.punctuation and
len(word)>1]
    return tokens_without_punctuation


df['cleaned_tokens'] = df['tokens'].apply(remove_stopwords)
df['clean_text'] = df['cleaned_tokens'].apply(lambda x: ' '.join(x))
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     /home/azureuser/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Token & class distribution

Below are the tables shown for the complete dataset and the classes.

We can see that the token counts within the classes and between the classes are varying. This could create difficulties when converting the docs into suitable input for the LLMs. The LLMs have a max token limit, meaning that whole documents cannot be given as input and thus the docs need to be represented in another way. Furthermore, we can see the distribution of the amount of docs per class is some what even, with begroting being an outlier, with a lot of docs. Actualiteit and Factsheets are a little underrepresented compared to the other classes.

Note: the minimum values are very small. This is likely a result of mistakes made during converting PDF to OCR. Further expection is needed to remove bad files.

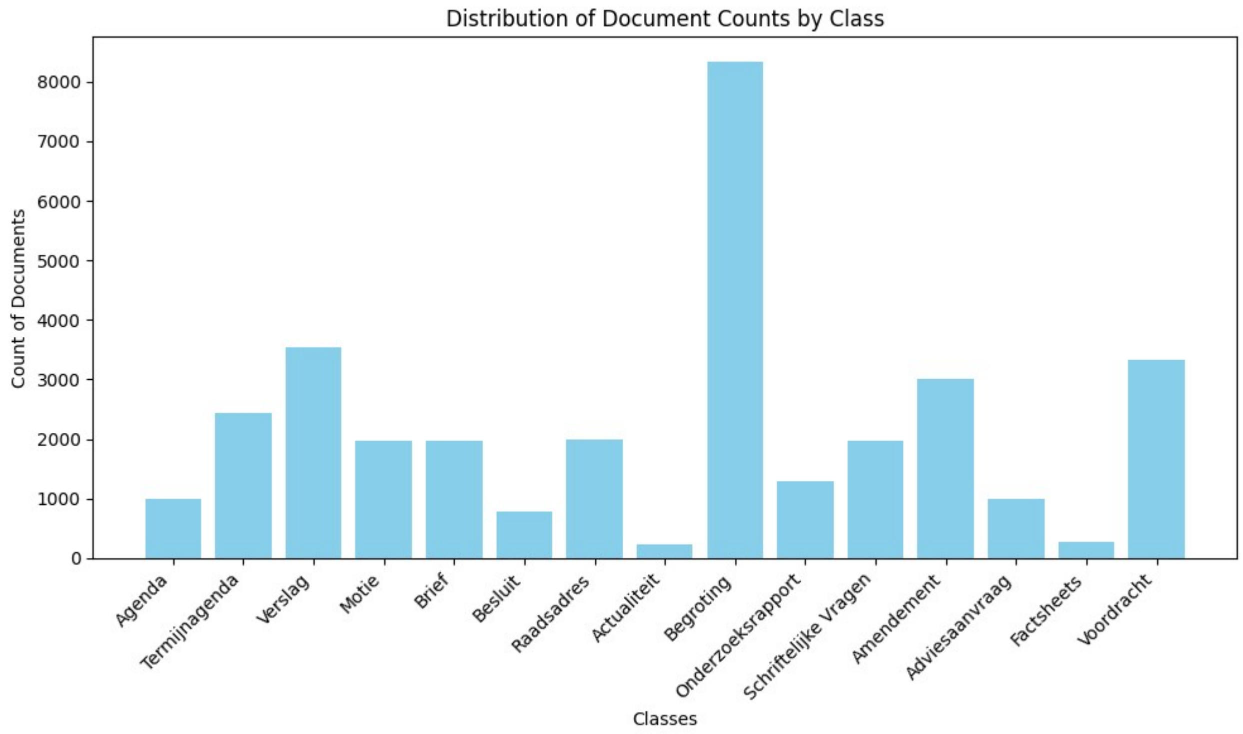
```python
display(df['token_length'].describe())
class_describe = df.groupby('label')['token_length'].describe()
display(class_describe)

classes = list(set(df['label']))
counts = list(class_describe['count'])
```

```python
# Plotting the distribution
plt.figure(figsize=(10, 6))
plt.bar(classes, counts, color='skyblue')
plt.xlabel('Classes')
plt.ylabel('Count of Documents')
plt.title('Distribution of Document Counts by Class')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```

