Current trends in transfer learning and language modelling

Tomasz Stanisławek

[1]Applica.ai

[2]Faculty of Mathematics and Information Science, Warsaw University of Technology

MI2 DataLab seminar, 09.03.2020

Transfer learning - History

Transfer of learning (Psychology) is the dependency of human conduct, learning, or performance on prior experience.(1901)[1]

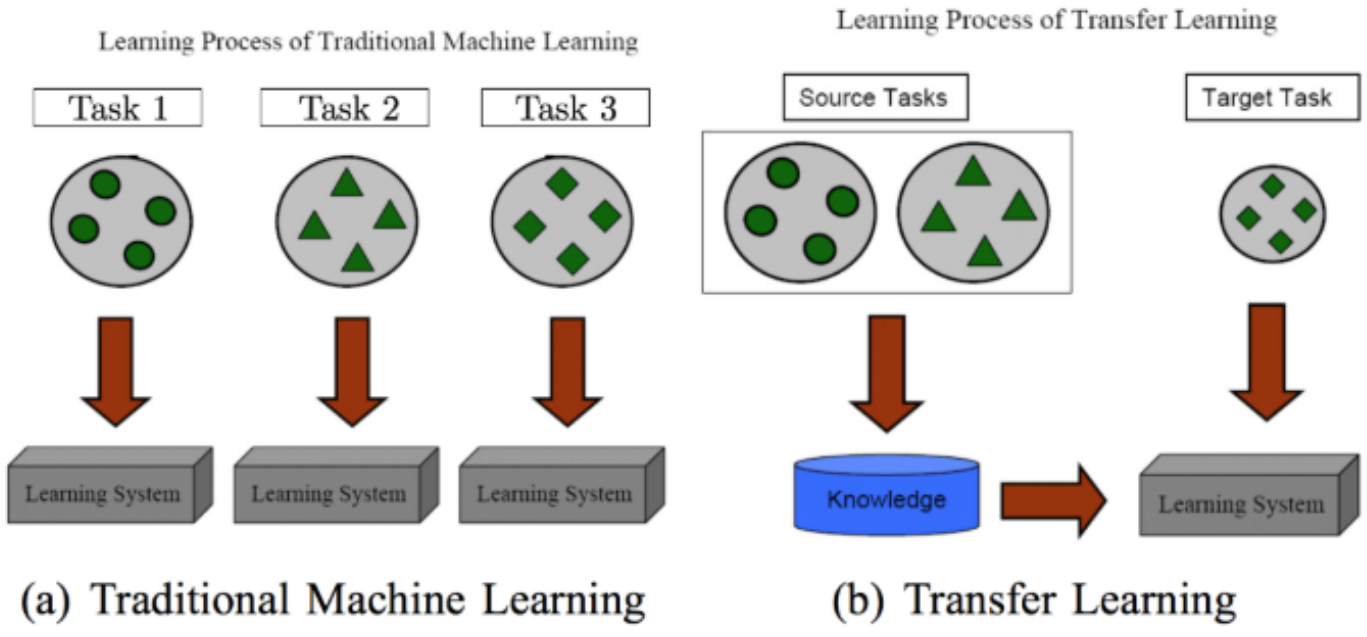
Transfer of learning can be described as the process and the effective extent to which past experiences (a.k.a. transfer source) affect learning and performance in a new situation (the transfer target).(1965)[2]

Transfer of knowledge goes far beyond simply repeating memorized material but to being able to take old knowledge and experiences and apply this old knowledge to a new concept and being able to use both the new and old knowledge to solving a problem that you have never encountered before.(2004)[3]

[1] Thorndike, E. L. and Woodworth, R. S. "The influence of improvement in one mental function upon the efficiency of other functions"
[2] Ellis, H. C. The Transfer of Learning. New York: The Macmillan Company.
[3] Phillips and Soltis. Perspectives on Learning. Teachers College. pp. 70–72.

Transfer learning - History

Transfer learning (Machine Learning) is a research problem in machine learning that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem.[1]



[1] https://en.wikipedia.org/wiki/Transfer_learning
[2] Pan and Yang (2010), A Survey on Transfer Learning

Transfer learning - History

- In 1993, Lorien Pratt published a paper on transfer in machine learning, formulating the DBT algorithm;[1]

Discriminability-Based Transfer between Neural Networks

L. Y. Pratt
Department of Mathematical and Computer Sciences
Colorado School of Mines
Golden, CO 80401
lpratt@mines.colorado.edu

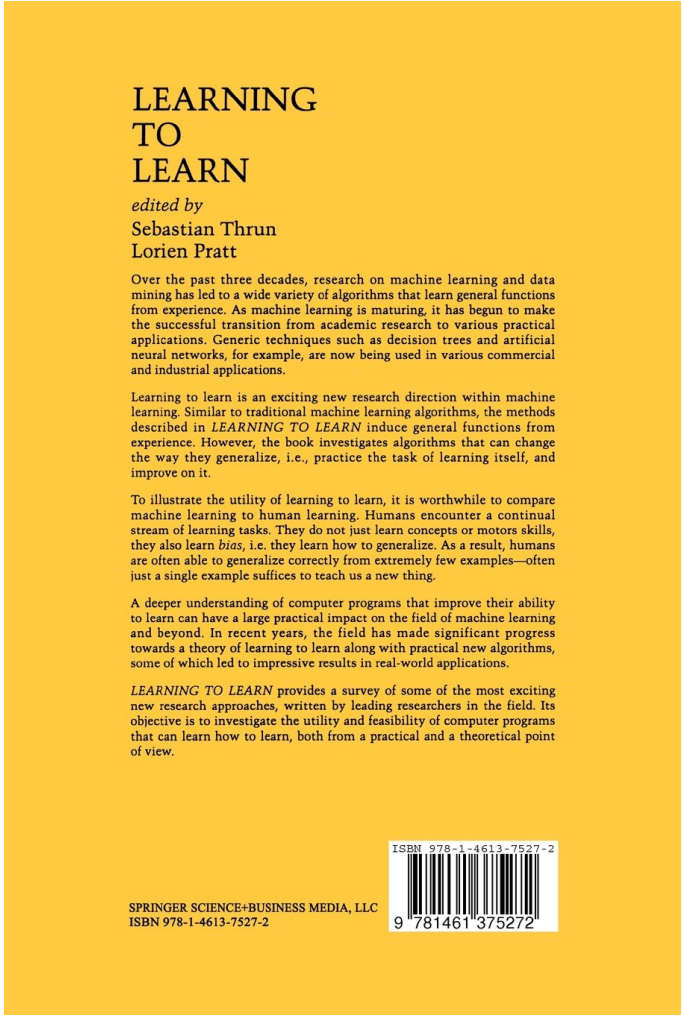
Abstract

Previously, we have introduced the idea of neural network *transfer*, where learning on a *target* problem is sped up by using the weights obtained from a network trained for a related *source* task. Here,

[1] Pratt, L. Y. "Discriminability-based transfer between neural networks" (PDF). NIPS Conference.

Transfer learning - History

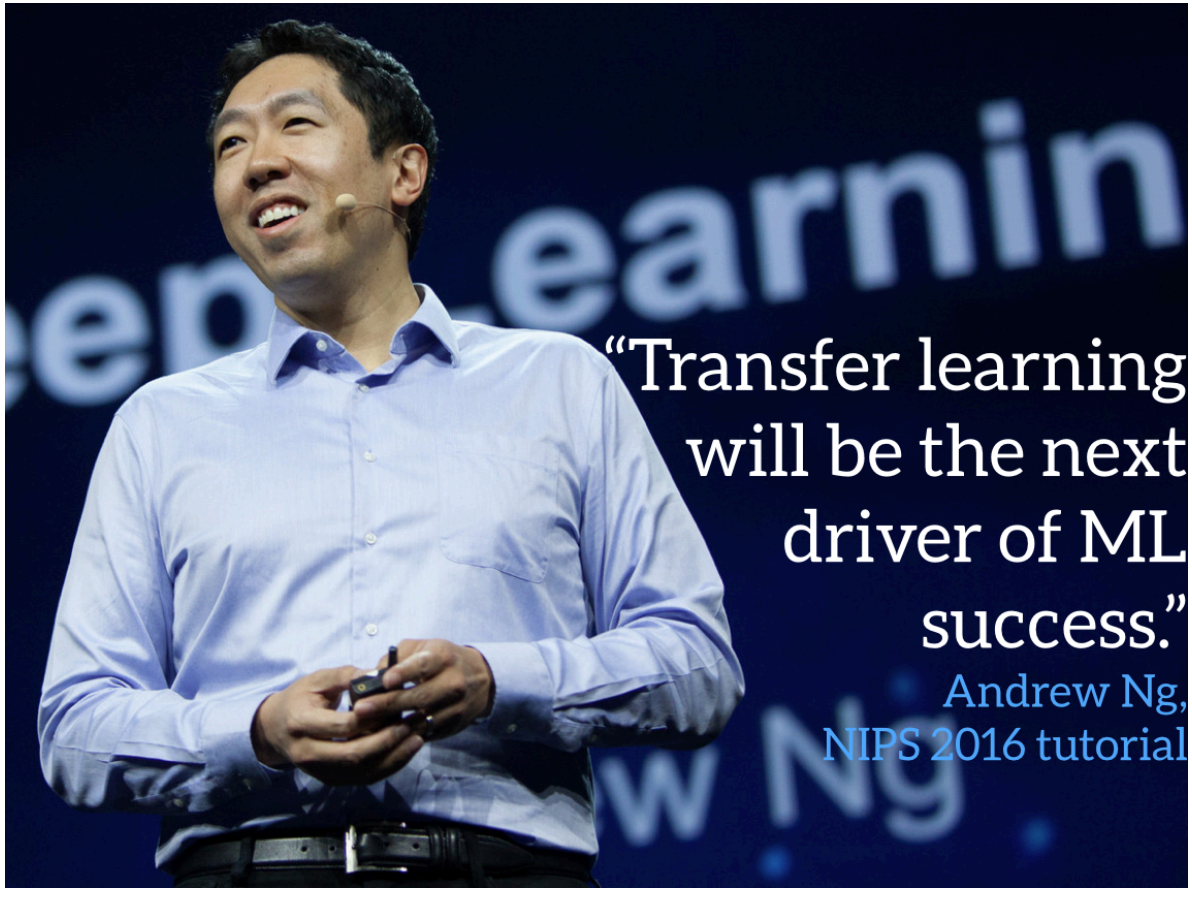
- In 1998 L. Pratt and S. Thrun review the subject in book ‘Learning to Learn‘ (include multi-task learning);[1][2]



[1] Caruana, R., "Multitask Learning", pp. 95-134 in Pratt & Thrun
[2] Thrun, Sebastian; Pratt, Lorien . Learning to Learn. Springer Science & Business Media.

Transfer learning - History

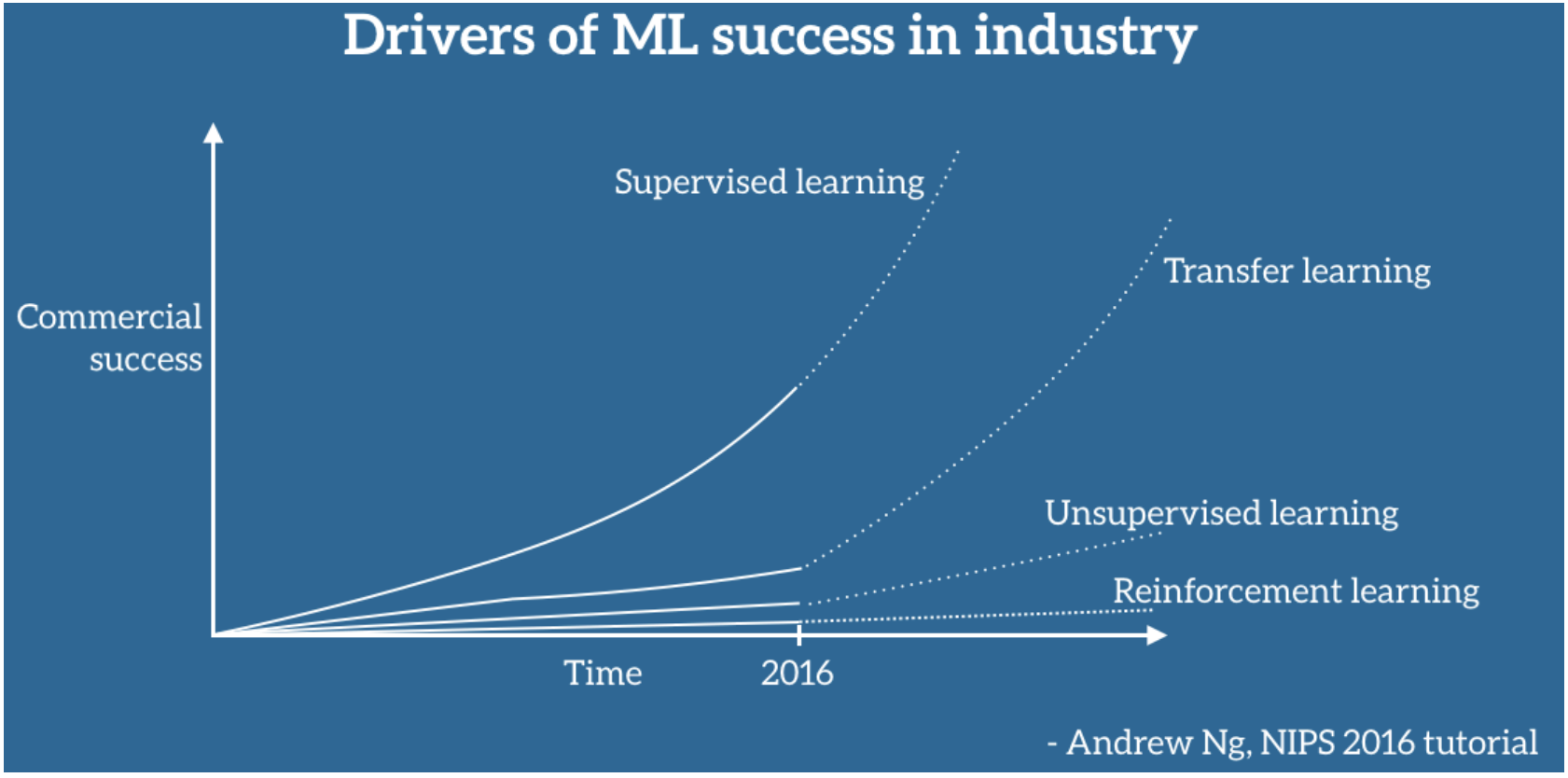
- In 2016 at NIPS tutorial Andrew Ng said that TL will be next driver of ML commercial success after supervised learning to highlight the importance of TL.[1]



