IMDB

In [1]:
```
%reload_ext autoreload
%autoreload 2
%matplotlib inline
```

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

Preparing the data

First let's download the dataset we are going to study. The [dataset (http://ai.stanford.edu/~amaas/data/sentiment/)](http://ai.stanford.edu/~amaas/data/sentiment/) has been curated by Andrew Maas et al. and contains a total of 100,000 reviews on IMDB. 25,000 of them are labelled as positive and negative for training, another 25,000 are labelled for testing (in both cases they are highly polarized). The remaning 50,000 is an additional unlabelled data (but we will find a use for it nonetheless).

We'll begin with a sample we've prepared for you, so that things run quickly before going over the full dataset.

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

Out[3]:
```
[PosixPath('/home/ml1/.fastai/data/imdb_sample/data_save.pkl'),
 PosixPath('/home/ml1/.fastai/data/imdb_sample/texts.csv')]
```

It only contains one csv file, let's have a look at it.

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]:
```
df['text'][1]
```

Out[5]:
```
'This is a extremely well-made film. The acting, script and camera-work are all
first-rate. The music is good, too, though it is mostly early in the film, when
things are still relatively cheery. There are no really superstars in the cast,
though several faces will be familiar. The entire cast does an excellent job wi
th the script.<br /><br />But it is hard to watch, because there is no good end
to a situation like the one presented. It is now fashionable to blame the Briti
sh for setting Hindus and Muslims against each other, and then cruelly separati
ng them into two countries. There is some merit in this view, but it\'s also tr
ue that no one forced Hindus and Muslims in the region to mistreat each other a
s they did around the time of partition. It seems more likely that the British
simply saw the tensions between the religions and were clever enough to exploit
them to their own ends.<br /><br />The result is that there is much cruelty and
inhumanity in the situation and this is very unpleasant to remember and to see
on the screen. But it is never painted as a black-and-white case. There is base
ness and nobility on both sides, and also the hope for change in the younger ge
neration.<br /><br />There is redemption of a sort, in the end, when Puro has t
o make a hard choice between a man who has ruined her life, but also truly love
d her, and her family which has disowned her, then later come looking for her.
But by that point, she has no option that is without great pain for her.<br /><
br />This film carries the message that both Muslims and Hindus have their grav
e faults, and also that both can be dignified and caring people. The reality of
partition makes that realisation all the more wrenching, since there can never
be real reconciliation across the India/Pakistan border. In that sense, it is s
imilar to "Mr & Mrs Iyer".<br /><br />In the end, we were glad to have seen the
film, even though the resolution was heartbreaking. If the UK and US could deal
with their own histories of racism with this kind of frankness, they would cert
ainly be better off.'
```

It contains one line per review, with the label ('negative' or 'positive'), the text and a flag to determine if it should be part of the validation set or the training set. If we ignore this flag, we can create a DataBunch containing this data in one line of code:

In [6]:
```
data_lm = TextDataBunch.from_csv(path, 'texts.csv')
```

By executing this line a process was launched that took a bit of time. Let's dig a bit into it. Images could be fed (almost) directly into a model because they're just a big array of pixel values that are floats between 0 and 1. A text is composed of words, and we can't apply mathematical functions to them directly. We first have to convert them to numbers. This is done in two differents steps: tokenization and numericalization. A `TextDataBunch` does all of that behind the scenes for you.

Before we delve into the explanations, let's take the time to save the things that were calculated.

In [7]:
```
data_lm.save()
```

Next time we launch this notebook, we can skip the cell above that took a bit of time (and that will take a lot more when you get to the full dataset) and load those results like this:

In [8]:
```
data = load_data(path)
```

Tokenization

The first step of processing we make the texts go through is to split the raw sentences into words, or more exactly tokens. The easiest way to do this would be to split the string on spaces, but we can be smarter:

- we need to take care of punctuation
- some words are contractions of two different words, like isn't or don't
- we may need to clean some parts of our texts, if there's HTML code for instance

To see what the tokenizer had done behind the scenes, let's have a look at a few texts in a batch.

