Pythonkurs - 13 - Fredag - FastAI Text

December 6, 2024

[1]:
```
import warnings
warnings.simplefilter(action='ignore', category=UserWarning)
```

[2]:
```
from fastai.text.all import *
```

[3]:
```
print(torch.backends.mps.is_built()) # Apple M-series␣
  ↪metal-performance-shaders-framework
print(torch.backends.mps.is_available()) # Apple M-series␣
  ↪metal-performance-shaders-framework

mps_device = default_device()
print(mps_device)
```

```
True
True
mps
```

[5]:
```
path = untar_data(URLs.IMDB) # https://docs.fast.ai/data.external.html
```

[6]:
```
path.ls()
(path/'train').ls()
```

[6]:
```
(#5) [Path('/Users/kristianbotnen/.fastai/data/imdb/train/.DS_Store'),Path('/Use
rs/kristianbotnen/.fastai/data/imdb/train/neg'),Path('/Users/kristianbotnen/.fas
tai/data/imdb/train/pos'),Path('/Users/kristianbotnen/.fastai/data/imdb/train/un
supBow.feat'),Path('/Users/kristianbotnen/.fastai/data/imdb/train/labeledBow.fea
t')]
```

[7]:
```
import shutil
#from pathlib import Path

def create_subset(src, dest, num_samples=256):
    dest.mkdir(parents=True, exist_ok=True)
    files = list(src.glob('*'))[:num_samples]
    for file in files:
        shutil.copy(file, dest/file.name)
```

```
train_unsup = path/'unsup'
train_pos = path/'train'/'pos'
train_neg = path/'train'/'neg'
test_pos = path/'test'/'pos'
test_neg = path/'test'/'neg'

# Create subset directories
top_datapath = path.parent
subset_path = top_datapath/'subset'

(subset_path/'unsup').mkdir(parents=True, exist_ok=True)
(subset_path/'train'/'pos').mkdir(parents=True, exist_ok=True)
(subset_path/'train'/'neg').mkdir(parents=True, exist_ok=True)
(subset_path/'test'/'pos').mkdir(parents=True, exist_ok=True)
(subset_path/'test'/'neg').mkdir(parents=True, exist_ok=True)

# Copy files to subset directories
create_subset(train_unsup, subset_path/'unsup')
create_subset(train_pos, subset_path/'train'/'pos')
create_subset(train_neg, subset_path/'train'/'neg')
create_subset(test_pos, subset_path/'test'/'pos')
create_subset(test_neg, subset_path/'test'/'neg')
```

[8]:
```
# Prepare the dataset. Both the training set and the validation set.
datablock = DataBlock(
    blocks=(TextBlock.from_folder(subset_path), CategoryBlock), # Input is␣
 ↪text, Output is categories (positive / negative).
    get_items=get_text_files, # Get text files in path recursively, only in␣
 ↪folders, if specified.
    splitter=GrandparentSplitter(valid_name='test'), # Split items from the␣
 ↪grand parent folder names (train_name and valid_name).
    get_y=parent_label, # Label item with the parent folder name.
)

dataloaders = datablock.dataloaders(subset_path, bs=16, device=mps_device) #␣
 ↪https://docs.fast.ai/data.transforms.html
```

