NewsRecLib:
A PyTorch-Lightning Library for Neural News Recommendation

Andreea Iana[1], Goran Glavaš[2], Heiko Paulheim[1]
[1] Data and Web Science Group, University of Mannheim, Germany
[2] Center For Artificial Intelligence and Data Science, University of Würzburg, Germany
{andreea.iana, heiko.paulheim}@uni-mannheim.de
goran.glavas@uni-wuerzburg.de

Abstract

NewsRecLib[1] is an open-source library based on Pytorch-Lightning and Hydra developed for training and evaluating neural news recommendation models. The foremost goals of NewsRecLib are to promote *reproducible research* and *rigorous experimental evaluation* by (i) providing a unified and highly configurable framework for exhaustive experimental studies and (ii) enabling a thorough analysis of the performance contribution of different model architecture components and training regimes. NewsRecLib is highly modular, allows specifying experiments in a single configuration file, and includes extensive logging facilities. Moreover, NewsRecLib provides out-of-the-box implementations of several prominent neural models, training methods, standard evaluation benchmarks, and evaluation metrics for news recommendation.

1 Introduction

Personalized news recommendation has become ubiquitous for customizing suggestions to users' interests (Li and Wang, 2019; Wu et al., 2023). In recent years, there has been a surge of effort towards neural content-based recommenders. With increasingly complex neural architectures able to ever more precisely capture users' content-based preferences, neural recommenders quickly replaced traditional recommendation models as the go-to paradigm for news recommendation.

Despite the abundance of model designs, research on neural news recommenders (NNRs) suffers from two major shortcomings: (i) a surprising amount of non-reproducible research (Ferrari Dacrema et al., 2021) and (ii) unfair model comparisons (Ferrari Dacrema et al., 2019; Sun et al., 2020). The former is, on the one hand, due to many NNR implementations not being publicly released (Sertkan and Neidhardt, 2022). Existing open source repositories, on the other hand, expose a multitude of programming languages, libraries, and implementation differences, hindering reproducibility and extensibility (Said and Bellogín, 2014). Moreover, a lack of transparency in terms of evaluation datasets, experimental setup and hyperparameter settings, as well as the adoption of ad-hoc evaluation protocols, further severely impede direct model comparisons. Many personalized news recommenders have been evaluated on proprietary datasets (e.g., Bing News (Wang et al., 2018), MSN News (Wu et al., 2019a,d), News App (Qi et al., 2022)). Even the models trained on the more recently introduced open benchmarks (e.g., Adressa (Gulla et al., 2017), MIND (Wu et al., 2020b)) cannot be directly compared due to the lack of standard dataset splits and evaluation protocols (Wu et al., 2021; Zhang et al., 2021; Gong and Zhu, 2022; Wang et al., 2022). Even more concerning, crucial details regarding the setup of the experiments are regularly omitted from the publications or hard-coded without explanation.

It is thus particularly difficult to evaluate the impact of specific components in NNR architecture and training (e.g., news encoder, user modeling, training objectives) on the overall performance of the model (Iana et al., 2023a). Many models simultaneously change multiple components in both the news and the user encoder, while carrying out only partial ablation studies or evaluating against suboptimal baselines (Rendle et al., 2019).

In this work, we introduce NewsRecLib, an open source library for NNRs, to remedy these critical limitations.[2] NewsRecLib aims to facilitate reproducible research and comprehensive experimental studies, using an end-to-end pipeline powered by a single configuration file that specifies a complete experiment – from dataset selection and preprocessing over model architecture and training to evaluation protocol and metrics. NewsRecLib is

[1] https://github.com/andreeaiana/newsreclib

[2] The library is licensed under a MIT license.

Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 296–310
December 6-10, 2023 ©2023 Association for Computational Linguistics

built based on the following guiding principles:

Modularity and extensibility. With PyTorch Lightning (Falcon and The PyTorch Lightning team, 2019) as its backbone, NewsRecLib is designed in a modular fashion, with core individual components being decoupled from one another. This enables mixing and matching different modules, as well as seamlessly integrating new ones.

Easy configurability and reproducibility. NewsRecLib is powered by Hydra (Yadan, 2019), in which each experiment is defined through a single configuration file composed from the configurations of specific pipeline components. The configuration of every experiment is automatically stored at the start of the run and as such trivially enables reproducibility.

Logging and profiling. The library supports multiple standard tools (e.g., WandB (Biewald, 2020), Tensorboard (Abadi et al., 2016)) for extensive logging, monitoring, and profiling of experiments with neural models – in terms of losses, evaluation metrics, runtime, memory usage, and model size.

Overall, NewsRecLib is designed to support the development and benchmarking of NNRs as well as the specific analysis of contributions of common components of the neural recommendation pipelines. In this paper, we discuss the building blocks of NewsRecLib and provide an overview of the readily available models. For a detailed documentation on the usage of the library, we refer to its project page.

2 NewsRecLib – the Library

Figure 1 depicts the structure of NewsRecLib, comprising different functional modules: from data modules for downloading and processing datasets to recommendation modules for training and evaluating a particular NNR. The overall pipeline of an experiment is built automatically from the high-level experimental flow provided by the user in the form of a single Hydra configuration file.

2.1 Modularization and Extensiblity

NewsRecLib is highly modularized: it decouples core components to the largest extent possible. This allows for combinations of different news encoders (e.g., over different input features – text, aspects, entities) with different user modeling techniques, click fusion strategies, and training objectives. NewsRecLib is easily extensible with new features: the user only needs to write a new subcomponent class (e.g., category encoder), or, in the case of new datasets or recommenders, to define a new PyTorch Lightning data module or (model) module, respectively.

Concretely, we decouple the essential building blocks of a NNR, namely the *news encoder* (NE), the *user encoder* (UE), and the *click predictor*. NE is further decomposed into a configurable set of feature encoders (i.e., components that embed different aspects of the news, e.g., title, topical category or named entities). Different model components can be interchanged with corresponding sub-modules of other recommenders, ensuring freedom in choosing each building block of a model independently of the other components (i.e., by mixing the NE of "NNR 1" with the UE of "NNR 2"), in contrast to practices in existing NNR libraries, in which sub-components are tied to concrete NNR architectures that introduced them. Because of this, NewsRecLib allows for clear-cut and comprehensive analyses of impact of NNR components on their overall performance.[3] NewsRecLib currently implements feature encoders used in pre-implemented models (see Appendix §B); users can, however, easily incorporate new ones (e.g., an image encoder) by extending the respective class.

2.2 Configurability and Reproducibility

Reproducibility strongly relies on the transparency of each step and component in the pipeline, as well as the availability of metadata regarding the factors that influence the model (e.g., hyperparameter values, training objective) and the environment in which it is trained and evaluated (e.g., library versions). Because of this, NewsRecLib leverages the Hydra[4] framework (Yadan, 2019) to decouple the experiment configuration (i.e., a pipeline of modules) from the concrete implementations (i.e., source code) of the modules.

Each concrete module setting is specified and retrieved automatically from a dedicated configuration file which can be accessed by all the pipeline components. A variety of callbacks supported by PyTorch Lightning (e.g., model checkpointing, early stopping, debugging) can be defined, and modified via a corresponding configuration. A single configuration file guides each experiment:

[3] E.g., we leveraged an earlier version of NewsRecLib to analyze the impact of click behavior fusion strategies and training objectives on NNRs' performance (Iana et al., 2023a).

[4] https://hydra.cc/

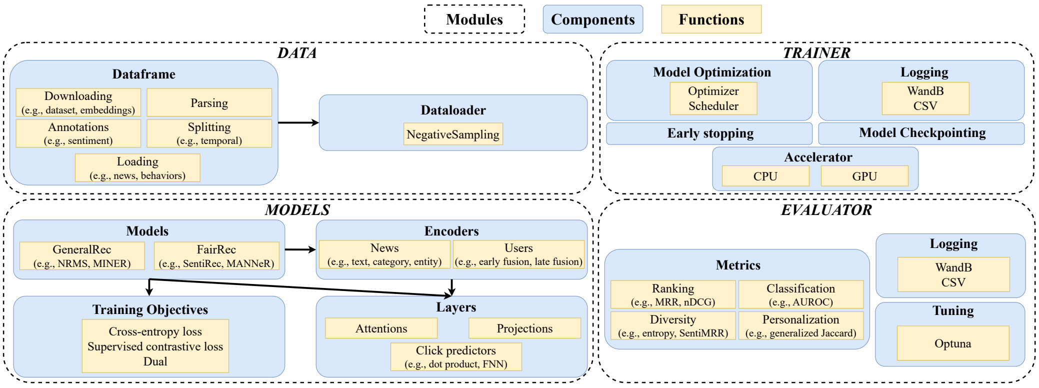
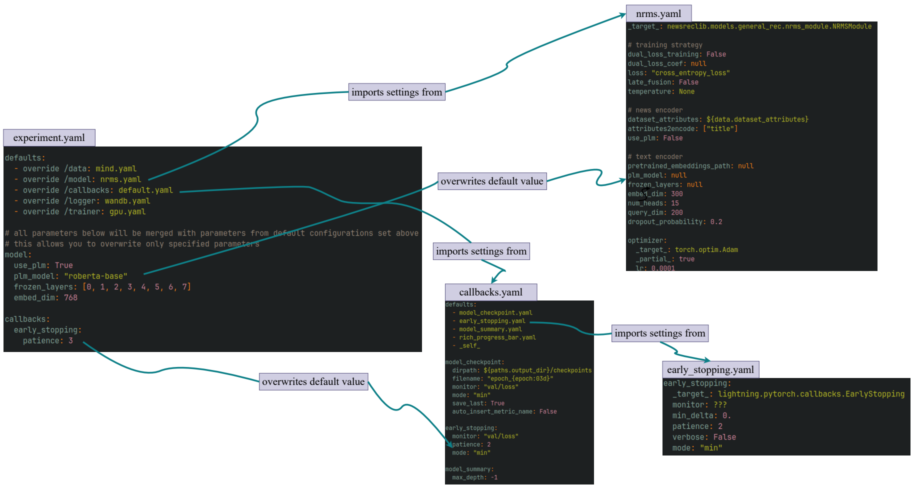

