Train English and German Nemed Entity Recognition models using SpaCy v3.0 with the CoNLL-2003 Datasets

Jinghua Xu
University of Tübingen
jinghua.xu@student.uni-tuebingen.de

Abstract

This paper describes my experiments of training a series of named entity recognition models with different configurations using the recently released spaCy v3.0 [1] with the datasets distributed in the shared-task CoNLL-2003 (Sang and De Meulder, 2003). I trained English and German named entity recognition models respectively with the corresponding datasets. All experiments were conducted through command line interface. In the experiments, the benchmark English NER model, a non-transformer CNN model using the pre-trained GloVe vectors (Pennington et al., 2014) trained on CPU showed the highest F-score during training at 92.80, and 89.22 evaluated on the test data. Among the German NER models, a transformer model using no pre-trained vectors trained on CPU presented the best F-score at 87.22 during training, and 83.29 evaluated on the test data.

1 Introduction

In the 1980s, research programs aimed to develop computer systems that are capable of automatic understanding documents were launched. In order for the computer systems to be able to understand the documents, the basic building blocks i.e. the named entities need to be identified. The concept therefore first emerged, named entities are linguistic objects following the need expressed above. (Nouvel et al., 2016) In different evaluation campaigns, the typology had been defined differently, in CoNLL-2003, through which the datasets used in this experiment were distributed, phrases that contain the names of persons, organizations and locations were considered named entities. Named Entity Recognition is an important sub-task of automatic understanding documents and it can be applied to and help many natural language processing tasks such as information extraction, automatic summary, machine translation etc..

A number of systems for NER had been developed over the years, spaCy [2] is one of those that shows great performance for the purpose. spaCy is an open-source python library for NLP written in Python and Cython. It offers pre-trained models for multi-language NER including English and German, as well as allowing training and deploying custom NER models on domain specific corpuses. spaCy's pre-trained English models were trained on OntoNotes 5 corpus (Hovy et al., 2006), the German models were trained on TIGER corpus (Brants et al., 2004) and WikiNER (Nothman et al., 2013). The new release spaCy v3.0 features remarkable new transformer-based pipelines, which also presented great performance comparing to non-transformer pipelines in this experiment.

As was mentioned at the beginning, the goal of this paper is to describe the experiments of training a series of named entity recognition models using spaCy v3.0 with the CoNLL-2003 datasets in order to obtain a model with the best performance in each language then compare and evaluate. With the datasets, I trained German and English NER models. In the following sections I describe the experiments and the results.

2 Data

The NER models were trained and evaluated on the data provided in the CoNLL-2003 shared-task. Four types of named entities were concentrated: persons, locations, organizations and names of miscellaneous entities that do not belong to the previous three groups. In each language, a trainning data file, a development data file (testa) and a test data file (testb) were supplied.

[1] https://explosion.ai/blog/spacy-v3

[2] https://spacy.io/

2.1 Raw data

The raw English data is a collection of news wire articles from the Reuters Corpus (Lewis et al., 2004), consists of Reuters news stories between August 1996 and August 1997. The raw data can be obtained upon request (free of charge for academic purposes) from NIST [3]. The raw German data is a collection of articles from the Frankfurter Rundschau. The data sets were taken from articles written in one week at the end of August 1992. which could once be obtained on LDC [4], however, this approach is no longer possible. For my experiments, I obtained the German data from a repository on GitHub [5].

The unannotated data contain 17 million tokens (English) and 14 million tokens (German). Sang and De Meulder (2003) presents more insights into the number of articles, sentences, tokens and named entities in each data file.

2.2 Annotations

Manual annotation had been performed at the University of Antwerp. [6] Apart from tokenized wordforms, the data provides predicted PoS-tags and chunks.

The data format of all annotated datasets is one token per line with columns separated by whitespace. The four columns are the word, its part-of-speech tag, its chunk tag and its named entity tag. Sentences are separated by blank lines and documents are separated by the line -DOCSTART--X- O O. The tagging scheme is the IOB scheme originally put forward by Ramshaw and Marcus (1995). The named entities were assumed to be non-recursive and non-overlapping. When a named entity is embedded in another named entity, usually only the top level entity has been annotated.

