This notebook is adapted from this one (https://github.com/fastai/fastai_docs/blob/master/dev_course/dl2/translation_transformer.ipynb) created by Sylvain Gugger.

See also The Annotated Transformer (http://nlp.seas.harvard.edu/2018/04/03/attention.html) from Harvard NLP.

Attention and the Transformer

Nvidia AI researcher Chip Huyen (https://huyenchip.com/) wrote a great post Top 8 trends from ICLR 2019 (https://huyenchip.com/2019/05/12/top-8-trends-from-iclr-2019.html) in which one of the trends is that *RNN is losing its luster with researchers*.

There's good reason for this, RNNs can be a pain: parallelization can be tricky and they can be difficult to debug. Since language is recursive, it seemed like RNNs were a good conceptual fit with NLP, but recently methods using *attention* have been achieving state of the art results on NLP.

This is still an area of very active research, for instance, a recent paper Pay Less Attention with Lightweight and Dynamic Convolutions (https://arxiv.org/abs/1901.10430) showed that convolutions can beat attention on some tasks, including English to German translation. More research is needed on the various strenghts of RNNs, CNNs, and transformers/attention, and perhaps on approaches to combine the best of each.

In [1]:

```
from fastai.text import *
```

In [2]:

```
path = Config().data_path()/'giga-fren'
path.ls()
```

Out[2]:

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

Load data

We reuse the same functions as in the translation notebook to load our data.

In [3]:
```
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 in samples])
    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]),LongTensor(s[1])
        else:
            res_x[i, :len(s[0])],res_y[i, :len(s[1])] = LongTensor(s[0]),LongTensor(s[1])
    if backwards: res_x,res_y = res_x.flip(1),res_y.flip(1)
    return res_x, res_y
```

In [4]:
```
class Seq2SeqDataBunch(TextDataBunch):
    "Create a `TextDataBunch` suitable for training an RNN classifier."
    @classmethod
    def create(cls, train_ds, valid_ds, test_ds=None, path='.', bs=32, val_bs=None, pad_idx=1,
               dl_tfms=None, pad_first=False, device=None, no_check=False, backwards=False, **dl_kwargs):
        "Function that transform the `datasets` in a `DataBunch` for classification. Passes `**dl_kwargs` on to `DataLoader()`"
        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_first, backwards=backwards)
        train_sampler = SortishSampler(datasets[0].x, key=lambda t: len(datasets[0][t][0].data), bs=bs//2)
        train_dl = DataLoader(datasets[0], batch_size=bs, sampler=train_sampler, drop_last=True, **dl_kwargs)
        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, **dl_kwargs))
        return cls(*dataloaders, path=path, device=device, collate_fn=collate_fn, no_check=no_check)
```

In [5]:
```
class Seq2SeqTextList(TextList):
    _bunch = Seq2SeqDataBunch
    _label_cls = TextList
```

Refer to notebook 7-seq2seq-translation for the code we used to create, process, and save this data.

In [6]:
```
data = load_data(path)
```

In [7]:
```python
data.show_batch()
```

text	target
xxbos quelles questions devraient être traitées respectivement au niveau international et au niveau national , ou quelle xxunk devrait être établie entre la réglementation internationale et la réglementation nationale ?	xxbos which issues should be dealt with internationally and which nationally , or what division should be made between international regulation and national regulation ?
xxbos comment la culture et les arts y vivent - ils , et comment la société civile les prend - elle en considération dans le développement de la ville ?	xxbos where do art and culture fit in , and how is civil society taking them into consideration in developing the city ?
xxbos qu’arrivera - t - il si les entreprises canadiennes et les gouvernements ne se xxunk pas sur la question du conflit entre le travail et la vie personnelle ?	xxbos what will likely happen if canadian organizations and governments do not deal with the issue of work – life conflict ?
xxbos qu'adviendra - t - il de l'examen de rendement et de la rémunération au rendement de xxunk et comment se xxunk - t - il à cet égard ?	xxbos what happens to xxunk 's evaluation review and performance pay and how will this make him feel ?
xxbos quels avantages prévoit - on en général pour la région du delta de beaufort par suite de la signature de xxunk / d'une entente future sur l'autonomie gouvernementale ?	xxbos what benefits to the beaufort delta region generally are expected as a result of this aip / future self - government ?

