Evaluation of a Sequence Tagging Tool for Biomedical Texts

Julien Tourille[1,6], Matthieu Doutreligne[1,2], Olivier Ferret[3], Nicolas Paris[1,2,4], Aurélie Névéol[1], Xavier Tannier[4,5]

[1]LIMSI, CNRS, Université Paris-Saclay, [2]WIND-DSI, AP-HP, [3]CEA, LIST, [4]Sorbonne Université, [5]Inserm, LIMICS, [6]Univ. Paris-Sud

{julien.tourille,matthieu.doutreligne,aurelie.neveol}@limsi.fr, olivier.ferret@cea.fr, nicolas.paris@aphp.fr, xavier.tannier@sorbonne-universite.fr

Abstract

Many applications in biomedical natural language processing rely on sequence tagging as an initial step to perform more complex analysis. To support text analysis in the biomedical domain, we introduce Yet Another SEquence Tagger (YASET), an open-source multi purpose sequence tagger that implements state-of-the-art deep learning algorithms for sequence tagging. Herein, we evaluate YASET on part-of-speech tagging and named entity recognition in a variety of text genres including articles from the biomedical literature in English and clinical narratives in French. To further characterize performance, we report distributions over 30 runs and different sizes of training datasets. YASET provides state-of-the-art performance on the CoNLL 2003 NER dataset (F1=0.87), MEDPOST corpus (F1=0.97), MERLoT corpus (F1=0.99) and NCBI disease corpus (F1=0.81). We believe that YASET is a versatile and efficient tool that can be used for sequence tagging in biomedical and clinical texts.

1 Introduction

Many applications in biomedical Natural Language Processing (NLP), including relation extraction or text classification, rely on sequence tagging as an initial step to perform more complex analysis. To support text analysis in the biomedical domain, we present Yet Another SEquence Tagger (YASET), an open-source multi-purpose sequence tagger written in Python. YASET aims at providing NLP researchers with fast and accurate implementations of cutting-edge deep learning sequence tagging models. YASET is built using TensorFlow (Abadi et al., 2015), an open source software library for numerical computation. The code is licensed under the version 3 of the *GNU General Public License*[1] and is freely available online[2]. The main contributions of this work are:

- a fast and accurate implementation of a state-of-the-art sequence tagging model based on Long Short-Term Memory Networks (LSTMs) (Hochreiter and Schmidhuber, 1997). The architecture is similar to the one described in Lample et al. (2016) and is able to process mini-batches for faster training. Furthermore, YASET supports the use of handcrafted features in combination with word and character embeddings;
- an easy-to-use interface based on a central configuration file that is used to setup experiments, with default parameters that are suitable for most sequence tagging tasks;
- an evaluation on various biomedical corpora and on the CoNLL 2003 corpus, studying the stability of our model and the effect of training data size. We compare YASET performance with state-of-the-art results published in the literature.

2 Related Work

Several open-source implementations for sequence tagging based on neural network architectures have become available over the last years. Lample et al. (2016) provide a Python implementation[3] for the two models presented in their paper. They are implemented with Theano (Al-Rfou et al., 2016), a Python library for deep learning.

[1]https://www.gnu.org/licenses/gpl-3.0.en.html
[2]https://github.com/jtourille/yaset
[3]https://github.com/glample/tagger

NeuroNER[4] (Dernoncourt et al., 2017) targets non-expert users and is based on Lample's Bi-LSTM model. The authors intended to make the tool easy to use by providing automatic format conversion from the brat format (Stenetorp et al., 2012) to the input format and from the output format to the brat format. The tool produces several plots during training for performance analysis. It is implemented in Python and makes use of the TensorFlow library. Both implementations of the Bi-LSTM model suffer from a very long training time which makes them cumbersome to use. YASET offers a faster implementation of the model by allowing mini-batch training and by using the pipeline API of TensorFlow.

Rei and Yannakoudakis (2016) released a Python implementation[5] of different models presented in their works (Rei and Yannakoudakis, 2016; Rei et al., 2016; Rei, 2017). One major difference with YASET resides in the possibility to use a language modeling objective during training.

Recently, Yang and Zhang (2018) introduced NCRF++[6], a tool presented as *the neural version of CRF++*[7] and implemented with PyTorch (Paszke et al., 2017). The tool is very close to YASET, with the possibility to define handcrafted word features and to perform *nbest* decoding.

Another type of neural network model, based on Convolutional Neural Networks (CNNs) for character level representation, is presented in Ma and Hovy (2016). The authors implemented the architecture with PyTorch. The code is freely available online[8]. The same type of architecture is implemented in the tool released by Reimers and Gurevych (2017). It is implemented with Keras (Chollet et al., 2015) and is freely available online[9].

