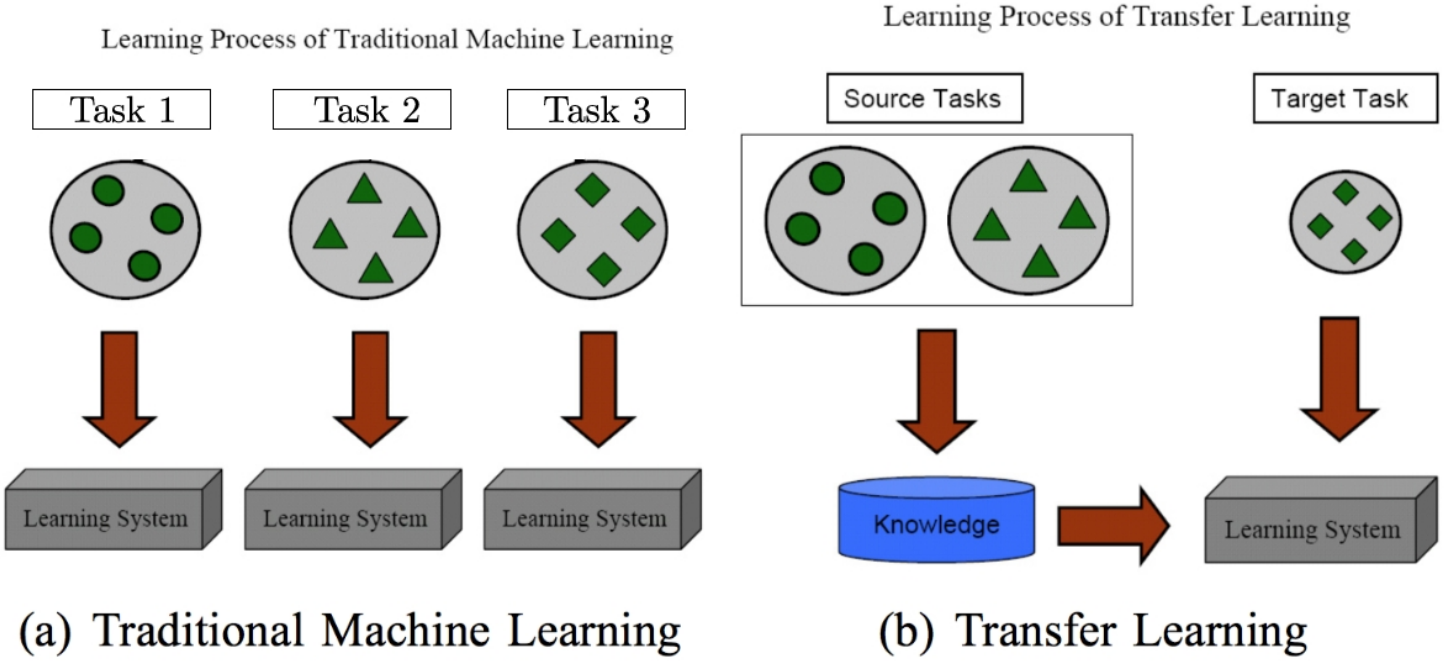
Transfer Learning in NLP (ULMFiT)

Adrian Brasoveanu, 11/14/2019

[based on slides by Sebastian Ruder]

What is transfer learning?



