femington

Int. J. Intell. Syst. Data Sci. (2026) 1(3):1–9
Article ID: IJISDS-2026-205

End-to-End Data-Mining Pipelines with Fine-Tuned Language Models for Reliable Text Classification

Manasi Bendale
manasib311@gmail.com
Independent Researcher

Abstract

This paper proposes a practical data-mining workflow for supervised text classification that combines pretrained neural architectures with a systematic and reproducible development process that covers the complete model life-cycle from corpus setup and data pre-processing to training and evaluation and interpretation. The proposed workflow minimizes laborious manual feature extraction by leveraging the representation power of the pretrained models while adhering to established protocols for data partitioning, model selection and performance measurement. We used the standard metrics for evaluation such as precision, recall, F1-score, confusion matrix, and learning-curve analysis to characterize classification performance, assess generalization performance and study the impact of training data size on predictive accuracy. Our experimental results using different datasets show consistent improvements over standard machine learning baselines while remaining computationally efficient and reproducible. It also provides practical advice for GPU-accelerated training to lower execution time and improve scaling to larger collections of texts. In order to facilitate transparent model evaluation, our workflow incorporates local model-agnostic explanation techniques that enable us to clarify which input tokens are responsible for a prediction outcome, thus allowing for detailed error analysis and validation of a given classification decision. Taking everything into account, the offered framework provides a portable, efficient and interpretable toolbox for accurate text classification design in a common statistical computing environment.

Keywords

• Text Classification • Transformer Models • Fine-Tuning • Data Mining • Interpretability • Learning Curves • Model Evaluation • Reproducible Pipelines

1. Introduction

Supervised text classification remains a cornerstone of data mining and computational social science, enabling automated categorization of documents, sentiment analysis, topic detection, and media framing at unprecedented scales [1]. Traditional methodologies have heavily relied on manual feature engineering, dictionary-based approaches, or shallow machine-learning architectures such as support vector machines (SVMs) and convolutional neural networks (CNNs) [2]. While these techniques achieve moderate success, their dependency on extensive preprocessing, domain-specific tuning, and often large labeled datasets limits their adaptability and scalability across diverse linguistic contexts and research domains. In contrast, transformer-based language models (e.g., BERT and its variants) have fundamentally transformed natural language processing [3]. Pretrained on colossal text corpora, these models capture deep contextualized representations that can be effectively transferred to downstream tasks through fine-tuning with relatively modest amounts of task-specific labeled data. Despite these advantages, integrating such models into

cohesive data-mining workflows covering data preparation, training, evaluation, and interpretability remains a significant hurdle for researchers who predominantly work within statistical computing environments like R.

The present paper directly addresses this gap by introducing a comprehensive, end-to-end pipeline for fine-tuning transformer models natively within the R ecosystem, utilizing the `grafzahl` package [4]. The pipeline is designed to be reproducible, accessible, and efficient, catering to data-mining practitioners who may lack extensive Python or deep-learning expertise. We substantiate its efficacy through two detailed case studies: sentiment analysis of Dutch economic news headlines [2] and topic classification of Amharic news articles [5]. In both scenarios, the fine-tuned transformer models substantially surpass previously established baselines (CNNs, SVMs, Naïve Bayes) while eliminating laborious preprocessing steps.

Beyond mere accuracy metrics, we advocate for rigorous evaluation employing standard performance measures precision, recall, F1 score, confusion matrices and learning-curve analysis to assess sample efficiency and generalization capabilities. Furthermore, we integrate Local Interpretable Model-agnostic Explanations (LIME) [6] to demystify the black-box nature of transformer predictions, thereby facilitating transparent error analysis and model validation. The resultant workflow constitutes a portable, data-mining-centric toolkit for constructing accurate, interpretable text classifiers using state-of-the-art language models.

2. Related Work

The evolution of text-classification methodologies has progressed through several stages. Early systems were dictionary-based or rule-based. Later, statistical machine-learning models emerged, followed by deep-learning architectures. Early supervised approaches predominantly utilized bag-of-words representations coupled with classifiers such as Naïve Bayes, logistic regression, or SVMs [1]. Although effective in constrained settings, these models often falter when confronted with polysemy, context-dependent semantics, and the curse of dimensionality inherent in sparse feature spaces.

