LD-Net Documentation

Liyuan Liu

Dec 25, 2018

Language Modeling

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.

This project provides high-performance word-level language model, and sequence labeling with contextualized representation. The key feature of this project is the support of langugage model pruning without retraining.

Details about LD-Net can be accessed at: https://arxiv.org/abs/1804.07827.

CHAPTER 1

Language Modeling

1.1 model_word_ada.adaptive module

class model_word_ada.adaptive.**AdaptiveSoftmax**(*input_size*, *cutoff*)

The adaptive softmax layer. Modified from: https://github.com/rosinality/adaptive-softmax-pytorch/blob/master/adasoft.py

Parameters

- **input_size** (int, required.) – The input dimension.
- **cutoff** (list, required.) – The list of cutoff values.

forward(*w_in*, *target*)

Calculate the log-likihood w.o. calculate the full distribution.

Parameters

- **w_in** (torch.FloatTensor, required.) – the input tensor, of shape (word_num, input_dim).
- **target** (torch.FloatTensor, required.) – the target of the language model, of shape (word_num).

Returns **loss** – The NLL loss.

Return type torch.FloatTensor.

log_prob(*w_in*, *device*)

Calculate log-probability for the whole dictionary.

Parameters

- **w_in** (torch.FloatTensor, required.) – the input tensor, of shape (word_num, input_dim).
- **device** (torch.device, required.) – the target device for calculation.

Returns **prob** – The full log-probability.

Return type `torch.FloatTensor`.

rand_ini()
Random Initialization.

1.2 model_word_ada.basic module

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

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

Parameters

- **unit** (`str`, 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.

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.

1.3 model_word_ada.dataset module

class model_word_ada.dataset.**EvalDataset**(*dataset*, *sequence_length*)
Dataset for Language Modeling

Parameters

- **dataset** (list, required.) – The encoded dataset (outputs of preprocess scripts).
- **sequence_length** (int, required.) – Sequence Length.

construct_index()
construct index for the dataset.

get_tqdm(*device*)
construct dataset reader and the corresponding tqdm.

Parameters **device** (torch.device, required.) – the target device for the dataset loader.

reader(*device*)
construct dataset reader.

Parameters **device** (torch.device, required.) – the target device for the dataset loader.

Returns **reader** – A lazy iterable object

Return type iterator.

class model_word_ada.dataset.**LargeDataset**(*root*, *range_idx*, *batch_size*, *sequence_length*)
Lazy Dataset for Language Modeling

Parameters

- **root** (str, required.) – The root folder for dataset files.
- **range_idx** (int, required.) – The maximum file index for the input files (train_*.pk).
- **batch_size** (int, required.) – Batch size.
- **sequence_length** (int, required.) – Sequence Length.

get_tqdm(*device*)
construct dataset reader and the corresponding tqdm.

Parameters **device** (torch.device, required.) – the target device for the dataset loader.

open_next()
Open the next file.

reader(*device*)
construct dataset reader.

Parameters **device** (torch.device, required.) – the target device for the dataset loader.

Returns **reader** – A lazy iterable object

Return type iterator.

shuffle()
shuffle dataset

1.4 model_word_ada.densenet module

class model_word_ada.densenet.**BasicUnit**(*unit*, *input_dim*, *increase_rate*, *droprate*)
The basic recurrent unit for the densely connected RNNs.

Parameters

- **unit** (torch.nn.Module, required.) – The type of rnn unit.
- **input_dim** (float, required.) – The input dimension fo the unit.
- **increase_rate** (float, required.) – The hidden dimension fo the unit.
- **droprate** (float, required.) – The dropout ratrio.

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.

class model_word_ada.densenet.**DenseRNN**(*layer_num*, *unit*, *emb_dim*, *hid_dim*, *droprate*)
The multi-layer recurrent networks for the densely connected RNNs.

Parameters

- **layer_num** (float, required.) – The number of layers.
- **unit** (torch.nn.Module, required.) – The type of rnn unit.
- **input_dim** (float, required.) – The input dimension fo the unit.
- **hid_dim** (float, required.) – The hidden dimension fo the unit.
- **droprate** (float, required.) – The dropout ratrio.

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.

1.5 model_word_ada.ldnet module

class model_word_ada.ldnet.**BasicUnit**(*unit*, *input_dim*, *increase_rate*, *droprate*, *layer_drop=0*)
The basic recurrent unit for the densely connected RNNs with layer-wise dropout.

Parameters

- **unit** (torch.nn.Module, required.) – The type of rnn unit.
- **input_dim** (float, required.) – The input dimension fo the unit.
- **increase_rate** (float, required.) – The hidden dimension fo the unit.
- **droprate** (float, required.) – The dropout ratrio.
- **layer_dropout** (float, required.) – The layer-wise dropout ratrio.

forward(*x*, *p_out*)
Calculate the output.

Parameters

- **x** (torch.LongTensor, required.) – the input tensor, of shape (seq_len, batch_size, input_dim).
- **p_out** (torch.LongTensor, required.) – the final output tensor for the softmax, of shape (seq_len, batch_size, input_dim).

Returns

- **out** (torch.FloatTensor.) – The undropped outputs of RNNs to the softmax.
- **p_out** (torch.FloatTensor.) – The dropped outputs of RNNs to the next_layer.

init_hidden()
Initialize hidden states.

rand_ini()
Random Initialization.

class model_word_ada.ldnet.**LDRNN**(*layer_num*, *unit*, *emb_dim*, *hid_dim*, *droprate*, *layer_drop*)
The multi-layer recurrent networks for the densely connected RNNs with layer-wise dropout.

Parameters

- **layer_num** (float, required.) – The number of layers.
- **unit** (torch.nn.Module, required.) – The type of rnn unit.
- **input_dim** (float, required.) – The input dimension fo the unit.
- **hid_dim** (float, required.) – The hidden dimension fo the unit.
- **droprate** (float, required.) – The dropout ratrio.
- **layer_dropout** (float, required.) – The layer-wise dropout ratrio.

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 to the Softmax.

Return type torch.FloatTensor.

init_hidden()
Initialize hidden states.

rand_ini()
Random Initialization.

to_params()
To parameters.

1.6 model_word_ada.LM module

class model_word_ada.LM.**LM**(*rnn*, *soft_max*, *w_num*, *w_dim*, *droprate*, *label_dim=-1*, *add_relu=False*)
The language model model.

Parameters

- **rnn** (torch.nn.Module, required.) – The RNNs network.
- **soft_max** (torch.nn.Module, required.) – The softmax layer.
- **w_num** (int , required.) – The number of words.
- **w_dim** (int , required.) – The dimension of word embedding.
- **droprate** (float , required) – The dropout ratio.
- **label_dim** (int , required.) – The input dimension of softmax.

forward(*w_in*, *target*)
Calculate the loss.

Parameters

- **w_in** (torch.FloatTensor, required.) – the input tensor, of shape (word_num, input_dim).
- **target** (torch.FloatTensor, required.) – the target of the language model, of shape (word_num).

Returns **loss** – The NLL loss.

Return type torch.FloatTensor.

init_hidden()
Initialize hidden states.

load_embed(*origin_lm*)
Load embedding from another language model.

log_prob(*w_in*)
Calculate log-probability for the whole dictionary.

Parameters **w_in** (torch.FloatTensor, required.) – the input tensor, of shape (word_num, input_dim).

Returns **prob** – The full log-probability.

Return type `torch.FloatTensor`.

rand_ini()
Random initialization.

1.7 model_word_ada.utils module

model_word_ada.utils.**adjust_learning_rate**(*optimizer*, *lr*)
adjust learning to the the new value.

Parameters

- **optimizer** (*required.*) – pytorch optimizer.
- **float** (float, required.) – the target learning rate.

model_word_ada.utils.**init_embedding**(*input_embedding*)
random initialize embedding

model_word_ada.utils.**init_linear**(*input_linear*)
random initialize linear projection.

model_word_ada.utils.**init_lstm**(*input_lstm*)
random initialize lstms

model_word_ada.utils.**repackage_hidden**(*h*)
Wraps hidden states in new Variables, to detach them from their history

Parameters **h** (Tuple or Tensors, required.) – Tuple or Tensors, hidden states.

Returns **hidden** – detached hidden states

Return type Tuple or Tensors.

model_word_ada.utils.**to_scalar**(*var*)
convert a tensor to a scalar number

CHAPTER 2

Sequence Labeling

2.1 model_seq.crf module

class model_seq.crf.**CRF**(*hidden_dim: int*, *tagset_size: int*, *if_bias: bool = True*)
Conditional Random Field Module

Parameters

- **hidden_dim** (int, required.) – the dimension of the input features.
- **tagset_size** (int, required.) – the size of the target labels.
- **if_bias** (bool, optional, (default=True).) – whether the linear transformation has the bias term.

forward(*feats*)
calculate the potential score for the conditional random field.

Parameters **feats** (torch.FloatTensor, required.) – the input features for the conditional random field, of shape (*, hidden_dim).

Returns **output** – A float tensor of shape (ins_num, from_tag_size, to_tag_size)

Return type torch.FloatTensor.

rand_init()
random initialization

class model_seq.crf.**CRFDecode**(*y_map: dict*)
The negative loss for the Conditional Random Field Module

Parameters **y_map** (dict, required.) – a dict maps from tag string to tag index.

decode(*scores*, *mask*)
find the best path from the potential scores by the viterbi decoding algorithm.

Parameters

- **scores** (torch.FloatTensor, required.) – the potential score for the conditional random field, of shape (seq_len, batch_size, from_tag_size, to_tag_size).
- **mask** (torch.ByteTensor, required.) – the mask for the unpadded sentence parts, of shape (seq_len, batch_size).

Returns **output** – A LongTensor of shape (seq_len - 1, batch_size)

Return type torch.LongTensor.

to_spans(*sequence*)
decode the best path to spans.

Parameters **sequence** (*list, required.*) – the list of best label indexes paths .

Returns **output** – A set of chunks contains the position and type of the entities.

Return type set.

class model_seq.crf.**CRFLoss**(*y_map: dict*, *average_batch: bool = True*)
The negative loss for the Conditional Random Field Module

Parameters

- **y_map** (dict, required.) – a dict maps from tag string to tag index.
- **average_batch** (bool, optional, (default=True).) – whether the return score would be averaged per batch.

forward(*scores*, *target*, *mask*)
calculate the negative log likehood for the conditional random field.

Parameters

- **scores** (torch.FloatTensor, required.) – the potential score for the conditional random field, of shape (seq_len, batch_size, from_tag_size, to_tag_size).
- **target** (torch.LongTensor, required.) – the positive path for the conditional random field, of shape (seq_len, batch_size).
- **mask** (torch.ByteTensor, required.) – the mask for the unpadded sentence parts, of shape (seq_len, batch_size).

Returns **loss** – The NLL loss.

Return type torch.FloatTensor.

2.2 model_seq.dataset module

class model_seq.dataset.**SeqDataset**(*dataset: list*, *flm_pad: int*, *blm_pad: int*, *w_pad: int*, *c_con: int*, *c_pad: int*, *y_start: int*, *y_pad: int*, *y_size: int*, *batch_size: int*)
Dataset for Sequence Labeling

Parameters

- **dataset** (list, required.) – The encoded dataset (outputs of preprocess scripts).
- **flm_pad** (int, required.) – The pad index for the forward language model.
- **blm_pad** (int, required.) – The pad index for the backward language model.
- **w_pad** (int, required.) – The pad index for the word-level inputs.