LEICESTERSHIRE	NNP	I-NP	I-ORG
TAKE	NNP	I-NP	O
OVER	IN	I-PP	O
AT	NNP	I-NP	O
TOP	NNP	I-NP	O
AFTER	NNP	I-NP	O
INNINGS	NNP	I-NP	O
VICTORY	NN	I-NP	O
.	.	O	O

[3]https://trec.nist.gov/data/reuters/reuters.html

[4]https://www.ldc.upenn.edu

[5]https://github.com/MaviccPRP/ger_ner_evals

[6]More than one versions of annotations exit for the CoNLL-2003 data.

2.3 Data preprocessing

With the raw data ready, I added the annotations by running the binary file [7] provided in the shared task. The German data was first converted to utf-8 before it could be further processed in spaCy. All data files were converted into spaCy's binary data format, a serialized DocBin [8], before they could be used with the train command and other experiment management functions. The conversion was done using spaCy's convert command through command line interface.

3 Models

In each language, I trained a number of language-dependent NER models with distinctive configurations tried out in order to achieve a better F-score during training. All experiments were conducted through command line interface using spaCy's train, evaluate and other commands. With no access to GPU or Linux servers, all models were trained on CPU [9].

3.1 English Models

I trained in total eight English NER models configured with different hyperparameters, optimizers, batchers and pre-trained vectors, including a reproduction of spaCy's CoNLL-2003 English benchmark model [10]. The training of the benchmark model (cnn_glove_small) was tried both on Google Colab [11] and locally on my computer. Training on Google Colab took notably longer time therefore is not recommended.

3.1.1 Baseline model

The experiments started with the default model, which also serves as the baseline. I built the default configuration file in two steps: obtaining the base-config.cfg from spaCy's website as suggested and initializing the base configuration using the init command. The default configuartion used no pre-trained vectors, no data augmenters and hyperparameters were set to the default value. The default batcher was "spacy.batch_by_words.v1", the batch size was replaced by a comounding scheduler. The default optimizer was "Adam.v1", with 0.001 learning rate. And the default encoder

[7]https://www.clips.uantwerpen.be/conll2003/ner.tgz

[8]https://spacy.io/api/docbin

[9]It is recommended against training transformer-based models on CPU, training on GPU is normally 3-4X faster.

[10]https://github.com/explosion/projects/tree/v3/benchmarks/ner_conll03

[11]https://colab.research.google.com/

Configuration	Vectors	Data Augmentation	Notes	F-score
config_eff (baseline)	null	null	default	0.8690586745
config_acc	en_core_web_lg	null	default	0.9114201433
config_acc_aug	en_core_web_lg	en_orth_variants	default	0.9016145308
config_acc_batcher	en_core_web_lg	null	spacy.batch_by_padded.v1	0.9120554194
config_acc_optimizer	en_core_web_lg	null	SGD.v1	0.8337842408
config_acc_optimizer2	en_core_web_lg	null	RAdam.v1	0.9196466134
config_acc_optimizer_scheduler	en_core_web_lg	null	RAdam.v1 and decaying scheduler	0.9120148186
cnn_glove_small (benchmark)	GloVe vectors	null	hyperparameters [12]	0.9281870165

Table 1: English NER models and corresponding F-score during training.

was "spacy.MaxoutWindowEncoder.v2", with width set to 96, depth 4, window size 1 and maxout pieces 3. The baseline F-score durinig training evetually obatined is 0.8690586745 as shown in Table 1.

3.1.2 Following experiments

In order to improve the model performance from the default setting, in the following experiments I tried out the pre-trained vector "en_core_web_lg", data augmentation, alternative batcher "spacy.batch_by_padded.v1", alternative optimizers "SGD.v1", "RAdam.v1", and "RAdam.v1" with decaying scheduler. The training results are presented in Table 1.

It is noticeable that the use of "en_core_web_lg" vector significantly improved the model performance, while data augmentation influenced the F-score negatively in this case. The optimizer "RAdam.v1" marginally increased model's f-score during training, however, replacing the learning rate with a scheduler did not help. It is also markedly that using the optimizer "SGD.v1" during training took longer time than other optimizers, nevertheless the eventual F-score is not better.

