nerblackbox:
A High-level Library for Named Entity Recognition in Python

Felix Stollenwerk
AI Sweden
felix.stollenwerk@ai.se

Abstract

We present *nerblackbox*, a python library to facilitate the use of state-of-the-art transformer-based models for named entity recognition. It provides simple-to-use yet powerful methods to access data and models from a wide range of sources, for fully automated model training and evaluation as well as versatile model inference. While many technical challenges are solved and hidden from the user by default, *nerblackbox* also offers fine-grained control and a rich set of customizable features. It is thus targeted both at application-oriented developers as well as machine learning experts and researchers.

1 Introduction

Named Entity Recognition (NER) is an important natural language processing task with a multitude of applications (Lorica and Nathan, 2021). While generative AI is currently ubiquitous in the scientific literature and public debate, it has not (yet) replaced discriminative AI for information extraction tasks like NER. Fine-tuned, transformer-based encoder models are both SOTA in research[1] and commonly used by developers to solve real-world problems, see e.g. (Raza et al., 2022; Stollenwerk et al., 2022). Popular open source frameworks, like the ones provided by HuggingFace (Wolf et al., 2020; Lhoest et al., 2021; Von Werra et al., 2022), greatly facilitate the use of such models. They cover the whole workflow consisting of dataset integration, model training, evaluation and inference, see Fig. 1.

Figure 1: Essential stages in the life cycle of a machine learning model.

[1] http://nlpprogress.com/english/named_entity_recognition.html

However, they do require a certain degree of expertise and often some significant, use-case specific effort. Some of the (general and NER-specific) challenges are:

(i) There exist various sources for datasets. Regarding public datasets, HuggingFace and GitHub repositories are important sources. Private datasets may be stored on local filesystems or be created using annotation tools. Additional complexity is introduced by the circumstance that datasets often come in different formats. This may be true even for datasets from the same source. These issues typically require customized data preprocessing code for every new use case.

(ii) Data for NER is processed on three different levels: tokens, words and entities. Different parts of the workflow may operate on different levels, as shown in Tab. 1. Datasets may be pre-

stage	token	word	entity
dataset		×	×
training	×	×	
evaluation			×
inference		×	×

Table 1: Overview of the data levels that the different parts of a NER model workflow can operate on.

tokenized (word level) or not (entity level). At training time, labels for tokens that are not the first token of a word may be ignored (word level) or included (token level) in the computation of the loss. Model evaluation takes place primarily on the entity level (although it is labels on the token or word level that are employed for the computation). Finally, while model predictions are often made on the entity level, some use cases may require predictions on the word level, for instance if the associated probabilities are to be used for active learning. Handling these technical intricacies requires expert knowledge.

(iii) There exists a multitude of NER-specific annotation schemes and variants and it is important to be aware of the differences. For instance, during data preprocessing, existing word or entity labels need to be mapped to token labels, which is an annotation scheme dependent process. At evaluation time, there are different ways to cope with predictions that do not obey the rules of the given annotation scheme (we will get back to this in Sec. 4.6).

(iv) Training hyperparameters which lead to reasonable performance may depend on the employed model and dataset. For instance, while a small dataset often requires more training epochs, larger datasets can usually be trained for fewer epochs.

The aim of *nerblackbox* is to provide a high-level framework which makes the usage of SOTA NER models as simple as possible. As we will see in detail in Sec. 3, it offers easy access to datasets from various sources, automated training and evaluation as well as simple but versatile model inference. It does so by hiding all technical complications from the user[2] and is targeted at developers as well as people who are not necessarily experts in machine learning or NLP. However, *nerblackbox* also allows fine-grained control over all sorts of low-level parameters and provides many advanced features, some of which we will cover in Sec. 4. This might make the library appealing also for researchers and experts.

2 Related Work

The most commonly used framework for transformer-based NLP is arguably the HuggingFace ecosystem, in particular the open source libraries *transformers* (Wolf et al., 2020), *datasets* (Lhoest et al., 2021) and *evaluate* (Von Werra et al., 2022). Another popular alternative is *spacy* (Honnibal et al., 2020).