Outside the peer-reviewed scientific environment, many other implementations are freely available online. However, we will not review them in this paper.

[4]https://github.com/Franck-Dernoncourt/NeuroNER

[5]https://github.com/marekrei/sequence-labeler

[6]https://github.com/jiesutd/NCRFpp

[7]https://taku910.github.io/crfpp

[8]https://github.com/XuezheMax/NeuroNLP2

[9]https://github.com/UKPLab/emnlp2017-bilstm-cnn-crf

3 Neural Network Model

There is currently one neural network model implemented in YASET. This model is mostly based on Lample et al. (2016). However, similar architectures are presented in other work (Collobert et al., 2011; Ma and Hovy, 2016; Rei and Yannakoudakis, 2016; Rei et al., 2016). Other network architectures will be implemented in the future.

3.1 Main Component

Our approach relies on LSTMs. The architecture of our model is presented in Figure 1. For a given sequence of tokens, represented as vectors, we compute representations of left and right contexts of the sequence at every token. These representations are computed using two LSTMs (forward and backward LSTM in Figure 1).

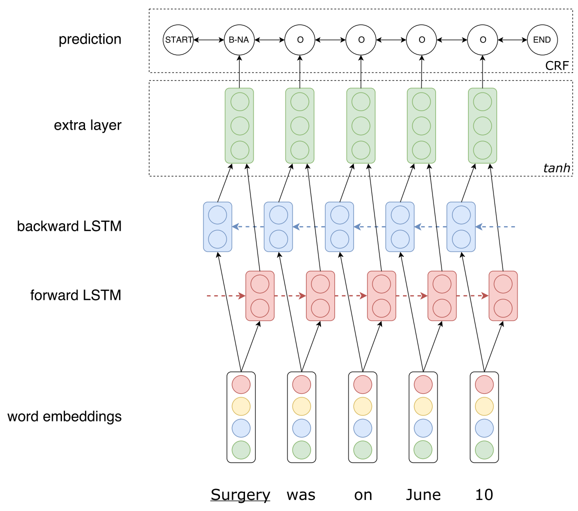

Figure 1: YASET neural network architecture overview. The example is extracted from the THYME corpus (Styler IV et al., 2014) used during the shared tasks Clinical TempEval (Bethard et al., 2015, 2016, 2017). One of the objectives was to extract medical events. In the example, the event *Surgery* is marked as a medical event with the type *N/A*.

These representations are concatenated and passed through a $tanh$ activation layer whose size is equal to the size of the concatenated vector.

Finally, the output of the last layer is linearly projected to a n-dimensional vector representing the number of categories. Following previous work (Dernoncourt et al., 2017; Lample et al., 2016; Ma and Hovy, 2016), we add a final Conditional Random Field (CRF) layer to take into account the previous label during training and prediction.

3.2 Input Embeddings

Vectors representing tokens are built by concatenating a character-based embedding, a word embedding and one embedding per feature.

An overview of the embedding computation is presented in Figure 2. Following Lample et al. (2016), the character-based representation is computed with a Bi-LSTM whose parameters are defined by users. First, a random embedding is generated for every character in the training corpus. Token characters are then processed with a forward and backward LSTM architecture. The final character-based representation results from the concatenation of the forward and backward representations. The character-based representation is then concatenated to a pre-trained word-embedding and one embedding per feature. We apply dropout on the final input embedding.

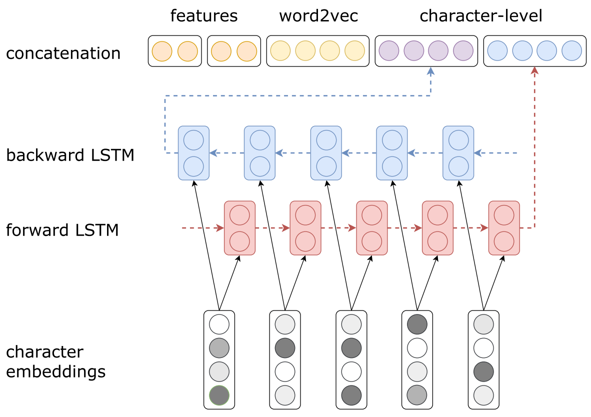

Figure 2: YASET input embedding computation overview. Embeddings result from the concatenation of a pre-trained word embedding, a character-level representation of the token and one embedding per categorical feature.

4 Tool Overview

In this section, we present a general overview of YASET. First we describe input data formats (sequences and pre-trained word embeddings). Then we present the input pipeline, the network training phase, the implemented evaluation metrics and the management of its parameters.

4.1 Input and Output Data

YASET takes CoNLL-like[10] formatted files as input. Sequences are separated by empty lines and there must be one token per line. For each token, the first and last columns are reserved namely for the token itself and the token label. Each token must have a label.