Modules
Components
Functions
DATA
Dataframe
Downloading (e.g., dataset, embeddings)
Parsing
Annotations (e.g., sentiment)
Splitting (e.g., temporal)
Loading (e.g., news, behaviors)
Dataloader
NegativeSampling
TRAINER
Model Optimization
Optimizer
Scheduler
Logging
WandB
CSV
Early stopping
Model Checkpointing
Accelerator
CPU
GPU
MODELS
Models
GeneralRec (e.g., NRMS, MINER)
FairRec (e.g., SentiRec, MANNeR)
Encoders
News (e.g., text, category, entity)
Users (e.g., early fusion, late fusion)
Training Objectives
Cross-entropy loss
Supervised contrastive loss
Dual
Layers
Attentions
Projections
Click predictors (e.g., dot product, FNN)
EVALUATOR
Metrics
Ranking (e.g., MRR, nDCG)
Classification (e.g., AUROC)
Diversity (e.g., entropy, SentiMRR)
Personalization (e.g., generalized Jaccard)
Logging
WandB
CSV
Tuning
Optuna

Figure 1: Illustration of the NewsRecLib framework.



```yaml
defaults:
  - override /data: mind.yaml
  - override /model: nrms.yaml
  - override /callbacks: default.yaml
  - override /logger: wandb.yaml
  - override /trainer: gpu.yaml

# all parameters below will be merged with parameters from default configurations set above
# this allows you to overwrite only specified parameters
model:
  use_plm: True
  plm_model: "roberta-base"
  frozen_layers: [0, 1, 2, 3, 4, 5, 6, 7]
  embed_dim: 768

callbacks:
  early_stopping:
    patience: 3
```

```yaml
_target_: newsreclib.models.general_rec.nrms_module.NRMSModule

# training strategy
dual_loss_training: False
dual_loss_coef: null
loss: "cross_entropy_loss"
late_fusion: False
temperature: None

# news encoder
dataset_attributes: ${data.dataset_attributes}
attributes2encode: ["title"]
use_plm: False

# text encoder
pretrained_embeddings_path: null
plm_model: null
frozen_layers: null
embed_dim: 300
num_heads: 15
query_dim: 200
dropout_probability: 0.2

optimizer:
  _target_: torch.optim.Adam
  _partial_: true
  lr: 0.0001
```

```yaml
defaults:
  - model_checkpoint.yaml
  - early_stopping.yaml
  - model_summary.yaml
  - rich_progress_bar.yaml
  - _self_

model_checkpoint:
  dirpath: ${paths.output_dir}/checkpoints
  filename: "epoch_{epoch:03d}"
  monitor: "val/loss"
  mode: "min"
  save_last: True
  auto_insert_metric_name: False

early_stopping:
  monitor: "val/loss"
  patience: 2
  mode: "min"

model_summary:
  max_depth: -1
```

```yaml
early_stopping:
  _target_: lightning.pytorch.callbacks.EarlyStopping
  monitor: ???
  min_delta: 0.
  patience: 2
  verbose: False
  mode: "min"
```

Figure 2: A minimal configuration example for training an NRMS (Wu et al., 2019d) model. All settings defined in the main and the imported configuration files are merged and persisted into a single configuration object.

the default configurations of the used modules and callbacks are hierarchically inherited and can be overridden. Experiment configurations can also be overwritten directly from the command line, removing the need to store many similar configuration files: this facilitates fast experimentation and minimizes boilerplate code. Experiments can be executed on CPU, GPU, and in a distributed fashion by specifying the type of accelerator supported in PyTorch Lightning. The integration with extensive logging capabilities (see §2.3) ensures that any modifications are persistently stored in the experiment directory, together with other log files and model checkpoints.

Fig. 2 shows a minimal configuration example for an experiment that trains an instance of the NRMS (Wu et al., 2019d) model. The main configuration file experiment.yaml guides the pipeline. It inherits the data and model-specific configurations from mind.yaml and nrms.yaml, which specify the default configurations of the data module and NNR model, respectively. experiment.yaml further uses the default configurations for the WandB logger, the trainer, and various callbacks. The example also illustrates the interplay between modularization and configurability: we replace the original NE of the NRMS model with a pretrained language model (in this case roberta-base).

2.3 Performance Evaluation and Profiling

With Hydra's pluggable architecture as its backbone, every part of the recommendation pipeline is transparent to the user. NewsRecLib records comprehensive information during training, including number of trainable model parameters and total model size, runtimes, training and validation losses. Moreover, it stores important metadata regarding hyperparameter settings, operating system, PyTorch version, environment details, and dependencies between libraries. Any profiler supported by PyTorch can be incorporated by a simple modification of the corresponding configuration file.

NewsRecLib supports widely used loggers like WandB[5] (Biewald, 2020) and Tensorboard[6] (Abadi et al., 2016). Moreover, users can export evaluation metrics for further analysis. Appendix A shows an example of the logging output. We rely on TorchMetrics[7] (Detlefsen et al., 2022) for model evaluation. Users can track numerous metrics ranging from accuracy-based to beyond-accuracy (e.g., diversity) performance. New metrics can be easily added to the pipeline, either by defining the necessary callbacks in the case of metrics already available in TorchMetrics, or by implementing a custom metric as a subclass of the base `Metric` class in TorchMetrics.

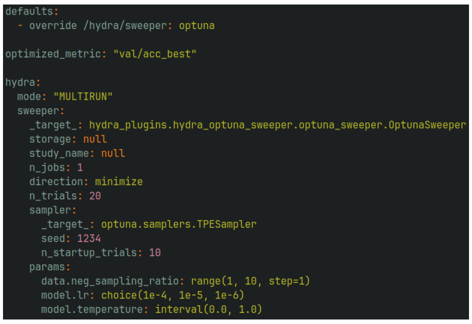
2.4 Hyperparameter Optimization

NNR performance heavily depends on model hyperparameters, making hyperparameter optimization a crucial ingredient in the empirical evaluations of NNRs. NewsRecLib supports hyperparameter tuning using the Optuna framework (Akiba et al., 2019), which offers a wide range of samplers, such as random search, grid search, and Bayesian optimization (Bergstra et al., 2011; Ozaki et al., 2020).[8] In conjunction with the modularity of NewsRecLib, this allows nearly every component of a news recommender to be treated as a hyperparameter, so that users can optimize the choice of encoders or scoring functions. Figure 3 shows a basic multi-objective hyperparameter search over the number of negative samples, the model's learning rate, and temperature for the supervised contrastive loss.

2.5 Available Modules

NewsRecLib currently encompasses two popular benchmark datasets, 13 news recommendation models, and various evaluation metrics.

Datasets. We provide out-of-the-box utilities for two prominent monolingual news recommendation benchmarks: MIND (Wu et al., 2020b) (with English news) and Adressa (Gulla et al., 2017) (with Norwegian news). For both datasets, NewsRecLib supports automatic downloading (when available)[9], data parsing, and pre-processing functionalities to create a unified PyTorch Lightning datamodule. For both datasets, we include their small and large versions, MINDsmall and MINDlarge, and Adressa-1 week and 10 weeks, respectively.



```yaml
defaults:
  - override /hydra/sweeper: optuna

optimized_metric: "val/acc_best"

hydra:
  mode: "MULTIRUN"
  sweeper:
    _target_: hydra_plugins.hydra_optuna_sweeper.optuna_sweeper.OptunaSweeper
    storage: null
    study_name: null
    n_jobs: 1
    direction: minimize
    n_trials: 20
    sampler:
      _target_: optuna.samplers.TPESampler
      seed: 1234
      n_startup_trials: 10
    params:
      data.neg_sampling_ratio: range(1, 10, step=1)
      model.lr: choice(1e-4, 1e-5, 1e-6)
      model.temperature: interval(0.0, 1.0)
```

Figure 3: Example of a hyperparameter optimization process. The configuration first runs 10 trials of a search using Bayesian optimization. The hyperparameter search space is defined by indicating the interval, range or choice of values for each desired parameter.

Since Wu et al. (2020b) do not publicly release test labels for MIND, we use the provided validation portion for testing, and split the respective training set into temporally disjoint training and validation portions. We follow established practices on splitting the Adressa dataset (Hu et al., 2020; Xu et al., 2023) into train, validation, and test sets. In contrast to MIND, which consists of impression log (lists of clicked and non-clicked news by the user), the Adressa dataset contains only positive samples (Gulla et al., 2017). Following Yi et al. (2021), we build impressions by randomly sampling 20 news as negatives for each clicked article.

We additionally automatically annotate datasets with sentiment labels obtained by VADER (Hutto and Gilbert, 2014), a monolingual (English) rule-based algorithm (only for MIND), and a multilingual sentiment classification model of Barbieri et al. (2022), fine-tuned from XLM-RoBERTa Base (Conneau et al., 2020).