```
count     33117.000000
mean       2691.847933
std       13409.818780
min           3.000000
25%         291.000000
50%         533.000000
75%        1225.000000
max      275597.000000
Name: token_length, dtype: float64
```

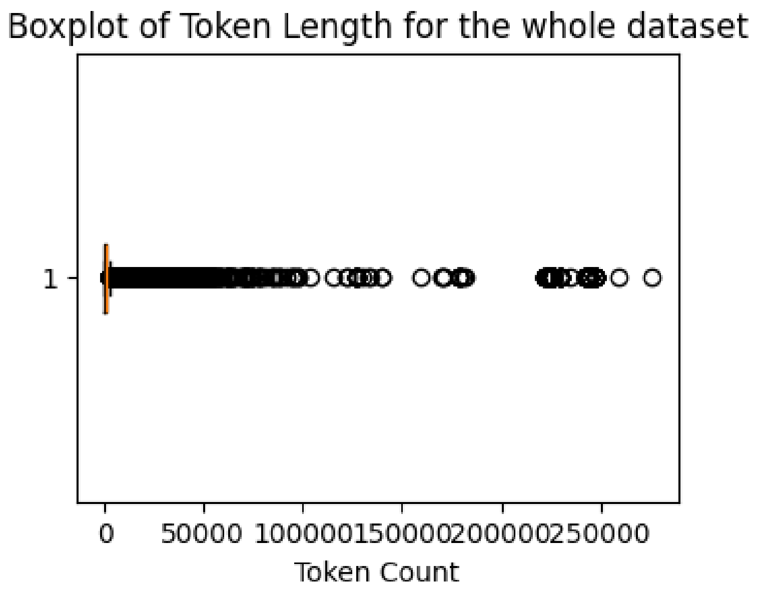
```
                       count          mean           std     min
25%  \
label

