This lesson was adapted from Fast.ai's Code-First Introduction to NLP (https://www.fast.ai/2019/07/08/fastai-nlp/).

Language Modeling & Analysis of Work Accident Outcomes

We will be looking at mine injury reports (https://developer.dol.gov/health-and-safety/mine-accident-injuries/). We want to determine if a mine accident resulted in missed work days, based on the text alone. In order to do this, we will be using **transfer learning**.

Transfer learning has been widely used with great success in computer vision for several years, but only in the last year or so has it been successfully applied to NLP (beginning with ULMFit, which we will use here, which was built upon by BERT and GPT-2).

As Sebastian Ruder wrote in The Gradient (https://thegradient.pub/) in summer 2018, NLP's ImageNet moment has arrived (https://thegradient.pub/nlp-imagenet/).

Language Models

Language modeling can be a fun creative form. Research scientist Janelle Shane blogs (https://aiweirdness.com/) & tweets (https://twitter.com/JanelleCShane) about her creative AI explorations, which often involve text. For instance, see her:

- Why did the neural network cross the road? (https://aiweirdness.com/post/174691534037/why-did-the-neural-network-cross-the-road)
- Try these neural network-generated recipes at your own risk. (https://aiweirdness.com/post/163878889437/try-these-neural-network-generated-recipes-at-your)
- D&D character bios - now making slightly more sense (https://aiweirdness.com/post/183471928977/dd-character-bios-now-making-slightly-more)

Using a GPU

You will need to have the fastai library installed for this lesson, and you will want to use a GPU to train your neural net. If you don't have a GPU you can use in your computer (currently, only Nvidia GPUs are fully supported by the main deep learning libraries), no worries! There are a number of cloud options you can consider:

GPU Cloud Options (https://course.fast.ai/#using-a-gpu)

Reminder: If you are using a cloud GPU, always be sure to shut it down when you are done!!! Otherwise, you could end up with an expensive bill!

In [64]:
```
%reload_ext autoreload
%autoreload 2
%matplotlib inline
```

In [65]:
```
from fastai import *
from fastai.text import *
```

In [66]:
```
# import fastai.utils.collect_env

# fastai.utils.collect_env.show_install()
```

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

In [67]:
```
# bs=48
# bs=24
bs=192
```

In [68]:
```
torch.cuda.set_device(0)
```

Preparing the data

Let's load the data

File failed to load: /extensions/MathZoom.js

In [7]:
```
df = pd.read_csv('msha_accident_tal_1.csv')
df.head()
```

Out[7]:

	ai_narr	days_lost
0	Employee was retrieving a material sample from...	0.0
1	While blowing bridged lime from a chute, the e...	0.0
2	EE was helping to un-plug a dust chute and got...	0.0
3	Employee arrived at work and went to his locke...	0.0
4	Working with sledge hammer. Index finger on le...	0.0

Our target is whether or not the worker loast any work days

In [9]:
```
df['days_lost'] = df['days_lost'] > 0
df['days_lost'].value_counts()
```

Out[9]:
```
False    128174
True     125027
Name: days_lost, dtype: int64
```

We can see that the dataset is well balanced.

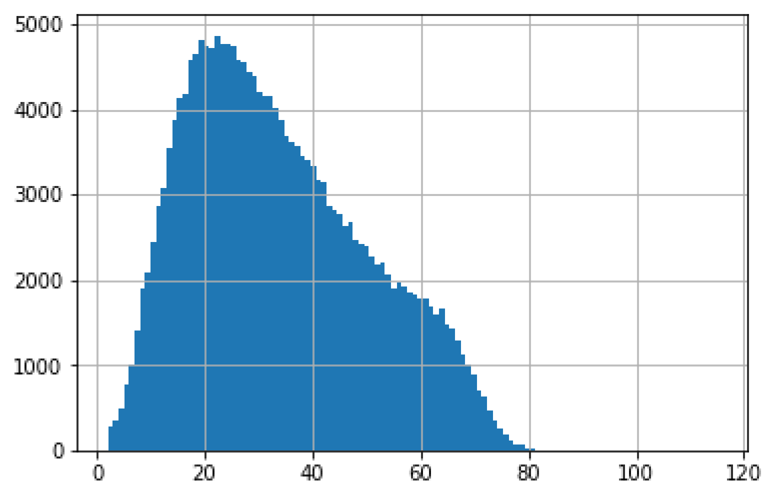
In [10]:
```
df['num words'] = df['ai_narr'].apply(lambda x: len(str(x).split()))
```

In [11]:
```
df['num words'].describe()
```

Out[11]:
```
count    253201.000000
mean         26.538691
std          20.010620
min           1.000000
25%          10.000000
50%          25.000000
75%          41.000000
max         115.000000
Name: num words, dtype: float64
```

In [12]:
```
# Here is a histogram of the number of words in the accident narrative
df.loc[df['num words'] > 1, 'num words'].hist(bins=114)
```

