# grafzahl: fine-tuning Transformers for text data from within R

Chung-hong Chan[1]

[1] GESIS - Leibniz-Institut für Sozialwissenschaften, Germany

## Author Note

Source code and data are available at https://github.com/chainsawriot/grafzahl. I would like to thank Jarvis Labs for providing discounted GPU cloud service for the development of this package.

Correspondence concerning this article should be addressed to Chung-hong Chan, Unter Sachsenhausen 6-8, 50667 Köln. E-mail: chung-hong.chan@gesis.org

Abstract

This paper introduces `grafzahl`, an R package for fine-tuning Transformers for text data from within R. The package combines the ease of use of the `quanteda` R ecosystem and the state-of-the-art `Transformers` Python library. The package is used in this paper to reproduce the analyses in communication papers or, of non-Germanic benchmark datasets. Very significant improvement in model accuacy over traditional machine learning approach such as Convoluted Neural Network is observed.

*Keywords:* machine learning, transformers, R, python, automated content analysis

Word count: 1951

grafzahl: fine-tuning Transformers for text data from within R

## Put the R back in Transformers

The purpose of this R package, `grafzahl`, is to provide the missing link between the `quanteda` ecosystem and modern Transformers language models. Under the hood, the training part is based on the Python packages `transformers` (Wolf et al., 2020) and `simpletransformers` (Rajapakse, 2022). The integration based on `reticulate` (Ushey, Allaire, & Tang, 2022) is seamless. With this seamless integration provided, communication researchers can produce the most advanced supervised learning models entirely from within R. Therefore, there is no need to switch back and forth between programming languages. This package provides the function `grazahl()`, which emulates the behaviors of `quanteda.textmodels` (Benoit et al., 2021).[1]

## Monolingual classification example

Van Atteveldt, Van der Velden, and Boukes (2021) compare various methods to analyze the tone of Dutch economic news' headlines. Headlines were coded into three categories: negative (-1), neutral (0), and positive (+1).

In the original paper, Van Atteveldt et al. (2021) show that the best method for predicting expert coding, other than coding by student helpers, is convoluted neural network (CNN) with Dutch word embeddings trained on Dutch news. The out-of-sample

[1] This package uses reasonable default settings which suit what communication researchers would like to achieve with these pretrained Transformers models. But the package also provides the freedom for communication researchers to finely adjust the parameters for their specific applications. However, the reanalysis of several examples in communication suggests that even the default settings can generate great improvement over the performance as reported in the original papers. Also, there is almost no need to conduct the cumbersome proprocessing and feature engineering steps, which all examples originally required.

F1 of .63, .66, and .56 were reported for the three categories. As the data (including the training-and-test split) are publicly available,[2] I can provide a head-to-head comparison between the reported CNN and the Transformer-based model trained with the current package.

In the released data (`sentences_ml.csv`), there are three important columns:

1. `headline`: the actual text data
2. `value`: the sentiment
3. `gold`: whether or not this row is "gold standard", i.e. test set. There are 6,038 and 300 headlines in the training and test set respectively.

## Workflow

**Step 0: Setup `grafzahl`.** This step only needs to be done once. A miniconda environment needs to be setup. If there is a GPU capable of performing CUDA, run:

```
require(grafzahl)
setup_grafzahl(cuda = TRUE) # set to FALSE otherwise
```

**Step 1: Get information of the pretrained Transformer.** The first step of training a Transformer-based model is to find a suitable pretrained Transformer model on Hugging Face,[3] which would work for the data. As the data are in Dutch, a pretrained Dutch Transformer model such as BERTje should work (de Vries et al., 2019, available from https://huggingface.co/GroNLP/bert-base-dutch-cased). The model name of this model is `GroNLP/bert-base-dutch-cased`. It is also important to note the citation information to properly cite the pretrained Transformer model.

[2] https://github.com/vanatteveldt/ecosent/

[3] Hugging Face (https://huggingface.co) is an online repository of pretrained machine learning models.