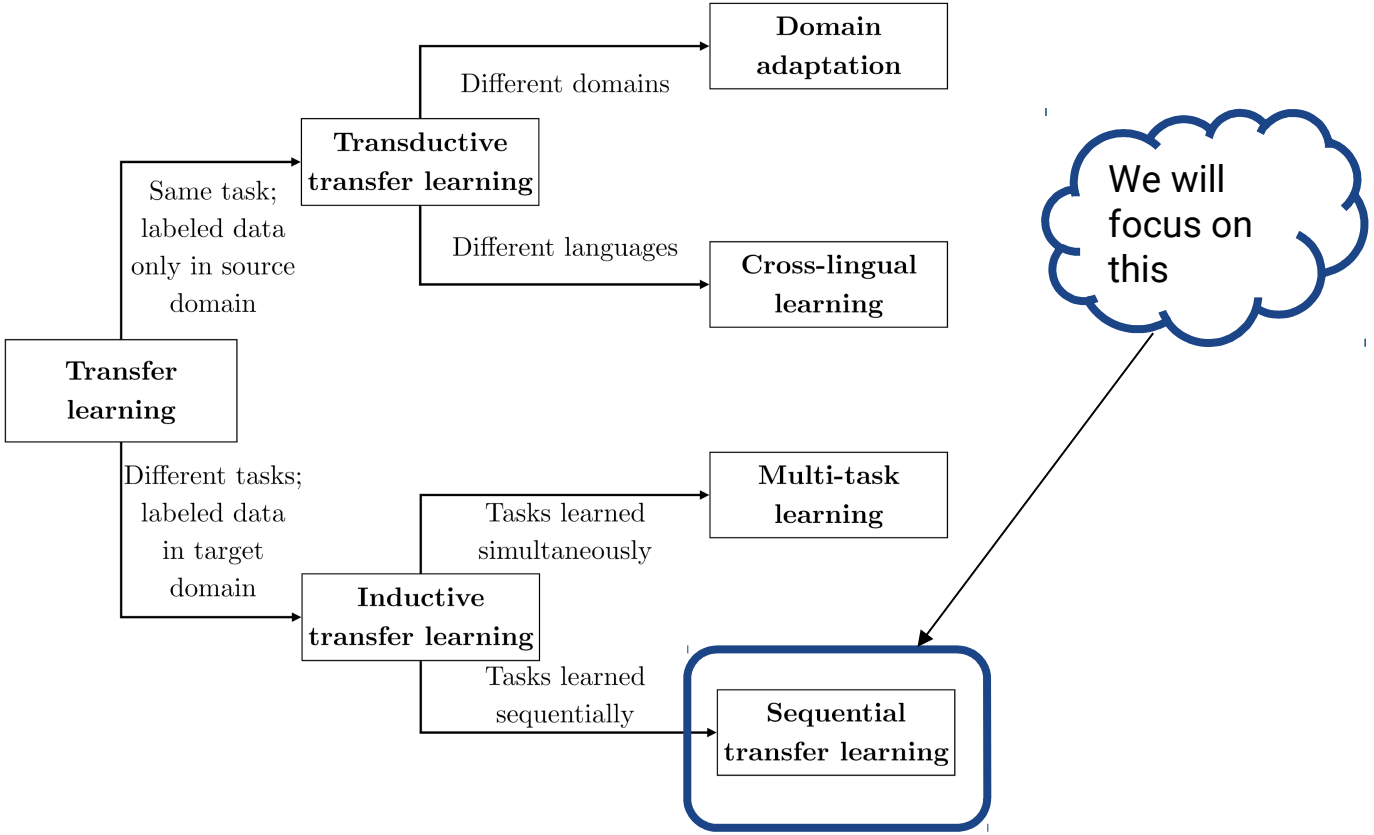
Pan and Yang (2010)

Why transfer learning in NLP?

- ❑ Many NLP tasks share common knowledge about language, e.g., cooccurence/distributional similarities, meaning representations
- ❑ Tasks can inform each other, e.g., building a language model for English can help with subsequent text classification

- ❑ Annotated data is (relatively) rare, unlabeled data is (usually) not; leverage unlabeled data to extract as much information as possible from annotated data
- ❑ Empirically, transfer learning has resulted in SOTA for many supervised NLP tasks (classification, Q&A etc.) – see GLUE and Super-GLUE tasks
  - https://super.gluebenchmark.com/leaderboard

