hugdatafast

Oct 06, 2020

About

This package is to provide a elegant bridge between fastai and huggingface/datasets and some handy data transforms for NLPers.

Author: Richard Wang

Twitter: Richard Wang (You can follow to get news of the package if there is. Or see my recent research.)

CHAPTER 1

Installation

```
pip install hugdatafast
```

This will install also the lastest `fastai` and `datasets`.

1.1 Get Started

1.1.1 Base use case

```
>>> from datasets import load_dataset
>>> from hugdatafast import *
```

Note: This will also implicitly do `from fastai.text.all import *`

Can you turn your data pipeline into only 3 lines ?

```
>>> datasets = load_dataset('glue', 'cola')
-> {'train': datasets.Dataset, 'validation': datasets.Dataset, 'test': datasets.
↪Dataset}
>>> tokenized_datasets = datasets.map(simple_tokenize_func({'sentence':'text_idxs'},␣
↪hf_tokenizer))
>>> dls = HF_Datasets(tokenized_datasets, cols=['text_idxs', 'label'], hf_toker=hf_
↪tokenizer).dataloaders(bs=64)
```

Now you can enjoy

1. `show_batch()` of fastai n Inspect your processed data and quickly check if there is anything wrong with your data processing.

```
>>> dls.show_batch(max_n=2)
                                                                                                                    ␣
↪                      text_idxs       label
-------------------------------------------------------------------------------------------------------------------
↪--------------------------------------------------
0  everybody who has ever , worked in any office which contained any type ##writer␣
↪which had ever been used to type any      1
   letters which had to be signed by any administrator who ever worked in any␣
↪department like mine will know what i mean .
-------------------------------------------------------------------------------------------------------------------
↪--------------------------------------------------
1  playing with matches is ; lots of fun , but doing , so and empty ##ing gasoline␣
↪from one can to another at the same       1
   time is a sport best reserved for arson ##s . [PAD] [PAD] [PAD] [PAD] [PAD] [PAD]␣
↪[PAD] [PAD] [PAD] [PAD]
```

2. Train model on the data using fastai, and also show the prediction

```
>>> learn = Learner(dls, your_model, loss_func=CrossEntropyLossFlat())
>>> learn.fit(3)
>>> learn.show_results()
                                                                                                    text_
↪idxs     label label_
-------------------------------------------------------------------------------------------------------------------
↪---------------
0   [CLS] scientists at the south hanoi institute of technology have succeeded in␣
↪raising   1          1
  one dog with five legs , another with a cow ' s liver , and a third with no head .␣
↪[SEP]
-------------------------------------------------------------------------------------------------------------------
↪---------------
1 [CLS] as a teacher , you have to deal simultaneously with the administration ' s␣
↪pressure   0    1
  on you to succeed , and the children ' s to be a nice guy . [SEP] [PAD] [PAD]
```

3. Use it as normal Dataloaders if you don't use fastai .

```
>>> train_dataloader, val_dataloader, test_dataloader = dls[0], dls[1], dls[2]
>>> for b in train_dataloader: break
```

1.1.2 Other use cases

1. Use your own dataset ?
   - datasets.Dataset s from local structured files (csv, json, . . .)
   - datasets.Dataset s from custom loading script
2. Need to combine examples to generate new example ? (e.g. Traditional language model)

```
>>> lm_datasets = LMTransform(datasets, max_len=20, text_col='text_idxs').map()
>>> hf_tokenizer.decode(lm_datasets['validation'][-1]['x_text'])
. john talked to bill about himself
>>> hf_tokenizer.decode(lm_datasets['validation'][-1]['y_text'])
john talked to bill about himself.
```

If you want to implement your own logic to combine examples, try to extend `CombineTransform`.

1.1.3 `hugdatafast` in practice

You can see how to use `hugdatafast` in the real situations. Also, You are welcome to share how you use `hugdatafast` in your project, contact me via github or twitter to put your project link here.

