# CopyRIGHT Benchmark Report

Vivian van Oijen
vivian.vanoijen@surf.nl

Matthieu Laneuville
matthieu.laneuville@surf.nl

October 28, 2022

## 1 Introduction

CopyRIGHT is a tool that aims at classifying teaching materials in several categories to help universities estimate their copyright dues. The existing tool uses hardcoded rules and parameters that are manually tested. In this report, we demonstrate the potential of machine learning methods to improve the tool's performance and make it more maintainable. We experiment with two different machine learning models and compare their results to those of the existing tool[1].

## 2 Data analysis

### 2.1 Data description

The dataset contains 766 unique labeled datapoints, each of which represents a document. Each document belongs to one of 6 classes and was classified by an expert from an institution that uses the CopyRIGHT tool. We have the following class distribution:

| **Document type** | **Number of samples** |
|---|---:|
| eigen materiaal - overig | 280 |
| korte overname | 205 |
| middellange overname | 45 |
| lange overname | 24 |
| open access | 9 |
| eigen materiaal - powerpoint | 88 |

Figure 1: Class distribution.

The dataset has 87 columns, but not all can be used as input for our model. Some columns are constants or contain only null values, others are simple not relevant, such as the ID or filepath. Therefore, many columns are dropped.

There are 26 input features. Most of these are booleans that the current tool extracts and uses to make its predictions, such as whether a document has an ISBN or DOI and whether the document has more than 300 words per page or not. The machine learning model can also use additional features that the current tool cannot, such as the pagecount.

Many of the features are correlated, e.g., the word count and whether the document has more than 300 words per page. Figure 2 shows the correlation matrix, using Spearman's rank correlation coefficient.

[1] `https://github.com/sara-nl/copyright-ml`

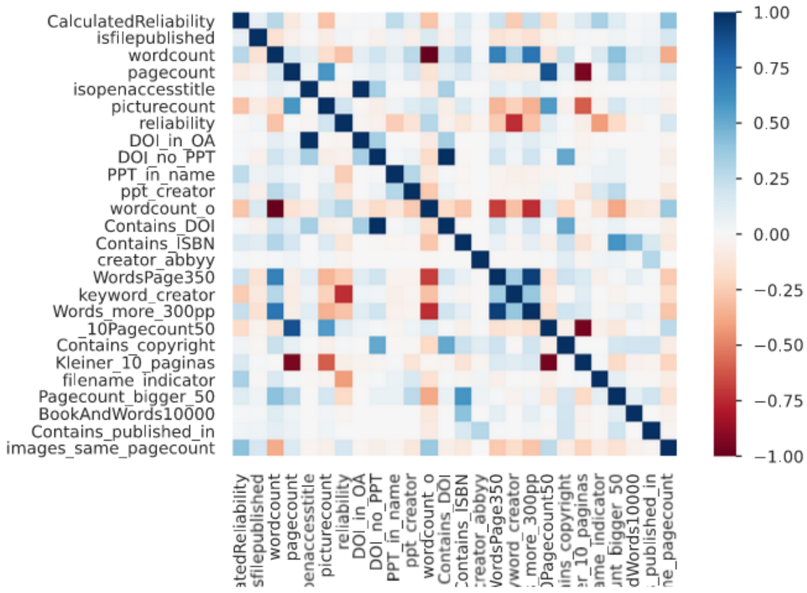

Figure 2: Correlation between features, using Spearman's rank correlation coefficient.

## 2.2 Pre-processing

The first step of data processing is dropping columns that are constant, have only null values and other columns that are not used.

The boolean values are transformed to numerical data, so that 'True' will be 1 and 'False' will be 0. Any null values in these columns are also transformed to 0. We set null values to 0 rather than 1, because False is the dominant class for the samples that are not null.

For one of the models that we used, logistic regression (Section 4.2), we standardize the data, i.e., transform it so that each feature's mean is 0 and the standard deviation is 1.

We use stratified K-fold for the training process, making five different data splits and preserving the percentage of samples per class. For each split, a model is trained from scratch on the train data and then evaluated on the test data. The results of each split are averaged.

## 3 Existing tool

The current tool uses the boolean features as its input and assigns a score to each class based on manually pre-determined feature importances. The class with the highest score is then chosen as the document's label.

The tool currently only uses the boolean features and it is not obvious how to expand it to use other features, such as wordcount. In addition, for every feature that is added, someone needs to manually implement a new rule and find a corresponding score that works well.

## 4 Machine learning Benchmark

### 4.1 XGBoost

The model we propose for this problem is XGBoost, a highly optimized implementation of gradient boosting. This is a tree-based model that is well-suited for classification problems and is relatively robust against highly correlated features. It is also easy to obtain feature importance from this model. This gives us more insight into what the model bases its predictions on. XGBoost performs well on a large variety of problems and is highly efficient[2].

[2]`https://arxiv.org/abs/1806.11248`

### 4.1.1 Implementation

For the implementation we use the XGBClassifier from the xgboost package for Python [3]. We use the default parameters [4].

### 4.1.2 Feature Importances

The top five important features, according to XGBoost can be seen in Figure 3 below. Importance scores are normalized to sum to 1; the absolute score cannot be interpreted, only how each feature scores relative to the others.

| Feature name | Feature importance |
|---|---|
| Words_more_300pp | 0.394 |
| pagecount | 0.132 |
| PPT_in_name | 0.095 |
| isopenaccesstitle | 0.094 |
| keyword_creator | 0.082 |

Figure 3: Feature importance top five as per XGBoost.

There are also features with an importance score of 0; these features have no influence on the model's prediction:

- reliability
- ppt_creator
- wordcount_o
- Contains_ISBN
- _10Pagecount50
- Contains_copyright
- filename_indicator
- Pagecount_bigger_50
- BookAndWords10000

The importance score of these features does not change if the maximum tree depth is increased. This might have something to do with the fact that they are all highly correlated with other features. Many are also skewed, with a large majority of the samples' value being False for that feature.

