CS 589 Fall 2021 Lecture

Frontier topics

Monday 6:30-9:00
Babbio 122

photo: https://www.scubedstudios.com/information-retrieval/

Frontier topics

- Language model compression for language models
- More on AutoML
- Language models for text generation
- Testing of NLP models

CS 589 Fall 2021 Lecture

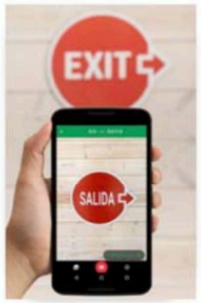
Knowledge distillation for language models

Monday 6:30-9:00
Babbio 122

source: "Compression of Deep Learning Models for NLP", Gupta et al. 2020

photo: https://www.scubedstudios.com/information-retrieval/

Deep learning on mobile/IoT devices



Phones

Drones

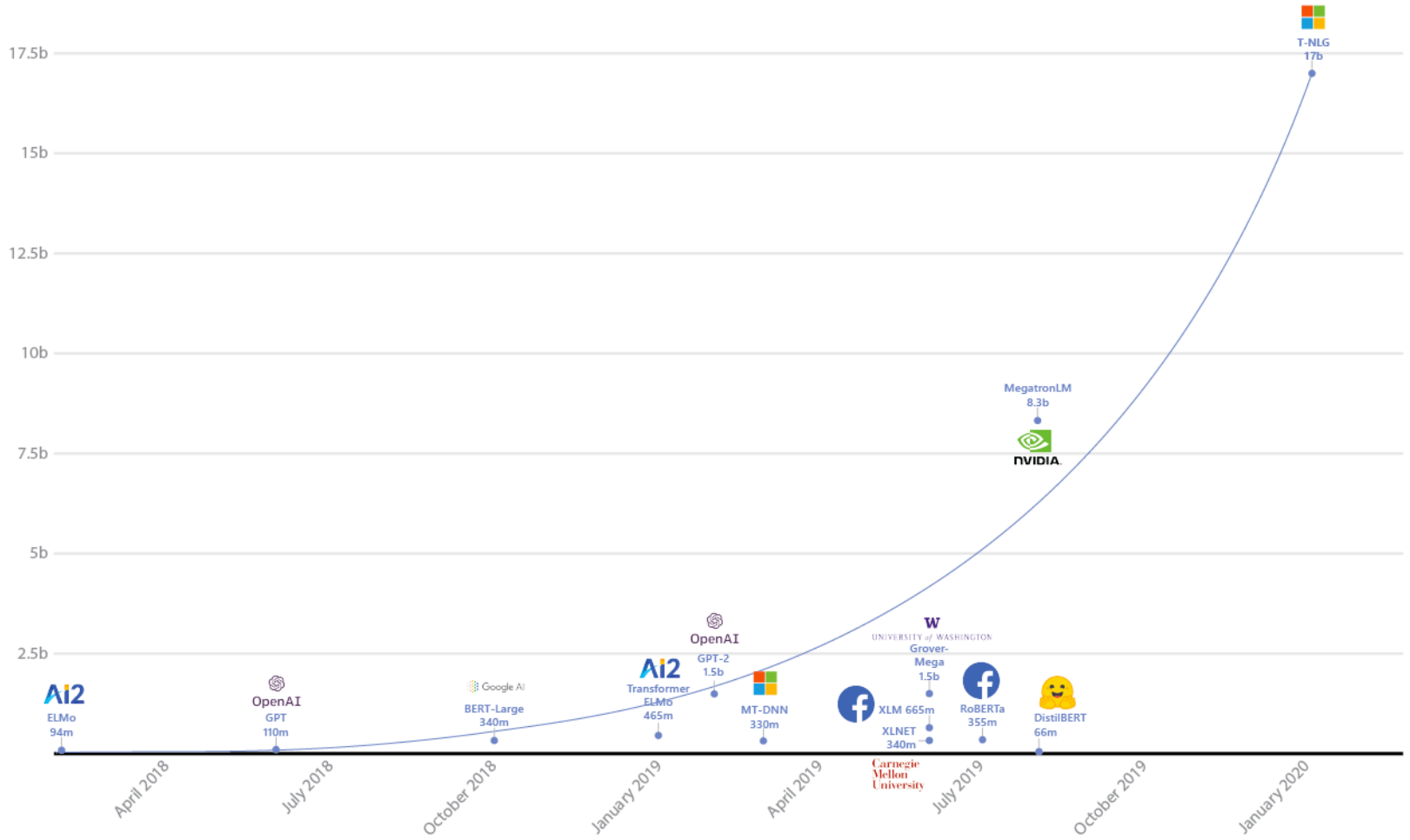
Robots

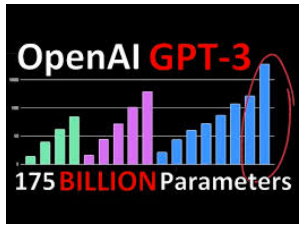
Glasses

Self Driving Cars

Battery Constrained!

Deep learning on mobile/IoT devices



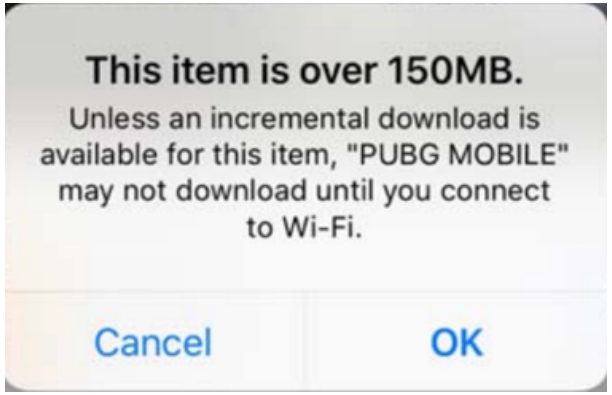


355 GPU year w/ V100

4.6 million $

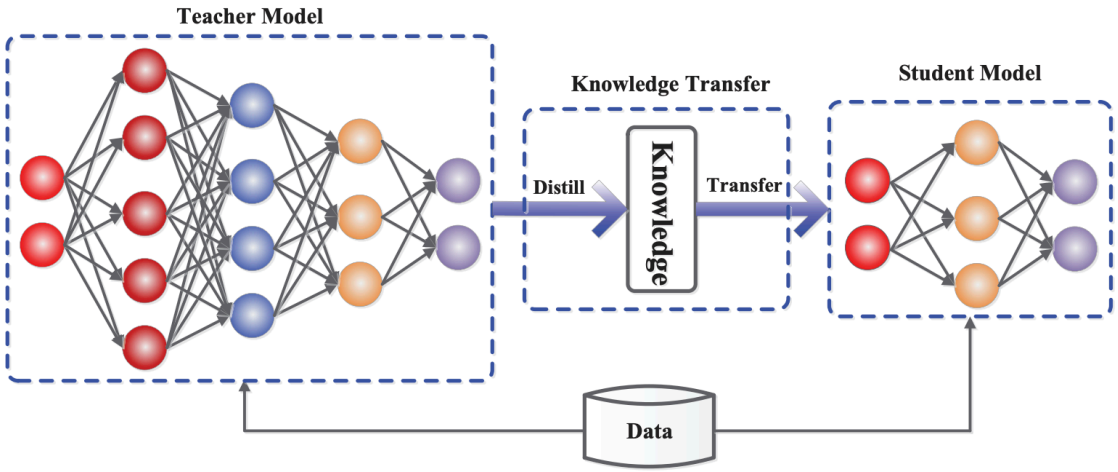
Deep learning on mobile/IoT devices

- GPU rams
  - Can only run models up to the RAM of GPU
- Latency requirements
  - Applications deploying to large number of users, e.g., Google
- Mobile-first computational requirements
  - Apps > 150MB will receive much more scrutiny
- Cloud computing
  - Cost for renting GPUs



Knowledge distillation: student-teacher framework

- Teacher model: the larger, cumbersome model
- Student model: the smaller model that learns from the teacher
- Knowledge transfer: train the student model to learn from the teacher's predictions on the data



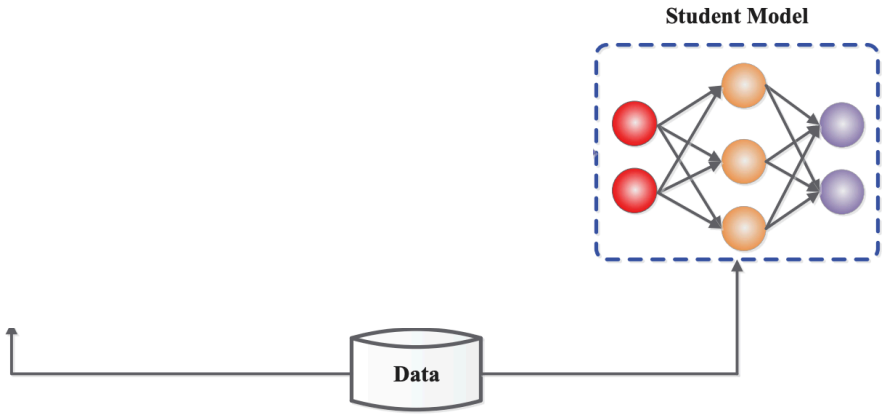

```
def train(self):
    """
    The real training loop.
    """
    if self.is_master:
        logger.info("Starting training")
    self.last_log = time.time()
    self.student.train()            # train the student parameter
    self.teacher.eval()             # use the teacher model for prediction
```

source: "Knowledge Distillation: A Survey", Gou et al. 2019

Is knowledge distillation is useful?

- Knowledge distillation is not necessarily useful
  - Without the teacher, we can still train the student model with the same data
  - Training the teacher takes extra time

- We have to show that the result of distilling a larger model to the smaller model > training the smaller model [Hinton et al. 2015]



Ditilling the knowledge in a neural network, Hinton et al. 2015

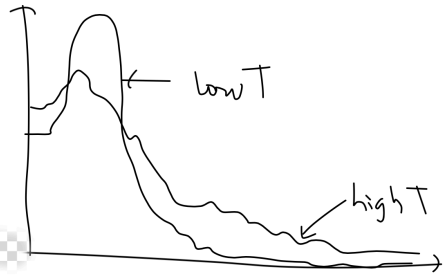
Knowledge distillation with the temperature T

- Hinton et al. first proposed the term of "knowledge distillation"

- Training the student network to match the teacher model's logits:
  - Student network: qi and zi
  - Teacher network: pi and vi
  - Minimizing the cross entropy loss between pi and qi:

$$C = -\Sigma_j p_j log(q_j)$$

$$q_j = \frac{exp(z_j/T)}{\Sigma_k exp(z_k/T)}$$



T: temperature parameter

Ditilling the knowledge in a neural network, Hinton et al. 2015

Effect of the temperature parameter T

- Compute the gradient of C w.r.t. z (derivation can be found on https://hackmd.io/@Raj-Ghugare/ryUfg7Qkw):

$$\frac{\partial C}{\partial z_i} = \frac{1}{T}(q_i - p_i) = \frac{1}{T}\left(\frac{e^{z_i/T}}{\sum_j e^{z_j/T}} - \frac{e^{v_i/T}}{\sum_j e^{v_j/T}}\right)$$

- When T is high (similar to optimizing the square loss):

$$\frac{\partial C}{\partial z_i} \approx \frac{1}{T}\left(\frac{1 + z_i/T}{N + \sum_j z_j/T} - \frac{1 + v_i/T}{N + \sum_j v_j/T}\right) \approx \frac{1}{NT^2}(z_i - v_i) \qquad \textit{when } \sum_j z_j = \sum_j v_j = 0$$

- When T is low:
  - Focus less on matching the very negative logits

- When the gap between model size is large, intermediate T is useful for ignoring the very negative logits

Knowledge distillation: Preliminary experiment on MNIST

- Data:
  - 60k training examples
- When the distilled net has > 300 units, all temperatures above 8 gave fairly similar results
- When radically reduced to 30 units, T from 2.5 to 4 works better
- Even if a digit is omitted in the data, the distilled model still learns to recognize it even if the learned bias on the cumbersome model is high

- **Cumbersome model**
  - 2 hidden layers
  - 1200 ReLU units
  - regularization
  - 67 errors

- **Small model**
  - 2 hidden layers
  - 800 ReLU units
  - no regularization
  - 146 errors
  - Distillation with T=20: 74 errors

Knowledge distillation: Experiments on ASR

- Data:
  - 700M training examples/2k hours
- Models:
  - Teacher model: a strong baseline used by Android voice search, with 85M parameters
  - Distillation: distilling 10xensembled teachers, i.e., 10 specialists model to a single student model

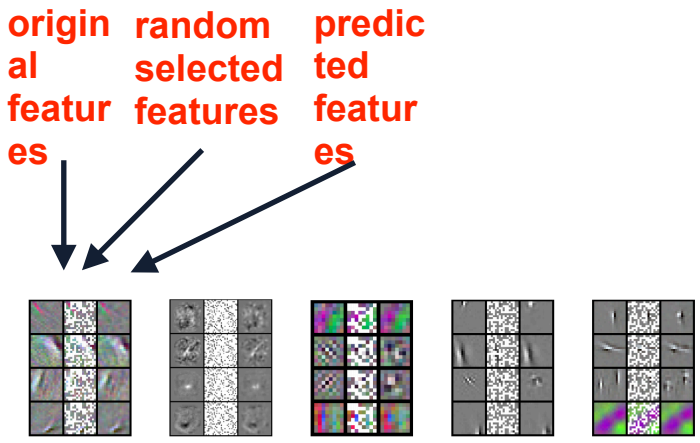
System	Test Frame Accuracy	WER
Baseline	58.9%	10.9%
10xEnsemble	61.1%	10.7%
Distilled Single model	60.8%	10.7%

The distilled single model outperforms the baseline with the same amount of parameters

Ditilling the knowledge in a neural network, Hinton et al. 2015

Explaining the why behind knowledge distillation

- How can smaller model learn more by distilling?
  - Significant redundancy in neural network models
- Representing the value of each pixel in the feature separately is redundant
  - Since it is highly likely that the value of a pixel will be equal to a weighted average of its neighbors
  - Predicting 95% parameters w/o loss in accuracy



(1) convnet trained on STL-10
(2) MLP
(3) convnet trained on cifar
(4) reconstruction ICA on Hyvarinen's dataset
(5) reconstruction ICA on STL-10

source: "Predicting parameters in deep learning.", Denil et al. 2013

Training students to learn beyond logits

- So far we have been using the logits (output layer of softmax) only
- However it is becoming common to have access to derivatives of the target output
- Optimizing the network to not only approximate the function's output but also the derivatives
  - Universal approximation theorem [Honik et al. 1991] in the Sobolev space, i.e., metric difference not only defined by value but also derivatives

$$\sum_{i=1}^{N} \ell(m(x_i|\theta), f(x_i))$$

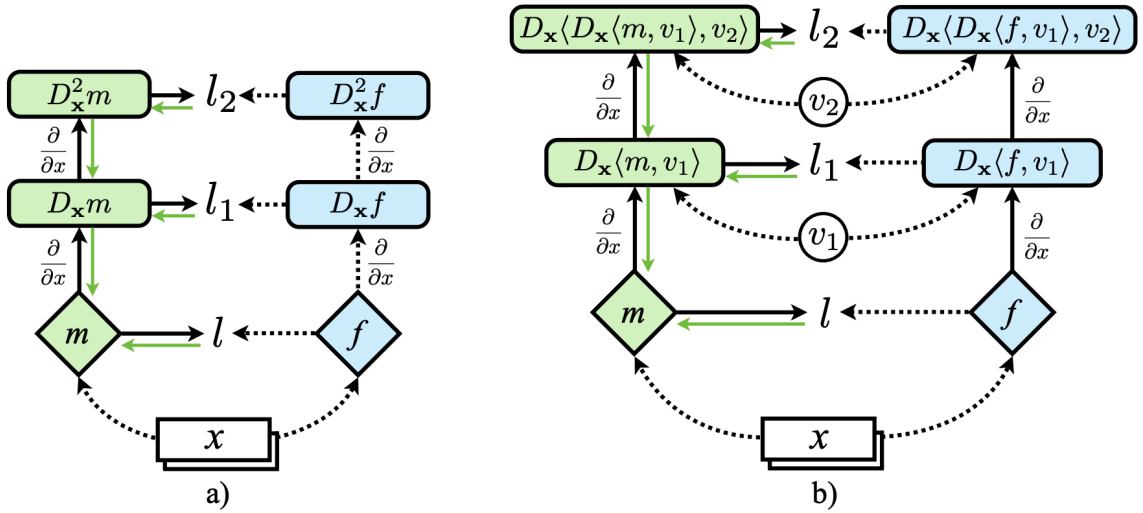
traditional training

$$\sum_{i=1}^{N} \left[\ell(m(x_i|\theta), f(x_i)) + \sum_{j=1}^{K} \ell_j \left(D^j_{\mathbf{x}} m(x_i|\theta), D^j_{\mathbf{x}} f(x_i) \right) \right]$$

Sobolev training

source: "Sobolev Training for Neural Networks", Czarnecki et al. 2015

Training students to learn beyond logits



Left: Sobolev Training of order 2. Diamond nodes m and f indicate parameterized functions, where m is trained to approximate f. Green nodes receive supervision.

Right: stochastic Sobolev Training. If f and m are multivariate functions, the gradients are Jacobian matrices. To avoid computing high dimensional objects, compute and fit their projections on a random vector vj

source: "Sobolev Training for Neural Networks", Czarnecki et al. 2015

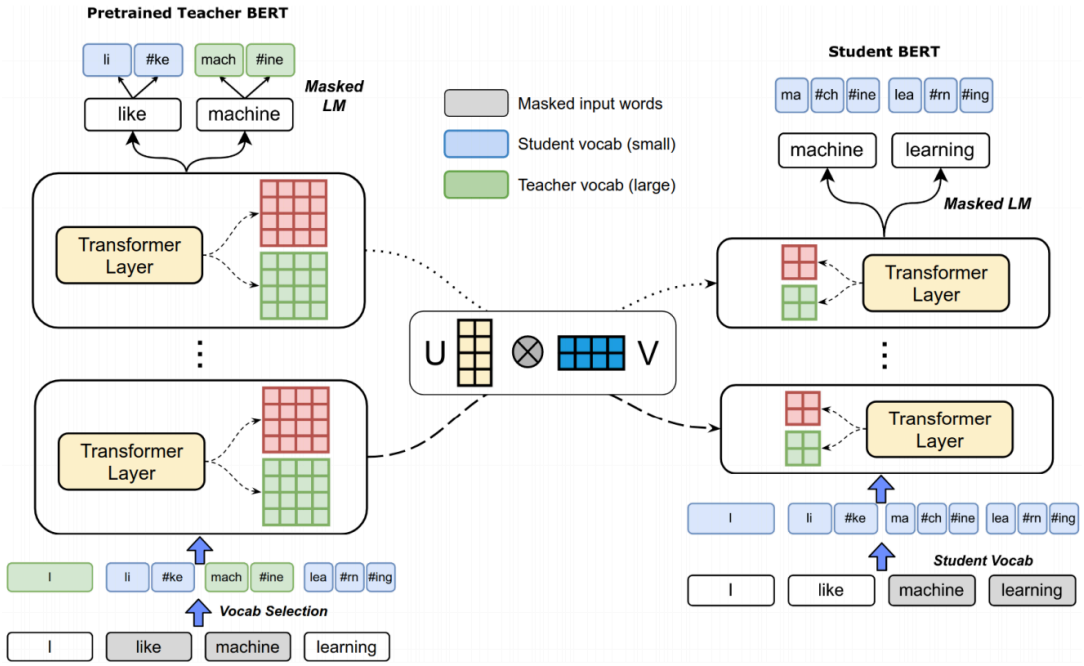
Reducing the embedding models for language models

- Distillation can help reduce the size of pretrained language models
- However, previous work has not considered reducing vocabulary size
  - The word embedding table of bert-base accounts for 21% of the model size, teacher: 30K, student: ~3k
- Can we reduce the size of vocabulary?
  - Poses a challenge, since existing trainings all assume the teacher and student share the same vocabulary
  - Use two strategies: **dual training** and **shared variable projection** to reduce the model to x61 in size

source: "Extreme language model compression with optimal subwords and shared project", Zhao et al. 2020

