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

Felix Stollenwerk

AI Sweden

Named Entity Recognition (NER)

Jim **PERSON** bought 300 **CARDINAL** shares of Acme Corp. **ORG** in 2006 **DATE** .

Importance

Named Entity Recognition is a very **common NLP task** [1]

SOTA

Fine-tuned transformer-based **encoder models** [2]

Standard library

HuggingFace **transformers** [3]

Challenges (Examples)

- Different **data sources**, different **data formats**, different **annotation schemes**
- Data for NER is processed on three different levels: **tokens, words and entities**

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

- Optimal **training hyperparameters** may depend on the employed model and dataset, especially the **dataset size**.

Why nerblackbox?

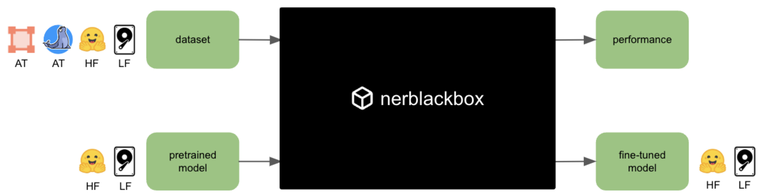
Main Features

✓ **nerblackbox** hides **technical complications** from the user
✓ **nerblackbox** requires very **little machine learning expertise**
✓ **nerblackbox** significantly **reduces the effort** to train & use SOTA NER models

Comparison to transformers

library	transformers	nerblackbox
NLP tasks	all	only NER
level	low-level	high-level
required expertise	high	low
main target group	machine learning engineers	application-oriented developers

nerblackbox in a nutshell



Specify a **dataset** and a **model** - **nerblackbox** takes care of the rest!

Sources for datasets and models

- HuggingFace (HF)
- Local Filesystem (LF)
- Annotation Tools (AT)

nerblackbox API

1. Dataset Integration

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

2. Model Training

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

3. Model Evaluation

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

4. Model Inference

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

2 lines of code for each step!

Advanced Features (Examples)

nerblackbox is highly customizable!

Training Hyperparameters

Specify individual hyperparameters or use available presets

Dataset Pruning

Use a fraction of the training, validation or test data for prototyping

Annotation Schemes

Translate between common annotation schemes at training time

Multiple Runs

Train multiple models using different random seeds to gain control over statistical uncertainties

Detailed Results

Detailed training and evaluation results (e.g. loss curves, confusion matrices) using mlflow and tensorboard

Careful Evaluation

Careful and transparent handling of annotation scheme inconsistencies

Compatibility with transformers

nerblackbox is heavily based on transformers such that compatibility is guaranteed

Links

- **Repository:** https://github.com/flxst/nerblackbox
- **Documentation:** https://flxst.github.io/nerblackbox
- **PyPI package:** https://pypi.org/project/nerblackbox

References

[1] Ben Lorica and Paco Nathan. 2021 NLP Survey Report, 2021.

[2] Sebastian Ruder. NLP-progress, February 2022.

[3] Thomas Wolf et al. 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, October 2020. Association for Computational Linguistics.