Out[12]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f41ba5bd550>
```



In [24]:
```
df['num words'].value_counts().sort_index().head(10)
```

Out[24]:
```
1     54683
2       268
3       354
4       492
5       768
6      1012
7      1396
8      1888
9      2081
10     2440
Name: num words, dtype: int64
```

In [13]:
```
# We will only keep records with the number of words greater than 10
df.loc[df['num words'] >= 10, 'days_lost'].value_counts(dropna=True)
```

Out[13]:
```
False    99739
True     90520
Name: days_lost, dtype: int64
```

In [15]:
```
# Export the smaller dataset to a spreadheet
df.loc[df['num words'] >= 10, ['ai_narr', 'days_lost']].to_csv('accident_dataset.csv')
```

File failed to load: /extensions/MathZoom.js

In [7]:
```
# Define the path for input files
path = Path('.')
```

In [8]:
```
data_lm = TextLMDataBunch.from_csv(path, 'accident_dataset.csv', label_cols='days_lost', text_cols='ai_narr
```

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.

Tokenization

The first step of processing we make 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.

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 his: "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 unkown 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 list 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 tokens is stored in the `vocab` attribute of our datasets, in a dictionary called `itos` (for int to string).

In [26]:
```
data_lm.vocab.itos[:10]
```

Out[26]:
```
['xxunk',
 'xxpad',
 'xxbos',
 'xxeos',
 'xxfld',
 'xxmaj',
 'xxup',
 'xxrep',
 'xxwrep',
 '.']
```

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

In [27]:
```
data_lm.train_ds
```

Out[27]:
```
LabelList (152207 items)
x: LMTextList
xxbos xxmaj while bonding a intake stopping he used his bare hands to put b - bond on a block stopping , he wa
s offered a pair of gloves . xxmaj he choose not to use them .,xxbos a xxmaj volvo xxup a40 g off road 6x6 tru
ck made contact with a xxmaj komatsu xxup pc360 excavator while approaching from the xxunk of the excavator .
xxmaj this incident was investigated by xxmaj mine xxmaj safety and xxmaj health xxmaj inspector on 09 / 12 /
2016 . xxmaj the complaint xxup id is xxunk . xxmaj we were notified on 10 / 03 / 2016 that the operator of th
e xxmaj volvo 6x6 truck had filed a claim of injury .,xxbos xxmaj the employee went to the company clinic on 8
/ 1 / 11 and was placed on restricted duty . xxmaj the employee reported that he lost his footing , fell and s
truck his r - knee cap on a protruding 5 / 8 \ bolt that secured the dust pan . ",xxbos xxmaj employee was ser
vicing a xxmaj cat 307 xxmaj excavator in the shop . xxmaj the employee had stepped down from the machine to g
et a tool and was climbing back up to remove an access cover from the hydraulic tank when hand slipped off the
rail and xxup ee fell backwards , injuring left elbow and left shoulder .,xxbos xxup injured xxup was xxup cle
aning xxup the xxup canopy xxup top xxup of xxup longwall xxup shield . a xxup rock xxup fell xxup while xxup
injured xxup was xxup pulling xxup rock xxup from xxup the xxup canopy . xxup the xxup rock xxup struck xxup i
njured xxup on xxup the xxup right xxup hand xxup resulting xxup in a xxup fracture xxup and xxup laceration .
y: LMLabelList
,,,,
Path: .
```

File failed to load: /extensions/MathZoom.js

But the underlying data is all numbers

In [28]:
```
data_lm.train_ds[0][0].data[:10]
```

Out[28]:
```
array([   2,    5,   31, 7853,   12,  879,  807,   21,  319,   19])
```

Alternative apporach: 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 various arguments to pass will appear in the step they're revelant, so it'll be more readable.

In [29]:
```
# data = (TextList.from_csv(path, 'texts.csv', cols='text')
#                 .split_from_df(col=2)
#                 .label_from_df(cols=0)
#                 .databunch())
```

Language model

We're not going to train a model that classifies the accidents from scratch. Like in computer vision, we'll use a model pretrained on a bigger dataset (a cleaned subset of wikipeia 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 injury reports isn't the same as the English of wikipedia, we'll need to adjust a little bit the parameters of our model. Plus there might be some words extremely common in that dataset that were barely present in wikipedia, and therefore might no be part of the vocabulary the model was trained on.

More about WikiText-103

We will be using the WikiText-103 (https://blog.einstein.ai/the-wikitext-long-term-dependency-language-modeling-dataset/) dataset created by Stephen Merity (https://smerity.com/) to pre-train a language model.

To quote Stephen's post (https://blog.einstein.ai/the-wikitext-long-term-dependency-language-modeling-dataset/):