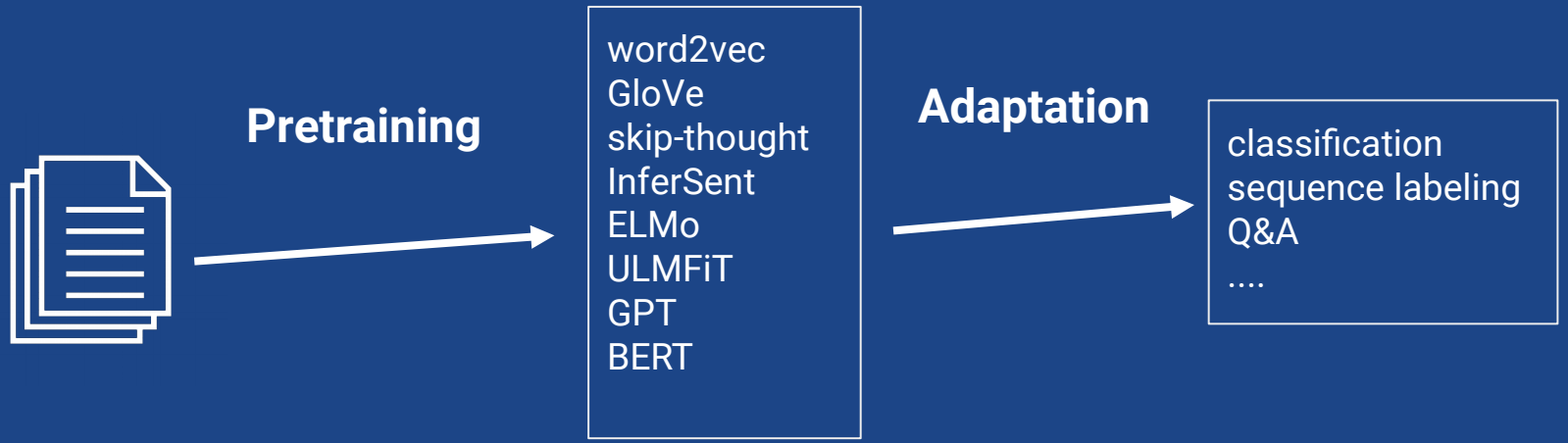
Types of transfer learning in NLP



Ruder (2019)

Sequential transfer learning

Learn on one task / dataset, then transfer to another task / dataset



Example: word vectors

Word embedding methods (e.g. word2vec) learn one vector per word:

cat = [0.1, -0.2, 0.4, …]

dog = [0.2, -0.1, 0.7, …]

Example: word vectors

Word embedding methods (e.g. word2vec) learn one vector per word:

cat = [0.1, -0.2, 0.4, …]

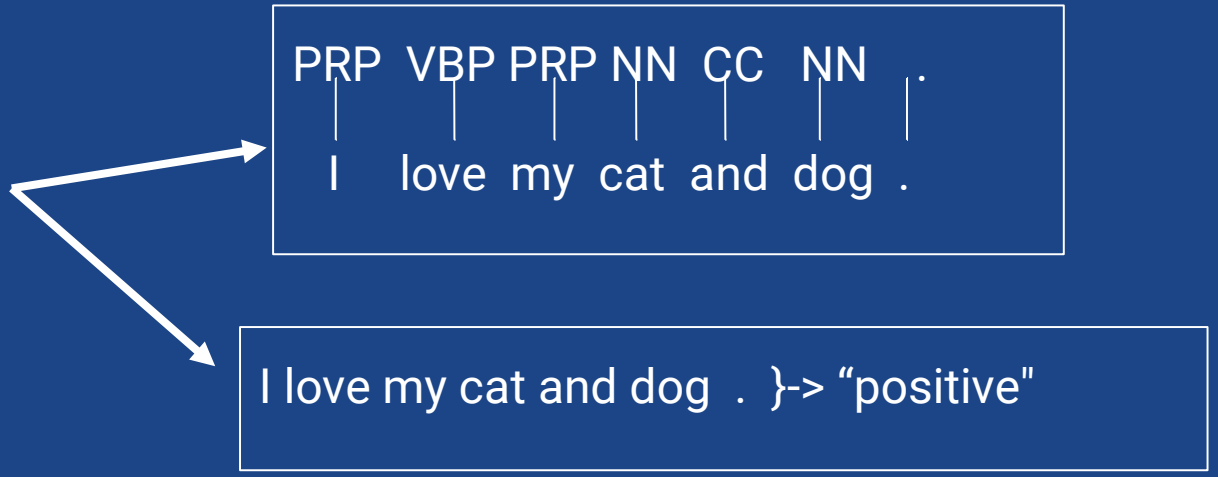
dog = [0.2, -0.1, 0.7, …]

Example: word vectors

Word embedding methods (e.g. word2vec) learn one vector per word:

cat = [0.1, -0.2, 0.4, …]

dog = [0.2, -0.1, 0.7, …]

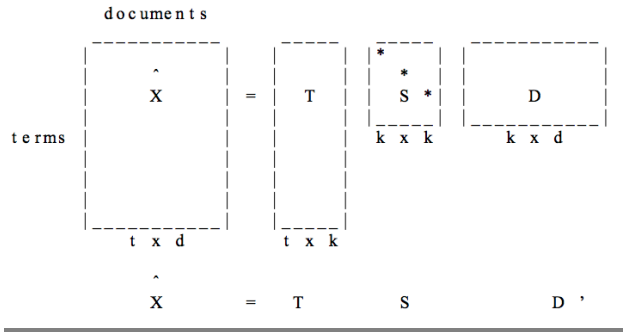


Why embed words?

