* Use matplotlib, seaborn or other Python libraries for visualization to investigate the dataset

The dataset contains rows with: topic; article.

From test.csv.:

Wirtschaft;'Die Gewerkschaft GPA-djp lanciert den "All-in-Rechner" und findet, … oder All-in.'

Sport;Franzosen verteidigen 2:1-Führung – Kritische Stimmen zu Schwedens Superstar, … getroffen.

| **Category** | **Train** | **Test** |
| --- | --- | --- |
| Web | 1510 | 168 |
| Panorama | 1509 | 168 |
| International | 1360 | 151 |
| Wirtschaft | 1270 | 141 |
| Sport | 1081 | 120 |
| Inland | 913 | 102 |
| Etat | 601 | 67 |
| Wissenschaft | 516 | 57 |
| Kultur | 485 | 54 |

From the dataset site:

“As in most real-world datasets the class distribution of the 10kGNAD is not balanced. The biggest class Web consists of 1678, while the smallest class Kultur contains only 539 articles. However articles from the Web class have on average the fewest words, while articles from the culture class have the second most words. See below for a detailed overview over the class size.”

Possible analysis:

* number of articles per class; histogram. How much is the dataset imbalanced?
* average number of words in an article, per class; histogram. Note that this visualization and the previous one are already found on the site
* Frequency distribution of words per class; a number of line/area plots?
* Or maybe, yet another histogram that highlights how much of the vocabulary used in a class is unique to that class.
* vocabulary for each class; is there any way to visualize the overlap? maybe a numerical table is the only way, where at each column there are C-1 values, expressing the overlap in vocabulary between the 2 classes. It could be a matrix with a color-coding on the cells (color-map), in addition to the numerical values.

# Technical Assignment (NLP)

* Download the dataset in German: <https://tblock.github.io/10kGNAD/>
* Use matplotlib, seaborn or other Python libraries for visualization to investigate the dataset
* Train a classifier to distinguish between the given labels (if you have enough time, make experiments using a few approaches) and compare the results counting metrics (precision, recall, f1, confusion matrix)
* Create a demo application that you could use to demonstrate the results (streamlit, flask etc.), it would be nice to have the option to upload a .txt file or paste a text and get predictions.
* Add the opportunity to run the app using Docker
* Optional: REST API for model usage
* Publish your code in github