Users can add several categorical features. Feature columns must be specified in the configuration file by providing their indexes. Besides the first and last columns, and the feature columns, users can add other columns. They will be ignored by the system.

YASET can take training and development files as input but can also create development instances if there is no development file provided by users. In this case, users can specify the train/dev split ratio and may specify a random seed for experiment reproducibility.

Train and development instances consistency is checked upon startup. Each label and feature values from the development instances must appear in the train instances. An example of YASET input file format is presented in Figure 3.

In prediction mode, test data is supplied in the same format, without the token label column, which will be added with the predicted labels.

```
...
Lien NNP I-NP I-PER
. . O O

China NNP I-NP I-LOC
says VBZ I-VP O
time NN I-NP O
right RB I-ADVP O
for IN I-PP O
Taiwan NNP I-NP I-LOC
talks NNS I-NP O
. . O O

BEIJING VBG I-VP I-LOC
1996-08-22 CD I-NP O
...
```

Figure 3: Example extracted from the CoNLL 2003 shared task corpus (Sang and Meulder, 2003). For each token (col. #1), there is a label (col. #4, IOB format) and two categorical features (cols. #2 and #3), the *part-of-speech tag* and a *chunk label* (IOB format).

4.2 Word Embeddings

YASET supports embeddings in the word2vec (Mikolov et al., 2013b) or Gensim (Řehůřek and Sojka, 2010) formats. Other formats of embeddings, for instance FastText (Bojanowski et al., 2017) or Glove (Pennington et al., 2014), must first be converted to either accepted format.

[10]http://universaldependencies.org/docs/format.html

Out Of Vocabulary (OOV) tokens can be addressed by two strategies. In the first one, users provide a vector for OOV tokens. In this case, the vector is expected to be part of the provided word embedding matrix and users must specify the vector ID. In the second strategy, we follow the methodology described in Lample et al. (2016). We insert a randomly initialized vector in the embedding matrix that will be used for OOV tokens. Then we replace singletons in the training corpus by the OOV token with a probability of p, which is defined by users. Word embeddings can be fine-tuned during training in both cases by setting the appropriate flag to *true* in the configuration file.

4.3 Input Pipeline

Minimizing the memory footprint was one of the core objectives during the development of the tool. YASET leverages the pipeline API of TensorFlow to build a lightweight and fast input pipeline. Input instances are stored in the binary TensorFlow format. This avoids the need to store the whole dataset in memory. Mini-batches are extracted randomly and fed to the network.

Training instances can be bucketized according to their lengths. This possibility can speedup training with large corpora. Buckets boundaries are automatically computed. To assert effective randomization during training, buckets will contain enough instances for several mini-batches.

4.4 Neural Network Training

YASET alternates between two phases during training. In the *iteration phase*, mini-batches are extracted from the input pipeline and fed to the neural network model. Optimization is performed according to the parameters set by users in the configuration file.

At the end of each iteration, YASET enters the *evaluation phase* where the current model state is used to make predictions on the development instances. The performance score is logged for further analysis and a snapshot of the model is kept if the performance score is the best obtained so far on the development instances. Since model snapshots can take a lot of memory, we only keep the best one, i.e. the one that performs best on the development instances.

Network training is performed via back-propagation. Users can select the neural network model architecture (only one available for now), the maximum number of iterations n and the patience criterion p. Training will stop if the maximum number n is reached or if there are p iterations without performance improvement on the development instances.

Users can also set several parameters related to the learning algorithm such as the initial learning rate, and gradient clipping and exponential decay factors. Finally, users can select the mini-batch size, the number of CPU cores to use and the evaluation metric.

Another set of parameters are related to the neural network model. These parameters allow to select the dropout rate and the different hidden layer and embedding sizes.

4.5 Metrics

Model performance is measured on the development instances at the end of each iteration. Two metrics are currently implemented in YASET: *accuracy* and *CoNLL*. The former measures the fraction of correctly predicted labels among all the predicted labels. The latter is an entity-based metric which outputs precision, recall and F1-measure scores. Precision measures the fraction of correctly predicted entities among all predicted entities. Recall assesses the fraction of correctly predicted entities among all the entities that should have been detected. F1-measure is the harmonic mean of precision and recall. The implementation of the CoNLL metric is inspired by the official evaluation script used during the CoNLL-2003 shared task (Sang and Meulder, 2003) and by the Python portage of the script by Sampo Pyysalo[11]. In the case of performance evaluation with the CoNLL metric, token labels must follow either the IOB, IOBE or IOBES tagging scheme[12].

4.6 Parameters

YASET parameters are fully customizable and centralized in one configuration file which is used to setup experiments. YASET also targets end-users from a broader community by providing hyperparameter value suggestions and insights on how to choose them for various sequence-tagging situations.