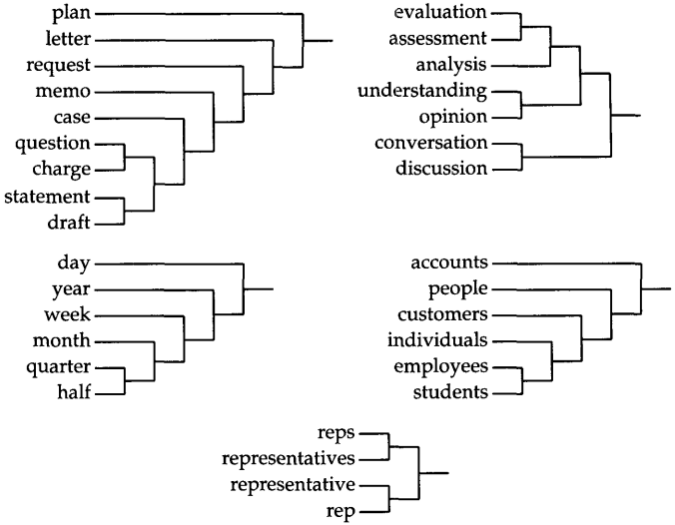
- ❑ Embeddings are themselves parameters, can be learned
- ❑ Much lower dimensional space than sparse one-hot encodings: easier to compute with
- ❑ We can generalize across words via their continuous dense word embeddings, as opposed to categorical one-hot encodings; mitigates data sparsity
- ❑ We can share word representations across tasks

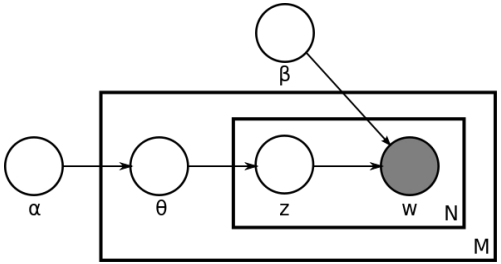
Unsupervised pretraining : Pre-Neural

Latent Semantic Analysis (LSA)—SVD of term-document matrix, ([Deerwester et al., 1990](#))



Brown clusters, hard hierarchical clustering based on n-gram LMs, ([Brown et al. 1992](#))





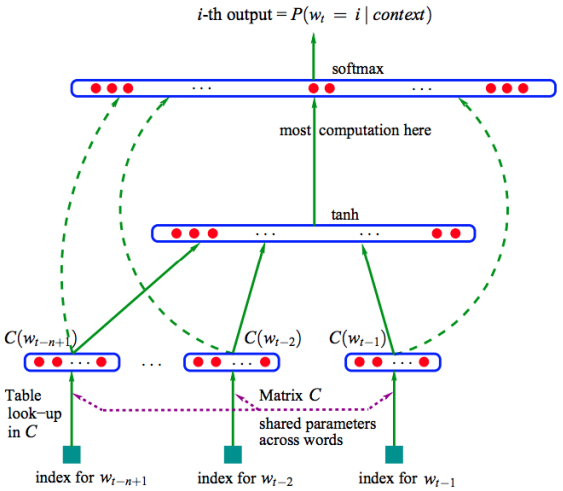
Latent Dirichlet Allocation (LDA)—Documents are mixtures of topics and topics are mixtures of words ([Blei et al., 2003](#))

Language Model Pretraining

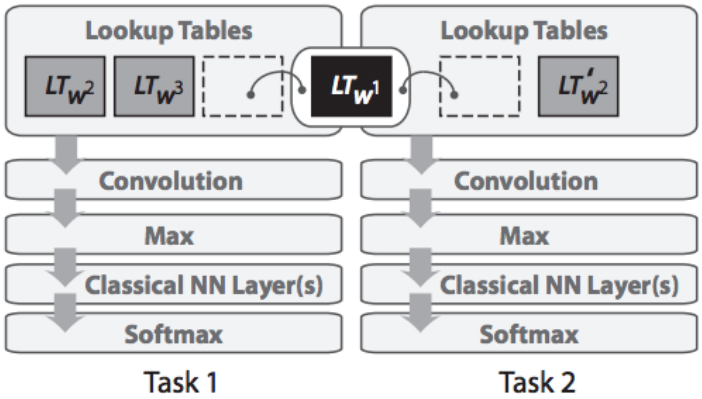
- Many successful pretraining approaches are based on language modeling
- Informally, a LM learns $P_\theta(text)$ or $P_\theta(text \mid some\ other\ text)$
- LMs don't require human annotation
- Many languages have enough text to learn high capacity model
- Versatile—can learn both sentence and word representations with a variety of objective functions