**Step 2: Create the corpus.** The second step is to read the data as a corpus. The easiest way to do that is to use the R package `readtext` (Benoit & Obeng, 2021).[4]

```
require(readtext)
require(quanteda)
url <- "https://raw.githubusercontent.com/vanatteveldt/ecosent/master/data/intermediate/
input <- readtext(url, text_field = "headline") %>%
    corpus
```

We can manipulate the corpus object using the functions provided by `quanteda`. For example, one can subset the training set using the function `corpus_subset()`.

```
## selecting documents where the docvar `gold` is FALSE
training_corpus <- corpus_subset(input, !gold)
```

**Step 3: Fine-tune the model.** With the corpus and model name, the `grafzahl` function is used to fine-tune the model.

```
model <- grafzahl(x = training_corpus,
                  y = "value",
                  model_name = "GroNLP/bert-base-dutch-cased")
```

In general, it is better to specify `output_dir` (where to put the saved model object). By default, it will be `output` in the current working directory. The R function `set.seed()` can also be used to preserve the random seed for reproducibility.

[4] This step is not absolutely needed. The package can also work with character vectors. The `corpus` data structure, which is a better representation of character vector, makes the workflow easier. Also, `readtext` can read the data directly from the url: https://raw.githubusercontent.com/vanatteveldt/ecosent/master/data/intermediate/sentences_ml.csv

On a regular off-the-shelf gaming laptop with a GeForce RTX 3050 Ti GPU and 4G of GPU ram, the process took around 20 minutes. It is in general not recommended to train this without a CUDA-compatible GPU. In those cases, the process might take days, if not weeks.

**Step 4: Make prediction.** Following the convention of `lm()` and many other R packages, the object returned by the function `grafzahl()` has a `predict()` S3 method. The following code gets the predicted sentiment of the headlines in the test set.

```
test_corpus <- corpus_subset(input, gold)
predicted_sentiment <- predict(model, test_corpus)
```

**Step 5: Evaluate performance.** With the predicted sentiment and the ground truth, there are many ways to evaluate the performance of the fine-tuned model. The simplest way is to construct a confusion matrix using the standard `table()` function.

```
cm <- table(predicted_sentiment, ground_truth = docvars(test_corpus, "value"))
```

The R package `caret` can also be used to calculate standard performance metrics such as Precision, Recall, and F1.[5]

```
require(caret)
confusionMatrix(cm, mode = "prec_recall")
```

The out-of-sample F1 measures of the fine-tuned model are .76, .67, and .72 (vs reported .63, .66, and .56). There is great improvement over the CNN model reported by

[5] The function `confusionMatrix()` can accept the predicted values and ground truth directly, without using `table()` first. But the predicted values and ground truth must be `factor`: `confusionMatrix(as.factor(predicted_sentiment), as.factor(docvars(test_corpus, "value")), mode = "prec_recall")`.

Van Atteveldt et al. (2021), although the prediction accuracy for the neutral category is just on par. Van Atteveldt et al. (2021) also provide the learning curve of CNN and Support Vector Machine (SVM). A learning curve plots the out-of-sample prediction performance as a function of number of training examples. I repeat the analysis in a similar manner to Van Atteveldt et al. (2021) and plot the learning curve of Transformer-based model trained using the default workflow of `grafzahl`. Figure 1 show the fine-tuned Transformer model's learning curve alongside CNN's and SVM's. The fune-tuned model has much better performance than CNN and SVM even with only 500 training examples. Unlike CNN and SVM, the gain in performance appears to plateau after 2500. It points to the fact that one does not need to have a lot of training data to fine-tune a Transformer model.