[1] NIPS 2016 tutorial: "Nuts and bolts of building AI applications using Deep Learning" by Andrew Ng

Transfer learning - History

- In 2016 at NIPS tutorial Andrew Ng said that TL will be next driver of ML commercial success after supervised learning to highlight the importance of TL.[1]



[1] NIPS 2016 tutorial: "Nuts and bolts of building AI applications using Deep Learning" by Andrew Ng

Transfer learning - Basic concepts

For this definition, we will closely follow the excellent survey by Pan and Yang (2010) [6] with binary document classification as a running example. Transfer learning involves the concepts of a domain and a task. A domain $\mathcal{D}$ consists of a feature space $\mathcal{X}$ and a marginal probability distribution $P(X)$ over the feature space, where $X = x_1, \cdots, x_n \in \mathcal{X}$. For document classification with a bag-of-words representation, $\mathcal{X}$ is the space of all document representations, x_i is the i-th term vector corresponding to some document and X is the sample of documents used for training.

Given a domain, $\mathcal{D} = \{\mathcal{X}, P(X)\}$, a task $\mathcal{T}$ consists of a label space $\mathcal{Y}$ and a conditional probability distribution $P(Y|X)$ that is typically learned from the training data consisting of pairs $x_i \in X$ and $y_i \in \mathcal{Y}$. In our document classification example, $\mathcal{Y}$ is the set of all labels, i.e. *True*, *False* and y_i is either *True* or *False*.

Given a source domain $\mathcal{D}_S$, a corresponding source task $\mathcal{T}_S$, as well as a target domain $\mathcal{D}_T$ and a target task $\mathcal{T}_T$, the objective of transfer learning now is to enable us to learn the target conditional probability distribution $P(Y_T|X_T)$ in $\mathcal{D}_T$ with the information gained from $\mathcal{D}_S$ and $\mathcal{T}_S$ where $\mathcal{D}_S \neq \mathcal{D}_T$ or $\mathcal{T}_S \neq \mathcal{T}_T$. In most cases, a limited number of labeled target examples, which is exponentially smaller than the number of labeled source examples are assumed to be available.

[1] https://ruder.io/transfer-learning/

Transfer learning - Basic concepts

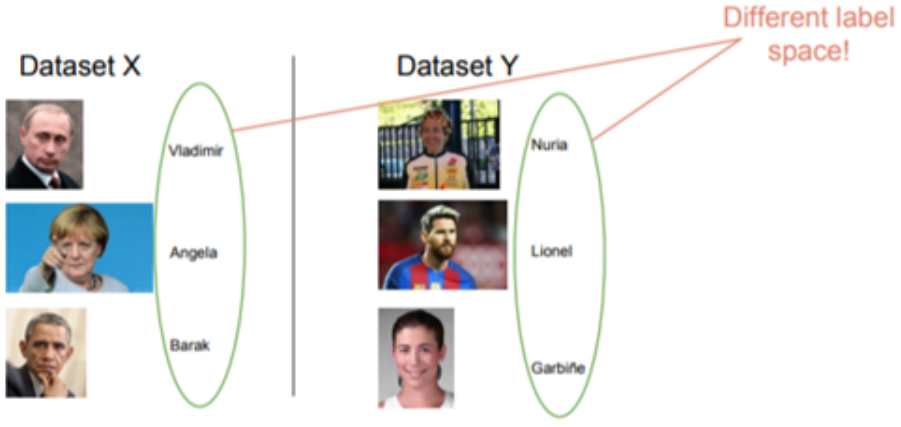
1. $\mathcal{X}_S \neq \mathcal{X}_T$. The feature spaces of the source and target domain are different, e.g. the documents are written in two different languages. In the context of natural language processing, this is generally referred to as cross-lingual adaptation.
2. $P(X_S) \neq P(X_T)$. The marginal probability distributions of source and target domain are different, e.g. the documents discuss different topics. This scenario is generally known as domain adaptation.
3. $\mathcal{Y}_S \neq \mathcal{Y}_T$. The label spaces between the two tasks are different, e.g. documents need to be assigned different labels in the target task. In practice, this scenario usually occurs with scenario 4, as it is extremely rare for two different tasks to have different label spaces, but exactly the same conditional probability distributions.
4. $P(Y_S|X_S) \neq P(Y_T|X_T)$. The conditional probability distributions of the source and target tasks are different, e.g. source and target documents are unbalanced with regard to their classes. This scenario is quite common in practice and approaches such as over-sampling, under-sampling, or SMOTE [7] are widely used.

[1] https://ruder.io/transfer-learning/

Transfer learning - Basic concepts

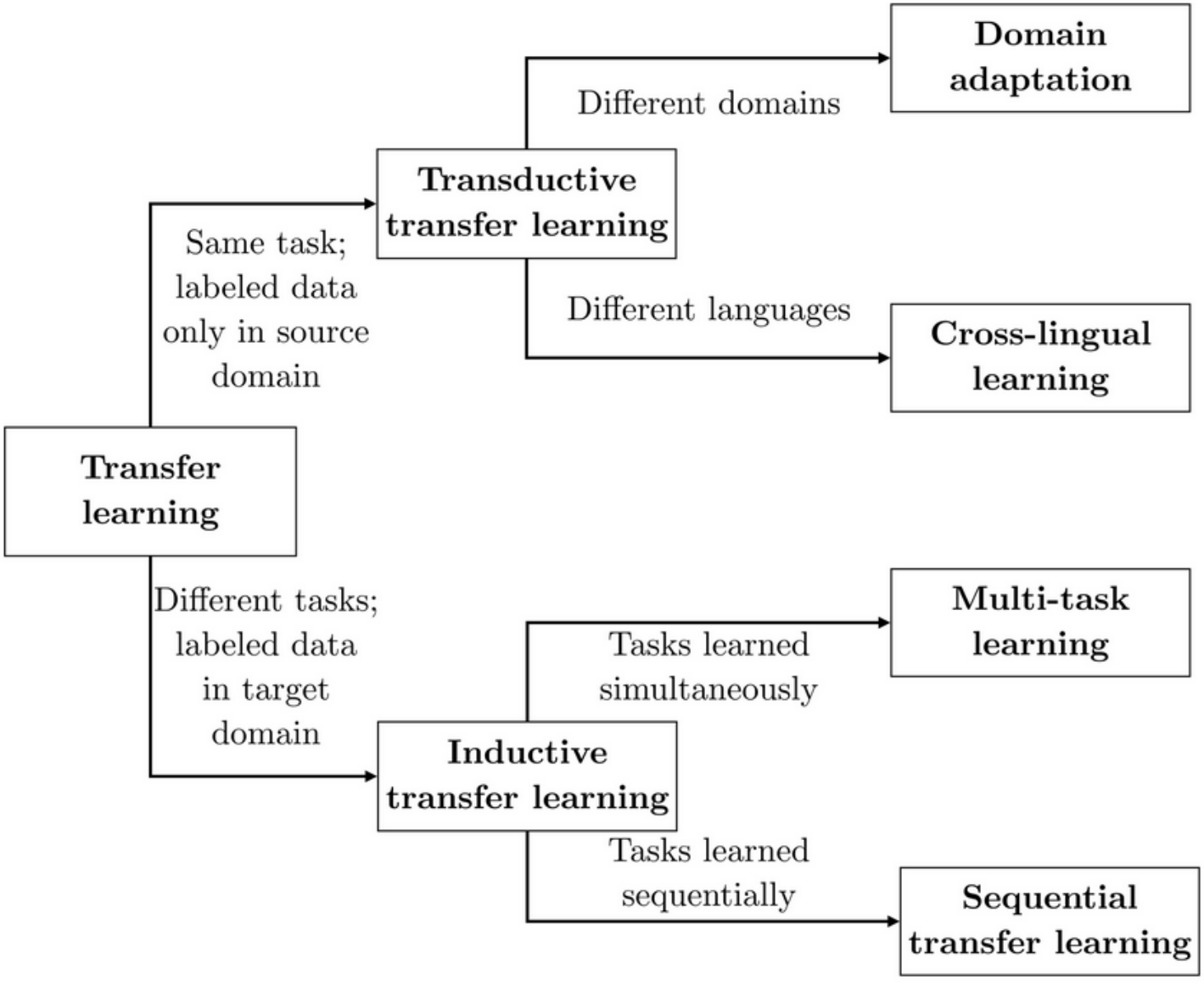
If two domains are different, they may have different feature spaces or different marginal distributions

If two tasks are different, they may have different label spaces or different conditional distributions



[1] https://towardsdatascience.com/a-comprehensive-hands-on-guide-to-transfer-learning-with-real-world-applications-in-deep-learning-212bf3b2f27a

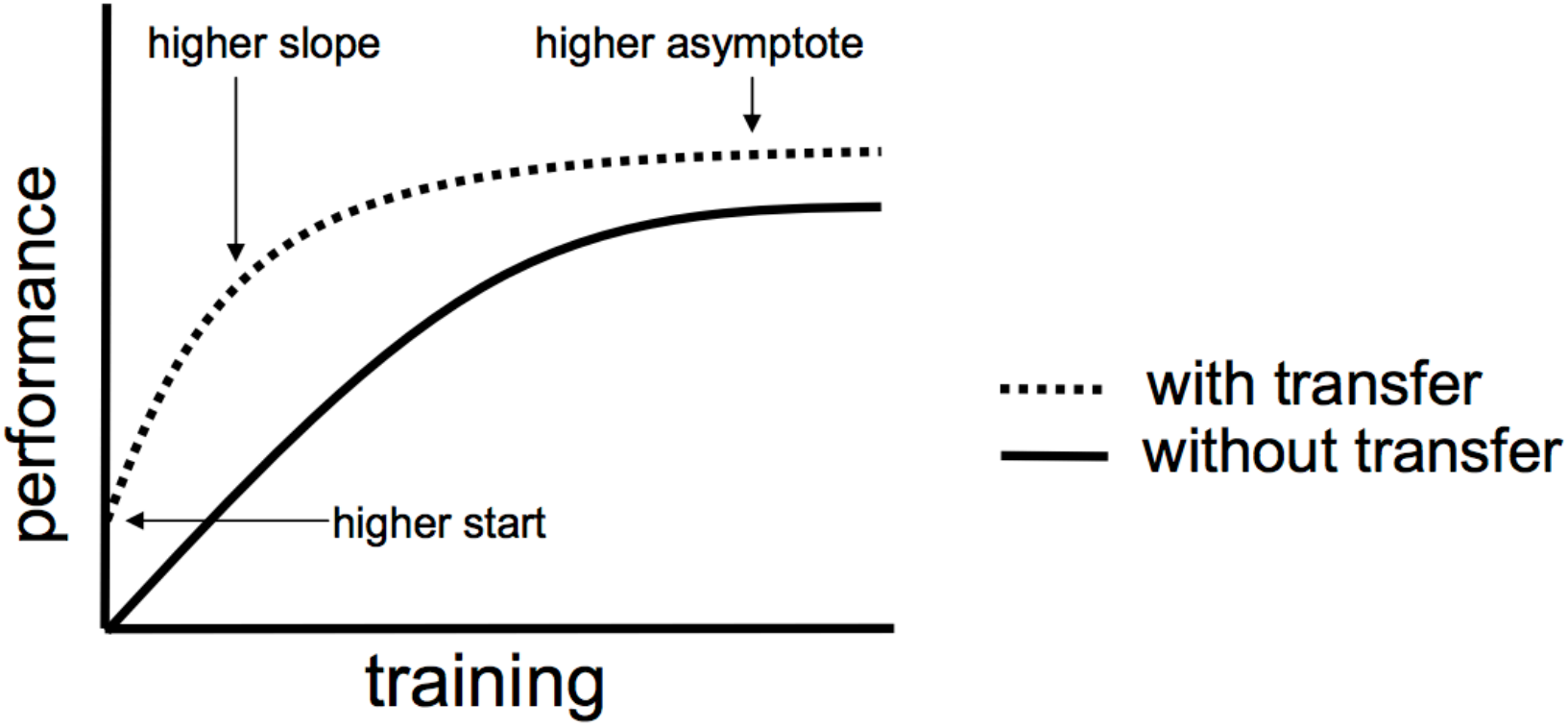
Transfer learning - Basic concepts



A taxonomy for transfer learning in NLP (Ruder, 2019).