In [9]:
```
data = TextClasDataBunch.from_csv(path, 'texts.csv')
data.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 xxmaj this film sat on my xxmaj tivo for weeks before i watched it . i dreaded a self - indulgent xxunk flick about relationships gone bad . i was wrong ; this was an xxunk xxunk into the screwed - up xxunk of xxmaj new xxmaj xxunk . \n \n xxmaj the format is the same as xxmaj max xxmaj xxunk ' " xxmaj la xxmaj ronde	positive
xxbos i really wanted to love this show . i truly , honestly did . \n \n xxmaj for the first time , gay viewers get their own version of the " xxmaj the xxmaj bachelor " . xxmaj with the help of his obligatory " hag " xxmaj xxunk , xxmaj james , a good looking , well - to - do thirty - something has the chance	negative
xxbos \n \n i 'm sure things did n't exactly go the same way in the real life of xxmaj homer xxmaj hickam as they did in the film adaptation of his book , xxmaj rocket xxmaj boys , but the movie " xxmaj october xxmaj sky " (an xxunk of the book 's title) is good enough to stand alone . i have not read xxmaj	positive
xxbos xxmaj to review this movie , i without any doubt would have to quote that memorable scene in xxmaj tarantino 's " xxmaj pulp xxmaj fiction " (xxunk) when xxmaj jules and xxmaj vincent are talking about xxmaj mia xxmaj wallace and what she does for a living . xxmaj jules tells xxmaj vincent that the " xxmaj only thing she did worthwhile was pilot " .	negative

The texts are truncated at 100 tokens for more readability. We can see that it did more than just split on space and punctuation symbols:

- the "'s" are grouped together in one token
- the contractions are separated like this: "did", "n't"
- content has been cleaned for any HTML symbol and lower cased
- there are several special tokens (all those that begin by xx), to replace unknown tokens (see below) or to introduce different text fields (here we only have one).

Numericalization

Once we have extracted tokens from our texts, we convert to integers by creating a list of all the words used. We only keep the ones that appear at least twice with a maximum vocabulary size of

60,000 (by default) and replace the ones that don't make the cut by the unknown token `UNK`.

The correspondance from ids to tokens is stored in the `vocab` attribute of our datasets, in a dictionary called `itos` (for int to string).

In [10]:
```
data.vocab.itos[:10]
```

Out[10]:
```
['xxunk',
 'xxpad',
 'xxbos',
 'xxeos',
 'xxfld',
 'xxmaj',
 'xxup',
 'xxrep',
 'xxwrep',
 'the']
```

And if we look at what a what's in our datasets, we'll see the tokenized text as a representation:

In [11]:
```
data.train_ds[0][0]
```

Out[11]:
```
Text xxbos xxmaj back in the forties , when movies touched on matters not yet x
xunk in " polite " society , they resorted to codes which supposedly xxunk over
the heads of most of the audience while xxunk those in the know to just what wa
s up . xxmaj probably no film of the decade was so xxunk with xxunk as the oddl
y obscure xxmaj desert xxmaj fury , set in a small gambling xxunk called xxmaj
xxunk somewhere in the xxmaj california desert . xxmaj xxunk of the xxmaj purpl
e xxmaj xxunk xxunk and casino is the astonishing xxmaj mary xxmaj astor , in s
lacks and sporting a cigarette xxunk ; into town drives her handful - of - a -
daughter , xxmaj xxunk xxmaj scott , looking , in xxmaj technicolor , like xxun
k bucks . xxmaj but listen to the dialogue between them , which suggests an old
er xxmaj lesbian and her young , xxunk companion ( one can only wonder if xxup
xxunk xxmaj xxunk ' original script made this relationship explicit ) . xxmaj e
ven more blatant are xxmaj john xxmaj hodiak as a gangster and xxmaj xxunk xxma
j xxunk as his xxunk jealous xxunk . xxmaj add xxmaj burt xxmaj lancaster as th
e town sheriff , stir , and sit back . xxmaj both xxmaj lancaster and ( surpris
ingly ) xxmaj hodiak fall for xxmaj scott . xxmaj it seems , however , that xxm
aj hodiak not only has a past with xxmaj astor , but had a wife who died under
suspicious circumstances . xxmaj the desert sun xxunk these ingredients up to a
hard boil , with face - xxunk aplenty and xxunk exchanges . xxmaj do n't pass u
p this xxunk melodrama , chock full of xxunk exotic xxunk , if it comes your wa
y ; it 's a remarkable movie .
```

But the underlying data is all numbers

In [12]:
```
data.train_ds[0][0].data[:10]
```

Out[12]:
```
array([   2,    5,  165,   18,    9, 3216,   10,   71,  122, 4648])
```

With the data block API

We can use the data block API with NLP and have a lot more flexibility than what the default factory methods offer. In the previous example for instance, the data was randomly split between train and validation instead of reading the third column of the csv.

With the data block API though, we have to manually call the tokenize and numericalize steps. This allows more flexibility, and if you're not using the defaults from fastai, the variaous arguments to pass will appear in the step they're revelant, so it'll be more readable.

In [13]:
```
data = (TextList.from_csv(path, 'texts.csv', cols='text')
                .split_from_df(col=2)
                .label_from_df(cols=0)
                .databunch())
```

Language model

Note that language models can use a lot of GPU, so you may need to decrease batchsize here.

In [14]:
```
bs=48
```

Now let's grab the full dataset for what follows.

In [15]:
```
path = untar_data(URLs.IMDB)
path.ls()
```

Out[15]:
```
[PosixPath('/home/ml1/.fastai/data/imdb/models'),
 PosixPath('/home/ml1/.fastai/data/imdb/test'),
 PosixPath('/home/ml1/.fastai/data/imdb/tmp_clas'),
 PosixPath('/home/ml1/.fastai/data/imdb/tmp_lm'),
 PosixPath('/home/ml1/.fastai/data/imdb/imdb.vocab'),
 PosixPath('/home/ml1/.fastai/data/imdb/README'),
 PosixPath('/home/ml1/.fastai/data/imdb/data_lm.pkl'),
 PosixPath('/home/ml1/.fastai/data/imdb/train'),
 PosixPath('/home/ml1/.fastai/data/imdb/data_clas.pkl'),
 PosixPath('/home/ml1/.fastai/data/imdb/unsup')]
```

In [16]:
```
(path/'train').ls()
```

Out[16]:
```
[PosixPath('/home/ml1/.fastai/data/imdb/train/labeledBow.feat'),
 PosixPath('/home/ml1/.fastai/data/imdb/train/neg'),
 PosixPath('/home/ml1/.fastai/data/imdb/train/pos'),
 PosixPath('/home/ml1/.fastai/data/imdb/train/unsupBow.feat')]
```

The reviews are in a training and test set following an imagenet structure. The only difference is that there is an `unsup` folder on top of `train` and `test` that contains the unlabelled data.

We're not going to train a model that classifies the reviews from scratch. Like in computer vision, we'll use a model pretrained on a bigger dataset (a cleaned subset of wikipedia called [wikitext-103](https://einstein.ai/research/blog/the-wikitext-long-term-dependency-language-modeling-dataset)).

That model has been trained to guess what the next word, its input being all the previous words. It has a recurrent structure and a hidden state that is updated each time it sees a new word. This hidden state thus contains information about the sentence up to that point.

We are going to use that 'knowledge' of the English language to build our classifier, but first, like for computer vision, we need to fine-tune the pretrained model to our particular dataset. Because the English of the reviews left by people on IMDB isn't the same as the English of wikipedia, we'll need to adjust the parameters of our model by a little bit. Plus there might be some words that would be extremely common in the reviews dataset but would be barely present in wikipedia, and therefore might not be part of the vocabulary the model was trained on.

This is where the unlabelled data is going to be useful to us, as we can use it to fine-tune our model. Let's create our data object with the data block API (next line takes a few minutes).

In [17]:
```
data_lm = (TextList.from_folder(path)
           #Inputs: all the text files in path
            .filter_by_folder(include=['train', 'test', 'unsup'])
           #We may have other temp folders that contain text files so we only kee
            .random_split_by_pct(0.1)
           #We randomly split and keep 10% (10,000 reviews) for validation
            .label_for_lm()
           #We want to do a language model so we label accordingly
            .databunch(bs=bs))
data_lm.save('data_lm.pkl')
```

We have to use a special kind of `TextDataBunch` for the language model, that ignores the labels (that's why we put 0 everywhere), will shuffle the texts at each epoch before concatenating them all together (only for training, we don't shuffle for the validation set) and will send batches that read that text in order with targets that are the next word in the sentence.

The line before being a bit long, we want to load quickly the final ids by using the following cell.

In [18]:
```
data_lm = load_data(path, 'data_lm.pkl', bs=bs)
```

In [19]:
```
data_lm.show_batch()
```

idx	text
0	grace " , which he also directed , and the results are as awkward and unbecoming as that title . xxmaj story of a famous singer returning to his hometown in the sticks , opening up old family wounds , boasts a screenplay by xxmaj larry mcmurtry , but the meandering film goes nowhere slowly . xxmaj the supporting cast is decent , including xxmaj kay xxmaj lenz (
1	, " xxmaj two xxmaj weddings and a xxmaj funeral " , during her second wedding speech is devoid of emotion (eg . hearts in our hearts) . \n \n i liked all of the other actors in their respective parts and they were all believable . xxmaj with improved writing and a lead change xxmaj tru xxmaj calling might have made it . xxbos xxmaj first
2	credibility , but his saying of lines like " xxmaj royal xxmaj xxunk , xxmaj north of the ... " just are n't effective in establishing him as this worldly and suave rogue . xxmaj savalas does n't do a bad job , but his characterization and behavior is more fitting of a mob gangster . xxmaj the best portrayed characters of the movie are those of xxmaj tracy
3	, she has stupid frizzy blonde hair and a xxunk red bunny like face . xxmaj she acts so innocent . xxmaj next we have the second child - the xxmaj geek - who thinks he 's so cool , with his long range shooting and his use of a silencer (a coat over the gun) and most of all his evil bratty smile . xxmaj the
4	unlikeable , and if you ca n't identify with at least one character , there is n't much to get excited about . xxmaj all in all , this is a classic example of trying to do too much with too little . xxbos xxmaj and again , i find myself in the minority . \n \n i did n't like this one . xxmaj this is the xxup

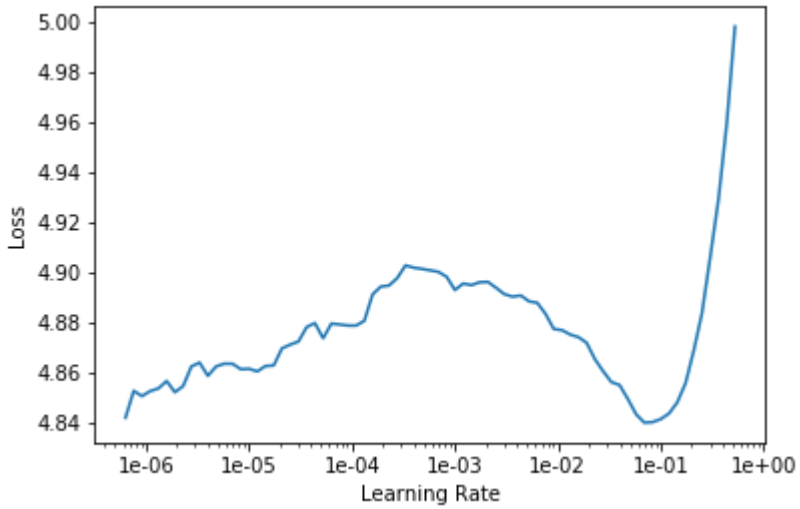
We can then put this in a learner object very easily with a model loaded with the pretrained weights. They'll be downloaded the first time you'll execute the following line and stored in `~/.fastai/models/` (or elsewhere if you specified different paths in your config file).

In [20]:
```
learn = language_model_learner(data_lm, AWD_LSTM, drop_mult=0.3)
```

In [21]:
```
learn.lr_find()
```

```
LR Finder is complete, type {learner_name}.recorder.plot() to see the graph.
```

In [22]:
```
learn.recorder.plot(skip_end=15)
```



In [23]:
```
learn.fit_one_cycle(1, 1e-2, moms=(0.8,0.7))
```

Total time: 21:34

epoch	train_loss	valid_loss	accuracy	time
0	4.200193	4.055476	0.293179	21:34

In [24]:
```
learn.save('fit_head')
```

In [25]:
```
learn.load('fit_head');
```

To complete the fine-tuning, we can then unfeeze and launch a new training.

In [26]:
```
learn.unfreeze()
```

In [27]:
```
learn.fit_one_cycle(10, 1e-3, moms=(0.8,0.7))
```

90.00% [9/10 3:39:51<24:25]

epoch	train_loss	valid_loss	accuracy	time
0	3.916237	3.879890	0.310272	24:27
1	3.878546	3.810675	0.319859	24:28
2	3.810352	3.765642	0.325609	24:28
3	3.765185	3.723465	0.330329	24:24
4	3.697526	3.692837	0.333842	24:24
5	3.631543	3.670253	0.336328	24:26
6	3.569497	3.650672	0.338566	24:26
7	3.538299	3.637485	0.340289	24:22
8	3.492487	3.632395	0.341009	24:23

58.05% [4674/8051 13:32<09:46 3.4684]

In [28]:
```
learn.save('fine_tuned')
```

How good is our model? Well let's try to see what it predicts after a few given words.

In [29]:
```
learn.load('fine_tuned');
```

In [30]:
```
TEXT = "I liked this movie because"
N_WORDS = 40
N_SENTENCES = 2
```

In [31]:
```
print("\n".join(learn.predict(TEXT, N_WORDS, temperature=0.75) for _ in range(N_
```

```
I liked this movie because i had recommended it to my friends and they said tha
t it was n't really about the industry but it was about the whole family . It w
as a very touching movie , i loved how there was drama
I liked this movie because of Lance Henriksen , who was great in the role of th
e sheriff . i especially liked his character , Jay , the man who had to make hi
s way through the town he was in .
```

We not only have to save the model, but also it's encoder, the part that's responsible for creating and updating the hidden state. For the next part, we don't care about the part that tries to guess the next word.

In [32]:
```
learn.save_encoder('fine_tuned_enc')
```

Classifier

Now, we'll create a new data object that only grabs the labelled data and keeps those labels. Again, this line takes a bit of time.

In [33]:
```
path = untar_data(URLs.IMDB)
```

In [34]:
```
data_clas = (TextList.from_folder(path, vocab=data_lm.vocab)
             #grab all the text files in path
             .split_by_folder(valid='test')
             #split by train and valid folder (that only keeps 'train' and 'test
             .label_from_folder(classes=['neg', 'pos'])
             #label them all with their folders
             .databunch(bs=bs))

data_clas.save('data_clas.pkl')
```

In [35]:
```
data_clas = load_data(path, 'data_clas.pkl', bs=bs)
```

In [36]:
```
data_clas.show_batch()
```

text	target
xxbos xxmaj match 1 : xxmaj tag xxmaj team xxmaj table xxmaj match xxmaj bubba xxmaj ray and xxmaj spike xxmaj dudley vs xxmaj eddie xxmaj guerrero and xxmaj chris xxmaj benoit xxmaj bubba xxmaj ray and xxmaj spike xxmaj dudley started things off with a xxmaj tag xxmaj team xxmaj table xxmaj match against xxmaj eddie xxmaj guerrero and xxmaj chris xxmaj benoit . xxmaj according to the rules	pos
xxbos xxmaj titanic directed by xxmaj james xxmaj cameron presents a fictional love story on the historical setting of the xxmaj titanic . xxmaj the plot is simple , xxunk , or not for those who love plots that twist and turn and keep you in suspense . xxmaj the end of the movie can be figured out within minutes of the start of the film , but the love	pos
xxbos xxmaj god ! xxmaj zorro has been the the subject of about as many movies as xxmaj tarzan , and probably had about as many actors in the title role . \n \n xxmaj this xxmaj serial is one of my own personal favourites , and as previously stated , it is one of the xxmaj top 5 xxmaj sound xxmaj serials . xxmaj oddly enough , this	pos
xxbos xxmaj waitress : xxmaj honey , here 's them eggs you ordered . xxmaj honey , like bee , get it ? xxmaj that 's called pointless foreshadowing . \n \n xxmaj edward xxmaj basket : xxmaj huh ? (xxmaj on the road) xxmaj basket : xxmaj here 's your doll back , little girl . xxmaj you really should n't be so careless with your	neg
xxbos xxmaj on xxmaj sunday xxmaj july 27 , 1997 , the first episode of a new science fiction series called " xxmaj stargate xxup sg-1 " was broadcast on xxmaj showtime . a spin - off of and sequel to the 1994 film " xxmaj stargate " starring xxmaj kurt xxmaj russell and xxmaj james xxmaj spader , the series begins approximately one year after the events portrayed in	pos

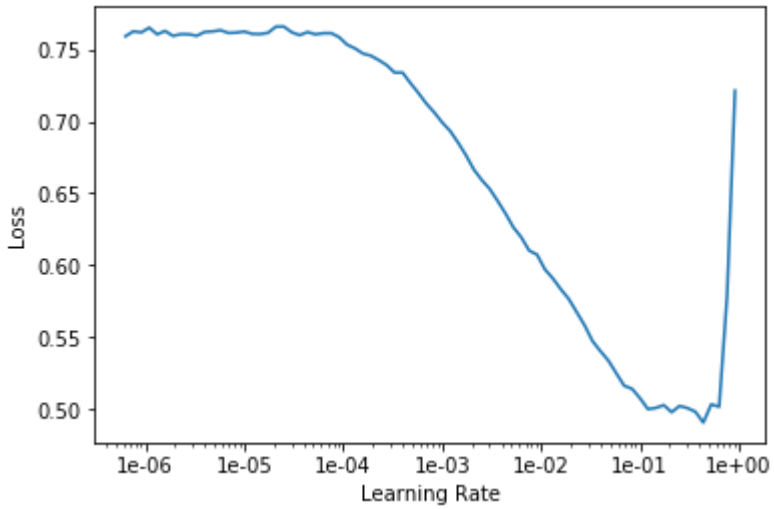
We can then create a model to classify those reviews and load the encoder we saved before.

In [37]:
```
learn = text_classifier_learner(data_clas, AWD_LSTM, drop_mult=0.5)
learn.load_encoder('fine_tuned_enc')
```

In [38]:
```
learn.lr_find()
```

```
LR Finder is complete, type {learner_name}.recorder.plot() to see the graph.
```

In [39]:
```
learn.recorder.plot()
```



In [40]:
```
learn.fit_one_cycle(1, 2e-2, moms=(0.8,0.7))
```

Total time: 03:35

epoch	train_loss	valid_loss	accuracy	time
0	0.283969	0.198556	0.922120	03:35

In [41]:
```
learn.save('first')
```

In [42]:
```
learn.load('first');
```

In [43]:
```
learn.freeze_to(-2)
learn.fit_one_cycle(1, slice(1e-2/(2.6**4),1e-2), moms=(0.8,0.7))
```

Total time: 03:44

epoch	train_loss	valid_loss	accuracy	time
0	0.246850	0.170124	0.936120	03:44

In [44]:
```
learn.save('second')
```

In [45]:
```
learn.load('second');
```

In [46]:
```
learn.freeze_to(-3)
learn.fit_one_cycle(1, slice(5e-3/(2.6**4),5e-3), moms=(0.8,0.7))
```

0.00% [0/1 00:00<00:00]

epoch	train_loss	valid_loss	accuracy	time

Interrupted

```
---------------------------------------------------------------------------
RuntimeError                              Traceback (most recent call last)
<ipython-input-46-644a31004af3> in <module>
      1 learn.freeze_to(-3)
----> 2 learn.fit_one_cycle(1, slice(5e-3/(2.6**4),5e-3), moms=(0.8,0.7))

~/anaconda3/envs/fastai/lib/python3.6/site-packages/fastai/train.py in fit_one_
cycle(learn, cyc_len, max_lr, moms, div_factor, pct_start, final_div, wd, callb
acks, tot_epochs, start_epoch)
     20     callbacks.append(OneCycleScheduler(learn, max_lr, moms=moms, div_fa
ctor=div_factor, pct_start=pct_start,
     21                                        final_div=final_div, tot_epochs=
tot_epochs, start_epoch=start_epoch))
---> 22     learn.fit(cyc_len, max_lr, wd=wd, callbacks=callbacks)
     23
     24 def lr_find(learn:Learner, start_lr:Floats=1e-7, end_lr:Floats=10, num_
it:int=100, stop_div:bool=True, wd:float=None):

~/anaconda3/envs/fastai/lib/python3.6/site-packages/fastai/basic_train.py in fi
t(self, epochs, lr, wd, callbacks)
    194         callbacks = [cb(self) for cb in self.callback_fns] + listify(ca
llbacks)
    195         if defaults.extra_callbacks is not None: callbacks += defaults.
extra_callbacks
--> 196         fit(epochs, self, metrics=self.metrics, callbacks=self.callback
s+callbacks)
    197
    198     def create_opt(self, lr:Floats, wd:Floats=0.)->None:

~/anaconda3/envs/fastai/lib/python3.6/site-packages/fastai/basic_train.py in fi
t(epochs, learn, callbacks, metrics)
     98             for xb,yb in progress_bar(learn.data.train_dl, parent=pbar)
:
     99                 xb, yb = cb_handler.on_batch_begin(xb, yb)
--> 100                 loss = loss_batch(learn.model, xb, yb, learn.loss_func,
learn.opt, cb_handler)
    101                 if cb_handler.on_batch_end(loss): break
    102

~/anaconda3/envs/fastai/lib/python3.6/site-packages/fastai/basic_train.py in lo
ss_batch(model, xb, yb, loss_func, opt, cb_handler)
     23     if not is_listy(xb): xb = [xb]
     24     if not is_listy(yb): yb = [yb]
---> 25     out = model(*xb)
     26     out = cb_handler.on_loss_begin(out)
     27
```

```
~/anaconda3/envs/fastai/lib/python3.6/site-packages/torch/nn/modules/module.py
 in __call__(self, *input, **kwargs)
    487             result = self._slow_forward(*input, **kwargs)
    488         else:
--> 489             result = self.forward(*input, **kwargs)
    490         for hook in self._forward_hooks.values():
    491             hook_result = hook(self, input, result)

~/anaconda3/envs/fastai/lib/python3.6/site-packages/torch/nn/modules/container.
py in forward(self, input)
     90     def forward(self, input):
     91         for module in self._modules.values():
---> 92             input = module(input)
     93         return input
     94

~/anaconda3/envs/fastai/lib/python3.6/site-packages/torch/nn/modules/module.py
 in __call__(self, *input, **kwargs)
    487             result = self._slow_forward(*input, **kwargs)
    488         else:
--> 489             result = self.forward(*input, **kwargs)
    490         for hook in self._forward_hooks.values():
    491             hook_result = hook(self, input, result)

~/anaconda3/envs/fastai/lib/python3.6/site-packages/fastai/text/learner.py in f
orward(self, input)
    233         raw_outputs,outputs,mask = input
    234         output = outputs[-1]
--> 235         avg_pool = output.masked_fill(mask[:,:,None], 0).mean(dim=1)
    236         avg_pool *= output.size(1) / (output.size(1)-mask.float().sum(d
im=1))[:,None]
    237         max_pool = output.masked_fill(mask[:,:,None], -float('inf')).ma
x(dim=1)[0]

~/anaconda3/envs/fastai/lib/python3.6/site-packages/torch/tensor.py in masked_f
ill(self, mask, value)
    335         r"""Out-of-place version of :meth:`torch.Tensor.masked_fill_`
    336         """
--> 337         return self.clone().masked_fill_(mask, value)
    338
    339     def unique(self, sorted=False, return_inverse=False, dim=None):

RuntimeError: CUDA out of memory. Tried to allocate 100.88 MiB (GPU 0; 7.77 GiB
total capacity; 6.40 GiB already allocated; 10.50 MiB free; 610.63 MiB cached)
```

In [ ]:
```
learn.save('third')
```

In [ ]:
```
learn.load('third');
```

In [ ]:
```
learn.unfreeze()
learn.fit_one_cycle(2, slice(1e-3/(2.6**4),1e-3), moms=(0.8,0.7))
```

In [ ]:
```
learn.predict("I really loved that movie, it was awesome!")
```

In [ ]:
```

```

In [ ]:
```

```