Universal Language Model Fine-tuning for Text Classification

Jeremy Howard, Sebastian Ruder

January, 2018

This talk

- What is the Universal Language Model
- Transfer learning
- Results
- Applications
- Summary

Introduction

Text classification - Applications

1. Finding documents relevant to a legal case
2. Identifying spam, bots, and offensive comments
3. Classifying positive and negative reviews of a product
4. Grouping articles by political orientation

Traditional Approach

While Deep Learning models have achieved state-of-the-art on many NLP tasks, these models are trained from scratch, requiring large datasets, and days to converge.

Transfer learning

1. Transfer learning has had a large impact on computer vision (CV).
2. Applied CV models (including object detection, classification, and segmentation) are rarely trained from scratch, but instead are fine-tuned from models that have been pre-trained on ImageNet, MS-COCO, and other datasets

Transfer learning in NLP

1. In light of the benefits of pre-training in CV we should be able to do better than *randomly initializing* the remaining parameters of our models
2. However, transfer learning via fine-tuning has been unsuccessful for NLP (word embeddings only target the first layer)
3. It's not the idea of LM fine-tuning but our lack of knowledge of how to train them effectively has been hindering wider adoption. LMs overfit to small datasets and suffered catastrophic forgetting when fine-tuned with a classifier.
4. Compared to CV, NLP models are typically more shallow and thus require different fine-tuning methods.

Universal Language Model Fine-tuning (ULMFiT)

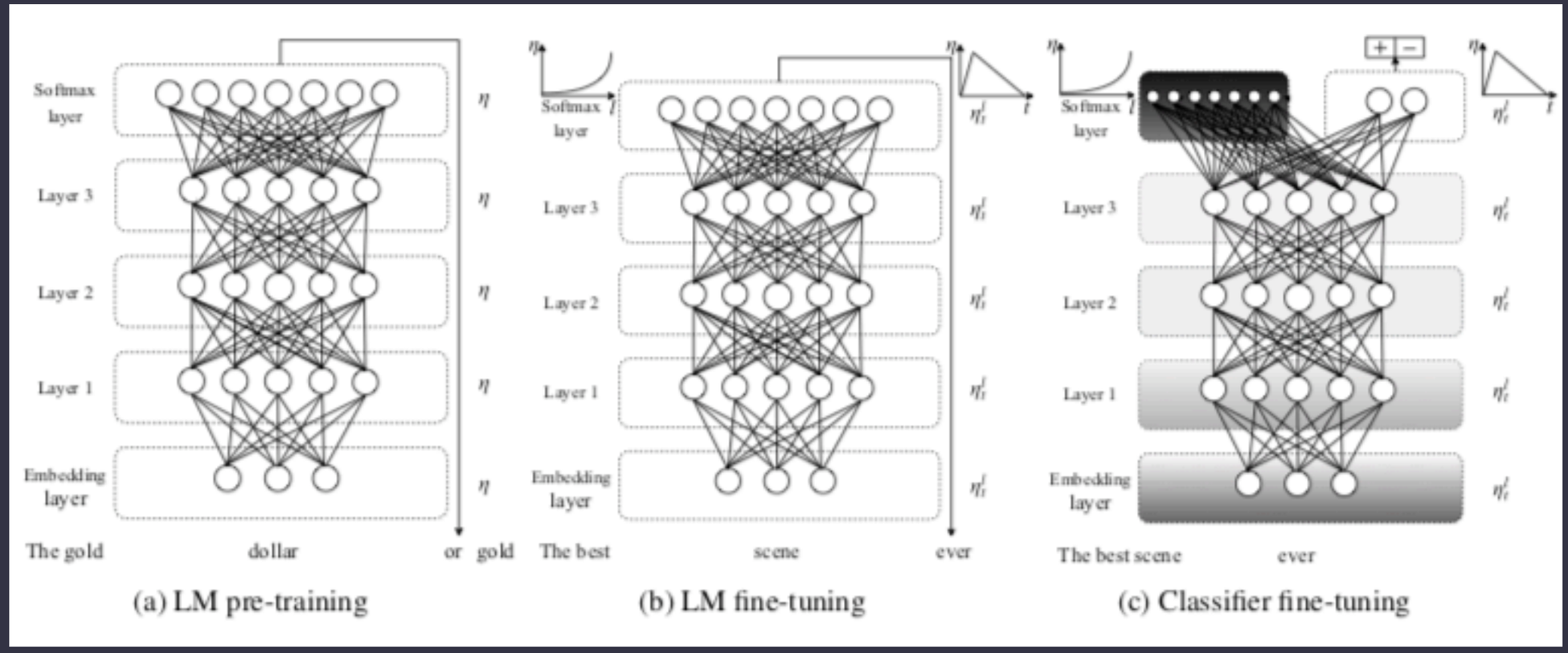
1. Addresses these issues and enables robust inductive transfer learning for any NLP task, similar to fine-tuning ImageNet models
2. 3-layer LSTM architecture - no additions other than tuned dropout hyper-parameters
3. On IMDb, with 100 labeled examples, ULMFiT matches the performance of training from scratch with 10× and—given 50k unlabeled examples—with 100× more data.