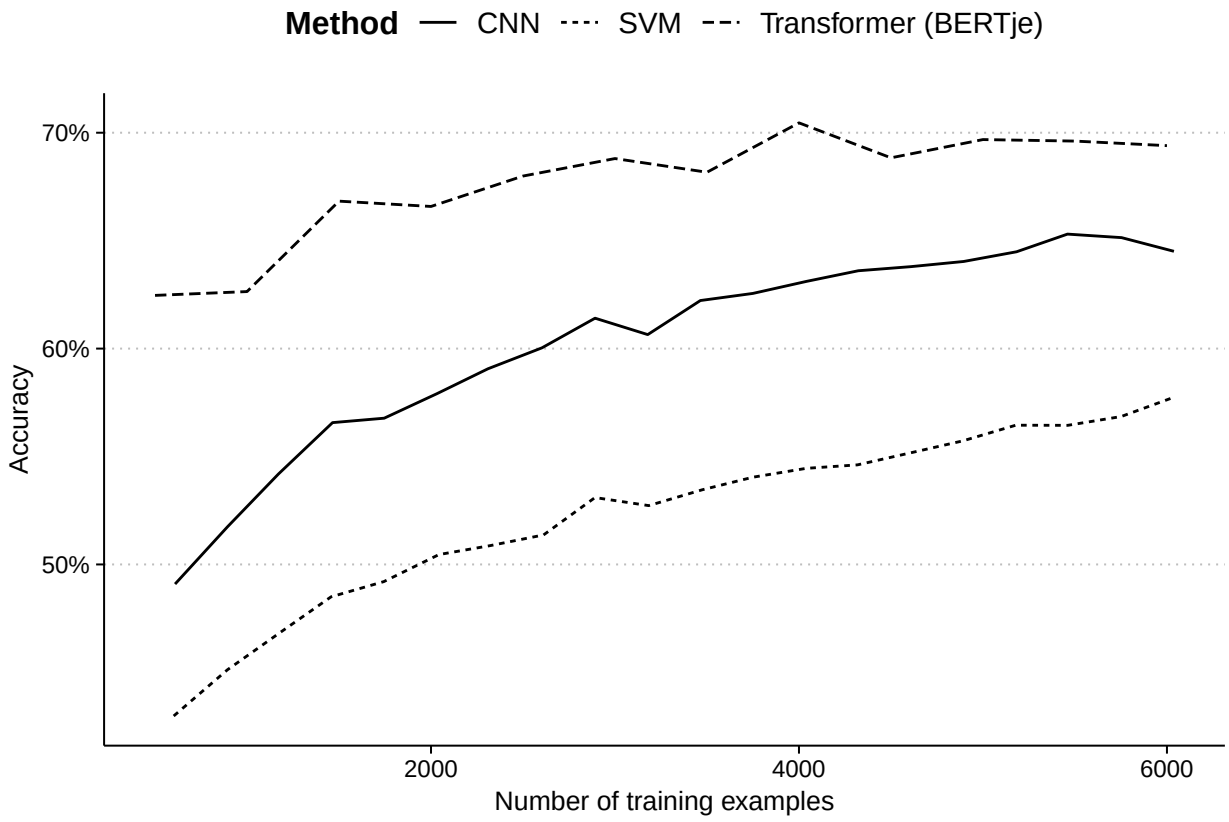


*Figure 1.* Learning curve of machine learning algorithms

**Step 5: Explain the prediction.** Unlike "glass-box" machine learning models (Dobbrick, Jakob, Chan, & Wessler, 2021), Transformer-based prediction models are "black-box". There are so many parameters in Transformers (the BERT base model has 110 million parameters) and this complexity makes each individual parameter of a model not interpretable.

A reasonable compromise is to make the prediction *explainable* instead. Generating Local Interpretable Model-agnostic Explanations (LIME) (Ribeiro, Singh, & Guestrin, 2016; R implementation by Pedersen & Benesty, 2021) is a good way to explain how the model makes its prediction. The gist of the method is to perturb the input text data by deleting parts of the sentence. For example: the sentence "I hate this movie" will be perturbed as "I this movie", "I hate movie", "I hate this", "I hate" etc. These perturbed sentences are then feed into the machine learning model to make predictions. The relationship between what get deleted and the prediction is studied. The parts that change the prediction a lot would be more *causal* to the original prediction.