[9]:
```
datablock.summary(subset_path)
```

```
Setting-up type transforms pipelines
Collecting items from /Users/kristianbotnen/.fastai/data/subset
Found 1536 items
2 datasets of sizes 512,512
Setting up Pipeline: Tokenizer -> Numericalize
Setting up Pipeline: parent_label -> Categorize -- {'vocab': None, 'sort': True,
'add_na': False}

Building one sample
```

```
  Pipeline: Tokenizer -> Numericalize
    starting from
      /Users/kristianbotnen/.fastai/data/subset/train/neg/1821_4.txt
    applying Tokenizer gives
      ['xxbos', 'xxmaj', 'working', 'with', 'one', 'of', 'the', 'best', 'xxmaj',
'shakespeare', 'sources', ',', 'this', 'film', 'manages', 'to', 'be',
'creditable', 'to', 'it', "'s", 'source', ',', 'whilst', 'still', 'appealing',
'to', 'a', 'wider', 'audience', '.', '\n\n', 'xxmaj', 'branagh', 'steals',
'the', 'film', 'from', 'under', 'xxmaj', 'fishburne', "'s", 'nose', ',', 'and',
'there', "'s", 'a', 'talented', 'cast', 'on', 'good', 'form', '.']
    applying Numericalize gives
      TensorText of size 54
  Pipeline: parent_label -> Categorize -- {'vocab': None, 'sort': True,
'add_na': False}
    starting from
      /Users/kristianbotnen/.fastai/data/subset/train/neg/1821_4.txt
    applying parent_label gives
      neg
    applying Categorize -- {'vocab': None, 'sort': True, 'add_na': False} gives
      TensorCategory(0)

Final sample: (TensorText([   2,    8,  739,   30,   44,   14,    9,  138,    8,
3284, 5455,
              11,   20,   32,  866,   15,   43,    0,   15,   18,   23, 3134,
              11, 1661,  150, 3073,   15,   13,    0,  313,   10,   25,    8,
               0, 1483,    9,   32,   53,  454,    8,    0,   23, 3883,   11,
              12,   56,   23,   13, 1352,  184,   36,   68,  711,   10]),
TensorCategory(0))


Collecting items from /Users/kristianbotnen/.fastai/data/subset
Found 1536 items
2 datasets of sizes 512,512
Setting up Pipeline: Tokenizer -> Numericalize
Setting up Pipeline: parent_label -> Categorize -- {'vocab': None, 'sort': True,
'add_na': False}
Setting up after_item: Pipeline: ToTensor
Setting up before_batch: Pipeline: Pad_Chunk -- {'pad_idx': 1, 'pad_first':
True, 'seq_len': 72}
Setting up after_batch: Pipeline:

Building one batch
Applying item_tfms to the first sample:
  Pipeline: ToTensor
    starting from
      (TensorText of size 54, TensorCategory(0))
    applying ToTensor gives
      (TensorText of size 54, TensorCategory(0))
```

```
Adding the next 3 samples

Applying before_batch to the list of samples
  Pipeline: Pad_Chunk -- {'pad_idx': 1, 'pad_first': True, 'seq_len': 72}
    starting from
      [(TensorText of size 54, TensorCategory(0)), (TensorText of size 234,
TensorCategory(0)), (TensorText of size 165, TensorCategory(0)), (TensorText of
size 494, TensorCategory(0))]
    applying Pad_Chunk -- {'pad_idx': 1, 'pad_first': True, 'seq_len': 72} gives
      ((TensorText of size 494, TensorCategory(0)), (TensorText of size 494,
TensorCategory(0)), (TensorText of size 494, TensorCategory(0)), (TensorText of
size 494, TensorCategory(0)))

Collating items in a batch

No batch_tfms to apply
```

[10]:
```
dataloaders.show_batch(max_n=3)
```

```
<IPython.core.display.HTML object>
```