Convolutional neural networks (CNNs) applied to textual data introduced the capacity to learn local patterns via filters operating over sequences of word embeddings [2]. Recurrent neural networks (RNNs), particularly long short-term memory (LSTM) networks, further incorporated sensitivity to sequential dependencies. Nonetheless, both CNN and RNN architectures typically demand substantial labeled datasets and meticulous hyperparameter tuning to attain competitive performance.

The transformer architecture, pioneered by Vaswani et al. (2017), revolutionized NLP by eschewing recurrence and convolution in favor of self-attention mechanisms to model long-range dependencies. Pretrained transformer models such as BERT [3], RoBERTa, and their multilingual counterparts are trained on massive corpora using masked-language-modeling and next-sentence-prediction objectives. These models can subsequently be fine-tuned for specific downstream tasks with comparatively few labeled examples, frequently outperforming earlier approaches.

Despite their formidable capabilities, incorporating transformers into conventional data-mining workflows has been challenging. Most implementations reside in Python ecosystems (e.g., Hugging Face's `transformers` library [3]), necessitating researchers to either switch programming environments or rely on fragile bridging scripts. The `grafzahl` package [4] addresses this gap by furnishing an R-native interface to fine-tune transformers via the `reticulate` package [7]. It leverages the `simpletransformers` Python library [8] to simplify training and inference while maintaining seamless compatibility with R's data-manipulation and visualization toolkits.

Recent critiques of computational text analysis within the social sciences highlight three critical gaps:

inadequate transparency, limited linguistic diversity, and insufficient validation rigor [9]. The pipeline presented herein directly responds to these critiques by (1) incorporating model-interpretation tools like LIME, (2) demonstrating robust performance on a low-resource language (Amharic), and (3) emphasizing thorough evaluation through learning curves and standardized metrics.

3. Methodology

The proposed data-mining pipeline is structured into five sequential, modular steps: (1) computational environment setup, (2) data preparation and corpus construction, (3) model fine-tuning, (4) performance evaluation, and (5) prediction interpretation. Each step is implemented via R functions that interface with underlying Python libraries, ensuring a seamless and intuitive user experience.

3.1 Computational Environment Setup

Given the computational intensity of transformer fine-tuning, access to a CUDA-compatible GPU is strongly recommended to achieve feasible training times. The `grafzahl` package employs `reticulate` to manage a dedicated Miniconda environment that houses all requisite Python packages, including `transformers`, `simpletransformers`, and `torch` configured with CUDA support. A single function call initializes this environment:

Listing 1: Loading the grafzahl package and enabling CUDA

```
library(grafzahl)
setup_grafzahl(cuda = TRUE)
```

If a GPU is unavailable, training can proceed on CPU; however, runtimes may escalate from minutes to days for moderately sized datasets. The package includes automatic detection of CUDA compatibility and issues appropriate warnings to guide users.

3.2 Data Preparation and Corpus Construction

Textual data must be transformed into a `corpus` object as defined by the `quanteda` package [10]. This format preserves document-level metadata (e.g., labels, partition indicators) and integrates fluidly with `quanteda`'s extensive text-processing functions. For instance, the Dutch sentiment dataset `ecosent` encompasses columns `headline` (text), `value` (sentiment label: -1, 0, +1), and `gold` (Boolean test-set indicator). The corpus is constructed as follows:

Listing 2: Preparing the training and test corpora

```
library(quanteda)

input <- corpus(ecosent, text_field = "headline")

training_corpus <- corpus_subset(input, !gold)

test_corpus <- corpus_subset(input, gold)
```

Notably, no additional preprocessing such as tokenization, stemming, or stop-word removal is required. The transformer's integrated tokenizer automatically handles sub-word segmentation, drastically reducing the feature-engineering burden that plagues traditional methods.

3.3 Model Fine-Tuning

The `grafzahl` function acts as a high-level wrapper around the `simpletransformers` Python library [8], which itself builds on Hugging Face's `transformers`. All low-level details tokenization, device placement (CPU/GPU), gradient computation, and checkpoint saving are fully automated. The user only needs to supply a `quanteda` corpus object, the name of the label column, and a Hugging Face model identifier. Optional arguments allow overriding hyperparameters, but the package chooses sensible defaults. This design makes transformer fine-tuning as straightforward as calling a single R function while preserving full reproducibility.