[11]https://github.com/spyysalo/conlleval.py

[12]Inside, Outside, Begin, End, Single

5 Experiments

We demonstrate the performance of YASET by applying the model to four different corpora selected to cover a variety of languages, text genres, sequence types and annotation densities. We focus our effort on biomedical texts, using MedPost (Smith et al., 2004), a corpus of biomedical abstracts annotated with Part-of-Speech (PoS) tags, the NCBI Disease corpus (Islamaj Dogan and Lu, 2012), a dataset of biomedical abstracts annotated with disease related entities and MERLoT (Campillos et al., 2018), a corpus of clinical documents written in French which were annotated with two different Named Entity Recognition (NER) tag sets (Protected Health Information (PHI) and biomedical entities).

We also apply YASET on the CoNLL 2003 English NER corpus (Sang and Meulder, 2003), a classic benchmark corpus for NER in the general domain.

5.1 Presentation of the Corpora

We use the corpus partition provided with the dataset distributions when available. For NCBI and CoNLL 2003, models are trained on the train sets and evaluated on the test sets while the development sets are used to determine early stopping. For MERLoT and MedPost, partition details are outlined below. The code used to preprocess and analyze the corpora is available online[13].

The MedPost corpus is a collection of tokenized MEDLINE abstracts annotated with PoS tags. It contains 6,701 sentences (≈182,000 tokens). The corpus is divided in 13 files of different sizes, from which we extracted one file to serve as development set (*tag_mb.ioc*). The tag set contains originally 63 unique entities that we grouped in a coarser set of 51 entities. Grouping affected punctuation signs, which were assigned a unique PUNCT tag.

The NCBI corpus contains 793 MEDLINE abstracts with 11,350 annotations of diseases and disease modifiers. There are 4 different entities. Train and development sets contain 6,594 sentences (≈151,000 tokens).

MERLoT is a French clinical corpus with several levels of annotations. It contains 500 medical reports with a total of 25,087 sentences (≈177,000 tokens). We used it for two tasks. The first one is de-identification with a tag set comprising 11 types of PHI (e.g. *names*, *dates*). The second one is medical NER with 19 entity classes (e.g. *anatomy*, *disorder*).

The medical entity annotations include nested entities. Because this study aims to experiment on a variety of datasets using the same model, we did not attempt to extract several levels of nested entities. When nested entities occur, our experiments only addressed the outer layer corresponding to the largest entity. For example, the mention "cancer du sein" (*breast cancer*) was originally annotated with one DISORDER entity (cancer du sein, *breast cancer*) and one ANATOMY entity (sein, *breast*). Nested entity removal reduced the annotations to only one tag per token and in this case, the DISORDER annotation was kept while the ANATOMY annotation was removed. In total, 5,218 entities (8.4% of the complete set) were pruned. For our experiments, we also removed the headers and footers of the documents, which were not available to annotators working towards the gold standard and could cause ambiguities with some entity classes (e.g. *person* or *hospital*). This results in a smaller corpus, compared to the corpus used for de-identification experiments (≈123,000 tokens).

The CoNLL 2003 corpus is a common dataset for evaluating NER algorithms. It is based on the Reuters corpus (Lewis et al., 2004) and is the only dataset outside the biomedical domain used in our experiments. The training and development set together contain 18,451 sentences (≈256,000 tokens) annotated with 4 unique entities.

A detailed overview of the corpus characteristics is available in Table 1.

5.2 Experimental Setup

In this section, we present the experimental setup used for our experiments. First, we describe how we selected the hyper-parameters. Then, we detail the pre-trained word embeddings that were used in this work. Finally, we present the neural network training parameters.

5.2.1 Hyper-parameter Selection

We used Hyperopt[14] (Bergstra et al., 2011, 2013) to select a set of hyper-parameters that performed well on the MERLoT de-identification dataset. Due to heavy computation times, we re-used this

[13]https://github.com/strayMat/bio-medical_ner

[14]http://hyperopt.github.io/hyperopt/

Corpus	# sent.	# tok.	# ann.	Entities
Name: MedPost **Task**: POS **Domain**: MEDLINE abstracts	6,701	182,319	182,319 (100%)	51 POS tagging detailed in Smith et al. (2004)
Name: NCBI disease **Task**: NER **Domain**: MEDLINE abstracts	7,279	151,005	11,350 (7.5%)	DiseaseClass, SpecificDisease, CompositeMention, Modifier
Name: MERLoT medical **Task**: NER **Domain**: Medical reports from the Hepato-gastro-enterology and Nutrition ward	5,137	123,942	56,680 (46%)	Concept_Idea, MedicalProcedure, Hospital, Persons, Temporal, BiologicalProcessOrFunction, Devices, Measurement, Disorder, Aspect, Chemicals_Drugs, Dosage, SignOrSymptom, Anatomy, Localization, Livingbeings, Strength, AdministrationRoute, Drugform
Name: MERLoT de-identification **Task**: NER **Domain**: Same as MERLoT medical	25,599	177,158	31,723 (18%)	first name, last name, initials, address, zip code, town, date, hospital, identifier, phone number, email
Name: CoNLL 2003 **Task**: NER **Domain**: News articles from the Reuters corpus	18,451	256,145	42,646 (17%)	PER, ORG, LOC, MISC

