```
#hide
! [ -e /content ] && pip install -Uqq fastbook
import fastbook
fastbook.setup_book()
```

```
#hide
from fastbook import *
from IPython.display import display,HTML
```

NLP Deep Dive: RNNs

Text Preprocessing

Tokenization

+ Código + Texto

Word Tokenization with fastai

```
from fastai.text.all import *
path = untar_data(URLs.IMDB)
```

```
files = get_text_files(path, folders = ['train', 'test', 'unsup'])
```

```
txt = files[0].open().read(); txt[:75]
```

```
'I never really started watching the show until it was canceled and started '
```

```
spacy = WordTokenizer()
toks = first(spacy([txt]))
print(coll_repr(toks, 30))
```

```
(#286) ['I','never','really','started','watching','the','show','until','it','was','canceled','and','started','showing','re','-','ru
```

```
first(spacy(['The U.S. dollar $1 is $1.00.']))
```

```
(#9) ['The','U.S.','dollar','$','1','is','$','1.00','.']
```

```
tkn = Tokenizer(spacy)
print(coll_repr(tkn(txt), 31))
```

```
(#313) ['xxbos','i','never','really','started','watching','the','show','until','it','was','canceled','and','started','showing','re'
```

```
defaults.text_proc_rules
```

```
[<function fastai.text.core.fix_html(x)>,
 <function fastai.text.core.replace_rep(t)>,
 <function fastai.text.core.replace_wrep(t)>,
 <function fastai.text.core.spec_add_spaces(t)>,
 <function fastai.text.core.rm_useless_spaces(t)>,
 <function fastai.text.core.replace_all_caps(t)>,
 <function fastai.text.core.replace_maj(t)>,
 <function fastai.text.core.lowercase(t, add_bos=True, add_eos=False)>]
```

```
coll_repr(tkn('&copy;   Fast.ai www.fast.ai/INDEX'), 31)
```

```
'(#11) ['xxbos','©','xxmaj','fast.ai','xxrep','3','w','.fast.ai','/','xxup','index']'
```

Subword Tokenization

```
txts = L(o.open(encoding="utf8").read() for o in files[:2000])
```

```
def subword(sz):
    sp = SubwordTokenizer(vocab_sz=sz)
    sp.setup([t.encode('utf-8') for t in txts])
    return ' '.join([tok.decode() for tok in first(sp([txt.encode('utf-8')]))[:40]])
```

```
subword(1000)
```

```
subword(200)
```

```
subword(10000)
```

Numericalization with fastai

```
toks = tkn(txt)
print(coll_repr(tkn(txt), 31))
```

```
toks200 = txts[:200].map(tkn)
toks200[0]
```

```
num = Numericalize()
num.setup(toks200)
coll_repr(num.vocab,20)
```

```
nums = num(toks)[:20]; nums
```

```
' '.join(num.vocab[o] for o in nums)
```

Putting Our Texts into Batches for a Language Model

```
stream = "In this chapter, we will go back over the example of classifying movie reviews we studied in chapter 1 and dig deeper under the
tokens = tkn(stream)
bs,seq_len = 6,15
d_tokens = np.array([tokens[i*seq_len:(i+1)*seq_len] for i in range(bs)])
df = pd.DataFrame(d_tokens)
display(HTML(df.to_html(index=False,header=None)))
```

```
bs,seq_len = 6,5
d_tokens = np.array([tokens[i*15:i*15+seq_len] for i in range(bs)])
df = pd.DataFrame(d_tokens)
display(HTML(df.to_html(index=False,header=None)))
```

```
bs,seq_len = 6,5
d_tokens = np.array([tokens[i*15+seq_len:i*15+2*seq_len] for i in range(bs)])
df = pd.DataFrame(d_tokens)
display(HTML(df.to_html(index=False,header=None)))
```

```
bs,seq_len = 6,5
d_tokens = np.array([tokens[i*15+10:i*15+15] for i in range(bs)])
df = pd.DataFrame(d_tokens)
display(HTML(df.to_html(index=False,header=None)))
```

```
nums200 = toks200.map(num)
```

```
dl = LMDataLoader(nums200)
```

```
x,y = first(dl)
x.shape,y.shape
```

```
' '.join(num.vocab[o] for o in x[0][:20])
```

```
' '.join(num.vocab[o] for o in y[0][:20])
```