Recommendation Models. NewsRecLib provides implementations for 10 general-purpose NNRs and 3 fairness-aware recommenders. To support analysis of model components, for the models that did not use PLMs in their NEs (but rather contextualized embeddings with convolutional or attention layers), we implement an additional variant with a PLM-based NE (as proposed in Wu et al. (2021)). Furthermore, models can be trained either with *early fusion*, i.e., learning a parameterized user encoder to aggregate embeddings of news or the

[5] https://wandb.ai/site

[6] https://www.tensorflow.org/tensorboard

[7] https://torchmetrics.readthedocs.io/en/stable/

[8] https://optuna.readthedocs.io/en/stable/index.html

[9] Note that for the Adressa dataset, only a limited version of the dataset is available for download. For the full version containing additional features, users should contact the authors, as detailed in https://reclab.idi.ntnu.no/dataset/

simpler *late fusion* strategy proposed in Iana et al. (2023a), which replaces explicit user encoders with parameter-efficient dot products between candidate and clicked news embeddings. Appendix B details all available configurations for each recommendation model.

Training Objectives. Most NNR models are trained with point-wise classification objectives (Wang et al., 2018; Wu et al., 2019a,d) with negative sampling (Wu et al., 2019b, 2022a). In Iana et al. (2023a), we have shown that contrastive learning constitutes a viable alternative. At the same time, combining point-wise classification with contrastive objectives has been successfully employed in related tasks (Gunel et al., 2020). We thus implement three training objectives: cross-entropy loss, supervised contrastive loss (Khosla et al., 2020), and a dual objective that is a weighted average between the two.

Evaluation Metrics. `NewsRecLib` integrates standard accuracy-based metrics, such as AUC, MRR, and nDCG@k. Additionally, we implement aspect-based diversity and aspect-based personalization defined in Iana et al. (2023b). The availability of these beyond-accuracy metrics enables multi-faceted evaluation of NNRs.

3 Comparison to Related Frameworks

In the past decade, numerous frameworks for the development and comprehensive evaluation of recommender systems have been proposed to address the problem of reproducibility in the field (Gantner et al., 2011; Ekstrand et al., 2011; Ekstrand, 2020; Guo et al., 2015; Kula, 2017; Da Costa et al., 2018; Salah et al., 2020; Hug, 2020; Sun et al., 2020; Anelli et al., 2021). News recommendation poses different challenges for practitioners in comparison to recommendation in domains such as movies, music, or e-commerce (Raza and Ding, 2022; Iana et al., 2022). However, few of the existing and widely used libraries offer support for news recommenders, and especially for the modern neural news recommendation models.

Microsoft Recommenders (Graham et al., 2019; Argyriou et al., 2020) and RecBole (Zhao et al., 2021, 2022) provide implementation for five and three NNRs, respectively, as well as utilities for the MIND dataset. Nonetheless, other datasets, more recent approaches, and in particular fairness-aware models and beyond-accuracy metrics are not supported. StreamingRec (Jugovac et al., 2018) is a framework for evaluating streaming-based news recommenders, covering a wide range of algorithms, from trivial baselines (e.g., recently published, most popular) or item-to-item based collaborative filtering or session-based nearest neighbor techniques, to association rule methods and content-based approaches. However, it does not support any of the recent neural models. In these libraries, the sub-modules of a specific recommender are not decoupled from the overall model, which impedes experimentation with and analysis of different model components and training strategies.

In contrast to these frameworks, `NewsRecLib` focuses solely on the state-of-the-art neural news recommendation models, providing utilities for the most used benchmark datasets, architectures, training techniques, and evaluation metrics tailored to news recommendation. `NewsRecLib` unifies disparate implementations of recent neural news recommenders in a single open-source library that is built on top of mature frameworks for deep learning (PyTorch Lightning), evaluation (TorchMetrics), and configuration (Hydra).

4 Experiments

We conduct experiments with the pre-implemented recommendation models from `NewsRecLib` to investigate their performance when (1) trained with the original architecture (e.g., NE based on word embeddings and contextualization layer) and (2) trained with a PLM-based NE (Wu et al., 2021).

4.1 Datasets and Experimental Setup

We carry out the evaluation on the MINDsmall (Wu et al., 2020b) (denoted MIND) and Adressa-1 week (denoted Adressa) (Gulla et al., 2017) benchmark datasets. We evaluate two versions of the models, namely (1) with the original NE and (2) the NE modified to use a PLM (Wu et al., 2021) (if not used in the original NE). We use RoBERTa Base (Liu et al., 2019) and NB-BERT Base (Kummervold et al., 2021; Nielsen, 2023) for experiments on MIND and Adressa, respectively. In both cases, we fine-tune only the last four layers of the PLM in the interest of computational efficiency. We use

[10] We use the LSTUR$_{\text{ini}}$ version of the model. For details, refer to An et al. (2019).

[11] We use the MINER *weighted* version of the model. For details, refer to Li et al. (2022).

[12] We use the MANNeR version which performs multi-aspect diversification with $\lambda_{ctg} = -0.15$ and $\lambda_{snt} = -0.25$ for MIND, and $\lambda_{ctg} = -0.35$ and $\lambda_{snt} = -0.25$ for Adressa, respectively. For details, refer to Iana et al. (2023b).

		MIND						**Adressa**					
	Model	AUC	MRR	nDCG@5	nDCG@10	D_{ctg}@10	D_{snt}@10	AUC	MRR	nDCG@5	nDCG@10	D_{ctg}@10	D_{snt}@10
GeneralRec	DKN	50.0±0.0	26.3±0.4	24.6±0.5	31.5±0.3	50.4±1.0	66.0±0.6	–	–	–	–	–	–
	NPA	55.1±0.6	28.5±1.1	26.4±1.1	32.9±1.0	51.8±0.2	67.5±0.7	53.3±3.5	31.8±1.9	30.4±0.3	38.2±2.8	31.6±0.3	60.7±0.4
	NRMS	54.1±0.8	27.2±0.6	25.3±0.5	31.9±0.4	52.1±0.6	65.9±1.7	63.8±4.7	30.5±3.6	28.6±5.5	37.1±4.8	31.7±0.3	60.7±0.7
	NAML	50.2±0.0	33.4±0.6	31.8±0.7	38.1±0.5	47.0±1.0	66.9±0.3	50.0±0.0	37.8±3.5	38.2±4.1	45.1±3.6	31.5±4.6	60.7±0.4
	LSTUR[10]	58.8±2.1	32.2±0.9	30.4±0.9	36.8±0.9	43.1±1.2	65.6±0.6	68.1±2.4	**38.0±1.7**	**39.1±2.3**	**45.9±2.4**	27.7±2.4	60.1±0.3
	TANR	53.0±4.1	30.7±0.6	29.0±0.5	35.3±0.4	50.5±0.4	66.7±0.8	50.3±0.5	33.4±3.4	32.9±4.7	40.0±4.1	29.8±0.9	60.1±0.3
	CAUM	59.5±0.6	33.1±0.4	31.2±0.5	37.7±0.5	47.4±0.5	66.7±0.8	72.5±2.3	36.0±3.1	37.7±4.4	44.9±3.1	29.3±3.3	60.5±0.3
	MINS	56.1±1.5	31.0±1.5	29.4±1.5	35.7±1.5	47.0±1.7	67.6±1.1	**73.8±3.2**	37.4±2.5	38.8±4.1	45.8±3.2	32.4±0.8	60.6±0.3
	CenNewsRec	54.7±1.3	26.9±0.8	25.4±0.8	32.0±0.7	50.9±0.7	68.1±0.7	62.3±2.1	29.3±2.6	26.9±3.9	35.1±3.2	31.7±0.5	60.7±0.3
FairRec	SentiRec	52.0±0.5	27.2±0.9	25.2±1.0	31.8±0.8	52.5±1.2	67.7±1.1	55.0±0.7	26.9±0.4	24.3±0.7	30.1±0.7	35.2±0.1	**66.1±0.7**
	SentiDebias	56.6±1.7	25.4±0.7	23.7±0.9	30.3±0.6	53.5±1.3	68.1±1.3	66.5±0.9	29.4±0.7	29.2±1.6	36.9±1.2	31.3±0.8	61.1±0.3
GeneralRec	NRMS-PLM	50.0±0.0	21.9±2.8	19.5±2.9	26.0±3.0	53.2±1.7	66.1±3.4	53.1±2.7	34.9±2.5	34.7±3.0	42.8±2.8	32.3±1.2	61.6±0.3
	NAML-PLM	52.8±2.4	30.0±1.2	28.2±1.3	34.7±1.2	39.3±2.5	66.9±0.6	50.0±0.0	35.3±2.8	35.0±3.8	41.3±3.7	26.7±6.6	60.6±0.5
	LSTUR-PLM	50.0±0.0	30.7±0.6	29.0±0.6	35.3±0.6	36.6±0.9	67.0±0.9	55.5±2.3	30.4±1.6	28.8±2.3	35.3±2.2	22.3±2.8	60.9±0.4
	TANR-PLM	50.0±0.7	25.9±3.5	23.3±3.6	29.8±3.4	47.6±6.8	61.4±3.0	50.0±0.0	35.5±3.8	35.1±5.1	41.8±4.7	24.8±10.0	59.9±1.0
	CAUM-PLM	59.7±2.0	32.8±0.5	31.0±0.6	37.2±0.5	44.3±2.2	67.5±0.8	66.1±2.3	30.7±68	30.6±8.4	35.7±9.9	22.9±3.4	60.4±0.4
	MINS-PLM	50.0±0.7	22.4±3.5	20.2±3.9	26.5±4.0	50.6±3.2	67.3±1.1	65.3±4.4	33.1±2.8	31.5±4.6	40.4±3.9	26.6±5.5	60.5±0.5
	CenNewsRec-PLM	50.0±0.2	21.2±2.8	18.9±2.9	25.4±2.8	54.2±1.3	67.0±1.7	54.4±5.3	35.8±3.1	35.9±3.3	42.8±2.1	31.6±0.8	61.0±0.6
	MINER[11]	51.2±0.4	24.2±0.5	22.0±0.6	28.2±0.5	**54.8±0.3**	68.8±0.6	55.3±6.9	33.5±2.2	33.1±3.3	39.1±3.3	32.4±1.4	61.2±1.4
FairRec	SentiRec-PLM	50.0±0.6	24.7±0.7	22.6±0.6	29.1±0.6	52.3±2.4	67.2±2.1	61.2±3.0	31.6±3.4	30.4±4.4	38.2±4.4	32.9±1.7	59.9±2.4
	SentiDebias-PLM	51.0±0.5	28.7±0.4	27.5±0.4	34.0±0.4	47.7±2.0	67.9±1.7	67.3±2.8	37.1±3.6	38.0±5.1	45.3±3.8	32.6±1.2	61.5±1.0
	MANNeR[12]	**66.2±1.0**	**36.7±1.3**	**35.1±1.3**	**41.1±1.1**	50.5±0.3	**68.2±0.4**	67.6±4.3	31.9±2.8	30.5±4.1	38.9±3.9	**39.2±0.4**	64.9±0.5