The best model obtained in the experiments also shows remarkable performance, the F-score is fairly close to the benchmark. With all default hyperparameters and settings, a pre-trained vector used and the alternative optimizer "RAdam.v1", the F-score during training is 0.9196466134, which is only 0.1 smaller than the benchmark.

3.1.3 Benchmark model

The benchmark model used the pre-trained Stanford GloVe vectors. Comparing to the default settings, the model was configured with "spacy.TransitionBasedParser.v1", 128 hidden-width and 3 maxout pieces. It is also noticeable that the width of the maxout encoder was set to 128 instead of 96. The F-score during training as presented in Table 1 is 0.9281870165.

3.2 German Models

In the experiments, I trained in total nine German NER models, including transformer-based and non-transformer ones.

3.2.1 Baseline model

The experiments started with training the default model, which also serves as the baseline. The default configuration file was built the same way it was in the experiments of training the English NER models. The settings except for language are the same as in the defalut configuration of English NER model. The F-score during training obatained in the baseline model is 0.5605189083, which is significantly lower than that of the English baseline model (0.8690586745).

3.2.2 Following experiments

In order to increase the model performance, in the following experiments, I tried the pre-trained vectors "de_core_news_lg", data augmentation, "spacy.TransitionBasedParser.v1", an alternative batcher "spacy.batch_by_padded.v1" and an alternative optimizer "RAdam.v1".

The configurations and corresponding F-scores during training are shown in Table 2. It is notable that the use of pre-trained vectors largely improved model performance, while data augmentation decreased the F-score. Both "spacy.TransitionBasedParser.v1" and "spacy.batch_by_padded.v1" had marginal effect on improving the model performance. Meanwhile, the alternative optimizer "RAdam.v1" increased the F-score during training. The observations are in general consistent with that in the experiments of training English NER models.

3.2.3 Transformer-based model

The above alternatives tried out strengthened the model performance to different extents,

[11] NER model hiden width, encoder width, depth etc.

Configuration	Vectors	Data Augmentation	Notes	F-score
defualt_config (baseline)	null	null	null	0.5605189083
config3	null	de_orth_variant	null	0.550796198
config_vec_corr	de_core_news_lg	null	null	0.747032967
config_vec_aug	de_core_news_lg	de_orth_variant	null	0.7319055464
cnn_small	null	null	spacy.TransitionBasedParser.v1	0.5803638176
cnn_vec_small	de_core_news_lg	null	spacy.TransitionBasedParser.v1	0.7486881768
vec_batcher	de_core_news_lg	null	spacy.batch_by_padded.v1	0.7491386018
vec_radam	de_core_news_lg	null	RAdam.v1	0.7651750111
config_transformer[13]	null	null	transformer	0.8721599832

Table 2: German NER models and corresponding F-scores during training.

however none was substantial. So far the models trained in the expriments were all non-transformer, therefore, on top of which I went on the experiment with training a tranformer-based (Wolf et al., 2019) NER model. Transformers are a family of neural network architectures that compute dense, context-sensitive representations for the tokens in documents. spaCy v3.0's transformer support allows one achieve higher accuracy in exchange for higher training and runtime costs. [14] The F-score obtained during training is 0.8721599832, which is a consequencial improvemeent from non-transformer based models.

4 Evaluation

I evaluated the best English and German NER models obtained in the experiments as well as the English benchmark model with the respective test data (testb). The metrics were calculated using spaCy's evaluate command [15] through command line interface. Given the nature of the CoNLL-2003 data that instead of actual text only tokens are provided, the –gold-preproc flag is set in the evaluate command in order for spaCy to train with no actual text available. This also makes it easier to compare the results against other systems. The precision, recall, and F-score of each model are presented in Table 6.

As shown in the table, both English models (the best working model obtained in the experiments and the benchmark model) outperform the best working German model according to the F-score. Meanwhile, it is notable that the non-transformer English models presents higher recall while the transformer-based German model has better precision.