The WikiText language modeling dataset is a collection of over 100 million tokens extracted from the set of verified Good and Featured articles on Wikipedia. The dataset is available under the Creative Commons Attribution-ShareAlike License.

Compared to the preprocessed version of Penn Treebank (PTB), WikiText-2 is over 2 times larger and WikiText-103 is over 110 times larger. The WikiText dataset also features a far larger vocabulary and retains the original case, punctuation and numbers - all of which are removed in PTB. As it is composed of full articles, the dataset is well suited for models that can take advantage of long term dependencies.

Creating the TextLMDataBunch

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 [30]:
```
# path.ls()
```

In [31]:
```
len(data_lm.vocab.itos),len(data_lm.train_ds)
```

Out[31]:
```
(21960, 152207)
```

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.

File failed to load: /extensions/MathZoom.js

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

idx	text
0	was investigated by xxmaj mine xxmaj safety and xxmaj health xxmaj inspector on 09 / 12 / 2016 . xxmaj the complaint xxup id is xxunk . xxmaj we were notified on 10 / 03 / 2016 that the operator of the xxmaj volvo 6x6 truck had filed a claim of injury . xxbos xxmaj the employee went to the company clinic on 8 / 1 / 11 and was
1	) employee 's foot slipped and he fell off the unloader to the ground (about 8 ft) causing laceration to left knee , requiring stitches along with contusions & scrapes to left side of face . xxbos xxmaj worker got cut on the head when a piece of 2 \ x2"x1 / xxunk " angle iron employee was attempting to bend back into position as railing swung back
2	. xxup ee was seen by a physician and assigned to restricted duty for a period of two weeks . xxbos xxmaj employee xxmaj threw his back out stacking bags . xxmaj felt spasms in lower back , xxmaj lost most function in his left arm from pain in the back . xxmaj is receiving chiropractic treatments and has not missed any time off from work . xxbos xxmaj while
3	. xxbos xxmaj the injured was on the operators side , pushing a cable bolt up to finish its installation when he was struck on the right side of his face by an 18 \ long bolt wrench which was ejected from within the scissor action of the bolter head under extreme force . " xxbos xxup ee was getting tools together to remove an inspection plate on the xxmaj
4	right shoulder , his neck and back . xxbos xxup injured xxup taking xxup down xxup fly xxup pads xxup and xxup cut xxup knuckle xxup on xxup roof xxup bolt xxup plate . xxbos xxup ee was welding when he turned to get a part when another xxup ee began grinding . xxmaj the grinding debris went under xxup ee 's safety glasses & struck his left eye .

Let's save our databunch for next time:

In [33]:
```
data_lm.save('lm_databunch')
```

Loading saved data, and creating the language model

In the future we can load the data:

In [34]:
```
path = Path('.')
data_lm = load_data(path, 'lm_databunch', bs=bs)
```

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 [35]:
```
learn_lm = language_model_learner(data_lm, AWD_LSTM, drop_mult=0.3)
```

In [36]:
```
wiki_itos = pickle.load(open(Config().model_path()/'wt103-fwd/itos_wt103.pkl', 'rb'))
```

In [37]:
```
wiki_itos[:10]
```

Out[37]:
```
['xxunk',
 'xxpad',
 'xxbos',
 'xxeos',
 'xxfld',
 'xxmaj',
 'xxup',
 'xxrep',
 'xxwrep',
 'the']
```

In [38]:
```
vocab = data_lm.vocab
```

In [39]:
```
vocab.stoi["injured"]
```

Out[39]:
```
80
```

In [40]:
```
vocab.itos[vocab.stoi["injury"]]
```

Out[40]:
```
'injury'
```

In [41]:
```
vocab.itos[vocab.stoi["mobula"]]
```

Out[41]:
```
'xxunk'
```

In [42]:
```
awd = learn_lm.model[0]
```

In [43]:
```
from scipy.spatial.distance import cosine as dist
```

In [44]:
```
enc = learn_lm.model[0].encoder
```

In [45]:
```
enc.weight.size()
```

Out[45]:
```
torch.Size([21960, 400])
```

File failed to load: /extensions/MathZoom.js

Difference in vocabulary between the Injury reports and Wikipedia

We are going to load wiki_itos, which can be downloaded along with wikitext-103. We will compare the vocabulary from wikitext with the vocabulary in IMDB. It is to be expected that the two sets have some different vocabulary words, and that is no problem for transfer learning!

In [46]:
```
len(wiki_itos)
```
Out[46]:
```
60000
```

In [47]:
```
len(vocab.itos)
```
Out[47]:
```
21960
```

In [48]:
```
i, unks = 0, []
while len(unks) < 50:
    if data_lm.vocab.itos[i] not in wiki_itos: unks.append((i,data_lm.vocab.itos[i]))
    i += 1
```

In [49]:
```
wiki_words = set(wiki_itos)
```

In [50]:
```
injury_rep_words = set(vocab.itos)
```

In [51]:
```
wiki_not_injury_rep = wiki_words.difference(injury_rep_words)
```

In [52]:
```
injury_rep_not_wiki = injury_rep_words.difference(wiki_words)
```

In [53]:
```
wiki_not_injury_rep_list = []