Table 1: Overview of the corpora. The number of annotations corresponds to the number of annotated tokens to be comparable to the size of the corpus measured in number of tokens

set of hyper-parameters on the other datasets. Parameter search space includes the number of units of the character Bi-LSTM, the number of units of the main Bi-LSTM, the dimension of the character embeddings and the dropout rate. The retained setup uses character embeddings of size 24, 32 units for the character-level Bi-LSTM, 64 units for the main Bi-LSTM and a dropout rate of 0.5.

5.2.2 Embeddings

Pre-trained word embeddings were shown to boost the performance in various NLP tasks (Collobert et al., 2011; Mikolov et al., 2013a) and specifically in NER (Lample et al., 2016; Dernoncourt et al., 2017).

For each corpus, we used pre-trained word embeddings created with datasets that were consistent in genre and domain with the corpora used in this work. All word embedding models were computed using word2vec. For CoNLL 2003, we trained a model on Google News. For NCBI disease and MedPost, which both comprise abstracts from biomedical articles, we trained a model on the PubMed corpus. For MERLoT, we trained a model on the corpus from which MERLoT is extracted. It contains about 138,000 clinical notes written in French (Campillos et al., 2018).

For each of these word vector models, we computed low-dimensional representations by applying Principal Component Analysis (PCA) on the original high-dimensional word vectors (300 dimensions). During hyper-parameter search, we varied the dimension of the word embeddings and observed important impacts on the performance (up to 2.5 points of F1). We finally picked word vectors of dimension 25 (reduced by PCA), which showed to be the most efficient. This choice improved the scores as well as diminished drastically the computation times.

We also ran a few experiments using FastText (Bojanowski et al., 2017) embeddings trained on Wikipedia. Preliminary results on CoNLL 2003 show a major performance improvement. An analysis of the impact of the word embedding model on performance will be conducted in following work.

5.2.3 Training

In all our experiments, the network was trained using the Adam optimizer (Kingma and Ba, 2015) with an initial learning rate of 0.001. Early-stopping was set to 10 epochs.

6 Results

In this section, we present the performance obtained on the corpora using the experimental setup described in the previous section. First, we report corpus-specific results. Then, we study the impact of the corpus size on performance.