Transfer learning - Basic concepts

	Inductive Transfer Learning	Transductive Transfer Learning	Unsupervised Transfer Learning
Instance-transfer	√	√	
Feature-representation-transfer	√	√	√
Parameter-transfer	√		
Relational-knowledge-transfer	√		

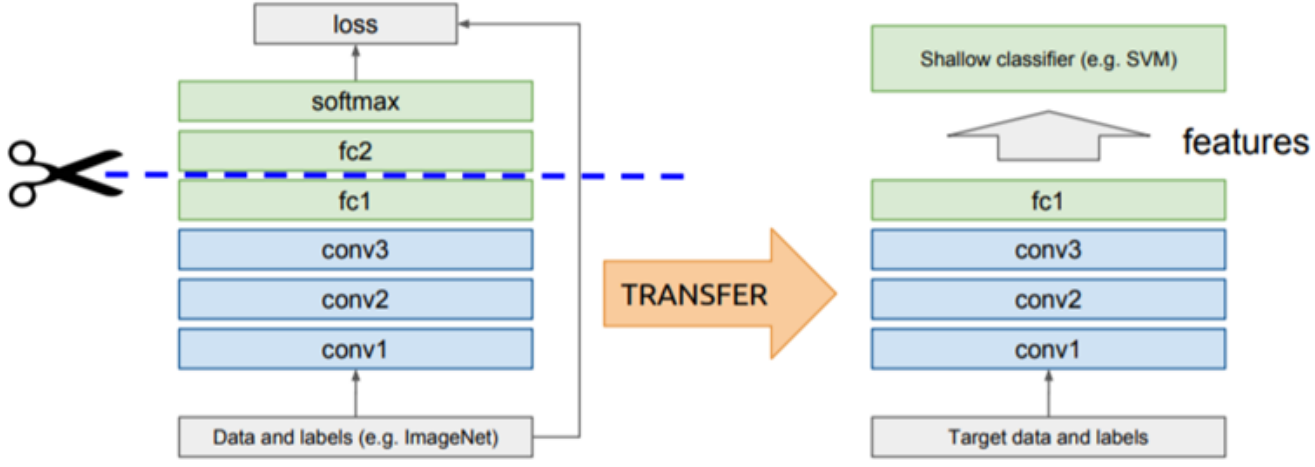


[1] https://towardsdatascience.com/a-comprehensive-hands-on-guide-to-transfer-learning-with-real-world-applications-in-deep-learning-212bf3b2f27a/
[2] Lisa Torrey and Jude Shavlik (2009), Transfer learning

Transfer learning - Simple examples

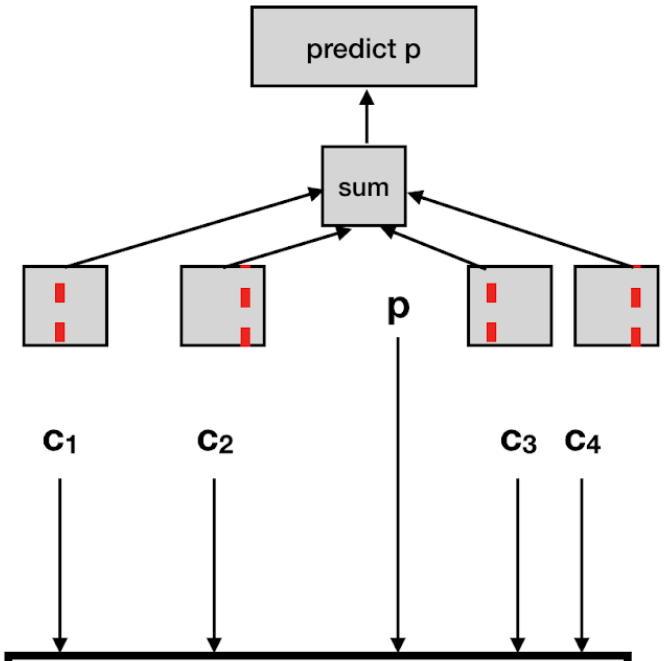
Idea: use outputs of one or more layers of a network trained on a different task as generic feature detectors. Train a new shallow model on these features.

Assumes that $D_S = D_T$



[1] https://towardsdatascience.com/a-comprehensive-hands-on-guide-to-transfer-learning-with-real-world-applications-in-deep-learning-212bf3b2f27a/

Transfer learning - Simple examples

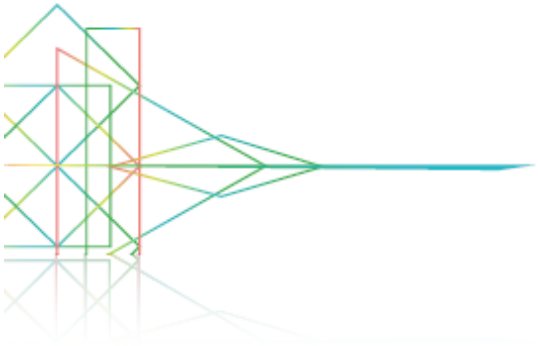


Word2Vec

Lorem ipsum dolor sit amet consectetur adipiscing elit sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa

[1] http://daniel-at-world.blogspot.com/2019/03/word2vec-implementation-in-rust.html

Transfer learning - More advanced NLP



Method

ELMo represents a word t_k as a linear combination of corresponding hidden layers (inc. its embedding)

ELMo is a task specific representation. A down-stream task learns weighting parameters

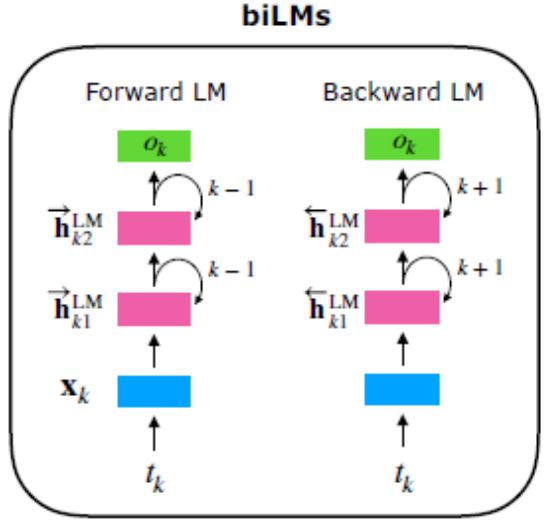
$$\mathbf{ELMo}_k^{\text{task}} = \gamma^{\text{task}} \times \sum \left\{ s_2^{\text{task}} \times \mathbf{h}_{k2}^{\text{LM}},\ s_1^{\text{task}} \times \mathbf{h}_{k1}^{\text{LM}},\ s_0^{\text{task}} \times \mathbf{h}_{k0}^{\text{LM}} \right.$$

$$\left. ([\mathbf{x}_k ; \mathbf{x}_k]) \right.$$

Concatenate hidden layers

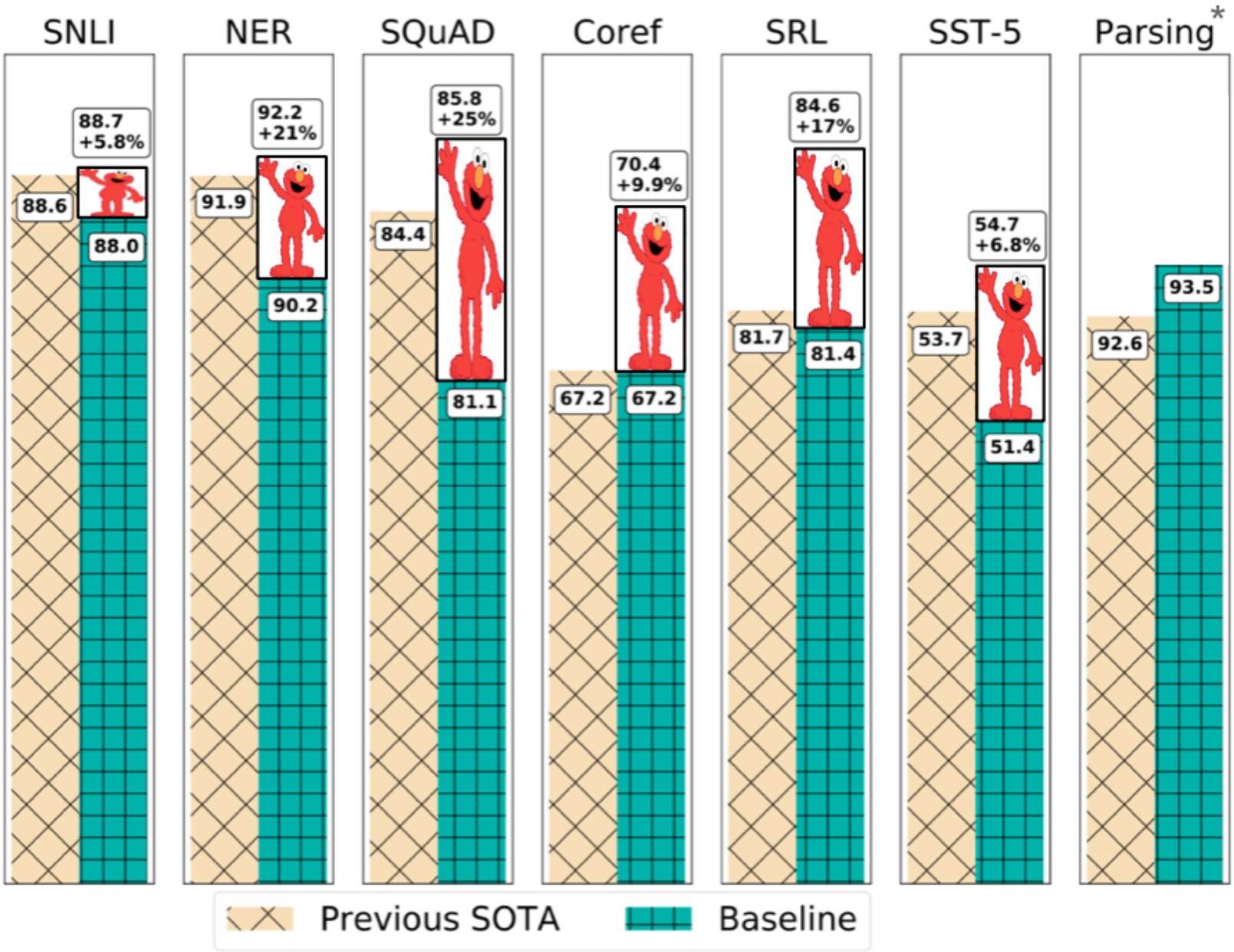
$$[\overrightarrow{\mathbf{h}}_{kj}^{\text{LM}} ; \overleftarrow{\mathbf{h}}_{kj}^{\text{LM}}]$$

Unlike usual word embeddings, ELMo is assigned to every *token* instead of a *type*