Transformer model

Transformer model

Shifting

We add a transform to the dataloader that shifts the targets right and adds a padding at the beginning.

In [8]:
```python
v = data.vocab
```

In [9]:
```python
v.stoi['xxpad']
```

Out[9]:
```
1
```

In [10]:
```python
def shift_tfm(b):
    x,y = b
    y = F.pad(y, (1, 0), value=1)
    return [x,y[:,:-1]], y[:,1:]
```

In [11]:
```python
data.add_tfm(shift_tfm)
```

Embeddings

The input and output embeddings are traditional PyTorch embeddings (and we can use pretrained vectors if we want to). The transformer model isn't a recurrent one, so it has no idea of the relative positions of the words. To help it with that, they had to the input embeddings a positional encoding which is cosine of a certain frequency:

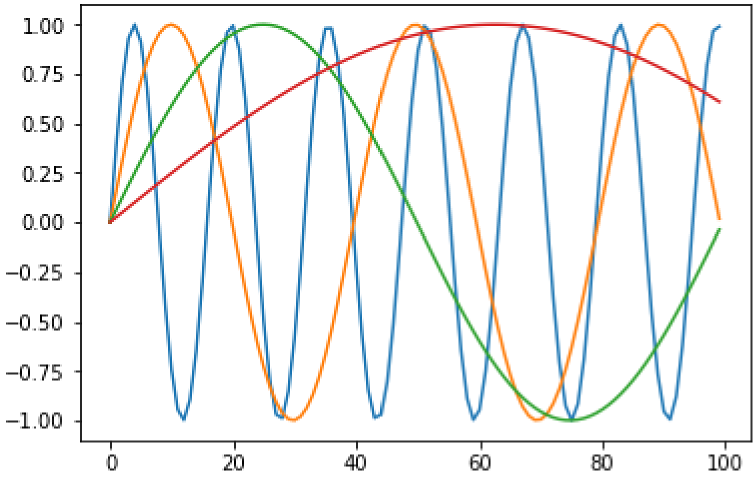
In [156]:
```
d = 30
torch.arange(0., d, 2.)/d
```

Out[156]:
```
tensor([0.0000, 0.0667, 0.1333, 0.2000, 0.2667, 0.3333, 0.4000, 0.4667, 0.533
3,
        0.6000, 0.6667, 0.7333, 0.8000, 0.8667, 0.9333])
```

In [12]:
```
class PositionalEncoding(nn.Module):
    "Encode the position with a sinusoid."
    def __init__(self, d):
        super().__init__()
        self.register_buffer('freq', 1 / (10000 ** (torch.arange(0., d, 2.)/d
)))

    def forward(self, pos):
        inp = torch.ger(pos, self.freq)
        enc = torch.cat([inp.sin(), inp.cos()], dim=-1)
        return enc
```

In [13]:
```
tst_encoding = PositionalEncoding(20)
res = tst_encoding(torch.arange(0,100).float())
_, ax = plt.subplots(1,1)
for i in range(1,5): ax.plot(res[:,i])
```



In [14]:
```
res[:6,:6]
```

Out[14]:
```
tensor([[ 0.0000,  0.0000,  0.0000,  0.0000,  0.0000,  0.0000],
        [ 0.8415,  0.3877,  0.1578,  0.0631,  0.0251,  0.0100],
        [ 0.9093,  0.7147,  0.3117,  0.1259,  0.0502,  0.0200],
        [ 0.1411,  0.9300,  0.4578,  0.1882,  0.0753,  0.0300],
        [-0.7568,  0.9998,  0.5923,  0.2497,  0.1003,  0.0400],
        [-0.9589,  0.9132,  0.7121,  0.3103,  0.1253,  0.0500]])
```

In [15]:

```
class TransformerEmbedding(nn.Module):
    "Embedding + positional encoding + dropout"
    def __init__(self, vocab_sz, emb_sz, inp_p=0.):
        super().__init__()
        self.emb_sz = emb_sz
        self.embed = embedding(vocab_sz, emb_sz)
        self.pos_enc = PositionalEncoding(emb_sz)
        self.drop = nn.Dropout(inp_p)

    def forward(self, inp):
        pos = torch.arange(0, inp.size(1), device=inp.device).float()
        return self.drop(self.embed(inp) * math.sqrt(self.emb_sz) + self.pos_e
nc(pos))
```

Feed forward

The feed forward cell is easy: it's just two linear layers with a skip connection and a LayerNorm.