for i in range(100):
    word = wiki_not_injury_rep.pop()
    wiki_not_injury_rep_list.append(word)
    wiki_not_injury_rep.add(word)
```

In [54]:
```
wiki_not_injury_rep_list[:15]
```
Out[54]:
```
['unicorn',
 'cluttered',
 'mccarron',
 'amazingly',
 'resnick',
 'moonbase',
 'glutamate',
 'así',
 'reuniting',
 'lent',
 'wefaq',
 'provinces',
 'roeg',
 'yak',
 'leete']
```

In [55]:
```
injury_rep_not_wiki_list = []

for i in range(100):
    word = injury_rep_not_wiki.pop()
    injury_rep_not_wiki_list.append(word)
    injury_rep_not_wiki.add(word)
```

In [56]:
```
injury_rep_not_wiki_list[:15]
```
Out[56]:
```
['rwc',
 'doctoring',
 'jersy',
 'unty',
 'lanced',
 's.c',
 '4066',
 '17l',
 '9:55am',
 "20'wide",
 'grounted',
 'sx155',
 'slede',
 "160'long",
 'tach']
```

All words that appear in the IMDB vocab, but not the wikitext-103 vocab, will be initialized to the same random vector in a model. As the model trains, we will learn these weights.

File failed to load: /extensions/MathZoom.js

In [57]:
```
vocab.stoi["modernisation"]
```
Out[57]:
```
0
```

In [58]:
```
"modernisation" in wiki_words
```
Out[58]:
```
True
```

In [59]:
```
vocab.stoi["excavator"]
```
Out[59]:
```
673
```

Generating fake injury reports (using wiki-text model)

In [48]:
```
TEXT = "He injured his knee"
N_WORDS = 50
N_SENTENCES = 2
```

In [49]:
```
print("\n\n".join(learn_lm.predict(TEXT, N_WORDS, temperature=0.75) for _ in range(N_SENTENCES)))
```
```
He injured his knee when he stepped off a lowboy trailer . He was on the bottom step to get a welding rod . He
then slipped and fell to the ground . He did not go to a doctor until 3 / 23 / 07 . xxbos Employee was

He injured his knee while crawling in the pit area . He was taken to the hospital and an MRI was performed and
a MRI was performed and results were received on 5 / 9 / 10 . a small tear of the meniscus was found and the d
octor had to
```

In [50]:
```
TEXT = "Working on the ladder,"
N_WORDS = 50
N_SENTENCES = 2
```

In [51]:
```
print("\n\n".join(learn_lm.predict(TEXT, N_WORDS, temperature=0.75) for _ in range(N_SENTENCES)))
```
```
Working on the ladder, employee reached for a wrench when the ladder slipped , catching hand between ladder &
ladder . xxbos Individual was installing a rub rail on the continuous miner , when a piece of rail came off of
the miner and struck employee in the right hand . The

Working on the ladder, a pry bar fell and struck him on the right side of his head . Required stitches . xxbos
Employee was working at the AB Plant plant on the 8th floor , when he stepped on a piece of concrete and twist
ed his right ankle .
```

In [52]:
```
doc(LanguageLearner.predict)
```

Lowering `temperature` will make the texts less randomized.

In [53]:
```
print("\n".join(learn_lm.predict(TEXT, N_WORDS, temperature=0.10) for _ in range(N_SENTENCES)))
```
```
Working on the ladder, the ladder slipped and he fell to the ground . xxbos Employee was walking down the stai
rs when he slipped and fell . He was taken to the ER and taken off work . Currently ONGOING . xxbos Employee w
as walking down the stairs
Working on the ladder, employee was climbing down ladder and slipped and fell to the ground . xxbos Employee w
as walking down the stairs at the Prep Plant when he slipped and fell . He landed on his left shoulder . He wa
s taken to the ER and
```

In [54]:
```
doc(LanguageLearner.predict)
```

In [55]:
```
print("\n".join(learn_lm.predict(TEXT, N_WORDS, temperature=0.10) for _ in range(N_SENTENCES)))
```
```
Working on the ladder, employee was climbing down ladder and slipped and fell to the ground . xxbos Employee w
as walking down the stairs from the Control Room when he slipped and fell . He was taken to the ER and was dia
gnosed with a fractured rib .
Working on the ladder, employee was climbing down the ladder and slipped and fell to the ground . xxbos Employ
ee was walking down the stairs from the Control Room and slipped on the second step . He caught himself with h
is right arm and felt a pop in his right
```

Training the model

Now, we want to choose a good learning rate.

In [69]:
```
path = Path('.')
data_lm = load_data(path, 'lm_databunch', bs=bs)
learn_lm = language_model_learner(data_lm, AWD_LSTM, drop_mult=0.3)
```

File failed to load: /extensions/MathZoom.js

In [71]:
```
learn_lm.lr_find()
```

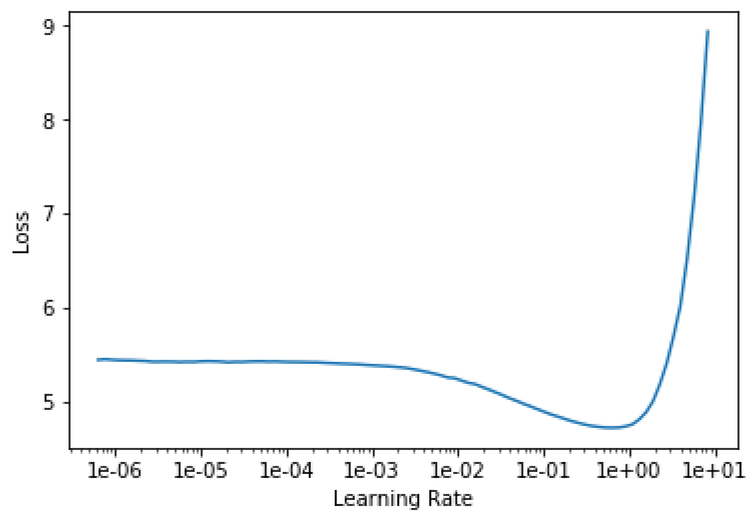
0.00% [0/1 00:00

epoch	train_loss	valid_loss	accuracy	time

17.34% [99/571 00:27

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

In [72]:
```
learn_lm.recorder.plot(skip_end=0)
```



In [73]:
```
lr = 1e-3
lr *= bs/48
```

In [74]:
```
learn_lm.to_fp16();
```

In [75]:
```
learn_lm.fit_one_cycle(1, lr*10, moms=(0.8,0.7))
```

epoch	train_loss	valid_loss	accuracy	time
0	2.840113	2.696728	0.457658	03:14

In [42]:
```
# 1e-4
# learn_lm.fit_one_cycle(1, lr*10, moms=(0.8,0.7))
```

epoch	train_loss	valid_loss	accuracy	time
0	3.006438	2.872717	0.438396	04:30

In [38]:
```
# 1e-3
# learn_lm.fit_one_cycle(1, lr*10, moms=(0.8,0.7))
```

epoch	train_loss	valid_loss	accuracy	time
0	2.861306	2.694536	0.457180	04:31

In [34]:
```
# 1e-2
# learn_lm.fit_one_cycle(1, lr*10, moms=(0.8,0.7))
```

epoch	train_loss	valid_loss	accuracy	time
0	3.365360	3.062014	0.423934	04:30

In [18]:
```
# 1e-1
# learn_lm.fit_one_cycle(1, lr*10, moms=(0.8,0.7))
```

epoch	train_loss	valid_loss	accuracy	time
0	4.587854	3.789413	0.354973	04:28

In [154]:
```
lr = 1e-3
epoch   train_loss   valid_loss   accuracy   time
0   2.979160    2.783141    0.449867    04:26
```