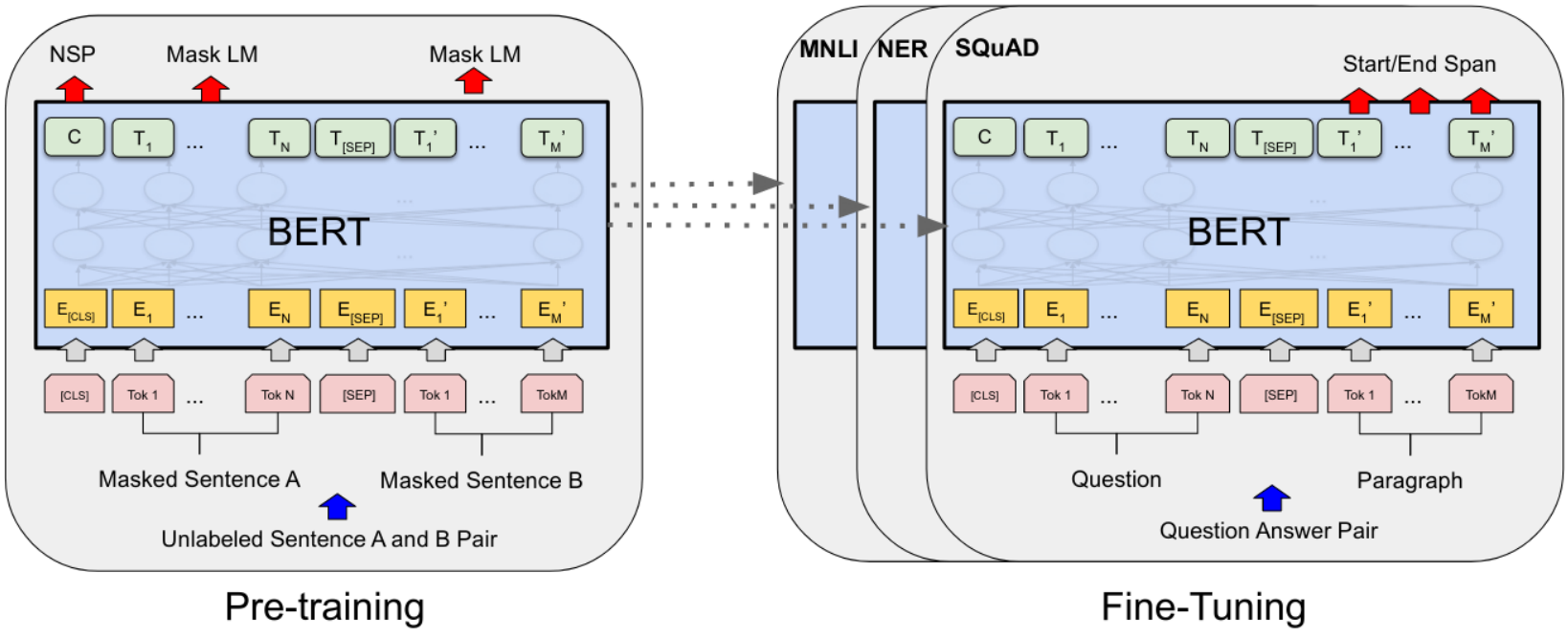
[1] Deep contextualized word representations, Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, Luke Zettlemoyer. NAACL 2018.

Transfer learning - More advanced NLP



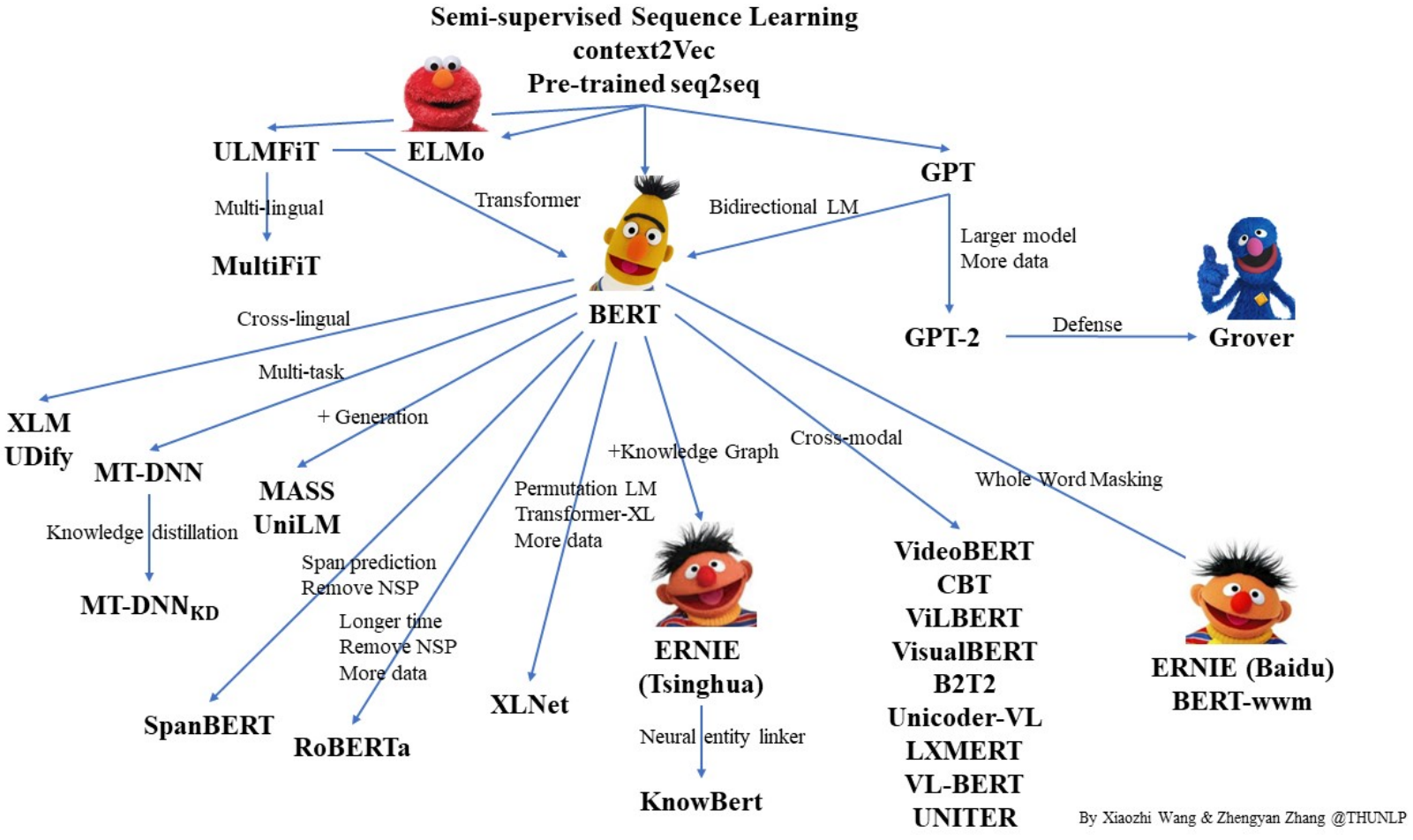
*Kitaev and Klein, ACL 2018 (see also Joshi et al., ACL 2018)

Transfer learning - More advanced NLP



[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (2018)

Language Models ZOO



https://github.com/thunlp/PLMpapers

Language Models - SOTA

	Rank	Name	Model	URL	Score	CoLA	SST-2	MRPC	STS-B	QQP	MNLI-m	MNLI-mm	QNLI	RTE	WNLI	AX
+	1	Alibaba DAMO NLP	StructBERT		90.3	75.3	97.1	93.9/91.9	93.0/92.5	74.8/91.0	90.9	90.7	96.4	90.2	94.5	49.1
	2	T5 Team - Google	T5		90.3	71.6	97.5	92.8/90.4	93.1/92.8	75.1/90.6	92.2	91.9	96.9	92.8	94.5	53.1
	3	ERNIE Team - Baidu	ERNIE		90.1	72.8	97.5	93.2/91.0	92.9/92.5	75.2/90.8	91.2	90.8	96.1	90.9	94.5	49.4
	4	Microsoft D365 AI & MSR AI & GATECH	MT-DNN-SMART		89.9	69.5	97.5	93.7/91.6	92.9/92.5	73.9/90.2	91.0	90.8	99.2	89.7	94.5	50.2
+	5	ELECTRA Team	ELECTRA-Large + Standard Tricks		89.4	71.7	97.1	93.1/90.7	92.9/92.5	75.6/90.8	91.3	90.8	95.8	89.8	91.8	50.7
+	6	Microsoft D365 AI & UMD	FreeLB-RoBERTa (ensemble)		88.4	68.0	96.8	93.1/90.8	92.3/92.1	74.8/90.3	91.1	90.7	95.6	88.7	89.0	50.1
	7	Junjie Yang	HIRE-RoBERTa		88.3	68.6	97.1	93.0/90.7	92.4/92.0	74.3/90.2	90.7	90.4	95.5	87.9	89.0	49.3
	8	Facebook AI	RoBERTa		88.1	67.8	96.7	92.3/89.8	92.2/91.9	74.3/90.2	90.8	90.2	95.4	88.2	89.0	48.7
+	9	Microsoft D365 AI & MSR AI	MT-DNN-ensemble		87.6	68.4	96.5	92.7/90.3	91.1/90.7	73.7/89.9	87.9	87.4	96.0	86.3	89.0	42.8
	10	GLUE Human Baselines	GLUE Human Baselines		87.1	66.4	97.8	86.3/80.8	92.7/92.6	59.5/80.4	92.0	92.8	91.2	93.6	95.9	-
	11	Stanford Hazy Research	Snorkel MeTaL		83.2	63.8	96.2	91.5/88.5	90.1/89.7	73.1/89.9	87.6	87.2	93.9	80.9	65.1	39.9
	12	XLM Systems	XLM (English only)		83.1	62.9	95.6	90.7/87.1	88.8/88.2	73.2/89.8	89.1	88.5	94.0	76.0	71.9	44.7

https://gluebenchmark.com/leaderboard

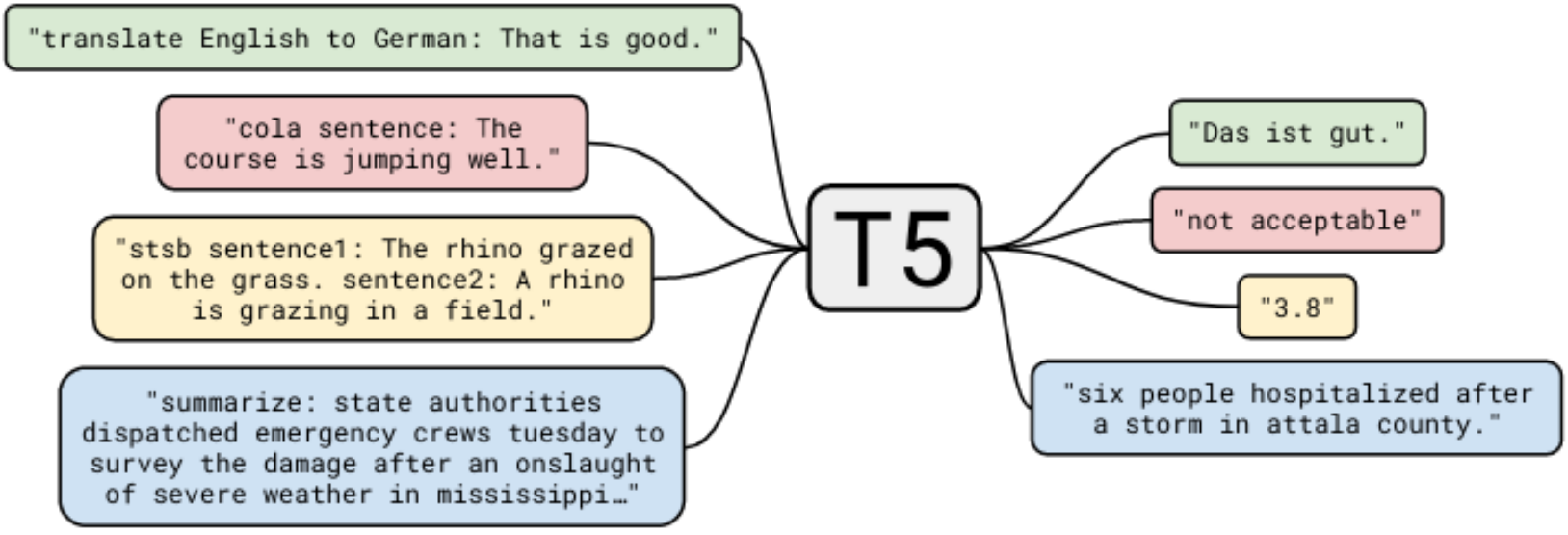
Language Models - SOTA

	Rank	Name	Model	URL	Score	BoolQ	CB	COPA	MultiRC	ReCoRD	RTE	WiC	WSC	AX-g	AX-b
	1	SuperGLUE Human Baselines	SuperGLUE Human Baselines		89.8	89.0	95.8/98.9	100.0	81.8/51.9	91.7/91.3	93.6	80.0	100.0	99.3/99.7	76.6
✚	2	T5 Team - Google	T5		89.3	91.2	93.9/96.8	94.8	88.1/63.3	94.1/93.4	92.5	76.9	93.8	92.7/91.9	65.6
	3	Zhuiyi Technology	RoBERTa-mtl-adv		85.7	87.1	92.4/95.6	91.2	85.1/54.3	91.7/91.3	88.1	72.1	91.8	91.0/78.1	58.5
	4	Facebook AI	RoBERTa		84.6	87.1	90.5/95.2	90.6	84.4/52.5	90.6/90.0	88.2	69.9	89.0	91.0/78.1	57.9
	5	IBM Research AI	BERT-mtl		73.5	84.8	89.6/94.0	73.8	73.2/30.5	74.6/74.0	84.1	66.2	61.0	97.8/57.3	29.6
	6	SuperGLUE Baselines	BERT++		71.5	79.0	84.8/90.4	73.8	70.0/24.1	72.0/71.3	79.0	69.6	64.4	99.4/51.4	38.0
			BERT		69.0	77.4	75.7/83.6	70.6	70.0/24.1	72.0/71.3	71.7	69.6	64.4	97.8/51.7	23.0
			Most Frequent Class		47.1	62.3	21.7/48.4	50.0	61.1/0.3	33.4/32.5	50.3	50.0	65.1	100.0/50.0	0.0
			CBoW		44.5	62.2	49.0/71.2	51.6	0.0/0.5	14.0/13.6	49.7	53.1	65.1	100.0/50.0	-0.4

https://super.gluebenchmark.com/leaderboard

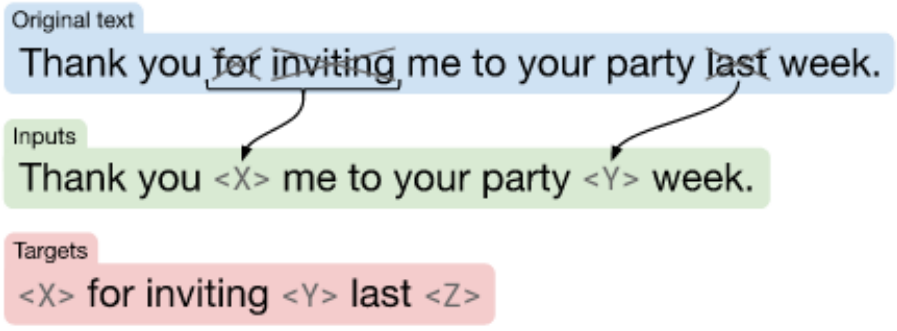
Language Models (sota) - T5

Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer, https://arxiv.org/abs/1910.10683