Word vector pretraining

n-gram neural language model
(Bengio et al. 2003)



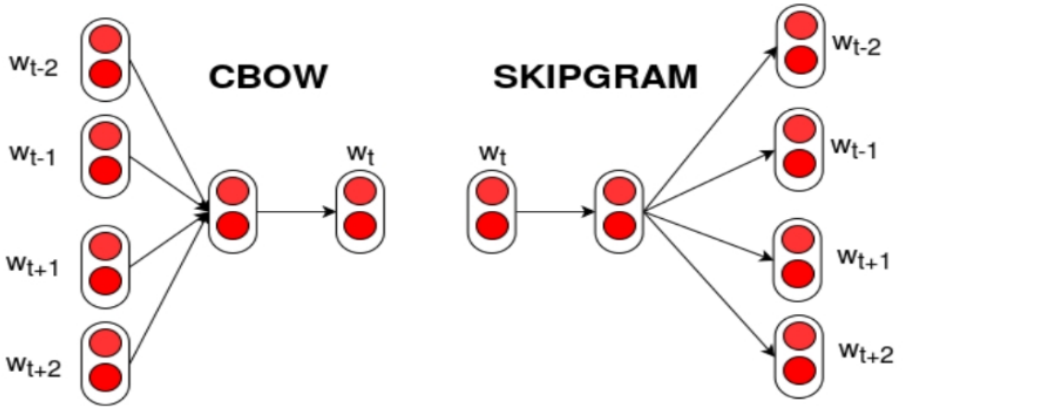
Supervised multitask word embeddings
(Collobert and Weston, 2008)



word2vec

Efficient algorithm + large scale training → high quality word vectors

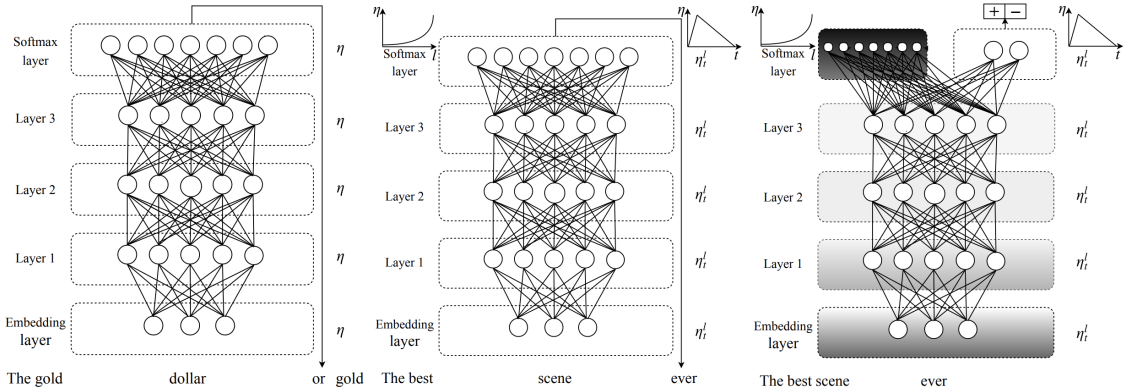
(Mikolov et al., 2013)



$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0} logp(w_t|w_{t+j}) \qquad \frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0} logp(w_{t+j}|w_t)$$

See also:
Pennington et al. (2014): GloVe
Bojanowski et al. (2017): fastText

ULMFiT



(a) LM pre-training (b) LM fine-tuning (c) Classifier fine-tuning

Pretrain AWD-LSTM LM, fine-tune LM in two stages with different adaptation techniques

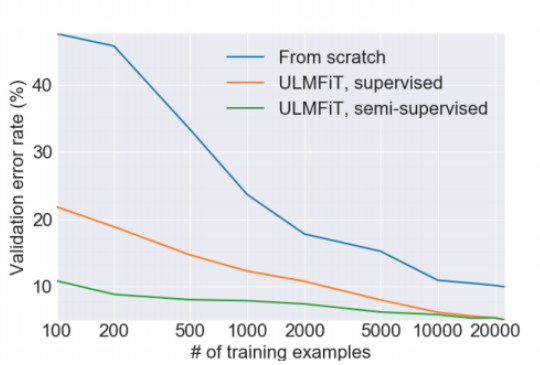
	Model	Test		Model	Test
IMDb	CoVe (McCann et al., 2017)	8.2	TREC-6	CoVe (McCann et al., 2017)	4.2
	oh-LSTM (Johnson and Zhang, 2016)	5.9		TBCNN (Mou et al., 2015)	4.0
	Virtual (Miyato et al., 2016)	5.9		LSTM-CNN (Zhou et al., 2016)	3.9
	ULMFiT (ours)	**4.6**		ULMFiT (ours)	**3.6**