- **c_con** (int, required.) – The index of connect character token for character-level inputs.
- **c_pad** (int, required.) – The pad index for the character-level inputs.
- **y_start** (int, required.) – The index of the start label token.
- **y_pad** (int, required.) – The index of the pad label token.
- **y_size** (int, required.) – The size of the tag set.
- **batch_size** (int, required.) – Batch size.

batchify(*batch*, *device*)
batchify a batch of data and move to a device.

Parameters

- **batch** (list, required.) – a sample from the encoded dataset (outputs of preprocess scripts).
- **device** (torch.device, required.) – the target device for the dataset loader.

construct_index(*dataset*)
construct index for the dataset.

Parameters **dataset** (list, required.) – the encoded dataset (outputs of preprocess scripts).

get_tqdm(*device*)
construct dataset reader and the corresponding tqdm.

Parameters **device** (torch.device, required.) – the target device for the dataset loader.

reader(*device*)
construct dataset reader.

Parameters **device** (torch.device, required.) – the target device for the dataset loader.

Returns **reader** – A lazy iterable object

Return type iterator.

shuffle()
shuffle dataset

2.3 model_seq.elmo module

class model_seq.elmo.**EBUnit**(*ori_unit*, *droprate*, *fix_rate*)
The basic recurrent unit for the ELMo RNNs wrapper.

Parameters

- **ori_unit** (torch.nn.Module, required.) – The original module of rnn unit.
- **droprate** (float, required.) – The dropout ratrio.
- **fix_rate** (bool, required.) – Whether to fix the rqtio.

forward(*x*)
Calculate the output.

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

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

Return type torch.FloatTensor.

class model_seq.elmo.**ERNN**(*ori_drnn*, *droprate*, *fix_rate*)
The multi-layer recurrent networks for the ELMo RNNs wrapper.

Parameters

- **ori_drnn** (torch.nn.Module, required.) – The original module of rnn networks.
- **droprate** (float, required.) – The dropout ratrio.
- **fix_rate** (bool, required.) – Whether to fix the rqtio.

forward(*x*)
Calculate the output.

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

Returns **output** – The ELMo outputs.

Return type torch.FloatTensor.

regularizer()
Calculate the regularization term.

Returns

Return type The regularization term.

class model_seq.elmo.**ElmoLM**(*ori_lm*, *backward*, *droprate*, *fix_rate*)
The language model for the ELMo RNNs wrapper.

Parameters

- **ori_lm** (torch.nn.Module, required.) – the original module of language model.
- **backward** (bool, required.) – whether the language model is backward.
- **droprate** (float, required.) – the dropout ratrio.
- **fix_rate** (bool, required.) – whether to fix the rqtio.

forward(*w_in*, *ind=None*)
Calculate the output.

Parameters

- **w_in** (torch.LongTensor, required.) – the input tensor, of shape (seq_len, batch_size).
- **ind** (torch.LongTensor, optional, (default=None).) – the index tensor for the backward language model, of shape (seq_len, batch_size).

Returns **output** – The ELMo outputs.

Return type torch.FloatTensor.

init_hidden()
initialize hidden states.

prox(*lambda0*)
the proximal calculator.

regularizer()
Calculate the regularization term.

Returns **reg** – The list of regularization terms.

Return type `list`.

2.4 model_seq.evaluator module

class `model_seq.evaluator.`**eval_batch**(*decoder*)
Base class for evaluation, provide method to calculate f1 score and accuracy.

Parameters **decoder** (`torch.nn.Module`, required.) – the decoder module, which needs to contain the `to_span()` method.

acc_score()
calculate the accuracy score based on the inner counter.

calc_acc_batch(*decoded_data*, *target_data*)
update statics for accuracy score.