With the trained model, we can explain the predictions made for the following two Dutch headlines: *"Dijsselbloem pessimistisch over snelle stappen Grieken"* (Dijsselbloem [the Former Minister of Finance of the Netherlands] pessimistic about rapid maneuvers from Greeks) and *"Aandelenbeurzen zetten koersopmars voort"* (Stock markets continue to rise). Models trained with `grafzahl` support the R package `lime` directly. One can get explanations using the following code:

```
require(lime)
sentences <- c("Dijsselbloem pessimistisch over snelle stappen Grieken",
               "Aandelenbeurzen zetten koersopmars voort")
explainer <- lime(training_corpus, model)
explanations <- explain(sentences, explainer, n_labels = 1,
                        n_features = 3)
```

```
plot_text_explanations(explanations)
```

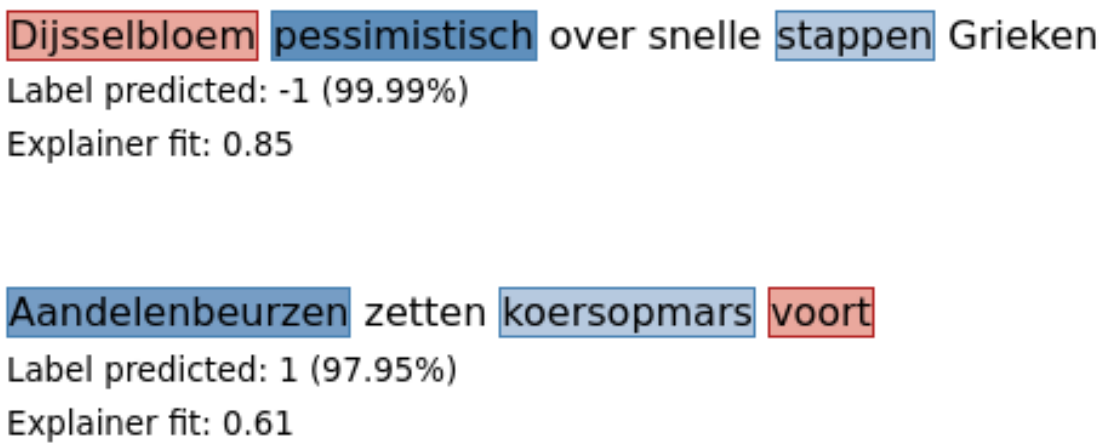


*Figure 2.* Generating Local Interpretable Model-agnostic Explanations (LIME) of two predictions from the trained Dutch sentiment model

Figure 2 shows that for the sentence *"Dijsselbloem pessimistisch over snelle stappen Grieken"* (classified as negative), the tokens *pessimistisch* and *stappen* are making the prediction towards the classified position (negative). But the token *Dijsselbloem* is making it away.

### Non-Germanic example: Amharic

I want to emphasize that `grafzahl` is not just another package focusing only on English, or Germanic languages such as Dutch. Baden, Pipal, Schoonvelde, and van der Velden (2021) criticize this tendency.

Amharic is a Semitic language mainly spoken in Ethiopia and is in general considered to be a "low resource" language. (Joshi, Santy, Budhiraja, Bali, & Choudhury, 2020) Only

recently, the first news classification dataset called "Amharic News Text classification Dataset" is available (Azime & Mohammed, 2021). The dataset contains 50,706 news articles curated from various Amharic websites. The original paper reports the baseline out-of-sample accuracy of 62.2% using Naive Bayes. The released data also contains the training-and-test split.[6] In this example, the AfriBERTa is used as the pretrained model (Ogueji, Zhu, & Lin, 2021). The AfriBERTa model was trained with a small corpus of 11 African languages. Similar to the previous example, the default settings of `grafzahl` are used.

```
url <- "https://huggingface.co/datasets/israel/Amharic-News-Text-classification-Dataset/
input <- readtext(url, text_field = "article") %>%
    corpus %>% corpus_subset(category != "")


model <- grafzahl(x = input,
                  y = "category",
                  model_name = "castorini/afriberta_base")
```

### Results

The final out-of-sample accuracy is 84.18%, a solid improvement from the baseline of 62.2%.