High-level libraries that are build on top of *transformers* exist in the form of *Simple Transformers* (Rajapakse, 2019) and *T-NER* (Ushio and Camacho-Collados, 2021). *Simple Transformers* is a high-level library that covers a broad range of NLP tasks with basic support for NER. *T-NER* is specific to NER with an emphasis on cross-domain and cross-lingual model evaluation. Of all the mentioned libraries, it is arguably the most similar to *nerblackbox*. However, as will be discussed in the following sections, *nerblackbox* offers many unique and powerful features that—to the best of our knowledge—make it distinct from any existing frameworks.

3 Basic Usage

nerblackbox provides a simple API to automate each step in the life cycle of a NER model (cf. Fig. 1) using very few lines of code. It does so in terms of the following classes:

```
>> from nerblackbox import Dataset,
    Training, Model
```

A high-level overview of the involved components is shown in Fig. 2.

3.1 Dataset Integration

nerblackbox allows seamless access to datasets from the following sources: HuggingFace (HF), the local filesystem (LF), built-in datasets (BI) and annotation tools (AT)[3].

Basically, a dataset can be set up for training and evaluation like in the following example:

```
>> dataset = Dataset(
       "conll2003",
       source="HF",
   )
>> dataset.set_up()
```

While this works out-of-the-box for the sources HF and BI, some additional information needs to be provided for the sources LF and AT in order for *nerblackbox* to be able to find the data. Integrating different datasets can be challenging as they may have different formatting (even on HuggingFace) and annotation schemes. Some datasets are pretokenized and split into training/validation/test subsets, while others are not. The `set_up()` method automatically deals with these challenges and makes sure that every dataset, irrespective of the source, is transformed into a standard format[4]. Apart from downloading, reformatting, and dataset splitting (if needed), it also includes an analysis of the data. For details, we refer to the library's documentation.

[2]This is where the name *nerblackbox* stems from: The framework does not require any knowledge about internal processes and can be used as a black box by only specifying inputs (pretrained model, dataset) and using the outputs (fine-tuned model). Note that there is no direct relation to explainability.

[3]Currently, the two commonly used (open source) annotation tools *LabelStudio* (Tkachenko et al., 2020) and *Doccano* (Nakayama et al., 2018) are supported.

[4]Datasets may still have different annotation schemes (IO, BIO, BILOU), and be pretokenized or not.

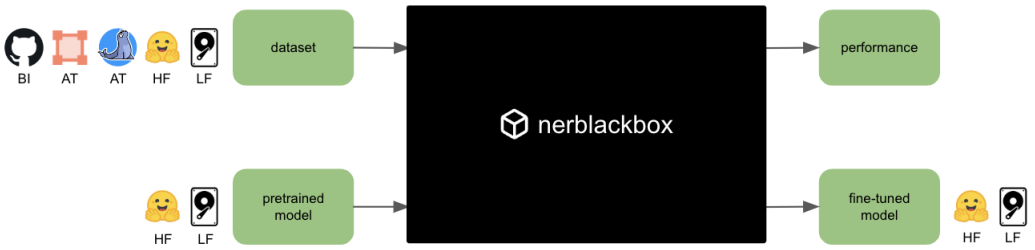

Figure 2: High-level overview of the *nerblackbox* library. It allows to easily fine-tune, evaluate and apply models for named entity recognition. The symbols to the left and right represent the sources that *nerblackbox* provides seamless access to. These are the Local Filesystem (LF), HuggingFace (HF), Annotation Tools (AT) as well as Built-in (BI) datasets that are fetched from GitHub.

3.2 Training

In order to train a model, one only needs to choose a name for the training run (for later reference) and specify the model and dataset names, like so:

```
>> training = Training(
       "my_training",
       model="bert-base-cased",
       dataset="conll2003",
   )
>> training.run()
```

In order to ensure stable results irrespective of the dataset, the training employs well-established hyperparameters by default (Mosbach et al., 2021). In particular, a specific learning rate schedule (Stollenwerk, 2022) based on early stopping and warm restarts (Loshchilov and Hutter, 2017) is used to accommodate different dataset sizes.