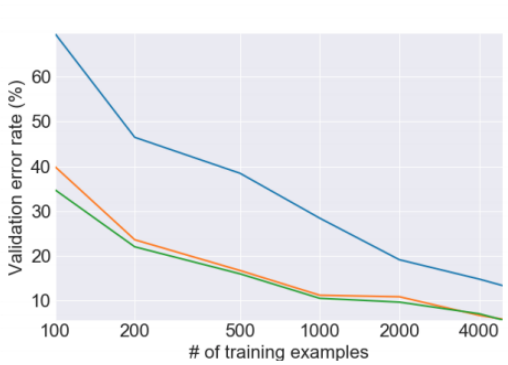
	AG	DBpedia	Yelp-bi	Yelp-full
Char-level CNN (Zhang et al., 2015)	9.51	1.55	4.88	37.95
CNN (Johnson and Zhang, 2016)	6.57	0.84	2.90	32.39
DPCNN (Johnson and Zhang, 2017)	6.87	0.88	2.64	30.58
ULMFiT (ours)	**5.01**	**0.80**	**2.16**	**29.98**

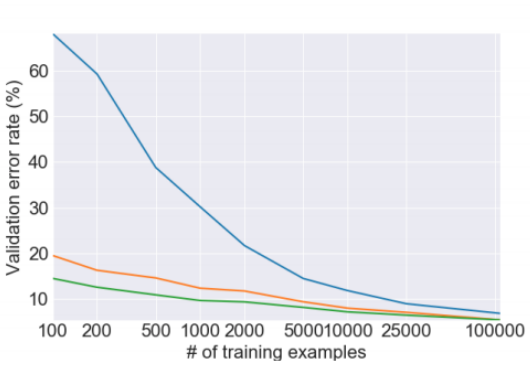
SOTA for six classification datasets

(Howard and Ruder, ACL 2018)

Pretraining reduces need for annotated data







(Howard and Ruder, ACL 2018)

Why does language modeling work so well?

- Language modeling is a very difficult task, even for humans.

- Language models are expected to compress any possible context into a vector that generalizes over possible completions.
  - "They walked down the street to ???"
- To have any chance at solving this task, a model is forced to learn syntax, semantics, encode facts about the world, etc.
- Given enough data, a huge model, and enough compute, can do a reasonable job!

- Empirically works better than translation, autoencoding: "Language Modeling Teaches You More Syntax than Translation Does" (Zhang et al. 2018)

Today

- ❑ Go through one example of sequential transfer learning for a classification task in full detail
- ❑ Data: the IMDB movie reviews (Maas et al. 2011)
- ❑ Transfer learning following the ULMFiT model (Howard and Ruder 2018)
- ❑ We use the AWD-LSTM model (Merity et al 2017), a vanilla LSTM with four kinds of dropout regularization
- ❑ Small, simple model: word embedding size of 300, 2 LSTM layers with 300 hidden size per layer, BPTT (back-propagation through time) of size 70
- ❑ We pretrain a language model on Wikitext-103 (Merity et al. 2017b): 28595 preprocessed Wikipedia articles, a total of 103 million words

- ❑ Pretrained model is small and simple (no attention, skip connections etc.) and the pretraining corpus is of modest size.

Today (ctd.)

- ❑ We finetune the pretrained AWD-LSTM model using discriminative (Yosinski et al. 2014) and slanted triangular (Smith 2017; Howard and Ruder 2018) learning rates
- ❑ Discriminative learning rates help us avoid catastrophic forgetting
- ❑ Slanted triangular learning rates improve (rate of) convergence in training
- ❑ We do the same kind of minimal preprocessing as in Howard and Ruder (2018)
- ❑ Notebooks based on the Fastai course, v3, part 2: https://course.fast.ai/part2
  - We start with an overview of the preprocessing steps
  - We introduce the AWD-LSTM model
  - We show how to pretrain the AWD-LSTM model on Wikipedia
  - We do ULMFiT-style transfer learning for IMDB sentiment classification