```
  File "<ipython-input-154-2a5e65573adf>", line 2
    epoch       train_loss      valid_loss      accuracy        time
                                ^
SyntaxError: invalid syntax
```

Since this is relatively slow to train, we will save our weights:

In [76]:
```
learn_lm.save('fit_1')
```

File failed to load: /extensions/MathZoom.js

In [77]:
```
learn_lm.load('fit_1');
```

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

In [78]:
```
learn_lm.unfreeze()
```

In [79]:
```
learn_lm.fit_one_cycle(6, lr, moms=(0.8,0.7))
```

epoch	train_loss	valid_loss	accuracy	time
0	2.512394	2.454537	0.488213	03:53
1	2.402684	2.376316	0.497396	03:54
2	2.300067	2.312649	0.506347	03:54
3	2.165342	2.277514	0.512368	03:54
4	2.026565	2.272043	0.514686	03:53
5	1.922111	2.286821	0.514122	03:55

In [80]:
```
learn_lm.save('fine_tuned')
```

We have to save not just 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 [81]:
```
learn_lm.save_encoder('fine_tuned_enc')
```

Loading our saved weights

In [82]:
```
learn_lm.load('fine_tuned');
```

Now that we've trained our model, different representations have been learned for the words that were in IMDB but not wiki (remember that at the beginning we had initialized them all to the same thing):

In [83]:
```
enc = learn_lm.model[0].encoder
```

In [86]:
```
# np.allclose(enc.weight[vocab.stoi["30-something"], :],
#             enc.weight[vocab.stoi["linklater"], :])
```

In [87]:
```
# np.allclose(enc.weight[vocab.stoi["30-something"], :], new_word_vec)
```

More generated movie reviews

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

In [88]:
```
TEXT = "Working on the ladder,"
N_WORDS = 50
N_SENTENCES = 2
```

In [89]:
```
print("\n\n".join(learn_lm.predict(TEXT, N_WORDS, temperature=0.75) for _ in range(N_SENTENCES)))
```

```
Working on the ladder, a fitting broke , causing him to fall and twist his knee . xxbos Employee was working o
n the # 28 mill with air bags when a cement filter fell off the ladder and struck the employee in the ankle .
xxbos Employee was operating a Cat

Working on the ladder, he was pulling the hose on the truck , and slipped in the mud , twisting his knee . Beg
an losing time 5 - 7 - 06 . xxbos The employee was cutting a piece of belt off to make a splice . When the bel
t knife
```

In [62]:
```
TEXT = "I injured my knee"
N_WORDS = 50
N_SENTENCES = 2
```

In [63]:
```
print("\n\n".join(learn_lm.predict(TEXT, N_WORDS, temperature=0.75) for _ in range(N_SENTENCES)))
```

```
I injured my knee when i stepped on a piece of rock . xxbos Employee was removing shaft from pulley when it sl
ipped and caught his thumb between shaft and shaft . xxbos Employee is picking up a granite block and felt a p
ain in his lower back . xxbos While