[11]:
```
print(type(dataloaders))
print(len(dataloaders))
print(len(dataloaders.train_ds), len(dataloaders.valid_ds))

for i, sample in enumerate(dataloaders.train_ds):
    print(sample)
    if i == 2:
        break
```

```
<class 'fastai.data.core.DataLoaders'>
2
512 512
(TensorText([   2,    8,  739,   30,   44,   14,    9,  138,    8, 3284, 5455,
               11,   20,   32,  866,   15,   43,    0,   15,   18,   23, 3134,
               11, 1661,  150, 3073,   15,   13,    0,  313,   10,   25,    8,
                0, 1483,    9,   32,   53,  454,    8,    0,   23, 3883,   11,
               12,   56,   23,   13, 1352,  184,   36,   68,  711,   10]),
TensorCategory(0))
(TensorText([   2,    8,   88,   71, 5475,    7,   19,   11,    9,  218,  677,
              152,   17, 2834,   12,   19,  305,    9,   27,  199,  653,   15,
              126,   10,  206,   11,   47, 6665,   15,  117, 5475,    7, 1241,
               12,    5,  156,   19,   10,    8, 1795,   87,   11,  179,  115,
              677,  173, 2941,  208,  125,   47, 2083,    9,   98,   44,   11,
               19,  490,   11, 2096,    0,    5,  156,   72,    8,  172,   11,
               91,    8,  559,  330,   16, 3120,   14, 6666,    9, 1067,   22,
              155,   17,    8,    0,  368,   70,   47,  797,  178,   44,   14,
              164,    0,    0,   14,   13,   27,   72,   22,    8, 5475,    7,
             4647,   78,   37,   43,  985,   13,  111,   27,   11,   17,  195,
               18,   78,   37,   43,   74,  985,   54,    0,   14,   13,  111,
               27,   11,   29,   18,  500,   73,   15,   75,   99,   21,   10,
                8,   28,   19, 2675,   15,  786,  170,   18,   11,   19, 1988,
               21,   79,  570,  677,   15,    0,   11,   12,   19, 5476,    0,
               21,    9,  128,  287,   53,    9, 1199,   70, 6667,  183,    9,
                7,  264,   12,  523,   87,   10,   86,   47,   80,  120,   21,
                0,    9,   96,   52,   90,    0,   17,    9,   97,  115,   73,
               28,  170,   47,   93,  186,   53,    9, 1443,  973,   70,  117,
                9,   27,   38,   30,    9, 1727,  181,    0,   34,  122,  151,
               72, 1254,   11,   19,   70,  130, 1937, 2146,   20,   27,   11,
               19,  490,   11,   56,   40, 6668,  748,   15,  185,  216,   21,
             2874,   88,   10]), TensorCategory(0))
(TensorText([   2,    8, 6685,   50,    8,   20,   44,   26,   13,  242, 1369,
               15,  786,  183,   10,    8,   18,   63,   13, 1036,   12, 1075,
              954,   11,   31,   18,   46,  292,   15,  495,   53,   56,   10,
                8, 3526,    8,    0,   16,  241,  222, 5188,   12,  579,   11,
               12,   39,   94,   37, 3995,   17,   20,   44,   10,    8, 4655,
                8,    0,    8,    0,   12,    8,  328,    8, 6686,  750,   17,
             1071,  509,  416,   10,    8, 1536,    8, 2976,   12,    8, 2256,
                8, 6537,   11,  756,  441,   11,  247,   18,  113,  140,    9,
              415,   10,   19,  133,   21,   23,    9,  113,    9,  601,   82,
              458,   11,   12,   18,   23,  253,   15, 1777,  175,   11,   69,
                9,  243,   12,  166,   41,  118,   58,  157, 6687,   10,    8,
               64,   33,  186,  173,   15,   41, 3537,  121,   11,   21,   23,
                7,  576,   11,   31,   33,  237, 3113,   41,   68,  825,   15,
              117,   18,  203,   10,    8,   18,   86,   35,  154,   10,    8,
              603,  269, 1701,   53,   20,   27,   64,   45,   46,  707,   10]),
TensorCategory(0))
```

0.1 Train and tune our model

[12]:
```
# Train and tune our model.
learn = text_classifier_learner(dataloaders, AWD_LSTM, drop_mult=0.5,␣
 ↪metrics=accuracy)
```

```
/opt/miniconda3/envs/pythonki/lib/python3.12/site-
packages/fastai/text/learner.py:149: FutureWarning: You are using `torch.load`
with `weights_only=False` (the current default value), which uses the default
pickle module implicitly. It is possible to construct malicious pickle data
which will execute arbitrary code during unpickling (See
https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models for
more details). In a future release, the default value for `weights_only` will be
flipped to `True`. This limits the functions that could be executed during
unpickling. Arbitrary objects will no longer be allowed to be loaded via this
mode unless they are explicitly allowlisted by the user via
`torch.serialization.add_safe_globals`. We recommend you start setting
`weights_only=True` for any use case where you don't have full control of the
```

```
loaded file. Please open an issue on GitHub for any issues related to this
experimental feature.
  wgts = torch.load(wgts_fname, map_location = lambda storage,loc: storage)
```