Table 1: Recommendation and aspectual diversity (in terms of topical categories D_{ctg} and sentiments D_{snt}) performance of different neural news recommenders. We report averages and standard deviations across five different runs. The best results per column are highlighted in bold, the second best are underlined. The dashed line separates the general (GeneralRec) from the fairness-aware (FairRec) recommendation models.

100-dimensional TransE embeddings (Bordes et al., 2013) pretrained on Wikidata as input to the entity encoder for models using named entities as input features to their NEs, a maximum history length of 50, and set all other model-specific hyperparameters to optimal values reported in the respective papers. We train all models with mixed precision, and optimize with the Adam algorithm (Kingma and Ba, 2014), with the learning rate of 1e-4. We train models with a PLM-empowered NE for 10 epochs, and the model variant without PLMs for 20 epochs. Since Adressa contains no abstract or disambiguated named entities, we use only the title for the models benchmarked on that dataset.

4.2 Results

Table 1 summarizes the results on content-based recommendation performance (w.r.t. AUC, MRR, nDCG@5, nDCG@10) and aspect diversification for topical categories (D_{ctg}) and sentiment (D_{snt}), as per Iana et al. (2023b). We find that PLM-based NEs do not necessarily lead to performance improvements. We hypothesize that this is due to the dataset size: a PLM-based NE requires training a larger number of parameters than one which contextualizes pretrained word embeddings with a CNN or attention network. Note that rather small improvements of PLM-empowered NEs over original NEs have been shown only for larger-scale datasets (Wu et al., 2021). These findings indicate that more research is needed to understand under which settings older NEs can still benefit NNRs. MANNeR, with its late click behavior fusion approach, outperforms all other models on MIND, but it underperforms on Adressa. Note that the contrastive learning training approach adopted by MANNeR (Iana et al., 2023b) benefits from larger training datasets, and MINDsmall has roughly five times as many news as Adressa 1-week. Expectedly, w.r.t. aspect-based diversity, NNRs with diversification objectives (e.g., for sentiment) outperform models trained only to maximize content-based accuracy.

5 Conclusion

In this work, we introduced NewsRecLib, a highly configurable, modular and easily extensible framework for neural news recommendation. Our library is specifically designed to foster reproducible research in recommender systems and rigorous evaluation of models – users only need to create a single configuration file for an experiment. We briefly described the underlying principles of NewsRecLib and the structure of its building blocks. The framework currently provides two standard benchmark datasets, loading and pre-processing functions, 13 neural recommendation models, different training objectives and hyperparameters optimization strategies, numerous evaluation metrics, extensive logging capabilities, and GPU support. We believe that NewsRecLib is a useful tool for the community that will (i) catalyze reproducible NNR research, (ii) foster fairer comparisons between the models, and (iii) facilitate identification of NNR components that drive their performance.

Limitations

While we have striven to build a comprehensive library for the design and fair evaluation of neural news recommendation models, several additional factors must be taken into account. Firstly, even though we aim to replicate the original implementations of the models to the highest degree possible, discrepancies in our code and results can arise from the usage of different frameworks, as well as scarce availability of implementation details in the source code or publications of some of the recommenders. Secondly, our library is heavily dependent on the changes and maintenance of the frameworks on which it is built, namely PyTorch Lightning (and by extension, PyTorch), Hydra, TorchMetrics, Optuna. As such, new plugins for logging (e.g., Neptune (Neptune team, 2019), Comet (Rei et al., 2020), MLFlow (Zaharia et al., 2018)) or hyperparamter optimization (e.g., Ax[13]) need to be integrated with PyTorch Lightning and Hydra.

Moreover, we rely on open benchmark news datasets for training and evaluating the recommenders. Consequently, any biases that might be contained in the news and user data could be propagated through the recommendation pipeline. Additionally, the usage of these datasets is intertwined with their public availability. Any changes to the datasets or access restrictions are likely to impact the way pre-implemented models in `NewsRecLib` can be trained and benchmarked.

Lastly, neural news recommendation is a computationally expensive endeavor which requires availability of large compute resources. Although `NewsRecLib` technically supports execution of experiments on CPU, this would be not only highly inefficient and time-consuming, but also infeasible for large-scale datasets with hundreds of thousands of users and news. Consequently, users should ideally have access to GPUs to efficiently use our library.

Ethics Statement

Users of our library should differentiate the recommendation models available in `NewsRecLib` from the originals. Consequently, they should explicitly credit and cite both `NewsRecLib`, as well as the original implementations, as specified on our GitHub page.

[13] https://ax.dev/

Acknowledgements

The authors acknowledge support by the state of Baden-Württemberg through bwHPC and the German Research Foundation (DFG) through grant INST 35/1597-1 FUGG.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Mingxiao An, Fangzhao Wu, Chuhan Wu, Kun Zhang, Zheng Liu, and Xing Xie. 2019. Neural news recommendation with long-and short-term user representations. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 336–345.

Vito Walter Anelli, Alejandro Bellogín, Antonio Ferrara, Daniele Malitesta, Felice Antonio Merra, Claudio Pomo, Francesco Maria Donini, and Tommaso Di Noia. 2021. Elliot: A comprehensive and rigorous framework for reproducible recommender systems evaluation. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*, pages 2405–2414.

Andreas Argyriou, Miguel González-Fierro, and Le Zhang. 2020. Microsoft recommenders: Best practices for production-ready recommendation systems. In *Companion Proceedings of the Web Conference 2020*, pages 50–51.

Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015*.

Francesco Barbieri, Luis Espinosa Anke, and Jose Camacho-Collados. 2022. XLM-T: Multilingual language models in twitter for sentiment analysis and beyond. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 258–266.

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2011. Algorithms for hyper-parameter optimization. *Advances in neural information processing systems*, 24.

Lukas Biewald. 2020. Experiment tracking with weights and biases. Software available from wandb.com.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, page 1724. Association for Computational Linguistics.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Édouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451.

Arthur Da Costa, Eduardo Fressato, Fernando Neto, Marcelo Manzato, and Ricardo Campello. 2018. Case recommender: a flexible and extensible python framework for recommender systems. In *Proceedings of the 12th ACM Conference on Recommender Systems*, pages 494–495.

Nicki Skafte Detlefsen, Jiri Borovec, Justus Schock, Ananya Harsh, Teddy Koker, Luca Di Liello, Daniel Stancl, Changsheng Quan, Maxim Grechkin, and William Falcon. 2022. Torchmetrics - measuring reproducibility in pytorch.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Michael D Ekstrand. 2020. Lenskit for python: Next-generation software for recommender systems experiments. In *Proceedings of the 29th ACM international conference on information & knowledge management*, pages 2999–3006.

Michael D Ekstrand, Michael Ludwig, Joseph A Konstan, and John T Riedl. 2011. Rethinking the recommender research ecosystem: reproducibility, openness, and lenskit. In *Proceedings of the fifth ACM conference on Recommender systems*, pages 133–140.

William Falcon and The PyTorch Lightning team. 2019. PyTorch Lightning.

Maurizio Ferrari Dacrema, Simone Boglio, Paolo Cremonesi, and Dietmar Jannach. 2021. A troubling analysis of reproducibility and progress in recommender systems research. *ACM Transactions on Information Systems (TOIS)*, 39(2):1–49.

Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. 2019. Are we really making much progress? A worrying analysis of recent neural recommendation approaches. In *Proceedings of the 13th ACM conference on recommender systems*, pages 101–109.

Zeno Gantner, Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2011. MyMediaLite: A free recommender system library. In *Proceedings of the fifth ACM conference on Recommender systems*, pages 305–308.

Shansan Gong and Kenny Q Zhu. 2022. Positive, negative and neutral: Modeling implicit feedback in session-based news recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1185–1195.

Scott Graham, Jun-Ki Min, and Tao Wu. 2019. Microsoft recommenders: tools to accelerate developing recommender systems. In *Proceedings of the 13th ACM Conference on Recommender Systems*, pages 542–543.

Jon Atle Gulla, Lemei Zhang, Peng Liu, Özlem Özgöbek, and Xiaomeng Su. 2017. The adressa dataset for news recommendation. In *Proceedings of the international conference on web intelligence*, pages 1042–1048.

Beliz Gunel, Jingfei Du, Alexis Conneau, and Veselin Stoyanov. 2020. Supervised contrastive learning for pre-trained language model fine-tuning. In *International Conference on Learning Representations*.