I injured my knee with a grease gun . i was working on the wash plant and i was moving the water hose up and o
ver the side of my foot . xxbos EE was removing the bearing on the shaft of the cone crusher . He had a pry ba
r on
```

In [64]:
```
TEXT = "Employee stepped on"
N_WORDS = 50
N_SENTENCES = 2
```

File failed to load: /extensions/MathZoom.js

In [65]:
```
print("\n\n".join(learn_lm.predict(TEXT, N_WORDS, temperature=0.75) for _ in range(N_SENTENCES)))
```

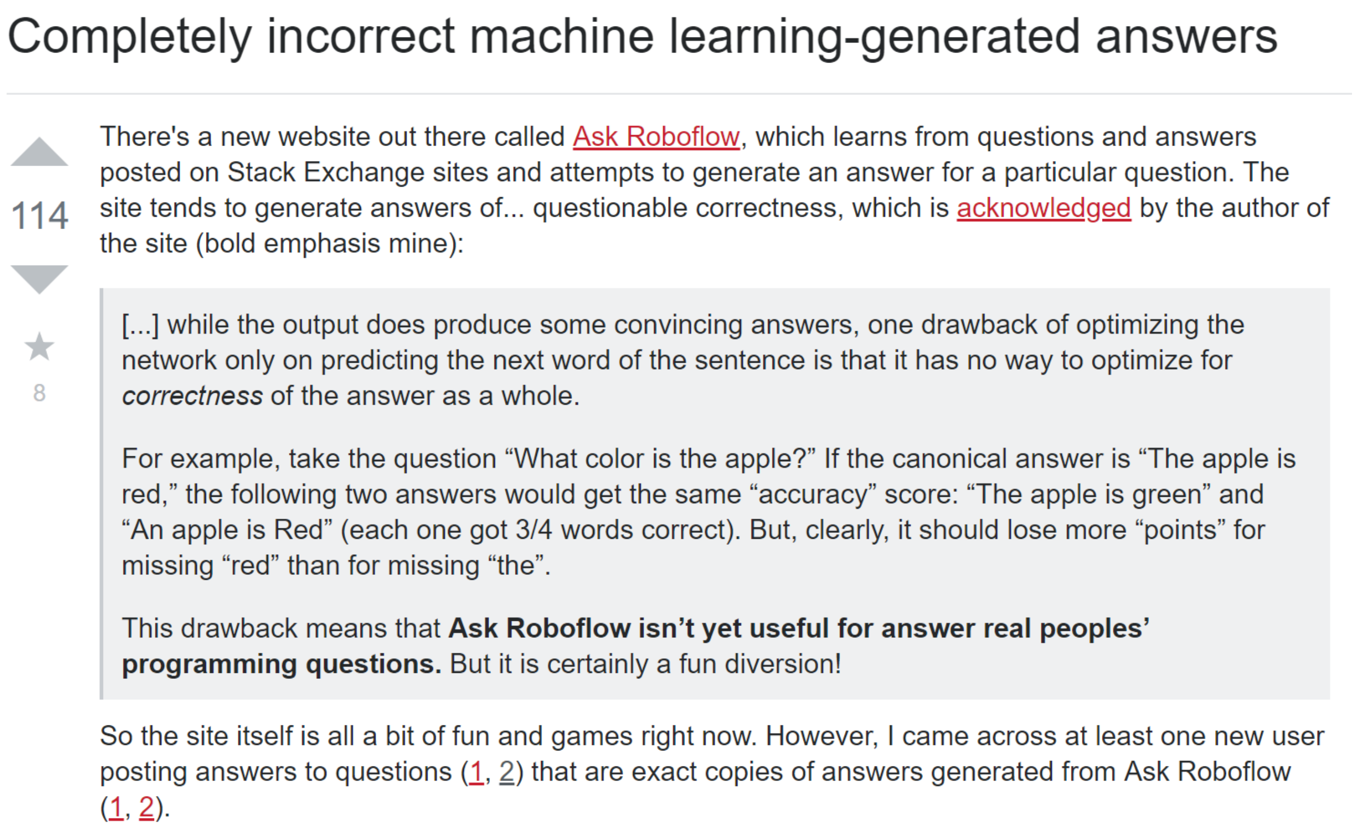
```
Employee stepped on a rock while exiting a pick - up truck and sprained / strained his back xxbos EE WAS REMOV
ING a PIECE OF ROCK FROM THE CONTINUOUS MINER IN THE PILLAR LINE WHEN HE