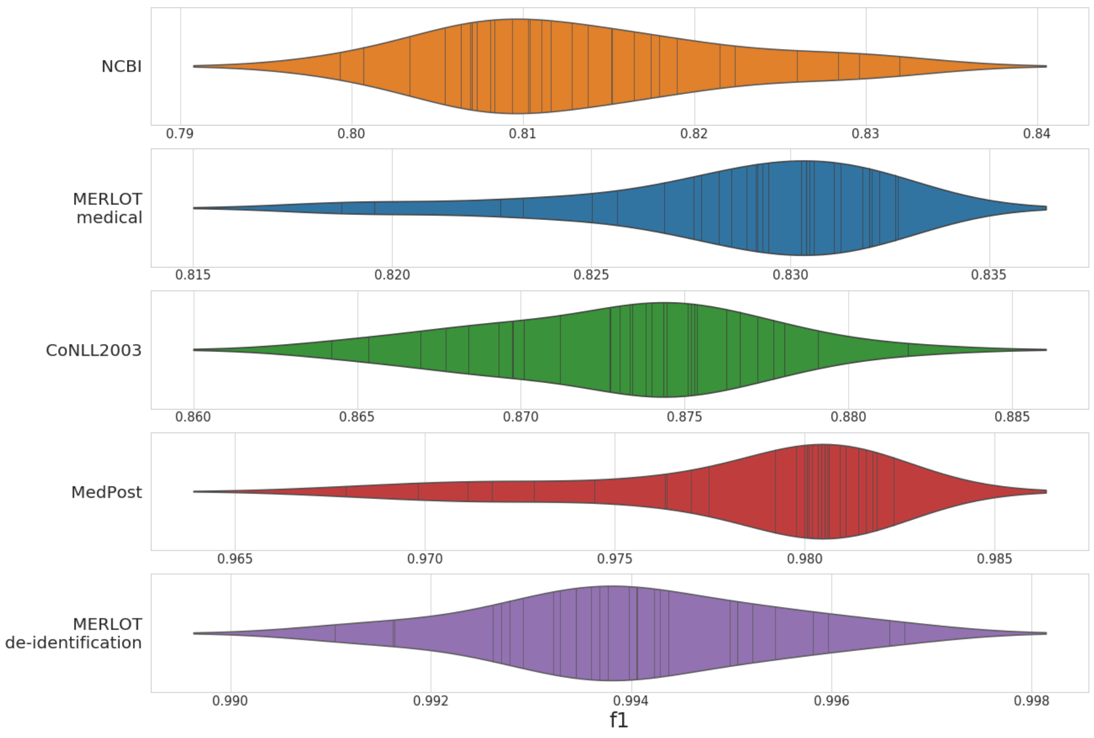

Figure 4: Distributions of F1 scores for the four corpora over 30 trainings for each corpus. Vertical lines refer to each sample. The scales are specific to each dataset for a proper visualization of each distribution.

6.1 Corpus Specific Results

Reimers and Gurevych (2017) report that neural network model training is highly non-deterministic and is subject to the random seed choice. Because of this variability during the training phase, it is crucial to report results on numerous trainings. Therefore we ran 30 experiments for each task presented in this work. We report F1-score statistics in Table 2. We also plot F1-score distributions for each task in Figure 4.

Our experiments show that performance varies across datasets, reflecting the heterogeneous difficulties of the tasks, inherent to the nature of the labels and the quality of the annotations. We notice that the standard deviations are similar for all NER tasks. It suggests that the variability of the score is independent from the task and mainly due to the model architecture. Previous work performances are reported in Table 3 for every dataset.

Task	Dataset	Mean F1	σ	Max. Diff.
NER	NCBI disease	81.33	0.83	3.27 (4.1%)
NER	MERLoT medical	82.87	0.36	1.40 (1.7%)
NER	CoNLL 2003	87.31	0.41	1.76 (2.0%)
POS	MedPost	97.83	0.39	1.44 (1.5%)
NER	MERLoT de-identification	99.40	0.14	0.568 (0.57%)

Table 2: Performance (%) of YASET on the 4 datasets.

Although our model does not show state-of-the-art performances for all of them, it obtains competitive results, demonstrating the generalization ability of the selected shared set of hyper-parameters and embeddings.

Dataset	Model	F1
NCBI	Dang et al. (2018)	**84.41**
	Islamaj Dogan and Lu (2012)	81.80
	This paper	81.33
MERLoT med.[a]	Campillos et al. (2018)	81.40
	This paper	**82.87**
CoNLL 2003	Lample et al. (2016)	90.94
	Ma and Hovy (2016)	91.21
	Yang and Zhang (2018)	91.35
	Peters et al. (2018)	**92.22**
	This paper	87.31
MedPost[a]	Smith et al. (2004)	97.43
	This paper	**97.83**
MERLoT PHI	Grouin and Névéol (2014)	94.00
	This paper	**99.40**

[a] Corpora where the experimental set-up differed between our experiments and that of prior work. For MEDPOST, we used 51 categories instead of 63; for MERLoT med. we removed nested entities.

Table 3: State of the art results obtained in prior work on the datasets.

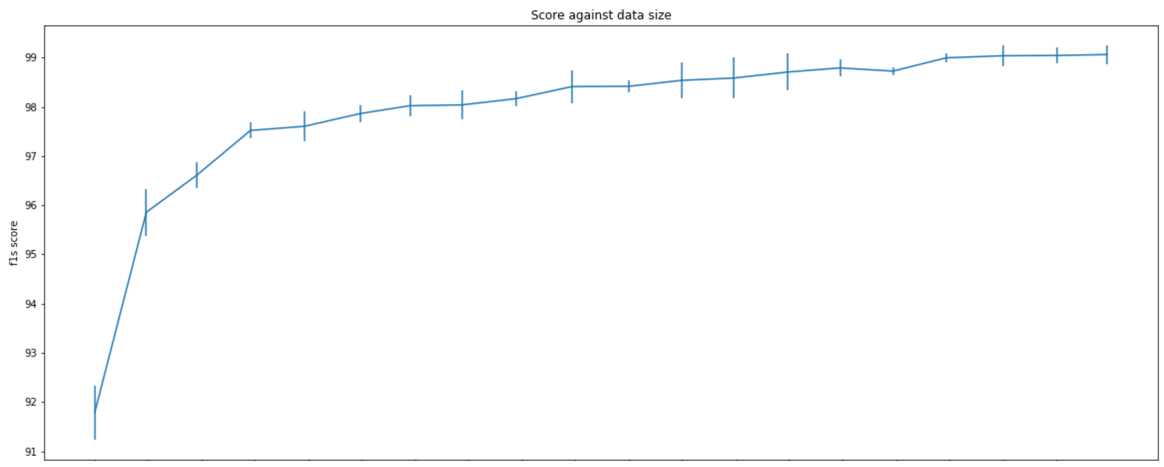

Figure 5: Effect of the training set size on the F1 score for the MERLoT dataset with de-identification annotations. These scores are computed over 6 training iterations per chunk. Vertical bars show the standard deviation σ.