Guibing Guo, Jie Zhang, Zhu Sun, and Neil Yorke-Smith. 2015. Librec: A java library for recommender systems. In *UMAP workshops*, volume 4, pages 38–45. Citeseer.

Linmei Hu, Siyong Xu, Chen Li, Cheng Yang, Chuan Shi, Nan Duan, Xing Xie, and Ming Zhou. 2020. Graph neural news recommendation with unsupervised preference disentanglement. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 4255–4264.

Nicolas Hug. 2020. Surprise: A python library for recommender systems. *Journal of Open Source Software*, 5(52):2174.

Clayton Hutto and Eric Gilbert. 2014. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Proceedings of the international AAAI conference on web and social media*, volume 8, pages 216–225.

Andreea Iana, Mehwish Alam, and Heiko Paulheim. 2022. A survey on knowledge-aware news recommender systems. *Semantic Web*, (Preprint):1–62.

Andreea Iana, Goran Glavas, and Heiko Paulheim. 2023a. Simplifying content-based neural news recommendation: On user modeling and training objectives. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2384–2388.

Andreea Iana, Goran Glavaš, and Heiko Paulheim. 2023b. Train once, use flexibly: A modular framework for multi-aspect neural news recommendation. *arXiv preprint arXiv:2307.16089*.

Michael Jugovac, Dietmar Jannach, and Mozhgan Karimi. 2018. Streamingrec: a framework for benchmarking stream-based news recommenders. In *Proceedings of the 12th ACM conference on recommender systems*, pages 269–273.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. 2020. Supervised contrastive learning. *Advances in neural information processing systems*, 33:18661–18673.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Maciej Kula. 2017. Spotlight. https://github.com/maciejkula/spotlight.

Per E Kummervold, Javier De La Rosa, Freddy Wetjen, and Svein Arne Brygfjeld. 2021. Operationalizing a national digital library: The case for a norwegian transformer model. In *Proceedings of the 23rd Nordic Conference on Computational Linguistics (NoDaLiDa)*, pages 20–29.

Jian Li, Jieming Zhu, Qiwei Bi, Guohao Cai, Lifeng Shang, Zhenhua Dong, Xin Jiang, and Qun Liu. 2022. MINER: multi-interest matching network for news recommendation. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 343–352.

Miaomiao Li and Licheng Wang. 2019. A survey on personalized news recommendation technology. *IEEE Access*, 7:145861–145879.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Neptune team. 2019. neptune.ai.

Dan Nielsen. 2023. ScandEval: A benchmark for Scandinavian natural language processing. In *Proceedings of the 24th Nordic Conference on Computational Linguistics (NoDaLiDa)*, pages 185–201, Tórshavn, Faroe Islands. University of Tartu Library.

Yoshihiko Ozaki, Yuki Tanigaki, Shuhei Watanabe, and Masaki Onishi. 2020. Multiobjective tree-structured parzen estimator for computationally expensive optimization problems. In *Proceedings of the 2020 genetic and evolutionary computation conference*, pages 533–541.

Tao Qi, Fangzhao Wu, Chuhan Wu, and Yongfeng Huang. 2022. News recommendation with candidate-aware user modeling. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1917–1921.

Tao Qi, Fangzhao Wu, Chuhan Wu, Yongfeng Huang, and Xing Xie. 2020. Privacy-preserving news recommendation model learning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1423–1432.

Shaina Raza and Chen Ding. 2022. News recommender system: a review of recent progress, challenges, and opportunities. *Artificial Intelligence Review*, pages 1–52.

Ricardo Rei, Craig Stewart, Ana C Farinha, and Alon Lavie. 2020. Comet: A neural framework for mt evaluation. *arXiv preprint arXiv:2009.09025*.

Steffen Rendle, Li Zhang, and Yehuda Koren. 2019. On the difficulty of evaluating baselines: A study on recommender systems. *arXiv preprint arXiv:1905.01395*.

Alan Said and Alejandro Bellogín. 2014. Comparative recommender system evaluation: benchmarking recommendation frameworks. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 129–136.

Aghiles Salah, Quoc-Tuan Truong, and Hady W Lauw. 2020. Cornac: A comparative framework for multimodal recommender systems. *The Journal of Machine Learning Research*, 21(1):3803–3807.

Mete Sertkan and Julia Neidhardt. 2022. Diversifying sentiments in news recommendation. In *Proceedings of the 2nd Perspectives on the Evaluation of Recommender Systems Workshop*.

Zhu Sun, Di Yu, Hui Fang, Jie Yang, Xinghua Qu, Jie Zhang, and Cong Geng. 2020. Are we evaluating rigorously? benchmarking recommendation for reproducible evaluation and fair comparison. In *Proceedings of the 14th ACM Conference on Recommender Systems*, pages 23–32.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Hongwei Wang, Fuzheng Zhang, Xing Xie, and Minyi Guo. 2018. DKN: Deep knowledge-aware network for news recommendation. In *Proceedings of the 2018 world wide web conference*, pages 1835–1844.

Rongyao Wang, Shoujin Wang, Wenpeng Lu, and Xueping Peng. 2022. News recommendation via multi-interest news sequence modelling. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7942–7946. IEEE.

Chuhan Wu, Fangzhao Wu, Mingxiao An, Jianqiang Huang, Yongfeng Huang, and Xing Xie. 2019a. Neural news recommendation with attentive multi-view learning. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 3863–3869.

Chuhan Wu, Fangzhao Wu, Mingxiao An, Jianqiang Huang, Yongfeng Huang, and Xing Xie. 2019b. NPA: neural news recommendation with personalized attention. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2576–2584.

Chuhan Wu, Fangzhao Wu, Mingxiao An, Yongfeng Huang, and Xing Xie. 2019c. Neural news recommendation with topic-aware news representation. In *Proceedings of the 57th Annual meeting of the association for computational linguistics*, pages 1154–1159.

Chuhan Wu, Fangzhao Wu, Suyu Ge, Tao Qi, Yongfeng Huang, and Xing Xie. 2019d. Neural news recommendation with multi-head self-attention. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pages 6389–6394.

Chuhan Wu, Fangzhao Wu, and Yongfeng Huang. 2022a. Rethinking InfoNCE: How many negative samples do you need? In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence ((IJCAI-22))*, pages 2509–2515.

Chuhan Wu, Fangzhao Wu, Yongfeng Huang, and Xing Xie. 2023. Personalized news recommendation: Methods and challenges. *ACM Transactions on Information Systems*, 41(1):1–50.

Chuhan Wu, Fangzhao Wu, Tao Qi, and Yongfeng Huang. 2020a. Sentirec: Sentiment diversity-aware neural news recommendation. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 44–53.

Chuhan Wu, Fangzhao Wu, Tao Qi, and Yongfeng Huang. 2021. Empowering news recommendation with pre-trained language models. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1652–1656.

Chuhan Wu, Fangzhao Wu, Tao Qi, Wei-Qiang Zhang, Xing Xie, and Yongfeng Huang. 2022b. Removing ai's sentiment manipulation of personalized news delivery. *Humanities and Social Sciences Communications*, 9(1):1–9.

Fangzhao Wu, Ying Qiao, Jiun-Hung Chen, Chuhan Wu, Tao Qi, Jianxun Lian, Danyang Liu, Xing Xie, Jianfeng Gao, Winnie Wu, et al. 2020b. Mind: A large-scale dataset for news recommendation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3597–3606.

Hongyan Xu, Qiyao Peng, Hongtao Liu, Yueheng Sun, and Wenjun Wang. 2023. Group-based personalized news recommendation with long-and short-term fine-grained matching. *ACM Transactions on Information Systems*.

Omry Yadan. 2019. Hydra - a framework for elegantly configuring complex applications. Github.

Jingwei Yi, Fangzhao Wu, Chuhan Wu, Ruixuan Liu, Guangzhong Sun, and Xing Xie. 2021. Efficient-FedRec: Efficient federated learning framework for privacy-preserving news recommendation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2814–2824.

Matei Zaharia, Andrew Chen, Aaron Davidson, Ali Ghodsi, Sue Ann Hong, Andy Konwinski, Siddharth Murching, Tomas Nykodym, Paul Ogilvie, Mani Parkhe, et al. 2018. Accelerating the machine learning lifecycle with mlflow. *IEEE Data Eng. Bull.*, 41(4):39–45.

Qi Zhang, Qinglin Jia, Chuyuan Wang, Jingjie Li, Zhaowei Wang, and Xiuqiang He. 2021. Amm: Attentive multi-field matching for news recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1588–1592.

Wayne Xin Zhao, Yupeng Hou, Xingyu Pan, Chen Yang, Zeyu Zhang, Zihan Lin, Jingsen Zhang, Shuqing Bian, Jiakai Tang, Wenqi Sun, et al. 2022. Recbole 2.0: towards a more up-to-date recommendation library. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 4722–4726.

Wayne Xin Zhao, Shanlei Mu, Yupeng Hou, Zihan Lin, Yushuo Chen, Xingyu Pan, Kaiyuan Li, Yujie Lu, Hui Wang, Changxin Tian, et al. 2021. Recbole: Towards a unified, comprehensive and efficient framework for recommendation algorithms. In *proceedings of the 30th acm international conference on information & knowledge management*, pages 4653–4664.

A Logging

A.1 Configuration Logging

Figs. 4 and 5 illustrate an example of how the configuration of each of the pipeline's components is logged when the training process is initiated.

A.2 Model Metadata Logging

Fig. 6 shows an example of logging relevant metadata information regarding a model's size and number of parameters.