- electra_pytorch : Pretrain ELECTRA and finetune on GLUE benchmark

1.2 `hugdatafast.fastai`

1.2.1 Module Contents

Classes

MySortedDL	A `DataLoader` that do smart batching and dynamic padding. Different from `SortedDL`, it automatically pad every attribute of samples, is able to filter samples, and can be cached to sort/filter only at first time.
HF_Dataset	A wrapper for `datasets.Dataset`. It will behavior like original `datasets.Dataset`,
HF_Datasets	Function as `fastai.data.core.Datasets` to create `fastai.data.core.Dataloaders` from a group of :class:‘datasets.Dataset‘s

class hugdatafast.fastai.**MySortedDL**(*dataset*, *srtkey_fc=None*, *filter_fc=False*, *pad_idx=None*, *cache_file=None*, ***kwargs*)

A `DataLoader` that do smart batching and dynamic padding. Different from `SortedDL`, it automatically pad every attribute of samples, is able to filter samples, and can be cached to sort/filter only at first time.

Parameters

- **dataset** (*HF_Dataset*) – Actually any object implements `__len__` and `__getitem__` that return a tuple as a sample.
- **srtkey_fc** (`*args->int`, optional) – Get key for decending sorting from a sample .
  - If `None`, sort by length of first element of a sample.
  - If `False`, not sort.
- **filter_fc** (`*args->bool`, optional) – Return `True` to keep the sample.
- **pad_idx** (`int`, optional) – pad each attribute of samples to the max length of its max length within the batch.
  - If `List[int]`, specify pad_idx for each attribute of a sample. e.g. a sample is a tuple (masked_inputs, labels), *pad_idx=[0 ,-100]* pad masked_inputs with 0, labels with -100.
  - If `False`, do no padding.
  - If `None`, try `dataset.pad_idx`, do no padding if no such attribute.
- **cache_file** (`str`, optional) – Path of a json file to cache info for sorting and filtering.
- **kwargs** – key arguments for *TfmDl* or *DataLoader*

Example

```
>>> samples = [ (torch.tensor([1]), torch.tensor([7,8]), torch.tensor(1)),,
...              (torch.tensor([2,3]), torch.tensor([9,10,11]), torch.tensor(2)),
...              (torch.tensor([4,5,6]), torch.tensor([11,12,13,14]), torch.
↪tensor(3)), ]
... dl = MySortedDL(samples,
...                 srtkey_fc=lambda *args: len(args[0]),
...                 filter_fc=lambda x1,y1: y1<3,
...                 pad_idx=-1,
...                 cache_file='/tmp/cache.json', # calls after this will load␣
↪cache
...                 bs=999, # other parameters go to `TfmDL` and `DataLoader`
...                 )
... dl.one_batch()
(tensor([[ 2,  3],
         [ 1, -1]]),
 tensor([[ 9, 10, 11],
         [ 7,  8, -1]]),
 tensor([2, 1]))
```

class hugdatafast.fastai.**HF_Dataset**(*hf_dset*, *cols=None*, *hf_toker=None*, *neat_show=False*, *n_inp=1*)

A wrapper for `datasets.Dataset`. It will behavior like original `datasets.Dataset`, but also function as a `fastai.data.core.datasets` that provides samples and decodes.

Parameters

- **hf_dset** (`datasets.Dataset`) – Prerocessed Hugging Face dataset to be wrapped.
- **cols** (*dict, optional*) – columns of `datasets.Dataset` to be used to construct samples, and (optionally) semantic tensor type for each of those columns to decode.
  - cols(`Dict[Fastai Semantic Tensor]`): encode/decode column(key) with semantic tensor type(value). If {value} is `noop`, semantic tensor of the column is by default *TensorTuple*.
  - cols(`list[str]`): specify only columns and take default setting for semantic tensor type of them.
    - if length is 1, regard the 1st element as *TensorText*
    - if length is 2, regard the 1st element as *TensorText*, 2nd element as *TensorCategory*
    - Otherwise, regard all elements as *TensorTuple*
  - cols(None): pass `hf_dset.column_names` (list[str]) as cols.
- **hf_toker** (`transformers.PreTrainedTokenizer`, optional) – Hugging Face tokenizer, used in decode and provide `pad_idx` for dynamic padding
- **neat_show** (*bool, optional*) – Show the original sentence instead of tokens joined by space.
- **n_inp** (*int, optional*) – take the first `n_inp` columns of `cols` as x, and the rest as y .