Table 3, Table 4 and Table 5 presents the evaluation metrics per entity type of the English benchmark model, the best working English and German models obtained in the experiments. In each model, the best recognized type is different.

Type	Precision	Recall	F-score
LOC	91.16	92.75	91.95
PER	93.45	93.51	93.48
ORG	87.18	85.97	86.57
MISC	79.34	78.77	79.06

Table 3: Evaluation results per type of the English benchmark model.

Type	Precision	Recall	F-score
PER	80.24	90.17	84.92
LOC	89.91	91.91	90.90
ORG	83.47	82.06	82.76
MISC	74.33	78.77	76.49

Table 4: Evaluation results per type of the best working English NER model obtained in the experiments.

Type	Precision	Recall	F-score
LOC	84.41	83.19	83.80
ORG	79.59	71.15	75.14
PER	92.93	93.56	93.24
MISC	72.71	74.78	73.73

Table 5: Evaluation results per type of the best working German NER model obtained in the experiments.

5 Conlusions

In this paper, I demontrated my experiments of training a number of language-dependent English and German NER models using spaCy v3.0

[12] spaCy does not provide a benchmark model in German NER, the transformer model in this experiment should approach close to benchmark.

[14] In this experiment, the training of the transformer models took as long as 48 hours. Although training on GPU can be 3-4X faster. With no access to GPU, all experiments were done using CPU.

[15] With the "-dp" flag set in command, the parsed output are also rendered into HTML, in which each recognized entity gets marked in the color that indicates the category of an entity.

Language	Model	Precision	Recall	F-score
English	cnn_glove_small (benchmark)	89.20	89.24	89.22
English	config_acc_optimizer2	83.18	86.88	84.99
German	transformer	85.08	82.30	83.67

Table 6: Evaluation results of the best working English and German NER Models obtained in the experiments as well as the English benchmark model.

through command line interface with the CoNLL-2003 data. My experimental process includes data pre-processing, model training and model evaluation. In total eight configurations including a reproduction of the benchmark were experimented for the English models and nine including transformer-based and non-transformer based configurations were experimented for the German models. I picked the best working models according to the F-score during training for model configurations. In general, English models present higher F-scores comparing to the German models, the transformer-based model presents exceeding performance compared to the non-transformer models in German. The F-scores obtained in the evaluation of the best working and benchmark models approach the current state-of-the-art, however, other systems and methods such as Flair (Akbik et al., 2018), Stanza (Qi et al., 2020), and CNN-bidirectional LSTM-CRF (Ma and Hovy, 2016) show better performance with the same data, meanwhile spaCy performs better with other datasets such as OntoNotes. Since in this experiment only one transformer-based model, which is one of the main new features in SpaCy v3.0, was trained and it presents outstanding performance, future NER model training experiments can investigate further in spaCy's transformer-based models, and even other transformer-based systems for the NER task.

References

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th international conference on computational linguistics*, pages 1638–1649.

Sabine Brants, Stefanie Dipper, Peter Eisenberg, Silvia Hansen-Schirra, Esther König, Wolfgang Lezius, Christian Rohrer, George Smith, and Hans Uszkoreit. 2004. Tiger: Linguistic interpretation of a german corpus. *Research on language and computation*, 2(4):597–620.

Eduard Hovy, Mitch Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. Ontonotes: the 90% solution. In *Proceedings of the human language technology conference of the NAACL, Companion Volume: Short Papers*, pages 57–60.

David D Lewis, Yiming Yang, Tony Russell-Rose, and Fan Li. 2004. Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research*, 5(Apr):361–397.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany. Association for Computational Linguistics.

Joel Nothman, Nicky Ringland, Will Radford, Tara Murphy, and James R Curran. 2013. Learning multilingual named entity recognition from wikipedia. *Artificial Intelligence*, 194:151–175.

Damien Nouvel, Maud Ehrmann, and Sophie Rosset. 2016. *Named entities for computational linguistics*. Wiley Online Library.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D Manning. 2020. Stanza: A python natural language processing toolkit for many human languages. *arXiv preprint arXiv:2003.07082*.

Erik F Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. *arXiv preprint cs/0306050*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.