Parameters

- **decoded_data** (`torch.LongTensor`, required.) – the decoded best label index pathes.
- **target_data** (`torch.LongTensor`, required.) – the golden label index pathes.

calc_f1_batch(*decoded_data*, *target_data*)
update statics for f1 score.

Parameters

- **decoded_data** (`torch.LongTensor`, required.) – the decoded best label index pathes.
- **target_data** (`torch.LongTensor`, required.) – the golden label index pathes.

eval_instance(*best_path*, *gold*)
Calculate statics to update inner counters for one instance.

Parameters

- **best_path** (*required.*) – the decoded best label index pathe.
- **gold** (*required.*) – the golden label index pathes.

f1_score()
calculate the f1 score based on the inner counter.

reset()
reset counters.

class `model_seq.evaluator.`**eval_wc**(*decoder*, *score_type*)
evaluation class for LD-Net

Parameters

- **decoder** (`torch.nn.Module`, required.) – the decoder module, which needs to contain the `to_span()` and `decode()` method.
- **score_type** (`str`, required.) – whether the f1 score or the accuracy is needed.

calc_score(*seq_model*, *dataset_loader*)
calculate scores

Parameters

- **seq_model** (*required.*) – sequence labeling model.

- **dataset_loader** (*required.*) – the dataset loader.

Returns **score** – calculated score.

Return type float.

2.5 model_seq.seqlabel module

class model_seq.seqlabel.**SeqLabel**(*f_lm*, *b_lm*, *c_num: int*, *c_dim: int*, *c_hidden: int*, *c_layer: int*, *w_num: int*, *w_dim: int*, *w_hidden: int*, *w_layer: int*, *y_num: int*, *droprate: float*, *unit: str = 'lstm'*)

Sequence Labeling model augumented with language model.

Parameters

- **f_lm** (torch.nn.Module, required.) – The forward language modle for contextualized representations.
- **b_lm** (torch.nn.Module, required.) – The backward language modle for contextualized representations.
- **c_num** (int , required.) – The number of characters.
- **c_dim** (int , required.) – The dimension of character embedding.
- **c_hidden** (int , required.) – The dimension of character hidden states.
- **c_layer** (int , required.) – The number of character lstms.
- **w_num** (int , required.) – The number of words.
- **w_dim** (int , required.) – The dimension of word embedding.
- **w_hidden** (int , required.) – The dimension of word hidden states.
- **w_layer** (int , required.) – The number of word lstms.
- **y_num** (int , required.) – The number of tags types.
- **droprate** (float , required) – The dropout ratio.
- **unit** (*"str", optional, (default = 'lstm')*) – The type of the recurrent unit.

forward(*f_c*, *f_p*, *b_c*, *b_p*, *flm_w*, *blm_w*, *blm_ind*, *f_w*)

Calculate the output (crf potentials).

Parameters

- **f_c** (torch.LongTensor, required.) – Character-level inputs in the forward direction.
- **f_p** (torch.LongTensor, required.) – Ouput position of character-level inputs in the forward direction.
- **b_c** (torch.LongTensor, required.) – Character-level inputs in the backward direction.
- **b_p** (torch.LongTensor, required.) – Ouput position of character-level inputs in the backward direction.
- **flm_w** (torch.LongTensor, required.) – Word-level inputs for the forward language model.

- **blm_w** (torch.LongTensor, required.) – Word-level inputs for the backward language model.
- **blm_ind** (torch.LongTensor, required.) – Ouput position of word-level inputs for the backward language model.
- **f_w** (torch.LongTensor, required.) – Word-level inputs for the sequence labeling model.

Returns output – A float tensor of shape (sequence_len, batch_size, from_tag_size, to_tag_size)

Return type torch.FloatTensor.

load_pretrained_word_embedding(*pre_word_embeddings*)
Load pre-trained word embedding.

prune_dense_rnn()
Prune dense rnn to be smaller by delecting layers.

rand_init()
Random initialization.

set_batch_seq_size(*sentence*)
Set the batch size and sequence length.

to_params()
To parameters.

class model_seq.seqlabel.**Vanilla_SeqLabel**(*f_lm*, *b_lm*, *c_num*, *c_dim*, *c_hidden*, *c_layer*, *w_num*, *w_dim*, *w_hidden*, *w_layer*, *y_num*, *droprate*, *unit='lstm'*)
Sequence Labeling model augumented without language model.