Example

```
>>> tokenized_cola_train_set[0]
{'sentence': "Our friends won't buy this analysis, let alone the next one we␣
↪propose.",
 'label': 1,
 'idx': 0,
 'text_idxs': [ 2256,  2814,  2180,  1005,  1056,  4965,  2023,  4106,  1010, ␣
↪2292, 2894,  1996,  2279,  2028,  2057, 16599,  1012]}
>>> hf_dset = HF_Datset(tokenized_cola_train_set, cols=['text_idxs', 'label'], hf_
↪toker=tokenizer_electra_small_fast)
>>> len(hf_dset), hf_dset[0]
8551, (TensorText([ 2256,  2814,  2180,  1005,  1056,  4965,  2023,  4106,  1010,␣
↪ 2292, 2894,  1996,  2279,  2028,  2057, 16599,  1012]), TensorCategory(1))
>>> hf_dset.decode(hf_dset[0])
("our friends won ' t buy this analysis , let alone the next one we propose .", '1
↪')
# The wrapped dataset "is" also the original huggingface dataset
>>> hf_dset.column_names == tokenized_cola_train_set.column_names
True
# Manually specify `cols` with dict, here it is equivalent to the above. And␣
↪addtionally, neatly decode samples.
>>> neat_hf_dset = HF_Datset(tokenized_cola_train_set, {'text_idxs':TensorText,
↪'label':TensorCategory}, hf_toker=tokenizer_electra_small_fast, neat_show=True)
>>> neat_hf_dset.decode(neat_hf_dset[0])
("our friends won't buy this analysis, let alone the next one we propose.", '1')
# Note: Original set will be set to Pytorch format with columns specified in␣
↪`cols`
>>> tokenized_cola_train_set[0]
{'label': tensor(1),
 'text_idxs': tensor([ 2256,  2814,  2180,  1005,  1056,  4965,  2023,  4106, ␣
↪1010,  2292, 2894,  1996,  2279,  2028,  2057, 16599,  1012])}
```

class hugdatafast.fastai.**HF_Datasets**(*hf_dsets: dict*, *test_with_y=False*, ***kwargs*)

Function as `fastai.data.core.Datasets` to create `fastai.data.core.Dataloaders` from a group of :class:‘datasets.Dataset‘s

Parameters

- **hf_dsets** (*Dict[datasets.Dataset]*) – Prerocessed Hugging Face Datasets, {key} is split name, {value} is `datasets.Dataset`, order will become the order in `fastai.data.core.Dataloaders`.
- **test_with_y** (*bool, optional*) – Whether the test set come with y (answers) but not with fake y (e.g. all -1 label). If `False`, tell only test set to construct samples from first `n_inp` columns (do not output fake y). And all datasets passed in `hf_dsets` with its name starts with “test” will be regarded as test set.
- **kwargs** – Passed to *HF_Dataset*. Be sure to pass arguments that *HF_Dataset* needs !!

dataloaders(*self*, *device='cpu'*, *cache_dir=None*, *cache_name=None*, *dl_kwargs=None*, ***kwargs*)

Parameters

- **device** (*str*) – device where outputed batch will be on. Because a batch will be loaded to test when creating :class: *fastai.data.core.Dataloaders*, to prevent always leaving a batch of tensor in cuda:0, using default value cpu and then `dls.to(other device)` at the time you want is suggested.

- **cache_dir** (*str, optional*) – directory to store caches of *MySortedDL*. if None, use cache directory of the first datasets.Dataset in hf_dsets that passed to **:method:'HF_Datasets.__init__'**.
- **cache_name** (*str, optional*) – format string that includes one param "{split}", which will be replaced with name of split as cache file name under *cache_dir* for each split. If None, tell :class:MySortedDL don't do caching.
- **dl_kwargs** (*list[dict], optional*) – ith item is addtional kwargs to be passed to initialization of ith dataloader for ith split
- **kwargs** – Passed to fastai.data.core.FilteredBase.dataloaders()

