YASET Documentation

Release 0

Julien Tourille, Olivier Ferret, Aurélie Névéol, Xavier Tannier

Jan 23, 2018

User Documentation

CHAPTER 1

Getting Started

This document will show you how to install and upgrade YASET.

1.1 Requirements

- You need a working **Python 3.3+** environment.
- **Optional**: we recommend the use of *Graphics Processing Units* (GPU). YASET implements state-of-the-art neural network models for sequence tagging. Hence, it can leverage GPU computational power to speed up computation. However, this requirement is optional and the tool should be able to learn models in a reasonable time frame using only *Central Processing Units* (CPU).

1.2 Installation

Here are the steps to follow in order to install YASET.

1. **Download** the latest stable version on GitHub or **clone the repository** if you want to use the cutting-edge development version.

```
$ git clone git@github.com:jtourille/yaset.git
```

2. Uncompress the file if necessary and move to the newly created directory. Install YASET by invoking *pip*.

```
$ cd yaset
$ pip install .
```

1.3 GPU Support

YASET install the non-GPU version of TensorFlow by default. If you want to use GPUs with YASET, upgrade the TensorFlow package.

```
$ pip install tensorflow-gpu==1.2.0
```

1.4 Upgrade

If you want to upgrade to a newer version, download the latest release or pull the latest version of the repository and then upgrade the package. If you switched to the GPU version of TensorFlow, the change will be kept. You do not need to repeat TensorFlow upgrade.

```
$ git pull
$ pip install --upgrade .
```

CHAPTER 2

Data

This document presents input file formatting requirements for train and test instances, and word embeddings.

2.1 Train and Test Instances

YASET accepts *CoNLL*-like formatted data:

- one token per line
- sequences separated by blank lines

The first column **must** contain the tokens and the last column **must** contain the labels. You can add as many other columns as you need, they will be ignored by the system. Columns **must** be separated by **tabulations**.

The example below which is extracted from the English part of the CoNLL-2003 Shared Task corpus (Tjong et al., 2003 *[3]*) illustrates this format.

```
...

EU  NNP     I-NP    I-ORG
rejects     VBZ     I-VP    O
German      JJ      I-NP    I-MISC
call        NN      I-NP    O
to  TO      I-VP    O
boycott     VB      I-VP    O
British     JJ      I-NP    I-MISC
lamb        NN      I-NP    O
.   .       O       O

...
```

2.2 Word Embeddings

YASET supports two word embedding formats:

- gensim models (Řehůřek et al., 2010 *[4]*)
- word2vec models (Mikolov et al., 2013 *[1]*)

If you want to use other types of embeddings, you must first convert them to one of these two formats. For instance, if you have computed word embeddings using Glove (Pennington et al., 2014 *[2]*), you can convert the file to word2vec text format by using the script provided within the gensim library.

References

CHAPTER 3

Train a model

This document explains how to train a model with YASET.

3.1 Quick Start

To train a model, make a copy of the configuration file sample and adjust the parameters to you situation.

```
$ cp config.ini config-xp.ini
```

Invoke the YASET command. You can turn on the *verbose mode* with the debug flag (`--debug`).

```
$ yaset [--debug] LEARN --config config-xp.ini
```

3.2 Configuration Parameters

The configuration file is divided into 3 parts:

- **general**: parameters related to the experiment (see below for further explanations)
- **data**: parameters related to training instances and word embedding models
- **training**: parameters related to model training (e.g. learning algorithm, evaluation metrics or mini-batch size)
- **<model parameters>**: depending on your choice regarding the neural network model to use (specified in the *training* section), you can modify the model parameters (e.g. hidden layer sizes or character embedding size)

3.2.1 general section

`batch_mode:` **`bool`** Set this parameter to `true` if you want to perform multiple runs of the same experiment. This allows to check the model robustness to random seed initial value (Reimers et al. (2017) *[3]*).

batch_iter: **int** Specify the number of runs to perform. This will be ignored if the value of the parameter `batch_mode` is `false`

experiment_name: **str** Specify the experiment name. The name will be used for directory and file naming.