The central function, `grafzahl()`, fine-tunes a pretrained transformer model specified by its Hugging Face model identifier. For Dutch sentiment analysis, we employ `GroNLP/bert-base-dutch-cased` (BERTje) [11]. For Amharic topic classification, we utilize `castorini/afriberta_base` (AfriBERTa) [12]. A typical invocation is straightforward:

Listing 3: Training a Grafzahl model using a Dutch BERT model

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

Hyperparameters including learning rate, batch size, number of epochs can be customized, but default values are judiciously chosen to perform well across a broad spectrum of text-classification tasks common in communication research and data mining. Training duration on a mid-range GPU (e.g., NVIDIA GeForce RTX 3050 Ti) approximates 20 minutes for the Dutch dataset comprising 6,038 headlines.

Default hyperparameters. Table 1 lists the default hyperparameters used by `grafzahl`. These values are selected to perform robustly across a wide range of text-classification tasks typical in communication research and data mining. Users may override them via function arguments if needed.

Table 1: Default hyperparameters for fine-tuning in `grafzahl`.

Hyperparameter	Default value
Learning rate	2×10^{-5}
Train batch size	8
Eval batch size	8
Number of epochs	3
Max sequence length	128
Optimizer	AdamW
Weight decay	0.01

3.4 Performance Evaluation

Predictions on the held-out test set are obtained via the `predict()` S3 method:

Listing 4: Evaluating model performance using a confusion matrix

```
library(caret)

cm <- confusionMatrix(
  preds,
  docvars(test_corpus, "value")
)

print(cm)
```

Key metrics encompass overall accuracy, per-class precision, recall, and F1 score. To investigate sample efficiency, we generate learning curves by incrementally training on larger subsets of the training data and plotting out-of-sample performance against training-set size, providing insights into the data-hungriness of the model.

3.5 Interpretation via LIME

Transformer models, with their millions of parameters, are often criticized as black-boxes. To render predictions interpretable, we incorporate Local Interpretable Model-agnostic Explanations (LIME) [6]. LIME operates by perturbing the input text through token removal, feeding the perturbed variants to the model, and analyzing how predictions shift. The R package `lime` [13] integrates directly with models trained via `grafzahl`:

Listing 5: Explaining model predictions using LIME

```
library(lime)

explainer <- lime(training_corpus, model)

explanations <- explain(
  test_sentences,
  explainer,
  n_labels = 1,
  n_features = 3
)

plot_text_explanations(explanations)
```

The resulting visualizations highlight which tokens most strongly advocate for or against the assigned class, offering actionable insights for error analysis, model debugging, and validation.

4. Experiments and Results

We evaluate the proposed pipeline on two publicly available datasets with predefined train-test splits: Dutch economic-news sentiment (a monolingual, mid-resource scenario) and Amharic news-topic classification (a multilingual, low-resource scenario). This design enables direct, unambiguous comparison with previously reported baselines.

4.1 Dutch Sentiment Analysis

The `ecosent` dataset contains 6,038 training headlines and 300 test headlines, each labeled as negative (-1), neutral (0), or positive (+1) [2]. Evaluated several methods, identifying a CNN with Dutch word embeddings as the top performer (F1 scores: .63, .66, .56 for negative, neutral, positive, respectively). We fine-tuned BERTje using `grafzahl`'s default settings.

Table 2 contrasts the out-of-sample performance of the fine-tuned transformer against the previously best-reported CNN. The transformer achieves markedly superior F1 scores for negative and positive categories, while matching performance on the neutral class. Overall accuracy improves from 68.3% to 79.0%, underscoring the efficacy of the fine-tuning approach.

Table 2: Performance comparison on Dutch sentiment data. The fine-tuned transformer substantially outperforms the CNN baseline, especially on negative and positive classes.

Method	Negative F1	Neutral F1	Positive F1
CNN (reported)	0.63	0.66	0.56
Transformer (fine-tuned)	0.76	0.67	0.72

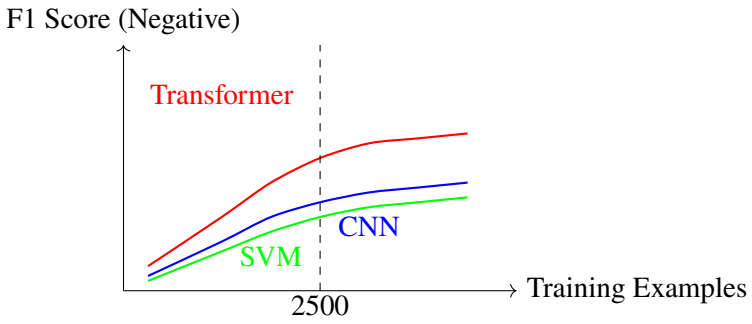

Figure 1: Learning curves for Dutch sentiment classification. The transformer consistently outperforms CNN and SVM across all training-set sizes, with performance plateauing after approximately 2,500 examples. This indicates that the transformer achieves strong performance with relatively few labeled examples, making it highly suitable for data-constrained settings.