Example

```
>>> tokenized_cola
{'train': datasets.Dataset, 'validation': datasets.Dataset, 'test': datasets.
↪Dataset}
>>> tokenized_cola['test'][0]
{'sentence': 'Bill whistled past the house.',
 'label': -1, # Fake label. True labels are not open to the public.
 'idx': 0,
 'text_idxs': [3021, 26265, 2627, 1996, 2160, 1012]}
>>> dls = HF_Datasets(tokenized_cola,
...                   cols=['text_idxs', 'label'], hf_toker=hf_tokenizer,  #␣
↪args for HF_Dataset
...                   ).dataloaders(bs=32 , cache_name="dl_cached_for_{split}
↪") # args for MySortedDL
>>> dls.show_batch(max_n=2)
                                                                                ␣
↪                                      text_idxs           label
------------------------------------------------------------------------------
↪----------------------------------------------------------------
0  everybody who has ever, worked in any office which contained any␣
↪typewriter which had ever been used to type any letters which had    1
   to be signed by any administrator who ever worked in any department like␣
↪mine will know what i mean.
------------------------------------------------------------------------------
↪----------------------------------------------------------------
1  playing with matches is ; lots of fun, but doing, so and emptying gasoline␣
↪from one can to another at the same time is a sport best   1
   reserved for arsons.
# test set won't produce label becuase of `test_with_y=False`
>>> dls[-1].show_batch(max_n=2)
                                                                                ␣
↪ text_idxs
------------------------------------------------------------------------------
↪------------
0  cultural commissioner megan smith said that the five ` ` soundscape''␣
↪pieces would ` `
   give a festive air to park square, they're fun and interesting''.
------------------------------------------------------------------------------
↪------------
1  wendy is eager to sail around the world and bruce is eager to climb␣
↪kilimanjaro, but
   neither of them can because money is too tight.
```

1.3 hugdatafast.transform

1.3.1 Module Contents

Classes

SimpleTokenize	Initialize self. See help(type(self)) for accurate signature.
CombineTransform	Base Class for Transform that combine multiple original samples into a new sample.
LMTransform	Transform any dataset has tokenized text into dataset (autotgressive) language model.
ELECTRADataTransform	Process any text corpus for ELECTRA's use

Functions

my_map(self: datasets.arrow_dataset.Dataset, *args, **kwargs)	The same as `datasets.arrow_dataset.Dataset` , but it can add cache directory and .arrow to cache_file_name autmomatically for us.
my_map(self: datasets.arrow_dataset.Dataset, *args, **kwargs)	The same as `datasets.arrow_dataset.Dataset` , but it can add cache directory and .arrow to cache_file_name autmomatically for us.

hugdatafast.transform.**my_map**(*self: datasets.arrow_dataset.Dataset*, **args*, ***kwargs*)

The same as `datasets.arrow_dataset.Dataset` , but it can add cache directory and .arrow to cache_file_name autmomatically for us.

Example

```
>>> dataset.map(a_func, cache_file_name='processed')
# cache file path become "<dataset cache directory>/processed.arrow"
```

hugdatafast.transform.**my_map**(*self: datasets.dataset_dict.DatasetDict*, **args*, ***kwargs*)

The same as `datasets.dataset_dict.DatasetDict` , but it can infer cache names for us.

Example

```
>>> datasets.map(a_func, cache_file_names='processed_{split}')
# cache file paths : "<dataset cache directory>/processed_train.arrow", "<dataset␣
↪cache directory>/processed_validation.arrow", "<dataset cache directory>/
↪processed_test.arrow"
```

class hugdatafast.transform.**SimpleTokenize**(*cols*, *hf_toker*)

Initialize self. See help(type(self)) for accurate signature.

class hugdatafast.transform.**CombineTransform**(*hf_dset*, *in_cols*, *out_cols*, *drop_last=False*)

Base Class for Transform that combine multiple original samples into a new sample.

Parameters

- **hf_dset** (Dataset or DatasetDict) – The Hugging Face dataset(s) to do the transformation
- **in_cols** (*List[str]*) – names of input columns that used to produce samples
- **out_cols** (*List[str]*) – names of output columns to put combined samples.
- **(Optional[bool], default** (*drop_last `*) – *False*): whether to drop the last accumulated sample.

reset_states(*self*)
Child Class should implement this method.

