torchtext Documentation

Release master (0.3.0)

Torch Contributors

Sep 24, 2018

Package Reference

The `torchtext` package consists of data processing utilities and popular datasets for natural language.

CHAPTER 1

torchtext.data

The data module provides the following:

- Ability to define a preprocessing pipeline
- Batching, padding, and numericalizing (including building a vocabulary object)
- Wrapper for dataset splits (train, validation, test)
- Loader a custom NLP dataset

1.1 Dataset, Batch, and Example

1.1.1 Dataset

1.1.2 TabularDataset

1.1.3 Batch

1.1.4 Example

1.2 Fields

1.2.1 RawField

1.2.2 Field

1.2.3 ReversibleField

1.2.4 SubwordField

1.2.5 NestedField

1.3 Iterators

1.3.1 Iterator

1.3.2 BucketIterator

1.3.3 BPTTIterator

1.4 Pipeline

1.4.1 Pipeline

1.5 Functions

1.5.1 batch

1.5.2 pool

1.5.3 get_tokenizer

1.5.4 interleave_keys

CHAPTER 2

torchtext.datasets

All datasets are subclasses of `torchtext.data.Dataset`, which inherits from `torch.utils.data.Dataset` i.e, they have `split` and `iters` methods implemented.

General use cases are as follows:

Approach 1, `splits`:

```
# set up fields
TEXT = data.Field(lower=True, include_lengths=True, batch_first=True)
LABEL = data.Field(sequential=False)

# make splits for data
train, test = datasets.IMDB.splits(TEXT, LABEL)

# build the vocabulary
TEXT.build_vocab(train, vectors=GloVe(name='6B', dim=300))
LABEL.build_vocab(train)

# make iterator for splits
train_iter, test_iter = data.BucketIterator.splits(
    (train, test), batch_size=3, device=0)
```

Approach 2, `iters`:

```
# use default configurations
train_iter, test_iter = datasets.IMDB.iters(batch_size=4)
```

The following datasets are available:

Datasets

- *Sentiment Analysis*
  - *SST*

2.1 Sentiment Analysis

2.1.1 SST

2.1.2 IMDb

2.2 Question Classification

2.2.1 TREC

2.3 Entailment

2.3.1 SNLI

2.3.2 MultiNLI

2.4 Language Modeling

Language modeling datasets are subclasses of `LanguageModelingDataset` class.

2.4.1 WikiText-2

2.4.2 WikiText103

2.4.3 PennTreebank

2.5 Machine Translation

Machine translation datasets are subclasses of `TranslationDataset` class.

2.5.1 Multi30k

2.5.2 IWSLT

2.5.3 WMT14

2.6 Sequence Tagging

Sequence tagging datasets are subclasses of `SequenceTaggingDataset` class.

2.6.1 UDPOS

2.6.2 CoNLL2000Chunking

2.7 Question Answering

2.7.1 BABI20

CHAPTER 3

torchtext.vocab

3.1 Vocab

3.2 SubwordVocab

3.3 Vectors

3.3.1 Pretrained Word Embeddings

3.4 GloVe

3.5 FastText

3.6 CharNGram

3.6.1 Misc.

3.7 _default_unk_index

CHAPTER 4

torchtext.utils

4.1 reporthook

4.2 download_from_url

CHAPTER 5

Examples

Ability to describe declaratively how to load a custom NLP dataset that's in a "normal" format:

```
pos = data.TabularDataset(
path='data/pos/pos_wsj_train.tsv', format='tsv',
fields=[('text', data.Field()),
        ('labels', data.Field())])

sentiment = data.TabularDataset(
    path='data/sentiment/train.json', format='json',
    fields={'sentence_tokenized': ('text', data.Field(sequential=True)),
            'sentiment_gold': ('labels', data.Field(sequential=False))})
```

Ability to define a preprocessing pipeline:

```
src = data.Field(tokenize=my_custom_tokenizer)
trg = data.Field(tokenize=my_custom_tokenizer)
mt_train = datasets.TranslationDataset(
    path='data/mt/wmt16-ende.train', exts=('.en', '.de'),
    fields=(src, trg))
```

Batching, padding, and numericalizing (including building vocabulary object):

```
# continuing from above
mt_dev = data.TranslationDataset(
    path='data/mt/newstest2014', exts=('.en', '.de'),
    fields=(src, trg))
src.build_vocab(mt_train, max_size=80000)
trg.build_vocab(mt_train, max_size=40000)
# mt_dev shares the fields, so it shares their vocab objects

train_iter = data.BucketIterator(
    dataset=mt_train, batch_size=32,
    sort_key=lambda x: data.interleave_keys(len(x.src), len(x.trg)))
# usage
```

(continues on next page)

(continued from previous page)

```
>>>next(iter(train_iter))
<data.Batch(batch_size=32, src=[LongTensor (32, 25)], trg=[LongTensor (32, 28)])>
```

Wrapper for dataset splits (train, validation, test):

```
TEXT = data.Field()
LABELS = data.Field()

train, val, test = data.TabularDataset.splits(
    path='/data/pos_wsj/pos_wsj', train='_train.tsv',
    validation='_dev.tsv', test='_test.tsv', format='tsv',
    fields=[('text', TEXT), ('labels', LABELS)])

train_iter, val_iter, test_iter = data.BucketIterator.splits(
    (train, val, test), batch_sizes=(16, 256, 256),
    sort_key=lambda x: len(x.text), device=0)

TEXT.build_vocab(train)
LABELS.build_vocab(train)
```

CHAPTER 6

Indices and tables

- genindex
- modindex
- search