3.3 Evaluation

Any NER model, whether it was trained using *nerblackbox* or is taken directly from HuggingFace (HF), can be evaluated on any dataset that is accessible via *nerblackbox* (see Sec. 3.1)

```
>> model = Model.from_training(
       "my_training"
   )
>> results = model.evaluate_on_dataset(
       "conll2003",
       phase="test",
   )
>> results["micro"]["entity"]["f1"]
## 0.9045
```

The standard metrics for NER are used, i.e. precision, recall and the f1 score. Each metric is computed as a micro- and macro-average as well as for the individual classes. All metrics are determined both on the entity and word level.

3.4 Inference

Similar to evaluation, both NER models trained using *nerblackbox* and models taken directly from HuggingFace (HF) can be used for inference.

```
>> model = Model.from_training(
       "my_training"
   )
>> model.predict("The United Nations")
## [[{
##    'char_start': '4',
##    'char_end': '18',
##    'token': 'United Nations',
##    'tag': 'ORG'
## }]]
```

Apart from the predictions on the entity level for a single document shown above, *nerblackbox* also supports predictions on the word level (with or without probabilities) and batch inference. In addition, a model can be applied directly to a file containing raw data, which may be useful for inference at large scale (e.g. in production).

4 Advanced Usage

The *nerblackbox* workflow and the API are designed to be as simple as possible and to conceal technical complications from the user. However, they are also highly customizable in terms of optional function arguments, which may be particularly interesting for machine learning experts and researchers. In this section, we are going to cover a non-exhaustive selection of *nerblackbox*'s advanced features, with a slight emphasis on the training part. For further information, the reader is referred to the library's documentation.

4.1 Training Hyperparameters and Presets

While *nerblackbox* uses sensible default values for the training hyperparameters (see Sec. 3.2), one may also opt to specify them manually. In par-

ticular, all aspects of the learning rate schedule (e.g. maximum learning rate, epochs, early stopping parameters etc.) can be chosen at one's own discretion. In addition, the `Training` class offers several popular hyperparameter presets via the instantiation argument `from_preset`. Among them are the learning rate schedules from (Devlin et al., 2019) and (Mosbach et al., 2021), which may work well for larger and smaller datasets, respectively. Hyperparameters search is also supported.

4.2 Dataset Pruning

nerblackbox provides the option to only use a subset of the training, validation or test data by specifying parameters like `train_fraction`. This may be useful to accelerate the training (for instance in the development phase of a product) or if one wants to investigate the effect of the dataset size (for instance to see if the model has saturated, or for research).

4.3 Annotation Schemes

While every dataset is associated with a certain annotation scheme, *nerblackbox* provides the option to translate between schemes at training time. The desired annotation scheme can simply be specified via the training parameter `annotation_scheme`. This may be interesting for users who aim to optimize their model's performance as well as researchers who systematically want to investigate the impact of the annotation scheme.

4.4 Multiple Runs

Since the training of a neural network includes stochastic processes, the performance of the resulting model depends on the employed random seed. In order to gain control over the associated statistical uncertainties, one may train multiple models using different random seeds. With *nerblackbox*, this can trivially be done by setting the training parameter `multiple_runs` to an integer greater than 1. In that case, the evaluation metrics will be given in terms of the mean and its associated uncertainty. For inference, the best model is automatically used.

4.5 Detailed Results

nerblackbox saves detailed training and evaluation results (e.g. loss curves, confusion matrices) using *MLflow*[5] and TensorBoard. This is useful in order to keep an overview of trained models, inspect their detailed properties as well as optimize and cross-check the training process.

4.6 Careful Evaluation

A model may predict labels for a sequence of tokens that are inconsistent with the employed annotation scheme. For instance, if the BIO annotation scheme is used, the combination `O I-PER` is incorrect[6]. When translated to *entity* predictions, *nerblackbox* ignores incorrect labels by default, both at evaluation and inference time. However, the popular *evaluate* (Von Werra et al., 2022) and *seqeval* (Nakayama, 2018) libraries do take inconsistent predictions into account during evaluation. For this reason, the `evaluate_on_dataset()` method (see Sec. 3.3) returns results for both approaches.