Employee stepped on loose stone and sprained his right ankle on a rock . xxbos Employee was struck in the mout
h by a steel cross member which fell from approximately 5 feet up onto a conveyor frame when the jack hammer h
andle was struck by a hammer . * * *
```

Risks of language models

We will talk about ethical concerns raised by very accurate language models in lesson 7, but here are a few brief notes:

In reference to OpenAI's GPT-2 (https://www.theverge.com/2019/2/14/18224704/ai-machine-learning-language-models-read-write-openai-gpt2): Jeremy Howard said, *I've been trying to warn people about this for a while. We have the technology to totally fill Twitter, email, and the web up with reasonable-sounding, context-appropriate prose, which would drown out all other speech and be impossible to filter.*

For a small example, consider when completely incorrect (but reasonable sounding) ML generated answers were posted to StackOverflow (https://meta.stackoverflow.com/questions/384596/completely-incorrect-machine-learning-generated-answers?stw=2):


Completely incorrect machine learning-generated answers

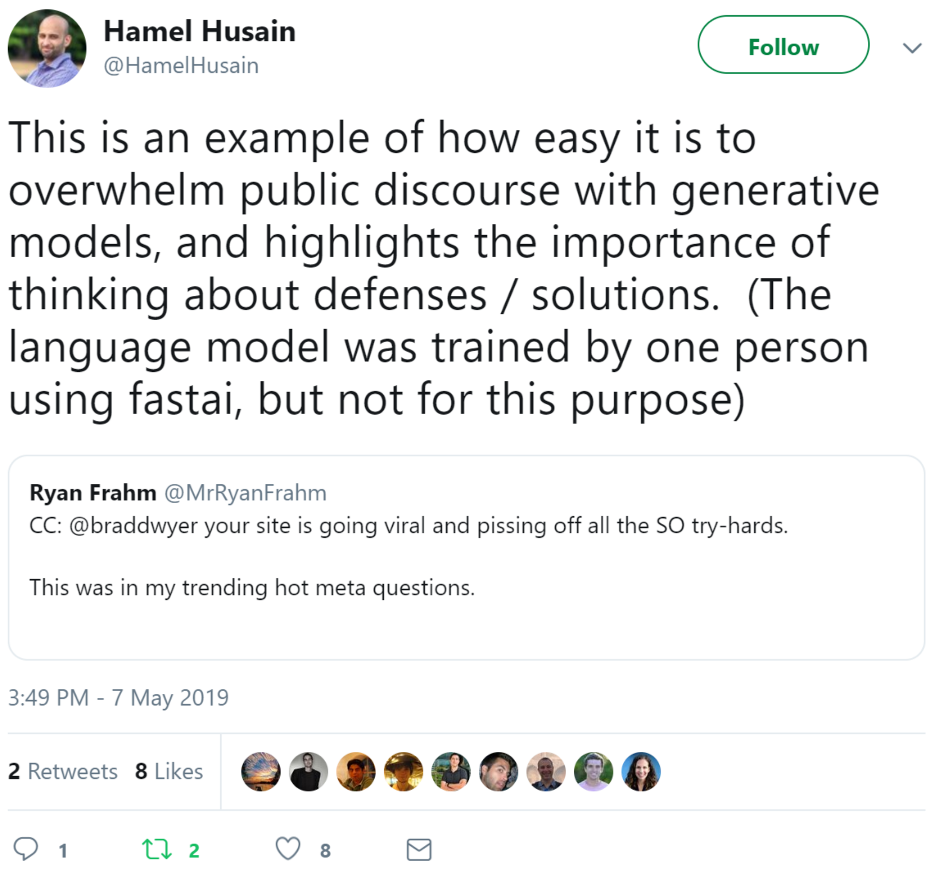
114

8

There's a new website out there called Ask Roboflow, which learns from questions and answers posted on Stack Exchange sites and attempts to generate an answer for a particular question. The site tends to generate answers of... questionable correctness, which is acknowledged by the author of the site (bold emphasis mine):

> [...] while the output does produce some convincing answers, one drawback of optimizing the network only on predicting the next word of the sentence is that it has no way to optimize for *correctness* of the answer as a whole.
>
> For example, take the question "What color is the apple?" If the canonical answer is "The apple is red," the following two answers would get the same "accuracy" score: "The apple is green" and "An apple is Red" (each one got 3/4 words correct). But, clearly, it should lose more "points" for missing "red" than for missing "the".
>
> This drawback means that **Ask Roboflow isn't yet useful for answer real peoples' programming questions.** But it is certainly a fun diversion!

So the site itself is all a bit of fun and games right now. However, I came across at least one new user posting answers to questions (1, 2) that are exact copies of answers generated from Ask Roboflow (1, 2).


Hamel Husain
@HamelHusain
Follow

This is an example of how easy it is to overwhelm public discourse with generative models, and highlights the importance of thinking about defenses / solutions. (The language model was trained by one person using fastai, but not for this purpose)

Ryan Frahm @MrRyanFrahm
CC: @braddwyer your site is going viral and pissing off all the SO try-hards.

This was in my trending hot meta questions.

3:49 PM - 7 May 2019

2 Retweets 8 Likes

1 2 8

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 [8]:
```
bs=48
```

File failed to load: /extensions/MathZoom.js

In [100]:
```
doc(TextClasDataBunch.from_csv)
```

In [9]:
```
# 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' so no need to filter)
#              .label_from_folder(classes=['neg', 'pos'])
#              #label them all with their folders
#              .databunch(bs=bs, num_workers=1))
```

In [101]:
```
# Classifier model data
data_clas = TextClasDataBunch.from_csv(path, 'accident_dataset.csv', vocab=data_lm.train_ds.vocab, bs=bs,
                                       label_cols='days_lost')
```

In [102]:
```
data_clas.save('injury_textlist_class')
```

In [103]:
```
data_clas = load_data(path, 'injury_textlist_class', bs=bs, num_workers=1)
```

In [104]:
```
data_clas.show_batch()
```

text	target
xxbos xxup employee xxup called 3 / 7 / 03 xxup to xxup say xxup he xxup had a xxup sharp xxup pain xxup in xxup right xxup thigh . xxup he xxup says xxup he xxup may xxup have xxup hurt xxup it xxup by xxup walking xxup the \ xxup wrong xxup way " xxup on xxup the xxup mud xxup belt xxup on 3 / 5 /	True