B Supported Recommendation Models and Configurations

NewsRecLib provides, to date, implementations of 10 general NNRs:

- *DKN* (Wang et al., 2018) uses a word-entity aligned knowledge-aware convolutional neural network (CNN) (Kim, 2014) to produce news embeddings. It learns candidate-aware representations of users as the weighted sum of their clicked news embeddings, where the weights are computed by an attention network that takes as input the embeddings of the candidate and of the clicked news.

- *NPA* (Wu et al., 2019b) contextualizes pretrained word embeddings with a CNN, followed by a personalized attention module. Its UE consists of a similar personalized attention module which aggregates the representations of the users' clicked news, with projected embeddings of the users IDs as attention queries.

- *NAML* (Wu et al., 2019a) uses a sequence of CNN and additive attention (Bahdanau et al., 2015) to contextualize pretrained word embeddings in its NE. Additionally, it leverages category information, with categories embedded through a linear layer. User representations are learned from the embeddings of users' clicked news with another additive attention layer.

- *NRMS* (Wu et al., 2019d) learns news representations from pretrained word embeddings and a combination of multi-head self-attention (Vaswani et al., 2017) and additive attention; it embeds users with a two-layer encoder consisting also of multi-head self-attention and additive attention.

- *LSTUR* (An et al., 2019) embeds news similarly to NAML (Wu et al., 2019a). However, it learns user representations via recurrent neural networks: it produces short-term user embeddings from the clicked news with a GRU (Cho et al., 2014), which it combines with a long-term embedding, consisting of a randomly initialized and fine-tuned part.

- *TANR* (Wu et al., 2019c) injects information on topical categories, by jointly optimizing the NNR for content personalization and topic classification. It uses the same UE and NE architecture as NAML (Wu et al., 2019a), but does not embed categories.

- *CAUM* (Qi et al., 2022) uses a NRMS-based NE, and additionally encodes title entities with attention layers. Moreover, its candidate-aware UE combines a candidate-aware self-attention network which models long-range dependencies between clicked news, conditioned on the candidate, with a candidate-aware CNN that captures short-term user interests from adjacent clicks, again conditioned on the candidate's content.

- *MINS* (Wang et al., 2022) embeds textual features of news (i.e., title, abstract) in the same manner as NRMS (Wu et al., 2019d), and categories through a linear embedding layer. Moreover, it uses a combination of multi-head self-attention, multi-channel GRU-based recurrent network, and additive attention to encode users.

- *CenNewsRec* (Qi et al., 2020) combines a CNN network with multi-head self-attention and additive attention modules to produce contextualized representations of news. Its UE resembles that of LSTUR (An et al., 2019), but it learns long-term user vectors from clicked news using a sequence of multi-head self-attention and attentive pooling networks, as opposed to storing an explicit embedding per user.

- *MINER* (Li et al., 2022) uses a pretrained BERT (Devlin et al., 2019) model as NE. Its UE learns multiple user representation vectors using a poly attention scheme that extracts interests vectors through additive attention layers.

		News Encoder		Click Behavior Fusion		Training Objective		
	Model	Word emb. + contextualization	PLM	EF	LF	CE	SCL	Dual
GeneralRec	DKN (Wang et al., 2018)	✓	✗	✓	✓	✓	✓	✓
	NPA (Wu et al., 2019b)	✓	✗	✓	✓	✓	✓	✓
	NRMS (Wu et al., 2019d)	✓	✓	✓	✓	✓	✓	✓
	NAML (Wu et al., 2019a)	✓	✓	✓	✓	✓	✓	✓
	LSTUR (An et al., 2019)	✓	✓	✓	✓	✓	✓	✓
	TANR (Wu et al., 2019c)	✓	✓	✓	✓	✓	✓	✓
	CAUM (Qi et al., 2022)	✓	✓	✓	✓	✓	✓	✓
	MINS (Wang et al., 2022)	✓	✓	✓	✓	✓	✓	✓
	CenNewsRec (Qi et al., 2020)	✓	✓	✓	✓	✓	✓	✓
	MINER (Li et al., 2022)	✓	✓	✓	✓	✓	✓	✓
FairRec	SentiRec (Wu et al., 2020a)	✓	✓	✓	✓	✓	✓	✓
	SentiDebias (Wu et al., 2022b)	✓	✓	✓	✓	✓	✗	✗
	MANNeR ((Iana et al., 2023b))	✓	✓	✓	✓	✗	✓	✗

Table 2: List of currently available models in `NewsRecLib`, and supported configurations. For *click behavior fusion* we differentiate between `early fusion` (EF) and `late fusion` (LF). Models can be trained with `cross-entropy loss` (CE), `supervised contrastive loss` (SCL), and a `dual objective` combining both CE and SCL losses as weighted average (Dual). The dashed line separates the general (`GeneralRec`) from the fairness-aware (`FairRec`) recommendation models.

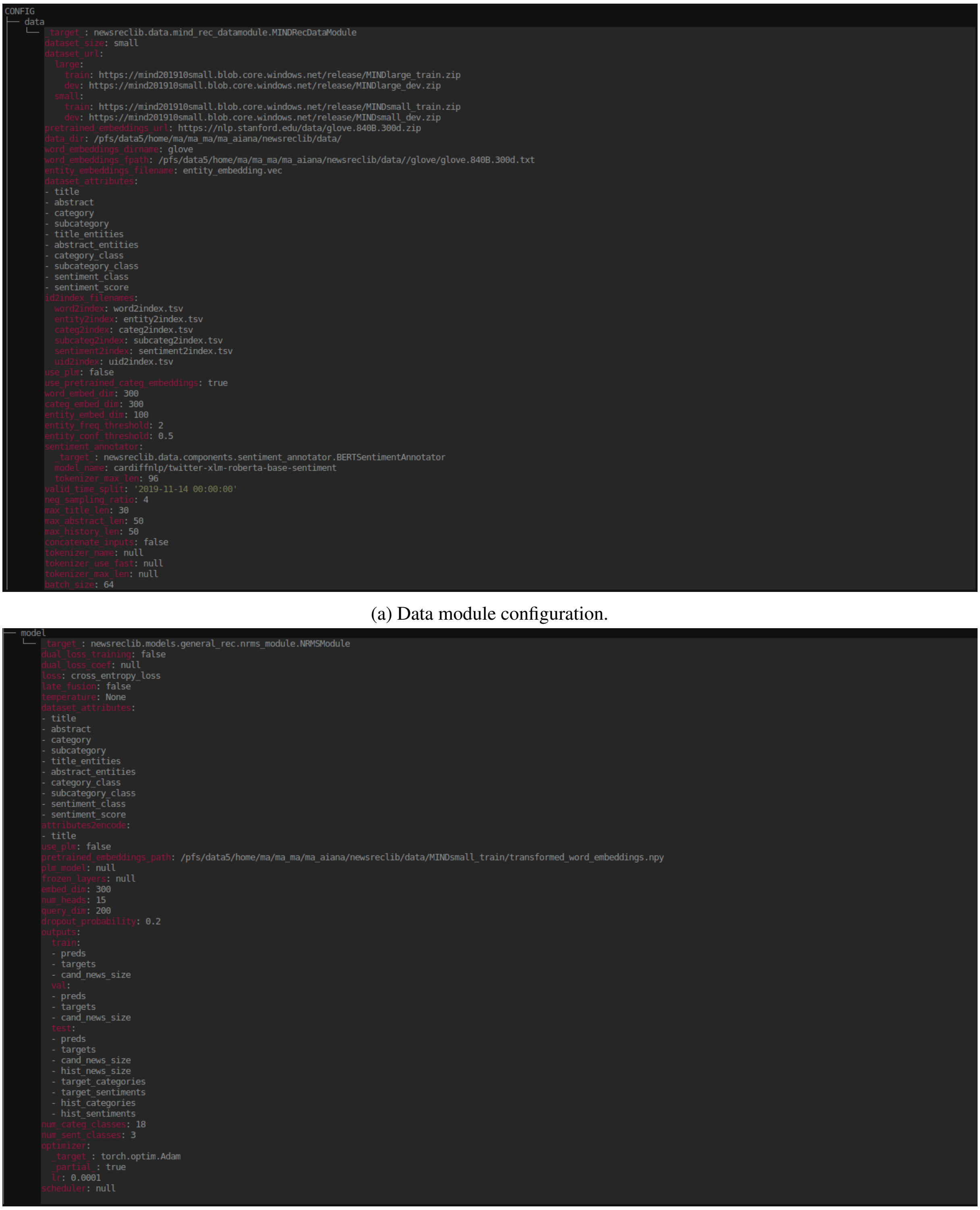
Additionally, `NewsRecLib` integrates 3 fairness-aware models, namely NNRs that target diversity of recommendations along with pure content-based personalization:

- *SentiRec* (Wu et al., 2020a) uses a similar architecture to NRMS (Wu et al., 2019d) and injects sentiment information by optimizing simultaneously for content personalization, as well as sentiment prediction. Additionally, it regularizes the NNR for sentiment diversity.

- *SentiDebias* (Wu et al., 2022b) is a framework for sentiment debiasing which uses the architecture of NRMS (Wu et al., 2019d), as well as adversarial learning to reduce the model's sentiment bias (originating from the user data) and generate sentiment-agnostic and diverse recommendations.

- *MANNeR* (Iana et al., 2023b) is a modular framework for multi-aspect neural news recommendation, which comprises two types of modules, each with a corresponding NE (which combines a PLM-based text encoder with an entity embedder consisting of a pretrained embedding and multi-head self-attention layer), which are responsible for content-based, and respectively, aspect-based personalization. Both modules are trained with a contrastive metric objective. MANNeR uses late fusion (Iana et al., 2023a) instead of standard user encoders. At inference time, the aspect-specific similarity scores are arbitrarily aggregated depending on the downstream task (e.g., content-based personalization, aspect-based diversification) to produce a final ranking of the news.