Training a Text Classifier

Language Model Using DataBlock

```
get_imdb = partial(get_text_files, folders=['train', 'test', 'unsup'])

dls_lm = DataBlock(
    blocks=TextBlock.from_folder(path, is_lm=True),
    get_items=get_imdb, splitter=RandomSplitter(0.1)
).dataloaders(path, path=path, bs=128, seq_len=80)
```

```
dls_lm.show_batch(max_n=2)
```

Fine-Tuning the Language Model

```
learn = language_model_learner(
    dls_lm, AWD_LSTM, drop_mult=0.3,
    metrics=[accuracy, Perplexity()]).to_fp16()
```

```
learn.fit_one_cycle(1, 2e-2)
```

Saving and Loading Models

```
learn.save('1epoch')
```

```
learn = learn.load('1epoch')
```

```
learn.unfreeze()
learn.fit_one_cycle(10, 2e-3)
```

```
learn.save_encoder('finetuned')
```

Text Generation

```
learn.load(gdrive/'finetuned-lm')
```

```
TEXT = "I liked this movie because"
N_WORDS = 40
N_SENTENCES = 2
preds = [learn.predict(TEXT, N_WORDS, temperature=0.75)
         for _ in range(N_SENTENCES)]
```

```
print("\n".join(preds))
```

Creating the Classifier DataLoaders

```
dls_clas = DataBlock(
    blocks=(TextBlock.from_folder(path, vocab=dls_lm.vocab),CategoryBlock),
    get_y = parent_label,
    get_items=partial(get_text_files, folders=['train', 'test']),
    splitter=GrandparentSplitter(valid_name='test')
).dataloaders(path, path=path, bs=128, seq_len=72)
```

```
dls_clas.show_batch(max_n=3)
```

```
nums_samp = toks200[:10].map(num)
```

```
nums_samp.map(len)
```

```
learn = text_classifier_learner(dls_clas, AWD_LSTM, drop_mult=0.5,
                                metrics=accuracy).to_fp16()
```

```
learn = learn.load_encoder(gdrive/'finetuned')
```

Fine-Tuning the Classifier

```
learn.fit_one_cycle(1, 2e-2)
```

```
learn.freeze_to(-2)
learn.fit_one_cycle(1, slice(1e-2/(2.6**4),1e-2))
```

```
learn.freeze_to(-3)
learn.fit_one_cycle(1, slice(5e-3/(2.6**4),5e-3))
```

```
learn.unfreeze()
learn.fit_one_cycle(2, slice(1e-3/(2.6**4),1e-3))
```

Disinformation and Language Models

Conclusion

Questionnaire

1. What is "self-supervised learning"?
2. What is a "language model"?
3. Why is a language model considered self-supervised?
4. What are self-supervised models usually used for?
5. Why do we fine-tune language models?
6. What are the three steps to create a state-of-the-art text classifier?
7. How do the 50,000 unlabeled movie reviews help us create a better text classifier for the IMDb dataset?
8. What are the three steps to prepare your data for a language model?
9. What is "tokenization"? Why do we need it?
10. Name three different approaches to tokenization.
11. What is `xxbos`?
12. List four rules that fastai applies to text during tokenization.
13. Why are repeated characters replaced with a token showing the number of repetitions and the character that's repeated?
14. What is "numericalization"?
15. Why might there be words that are replaced with the "unknown word" token?
16. With a batch size of 64, the first row of the tensor representing the first batch contains the first 64 tokens for the dataset. What does the second row of that tensor contain? What does the first row of the second batch contain? (Careful—students often get this one wrong! Be sure to check your answer on the book's website.)
17. Why do we need padding for text classification? Why don't we need it for language modeling?
18. What does an embedding matrix for NLP contain? What is its shape?
19. What is "perplexity"?
20. Why do we have to pass the vocabulary of the language model to the classifier data block?
21. What is "gradual unfreezing"?
22. Why is text generation always likely to be ahead of automatic identification of machine-generated texts?

Further Research

1. See what you can learn about language models and disinformation. What are the best language models today? Take a look at some of their outputs. Do you find them convincing? How could a bad actor best use such a model to create conflict and uncertainty?
2. Given the limitation that models are unlikely to be able to consistently recognize machine-generated texts, what other approaches may be needed to handle large-scale disinformation campaigns that leverage deep learning?