Actualiteit            996.0    696.773092   3462.293848    72.0
236.00
Adviesaanvraag        2442.0   1644.871417   2993.302964    57.0
557.00
Agenda                3537.0   1048.904156   4244.783305    20.0
325.00
Amendement            1969.0   2658.493652   7044.470410    30.0
236.00
Begroting             1967.0  13160.190646  46780.793626    40.0
248.00
Besluit                775.0    986.649032   1749.006500    70.0
168.50
Brief                 1995.0   1764.259649   1728.612069     3.0
734.00
Factsheets             234.0   6008.987179  14742.837685   112.0
1154.00
Motie                 8336.0    521.707893   1129.571368   105.0
234.00
Onderzoeksrapport     1286.0  15329.244168  18086.586572   233.0
5896.25
Raadsadres            1975.0   1072.450633   1666.327342    34.0
358.50
Schriftelijke Vragen  3004.0   1742.592543   5832.719196    26.0
871.75
Termijnagenda          996.0    545.188755    346.153706    48.0
```

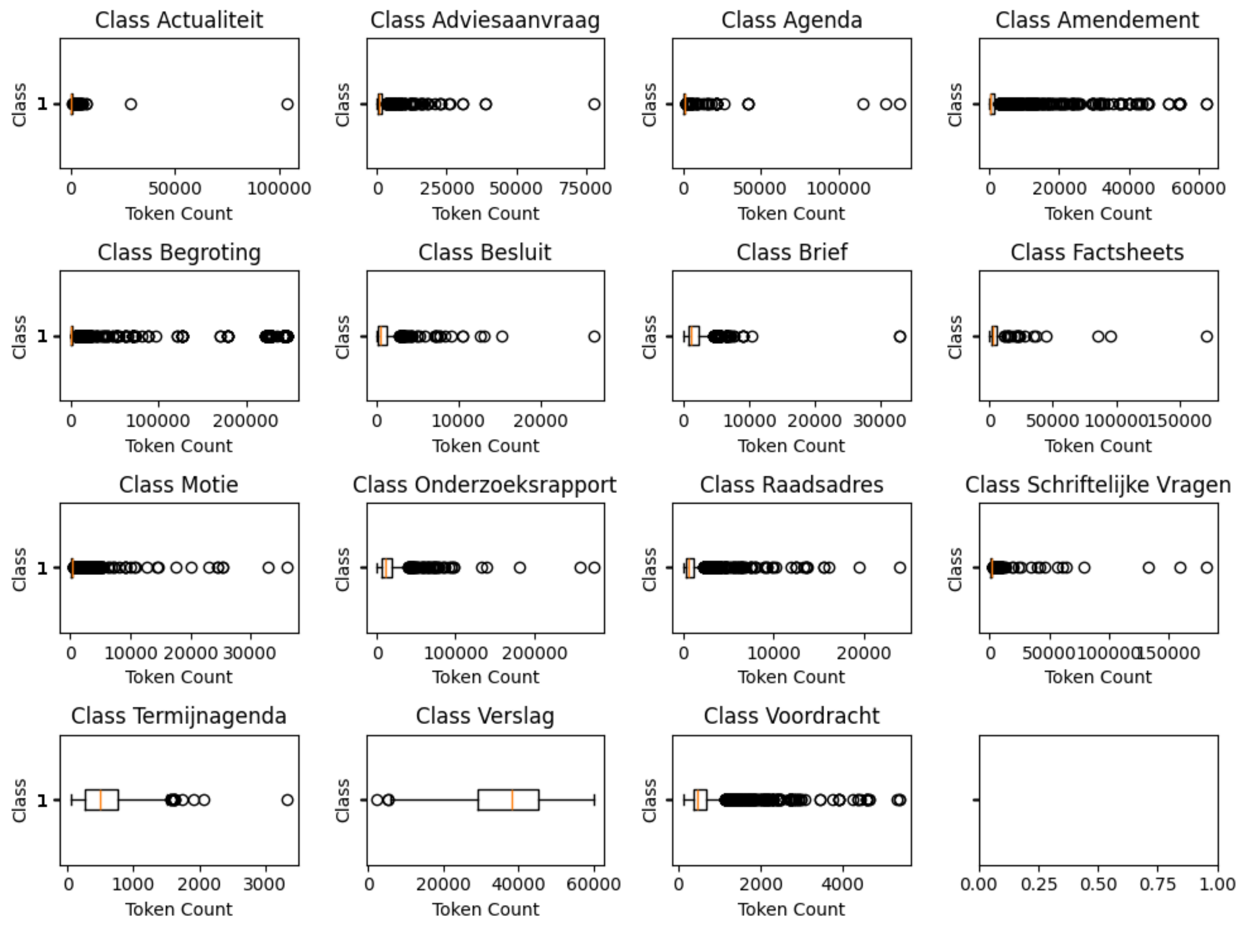
```
257.00
Verslag                  273.0  36233.520147  12422.807218  2398.0
29160.00
Voordracht              3332.0    619.479892    497.505838   121.0
368.00

                          50%      75%       max
label
Actualiteit             344.5    592.0  103700.0
Adviesaanvraag          955.0   1672.0   77769.0
Agenda                  616.0    939.0  139648.0
Amendement              359.0   1184.0   62304.0
Begroting               385.0   2324.0  247184.0
Besluit                 439.0   1216.0   26523.0
Brief                  1269.0   2291.5   32957.0
Factsheets             2753.5   5462.0  171297.0
Motie                   292.5    402.0   36091.0
Onderzoeksrapport     10999.0  19127.5  275597.0
Raadsadres              627.0   1104.0   24021.0
Schriftelijke Vragen   1209.5   1675.0  181725.0
Termijnagenda           486.0    761.5    3338.0
Verslag               38222.0  45151.0   59938.0
Voordracht              468.0    673.5    5390.0
```



```python
import matplotlib.pyplot as plt
```

```python
plt.figure(figsize=(4, 3))
plt.boxplot(df['token_length'], vert=False)
plt.title('Boxplot of Token Length for the whole dataset')
plt.xlabel('Token Count')
plt.show()

grouped = df.groupby('label')

# Create subplots
fig, axs = plt.subplots(4, 4, figsize=(10, 7.5), sharey=True)

# Iterate over groups and plot boxplots
for i, (label, group) in enumerate(grouped):
    row = i // 4
    col = i % 4
    axs[row, col].boxplot(group['token_length'], vert=False)
    axs[row, col].set_title(f'Class {label}')
    axs[row, col].set_xlabel('Token Count')
    axs[row, col].set_ylabel('Class')