Parameters

- **f_lm** (torch.nn.Module, required.) – forward language modle for contextualized representations.
- **b_lm** (torch.nn.Module, required.) – backward language modle for contextualized representations.
- **c_num** (int , required.) – number of characters.
- **c_dim** (int , required.) – dimension of character embedding.
- **c_hidden** (int , required.) – dimension of character hidden states.
- **c_layer** (int , required.) – number of character lstms.
- **w_num** (int , required.) – number of words.
- **w_dim** (int , required.) – dimension of word embedding.
- **w_hidden** (int , required.) – dimension of word hidden states.
- **w_layer** (int , required.) – number of word lstms.
- **y_num** (int , required.) – number of tags types.
- **droprate** (float , required) – dropout ratio.
- **unit** (*"str", optional, (default = 'lstm')*) – type of the recurrent unit.

forward(*f_c*, *f_p*, *b_c*, *b_p*, *flm_w*, *blm_w*, *blm_ind*, *f_w*)
Calculate the output (crf potentials).

Parameters

- **f_c** (torch.LongTensor, required.) – Character-level inputs in the forward direction.
- **f_p** (torch.LongTensor, required.) – Ouput position of character-level inputs in the forward direction.
- **b_c** (torch.LongTensor, required.) – Character-level inputs in the backward direction.
- **b_p** (torch.LongTensor, required.) – Ouput position of character-level inputs in the backward direction.
- **flm_w** (torch.LongTensor, required.) – Word-level inputs for the forward language model.
- **blm_w** (torch.LongTensor, required.) – Word-level inputs for the backward language model.
- **blm_ind** (torch.LongTensor, required.) – Ouput position of word-level inputs for the backward language model.
- **f_w** (torch.LongTensor, required.) – Word-level inputs for the sequence labeling model.

Returns output – A float tensor of shape (sequence_len, batch_size, from_tag_size, to_tag_size)

Return type torch.FloatTensor.

load_pretrained_word_embedding(*pre_word_embeddings*)
Load pre-trained word embedding.

rand_init()
Random initialization.

set_batch_seq_size(*sentence*)
set batch size and sequence length

2.6 model_seq.seqlm module

class model_seq.seqlm.**BasicSeqLM**(*ori_lm*, *backward*, *droprate*, *fix_rate*)
The language model for the dense rnns.

Parameters

- **ori_lm** (torch.nn.Module, required.) – the original module of language model.
- **backward** (bool, required.) – whether the language model is backward.
- **droprate** (float, required.) – the dropout ratrio.
- **fix_rate** (bool, required.) – whether to fix the rqtio.

forward(*w_in*, *ind=None*)
Calculate the output.

Parameters

- **w_in** (torch.LongTensor, required.) – the input tensor, of shape (seq_len, batch_size).

- **ind** (torch.LongTensor, optional, (default=None).) – the index tensor for the backward language model, of shape (seq_len, batch_size).

Returns **output** – The ELMo outputs.

Return type torch.FloatTensor.

init_hidden()
initialize hidden states.

regularizer()
Calculate the regularization term.

Returns **reg** – The list of regularization terms.

Return type list.

to_params()
To parameters.

2.7 model_seq.sparse_lm module

class model_seq.sparse_lm.**SBUnit**(*ori_unit*, *droprate*, *fix_rate*)
The basic recurrent unit for the dense-RNNs wrapper.