ULMFiT - how does it work?

1. Pre-trains a language model (LM) on a large general-domain corpus and fine-tunes it on the target task using novel techniques
2. The method is *universal* in the sense that:
    A. It works across tasks varying in document size, number, and label type
    B. It uses a single architecture and training process
    C. It requires no custom feature engineering or preprocessing
    D. It does not require additional in-domain documents or labels

Universal Language Model Fine-tuning (ULMFiT) -how does it work?

1. General-domain LM pre-training
2. Target task LM fine-tuning
3. Target task classifier fine-tuning



Universal Language Model Fine-tuning (ULMFiT) -how does it work?

1. The LM is trained on a general-domain corpus to capture general features of the language in different layers
2. The full LM is fine-tuned on target task data using discriminative fine-tuning (‘*Discr*’) and slanted triangular learning rates (STLR) to learn task-specific features
3. The classifier is fine-tuned on the target task using gradual unfreezing, ‘*Discr*’, and STLR to preserve low-level representations and adapt high-level ones

Step 1: General-domain LM pre-training

1. An ImageNet-like corpus for language should be large and capture general properties of language.
2. Pre-train the language model on Wikitext-103 consisting of ~29K preprocessed Wikipedia articles and 103M words
3. Pre-training is most beneficial for tasks with small datasets and enables generalization even with 100 labeled examples.
4. While this stage is the most expensive, it only needs to be performed once and improves performance and convergence of downstream models.

Step 2: Target task LM fine-tuning

1. No matter how diverse the general-domain data used for pre-training is, the data of the target task will likely come from a different distribution (i.e. medical, financial)
2. Fine-tune the LM on data of the target task. Given a pre-trained general-domain LM, this stage converges faster as it only needs to adapt to the idiosyncrasies of the target data, and it allows to train a robust LM even for small datasets
3. *Discriminative fine-tuning* and *slanted triangular learning rates* for fine-tuning the LM

Step 2: Discriminative fine-tuning

1. As different layers capture different types of information they should be fine-tuned to *differ- ent extents*
    1. *P*ropose a novel fine- tuning method, discriminative fine-tuning[3]
2. Instead of using the same learning rate for *all* layers of the model, discriminative fine-tuning allows to tune *each* layer with different learning rates
3. $\eta^{l-1} = \eta^{l}/2.6$ (where η^{l} is learning rate of last layer)

Step 2: Slanted triangular learning rates

1. For adapting its parameters to task-specific features, we would like the model to quickly converge to a suitable region of the parameter space in the beginning of training and then refine its parameters.
2. Using the same learning rate (LR) or an annealed learning rate throughout training is not the best way to achieve this behavior.
3. Instead, propose *slanted triangular learning rates* (STLR), which first linearly increases the learning rate and then linearly decays it according to the following update schedule
4. STLR modifies triangular learning rates with a short increase and a long decay period, which we found (empirically?) key for good performance

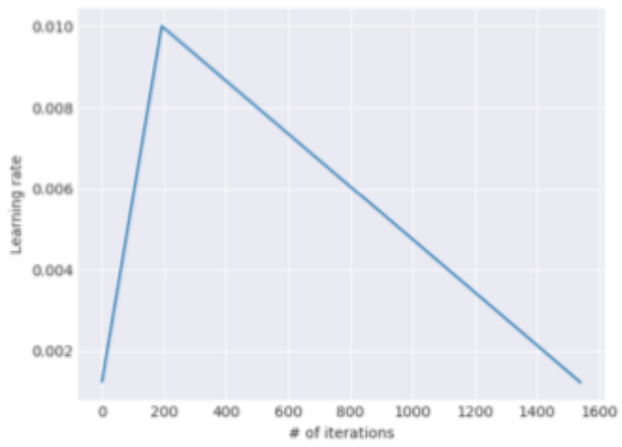


Figure 2: The slanted triangular learning rate schedule used for ULMFiT as a function of the number of training iterations.

