Amsterdam University Press

COMPUTATIONAL COMMUNICATION RESEARCH 5.1 (2023) 76–85
HTTPS://DOI.ORG/10.5117/CCR2023.1.003.CHAN

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

Chung-hong Chan
GESIS - Leibniz-Institut für Sozialwissenschaften, Germany

Abstract
This paper introduces `grafzahl`, an R package for fine-tuning Transformers for text data from within R. The package is used in this paper to reproduce the analyses in other papers. Very significant improvement in model accuracy over traditional machine learning approaches such as convolutional Neural Network is observed.

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

Put the R back in Transformers

The purpose of this R package, `grafzahl`, is to provide the missing link between R 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, and Tang 2022) is seamless. With this seamless integration provided, communication researchers can produce the most advanced supervised learning models entirely from within R. This package provides the function `grafzahl()`, which emulates the behaviors of `quanteda.textmodels` (Benoit et al. 2021). [1]

Two examples (Van Atteveldt, Van der Velden, and Boukes 2021; Azime and Mohammed 2021) are presented here. Additional examples (Theocharis et al. 2020; Dobbrick et al. 2021; Çöltekin 2020) are available in the Github repository of the package (https://github.com/chainsawriot/grafzahl).

[1] This package uses reasonable default settings which suit what communication researchers would like to achieve with these 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.

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, Van der Velden, and Boukes (2021) show that the best method for predicting expert coding, other than coding by student helpers, is convolutional neural network (CNN) with Dutch word embeddings trained on Dutch news. The out-of-sample 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] and included in this package (as `ecosent`), I can provide a head-to-head comparison between the reported CNN and the Transformer-based model trained with `grafzahl`.

There are three important columns in the `ecosent` data frame:

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. It is in general not recommended to use this package without a CUDA-compatible GPU. Without a CUDA-compatible GPU, the fine-tuning processes below might take days, if not weeks.

If there is a GPU capable of performing CUDA, run:

```
## Github version
## remotes::install_github("chainsawriot/grafzahl")
install.packages("grafzahl") ## CRAN version
require(grafzahl)
setup_grafzahl(cuda = TRUE) # set to FALSE otherwise
detect_cuda()
```

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

If the automatic setup failed, one can also set up the miniconda environment manually to diagnose what went wrong. The complete instructions are available here: https://github.com/chainsawriot/grafzahl/wiki/setup_grafzahl.

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, the pretrained Dutch Transformer model BERTje should work (de Vries et al. 2019) [4]. The model name of BERTje is `GroNLP/bert-base-dutch-cased`. It is also important to note the citation information to properly cite the pretrained Transformer model.

Step 2: Create the corpus

The second step is to read the data as a corpus. [5]

```
require(readtext)
require(quanteda)
input <- corpus(ecosent, text_field = "headline")
```

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.

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

[4] Available from https://huggingface.co/GroNLP/bert-base-dutch-cased

[5] This step is not absolutely needed. The package can also work with character vectors. The `corpus` data structure is a better representation of character vector.

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

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

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.

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` (Kuhn 2008) can also be used to calculate standard performance metrics such as Precision, Recall, and F1 [6].

[6]The function `confusionMatrix()` can accept the predicted values and ground truth di-

```
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 Van Atteveldt, Van der Velden, and Boukes (2021), although the prediction accuracy for the neutral category is just on par. Van Atteveldt, Van der Velden, and Boukes (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, Van der Velden, and Boukes (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 [7]. 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.

Step 5: Explain the prediction

Unlike "glass-box" machine learning models (Dobbrick et al. 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, and Guestrin 2016; R implementation by Pedersen and 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

rectly, 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")`.

[7] The R code for generating the learning curves is available in the official repository: https://github.com/chainsawriot/grafzahl

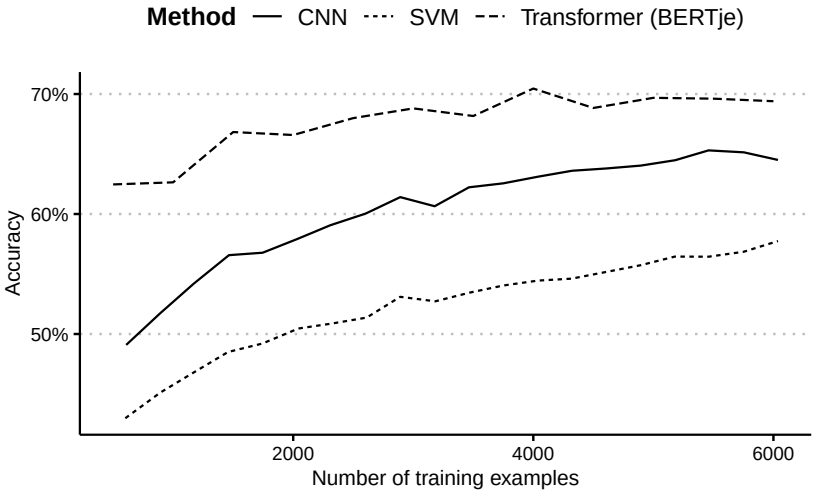

Figure 1: Learning curve of machine learning algorithms

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)
```

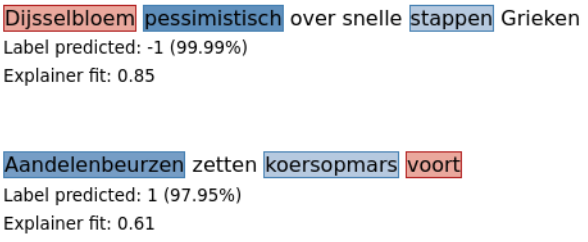

Figure 2: Generating Local Interpretable Model-agnostic Explanations (LIME) of two predictions from the trained Dutch sentiment model

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 et al. (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 et al. 2020) Only recently, the first news classification dataset called "Amharic News Text classification Dataset" is available (Azime and 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 [8]. In this example, the AfriBERTa is used as the pretrained model (Ogueji, Zhu, and 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.

```
input <- get_amharic_data()
model <- grafzahl(x = input$training,
```

[8] https://huggingface.co/datasets/israel/Amharic-News-Text-classification-Dataset

```
                    y = "category",
                    model_name = "castorini/afriberta_base")

## Calculate the out-of-sample accuracy

preds <- predict(model, newdata = input$test)
caret::confusionMatrix(table(preds, docvars(input$test,
                                            "category")))
```

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 communication research.

Acknowledgments

I would like to thank 1) Jarvis Labs for providing discounted GPU cloud service for the development of this package; 2) Pablo Barberá (University of Southern California) and Wouter van Atteveldt (VU Amsterdam) for allowing me to include their datasets in this package.

References

Azime, Israel Abebe, and Nebil Mohammed. 2021. "An Amharic News Text classification Dataset." *arXiv Preprint arXiv:2103.05639*.

Baden, Christian, Christian Pipal, Martijn Schoonvelde, and Mariken A. C. G van der Velden. 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, Kenneth, Kohei Watanabe, Haiyan Wang, Patrick O. Perry, Benjamin Lauderdale, Johannes Gruber, and William Lowe. 2021. *Quanteda.textmodels:*

Scaling Models and Classifiers for Textual Data. https://CRAN.R-project.org/package=quanteda.textmodels.

Çöltekin, Çağrı. 2020. "A corpus of Turkish offensive language on social media." In *Proceedings of the 12th Language Resources and Evaluation Conference*, 6174–84.

de Vries, Wietse, Andreas van Cranenburgh, Arianna Bisazza, Tommaso Caselli, Gertjan van Noord, and Malvina Nissim. 2019. "Bertje: A Dutch BERT model." *arXiv Preprint arXiv:1912.09582*.

Dobbrick, Timo, Julia Jakob, Chung-Hong Chan, and Hartmut Wessler. 2021. "Enhancing Theory-Informed Dictionary Approaches with 'Glass-Box' Machine Learning: The Case of Integrative Complexity in Social Media Comments." *Communication Methods and Measures*, November, 1–18. https://doi.org/10.1080/19312458.2021.1999913.

Joshi, Pratik, Sebastin Santy, Amar Budhiraja, Kalika Bali, and Monojit Choudhury. 2020. "The State and Fate of Linguistic Diversity and Inclusion in the NLP World." *arXiv Preprint arXiv:2004.09095*.

Kuhn, Max. 2008. "Building Predictive Models in R Using the Caret Package." *Journal of Statistical Software* 28 (5). https://doi.org/10.18637/jss.v028.i05.

Ogueji, Kelechi, Yuxin Zhu, and Jimmy Lin. 2021. "Small Data? No Problem! Exploring the Viability of Pretrained Multilingual Language Models for Low-Resourced Languages." In *Proceedings of the 1st Workshop on Multilingual Representation Learning*, 116–26.

Pedersen, Thomas Lin, and Michaël Benesty. 2021. *Lime: Local Interpretable Model-Agnostic Explanations*. https://CRAN.R-project.org/package=lime.

Rajapakse, Thilina. 2022. *Simple Transformers*. https://simpletransformers.ai/.

Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin. 2016. ""Why Should i Trust You?" Explaining the Predictions of Any Classifier." In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1135–44.

Theocharis, Yannis, Pablo Barberá, Zoltán Fazekas, and Sebastian Adrian Popa. 2020. "The Dynamics of Political Incivility on Twitter." *SAGE Open* 10 (2): 215824402091944. https://doi.org/10.1177/2158244020919447.

Ushey, Kevin, JJ Allaire, and Yuan Tang. 2022. *reticulate: Interface to 'Python'*. https://CRAN.R-project.org/package=reticulate.

Van Atteveldt, Wouter, Mariken A. C. G. Van der Velden, and Mark Boukes. 2021. "The Validity of Sentiment Analysis:comparing Manual Annota-

tion, Crowd-Coding, Dictionary Approaches, and Machine Learning Algorithms." *Communication Methods and Measures*, January, 1–20. https://doi.org/10.1080/19312458.2020.1869198.

Wolf, Thomas, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, et al. 2020. "Transformers: State-of-the-Art Natural Language Processing." In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 38–45. Online: Association for Computational Linguistics. https://www.aclweb.org/anthology/2020.emnlp-demos.6.