Reset all containers, flags to their initial values.

accumulate(*self*, **args*)
Child Class should implement this method.

Given a example, do *self.commit_example(self.create_example()) when a new combined sample is ready.*
:param args: values of inp_cols (passed to __init__()) of an example

create_example(*self*)
Child Class should implement this method.

Use internal states stored in the child class instance to create a combined example (dict).

When nothing can't be created, return None or raise any exception to show it.

map(*self*, *batch_size=1000*, *cache_file_name=None*, ***kwargs*)

Parameters

- **batch_size** (*int*) – See datasets.Dataset.map, shouldn't be None here
- **cache_file_name** – The same with the one of *my_map()*
- **kwargs** – passed to datasets.Dataset.map

class hugdatafast.transform.**LMTransform**(*tokenized_hf_dset*, *max_len*, *text_col*, *x_text_col='x_text'*, *y_text_col='y_text'*, ***kwargs*)

Transform any dataset has tokenized text into dataset (autotgressive) language model. !! Caution: This span context window across examples. So make sure your texts in examples of the datasets are consecutive or relative. Or you are knowing what you are doing.

Parameters

- **tokenized_hf_dset** (Dataset or DatasetDict) – tokenized Hugging Face dataset(s) to do LM transform
- **max_len** (*int*) – the length of a sentence
- **text_col** (*str*) – the name of column that contains tokenized text (ids) of tokenized_hf_dset
- **x_text_col** (*str*) – the name of the output column
- **y_text_col** (*str*) – the name fo the output column
- **kwargs** – passed to :class:CombineTransform

Example

```
>>> lm_dataset = LMTransform(tokenized_cola['validation'], max_len=20, text_col=
↪'text_idxs').map()
>>> lm_dataset[0]
{'x_text': [ 1996, 11279,  8469,  1996,  9478,  3154,  1997,  1996,  5749,  1012,
    1996, 15871,  2081,  1996,  8164,  7683,  2058,  1996,  4139,  3240],
 'y_text': [11279,  8469,  1996,  9478,  3154,  1997,  1996,  5749,  1012,  1996,
   15871,  2081,  1996,  8164,  7683,  2058,  1996,  4139,  3240,  1012]}
```

reset_states (*self*)
Child Class should implement this method.

Reset all containers, flags to their initial values.

create_example (*self*)
Child Class should implement this method.

Use internal states stored in the child class instance to create a combined example (dict).

When nothing can't be created, return `None` or raise any exception to show it.

accumulate (*self*, *text*)
Child Class should implement this method.

Given a example, do *self.commit_example(self.create_example()) when a new combined sample is ready.*
:param args: values of `inp_cols` (passed to `__init__()`) of an example

class hugdatafast.transform.**ELECTRADataTransform**(*hf_dset*, *is_docs*, *text_col*, *max_length*, *hf_toker*, *delimiter='n'*, ***kwargs*)

Process any text corpus for ELECTRA's use

Parameters

- **hf_dset** (`Dataset` or `DatasetDict`) – **untokenized** Hugging Face dataset(s) to do the transform
- **is_docs** (*bool*) – Whether each sample of this dataset is a doc
- **text_col** (*str*) – the name of column of the dataset contains text
- **max_length** (*str*) – max length of each sentence
- **hf_toker** (`transformers.PreTrainedTokenizer`) – Hugging Face tokenizer
- **delimiter** (*str*) – what is the delimiter to segment sentences in the input text
- **kwargs** – passed to *CombineTransform*

reset_states (*self*)
Child Class should implement this method.

Reset all containers, flags to their initial values.

accumulate (*self*, *text*)
Child Class should implement this method.

Given a example, do *self.commit_example(self.create_example()) when a new combined sample is ready.*
:param args: values of `inp_cols` (passed to `__init__()`) of an example

create_example (*self*)
Child Class should implement this method.

Use internal states stored in the child class instance to create a combined example (dict).

When nothing can't be created, return `None` or raise any exception to show it.

add_line(*self*, *tokids*)
Adds a line of text to the current example being built.

Python Module Index

h

Index