# Hide empty subplots if any
for i in range(len(grouped), 3*5):
    row = i // 5
    col = i % 5
    axs[row, col].axis('off')

plt.tight_layout()
plt.show()
```





Based on the plots above we can see that there a classes that dont have (many) outliers in token counts, such as termijnagenda, verslag and brief. On the other hand there are also classes that have many outliers in token count, such as voordracht, begroting and moties. It's noticeable that the outliers are on the right side, meaning that the outliers have a high token count.

## Top words for each class

The goal is to find words that are identifiers for the classes. We suspect that the name of the class, for example 'agenda', 'motie', 'begroting', will be strong identicators for a class. However, we also know that these words are used in documents that don't belong to that class. The goal of this part of the analysis is to find how severe this is.

1. select top 50 words with highest TF-IDF mean for each class.
2. Then select the unique words for each class. Words that do not occur in the top-50 of other classes.
3. Check if other class names are named in docs of a class.

```python
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
tfidf_vectorizer = TfidfVectorizer()
tfidf_matrix = tfidf_vectorizer.fit_transform(df['clean_text'])
tfidf_df = pd.DataFrame(tfidf_matrix.toarray(),
columns=tfidf_vectorizer.get_feature_names_out())
tfidf_df['label'] = df['label']

# Calculate mean TF-IDF score for each word across all documents in
each class
mean_tfidf_by_class = tfidf_df.groupby('label').mean()

# Get top 50 words with highest mean TF-IDF score for each class
top_50_words = {}
for class_name, class_tfidf in mean_tfidf_by_class.iterrows():
    top_50_words[class_name] = class_tfidf.nlargest(50).index.tolist()

unique_words_dict = {}
for class_name in top_50_words.keys():
    top_words = top_50_words[class_name]

    all_other_words = []
    for other_name in top_50_words.keys():
        if class_name != other_name:
            all_other_words.extend(top_50_words[other_name])
    all_other_words = set(all_other_words)

    unique_words = [word for word in top_words if word not in
all_other_words]
    print(f'{class_name}: ---- {unique_words}')
```

