AutoNER Documentation

Jingbo Shang, Liyuan Liu

Oct 08, 2018

Sequence Labeling

Check Our New NER Toolkit

- **Inference**:
  - LightNER: inference w. models pre-trained / trained w. *any* following tools, *efficiently*.
- **Training**:
  - LD-Net: train NER models w. efficient contextualized representations.
  - VanillaNER: train vanilla NER models w. pre-trained embedding.
- **Distant Training**:
  - AutoNER: train NER models w.o. line-by-line annotations and get competitive performance.

No line-by-line annotations, AutoNER trains named entity taggers with distant supervision.

Details about AutoNER can be accessed at: https://arxiv.org/abs/1804.07827.

CHAPTER 1

Sequence Labeling

1.1 model_partial_ner.basic module

class model_partial_ner.basic.**BasicRNN**(*layer_num*, *unit*, *emb_dim*, *hid_dim*, *droprate*, *batch_norm*)

The multi-layer recurrent networks for the vanilla stacked RNNs.

Parameters

- **layer_num** (int, required.) – The number of layers.
- **unit** (torch.nn.Module, required.) – The type of rnn unit.
- **input_dim** (int, required.) – The input dimension fo the unit.
- **hid_dim** (int, required.) – The hidden dimension fo the unit.
- **droprate** (float, required.) – The dropout ratrio.
- **batch_norm** (bool, required.) – Incorporate batch norm or not.

forward(*x*)

Calculate the output.

Parameters **x** (torch.LongTensor, required.) – the input tensor, of shape (seq_len, batch_size, input_dim).

Returns **output** – The output of RNNs.

Return type torch.FloatTensor.

init_hidden()

Initialize hidden states.

rand_ini()

Random Initialization.

to_params()

To parameters.

class model_partial_ner.basic.**BasicUnit**(*unit*, *input_dim*, *hid_dim*, *droprate*, *batch_norm*)
The basic recurrent unit for the vanilla stacked RNNs.

Parameters

- **unit** (torch.nn.Module, required.) – The type of rnn unit.
- **input_dim** (int, required.) – The input dimension fo the unit.
- **hid_dim** (int, required.) – The hidden dimension fo the unit.
- **droprate** (float, required.) – The dropout ratrio.
- **batch_norm** (bool, required.) – Incorporate batch norm or not.

forward(*x*)
Calculate the output.

Parameters **x** (torch.LongTensor, required.) – the input tensor, of shape (seq_len, batch_size, input_dim).

Returns **output** – The output of RNNs.

Return type torch.FloatTensor.

init_hidden()
Initialize hidden states.

rand_ini()
Random Initialization.

1.2 model_partial_ner.dataset module

1.3 model_partial_ner.highway module

class model_partial_ner.highway.**highway**(*size*, *num_layers=1*, *droprate=0.5*)
Highway layers

Parameters

- **size** (int, required.) – Input and output dimension.
- **num_layers** (int, required.) – Number of layers.
- **droprate** (float, required.) – Dropout ratio

forward(*x*)
update statics for f1 score

Parameters **(ins_num, hidden_dim)** (*x*) –

Returns **output** – output tensor (ins_num, hidden_dim)

Return type torch.FloatTensor.

rand_ini()
random initialization

1.4 model_partial_ner.ner module

1.5 model_partial_ner.object module

model_partial_ner.object.**hinge_loss**(*score*, *label*)
Hinge loss for distant supervision.

class model_partial_ner.object.**softCE**(*if_average=True*)
The objective function for the distant supervised typing.

Parameters **if_average** (bool, optional, (default = True).) – Whether to average over batches or not.

forward(*scores*, *target*)
Calculate the cross entropy loss for distant supervision.

Parameters

- **scores** (torch.FloatTensor, required.) – The input of the softmax.
- **target** (torch.ByteTensor , required.) – The target as the mask for the softmax input.

static soft_max(*vec*, *mask*)
Calculate the softmax for the input with regard to a mask.

Parameters

- **vec** (torch.FloatTensor, required.) – The input of the softmax.
- **mask** (torch.ByteTensor , required.) – The mask for the softmax input.

1.6 model_partial_ner.utils module

model_partial_ner.utils.**adjust_learning_rate**(*optimizer*, *lr*)
Shrink learning rate for pytorch

model_partial_ner.utils.**evaluate_chunking**(*iterator*, *ner_model*, *none_idx*)
Evaluate the chunking performance.

Parameters

- **iterator** (iterator, required.) – Dataset loader.
- **ner_model** (torch.nn.Module , required.) – Sequence labeling model for evaluation.
- **none_idx** (int, required.) – The index for the not-target-type entities.

model_partial_ner.utils.**evaluate_ner**(*iterator*, *ner_model*, *none_idx*, *id2label*)
Evaluate the NER performance.

Parameters

- **iterator** (iterator, required.) – Dataset loader.
- **ner_model** (torch.nn.Module , required.) – Sequence labeling model for evaluation.
- **none_idx** (int, required.) – The index for the not-target-type entities.

model_partial_ner.utils.**evaluate_typing**(*iterator*, *ner_model*, *none_idx*)
Evaluate the typing performance.

Parameters

- **iterator** (iterator, required.) – Dataset loader.
- **ner_model** (torch.nn.Module , required.) – Sequence labeling model for evaluation.
- **none_idx** (int, required.) – The index for the not-target-type entities.

model_partial_ner.utils.**init_embedding**(*input_embedding*)
Initialize embedding

model_partial_ner.utils.**init_linear**(*input_linear*)
Initialize linear transformation

model_partial_ner.utils.**init_lstm**(*input_lstm*)
Initialize lstm

model_partial_ner.utils.**to_scalar**(*var*)
Turn the first element of a tensor to scalar

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

b

h

m

u

Index