[13]:
```
learn.fine_tune(4, 1e-2)
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

[14]:
```
learn.show_results()
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

[16]:
```
# Use our model by passing it a review.
category,_,probs = learn.predict("I really liked that movie")

print(f"This is a: {category}.")
print(f"Probability it's a positive: {probs[1]:.4f}")

category,_,probs = learn.predict("I did not like that movie, it was awful. It␣
↪was the worst thing I have ever seen")

print(f"This is a: {category}.")
print(f"Probability it's a positive: {probs[1]:.4f}")
```

```
<IPython.core.display.HTML object>
```

```
This is a: pos.
Probability it's a positive: 0.8641
```

```
<IPython.core.display.HTML object>
```

```
This is a: neg.
Probability it's a positive: 0.3807
```

0.2 ULMFiT

[17]:
```
dataloaders_lm = TextDataLoaders.from_folder(subset_path/'unsup', is_lm=True,␣
↪valid_pct=0.1)
```

[18]:
```
dataloaders_lm.show_batch(max_n=3)
```

```
<IPython.core.display.HTML object>
```

[19]:
```
llm_learn = language_model_learner(dataloaders_lm, AWD_LSTM, metrics=[accuracy,␣
↪Perplexity()], path=subset_path/'unsup', wd=0.1)
```

```
/opt/miniconda3/envs/pythonki/lib/python3.12/site-
packages/fastai/text/learner.py:149: FutureWarning: You are using `torch.load`
with `weights_only=False` (the current default value), which uses the default
```

```
pickle module implicitly. It is possible to construct malicious pickle data
which will execute arbitrary code during unpickling (See
https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models for
more details). In a future release, the default value for `weights_only` will be
flipped to `True`. This limits the functions that could be executed during
unpickling. Arbitrary objects will no longer be allowed to be loaded via this
mode unless they are explicitly allowlisted by the user via
`torch.serialization.add_safe_globals`. We recommend you start setting
`weights_only=True` for any use case where you don't have full control of the
loaded file. Please open an issue on GitHub for any issues related to this
experimental feature.
  wgts = torch.load(wgts_fname, map_location = lambda storage,loc: storage)
```

[20]:
```python
llm_learn.fit_one_cycle(4, 1e-2) # 0.01 | https://iconof.com/
 ↪1cycle-learning-rate-policy/
```

```
<IPython.core.display.HTML object>
```

[21]:
```python
llm_learn.save('4epoch')
# llm_learn = llm_learn.load('1epoch')
```

[21]:
```
Path('/Users/kristianbotnen/.fastai/data/subset/unsup/models/4epoch.pth')
```

[22]:
```python
llm_learn.unfreeze()
llm_learn.fit_one_cycle(10, 1e-3) # 0.001 | https://iconof.com/
 ↪1cycle-learning-rate-policy/
```

```
<IPython.core.display.HTML object>
```

[23]:
```python
llm_learn.save_encoder('10epoch_finetuned')
```

[30]:
```python
print(llm_learn.predict("The man is a good", 50, temperature=0.75))
```

```
<IPython.core.display.HTML object>

The man is a good guy and has a lot of passion for merit , love and love . He
has a great time and finds some good friends and family , but he has no great
experience in it . He has a lot of have to do with some sort of
```

[31]:
```python
the_best_review_starts_with = "I liked this movie because: "
n_words = 40
n_sentences = 2
preds = [llm_learn.predict(the_best_review_starts_with, n_words, temperature=0.
 ↪75)
         for _ in range(n_sentences)]
```

```
<IPython.core.display.HTML object>

<IPython.core.display.HTML object>
```

[32]:
```python
print(preds)
```

```
['i liked this movie because : " it \'s so hard to believe that this movie was
made by a filmmaker . It was a matter of fact , but because of the length , it
was not a movie or a movie .', "i liked this movie because : i have a great idea
of how this film could deal with family and family . It is a fascinating topic ,
because it has a lot of love and it 's not really quite an appropriate thing
for"]
```

0.3 Skip this part?

[33]:
```
dataloaders_classifier = TextDataLoaders.from_folder(subset_path, valid='test',␣
↪text_vocab=dataloaders_lm.vocab)
```