In [16]:

```
def feed_forward(d_model, d_ff, ff_p=0., double_drop=True):
    layers = [nn.Linear(d_model, d_ff), nn.ReLU()]
    if double_drop: layers.append(nn.Dropout(ff_p))
    return SequentialEx(*layers, nn.Linear(d_ff, d_model), nn.Dropout(ff_p), M
ergeLayer(), nn.LayerNorm(d_model))
```

Multi-head attention

Multi head attention

In [117]:

```
class MultiHeadAttention(nn.Module):
    def __init__(self, n_heads, d_model, d_head=None, p=0., bias=True, scale=True):
        super().__init__()
        d_head = ifnone(d_head, d_model//n_heads)
        self.n_heads,self.d_head,self.scale = n_heads,d_head,scale
        self.q_wgt,self.k_wgt,self.v_wgt = [nn.Linear(
            d_model, n_heads * d_head, bias=bias) for o in range(3)]
        self.out = nn.Linear(n_heads * d_head, d_model, bias=bias)
        self.drop_att,self.drop_res = nn.Dropout(p),nn.Dropout(p)
        self.ln = nn.LayerNorm(d_model)

    def forward(self, q, kv, mask=None):
        return self.ln(q + self.drop_res(self.out(self._apply_attention(q, kv, mask=mask))))

    def create_attn_mat(self, x, layer, bs):
        return layer(x).view(bs, x.size(1), self.n_heads, self.d_head
                            ).permute(0, 2, 1, 3)

    def _apply_attention(self, q, kv, mask=None):
        bs,seq_len = q.size(0),q.size(1)
        wq,wk,wv = map(lambda o: self.create_attn_mat(*o,bs),
                       zip((q,kv,kv),(self.q_wgt,self.k_wgt,self.v_wgt)))
        attn_score = wq @ wk.transpose(2,3)
        if self.scale: attn_score /= math.sqrt(self.d_head)
        if mask is not None:
            attn_score = attn_score.float().masked_fill(mask, -float('inf')).type_as(attn_score)
        attn_prob = self.drop_att(F.softmax(attn_score, dim=-1))
        attn_vec = attn_prob @ wv
        return attn_vec.permute(0, 2, 1, 3).contiguous().view(bs, seq_len, -1)
```

Masking

The attention layer uses a mask to avoid paying attention to certain timesteps. The first thing is that we don't really want the network to pay attention to the padding, so we're going to mask it. The second thing is that since this model isn't recurrent, we need to mask (in the output) all the tokens we're not supposed to see yet (otherwise it would be cheating).

In [18]:

```
def get_output_mask(inp, pad_idx=1):
    return torch.triu(inp.new_ones(inp.size(1),inp.size(1)), diagonal=1)[None,None].byte()
#     return ((inp == pad_idx)[:,None,:,None].long() + torch.triu(inp.new_ones(inp.size(1),inp.size(1)), diagonal=1)[None,None] != 0)
```

Example of mask for the future tokens:

In [19]:
```
torch.triu(torch.ones(10,10), diagonal=1).byte()
```

Out[19]:
```
tensor([[0, 1, 1, 1, 1, 1, 1, 1, 1, 1],
        [0, 0, 1, 1, 1, 1, 1, 1, 1, 1],
        [0, 0, 0, 1, 1, 1, 1, 1, 1, 1],
        [0, 0, 0, 0, 1, 1, 1, 1, 1, 1],
        [0, 0, 0, 0, 0, 1, 1, 1, 1, 1],
        [0, 0, 0, 0, 0, 0, 1, 1, 1, 1],
        [0, 0, 0, 0, 0, 0, 0, 1, 1, 1],
        [0, 0, 0, 0, 0, 0, 0, 0, 1, 1],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 1],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]], dtype=torch.uint8)
```

Encoder and decoder blocks

We are now ready to regroup these layers in the blocks we add in the model picture:

Transformer model

In [125]:
```
class EncoderBlock(nn.Module):
    "Encoder block of a Transformer model."
    #Can't use Sequential directly cause more than one input...
    def __init__(self, n_heads, d_model, d_head, d_inner, p=0., bias=True, scale=True, double_drop=True):
        super().__init__()
        self.mha = MultiHeadAttention(n_heads, d_model, d_head, p=p, bias=bias, scale=scale)
        self.ff  = feed_forward(d_model, d_inner, ff_p=p, double_drop=double_drop)

    def forward(self, x, mask=None): return self.ff(self.mha(x, x, mask=mask))
```

In [126]:
```
class DecoderBlock(nn.Module):
    "Decoder block of a Transformer model."
    #Can't use Sequential directly cause more than one input...
    def __init__(self, n_heads, d_model, d_head, d_inner, p=0., bias=True, scale=True, double_drop=True):
        super().__init__()
        self.mha1 = MultiHeadAttention(n_heads, d_model, d_head, p=p, bias=bias, scale=scale)
        self.mha2 = MultiHeadAttention(n_heads, d_model, d_head, p=p, bias=bias, scale=scale)
        self.ff   = feed_forward(d_model, d_inner, ff_p=p, double_drop=double_drop)

    def forward(self, x, enc, mask_out=None): return self.ff(self.mha2(self.mha1(x, x, mask_out), enc))
```

The whole model

In [127]:

```
class Transformer(Module):
    def __init__(self, inp_vsz, out_vsz, n_layers=6, n_heads=8, d_model=256, d
_head=32,
                 d_inner=1024, p=0.1, bias=True, scale=True, double_drop=True,
pad_idx=1):
        self.enc_emb = TransformerEmbedding(inp_vsz, d_model, p)
        self.dec_emb = TransformerEmbedding(out_vsz, d_model, 0.)
        args = (n_heads, d_model, d_head, d_inner, p, bias, scale, double_drop
)
        self.encoder = nn.ModuleList([EncoderBlock(*args) for _ in range(n_lay
ers)])
        self.decoder = nn.ModuleList([DecoderBlock(*args) for _ in range(n_lay
ers)])
        self.out = nn.Linear(d_model, out_vsz)
        self.out.weight = self.dec_emb.embed.weight
        self.pad_idx = pad_idx

    def forward(self, inp, out):
        mask_out = get_output_mask(out, self.pad_idx)
        enc,out = self.enc_emb(inp),self.dec_emb(out)
        enc = compose(self.encoder)(enc)
        out = compose(self.decoder)(out, enc, mask_out)
        return self.out(out)
```

Bleu metric (see dedicated notebook)

In [23]:

```
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
(self.ngram)]))
```

In [24]:

```
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 [25]:

```
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,
max_n=max_n)
    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_gra
ms)
```

In [26]:

```
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 <
self.targ_len else 1
        bleu = len_penalty * ((precs[0]*precs[1]*precs[2]*precs[3]) ** 0.25)
        return add_metrics(last_metrics, bleu)
```

Training

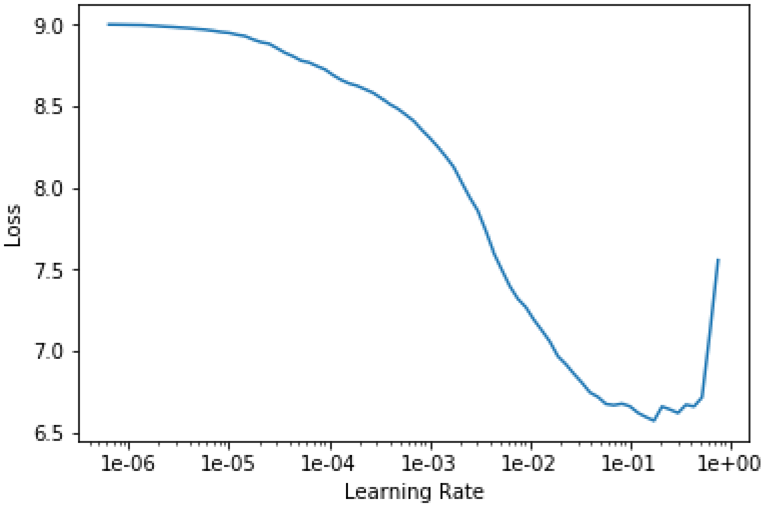
In [128]:

```
n_x_vocab,n_y_vocab = len(data.train_ds.x.vocab.itos), len(data.train_ds.y.voc
ab.itos)

model = Transformer(n_x_vocab, n_y_vocab, d_model=256)
learn = Learner(data, model, metrics=[accuracy, CorpusBLEU(n_y_vocab)], loss_f
unc = CrossEntropyFlat())
```

In [129]:
```
learn.lr_find()
learn.recorder.plot()
```

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



In [154]:
```
learn.fit_one_cycle(8, 5e-4, div_factor=5)
```

epoch	train_loss	valid_loss	accuracy	bleu	time
0	2.388738	2.521896	0.608056	0.440603	02:04
1	1.901857	1.958518	0.687528	0.504890	01:55
2	1.565219	1.652850	0.723345	0.536689	02:00
3	1.306575	1.455323	0.747950	0.565299	02:02
4	1.137455	1.342423	0.764108	0.587329	02:00
5	0.949592	1.282115	0.774509	0.601252	02:00
6	0.808173	1.269965	0.778810	0.607137	01:58
7	0.767959	1.274921	0.778822	0.607116	01:51

In [49]:
```
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[0],xb[1],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 [50]:
```
inputs, targets, outputs = get_predictions(learn)
```

```
100.00% [149/149 00:28<00:00]
```

In [ ]:
```
inputs[10],targets[10],outputs[10]
```

Out[ ]:
```
(Text xxbos xxmaj pendant que xxunk les activités requises pour maintenir mon
xxunk physique , est - ce que je xxunk de la protection d’un régime d’assuran
ce ou de pension ?,
 Text xxbos xxmaj while i go about maintaining this high degree of fitness ,
am i protected under an insurance or pension plan ?,
 Text xxbos xxmaj while i do to the my physical physical of physical , do i a
ware by the pension plan service plan ?)
```

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

Out[ ]:
```
(Text xxbos xxmaj quelles sont les conséquences sur la recherche , la mise en
pratique et les politiques en ce qui a trait à l'ac ?,
 Text xxbos xxmaj what are the xxunk for xxup kt research , practice / policy
?,
 Text xxbos xxmaj what are the implications implications research kt , , poli
cy and policies in)
```

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

Out[ ]:
```
(Text xxbos xxmaj quelle est la position des xxmaj états - xxmaj unis , du xx
maj canada et de la xxup xxunk à ce propos ?,
 Text xxbos xxmaj where do the xxup us , xxmaj canada and xxup xxunk stand ?,
 Text xxbos xxmaj what is xxmaj xxup us xxmaj xxmaj united and the xxunk fit
in)
```

In [ ]:
```
inputs[2500],targets[2500],outputs[2500]
```

Out[ ]:
```
(Text xxbos xxmaj quels sont les atouts particuliers du xxmaj canada en reche
rche sur l'obésité sur la scène internationale ?,
 Text xxbos xxmaj what are the unique xxmaj canadian strengths in obesity res
earch that set xxmaj canada apart on an international front ?,
 Text xxbos xxmaj what are xxmaj specific strengths canada strengths in obesi
ty - ? are up canada ? from international international stage ?)
```

In [ ]:
```
inputs[4002],targets[4002],outputs[4002]
```

Out[ ]:
```
(Text xxbos xxmaj quelles sont les répercussions politiques à long terme de c
ette révolution scientifique mondiale ?,
 Text xxbos xxmaj what are some of the long - term policy implications of thi
s global knowledge revolution ?,
 Text xxbos xxmaj what are the long the long - term policies implications of
this global scientific ? ?)
```

Label smoothing

They point out in the paper that using label smoothing helped getting a better BLEU/accuracy, even if it made the loss worse.