3.2.2 data section

train_file_path: **str** Specify the *training instance file path* (absolute or relative). Please refer to the *data formatting section* of the *data document* for further information about file format.

dev_file_use: **bool** Set this parameter to `true` if you want to use a development instance file, `false` otherwise.

dev_file_path: **str** Specify the *development instance file path* (absolute or relative). This parameter will be ignored if the value of the parameter `dev_file_use` is set to `false`. Please refer to the *data formatting section* of the *data document* for further information about file format.

dev_random_ratio: **float** Specify the percentage of training instances that should be kept as development instances (float between 0 and 1, e.g. 0.2). This will be ignored if the value of the parameter `dev_file_use` is `true`.

dev_random_seed_use: **bool** Set this parameter to `true` if you want to use a random seed for train/dev split. This will be ignored if the value of the parameter `dev_file_use` is `true`.

dev_random_seed_value: **int** Specify the random seed value (integer). This will be ignored if the value of the parameter `dev_file_use` is `true` or if the value of the parameter `dev_random_seed_use` is `false`

preproc_lower_input: **bool** Set this parameter to `true` if you want YASET to lowercase tokens before token-vector looking-up, `false` otherwise. This is useful if you have pre-trained word embeddings using a lowercased corpus.

preproc_replace_digits: **bool** Set this parameter to `true` if you want YASET to replace digits by the digit 0 before token-vector looking-up, `false` otherwise (e.g. "4,5mg" will be changed to "0,0mg").

embedding_model_type: **str** Specify the format of the pre-trained word embeddings that you want to use to train the system. Two formats are supported:

- `gensim`: models pre-trained with the Python library gensim
- `word2vec`: models pre-trained with the tool word2vec

embedding_model_path: **str** Specify the path of the pre-trained word embedding file (absolute or relative).

embedding_oov_strategy: **str** Specify the strategy for Out-Of-Vocabulary (OOV) tokens. Two strategies are available:

- `map`: a vector for OOV tokens is provided in the embedding file. Set *embedding_oov_strategy* to `map` and specify the OOV vector ID (`embedding_oov_map_token_id` parameter)
- `replace`: following Lample et al. (2016) *[2]*, an OOV vector will be randomly initialized and trained by randomly replacing singletons in the training instances by this vector. You can adjust the replacement rate by changing the value of the parameter `embedding_oov_replace_rate`.

embedding_oov_map_token_id: **str** Specify the OOV token ID if you use the strategy `map`. This will be ignored if the value of the parameter `embedding_oov_strategy` is not `map`.

embedding_oov_replace_rate: **float** Specify the replacement rate if you want to use the strategy `replace` (float between 0 and 1, e.g. 0.2). This will be ignored if the value of the parameter `embedding_oov_strategy` is not `replace`.

working_dir: **str** Specify the working directory path where a timestamped working directory will be created for the current run. For instance, if you specify `$USER/temp`, the directory `$USER/temp/yaset-learn-YYYYMMDD` will be created.

3.2.3 training

model_type: **str** Specify the neural network model that you want to use. There is only one choice at this time. Other models will be implemented in the next releases.

- `bilstm-char-crf`: implementation of the model presented in Lample et al. (2016) *[2]*. More information can be found in the original paper. Model parameters can be set in the *bilstm-char-crf section* of the configuration file.

max_iterations: **int** Specify the maximum number of training iterations. Training will be stopped if early stopping criterion is not reached before this iteration number (see `patience` parameter).

patience: **int** Specify the number of iterations to wait before early stop if there is no performance improvement on the validation instances.

dev_metric: **str** Specify the metric used for performance computation on the validation instances.