```
Actualiteit: ---- ['actualiteit', 'spoedeisendheid', 'reden',
'raadsactualiteit', 'behandeling', 'maart', 'poot', 'supplement',
'wassink', 'februari', 'indieners']
Adviesaanvraag: ---- ['stadsdeelcommissie', 'adviesaanvraag', 'art',
'concept', 'stadsdelen', 'staf', 'invullen', 'adviezen', '2605',
'81484', 'z17', 'sdc', 'verordening', 'directie', 'kaders', 'kader',
'meegestuurde', 'weekstart', 'conceptadvies', 'procesbegeleider',
'formuleer', 'vraagt', 'optioneel']
Agenda: ---- ['dient', 'aanvang', 'bd2013']
Amendement: ---- ['pvda', 'stadsdeelraad', 'sp', 'dhr', 'groenlinks',
'deelraad', 'bijeen', 'bnw81', 'cda', 'dag']
Begroting: ---- ['2010', '2009', '2008', 'miljoen', '2007',
'december', 'lasten', 'baten', 'reserve', 'euro', 'reserves',
'kosten', 'middelen', 'saldo', 'behandelen', 'bedrag']
Besluit: ---- ['algemeen', 'ab', 'int', 'bestemmingsplan', 'decos',
'nieuwwest', 'ivar', 'baâdoud', '1064', '14020', 'wink', 'sw', '2003',
'plein', 'stadsdeelsecretaris', 'bestuurscommissies', 'ca',
'bezoekadres', 'vast', 'ontwerpbestemmingsplan', 'afschrift', 'gelet']
Brief: ---- ['routebeschrijving', 'vindt', 'portefeuillehouder',
'2024', 'geïnformeerd', 'maatregelen', 'ontwikkeling', 'portefeuille',
'verschillende', 'vit', 'daarnaast']
Factsheets: ---- ['aandeel', 'bron', '65', 'minima', 'huishoudens',
'minimahuishoudens', 'gemiddelde', 'the', 'zuidoost', 'buurten',
'gemiddeld', 'and', 'inkomen', 'factsheet', 'cbs', '18', 'ouderen',
'cijfers', 'ois', 'leerlingen']
Motie: ---- ['status', 'verzoekt', 'ondergetekenden', 'verworpen',
'amsterdamse']
Onderzoeksrapport: ---- ['es', 'ie', 're', 'ed', 'el', 'ke', 'pe',
'et', 'tussen', 'figuur', 'mn', 'he', 'le', 'scholen', 'be']
Raadsadres: ---- ['kenmerk', '202', '020', 'mail', 'www', 'graag',
'bijlage', '552', 'komen', 'alleen', 'verzonden', 'binnenstad',
'retouradres']
Schriftelijke Vragen: ---- ['antwoord', 'schriftelijke', 'politie',
'beantwoording', 'vorenstaande', 'welke', 'grond', 'nee',
'toelichting', 'vragensteller', 'waarom', 'reglement', 'bekend',
'akkoord', '45', 'neng', 'augustus']
Termijnagenda: ---- ['termijnagenda', 'bd2015', 'bd2012',
'actualiteiten', '15', 'cie']
Verslag: ---- ['heel', 'raadsnotulen', 'woord', 'even', 'denk',
'natuurlijk', 'goed', 'echt', 'vind', 'zeggen', 'eigenlijk', 'gewoon',
'geeft', 'willen', 'misschien', 'dank', 'stemming', 'gezegd', 'zegt',
'zitten', 'weet']
Voordracht: ---- ['pdf', 'nvt', 'gegenereerd', 'vl', 'kennisneming',
'ad2023', 'naam', 'bespreking', 'ad2022', 'ad2021', 'gemeentewet',
'geheimhouding', 'raadscommissies', 'toezegging', 'tijdelijke',
'indienend', 'uitgenodigde', 'afgedaan', 'mailadres', 'behandelend',
'treft', 'telefoonnummer', 'extern', 'grondslag', 'ambtenaar',
'algemene', 'inzage', 'wettelijke', 'achtergrond', 'bestuurlijke',
'gevraagd', 'uitkomsten']


# get dict with all tokens (including duplicates) for each class
tokens_class = dict()
for class_name in set(df['label']):
    all_tokens = list(df.loc[df['label']==class_name]
['cleaned_tokens'].values)
    all_tokens = [token for sublist in all_tokens for token in
sublist]
    tokens_class[class_name] = all_tokens

# For each class count how many times the class names are in the
tokens
df_col = list(tokens_class.keys())
df_col.append("Total Doc in Class")
class_count_df = pd.DataFrame(columns=df_col)

for class_name in tokens_class.keys():
    tokens = tokens_class[class_name]
    counts = {label: tokens.count(label.lower()) for label in
```

```
tokens_class.keys()}
    counts['Total Doc in Class'] =
len(df.loc[df['label']==class_name])
    class_count_df.loc[len(class_count_df)] = counts

class_count_df.set_index(pd.Index(list(tokens_class.keys())),
inplace=True)


# high the df to make result more interpretable
def highlight_max_and_second_highest_except_total(s):
    max_val = s[:-1].max()
    second_max_val = s[:-1].nlargest(2).iloc[-1]  # Get the second
largest value
    is_max = s[:-1] == max_val
    is_max['Total Doc in Class'] = False
    is_second_max = s[:-1] == second_max_val
    is_second_max['Total Doc in Class'] = False

    styles = ['background-color: green' if v else '' for v in is_max]
    styles_second = ['background-color: orange' if v else '' for v in
is_second_max]
    combined_styles = [f'{styles[i]}; {styles_second[i]}' for i in
range(len(styles))]
    return combined_styles

styled_df =
class_count_df.style.apply(highlight_max_and_second_highest_except_tot
al, axis=1)