Table 2 provides an overview of the supported configurations for the available models. For each model, users can choose the type of news encoder, click behavior fusion, and training objective. Note that for some models, due to the high interdependencies between NE and UE, it is not possible to easily replace the original NE with a PLM-based one without breaking the framework's modularity. Similarly, some models have been designed from the start with a PLM-based NE. In both of these cases, we only provide support for the original NE. Due to the design of some model architectures, changing the training objective would modify the functionality of the model (e.g., using different loss functions in the `CR-Module` and `A-Module` of MANNeR (Iana et al., 2023b)). In these cases, we only provide support for one training objective.



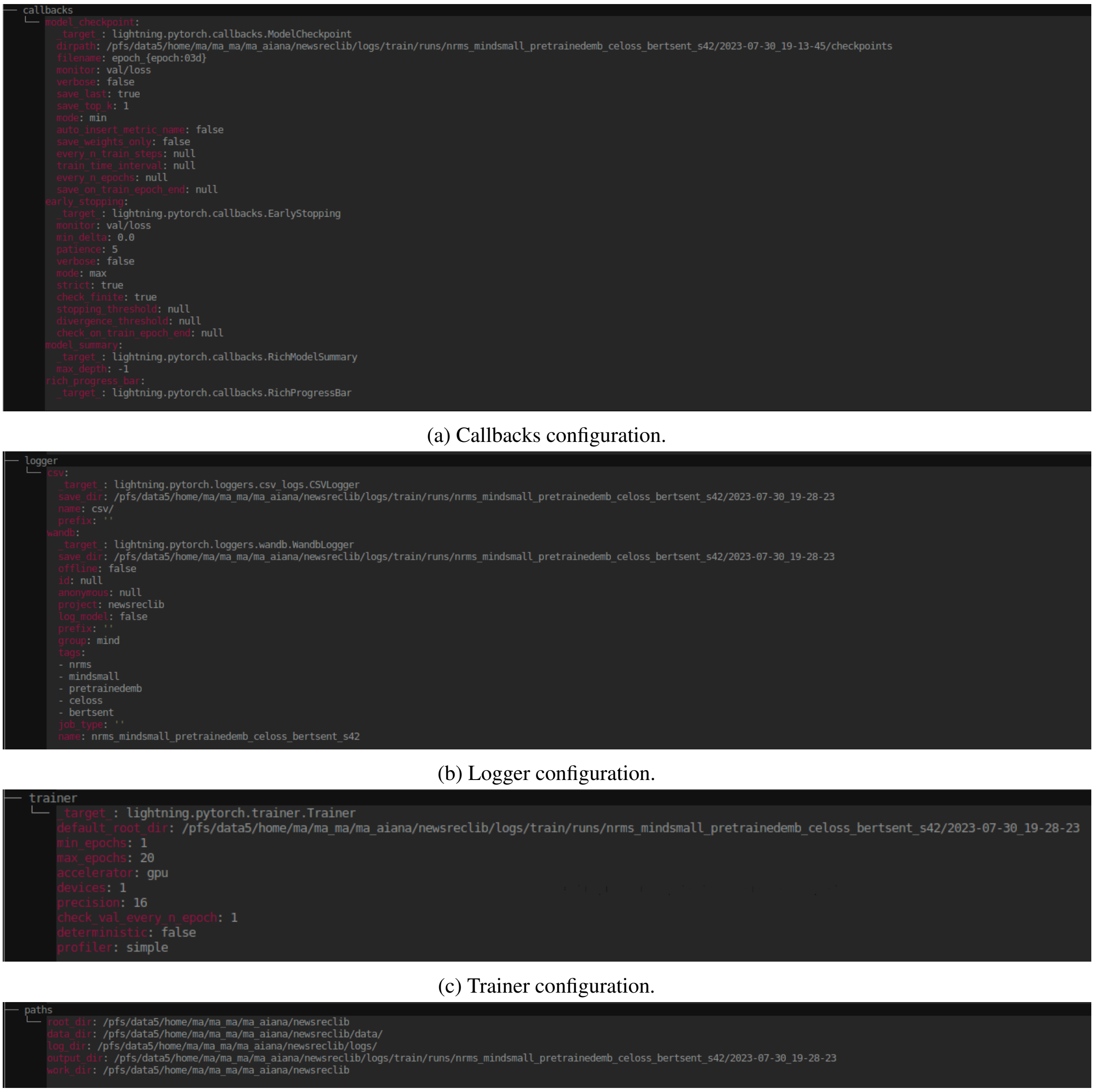
```
CONFIG
├── data
│   └── _target_: newsreclib.data.mind_rec_datamodule.MINDRecDataModule
        dataset_size: small
        dataset_url:
          large:
            train: https://mind201910small.blob.core.windows.net/release/MINDlarge_train.zip
            dev: https://mind201910small.blob.core.windows.net/release/MINDlarge_dev.zip
          small:
            train: https://mind201910small.blob.core.windows.net/release/MINDsmall_train.zip
            dev: https://mind201910small.blob.core.windows.net/release/MINDsmall_dev.zip
        pretrained_embeddings_url: https://nlp.stanford.edu/data/glove.840B.300d.zip
        data_dir: /pfs/data5/home/ma/ma_ma/ma_aiana/newsreclib/data/
        word_embeddings_dirname: glove
        word_embeddings_fpath: /pfs/data5/home/ma/ma_ma/ma_aiana/newsreclib/data//glove/glove.840B.300d.txt
        entity_embeddings_filename: entity_embedding.vec
        dataset_attributes:
        - title
        - abstract
        - category
        - subcategory
        - title_entities
        - abstract_entities
        - category_class
        - subcategory_class
        - sentiment_class
        - sentiment_score
        id2index_filenames:
          word2index: word2index.tsv
          entity2index: entity2index.tsv
          categ2index: categ2index.tsv
          subcateg2index: subcateg2index.tsv
          sentiment2index: sentiment2index.tsv
          uid2index: uid2index.tsv
        use_plm: false
        use_pretrained_categ_embeddings: true
        word_embed_dim: 300
        categ_embed_dim: 300
        entity_embed_dim: 100
        entity_freq_threshold: 2
        entity_conf_threshold: 0.5
        sentiment_annotator:
          _target_: newsreclib.data.components.sentiment_annotator.BERTSentimentAnnotator
          model_name: cardiffnlp/twitter-xlm-roberta-base-sentiment
          tokenizer_max_len: 96
        valid_time_split: '2019-11-14 00:00:00'
        neg_sampling_ratio: 4
        max_title_len: 30
        max_abstract_len: 50
        max_history_len: 50
        concatenate_inputs: false
        tokenizer_name: null
        tokenizer_use_fast: null
        tokenizer_max_len: null
        batch_size: 64
```

(a) Data module configuration.

```
├── model
│   └── _target_: newsreclib.models.general_rec.nrms_module.NRMSModule
        dual_loss_training: false
        dual_loss_coef: null
        loss: cross_entropy_loss
        late_fusion: false
        temperature: None
        dataset_attributes:
        - title
        - abstract
        - category
        - subcategory
        - title_entities
        - abstract_entities
        - category_class
        - subcategory_class
        - sentiment_class
        - sentiment_score
        attributes2encode:
        - title
        use_plm: false
        pretrained_embeddings_path: /pfs/data5/home/ma/ma_ma/ma_aiana/newsreclib/data/MINDsmall_train/transformed_word_embeddings.npy
        plm_model: null
        frozen_layers: null
        embed_dim: 300
        num_heads: 15
        query_dim: 200
        dropout_probability: 0.2
        outputs:
          train:
          - preds
          - targets
          - cand_news_size
          val:
          - preds
          - targets
          - cand_news_size
          test:
          - preds
          - targets
          - cand_news_size
          - hist_news_size
          - target_categories
          - target_sentiments
          - hist_categories
          - hist_sentiments
        num_categ_classes: 18
        num_sent_classes: 3
        optimizer:
          _target_: torch.optim.Adam
          _partial_: true
          lr: 0.0001
        scheduler: null
```

(b) Recommendation module configuration.

Figure 4: Example for logging the configurations of the data and the recommendation modules.



```
── callbacks
   └── model_checkpoint:
         _target_: lightning.pytorch.callbacks.ModelCheckpoint
         dirpath: /pfs/data5/home/ma/ma_ma/ma_aiana/newsreclib/logs/train/runs/nrms_mindsmall_pretrainedemb_celoss_bertsent_s42/2023-07-30_19-13-45/checkpoints
         filename: epoch_{epoch:03d}
         monitor: val/loss
         verbose: false
         save_last: true
         save_top_k: 1
         mode: min
         auto_insert_metric_name: false
         save_weights_only: false
         every_n_train_steps: null
         train_time_interval: null
         every_n_epochs: null
         save_on_train_epoch_end: null
       early_stopping:
         _target_: lightning.pytorch.callbacks.EarlyStopping
         monitor: val/loss
         min_delta: 0.0
         patience: 5
         verbose: false
         mode: max
         strict: true
         check_finite: true
         stopping_threshold: null
         divergence_threshold: null
         check_on_train_epoch_end: null
       model_summary:
         _target_: lightning.pytorch.callbacks.RichModelSummary
         max_depth: -1
       rich_progress_bar:
         _target_: lightning.pytorch.callbacks.RichProgressBar
```

(a) Callbacks configuration.

```
── logger
   └── csv:
         _target_: lightning.pytorch.loggers.csv_logs.CSVLogger
         save_dir: /pfs/data5/home/ma/ma_ma/ma_aiana/newsreclib/logs/train/runs/nrms_mindsmall_pretrainedemb_celoss_bertsent_s42/2023-07-30_19-28-23
         name: csv/
         prefix: ''
       wandb:
         _target_: lightning.pytorch.loggers.wandb.WandbLogger
         save_dir: /pfs/data5/home/ma/ma_ma/ma_aiana/newsreclib/logs/train/runs/nrms_mindsmall_pretrainedemb_celoss_bertsent_s42/2023-07-30_19-28-23
         offline: false
         id: null
         anonymous: null
         project: newsreclib
         log_model: false
         prefix: ''
         group: mind
         tags:
         - nrms
         - mindsmall
         - pretrainedemb
         - celoss
         - bertsent
         job_type: ''
         name: nrms_mindsmall_pretrainedemb_celoss_bertsent_s42
```

(b) Logger configuration.