6.2 Performance According to Training Data Size

The development of a labeled dataset for training annotation models is often a heavy investment in time and resources. Having some insight on the performance of the model for different training data sizes is crucial. Thus, we investigate the impact of this parameter on model performance. Focusing on the de-identification task of MERLoT, we split the training set into 20 subsets of equal sizes. Then, we built sub-training sets of growing size, that we refer as *chunks*. This resulted in 20 chunks where small chunks are subsets of the bigger one. Finally, we train YASET 6 times on each chunk, measuring how the performance improves with the size of the chunks. Figure 5 shows the progression of the F1-score according to the number of tokens. Table 4 presents the model performance according to the dataset size.

Dataset	5%	10%	25%	50%	100%
MERLoT PHI	91.79 (0.55)	95.86 (0.47)	97.60 (0.30)	98.41 (0.33)	99.06 (0.19)

Table 4: Performance (F1) against train set size (as percentage of the total training set indicated on the first row). Standard errors appear between parentheses.

We observe that the performance improves logarithmically with every chunk added in the training data as shown in Table 4. This finding is similar to the observation of Sun et al. (2017) for vision tasks. Further addition of data will slightly improve the performance as the maximum performance plateau is almost reached.

7 Conclusion and Future work

We propose an easy-to-use annotation tool implementing a state-of-the-art Bi-LSTM-CRF neural architecture. We apply the tool to PoS tagging and NER on clinical, biomedical and general domain texts. By running multiple experiment for each dataset, we confirm previous observations that neural network training is highly nondeterministic and dependent on the random seed choice.

Although we did not search for optimal hyper-parameters for each task, we obtain high performance in all our experiments, suggesting that tailoring hyper-parameters for a specific task improves only lightly the performance of the model and the neural network architecture implemented in YASET is robust with regards to the performance obtained.

Concerning the impact of the training dataset size on model performance, we confirm the intuition that adding more data allows to improve the performance of neural network models. However, result analysis suggests that the growth is logarithmic with the training data size.

One limitation of our model is its inability to directly handle nested entities such as those found in the MERLoT medical dataset and commonly used in the biomedical and clinical domains. When filtering these nested entities, specific classes are heavily impacted, including anatomy, disorder, localization, and medical procedure entities. Several strategies have been proposed to handle such cases. Campillos et al. (2018) present a 3-layer CRF model that annotates different nonoverlapping clinical entities at each layer. Ju et al. (2018) present a dynamic end-to-end neural network model capable of handling an undetermined number of nesting levels. Katiyar and Cardie (2018) model the task as an hypergraph whose structure is learned with an LSTM network.

Future research will focus on the influence of word embedding models which were shown to significantly impact on performance. Specifically, models taking into account sub-token information (Bojanowski et al., 2017) or emphasizing context (Peters et al., 2018) should be further explored. Moreover, other neural network models for NER such as the ones proposed by Rei et al. (2016) and Ma and Hovy (2016) will be investigated and implemented in YASET. Having a centralized implementation of different NER models will allow us to compare their performances on various corpora.

Acknowledgements

The authors thank the Biomedical Informatics Department at the Rouen University Hospital for providing access to the LERUDI corpus for this work. This work was supported in part by the French National Agency for Research under grant CABeRneT ANR-13-JS02-0009-01 and by Labex Digicosme, operated by the Foundation for Scientific Cooperation (FSC) Paris-Saclay, under grant CÔT.

References

Martín Abadi, Ashish Agarwal, Paul Barham, et al. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems.

Rami Al-Rfou, Guillaume Alain, Amjad Almahairi, et al. 2016. Theano: A Python framework for fast computation of mathematical expressions. *CoRR*.

James Bergstra, Daniel Yamins, and David Cox. 2013. Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures. In *Proceedings of the 30th International Conference on Machine Learning*, pages 115–123.

James S. Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2011. Algorithms for Hyper-Parameter Optimization. In *Advances in Neural Information Processing Systems 24*, pages 2546–2554.

Steven Bethard, Leon Derczynski, Guergana Savova, James Pustejovsky, and Marc Verhagen. 2015. SemEval-2015 Task 6: Clinical TempEval. In *Proceedings of the 9th International Workshop on Semantic Evaluation*, pages 806–814. Association for Computational Linguistics.