xxbos xxup ee xxup states , xxup he xxup was xxup stacking 50 # xxup bags xxup of xxup clay xxup product xxup on a xxup pallet xxup and xxup picked xxup one xxup of xxup the xxup bags xxup up xxup wrong . xxup he xxup states xxup he xxup felt a xxup sharp xxup pain xxup in xxup his xxup lower xxup back . xxup it xxup went	True
xxbos xxup ee xxup was xxup asked xxup to xxup remove xxup the xxup drive xxup line xxup on a xxup pick xxup up xxup that xxup was xxup needing xxup xxunk . xxup he xxup jacked xxup the xxup pickup , xxup placing xxup blocks xxup behind xxup and xxup in xxup front xxup of xxup the xxup rear xxup wheels . xxup he xxup did xxup not xxup	True
xxbos xxup ee xxup went xxup to xxup dr xxup on 3 / 31 / 03 xxup with a xxup complaint xxup of xxup pain xxup in xxup his xxup left xxup rib - xxup cage xxup area . x - xxup rays xxup were xxup taken & xxup he xxup was xxup referred xxup to xxup another xxup dr xxup who xxup ordered xxup more xxup tests . xxup	False
xxbos xxup ee xxup injured xxup his xxup shoulder xxup moving a xxup spool xxup of xxup cable xxup on 1 / 2 / 04 . xxup he xxup did xxup not xxup seek xxup medical xxup attention xxup until 2 / 17 / 04 . xxup he xxup con't xxup to xxup work xxup to xxup this xxup date . xxup due xxup to xxup cont'd xxup pain xxup	False

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

In [124]:
```
learn_c = text_classifier_learner(data_clas, AWD_LSTM, drop_mult=0.3) #.to_fp16()
learn_c.load_encoder('fine_tuned_enc')
learn_c.freeze()
```

In [106]:
```
learn_c.lr_find()
```

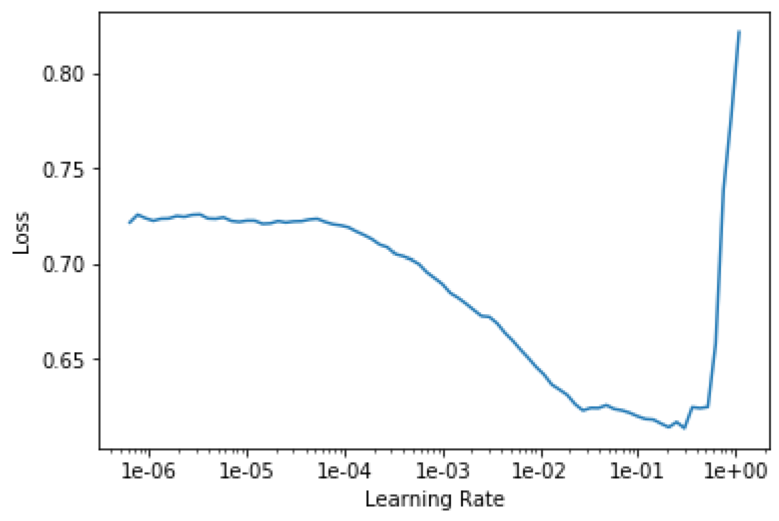
0.00% [0/1 00:00

epoch	train_loss	valid_loss	accuracy	time

11.74% [93/792 00:12

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

In [107]:
```
learn_c.recorder.plot()
```



In [125]:
```
learn_c.fit_one_cycle(1, 1e-2, moms=(0.8,0.7))
```

epoch	train_loss	valid_loss	accuracy	time
0	0.490319	0.470048	0.764690	01:17

In [ ]:
```
# 2e-2
#0  0.488632    0.472631    0.762509    01:17
```

File failed to load: /extensions/MathZoom.js

In [ ]:
```
# 7e-2
# 0 0.487383    0.477987    0.756728    01:19
```

In [ ]:
```
# 2e-3
# 0 0.479893    0.473694    0.761878    01:11
```

In [ ]:
```
# 7e-3
# 0 0.487481    0.472954    0.760249    01:19
```

In [ ]:
```
# 2e-4
# 0 0.497973    0.493928    0.749790    01:12
```

In [ ]:
```
# 1e-2
# 0 0.474412    0.471605    0.764033    01:10
```

In [ ]:
```
# 2e-1
0   0.505911    0.487831    0.750946    01:20
```

In [ ]:
```
# 7e-1
# 0 0.520080    0.511026    0.747924    01:13
```

In [126]:
```
learn_c.save('first')
```

In [127]:
```
learn_c.load('first');
```

In [128]:
```
learn_c.freeze_to(-2)
learn_c.fit_one_cycle(1, slice(1e-2/(2.6**4),1e-2), moms=(0.8,0.7))
```

epoch	train_loss	valid_loss	accuracy	time
0	0.447313	0.443485	0.781720	01:21

In [129]:
```
learn_c.save('2nd')
```

In [130]:
```
learn_c.freeze_to(-3)
learn_c.fit_one_cycle(1, slice(5e-3/(2.6**4),5e-3), moms=(0.8,0.7))
```

epoch	train_loss	valid_loss	accuracy	time
0	0.440746	0.434366	0.787712	02:26

In [132]:
```
learn_c.save('3rd')
```

In [133]:
```
learn_c.unfreeze()
learn_c.fit_one_cycle(2, slice(1e-3/(2.6**4),1e-3), moms=(0.8,0.7))
```

epoch	train_loss	valid_loss	accuracy	time
0	0.388018	0.440412	0.787344	02:38
1	0.392706	0.444074	0.788132	02:40

In [134]:
```
learn_c.save('clas')
```

In [2]:
```
learn_c.predict("Employee fell of the 3 ladder and broke his leg. He spent three days in the hospital.")
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-2-fedc2db86e49> in <module>
----> 1 learn_c.predict("Employee fell of the 3 ft ladder and broke his leg. He spent three days in the hospit
al.")

NameError: name 'learn_c' is not defined
```

In [136]:
```
learn_c.predict("Employee tripped on a rock and bruised his arm. He returned to work that afternoon")
```

Out[136]:
```
(Category tensor(0), tensor(0), tensor([0.9644, 0.0356]))
```

In [ ]:
```

```

File failed to load: /extensions/MathZoom.js