Every nlp task can be solved by text generation

Language Models (sota) - T5



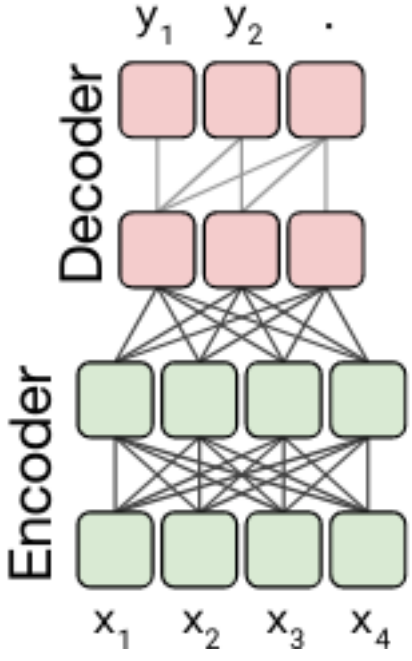
Objective	Inputs	Targets
Prefix language modeling	Thank you for inviting	me to your party last week .
BERT-style	Thank you <M> <M> me to your party apple week .	*(original text)*
Deshuffling	party me for your to . last fun you inviting week Thank	*(original text)*
I.i.d. noise, mask tokens	Thank you <M> <M> me to your party <M> week .	*(original text)*
I.i.d. noise, replace spans	Thank you <X> me to your party <Y> week .	<X> for inviting <Y> last <Z>
I.i.d. noise, drop tokens	Thank you me to your party week .	for inviting last
Random spans	Thank you <X> to <Y> week .	<X> for inviting me <Y> your party last <Z>

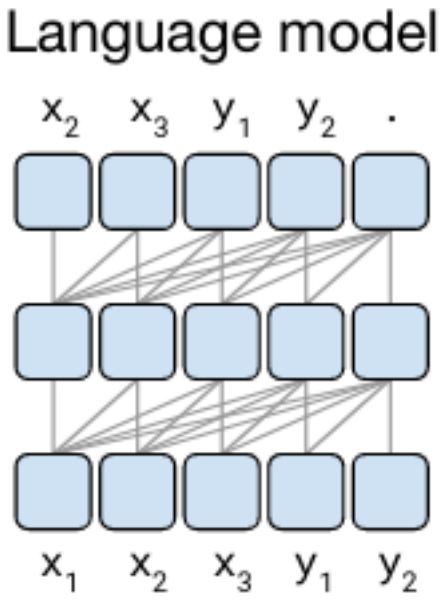
Objective	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
Prefix language modeling	80.69	18.94	77.99	65.27	**26.86**	39.73	**27.49**
BERT-style [Devlin et al., 2018]	**82.96**	**19.17**	**80.65**	**69.85**	**26.78**	**40.03**	**27.41**
Deshuffling	73.17	18.59	67.61	58.47	26.11	39.30	25.62

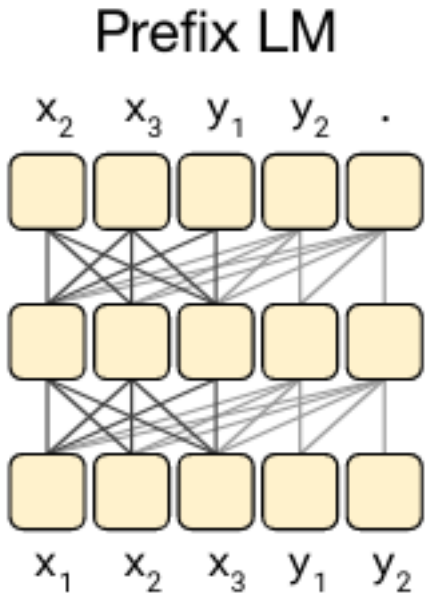
Objective	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
BERT-style [Devlin et al., 2018]	82.96	19.17	**80.65**	69.85	26.78	**40.03**	27.41
MASS-style [Song et al., 2019]	82.32	19.16	80.10	69.28	26.79	**39.89**	27.55
★ Replace corrupted spans	83.28	**19.24**	**80.88**	**71.36**	**26.98**	39.82	**27.65**
Drop corrupted tokens	**84.44**	**19.31**	**80.52**	68.67	**27.07**	39.76	**27.82**

Language Models (sota) - T5

Architecture	Objective	Params	Cost	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Encoder-decoder	Denoising	$2P$	M	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
Enc-dec, shared	Denoising	P	M	82.81	18.78	**80.63**	**70.73**	26.72	39.03	**27.46**
Enc-dec, 6 layers	Denoising	P	$M/2$	80.88	18.97	77.59	68.42	26.38	38.40	26.95
Language model	Denoising	P	M	74.70	17.93	61.14	55.02	25.09	35.28	25.86
Prefix LM	Denoising	P	M	81.82	18.61	78.94	68.11	26.43	37.98	27.39
Encoder-decoder	LM	$2P$	M	79.56	18.59	76.02	64.29	26.27	39.17	26.86
Enc-dec, shared	LM	P	M	79.60	18.13	76.35	63.50	26.62	39.17	27.05
Enc-dec, 6 layers	LM	P	$M/2$	78.67	18.26	75.32	64.06	26.13	38.42	26.89
Language model	LM	P	M	73.78	17.54	53.81	56.51	25.23	34.31	25.38
Prefix LM	LM	P	M	79.68	17.84	76.87	64.86	26.28	37.51	26.76





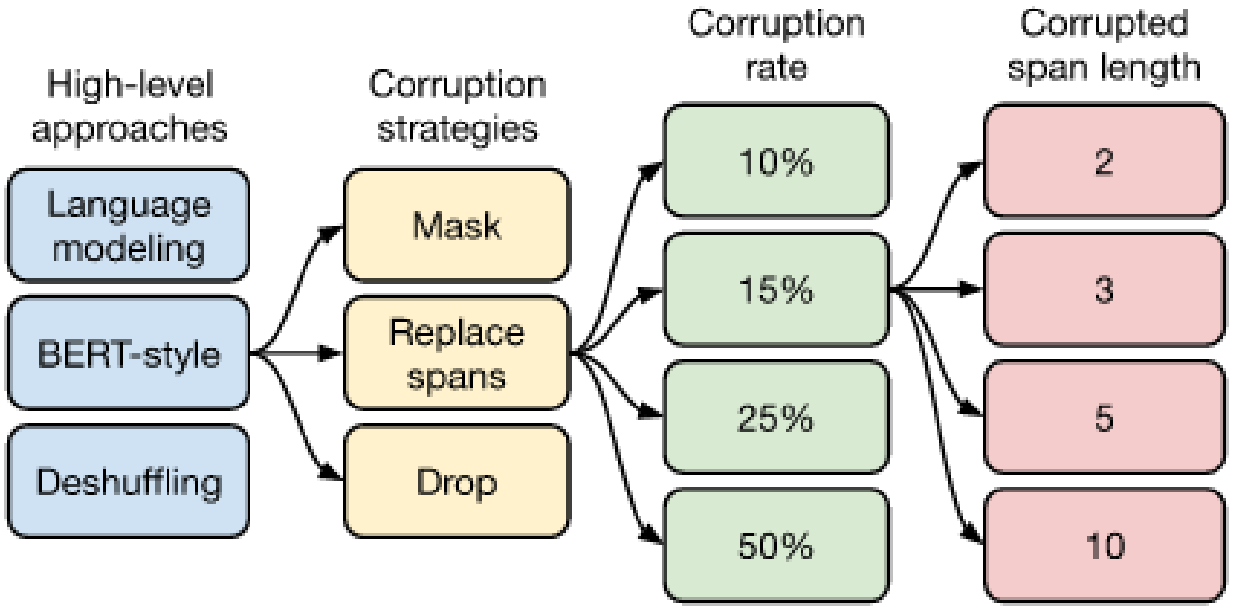


Language Models (sota) - T5

Dataset	Size	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ C4	745GB	83.28	**19.24**	80.88	71.36	**26.98**	**39.82**	**27.65**
C4, unfiltered	6.1TB	81.46	19.14	78.78	68.04	26.55	39.34	27.21
RealNews-like	35GB	**83.83**	**19.23**	80.39	72.38	**26.75**	**39.90**	**27.48**
WebText-like	17GB	**84.03**	**19.31**	**81.42**	71.40	**26.80**	**39.74**	**27.59**
Wikipedia	16GB	81.85	**19.31**	81.29	68.01	**26.94**	39.69	**27.67**
Wikipedia + TBC	20GB	83.65	**19.28**	**82.08**	**73.24**	**26.77**	39.63	**27.57**

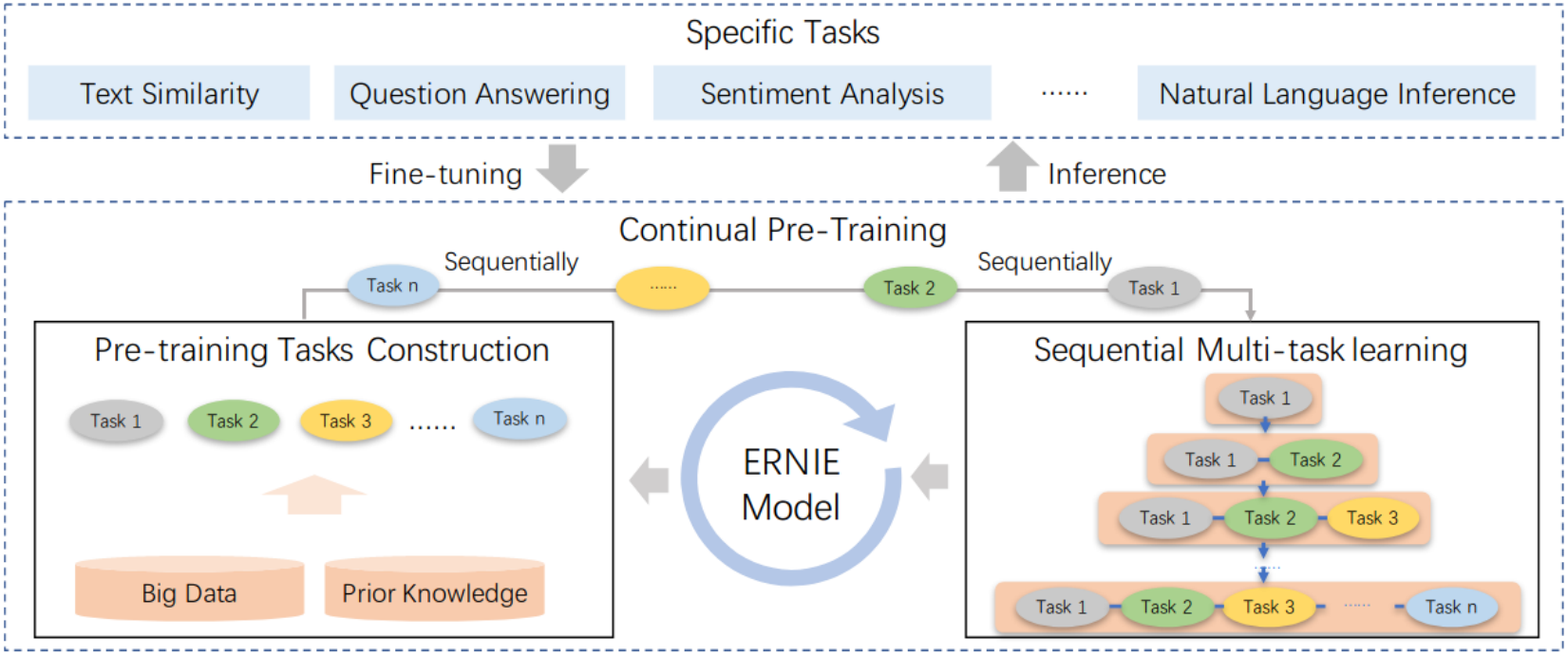
- C4 - based on one month of common crawl text (20TB)
- english only, lines which end with punctuation mark
- removed bad words, code, {}, lorem ipslums, duplicates