4.7 Compatibility with *transformers*

nerblackbox is heavily based on *transformers* (Wolf et al., 2020) such that compatibility is guaranteed. In particular, the `Model` class has the attributes `tokenizer` and `model`, which are ordinary *transformers* classes and can be used as such. GPU support (i.e. automatic detection and use) is also provided through *transformers*.

5 Resources and Code Quality

nerblackbox is available as a package on PyPI[7]. The associated GitHub repository is public at `https://github.com/flxst/nerblackbox` and contains the source code as well as multiple example notebooks. A detailed documentation is provided[8]. It includes a pedagogical introduction to the library, an in-depth discussion of its features as well as docs for the python API. Consistent code syntax and typing are ensured by usage of *black*[9] and *mypy*[10], respectively. We employ unit and end-to-end testing. As an additional cross-check, numerical results from the literature are reproduced using *nerblackbox* (details can be found in the documentation).

Acknowledgements

This work was supported by Vinnova through the grants 2019-02996 and 2021-03630.

[5] `https://pypi.org/project/mlflow/`

[6] The variant of the BIO scheme which we assume here is also known as IOB2

[7] `https://pypi.org/project/nerblackbox/`

[8] `https://flxst.github.io/nerblackbox/`

[9] `https://pypi.org/project/black/`

[10] `https://pypi.org/project/mypy/`

References

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, Minneapolis, Minnesota. Association for Computational Linguistics.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python.

Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, Joe Davison, Mario Šaško, Gunjan Chhablani, Bhavitvya Malik, Simon Brandeis, Teven Le Scao, Victor Sanh, Canwen Xu, Nicolas Patry, Angelina McMillan-Major, Philipp Schmid, Sylvain Gugger, Clément Delangue, Théo Matussière, Lysandre Debut, Stas Bekman, Pierric Cistac, Thibault Goehringer, Victor Mustar, François Lagunas, Alexander Rush, and Thomas Wolf. 2021. Datasets: A Community Library for Natural Language Processing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 175–184, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Ben Lorica and Paco Nathan. 2021. 2021 NLP Survey Report.

Ilya Loshchilov and Frank Hutter. 2017. SGDR: Stochastic Gradient Descent with Warm Restarts. ArXiv:1608.03983 [cs, math].

Marius Mosbach, Maksym Andriushchenko, and Dietrich Klakow. 2021. On the Stability of Fine-tuning BERT: Misconceptions, Explanations, and Strong Baselines. ArXiv:2006.04884 [cs, stat].

Hiroki Nakayama. 2018. seqeval: A Python framework for sequence labeling evaluation.

Hiroki Nakayama, Takahiro Kubo, Junya Kamura, Yasufumi Taniguchi, and Xu Liang. 2018. doccano: Text Annotation Tool for Human.

T. C. Rajapakse. 2019. Simple Transformers.

Shaina Raza, Deepak John Reji, Femi Shajan, and Syed Raza Bashir. 2022. Large-Scale Application of Named Entity Recognition to Biomedicine and Epidemiology. Pages: 2022.09.22.22280246.

Felix Stollenwerk. 2022. Adaptive Fine-Tuning of Transformer-Based Language Models for Named Entity Recognition. ArXiv:2202.02617 [cs].

Felix Stollenwerk, Niklas Fastlund, Anna Nyqvist, and Joey Öhman. 2022. Annotated Job Ads with Named Entity Recognition. *SLTC*.

Maxim Tkachenko, Mikhail Malyuk, Andrey Holmanyuk, and Nikolai Liubimov. 2020. Label Studio: Data labeling software.

Asahi Ushio and Jose Camacho-Collados. 2021. T-NER: An All-Round Python Library for Transformer-based Named Entity Recognition. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 53–62, Online. Association for Computational Linguistics.

Leandro Von Werra, Lewis Tunstall, Abhishek Thakur, Sasha Luccioni, Tristan Thrush, Aleksandra Piktus, Felix Marty, Nazneen Rajani, Victor Mustar, and Helen Ngo. 2022. Evaluate & Evaluation on the Hub: Better Best Practices for Data and Model Measurements. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 128–136, Abu Dhabi, UAE. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. 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, Online. Association for Computational Linguistics.