## 4.2 Logistic regression

Another model we train on the dataset is logistic regression, because it is similar to how the current tool works. Logistic regression is a linear model that assigns a learned weight to each feature. The current tool also assigns a weight to each feature; the difference is that in the current tool the weights are handpicked while logistic regression mathematically optimizes them. Another difference is that logistic regression, as well as XGBoost, outputs a prediction probability which may be useful for the end user. Comparing the two illustrates the difference between manually estimating a classification model and using machine learning to optimize it computationally.

## 5 Results

Figure 4 shows the accuracy and F1-score of the current tool and the two models, using K-fold; F1-score is the harmonic mean between precision and recall, combining them into one number and therefore accounting for both false positives and false negatives. XGBoost has the best performance. Both machine learning models perform significantly better than the current tool.

In Figure 5 we see more details about the performance of XGBoost, including the precision and recall per class. This classification report is made based on a single datasplit, therefore the numbers are slightly different from Figure 4, where K-fold was used. We see that on average recall is higher than precision. We also notice a difference depending on the support, how many samples were available per

[3]`https://xgboost.readthedocs.io/en/stable/index.html`
[4]`https://xgboost.readthedocs.io/en/stable/parameter.html`

class. The performance is relatively bad for the classes “open access” and “lange overname”, which are also the classes that we have the fewest samples of.

Figure 6 shows the confusion matrix for XGBoost. Here we can see exactly how many datapoints are classified correctly per class, and also how incorrectly classified datapoints are labeled by the model.

| | **Accuracy** | **F1-score** |
|---|---|---|
| *Current tool* | 31.15% | 0.2882 |
| *Current tool excl. ”onbekend”* | 50.64% | 0.3744 |
| *XGBoost* | **95.41%** | **0.8777** |
| *Logistic regression* | 92.67% | 0.8466 |

Figure 4: Comparison of different methods’ results. For the machine learning models, these are the results with K-fold. The current tool has classified some datapoints as “onbekend”, which means unknown; the performance on the entire dataset as well as the dataset excluding the datapoints with label “onbekend” is shown.

| | **Precision** | **Recall** | **F1-score** | **Support** |
|---|---|---|---|---|
| *eigen materiaal - overig* | 0.96 | 0.98 | 0.97 | 106 |
| *eigen materiaal - powerpoint* | 0.92 | 0.89 | 0.91 | 27 |
| *korte overname* | 1.00 | 0.96 | 0.98 | 75 |
| *lange overname* | 0.83 | 1.00 | 0.91 | 5 |
| *middellange overname* | 0.92 | 1.00 | 0.96 | 12 |
| *open access* | 0.80 | 0.80 | 0.80 | 5 |
| **Macro average** | 0.91 | 0.94 | 0.92 | 230 |
| **Weighted average** | 0.96 | 0.96 | 0.96 | 230 |

Figure 5: More detailed classification report of XGBoost.

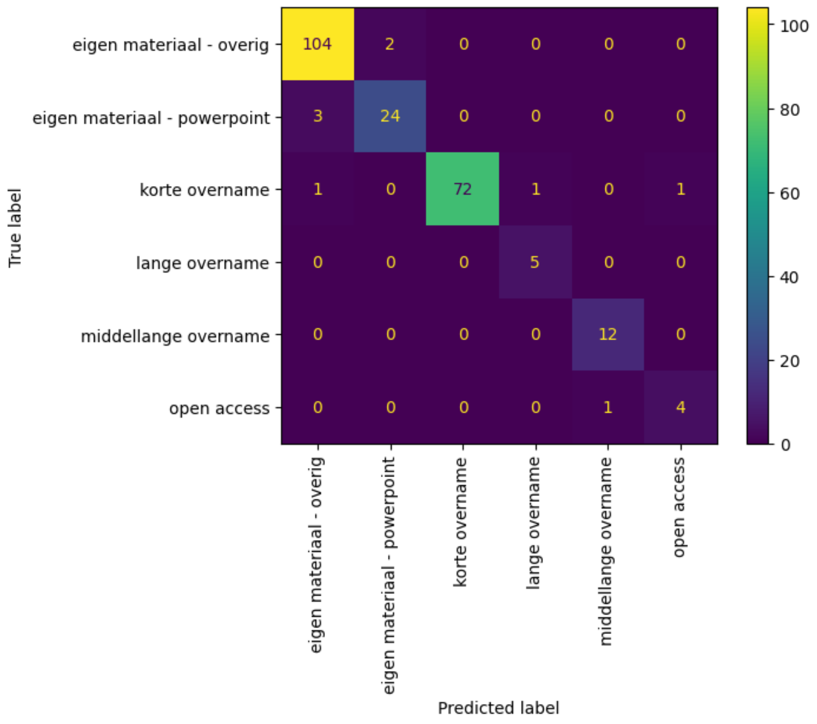

Figure 6: Confusion matrix for XGBoost.

## 6 Discussion & Conclusion

The goal of this experiment is to investigate the potential of machine learning for this problem. Both of the machine learning models that we tried drastically improve performance compared to the existing classification tool, which suggests that expanding the tool with a machine learning model is worthwhile.

We should keep in mind however that this experiment was done on a small subset of the data and that we have not had the chance to test the machine learning models on the full dataset, because most of the data has not been labeled by an expert. Therefore, we cannot say if the model would achieve the same results on a large scale dataset.

It is also important to note that not all classes are of equal importance. The class "lange overname" has the highest priority because of the copyright dues that are associated with it. Therefore, it might be useful to train a separate model solely on finding those documents for example.