Quick Start: Training an IMDb sentiment model with *ULMFiT*

Let's start with a quick end-to-end example of training a model. We'll train a sentiment classifier on a sample of the popular IMDb data, showing 4 steps:

1. Reading and viewing the IMDb data
2. Getting your data ready for modeling
3. Fine-tuning a language model
4. Building a classifier

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

Contrary to images in Computer Vision, text can't directly be transformed into numbers to be fed into a model. The first thing we need to do is to preprocess our data so that we change the raw texts to lists of words, or tokens (a step that is called tokenization) then transform these tokens into numbers (a step that is called numericalization). These numbers are then passed to embedding layers that will convert them in arrays of floats before passing them through a model.

Steps:

1. Get your data preprocessed and ready to use,
2. Create a language model with pretrained weights that you can fine-tune to your dataset,
3. Create other models such as classifiers on top of the encoder of the language model.

To show examples, we have provided a small sample of the [IMDB dataset (https://www.imdb.com/interfaces/)](https://www.imdb.com/interfaces/) which contains 1,000 reviews of movies with labels (positive or negative).

In [3]:
```
path = untar_data(URLs.IMDB_SAMPLE)
```

In [4]:
```
df = pd.read_csv(path/'texts.csv')
df.head()
```

Out[4]:

	label	text	is_valid
0	negative	Un-bleeping-believable! Meg Ryan doesn't even ...	False
1	positive	This is a extremely well-made film. The acting...	False
2	negative	Every once in a long while a movie will come a...	False
3	positive	Name just says it all. I watched this movie wi...	False
4	negative	This movie succeeds at being one of the most u...	False

In [5]:
```
data_lm = TextLMDataBunch.from_csv(path, 'texts.csv')
data_clas = TextClasDataBunch.from_csv(path, 'texts.csv', vocab=data_lm.train_
ds.vocab, bs=32)
```

In [6]:
```
data_lm.save('data_lm_export.pkl')
data_clas.save('data_clas_export.pkl')
```

In [8]:
```
bs=192
```

In [9]:
```
data_lm = load_data(path, 'data_lm_export.pkl', bs=bs)
data_clas = load_data(path, 'data_clas_export.pkl', bs=bs)
```

Note that you can load the data with different `DataBunch` (/basic_data.html#DataBunch) parameters (batch size, `bptt` ,...)

Fine-tuning a language model

We can use the `data_lm` object we created earlier to fine-tune a pretrained language model. fast.ai (http://www.fast.ai/) has an English model with an AWD-LSTM architecture available that we can download. We can create a learner object that will directly create a model, download the pretrained weights and be ready for fine-tuning.

In [10]:
```
torch.cuda.set_device(1)
```

In [25]:
```
learn = language_model_learner(data_lm, AWD_LSTM, drop_mult=0.5)
learn.fit_one_cycle(1, 1e-2)
```

epoch	train_loss	valid_loss	accuracy	time
0	4.553834	4.097589	0.269449	00:04

In [48]:
```
language_model_learner??
```

In [45]:
```
learn.load_pretrained??
```

In [47]:
```
convert_weights??
```

You can use Visual Studio Code (https://code.visualstudio.com/) (vscode - open source editor that comes with recent versions of Anaconda, or can be installed separately), or most editors and IDEs, to browse code. vscode things to know:

- Command palette (`Ctrl-shift-p`)
- Go to symbol (`Ctrl-t`)
- Find references (`Shift-F12`)
- Go to definition (`F12`)
- Go back (`alt-left`)
- View documentation
- Hide sidebar (`Ctrl-b`)
- Zen mode (`Ctrl-k,z`)

Like a computer vision model, we can then unfreeze the model and fine-tune it.

In [26]:
```
learn.unfreeze()
learn.fit_one_cycle(3, slice(1e-4,1e-2))
```

epoch	train_loss	valid_loss	accuracy	time
0	4.296110	3.958033	0.281652	00:05
1	4.105352	3.877306	0.284554	00:04
2	3.926521	3.866480	0.286250	00:05

To evaluate your language model, you can run the `Learner.predict` (/basic_train.html#Learner.predict) method and specify the number of words you want it to guess.

In [27]:
```
learn.predict("This is a review about", n_words=10)
```

Out[27]:
```
"This is a review about the award 's effect on the Cuban population ."
```

It doesn't make much sense (we have a tiny vocabulary here and didn't train much on it) but note that it respects basic grammar (which comes from the pretrained model).

Finally we save the encoder to be able to use it for classification in the next section.

In [28]:
```
learn.save('ft')
learn.save_encoder('ft_enc')
```

Building a classifier

In [38]:
```
learn = text_classifier_learner(data_clas, AWD_LSTM, drop_mult=0.5).to_fp16()
learn.load_encoder('ft_enc')
```

In [39]:
```
data_clas.show_batch()
```

text	target
xxbos xxmaj raising xxmaj victor xxmaj vargas : a xxmaj review \n \n xxmaj you know , xxmaj raising xxmaj victor xxmaj vargas is like sticking your hands into a big , steaming bowl of xxunk . xxmaj it 's warm and gooey , but you 're not sure if it feels right . xxmaj try as i might , no matter how warm and gooey xxmaj raising xxmaj	negative
xxbos xxup the xxup shop xxup around xxup the xxup xxunk is one of the xxunk and most feel - good romantic comedies ever made . xxmaj there 's just no getting around that , and it 's hard to actually put one 's feeling for this film into words . xxmaj it 's not one of those films that tries too hard , nor does it come up with	positive
xxbos xxmaj now that xxmaj xxunk) has finished its relatively short xxmaj australian cinema run (extremely limited xxunk screen in xxmaj xxunk , after xxunk) , i can xxunk join both xxunk of " xxmaj at xxmaj the xxmaj movies " in taking xxmaj steven xxmaj xxunk to task . \n \n xxmaj it 's usually satisfying to watch a film director change his style /	negative
xxbos xxmaj this film sat on my xxmaj xxunk for weeks before i watched it . i xxunk a self - indulgent xxunk flick about relationships gone bad . i was wrong ; this was an xxunk xxunk into the xxunk - up xxunk of xxmaj new xxmaj xxunk . \n \n xxmaj the format is the same as xxmaj max xxmaj xxunk ' " xxmaj la xxmaj xxunk	positive
xxbos xxmaj many neglect that this is n't just a classic due to the fact that it 's the first xxup 3d game , or even the first xxunk - up . xxmaj it 's also one of the first stealth games , one of the xxunk definitely the first) truly claustrophobic games , and just a pretty well - rounded gaming experience in general . xxmaj with graphics	positive

In [40]:
```
learn.fit_one_cycle(1, 1e-2)
```

epoch	train_loss	valid_loss	accuracy	time
0	0.663383	0.679151	0.562189	00:04

Again, we can unfreeze the model and fine-tune it.

In [41]:
```
learn.unfreeze()
learn.fit_one_cycle(3, slice(1e-4, 1e-2))
```

epoch	train_loss	valid_loss	accuracy	time
0	0.547386	0.668068	0.557214	00:08
1	0.502200	0.591302	0.681592	00:08
2	0.463140	0.556154	0.746269	00:09

Again, we can predict on a raw text by using the `Learner.predict` (/basic_train.html#Learner.predict) method.

In [42]:
```
learn.predict("This was a great movie!")
```

Out[42]:
```
(Category positive, tensor(1), tensor([0.3933, 0.6067]))
```

In [ ]: