In [1]:
```
from fastai.text import *
```

In [2]:
```
path = Config().data_path()
```

In [3]:
```
# ! wget https://s3.amazonaws.com/fast-ai-nlp/giga-fren.tgz -P {path}
```

In [4]:
```
# ! tar xf {path}/giga-fren.tgz -C {path}
```

In [3]:
```
path = Config().data_path()/'giga-fren'
path.ls()
```

Out[3]:
```
[PosixPath('/home/racheltho/.fastai/data/giga-fren/models'),
 PosixPath('/home/racheltho/.fastai/data/giga-fren/giga-fren.release2.fixed.f
r'),
 PosixPath('/home/racheltho/.fastai/data/giga-fren/cc.en.300.bin'),
 PosixPath('/home/racheltho/.fastai/data/giga-fren/data_save.pkl'),
 PosixPath('/home/racheltho/.fastai/data/giga-fren/giga-fren.release2.fixed.e
n'),
 PosixPath('/home/racheltho/.fastai/data/giga-fren/cc.fr.300.bin'),
 PosixPath('/home/racheltho/.fastai/data/giga-fren/questions_easy.csv')]
```

In [6]:
```
# with open(path/'giga-fren.release2.fixed.fr') as f: fr = f.read().split('\n')
```

In [7]:
```
# with open(path/'giga-fren.release2.fixed.en') as f: en = f.read().split('\n')
```

We will use regex to pick out questions by finding the strings in the English dataset that start with "Wh" and end with a question mark. You only need to run these lines once:

In [8]:
```
# re_eq = re.compile('^(Wh[^?.!]+\?)')
# re_fq = re.compile('^([^?.!]+\?)')
# en_fname = path/'giga-fren.release2.fixed.en'
# fr_fname = path/'giga-fren.release2.fixed.fr'
```

In [9]:
```
# lines = ((re_eq.search(eq), re_fq.search(fq))
#          for eq, fq in zip(open(en_fname, encoding='utf-8'), open(fr_fname, encc
# qs = [(e.group(), f.group()) for e,f in lines if e and f]
```

In [10]:
```
# qs = [(q1,q2) for q1,q2 in qs]
# df = pd.DataFrame({'fr': [q[1] for q in qs], 'en': [q[0] for q in qs]}, columns
# df.to_csv(path/'questions_easy.csv', index=False)
```

In [11]:

```
path.ls()
```

Out[11]:

```
[PosixPath('/home/racheltho/.fastai/data/giga-fren/models'),
 PosixPath('/home/racheltho/.fastai/data/giga-fren/giga-fren.release2.fixed.f
r'),
 PosixPath('/home/racheltho/.fastai/data/giga-fren/cc.en.300.bin'),
 PosixPath('/home/racheltho/.fastai/data/giga-fren/data_save.pkl'),
 PosixPath('/home/racheltho/.fastai/data/giga-fren/giga-fren.release2.fixed.e
n'),
 PosixPath('/home/racheltho/.fastai/data/giga-fren/cc.fr.300.bin'),
 PosixPath('/home/racheltho/.fastai/data/giga-fren/questions_easy.csv')]
```

Load our data into a DataBunch

Our questions look like this now:

In [4]:

```
df = pd.read_csv(path/'questions_easy.csv')
df.head()
```

Out[4]:

	en	fr
0	What is light ?	Qu’est-ce que la lumière?
1	Who are we?	Où sommes-nous?
2	Where did we come from?	D'où venons-nous?
3	What would we do without it?	Que ferions-nous sans elle ?
4	What is the absolute location (latitude and lo...	Quelle sont les coordonnées (latitude et longi...

To make it simple, we lowercase everything.

In [5]:

```
df['en'] = df['en'].apply(lambda x:x.lower())
df['fr'] = df['fr'].apply(lambda x:x.lower())
```

The first thing is that we will need to collate inputs and targets in a batch: they have different lengths so we need to add padding to make the sequence length the same;

In [7]:

```
def seq2seq_collate(samples, pad_idx=1, pad_first=True, backwards=False):
    "Function that collect samples and adds padding. Flips token order if needed"
    samples = to_data(samples)
    max_len_x,max_len_y = max([len(s[0]) for s in samples]),max([len(s[1]) for s
    res_x = torch.zeros(len(samples), max_len_x).long() + pad_idx
    res_y = torch.zeros(len(samples), max_len_y).long() + pad_idx
    if backwards: pad_first = not pad_first
    for i,s in enumerate(samples):
        if pad_first:
            res_x[i,-len(s[0]):],res_y[i,-len(s[1]):] = LongTensor(s[0]),LongTens
        else:
            res_x[i,:len(s[0]):],res_y[i,:len(s[1]):] = LongTensor(s[0]),LongTens
    if backwards: res_x,res_y = res_x.flip(1),res_y.flip(1)
    return res_x,res_y
```

Then we create a special `DataBunch` that uses this collate function.

In [8]:

```
doc(Dataset)
```

In [9]:

```
doc(DataLoader)
```

In [6]:

```
doc(DataBunch)
```

In [20]:

```
class Seq2SeqDataBunch(TextDataBunch):
    "Create a `TextDataBunch` suitable for training an RNN classifier."
    @classmethod
    def create(cls, train_ds, valid_ds, test_ds=None, path:PathOrStr='.', bs:int=
               dl_tfms=None, pad_first=False, device:torch.device=None, no_check:
        "Function that transform the `datasets` in a `DataBunch` for classificati
        datasets = cls._init_ds(train_ds, valid_ds, test_ds)
        val_bs = ifnone(val_bs, bs)
        collate_fn = partial(seq2seq_collate, pad_idx=pad_idx, pad_first=pad_firs
        train_sampler = SortishSampler(datasets[0].x, key=lambda t: len(datasets[
        train_dl = DataLoader(datasets[0], batch_size=bs, sampler=train_sampler,
        dataloaders = [train_dl]
        for ds in datasets[1:]:
            lengths = [len(t) for t in ds.x.items]
            sampler = SortSampler(ds.x, key=lengths.__getitem__)
            dataloaders.append(DataLoader(ds, batch_size=val_bs, sampler=sampler,
        return cls(*dataloaders, path=path, device=device, collate_fn=collate_fn,
```

In [ ]:

```
SortishSampler??
```

And a subclass of `TextList` that will use this `DataBunch` class in the call `.databunch` and will use `TextList` to label (since our targets are other texts).

In [21]:
```
class Seq2SeqTextList(TextList):
    _bunch = Seq2SeqDataBunch
    _label_cls = TextList
```

Thats all we need to use the data block API!

In [22]:
```
src = Seq2SeqTextList.from_df(df, path = path, cols='fr').split_by_rand_pct(seed=
```

In [23]:
```
np.percentile([len(o) for o in src.train.x.items] + [len(o) for o in src.valid.x.
```

Out[23]:
```
28.0
```

In [24]:
```
np.percentile([len(o) for o in src.train.y.items] + [len(o) for o in src.valid.y.
```

Out[24]:
```
23.0
```

We remove the items where one of the target is more than 30 tokens long.

In [25]:
```
src = src.filter_by_func(lambda x,y: len(x) > 30 or len(y) > 30)
```

In [26]:
```
len(src.train) + len(src.valid)
```

Out[26]:
```
48352
```

In [27]:
```
data = src.databunch()
```

In [28]:
```
data.save()
```

In [29]:
```
data
```

Out[29]:
```
Seq2SeqDataBunch;

Train: LabelList (38706 items)
x: Seq2SeqTextList
xxbos qu’est - ce que la lumière ?,xxbos où sommes - nous ?,xxbos d'où venons -
nous ?,xxbos que ferions - nous sans elle ?,xxbos quel est le groupe autochtone
principal sur l’île de vancouver ?
y: TextList
xxbos what is light ?,xxbos who are we ?,xxbos where did we come from ?,xxbos w
hat would we do without it ?,xxbos what is the major aboriginal group on vancou
ver island ?
Path: /home/racheltho/.fastai/data/giga-fren;

Valid: LabelList (9646 items)
x: Seq2SeqTextList
xxbos quels pourraient être les effets sur l’instrument de xxunk et sur l’aide
humanitaire qui ne sont pas co - xxunk ?,xxbos quand la source primaire a - t -
elle été créée ?,xxbos pourquoi tant de soldats ont - ils fait xxunk de ne pas
voir ce qui s'est passé le 4 et le 16 mars ?,xxbos quels sont les taux d'impôt
sur le revenu au canada pour 2007 ?,xxbos pourquoi le programme devrait - il in
téresser les employeurs et les fournisseurs de services ?
y: TextList
xxbos what would be the resulting effects on the pre - accession instrument and
humanitarian aid that are not co - decided ?,xxbos when was the primary source
created ?,xxbos why did so many soldiers look the other way in relation to the
incidents of march 4th and march xxunk ?,xxbos what are the income tax rates in
canada for 2007 ?,xxbos why is the program good for employers and service provi
ders ?
Path: /home/racheltho/.fastai/data/giga-fren;

Test: None
```

In [30]:
```
path
```

Out[30]:
```
PosixPath('/home/racheltho/.fastai/data/giga-fren')
```

In [31]:
```
data = load_data(path)
```

In [32]:
```
data.show_batch()
```

text	target
xxbos quelle position devrait - il défendre pour concilier les objectifs stratégiques des divers traités internationaux sur la propriété intellectuelle , l’environnement , et les droits sociaux et économiques ?	xxbos what position should canada advocate with respect to xxunk the policy objectives of various international treaties on intellectual property , the environment , and social and economic rights ?
xxbos que faire s’il semble que pour sauver un stock local de poisson de fond , il xxunk réduire ou éliminer la prédation par les phoques dans le secteur ?	xxbos what if it appears that in some xxunk , saving a local groundfish stock would require reducing or xxunk seal predation in that area ?
xxbos quels sont les impacts économiques produits par les xxunk millions de dollars dépensés par les résidents du yukon qui ont participé à des activités reliées à la nature ?	xxbos what are the economic impacts that result from participation in nature - related activities by residents of the yukon ?
xxbos quelles pourraient être les raisons pour lesquelles un programme n ' a pas marché aussi bien que prévu , même si les employés ont effectué un travail excellent ?	xxbos what would be some of the reasons why a program could be less than successful , even if staff were excellent ?
xxbos quand les pièces , les feuilles ou les fils métalliques contenant des substances de l’inrp figurant dans les parties 1a et 1b perdent - ils leur statut xxunk ?	xxbos when do metal parts , sheets or xxunk containing npri part xxunk and xxunk substances lose their status as articles ?

Create our Model

To install fastText:

```
$ git clone https://github.com/facebookresearch/fastText.git
$ cd fastText
$ pip install .
```

In [33]:
```
import fastText as ft
```

The lines to download the word vectors only need to be run once:

In [60]:
```
# ! wget https://dl.fbaipublicfiles.com/fasttext/vectors-crawl/cc.en.300.bin.gz -
# ! wget https://dl.fbaipublicfiles.com/fasttext/vectors-crawl/cc.fr.300.bin.gz -
```

In [61]:
```
# gunzip {path} / cc.en.300.bin.gz
# gunzip {path} / cc.fr.300.bin.gz
```

In [34]:
```
fr_vecs = ft.load_model(str((path/'cc.fr.300.bin')))
en_vecs = ft.load_model(str((path/'cc.en.300.bin')))
```

We create an embedding module with the pretrained vectors and random data for the missing parts.

In [35]:

```
def create_emb(vecs, itos, em_sz=300, mult=1.):
    emb = nn.Embedding(len(itos), em_sz, padding_idx=1)
    wgts = emb.weight.data
    vec_dic = {w:vecs.get_word_vector(w) for w in vecs.get_words()}
    miss = []
    for i,w in enumerate(itos):
        try: wgts[i] = tensor(vec_dic[w])
        except: miss.append(w)
    return emb
```

In [36]:

```
emb_enc = create_emb(fr_vecs, data.x.vocab.itos)
emb_dec = create_emb(en_vecs, data.y.vocab.itos)
```

In [37]:

```
emb_enc.weight.size(), emb_dec.weight.size()
```

Out[37]:

```
(torch.Size([11336, 300]), torch.Size([8152, 300]))
```

In [38]:

```
model_path = Config().model_path()
```

In [39]:

```
torch.save(emb_enc, model_path/'fr_emb.pth')
torch.save(emb_dec, model_path/'en_emb.pth')
```

In [40]:

```
emb_enc = torch.load(model_path/'fr_emb.pth')
emb_dec = torch.load(model_path/'en_emb.pth')
```

Our Model

In [43]:

```
class Seq2SeqRNN(nn.Module):
    def __init__(self, emb_enc, emb_dec,
                 nh, out_sl,
                 nl=2, bos_idx=0, pad_idx=1):
        super().__init__()
        self.nl,self.nh,self.out_sl = nl,nh,out_sl
        self.bos_idx,self.pad_idx = bos_idx,pad_idx
        self.em_sz_enc = emb_enc.embedding_dim
        self.em_sz_dec = emb_dec.embedding_dim
        self.voc_sz_dec = emb_dec.num_embeddings

        self.emb_enc = emb_enc
        self.emb_enc_drop = nn.Dropout(0.15)
        self.gru_enc = nn.GRU(self.em_sz_enc, nh, num_layers=nl,
                              dropout=0.25, batch_first=True)
        self.out_enc = nn.Linear(nh, self.em_sz_dec, bias=False)

        self.emb_dec = emb_dec
        self.gru_dec = nn.GRU(self.em_sz_dec, self.em_sz_dec, num_layers=nl,
                              dropout=0.1, batch_first=True)
        self.out_drop = nn.Dropout(0.35)
        self.out = nn.Linear(self.em_sz_dec, self.voc_sz_dec)
        self.out.weight.data = self.emb_dec.weight.data

    def encoder(self, bs, inp):
        h = self.initHidden(bs)
        emb = self.emb_enc_drop(self.emb_enc(inp))
        _, h = self.gru_enc(emb, h)
        h = self.out_enc(h)
        return h

    def decoder(self, dec_inp, h):
        emb = self.emb_dec(dec_inp).unsqueeze(1)
        outp, h = self.gru_dec(emb, h)
        outp = self.out(self.out_drop(outp[:,0]))
        return h, outp

    def forward(self, inp):
        bs, sl = inp.size()
        h = self.encoder(bs, inp)
        dec_inp = inp.new_zeros(bs).long() + self.bos_idx

        res = []
        for i in range(self.out_sl):
            h, outp = self.decoder(dec_inp, h)
            dec_inp = outp.max(1)[1]
            res.append(outp)
            if (dec_inp==self.pad_idx).all(): break
        return torch.stack(res, dim=1)

    def initHidden(self, bs): return one_param(self).new_zeros(self.nl, bs, self.
```

In [44]:

```
xb,yb = next(iter(data.valid_dl))
```

In [45]:

```
xb.shape
```

Out[45]:

```
torch.Size([64, 30])
```

In [46]:

```
rnn = Seq2SeqRNN(emb_enc, emb_dec, 256, 30)
```

In [47]:

```
rnn
```

Out[47]:

```
Seq2SeqRNN(
  (emb_enc): Embedding(11336, 300, padding_idx=1)
  (emb_enc_drop): Dropout(p=0.15)
  (gru_enc): GRU(300, 256, num_layers=2, batch_first=True, dropout=0.25)
  (out_enc): Linear(in_features=256, out_features=300, bias=False)
  (emb_dec): Embedding(8152, 300, padding_idx=1)
  (gru_dec): GRU(300, 300, num_layers=2, batch_first=True, dropout=0.1)
  (out_drop): Dropout(p=0.35)
  (out): Linear(in_features=300, out_features=8152, bias=True)
)
```

In [48]:

```
len(xb[0])
```

Out[48]:

```
30
```

In [51]:

```
h = rnn.encoder(64, xb.cpu())
```

In [52]:

```
h.size()
```

Out[52]:

```
torch.Size([2, 64, 300])
```

The loss pads output and target so that they are of the same size before using the usual flattened version of cross entropy. We do the same for accuracy.

In [53]:

```
def seq2seq_loss(out, targ, pad_idx=1):
    bs,targ_len = targ.size()
    _,out_len,vs = out.size()
    if targ_len>out_len: out  = F.pad(out,  (0,0,0,targ_len-out_len,0,0), value=p
    if out_len>targ_len: targ = F.pad(targ, (0,out_len-targ_len,0,0), value=pad_i
    return CrossEntropyFlat()(out, targ)
```

Train our model

In [54]:

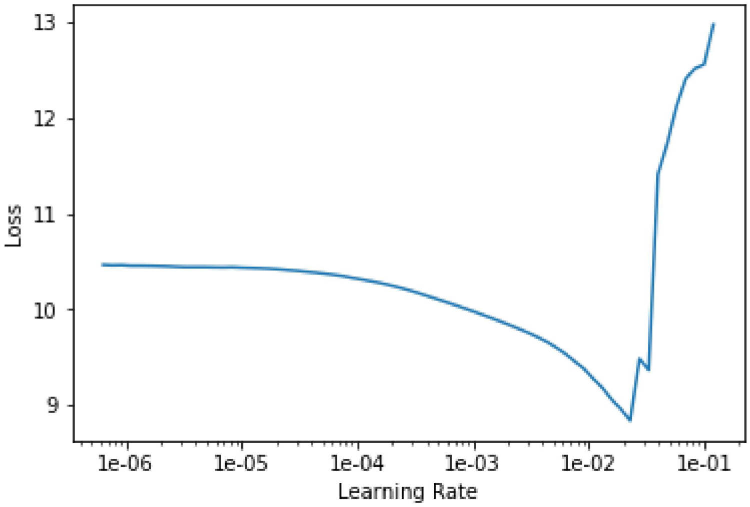
```
learn = Learner(data, rnn, loss_func=seq2seq_loss)
```

In [55]:

```
learn.lr_find()
```

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

In [56]:
```
learn.recorder.plot()
```



In [57]:
```
learn.fit_one_cycle(4, 1e-2)
```

epoch	train_loss	valid_loss	time
0	5.826065	6.018060	00:47
1	5.041347	5.650850	00:44
2	4.651917	4.839034	00:47
3	4.046178	4.601678	00:53

Let's free up some RAM

In [58]:
```
del fr_vecs
del en_vecs
```

As loss is not very interpretable, let's also look at the accuracy. Again, we will add padding so that the output and target are of the same length.

In [59]:
```
def seq2seq_acc(out, targ, pad_idx=1):
    bs,targ_len = targ.size()
    _,out_len,vs = out.size()
    if targ_len>out_len: out  = F.pad(out,  (0,0,0,targ_len-out_len,0,0), value=p
    if out_len>targ_len: targ = F.pad(targ, (0,out_len-targ_len,0,0), value=pad_i
    out = out.argmax(2)
    return (out==targ).float().mean()
```

Bleu metric (see dedicated notebook)

In translation, the metric usually used is BLEU.

A great post by Rachael Tatman: Evaluating Text Output in NLP: BLEU at your own risk (https://towardsdatascience.com/evaluating-text-output-in-nlp-bleu-at-your-own-risk-e8609665a213)

In [60]:

```
class NGram():
    def __init__(self, ngram, max_n=5000): self.ngram,self.max_n = ngram,max_n
    def __eq__(self, other):
        if len(self.ngram) != len(other.ngram): return False
        return np.all(np.array(self.ngram) == np.array(other.ngram))
    def __hash__(self): return int(sum([o * self.max_n**i for i,o in enumerate(se
```

In [61]:

```
def get_grams(x, n, max_n=5000):
    return x if n==1 else [NGram(x[i:i+n], max_n=max_n) for i in range(len(x)-n+1
```

In [62]:

```
def get_correct_ngrams(pred, targ, n, max_n=5000):
    pred_grams,targ_grams = get_grams(pred, n, max_n=max_n),get_grams(targ, n, ma
    pred_cnt,targ_cnt = Counter(pred_grams),Counter(targ_grams)
    return sum([min(c, targ_cnt[g]) for g,c in pred_cnt.items()]),len(pred_grams)
```

In [63]:

```
class CorpusBLEU(Callback):
    def __init__(self, vocab_sz):
        self.vocab_sz = vocab_sz
        self.name = 'bleu'

    def on_epoch_begin(self, **kwargs):
        self.pred_len,self.targ_len,self.corrects,self.counts = 0,0,[0]*4,[0]*4

    def on_batch_end(self, last_output, last_target, **kwargs):
        last_output = last_output.argmax(dim=-1)
        for pred,targ in zip(last_output.cpu().numpy(),last_target.cpu().numpy())
            self.pred_len += len(pred)
            self.targ_len += len(targ)
            for i in range(4):
                c,t = get_correct_ngrams(pred, targ, i+1, max_n=self.vocab_sz)
                self.corrects[i] += c
                self.counts[i]   += t

    def on_epoch_end(self, last_metrics, **kwargs):
        precs = [c/t for c,t in zip(self.corrects,self.counts)]
        len_penalty = exp(1 - self.targ_len/self.pred_len) if self.pred_len < sel
        bleu = len_penalty * ((precs[0]*precs[1]*precs[2]*precs[3]) ** 0.25)
        return add_metrics(last_metrics, bleu)
```

Training with metrics

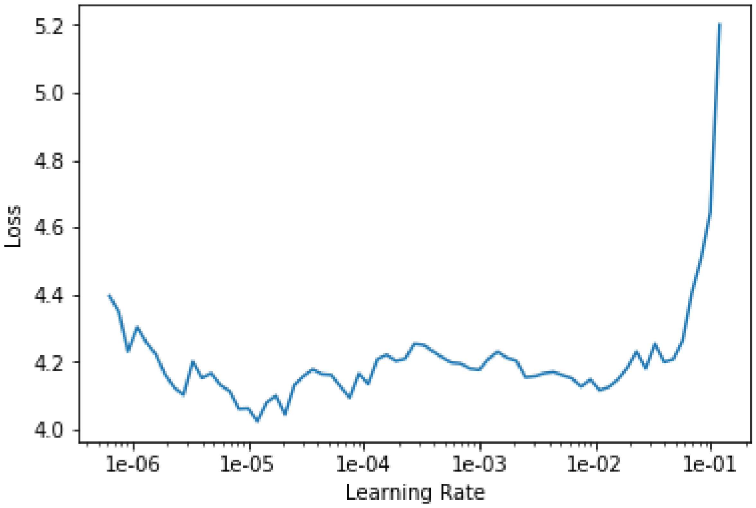
In [64]:

```
learn = Learner(data, rnn, loss_func=seq2seq_loss, metrics=[seq2seq_acc, CorpusBL
```

In [65]:
```
learn.lr_find()
```

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

In [66]:
```
learn.recorder.plot()
```



In [67]:
```
learn.fit_one_cycle(4, 1e-2)
```

epoch	train_loss	valid_loss	seq2seq_acc	bleu	time
0	4.004895	5.146360	0.297541	0.233810	01:02
1	4.265952	4.897265	0.321518	0.269219	01:03
2	3.971066	4.402504	0.366261	0.277486	01:05
3	3.240123	4.291171	0.378903	0.286524	01:06

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

So how good is our model? Let's see a few predictions.

In [68]:
```
def get_predictions(learn, ds_type=DatasetType.Valid):
    learn.model.eval()
    inputs, targets, outputs = [],[],[]
    with torch.no_grad():
        for xb,yb in progress_bar(learn.dl(ds_type)):
            out = learn.model(xb)
            for x,y,z in zip(xb,yb,out):
                inputs.append(learn.data.train_ds.x.reconstruct(x))
                targets.append(learn.data.train_ds.y.reconstruct(y))
                outputs.append(learn.data.train_ds.y.reconstruct(z.argmax(1)))
    return inputs, targets, outputs
```

In [88]:
```
inputs, targets, outputs = get_predictions(learn)
```

```
100.00% [151/151 00:24<00:00]
```

In [89]:
```
inputs[700], targets[700], outputs[700]
```

Out[89]:
```
(Text xxbos quels sont les résultats prévus à court et à long termes de xxunk ,
et dans quelle mesure ont - ils été obtenus ?,
 Text xxbos what are the short and long - term expected outcomes of the ali and
to what extent have they been achieved ?,
 Text xxbos what were the results , the , , , , , , and and and and and)
```

In [90]:
```
inputs[701], targets[701], outputs[701]
```

Out[90]:
```
(Text xxbos de quel(s ) xxunk ) a - t - on besoin pour xxunk les profits réels
de la compagnie pour l'année qui vient ?,
 Text xxbos which of the following additional information is necessary to estim
ate the company 's actual profit for the coming year ?,
 Text xxbos what is the the to to to the the ( ( ) ))
```

In [91]:
```
inputs[2513], targets[2513], outputs[2513]
```

Out[91]:
```
(Text xxbos de quelles façons l'expérience et les capacités particulières des a
gences d'exécution contribuent - elles au projet ?,
 Text xxbos what experience and specific capacities do the implementing organiz
ations bring to the project ?,
 Text xxbos what are the key and and and and and and of of of of of of ?)
```

In [92]:
```
inputs[4000], targets[4000], outputs[4000]
```

Out[92]:
```
(Text xxbos qu'est - ce que la maladie de xxunk - xxunk ( mcj ) ?,
 Text xxbos what is xxunk - xxunk disease ( cjd ) ?,
 Text xxbos what is the xxunk ( ( ) ))
```

It's usually beginning well, but falls into repeated words at the end of the question.

Teacher forcing

One way to help training is to help the decoder by feeding it the real targets instead of its predictions (if it starts with wrong words, it's very unlikely to give us the right translation). We do that all the time at the beginning, then progressively reduce the amount of teacher forcing.

In [83]:

```
class TeacherForcing(LearnerCallback):

    def __init__(self, learn, end_epoch):
        super().__init__(learn)
        self.end_epoch = end_epoch

    def on_batch_begin(self, last_input, last_target, train, **kwargs):
        if train: return {'last_input': [last_input, last_target]}

    def on_epoch_begin(self, epoch, **kwargs):
        self.learn.model.pr_force = 1 - epoch/self.end_epoch
```

We will add the following code to our `forward` method:

```
if (targ is not None) and (random.random()<self.pr_force):
    if i>=targ.shape[1]: break
    dec_inp = targ[:,i]
```

Additionally, `forward` will take an additional argument of `target` .

In [88]:

```
class Seq2SeqRNN_tf(nn.Module):
    def __init__(self, emb_enc, emb_dec, nh, out_sl, nl=2, bos_idx=0, pad_idx=1):
        super().__init__()
        self.nl,self.nh,self.out_sl = nl,nh,out_sl
        self.bos_idx,self.pad_idx = bos_idx,pad_idx
        self.em_sz_enc = emb_enc.embedding_dim
        self.em_sz_dec = emb_dec.embedding_dim
        self.voc_sz_dec = emb_dec.num_embeddings

        self.emb_enc = emb_enc
        self.emb_enc_drop = nn.Dropout(0.15)
        self.gru_enc = nn.GRU(self.em_sz_enc, nh, num_layers=nl,
                              dropout=0.25, batch_first=True)
        self.out_enc = nn.Linear(nh, self.em_sz_dec, bias=False)

        self.emb_dec = emb_dec
        self.gru_dec = nn.GRU(self.em_sz_dec, self.em_sz_dec, num_layers=nl,
                              dropout=0.1, batch_first=True)
        self.out_drop = nn.Dropout(0.35)
        self.out = nn.Linear(self.em_sz_dec, self.voc_sz_dec)
        self.out.weight.data = self.emb_dec.weight.data
        self.pr_force = 0.

    def encoder(self, bs, inp):
        h = self.initHidden(bs)
        emb = self.emb_enc_drop(self.emb_enc(inp))
        _, h = self.gru_enc(emb, h)
        h = self.out_enc(h)
        return h

    def decoder(self, dec_inp, h):
        emb = self.emb_dec(dec_inp).unsqueeze(1)
        outp, h = self.gru_dec(emb, h)
        outp = self.out(self.out_drop(outp[:,0]))
        return h, outp

    def forward(self, inp, targ=None):
        bs, sl = inp.size()
        h = self.encoder(bs, inp)
        dec_inp = inp.new_zeros(bs).long() + self.bos_idx

        res = []
        for i in range(self.out_sl):
            h, outp = self.decoder(dec_inp, h)
            res.append(outp)
            dec_inp = outp.max(1)[1]
            if (dec_inp==self.pad_idx).all(): break
            if (targ is not None) and (random.random()<self.pr_force):
                if i>=targ.shape[1]: continue
                dec_inp = targ[:,i]
        return torch.stack(res, dim=1)

    def initHidden(self, bs): return one_param(self).new_zeros(self.nl, bs, self.
```

In [90]:
```
emb_enc = torch.load(model_path/'fr_emb.pth')
emb_dec = torch.load(model_path/'en_emb.pth')
```

In [91]:
```
rnn_tf = Seq2SeqRNN_tf(emb_enc, emb_dec, 256, 30)

learn = Learner(data, rnn_tf, loss_func=seq2seq_loss, metrics=[seq2seq_acc, Corpu
                callback_fns=partial(TeacherForcing, end_epoch=3))
```

In [74]:
```
learn.lr_find()
```

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

In [75]:
```
learn.recorder.plot()
```

...

In [92]:
```
learn.fit_one_cycle(6, 3e-3)
```

epoch	train_loss	valid_loss	seq2seq_acc	bleu	time
0	2.305473	5.401867	0.195743	0.094855	01:25
1	2.663129	4.858545	0.372653	0.335771	01:13
2	3.337267	4.305145	0.386822	0.319585	01:07
3	4.280678	4.937834	0.314167	0.240478	01:01
4	3.461964	4.086816	0.401147	0.304925	01:06
5	3.154585	4.022432	0.407792	0.310715	01:07

In [77]:
```
inputs, targets, outputs = get_predictions(learn)
```

...

In [78]:
```
inputs[700],targets[700],outputs[700]
```

Out[78]:
```
(Text xxbos qui a le pouvoir de modifier le règlement sur les poids et mesures
et le règlement sur l'inspection de l'électricité et du gaz ?,
 Text xxbos who has the authority to change the electricity and gas inspection
regulations and the weights and measures regulations ?,
 Text xxbos who has the xxunk and xxunk and xxunk xxunk ?)
```

In [79]:
```
inputs[2513], targets[2513], outputs[2513]
```

Out[79]:
```
(Text xxbos quelles sont les deux tendances qui ont nuit à la pêche au saumon d
ans cette province ?,
 Text xxbos what two trends negatively affected the province ’s salmon fishery
?,
 Text xxbos what are the main reasons for the xxunk of the xxunk ?)
```

In [80]:
```
inputs[4000], targets[4000], outputs[4000]
```

Out[80]:
```
(Text xxbos où les aires marines nationales de conservation du canada seront -
elles situées ?,
 Text xxbos where will national marine conservation areas of canada be located
?,
 Text xxbos where are the canadian regulations located in the canadian ?)
```

In [ ]:
```

```