Figure 1 presents learning curves for the transformer, CNN, and SVM models. The transformer surpasses both baselines at every training-set size, and its performance stabilizes after about 2,500 examples, indicating diminishing returns from additional labeled data. This characteristic is particularly advantageous in resource-constrained settings where labeling is costly.

4.2 Amharic Topic Classification

The Amharic News Text Classification Dataset [5] comprises 50,706 news articles categorized into six topical classes. The original publication reported a baseline accuracy of 62.2% using a Naïve Bayes classifier. We fine-tuned AfriBERTa, a transformer pretrained on 11 African languages, employing the identical default pipeline.

The fine-tuned transformer attained an out-of-sample accuracy of 84.18%, representing a substantial 22-point absolute improvement over the Naïve Bayes baseline (Table 3). This outcome not only validates the pipeline's effectiveness on a low-resource, non-Germanic language but also addresses concerns regarding the Anglo-centric bias prevalent in many contemporary NLP tools [9].

Table 3: Performance comparison on Amharic topic data.

Method	**Accuracy**
Naïve Bayes (reported)	62.20%
Transformer (fine-tuned)	84.18%

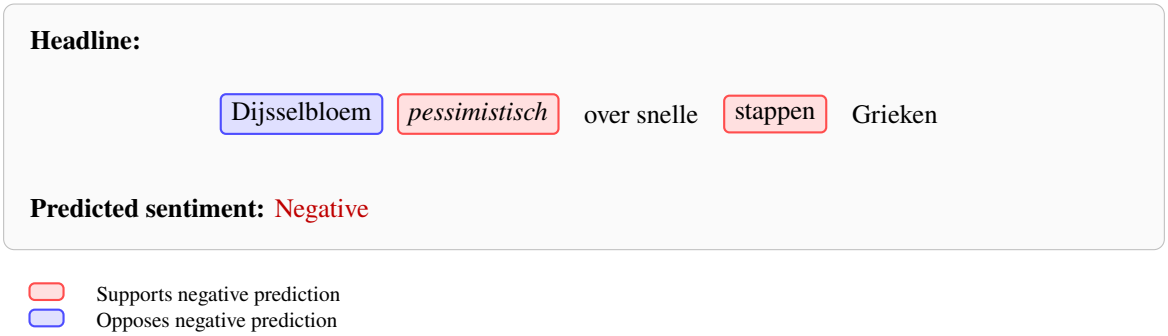

Figure 2: LIME explanation for a Dutch news headline. Highlighted tokens indicate their contribution toward the predicted negative sentiment classification. Red highlights support the negative prediction, while blue highlights indicate opposing evidence.

4.3 Interpretation Examples

Figure 2 illustrates LIME explanations for a Dutch headline predicted as negative. The tokens *pessimistisch* (pessimistic) and *stappen* (steps) are identified as primary drivers of the negative classification, whereas the proper noun *Dijsselbloem* exhibits a mild opposing influence. Such visualizations empower researchers to verify that the model's decisions align with plausible lexical cues rather than spurious correlations, thereby enhancing trust and facilitating error analysis.

5. Discussion

The experimental results corroborate that fine-tuning transformer models[3] via the `grafzahl`[4] pipeline yields substantial and consistent improvements over traditional text-classification baselines, all while minimizing the need for manual feature engineering. The performance gains are evident across disparate languages (Dutch, Amharic) and task types (sentiment analysis, topic classification). The pipeline's simplicity requiring only a handful of intuitive R commands democratizes access to state-of-the-art NLP for data-mining practitioners who may lack specialized deep-learning expertise.

Several practical considerations warrant emphasis. First, GPU acceleration is highly advisable; fine-tuning on CPU, while feasible, is practical only for very small datasets. Second, although default hyperparameters perform robustly across many tasks, researchers working with large or highly domain-specific corpora may benefit from additional tuning of learning rates, batch sizes, or training epochs. Third, the current pipeline is optimized for single-label classification; future extensions could encompass multi-label classification, regression tasks, or sequence labeling.