Steven Bethard, Guergana Savova, Wei-Te Chen, et al. 2016. SemEval-2016 Task 12: Clinical TempEval. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, pages 1052–1062. Association for Computational Linguistics.

Steven Bethard, Guergana Savova, Martha Palmer, and James Pustejovsky. 2017. SemEval-2017 Task 12: Clinical TempEval. In *Proceedings of the 11th International Workshop on Semantic Evaluation*, pages 565–572. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association of Computational Linguistics*, 5:135–146.

Leonardo Campillos, Louise Deléger, Cyril Grouin, et al. 2018. A French Clinical Corpus with Comprehensive Semantic Annotations: Development of the Medical Entity and Relation LIMSI annOtated Text corpus (MERLOT). *Language Resources and Evaluation*, 52(2):571–601.

François Chollet et al. 2015. Keras. `https://github.com/fchollet/keras`.

Ronan Collobert, Jason Weston, Léon Bottou, et al. 2011. Natural Language Processing (Almost) from Scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Thanh Hai Dang, Hoang-Quynh Le, Trang M Nguyen, and Sinh T Vu. 2018. D3NER: biomedical named entity recognition using CRF-biLSTM improved with fine-tuned embeddings of various linguistic information. *Bioinformatics*.

Franck Dernoncourt, Ji Young Lee, Özlem Uzuner, and Peter Szolovits. 2017. De-identification of Patient Notes with Recurrent Neural Networks. *Journal of the American Medical Informatics Association*, 24(3):596–606.

Cyril Grouin and Aurélie Névéol. 2014. De-identification of clinical notes in French: towards a protocol for reference corpus development. *Journal of Biomedical Informatics*, 50:151–161.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Rezarta Islamaj Dogan and Zhiyong Lu. 2012. An improved corpus of disease mentions in PubMed citations. *Proceedings of the 2012 Workshop on Biomedical Natural Language Processing*, pages 91–99.

Meizhi Ju, Makoto Miwa, and Sophia Ananiadou. 2018. A Neural Layered Model for Nested Named Entity Recognition. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1446–1459. Association for Computational Linguistics.

Arzoo Katiyar and Claire Cardie. 2018. Nested Named Entity Recognition Revisited. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 861–871. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *Proceedings of the 3rd International Conference on Learning Representations*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural Architectures for Named Entity Recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270. Association for Computational Linguistics.

David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. 2004. RCV1: A New Benchmark Collection for Text Categorization Research. *Journal of Machine Learning Research*, 5:361–397.

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*, pages 1064–1074. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient Estimation of Word Representations in Vector Space. *CoRR*, abs/1301.3781.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013b. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119.

Adam Paszke, Sam Gross, Soumith Chintala, et al. 2017. Automatic differentiation in PyTorch. In *NIPS-W*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1532–1543. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, et al. 2018. Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2227–2237. Association for Computational Linguistics.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50. European Language Resources Association.

Marek Rei. 2017. Semi-supervised Multitask Learning for Sequence Labeling. In *Proceedings of the 55th Conference of the Association for Computational Linguistics*, pages 2121–2130. Association for Computational Linguistics.

Marek Rei, Gamal Crichton, and Sampo Pyysalo. 2016. Attending to Characters in Neural Sequence Labeling Models. In *Proceedings of the 26th International Conference on Computational Linguistics*, pages 309–318.

Marek Rei and Helen Yannakoudakis. 2016. Compositional Sequence Labeling Models for Error Detection in Learner Writing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1181–1191. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2017. Reporting Score Distributions Makes a Difference: Performance Study of LSTM-networks for Sequence Tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 338–348.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In *Proceedings of the 7th Conference on Natural Language Learning*. Association for Computational Linguistics.

Laurence H. Smith, Thomas C. Rindflesch, and W. John Wilbur. 2004. MedPost: a Part-of-speech Tagger for BioMedical Text. *Bioinformatics*, 20(14):2320–2321.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, et al. 2012. brat: a Web-based Tool for NLP-Assisted Text Annotation. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, volume Demonstration Papers, pages 102–107. Association for Computational Linguistics.

William F. Styler IV, Steven Bethard, Sean Finan, et al. 2014. Temporal Annotation in the Clinical Domain. *Transactions of the Association for Computational Linguistics*, 2:143–154.

Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. 2017. Revisiting Unreasonable Effectiveness of Data in Deep Learning Era. *CoRR*, abs/1707.02968.

Jie Yang and Yue Zhang. 2018. NCRF++: An Open-source Neural Sequence Labeling Toolkit. In *Proceedings of ACL 2018, System Demonstrations*, pages 74–79. Association for Computational Linguistics.