Language Models (sota) - T5



- Finetuning methods - adapter layers, gradual unfreezing
- multitasking strategies
- scaling strategies

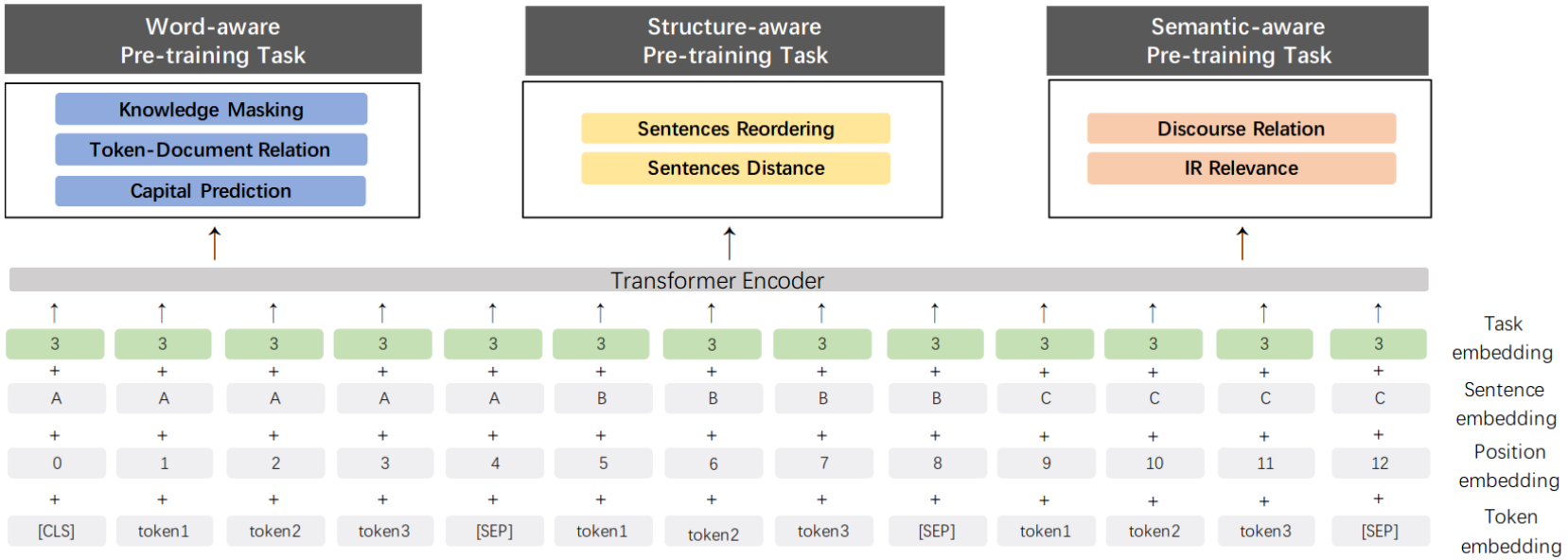
Language Models (sota) - ERNIE



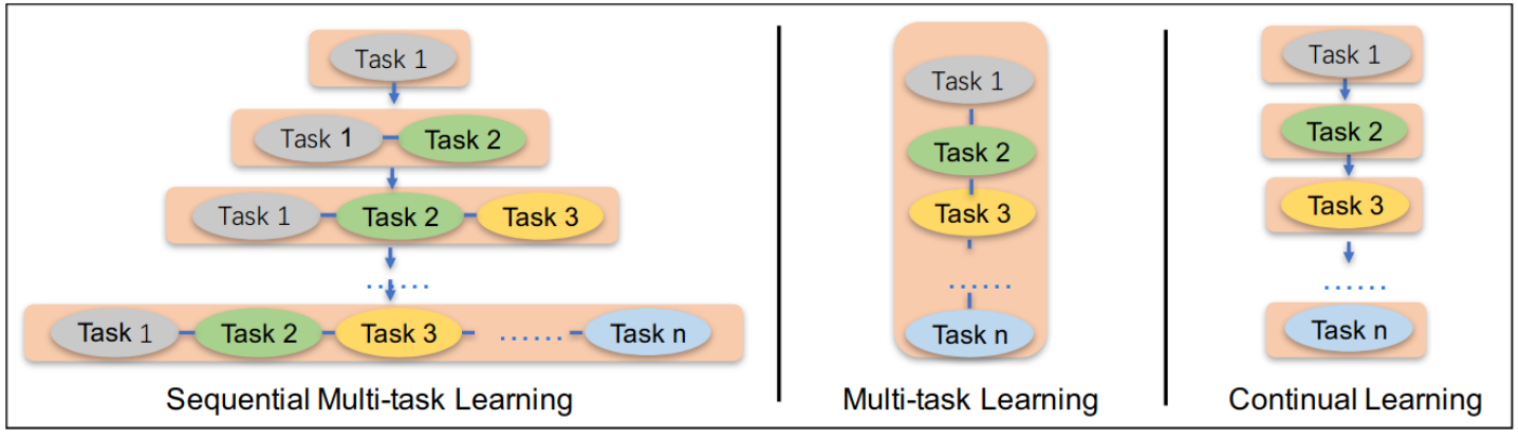
ERNIE 2.0: A Continual Pre-Training Framework for Language Understanding, https://arxiv.org/pdf/1907.12412.pdf

Continual Lifelong Learning with Neural Networks:A Review, https://arxiv.org/pdf/1802.07569.pdf

Language Models (sota) - ERNIE



Language Models (sota) - ERNIE



Pre-training method	Pre-training task	Training iterations (steps)				Fine-tuning result		
		Stage 1	Stage 2	Stage 3	Stage 4	MNLI	SST-2	MRPC
Continual Learning	Knowledge Masking	50k	-	-	-	77.3	86.4	82.5
	Capital Prediction	-	50k	-	-			
	Token-Document Relation	-	-	50k	-			
	Sentence Reordering	-	-	-	50k			
Multi-task Learning	Knowledge Masking	50k				78.7	87.5	83.0
	Capital Prediction	50k						
	Token-Document Relation	50k						
	Sentence Reordering	50k						
continual Multi-task Learning	Knowledge Masking	20k	10k	10k	10k	**79.0**	**87.8**	**84.0**
	Capital Prediction	-	30k	10k	10k			
	Token-Document Relation	-	-	40k	10k			
	Sentence Reordering	-	-	-	50k			

Table 7: The results of different methods of continual pre-training. We use knowledge masking, capital prediction, token-document relation and sentence reordering as our pre-training tasks. we sample 10% training data from our whole pre-training corpus. We train the model with 4 tasks altogether from scratch in multi-task learning method and train the model in 4 stages in other two learning methods. We train different tasks in different stages. The learning order of these tasks is the same as the above tasks listed. To compare the result fairly, each of these 4 tasks are updated in 50,000 steps . The size of pre-training model is same as ERNIE base. We choose MNLI-m, SST-2 and MRPC as our fine-tuning dataset. The fine-tuning result is average of five random start. the fine-tuning experiment set is same as Table 3.

Language Models (sota) - ERNIE

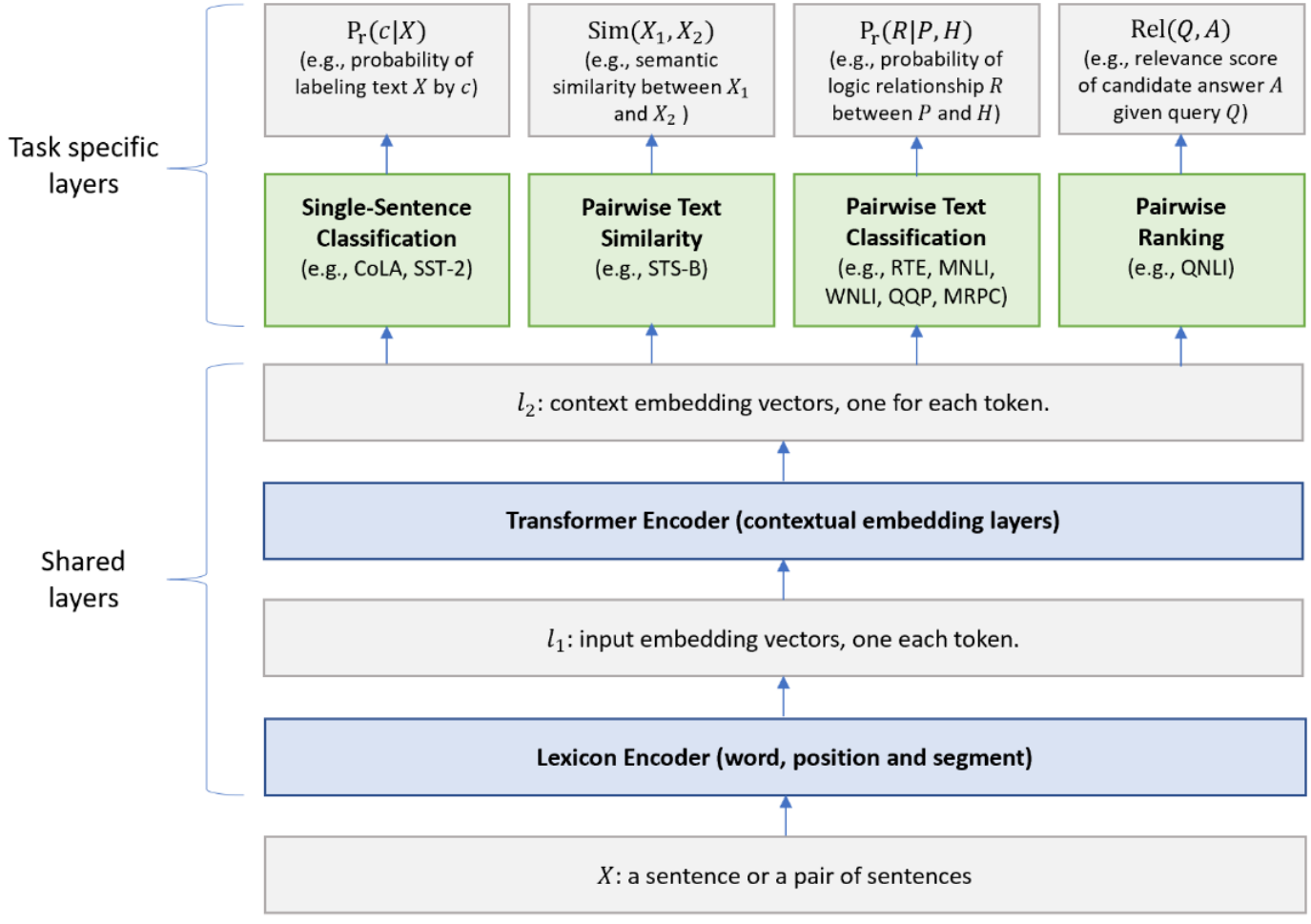
Corpus \ Task	Token-Level Loss			Sentence-Level Loss			
	Knowledge Masking	Capital Prediction	Token-Document Relation	Sentence Reordering	Sentence Distance	Discourse Relation	IR Relevance
Encyclopedia	✓	✓	✓	✓	✓	×	×
BookCorpus	✓	✓	✓	✓	✓	×	×
News	✓	✓	✓	✓	✓	×	×
Dialog	✓	✓	✓	✓	✓	×	×
IR Relevance Data	×	×	×	×	×	×	✓
Discourse Relation Data	×	×	×	×	×	✓	×

Table 1: The Relationship between pre-training task and pre-training dataset. We use different pre-training dataset to construct different tasks. A type of pre-trained dataset can correspond to multiple pre-training tasks.

Corpus Type	English(#tokens)	Chinese(#tokens)
Encyclopedia	2021M	7378M
BookCorpus	805M	-
News	-	1478M
Dialog	4908M	522M
IR Relevance Data	-	4500M
Discourse Relation Data	171M	1110M

Table 2: The size of pre-training datasets.