## Conclusion

This paper presents the R packages `grafzahl` and demonstrates its applicability to communication research by replicating the supervised machine learning part of published

[6] https://huggingface.co/datasets/israel/Amharic-News-Text-classification-Dataset/tree/main The direct url is https://huggingface.co/datasets/israel/Amharic-News-Text-classification-Dataset/resolve/main/train.csv

communication research. The Github repository of `grafzahl` provides additional examples.

## References

Azime, I. A., & Mohammed, N. (2021). An Amharic News Text classification Dataset. *arXiv Preprint arXiv:2103.05639*.

Baden, C., Pipal, C., Schoonvelde, M., & van der Velden, M. A. C. G. (2021). Three gaps in computational text analysis methods for social sciences: A research agenda. *Communication Methods and Measures*, *16*(1), 1–18. https://doi.org/10.1080/19312458.2021.2015574

Benoit, K., & Obeng, A. (2021). *Readtext: Import and Handling for Plain and Formatted Text Files*. Retrieved from https://CRAN.R-project.org/package=readtext

Benoit, K., Watanabe, K., Wang, H., Perry, P. O., Lauderdale, B., Gruber, J., & Lowe, W. (2021). *Quanteda.textmodels: Scaling Models and Classifiers for Textual Data*. Retrieved from https://CRAN.R-project.org/package=quanteda.textmodels

de Vries, W., Cranenburgh, A. van, Bisazza, A., Caselli, T., Noord, G. van, & Nissim, M. (2019). Bertje: A Dutch BERT model. *arXiv Preprint arXiv:1912.09582*.

Dobbrick, T., Jakob, J., Chan, C.-H., & Wessler, H. (2021). Enhancing theory-informed dictionary approaches with "glass-box" machine learning: The case of integrative complexity in social media comments. *Communication Methods and Measures*, 1–18. https://doi.org/10.1080/19312458.2021.1999913

Joshi, P., Santy, S., Budhiraja, A., Bali, K., & Choudhury, M. (2020). The state and fate of linguistic diversity and inclusion in the nlp world. *arXiv Preprint arXiv:2004.09095*.

Ogueji, K., Zhu, Y., & Lin, J. (2021). Small data? No problem! Exploring the viability of pretrained multilingual language models for low-resourced languages. *Proceedings of the 1st Workshop on Multilingual Representation Learning*, 116–126.

Pedersen, T. L., & Benesty, M. (2021). *Lime: Local interpretable model-agnostic explanations*. Retrieved from https://CRAN.R-project.org/package=lime

Rajapakse, T. (2022). *Simple Transformers*. Retrieved from https://simpletransformers.ai/

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why should i trust you?" Explaining the predictions of any classifier. *Proceedings of the 22nd Acm Sigkdd International Conference on Knowledge Discovery and Data Mining*, 1135–1144.

Ushey, K., Allaire, J., & Tang, Y. (2022). *reticulate: Interface to 'Python'*. Retrieved from https://CRAN.R-project.org/package=reticulate

Van Atteveldt, W., Van der Velden, M. A. C. G., & Boukes, M. (2021). The validity of sentiment analysis:Comparing manual annotation, crowd-coding, dictionary approaches, and machine learning algorithms. *Communication Methods and Measures*, 1–20. https://doi.org/10.1080/19312458.2020.1869198

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., … Rush, A. M. (2020). Transformers: State-of-the-art natural language processing. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 38–45. Online: Association for Computational Linguistics. Retrieved from https://www.aclweb.org/anthology/2020.emnlp-demos.6