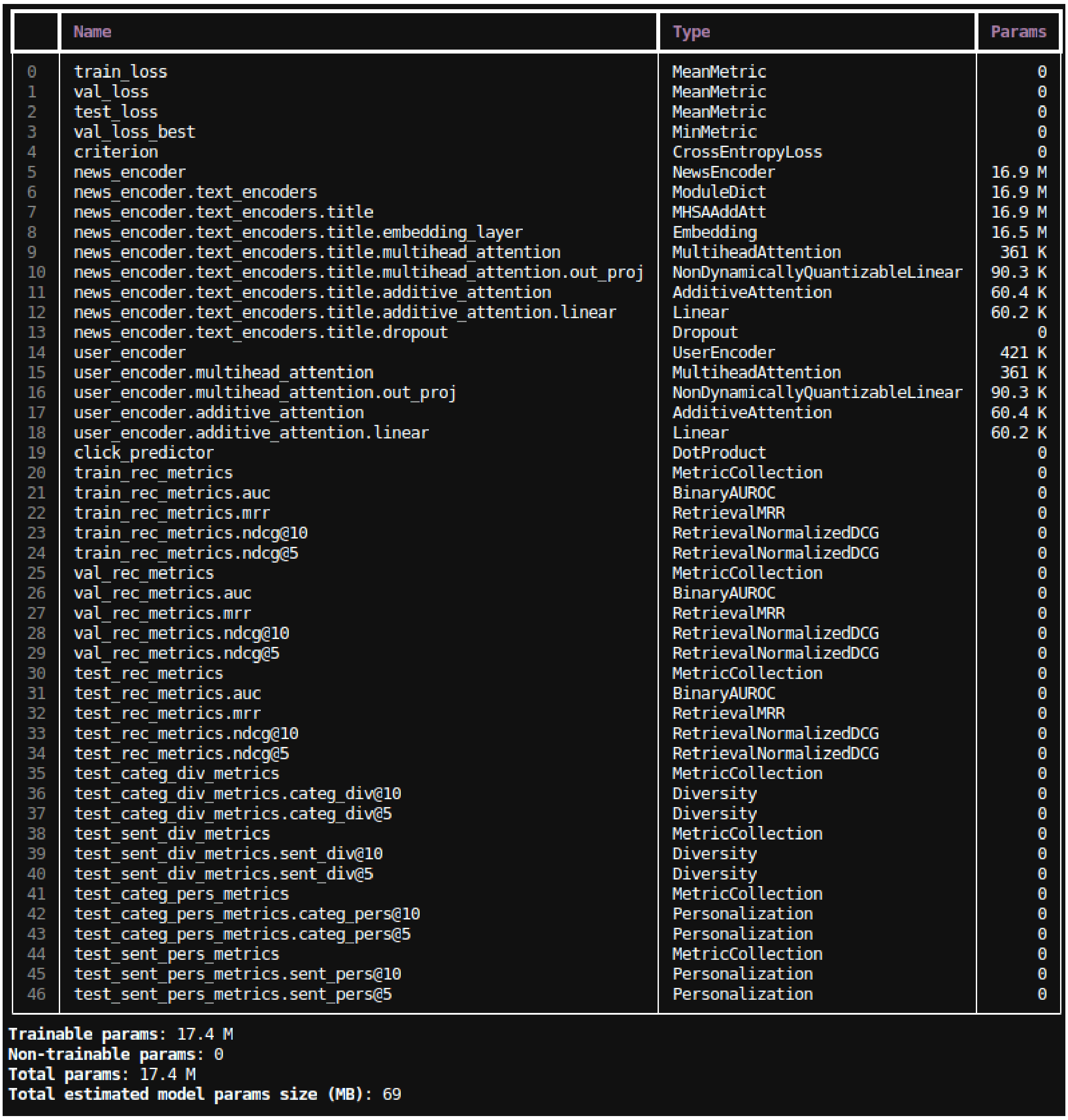
```
── trainer
   └── _target_: lightning.pytorch.trainer.Trainer
       default_root_dir: /pfs/data5/home/ma/ma_ma/ma_aiana/newsreclib/logs/train/runs/nrms_mindsmall_pretrainedemb_celoss_bertsent_s42/2023-07-30_19-28-23
       min_epochs: 1
       max_epochs: 20
       accelerator: gpu
       devices: 1
       precision: 16
       check_val_every_n_epoch: 1
       deterministic: false
       profiler: simple
```

(c) Trainer configuration.

```
── paths
   └── root_dir: /pfs/data5/home/ma/ma_ma/ma_aiana/newsreclib
       data_dir: /pfs/data5/home/ma/ma_ma/ma_aiana/newsreclib/data/
       log_dir: /pfs/data5/home/ma/ma_ma/ma_aiana/newsreclib/logs/
       output_dir: /pfs/data5/home/ma/ma_ma/ma_aiana/newsreclib/logs/train/runs/nrms_mindsmall_pretrainedemb_celoss_bertsent_s42/2023-07-30_19-28-23
       work_dir: /pfs/data5/home/ma/ma_ma/ma_aiana/newsreclib
```

(d) Paths configuration.

Figure 5: Example for logging the configurations of the callbacks, loggers, and trainer.



```
    Name                                                          Type                              Params
 0  train_loss                                                    MeanMetric                             0
 1  val_loss                                                      MeanMetric                             0
 2  test_loss                                                     MeanMetric                             0
 3  val_loss_best                                                 MinMetric                              0
 4  criterion                                                     CrossEntropyLoss                       0
 5  news_encoder                                                  NewsEncoder                       16.9 M
 6  news_encoder.text_encoders                                    ModuleDict                        16.9 M
 7  news_encoder.text_encoders.title                              MHSAAddAtt                        16.9 M
 8  news_encoder.text_encoders.title.embedding_layer              Embedding                         16.5 M
 9  news_encoder.text_encoders.title.multihead_attention          MultiheadAttention                 361 K
10  news_encoder.text_encoders.title.multihead_attention.out_proj NonDynamicallyQuantizableLinear   90.3 K
11  news_encoder.text_encoders.title.additive_attention           AdditiveAttention                 60.4 K
12  news_encoder.text_encoders.title.additive_attention.linear    Linear                            60.2 K
13  news_encoder.text_encoders.title.dropout                      Dropout                                0
14  user_encoder                                                  UserEncoder                        421 K
15  user_encoder.multihead_attention                              MultiheadAttention                 361 K
16  user_encoder.multihead_attention.out_proj                     NonDynamicallyQuantizableLinear   90.3 K
17  user_encoder.additive_attention                               AdditiveAttention                 60.4 K
18  user_encoder.additive_attention.linear                        Linear                            60.2 K
19  click_predictor                                               DotProduct                             0
20  train_rec_metrics                                             MetricCollection                       0
21  train_rec_metrics.auc                                         BinaryAUROC                            0
22  train_rec_metrics.mrr                                         RetrievalMRR                           0
23  train_rec_metrics.ndcg@10                                     RetrievalNormalizedDCG                 0
24  train_rec_metrics.ndcg@5                                      RetrievalNormalizedDCG                 0
25  val_rec_metrics                                               MetricCollection                       0
26  val_rec_metrics.auc                                           BinaryAUROC                            0
27  val_rec_metrics.mrr                                           RetrievalMRR                           0
28  val_rec_metrics.ndcg@10                                       RetrievalNormalizedDCG                 0
29  val_rec_metrics.ndcg@5                                        RetrievalNormalizedDCG                 0
30  test_rec_metrics                                              MetricCollection                       0
31  test_rec_metrics.auc                                          BinaryAUROC                            0
32  test_rec_metrics.mrr                                          RetrievalMRR                           0
33  test_rec_metrics.ndcg@10                                      RetrievalNormalizedDCG                 0
34  test_rec_metrics.ndcg@5                                       RetrievalNormalizedDCG                 0
35  test_categ_div_metrics                                        MetricCollection                       0
36  test_categ_div_metrics.categ_div@10                           Diversity                              0
37  test_categ_div_metrics.categ_div@5                            Diversity                              0
38  test_sent_div_metrics                                         MetricCollection                       0
39  test_sent_div_metrics.sent_div@10                             Diversity                              0
40  test_sent_div_metrics.sent_div@5                              Diversity                              0
41  test_categ_pers_metrics                                       MetricCollection                       0
42  test_categ_pers_metrics.categ_pers@10                         Personalization                        0
43  test_categ_pers_metrics.categ_pers@5                          Personalization                        0
44  test_sent_pers_metrics                                        MetricCollection                       0
45  test_sent_pers_metrics.sent_pers@10                           Personalization                        0
46  test_sent_pers_metrics.sent_pers@5                            Personalization                        0
Trainable params: 17.4 M
Non-trainable params: 0
Total params: 17.4 M
Total estimated model params size (MB): 69
```

Figure 6: Example for logging the model size, number of trainable and non-trainable model parameters.