Parameters

- **ori_unit** (torch.nn.Module, required.) – the original module of rnn unit.
- **droprate** (float, required.) – the dropout ratrio.
- **fix_rate** (bool, required.) – whether to fix the rqtio.

forward(*x*, *weight=1*)
Calculate the output.

Parameters

- **x** (torch.FloatTensor, required.) – The input tensor, of shape (seq_len, batch_size, input_dim).
- **weight** (torch.FloatTensor, required.) – The selection variable.

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

Return type torch.FloatTensor.

prune_rnn(*mask*)
Prune dense rnn to be smaller by delecting layers.

Parameters **mask** (torch.ByteTensor, required.) – The selection tensor for the input matrix.

class model_seq.sparse_lm.**SDRNN**(*ori_drnn*, *droprate*, *fix_rate*)
The multi-layer recurrent networks for the dense-RNNs wrapper.

Parameters

- **ori_unit** (torch.nn.Module, required.) – the original module of rnn unit.
- **droprate** (float, required.) – the dropout ratrio.
- **fix_rate** (bool, required.) – whether to fix the rqtio.

forward(*x*)
Calculate the output.

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

Returns **output** – The ELMo outputs.

Return type torch.FloatTensor.

prox()
the proximal calculator.

prune_dense_rnn()
Prune dense rnn to be smaller by delecting layers.

regularizer()
Calculate the regularization term.

Returns

- **reg0** (torch.FloatTensor.) – The value of reg0.
- **reg1** (torch.FloatTensor.) – The value of reg1.
- **reg2** (torch.FloatTensor.) – The value of reg2.

to_params()
To parameters.

class model_seq.sparse_lm.**SparseSeqLM**(*ori_lm*, *backward*, *droprate*, *fix_rate*)
The language model for the dense rnns with layer-wise selection.

Parameters

- **ori_lm** (torch.nn.Module, required.) – the original module of language model.
- **backward** (bool, required.) – whether the language model is backward.
- **droprate** (float, required.) – the dropout ratrio.
- **fix_rate** (bool, required.) – whether to fix the rqtio.

forward(*w_in*, *ind=None*)
Calculate the output.

Parameters

- **w_in** (torch.LongTensor, required.) – the input tensor, of shape (seq_len, batch_size).
- **ind** (torch.LongTensor, optional, (default=None).) – the index tensor for the backward language model, of shape (seq_len, batch_size).

Returns **output** – The ELMo outputs.

Return type torch.FloatTensor.

init_hidden()
initialize hidden states.

prox()
the proximal calculator.

prune_dense_rnn()
Prune dense rnn to be smaller by delecting layers.

regularizer()
Calculate the regularization term.

Returns **reg** – The list of regularization terms.

Return type list.

to_params()
To parameters.

2.8 model_seq.utils module

model_seq.utils.**adjust_learning_rate**(*optimizer*, *lr*)
adjust learning to the the new value.

Parameters

- **optimizer** (*required.*) – pytorch optimizer.
- **float** (float, required.) – the target learning rate.

model_seq.utils.**init_embedding**(*input_embedding*)
random initialize embedding

model_seq.utils.**init_linear**(*input_linear*)
random initialize linear projection.

model_seq.utils.**init_lstm**(*input_lstm*)
random initialize lstms

model_seq.utils.**log_sum_exp**(*vec*)
log sum exp function.

Parameters **vec** (torch.FloatTensor, required.) – input vector, of shape(ins_num, from_tag_size, to_tag_size)

Returns **sum** – log sum exp results, tensor of shape (ins_num, to_tag_size)

Return type torch.FloatTensor.

model_seq.utils.**repackage_hidden**(*h*)
Wraps hidden states in new Variables, to detach them from their history

Parameters **h** (Tuple or Tensors, required.) – Tuple or Tensors, hidden states.

Returns **hidden** – detached hidden states

Return type Tuple or Tensors.

model_seq.utils.**to_scalar**(*var*)
convert a tensor to a scalar number

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

Index

I

L

M

O

P

R

S

T

U

V