In [51]:
```
model = Transformer(len(data.train_ds.x.vocab.itos), len(data.train_ds.y.vocab
.itos), d_model=256)
```

In [52]:
```
learn = Learner(data, model, metrics=[accuracy, CorpusBLEU(len(data.train_ds.y
.vocab.itos))],
                loss_func=FlattenedLoss(LabelSmoothingCrossEntropy, axis=-1))
```

In [53]:
```
learn.fit_one_cycle(8, 5e-4, div_factor=5)
```

epoch	train_loss	valid_loss	accuracy	bleu	time
0	3.281034	3.357356	0.621848	0.458009	01:45
1	2.872045	2.923340	0.690921	0.510376	01:47
2	2.598603	2.653438	0.729291	0.545735	01:49
3	2.407944	2.514847	0.748187	0.567057	01:46
4	2.195246	2.403729	0.766409	0.592165	01:50
5	2.095695	2.362098	0.776127	0.604666	01:48
6	1.999303	2.358647	0.779535	0.609675	01:47
7	1.923621	2.359421	0.780211	0.610871	01:47

In [ ]:
```
learn.fit_one_cycle(8, 5e-4, div_factor=5)
```

In [ ]:
```
print("Quels sont les atouts particuliers du Canada en recherche sur l'obésité
sur la scène internationale ?")
print("What are Specific strengths canada strengths in obesity - ? are up cana
da ? from international international stage ?")
print("Quelles sont les répercussions politiques à long terme de cette révolut
ion scientifique mondiale ?")
print("What are the long the long - term policies implications of this global
 scientific ? ?")
```

```
Quels sont les atouts particuliers du Canada en recherche sur l'obésité sur l
a scène internationale ?
What are Specific strengths canada strengths in obesity - ? are up canada ? f
rom international international stage ?
Quelles sont les répercussions politiques à long terme de cette révolution sc
ientifique mondiale ?
What are the long the long - term policies implications of this global scient
ific ? ?
```

In [ ]:
```
inputs[10],targets[10],outputs[10]
```

Out[ ]:
```
(Text xxbos xxmaj quelle distance y a - t - il entre le point le plus rapproc
hé de la surface à xxunk et la position d’utilisation habituelle du tube radi
ogène ?,
 Text xxbos xxmaj what is the distance between the nearest point of the area
to be shielded and the usual operational position of the x - ray tube ?,
 Text xxbos xxmaj what is the xxmaj between the xxmaj and of the xxmaj ? the
? and the most ? ? of the xxmaj - ray tube ?)
```

In [ ]:

```
inputs[700],targets[700],outputs[700]
```

Out[ ]:

```
(Text xxbos xxmaj quels types de présentations xxmaj santé xxmaj canada xxunk
- t - il dans le format ectd à compter du 1er septembre ?,
 Text xxbos xxmaj what kind of submission types will xxmaj health xxmaj canad
a accept on xxmaj september 1 , 2004 in ectd format ?,
 Text xxbos xxmaj what is of information is of be canadian xxmaj canada take
? the canadian ? , and ? the format ?)
```

In [ ]:

```
inputs[701],targets[701],outputs[701]
```

Out[ ]:

```
(Text xxbos xxmaj quelles sont les trois caractéristiques qui vous incitent l
e plus à investir dans votre région ( xxup nommez - xxup les ) ?,
 Text xxbos xxmaj what are the three most attractive features about investing
in your region ( xxup name xxup it ) ?,
 Text xxbos xxmaj what is the main main important concerns of the in the coun
try ? xxup xxunk , xxunk ) ?)
```

In [ ]:

```
inputs[4001],targets[4001],outputs[4001]
```

Out[ ]:

```
(Text xxbos xxmaj quelles actions avez - vous prises et quel en a été le résu
ltat ?,
 Text xxbos xxmaj what were your actions and the outcomes ?,
 Text xxbos xxmaj what is the targets ? how main of)
```

Test leakage

If we change a token in the targets at position n, it shouldn't impact the predictions before that.

In [ ]:

```
learn.model.eval();
```

In [ ]:

```
xb,yb = data.one_batch(cpu=False)
```

In [ ]:

```
inp1,out1 = xb[0][:1],xb[1][:1]
inp2,out2 = inp1.clone(),out1.clone()
out2[0,15] = 10
```

In [ ]:

```
y1 = learn.model(inp1, out1)
y2 = learn.model(inp2, out2)
```

In [ ]:

```
(y1[0,:15] - y2[0,:15]).abs().mean()
```

Out[ ]:

```
tensor(0., device='cuda:0', grad_fn=<MeanBackward1>)
```