display(styled_df)
```

```
<pandas.io.formats.style.Styler at 0x7f46c7bd1070>
```

The green marked values are the highest count in that row and the orange ones are the second highest. We want the class_name to be named a lot in the class and very few in the other classes, that means that the class name is an identifier for that class.

Notes:

- schriftelijke vragen is never named in the docs. Which is understandable, since schriftelijke vragen are questions about very different subjects
- the distinction between agenda and termijnagenda might be hard, based on the count of agenda and termijnagenda in them. They both name agenda a lot.
- Verslag might be confused with Motie, since motie is named very often.

Overall there is quite some overlap between the classes, in regards of naming the class names.

## Similarity within class

If the docs within a class are similar, it will be easier to classify them, because then patterns are easier to identify by the model. The average cosine similarity between docs of a class is calculated. Since the texts need to be shortened to give as input, the similarity of the first 1000 tokens of each doc is also compared.

We can see in the tables below the similarity of the documents within the classes. It shows that the documents are not similar to each other, since the scores are low. Verslag is the class with the highest similarity between the documents. Additionally, the documents are more similar if the whole document is compared, than when the first 1000 tokens of the docs are compared. This could be a problem, or at least worse the performance if the docs are represented by the first N tokens. This could mean that other representation methods such as summarizing might be a better fit to represent the docs.

```python
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
import math


def similarity_within_class(input_df, text_len):
    work_df = input_df.copy()
    within_sim = pd.DataFrame(columns=['label', 'average_sim',
'num_pairs_exceeding_threshold'])

    # if not complete
    if text_len == 'complete':
        text_column = 'text'

    else:
        work_df['short_text'] = work_df['tokens'].apply(lambda tokens:
' '.join(tokens[:text_len]))
        text_column = 'short_text'


    for category in set(work_df['label']):
        subdf = work_df.loc[work_df['label']==category]

        # get tf-idf score
        tfidf_vectorizer = TfidfVectorizer()
        tfidf_matrix =
tfidf_vectorizer.fit_transform(subdf[text_column])

        # calculate cosine similarity
        cosine_similarities = cosine_similarity(tfidf_matrix,
tfidf_matrix)

        # take average sim
        average_similarity = round(np.mean(cosine_similarities), 2)
        # calculate how many pairs succeed threshold
        total_possible_pairs = len(cosine_similarities) *
(len(cosine_similarities))
        num_pairs_exceeding_threshold =
round(np.sum(cosine_similarities > 0.5) / total_possible_pairs, 2)

        # add to df
        within_sim.loc[len(within_sim)] = {'label':category,
'average_sim': average_similarity,
'num_pairs_exceeding_threshold':num_pairs_exceeding_threshold}

        within_sim = within_sim.sort_values(by='average_sim',
ascending=False)
    return within_sim

# sim_within_df_500tokens = similarity_within_class(df, 500)
# display(sim_within_df_500tokens)

sim_within_df_1000tokens = similarity_within_class(df, 1000)
display(sim_within_df_1000tokens)

sim_within_df_cleantext = similarity_within_class(df, 'complete')
display(sim_within_df_cleantext)
```

```
                   label  average_sim  num_pairs_exceeding_threshold
2                Verslag         0.47                           0.32
12        Adviesaanvraag         0.28                           0.01
4                  Brief         0.26                           0.00
10  Schriftelijke Vragen         0.25                           0.00
0                 Agenda         0.24                           0.04
5                Besluit         0.20                           0.01
1            Termijnagenda         0.19                           0.05
7            Actualiteit         0.18                           0.01
8              Begroting         0.18                           0.00
9      Onderzoeksrapport         0.18                           0.01
14            Voordracht         0.18                           0.00
6             Raadsadres         0.17                           0.00
13            Factsheets         0.17                           0.02
11            Amendement         0.16                           0.00
3                  Motie         0.14                           0.00
```

```
                   label  average_sim  num_pairs_exceeding_threshold
2                Verslag         0.91                           1.00
9      Onderzoeksrapport         0.48                           0.47
12        Adviesaanvraag         0.34                           0.05
4                  Brief         0.33                           0.05
8              Begroting         0.30                           0.07
10  Schriftelijke Vragen         0.29                           0.00
0                 Agenda         0.26                           0.05
13            Factsheets         0.25                           0.05
```

```
5             Besluit        0.22                              0.01
11         Amendement        0.22                              0.03
1       Termijnagenda        0.19                              0.05
6          Raadsadres        0.19                              0.00
7         Actualiteit        0.19                              0.01
14         Voordracht        0.18                              0.00
3               Motie        0.15                              0.00
```