- `accuracy`: standard token accuracy.
- `conll`: metric which operates at the entity level. This should be used with a IOB(ES) markup on Named Entity Recognition related tasks. The implementation is taken for most parts from the Python adaptation by Sampo Pyysalo of the original script developed for the CoNLL-2003 Shared Task (Tjong et al., 2003 *[3]*).

trainable_word_embeddings: **bool** Set this parameter to `true` if you want YASET to fine-tune word embeddings during network training, `false` otherwise.

cpu_cores: **int** Specify the number of CPU cores (upper-bound) that should be used during network training.

batch_size: **int** Specify the mini-batch size used during training.

store_matrices_on_gpu: **bool** Set this parameter to `true` if you want to keep the word embedding matrix on GPU memory, `false` otherwise.

bucket_use: **bool** Set this parameter to `true` if you want to bucketize training instances during network training. Bucket boundaries will be automatically computed.

opt_algo: **str** Specify the optimization algorithm used during network training. You can choose between between `adam` (Kingma et al.,2014 *[1]*) or `sgd`.

opt_lr: **float** Specify the initial learning rate applied during network training.

opt_gc_use: **bool** Set this parameter to `true` if you want to use gradient clipping during network training, `false` otherwise.

opt_gc_type: **str** Specify the gradient clipping type (`clip_by_norm` or `clip_by_value`) This will be ignored if the value of the parameter `opt_gc_use` is `false`.

opt_gs_val: **float** Specify the gradient clipping value. This parameter will be ignored if the value for the parameter `opt_gc_use` is `false`.

opt_decay_use: **bool** Set this parameter to `true` if you want to use learning rate decay during network training, `false` otherwise.

opt_decay_rate: **float** Specify the decay rate (float between 0 and 1, e.g. 0.2). This parameter will be ignored if the value for the parameter `opt_decay_use` is `false`.

opt_decay_iteration: **int** Specify the learning rate decay frequency. If you set the frequency to n, the learning rate lr will be decayed by the rate specified in the parameter `opt_decay_iteration` every n iterations.

3.2.4 bilstm-char-crf

These parameters are related to the neural network model presented in Lample et al. (2016) *[2]*.

hidden_layer_size: **int** Specify the main LSTM hidden layer size.

dropout_rate: **float** Specify the dropout rate to apply on input embeddings before feeding them to the main LSTM.

use_char_embeddings: **bool** Set this parameter to `true` if you want to use character embeddings in the model, `false` otherwise.

char_hidden_layer_size: **int** Specify the character LSTM hidden layer size. This parameter will be ignored if the value for the parameter `use_char_embeddings` is `false`.

char_embedding_size: **int** Specify the character embedding size. This parameter will be ignored if the value for the parameter `use_char_embeddings` is `false`.

CHAPTER 4

Apply a model

This document explains how to apply a YASET model on test data.

4.1 Data Format

YASET accepts *CoNLL*-like formatted data:

- one token per line
- sequences separated by blank lines

The difference with train and validation data is that test data do not need to have a label column. Hence the minimum number of column is one (i.e. the token column). You can add as many other columns you need. They will be ignored by the system.

```
...

EU  NNP     I-NP
rejects     VBZ     I-VP
German      JJ      I-NP
call        NN      I-NP
to  TO      I-VP
boycott     VB      I-VP
British     JJ      I-NP
lamb        NN      I-NP
.   .       O

...
```

During the APPLY phase, YASET will add one column to the document with the predicted labels.

4.2 Apply a model

To apply a model, run the following command

```
$ yaset [--debug] APPLY --working-dir /path/to/working_dir \
   --input-file /path/to/file.tab \
   --model-path /path/to/pre-trained-model
```

Argument description:

--working-dir Specify the working directory path where a timestamped working directory will be created for the current run. For instance, if you specify `$USER/temp`, the directory `$USER/temp/yaset-apply-YYYYMMDD` will be created.

--input-file Specify the input test file which contains the test instances.

--model-path Specify the path of the YASET model

Bibliography

[1] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. *Computing Research Repository*, 2013.

[2] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, 1532–1543. Copenhagen, Denmark, 2014. Association for Computational Linguistics.

[3] Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In *Proceedings of CoNLL-2003*, 142–147. Edmonton, Canada, 2003. Association for Computational Linguistics.

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

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

[2] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 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*, 260–270. San Diego, California, 2016. Association for Computational Linguistics.

[3] Nils Reimers and Iryna Gurevych. 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*. Association for Computational Linguistics, 2017.

[3] Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In *Proceedings of CoNLL-2003*, 142–147. Edmonton, Canada, 2003. Association for Computational Linguistics.