Transfer learning for NLP status quo
- Best practice: initialise first layer with pretrained word embeddings
- Recent approaches (McCann et al., 2017; Peters et al., 2018): Pretrained embeddings as fixed features. Peters et al. (2018) is task-specific.
- Why not initialise remaining parameters?
- Dai and Le (2015) first proposed fine-tuning a LM. However: No pretraining. Naive fine-tuning (require millions of in-domain documents).

Universal Language Model Fine-tuning (ULMFit)
3-step recipe for state-of-the-art on any text classification task:
1. Train language model (LM) on general domain data.
2. Fine-tune LM on target data.
3. Train classifier on labeled data.

(a) General-domain LM pretraining
Train LM on a large general domain corpus, e.g. WikiText-103.

(b) Target task LM fine-tuning
Discriminative fine-tuning
Different layers capture different types of information. They should be fine-tuned to different extents with different learning rates:
\[ \theta^l_i = \theta^l_{i-1} - \eta^l \cdot \nabla \theta J(\theta) \]

Slanted triangular learning rates
The model should converge quickly to a suitable region and then refine its parameters.

(c) Target task classifier fine-tuning
Train classification layer on top of LM.

Concat pooling
Concatenate pooled representations of hidden states to capture long document contexts:
\[ h_c = [h_T, \text{maxpool}(H), \text{meanpool}(H)] \]

Gradual unfreezing
Gradually unfreeze the layers starting from the last layer to prevent catastrophic forgetting.

Bidirectional language model
Pretrain both forward and backward LMs and fine-tune them independently.

Experiments
Comparison against state-of-the-art (SOTA) on six widely studied text classification datasets.

Previous SOTA vs. ULMFIT

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Previous SOTA</th>
<th>ULMFIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMDb</td>
<td>5.9</td>
<td>4.6</td>
</tr>
<tr>
<td>TREC-6</td>
<td>5.07</td>
<td>3.6</td>
</tr>
<tr>
<td>AG News</td>
<td>3.2</td>
<td>1.6</td>
</tr>
<tr>
<td>DBpedia</td>
<td>2.9</td>
<td>1.6</td>
</tr>
<tr>
<td>Yelp-bi</td>
<td>2.6</td>
<td>1.6</td>
</tr>
<tr>
<td>Yelp-full</td>
<td>3.9</td>
<td>1.6</td>
</tr>
</tbody>
</table>

Analysis
Low-shot learning
- 100 labeled examples: ULMFit matches performance of training from scratch with 10x and 20x more data (on IMDb and AG News).
- 100 labeled examples + 50-100k unlabeled examples: ULMFit matches performance of training from scratch with 50x and 100x more data (on IMDb and AG News).

Pretraining
- Most useful for small and medium-sized datasets.

LM quality
- Even a vanilla LM can perform well with fine-tuning.

LM fine-tuning
- Most useful for larger datasets.

Classifier fine-tuning
- ULMFit works well across all datasets.

Fine-tuning behaviour
- No catastrophic forgetting. Stable even across a large # of epochs.

Models and code: [http://nlp.fast.ai/ulmfit](http://nlp.fast.ai/ulmfit)