Language Models (sota) - MT-DNN



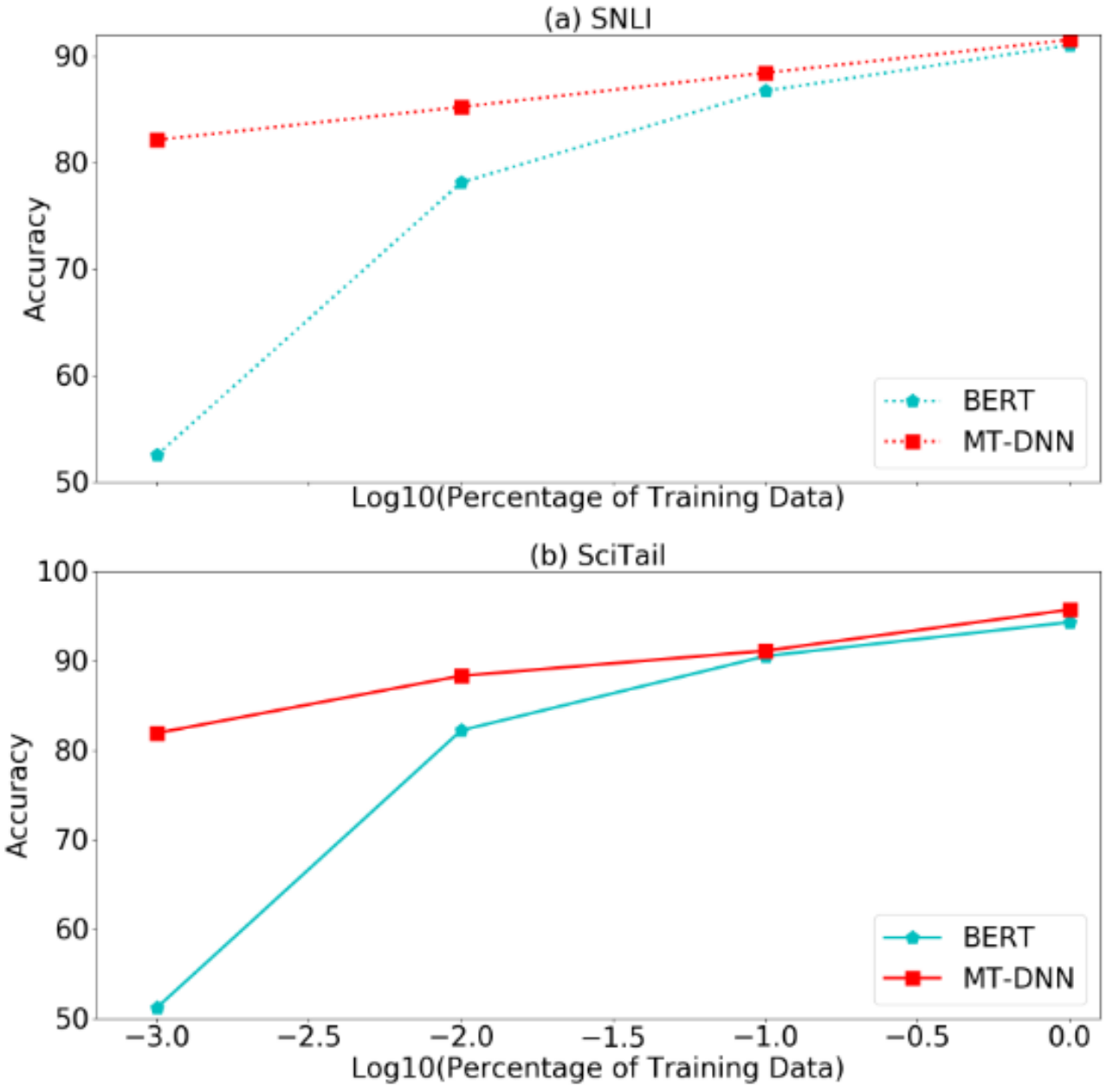
Multi-Task Deep Neural Networks for Natural Language Understanding, https://github.com/namisan/mt-dnn
https://www.aclweb.org/anthology/P19-1441/

Language Models (sota) - MT-DNN

Algorithm 1: Training a MT-DNN model.

Initialize model parameters Θ randomly.
Pre-train the shared layers (i.e., the lexicon encoder and the transformer encoder).
Set the max number of epoch: $epoch_{max}$.
//Prepare the data for T tasks.
for t *in* $1, 2, ..., T$ **do**
  Pack the dataset t into mini-batch: D_t.
end
for $epoch$ *in* $1, 2, ..., epoch_{max}$ **do**
  1. Merge all the datasets:
    $D = D_1 \cup D_2 ... \cup D_T$
  2. Shuffle D
  for b_t *in* D **do**
    //b_t is a mini-batch of task t.
    3. Compute loss : $L(\Theta)$
      $L(\Theta)$ = Eq. 6 for classification
      $L(\Theta)$ = Eq. 7 for regression
      $L(\Theta)$ = Eq. 8 for ranking
    4. Compute gradient: $\nabla(\Theta)$
    5. Update model: $\Theta = \Theta - \epsilon\nabla(\Theta)$
  end
end

Language Models (sota) - MT-DNN



Language Models (sota) - MT-DNN

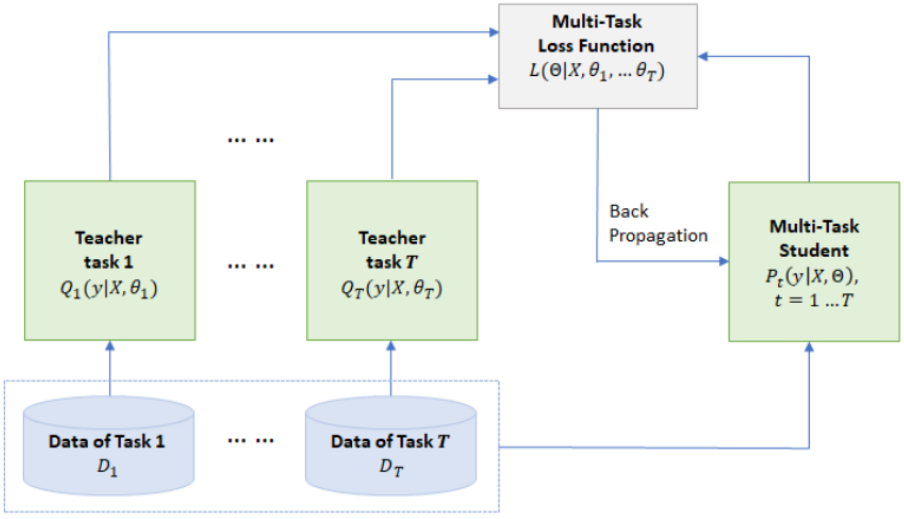


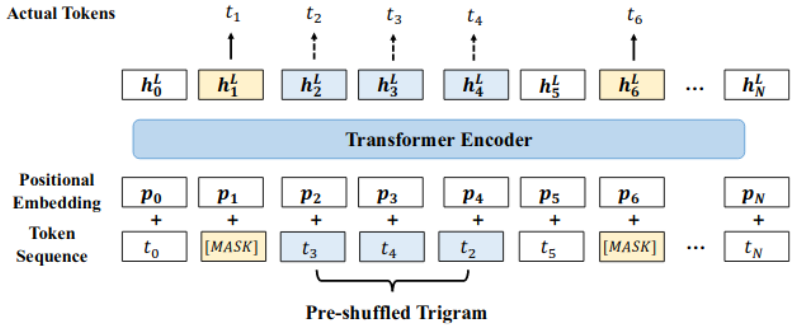
Figure 2: Process of knowledge distillation for multi-task learning. A set of tasks where there is task-specific labeled training data are picked. Then, for each task, an ensemble of different neural nets (teacher) is trained. The teacher is used to generate for each task-specific training sample a set of soft targets. Given the soft targets of the training datasets across multiple tasks, a single MT-DNN (student) is trained using multi-task learning and back propagation as described in Algorithm 1, except that if task t has a teacher, the task-specific loss in Line 3 is the average of two objective functions, one for the correct targets and the other for the soft targets assigned by the teacher.

Improving Multi-Task Deep Neural Networks via Knowledge Distillation for Natural Language Understanding, https://arxiv.org/pdf/1904.09482.pdf

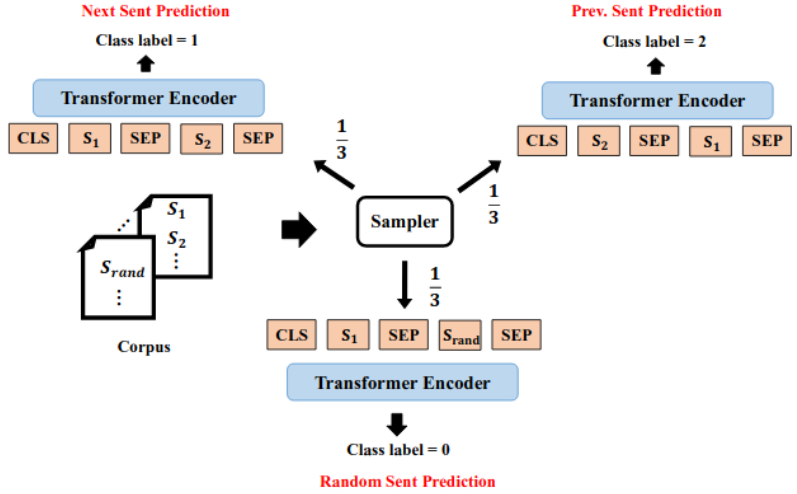
Language Models (sota) - MT-DNN

- Hybrid Neural Network Model for Commonsense Reasoning, https://arxiv.org/abs/1907.11983
- On the Variance of the Adaptive Learning Rate and Beyond, https://arxiv.org/abs/1908.03265
- SMART: Robust and Efficient Fine-Tuning for Pre-trained Natural Language Models through Principled Regularized Optimization, https://arxiv.org/abs/1911.03437

Language Models (sota) - StructBERT



(a) Word Structural Objective



(b) Sentence Structural Objective

Figure 1: Illustrations of the two new pre-training objectives

StructBERT: Incorporating Language Structures into Pre-training for Deep Language Understanding, https://arxiv.org/abs/1908.04577

Language Models (sota) - StructBERT

Task	CoLA (Acc)	SST-2 (Acc)	MNLI (Acc)	SNLI (Acc)	QQP (Acc)	SQuAD (F1)
StructBERTBase	**85.8**	**92.9**	**85.4**	91.5	**91.1**	**90.6**
-word structure	81.7	92.7	85.2	**91.6**	90.7	90.3
-sentence structure	84.9	**92.9**	84.1	91.1	90.5	89.1
BERTBase	80.9	92.7	84.1	91.3	90.4	88.5

Table 4: Ablation over the pre-training objectives using StructBERTBase architecture. Every result is the average score of 8 runs with different random seeds (the MNLI accuracy is the average score of the matched and mis-matched settings).

Batch size: 16, 24, 32; *Learning rate*: 2e-5, 3e-5, 5e-5; *Number of epochs*: 2, 3; *Dropout rate*: 0.05, 0.1

System	CoLA 8.5k	SST-2 67k	MRPC 3.5k	STS-B 5.7k	QQP 363k	MNLI 392k	QNLI 108k	RTE 2.5k	WNLI 634	AX	Average
Human Baseline	66.4	97.8	86.3/80.8	92.7/92.6	59.5/80.4	92.0/92.8	91.2	93.6	95.9	-	
BERTLarge [6]	60.5	94.9	89.3/85.4	87.6/86.5	72.1/89.3	86.7/85.9	92.7	70.1	65.1	39.6	80.5
BERT on STILTs [19]	62.1	94.3	90.2/86.6	88.7/88.3	71.9/89.4	86.4/85.6	92.7	80.1	65.1	28.3	82.0
SpanBERT [12]	64.3	94.8	90.9/87.9	89.9/89.1	71.9/89.5	88.1/87.7	94.3	79.0	65.1	45.1	82.8
Snorkel MeTaL [22]	63.8	96.2	91.5/88.5	90.1/89.7	73.1/89.9	87.6/87.2	93.9	80.9	65.1	39.9	83.2
MT-DNN++ [14]	65.4	95.6	91.1/88.2	89.6/89.0	72.7/89.6	87.9/87.4	95.8	85.1	65.1	41.9	83.8
MT-DNN ensemble [14]	65.4	96.5	92.2/89.5	89.6/89.0	73.7/89.9	87.9/87.4	96.0	85.7	65.1	42.8	84.2
StructBERTBase	57.2	94.7	89.9/86.1	88.5/87.6	72.0/89.6	85.5/84.6	92.6	76.9	65.1	39.0	80.9
StructBERTLarge	65.3	95.2	92.0/89.3	90.3/89.4	74.1/90.5	88.0/87.7	95.7	83.1	65.1	43.6	83.9
StructBERTLarge ensemble	68.6	95.2	92.5/90.1	91.1/90.6	74.4/90.7	88.2/87.9	95.7	83.1	65.1	43.9	84.5
XLNet ensemble [32]	67.8	96.8	93.0/90.7	91.6/91.1	74.2/90.3	90.2/89.8	98.6	86.3	90.4	47.5	88.4
RoBERTa ensemble [15]	67.8	96.7	92.3/89.8	92.2/91.9	74.3/90.2	90.8/90.2	98.9	88.2	89.0	48.7	88.5
Adv-RoBERTa ensemble	68.0	96.8	93.1/90.8	92.4/92.2	**74.8/90.3**	**91.1/90.7**	98.8	**88.7**	89.0	**50.1**	88.8
StructBERTRoBERTa ensemble	**69.2**	**97.1**	**93.6/91.5**	**92.8/92.4**	**74.4/90.7**	90.7/90.3	**99.2**	87.3	**89.7**	47.8	**89.0**

Table 1: Results of published models on the GLUE test set, which are scored by the GLUE evaluation server. The number below each task denotes the number of training examples. The state-of-the-art results are in bold. All the results are obtained from `https://gluebenchmark.com/leaderboard` (StructBERT submitted under a different model name ALICE).