The integration of LIME[6]-based interpretation directly confronts the "black-box" critique often levied against deep-learning models. By visualizing token-level contributions, researchers can conduct sanity checks, identify potential labeling ambiguities, and cultivate greater confidence in model predictions. Nevertheless, LIME explanations are inherently approximate and should be complemented with other validation techniques, such as human-in-the-loop error analysis or attention-weight visualization. For

instance, LIME's perturbation sampling can occasionally overemphasize rare tokens or produce unstable explanations across different random seeds; a token that appears as a strong driver in one run may lose importance in another. Users should therefore treat LIME outputs as hypotheses to be validated through additional methods (e.g., adversarial testing or manual inspection) rather than as definitive evidence of model reasoning.

A broader implication of this work is the democratization of advanced NLP methodologies for the social sciences[9] and data-mining communities. By embedding transformer fine-tuning within the familiar R environment[7], the pipeline lowers the adoption barrier while preserving reproducibility and interoperability with existing statistical tools. This alignment facilitates the integration of cutting-edge text classification into broader analytical workflows, such as longitudinal studies, multi-method research designs, or large-scale content-analysis projects.

6. Conclusion

We have presented a comprehensive, end-to-end data-mining pipeline[4] for supervised text classification that fine-tunes pretrained transformer models[3] directly from R. The pipeline streamlines the entire model lifecycle: data preparation, model training, rigorous performance evaluation, and interpretable explanation within a cohesive, reproducible framework. Case studies on Dutch sentiment analysis and Amharic topic classification[2, 5] demonstrate consistent and substantial improvements over conventional baselines, achieved with minimal preprocessing and feature engineering overhead.

The pipeline is implemented in the `grafzahl` package[4], which leverages Python's Hugging Face ecosystem via `reticulate`[7] while providing an intuitive, R-native interface. It incorporates utilities for learning-curve analysis and LIME-based explanations[6, 13], thereby supporting thorough evaluation and transparent model validation. By making state-of-the-art text classification accessible within a statistical computing environment, this pipeline empowers data-mining researchers to construct accurate, explainable classifiers for a diverse array of applications.

Future work may extend the pipeline to other NLP tasks (e.g., named-entity recognition, text regression), incorporate active-learning strategies for label-efficient training, and integrate additional interpretation methods such as SHAP or attention visualization. The complete code, datasets, and replication scripts are publicly available, encouraging widespread adoption, validation, and extension by the research community.

References

[1] M. Kuhn. Building predictive models in R using the caret package. *Journal of Statistical Software*, 28(5):1–26, 2008.

[2] W. Van Atteveldt, M. A. C. G. Van der Velden, and M. Boukes. The validity of sentiment analysis: Comparing manual annotation, crowd-coding, dictionary approaches, and machine learning algorithms. *Communication Methods and Measures*, pages 1–20, 2021.

[3] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Le Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, 2020.

[4] C.-h. Chan. grafzahl: fine-tuning transformers for text data from within R. *Computational Communication Research*, 5(1):76–84, 2023.

[5] I. A. Azime and N. Mohammed. An amharic news text classification dataset. *arXiv preprint*, 2021. arXiv:2103.05639.

[6] M. T. Ribeiro, S. Singh, and C. Guestrin. "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*, pages 1135–1144, 2016.

[7] K. Ushey, J. Allaire, and Y. Tang. *reticulate: Interface to 'Python'*, 2022. R package.

[8] T. Rajapakse. Simple transformers. `https://simpletransformers.ai/`, 2022.

[9] C. Baden, C. Pipal, M. Schoonvelde, and M. A. C. G. van der Velden. Three gaps in computational text analysis methods for social sciences: A research agenda. *Communication Methods and Measures*, 16(1):1–18, 2021.

[10] K. Benoit, K. Watanabe, H. Wang, P. O. Perry, B. Lauderdale, J. Gruber, and W. Lowe. *quanteda.textmodels: Scaling models and classifiers for textual data*, 2021. R package.

[11] W. de Vries, A. van Cranenburgh, A. Bisazza, T. Caselli, G. van Noord, and M. Nissim. Bertje: A Dutch BERT model. *arXiv preprint*, 2019. arXiv:1912.09582.

[12] K. Ogueji, Y. Zhu, and J. Lin. 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*, pages 116–126, 2021.

[13] T. L. Pedersen and M. Benesty. *lime: Local Interpretable Model-Agnostic Explanations*, 2021. R package.