Step 3: Target task classifier fine-tuning

1. Finally, for fine-tuning the classifier, we augment the pre-trained language model with two additional linear blocks.
2. Following standard practice for CV classifiers, each block uses batch normalization and dropout, with ReLU activations for the intermediate layer and a softmax activation that outputs a probability distribution over target classes at the last layer.

Experiments

Three common text classification tasks using widely-studied datasets:

1. Sentiment analysis
2. Question classification
3. Topic classification

Dataset	Type	# classes	# examples
TREC-6	Question	6	5.5k
IMDb	Sentiment	2	25k
Yelp-bi	Sentiment	2	560k
Yelp-full	Sentiment	5	650k
AG	Topic	4	120k
DBpedia	Topic	14	560k

Table 1: Text classification datasets and tasks with number of classes and training examples.

Hyperparameters

1. AWD-LSTM language model with an embedding size of 400
2. 3 layers, 1150 hidden activations per layer
3. Dropout of 0.4 to layers, 0.3 to RNN layers, 0.4 to input embedding layers, 0.05 to embedding layers, and weight dropout of 0.5 to the RNN hidden-to-hidden matrix.
4. The classifier has a hidden layer of size 50. Batch size of 64, a base learning rate of 0.004 and 0.01 for fine-tuning the LM and the classifier respectively, and tune the number of epochs on the validation set of each task

Results

	Model	Test		Model	Test
IMDb	CoVe (McCann et al., 2017)	8.2	TREC-6	CoVe (McCann et al., 2017)	4.2
	oh-LSTM (Johnson and Zhang, 2016)	5.9		TBCNN (Mou et al., 2015)	4.0
	Virtual (Miyato et al., 2016)	5.9		LSTM-CNN (Zhou et al., 2016)	3.9
	ULMFiT (ours)	**4.6**		ULMFiT (ours)	**3.6**

Table 2: Test error rates (%) on two text classification datasets used by McCann et al. (2017).

	AG	DBpedia	Yelp-bi	Yelp-full
Char-level CNN (Zhang et al., 2015)	9.51	1.55	4.88	37.95
CNN (Johnson and Zhang, 2016)	6.57	0.84	2.90	32.39
DPCNN (Johnson and Zhang, 2017)	6.87	0.88	2.64	30.58
ULMFiT (ours)	**5.01**	**0.80**	**2.16**	**29.98**

Table 3: Test error rates (%) on text classification datasets used by Johnson and Zhang (2017).

1. Better results compared to complex architectures, attention (McCann et al), sophisticated embedding schemes (Johnson & Zhang)
2. Relatively simple architecture- regular LSTM with dropout

Analysis: Impact of pre-training

1. Compare using no pre-training with pre-training on WikiText-103.
2. Pre-training is most useful for small and medium-sized datasets, which are most common in commercial applications. However, even for large datasets, pre-training improves performance.

Pretraining	IMDb	TREC-6	AG
Without pretraining	5.63	10.67	5.52
With pretraining	**5.00**	**5.69**	**5.38**

Table 4: Validation error rates for ULMFiT with and without pretraining.

Analysis: Impact of LM quality

1. In order to gauge the importance of choosing an appropriate LM, we compare a vanilla LM with the same hyperparameters **without any dropout**[8] with the AWD-LSTM LM with **tuned dropout parameters**.
2. Using fine-tuning techniques, even a regular LM reaches surprisingly good performance on the larger datasets.
3. On the smaller TREC-6, a vanilla LM without dropout runs the risk of overfitting, which decreases performance.

LM	IMDb	TREC-6	AG
Vanilla LM	5.98	7.41	5.76
AWD-LSTM LM	**5.00**	**5.69**	**5.38**

Table 5: Validation error rates for ULMFiT with a vanilla LM and the AWD-LSTM LM.

Discussion

1. NLP for non-English languages, where training data for supervised pre-training is scarce
2. New NLP tasks where no state-of-the-art architecture exists
3. Tasks with limited amounts of labelled data (and some of amounts of unlabeled data)

Conclusions

1. ULMFiT is an effective and extremely sample-efficient transfer learning method that can be applied to many NLP tasks
2. Proposed several novel fine-tuning techniques that in conjunction prevent catastrophic forgetting and enable robust learning across a diverse range of tasks
3. Method significantly outperformed existing transfer learning techniques and the state-of-the-art on six representative text classification tasks.