Language Models (ideas) - ELECTRA

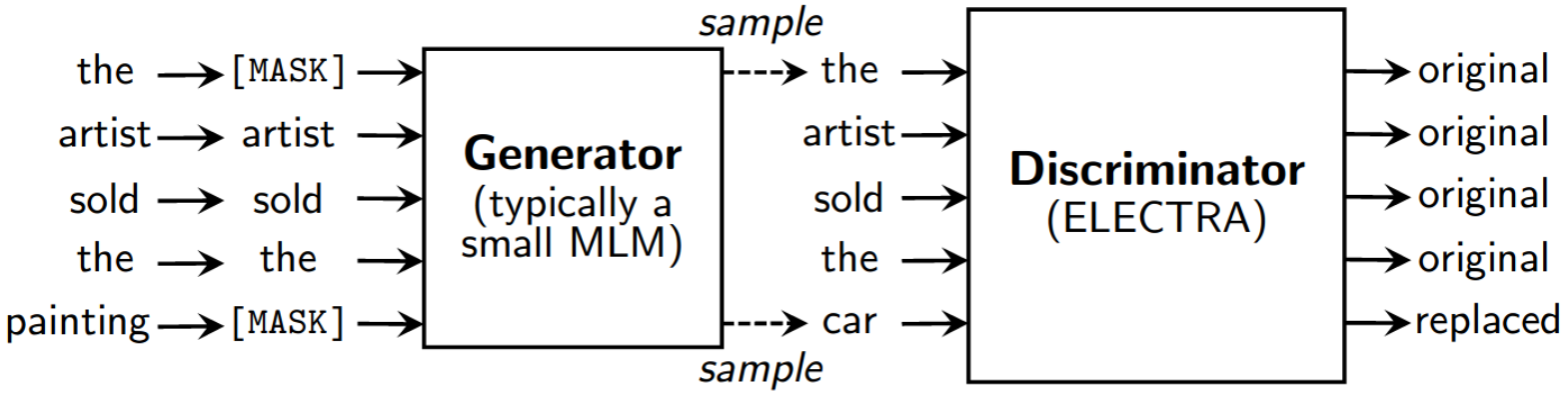


Figure 2: An overview of replaced token detection. The generator can be any model that produces an output distribution over tokens, but we usually use a small masked language model that is trained jointly with the discriminator. Although the models are structured like in a GAN, we train the generator with maximum likelihood rather than adversarially due to the difficulty of applying GANs to text. After pre-training, we throw out the generator and only fine-tune the discriminator on downstream tasks.

ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators, https://openreview.net/pdf?id=r1xMH1BtvB

Language Models (ideas) - ELECTRA

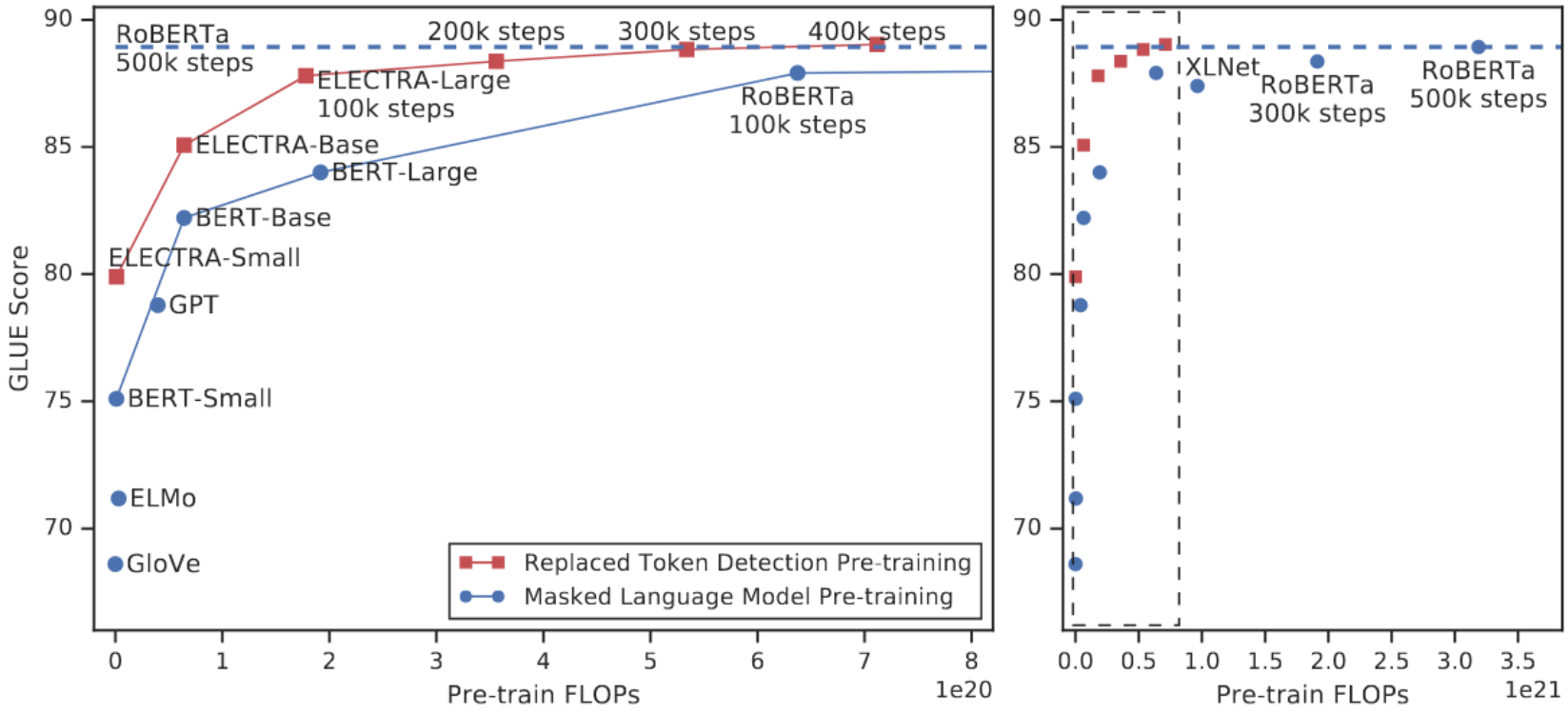


Figure 1: Replaced token detection pre-training consistently outperforms masked language model pre-training given the same compute budget. The left figure is a zoomed-in view of the dashed box.

Language Models (ideas) - ELECTRA

Model	Train / Infer FLOPs	Speedup	Params	Train Time + Hardware	GLUE
ELMo	3.3e18 / 2.6e10	19x / 1.2x	96M	14d on 3 GTX 1080 GPUs	71.2
GPT	4.0e19 / 3.0e10	1.6x / 0.97x	117M	25d on 8 P6000 GPUs	78.8
BERT-Small	1.4e18 / 3.7e9	45x / 8x	14M	4d on 1 V100 GPU	75.1
BERT-Base	6.4e19 / 2.9e10	1x / 1x	110M	4d on 16 TPUv3s	82.2
DistilBERT	– / 1.4e10	– / 2x	66M	–	77.8
ELECTRA-Small	1.4e18 / 3.7e9	45x / 8x	14M	4D on 1 V100 GPU	79.9
50% trained	7.1e17 / 3.7e9	90x / 8x	14M	2D on 1 V100 GPU	79.0
25% trained	3.6e17 / 3.7e9	181x / 8x	14M	1D on 1 V100 GPU	77.7
12.5% trained	1.8e17 / 3.7e9	361x / 8x	14M	12H on 1 V100 GPU	76.0
6.25% trained	8.9e16 / 3.7e9	722x / 8x	14M	6H on 1 V100 GPU	74.1
ELECTRA-Base	6.4e19 / 2.9e10	1x / 1x	110M	4D on 16 TPUv3s	85.1

Table 1: Comparison of small models on the GLUE dev set. BERT-Small/Base are our implementation and use the same hyperparameters as ELECTRA-Small/Base. Infer FLOPs assumes single length-128 input. Training times should be taken with a grain of salt as they are for different hardware and with sometimes un-optimized code.

Language Models (ideas) - ELECTRA

Model	Train FLOPs	Params	CoLA	SST	MRPC	STS	QQP	MNLI	QNLI	RTE	Avg.
BERT	1.9e20 (0.27x)	335M	60.6	93.2	88.0	90.0	91.3	86.6	92.3	70.4	84.0
XLNet	9.6e20 (1.3x)	360M	63.6	95.6	89.2	91.8	91.8	89.8	93.9	83.8	87.4
RoBERTa-100K	6.4e20 (0.90x)	356M	66.1	95.6	**91.4**	92.2	92.0	89.3	94.0	82.7	87.9
RoBERTa	3.2e21 (4.5x)	356M	68.0	**96.4**	90.9	**92.4**	92.2	90.2	**94.7**	86.6	88.9
BERT (ours)	7.1e20 (1x)	335M	67.0	95.9	89.1	91.2	91.5	89.6	93.5	79.5	87.2
ELECTRA	7.1e20 (1x)	335M	**69.3**	96.0	90.6	92.1	**92.4**	**90.5**	94.5	**86.8**	**89.0**
Test set results for models with standard single-task finetuning (no ensembling, task-specific tricks, etc.)											
BERT	1.9e20 (0.27x)	335M	60.5	94.9	85.4	86.5	89.3	86.7	92.7	70.1	83.3
SpanBERT	7.1e20 (1x)	335M	64.3	94.8	87.9	89.9	89.5	87.7	94.3	79.0	85.9
ELECTRA	7.1e20 (1x)	335M	**68.2**	**96.9**	**89.6**	**91.0**	**90.1**	**90.1**	**95.4**	**83.6**	**88.1**

Table 2: Comparison of large models on the GLUE dev and test sets. BERT dev results are from Clark et al. (2019). See Appendix B for some discussion on GLUE test set comparisons.

Language Models (ideas) - ELECTRA

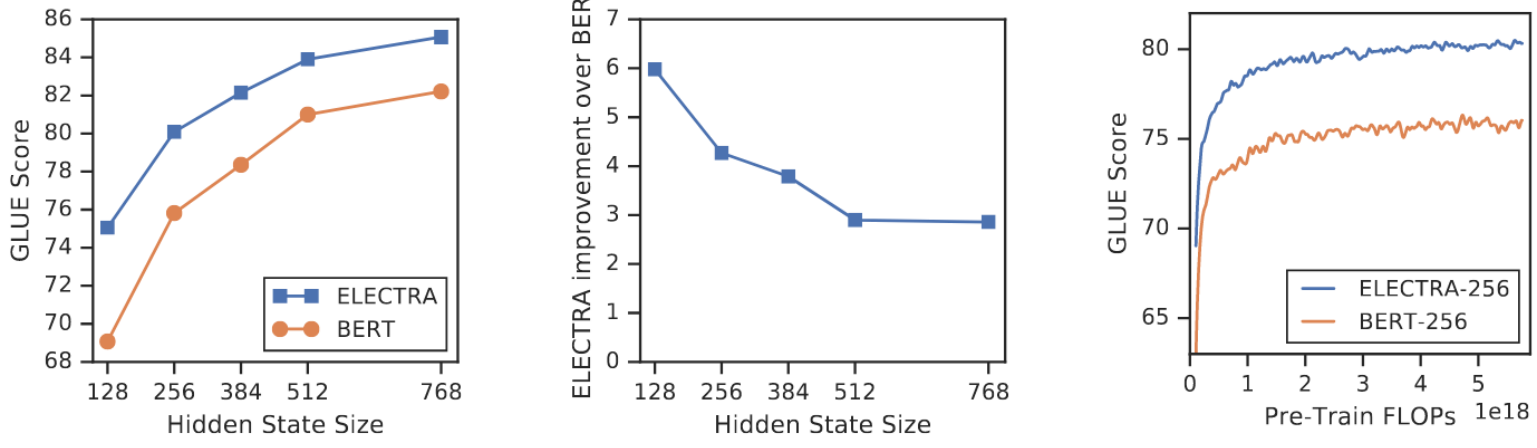


Figure 4: Left and Center: Comparison of BERT and ELECTRA for different model sizes. Right: A small ELECTRA model converges to higher downstream accuracy than BERT, showing the improvement comes from more than just faster training.

Thank you!