[34]:
```
learn_2pass = text_classifier_learner(dataloaders_classifier, AWD_LSTM,␣
↪drop_mult=0.5, metrics=accuracy)
```

```
/opt/miniconda3/envs/pythonki/lib/python3.12/site-
packages/fastai/text/learner.py:149: FutureWarning: You are using `torch.load`
with `weights_only=False` (the current default value), which uses the default
pickle module implicitly. It is possible to construct malicious pickle data
which will execute arbitrary code during unpickling (See
https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models for
more details). In a future release, the default value for `weights_only` will be
flipped to `True`. This limits the functions that could be executed during
unpickling. Arbitrary objects will no longer be allowed to be loaded via this
mode unless they are explicitly allowlisted by the user via
`torch.serialization.add_safe_globals`. We recommend you start setting
`weights_only=True` for any use case where you don't have full control of the
loaded file. Please open an issue on GitHub for any issues related to this
experimental feature.
  wgts = torch.load(wgts_fname, map_location = lambda storage,loc: storage)
```

[35]:
```
encoder_path = subset_path/'unsup/models'
learn_2pass = learn_2pass.load_encoder(encoder_path/'10epoch_finetuned')
```

```
/opt/miniconda3/envs/pythonki/lib/python3.12/site-
packages/fastai/text/learner.py:135: FutureWarning: You are using `torch.load`
with `weights_only=False` (the current default value), which uses the default
pickle module implicitly. It is possible to construct malicious pickle data
which will execute arbitrary code during unpickling (See
https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models for
more details). In a future release, the default value for `weights_only` will be
flipped to `True`. This limits the functions that could be executed during
unpickling. Arbitrary objects will no longer be allowed to be loaded via this
mode unless they are explicitly allowlisted by the user via
`torch.serialization.add_safe_globals`. We recommend you start setting
`weights_only=True` for any use case where you don't have full control of the
loaded file. Please open an issue on GitHub for any issues related to this
experimental feature.
```

```
  wgts = torch.load(join_path_file(file,self.path/self.model_dir, ext='.pth'),
map_location=device)
```

```
[36]: learn_2pass.fit_one_cycle(1, 2e-2) # 0.02 | https://iconof.com/
        ↪1cycle-learning-rate-policy/
```

```
<IPython.core.display.HTML object>
```

```
[ ]: #print(slice(1e-2/(2.6**4),1e-2))
     #print(slice(5e-3/(2.6**4),5e-3))
     #print(slice(1e-3/(2.6**4),1e-3))
```

```
[37]: learn_2pass.freeze_to(-2) # Last two layers
      learn_2pass.fit_one_cycle(1, slice(1e-2/(2.6**4),1e-2)) # epoch, lr group 0␣
        ↪(body), lr group 1 (head)
```

```
<IPython.core.display.HTML object>
```

```
[38]: learn_2pass.freeze_to(-3) # Last three layers
      learn_2pass.fit_one_cycle(1, slice(5e-3/(2.6**4),5e-3)) # epoch, lr group 0␣
        ↪(body), lr group 1 (head)
```

```
<IPython.core.display.HTML object>
```

```
[39]: learn_2pass.unfreeze() # All layers
      learn_2pass.fit_one_cycle(2, slice(1e-3/(2.6**4),1e-3)) # epoch, lr group 0␣
        ↪(body), lr group 1 (head)
```

```
<IPython.core.display.HTML object>
```

```
[40]: # Use our model by passing it a review.
      category,_,probs = learn_2pass.predict("I really liked that movie")

      print(f"This is a: {category}.")
      print(f"Probability it's a positive: {probs[1]:.4f}")

      category,_,probs = learn_2pass.predict("I did not like that movie, it was␣
        ↪awful")

      print(f"This is a: {category}.")
      print(f"Probability it's a positive: {probs[1]:.4f}")
```

```
<IPython.core.display.HTML object>

This is a: pos.
Probability it's a positive: 0.9358

<IPython.core.display.HTML object>

This is a: neg.
Probability it's a positive: 0.1404
```

```
[ ]:
```