Reducing the embedding models for language models

- Teacher has a mixture of vocabulary from student and teacher
- Forces the model to learn to predict words from student vocab using teacher vocab, and vice versa
- Projecting model parameters into the same space to encourage alignment

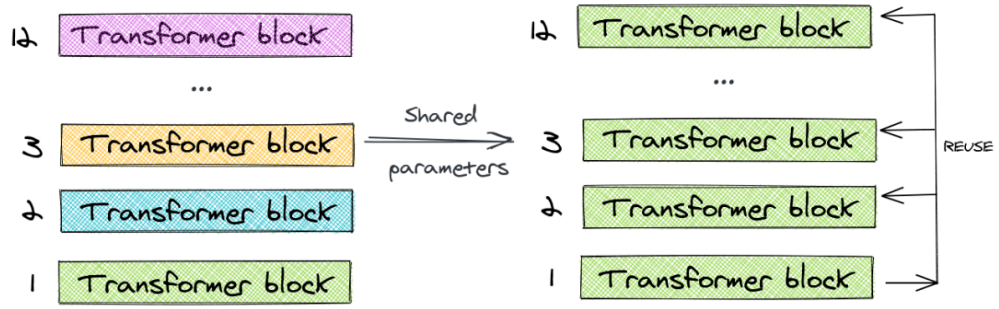


$$L_p^{\downarrow} = \sum_{\theta_t', \theta_s' \subset \theta_t, \theta_s} \left\| \mathbf{U}\theta_t'\mathbf{V} - \theta_s' \right\|^2$$

source: "Extreme language model compression with optimal subwords and shared project", Zhao et al. 2020

ALBERT: a lite bert for self-supervised learning of language representations

- Factorized embedding parameterization (horizontal)
  - Decomposing the large vocabulary embedding matrix into two small matrices, makes it easier to grow the hidden size from vocab size
- Cross-layer parameter sharing (vertical)
  - e.g., only sharing FFN parameters, or only sharing attention
- An ALBERT config similar to BERT-large has 18x fewer parameters and can train 1.7x times faster
- Achieving SOTA on GLUE, SQUAD 2.0, and RACE



source: "ALBERT: a lite bert for self-supervised learning of language representations", Lan et al. 2020

ALBERT: a lite bert for self-supervised learning of language representations

- The parameter reduction techniques also act as a form of regularization that stabilizes the training and helps with generalization.

Model		Parameters
BERT	base	108M
	large	334M
ALBERT	base	12M
	large	18M
	xlarge	60M
	xxlarge	235M

Models	MNLI	QNLI	QQP	RTE	SST	MRPC	CoLA	STS	WNLI	Avg
Single-task single models on dev										
BERT-large	86.6	92.3	91.3	70.4	93.2	88.0	60.6	90.0	-	-
XLNet-large	89.8	93.9	91.8	83.8	95.6	89.2	63.6	91.8	-	-
RoBERTa-large	90.2	94.7	**92.2**	86.6	96.4	**90.9**	68.0	92.4	-	-
ALBERT (1M)	90.4	95.2	92.0	88.1	96.8	90.2	68.7	92.7	-	-
ALBERT (1.5M)	**90.8**	**95.3**	**92.2**	**89.2**	**96.9**	**90.9**	**71.4**	**93.0**	-	-
Ensembles on test (from leaderboard as of Sept. 16, 2019)										
ALICE	88.2	95.7	**90.7**	83.5	95.2	92.6	**69.2**	91.1	80.8	87.0
MT-DNN	87.9	96.0	89.9	86.3	96.5	92.7	68.4	91.1	89.0	87.6
XLNet	90.2	98.6	90.3	86.3	96.8	93.0	67.8	91.6	90.4	88.4
RoBERTa	90.8	98.9	90.2	88.2	96.7	92.3	67.8	92.2	89.0	88.5
Adv-RoBERTa	91.1	98.8	90.3	88.7	96.8	93.1	68.0	92.4	89.0	88.8
ALBERT	**91.3**	**99.2**	90.5	**89.2**	**97.1**	**93.4**	69.1	**92.5**	**91.8**	**89.4**

source: "ALBERT: a lite bert for self-supervised learning of language representations", Lan et al. 2020

ALBERT: a lite bert for self-supervised learning of language representations

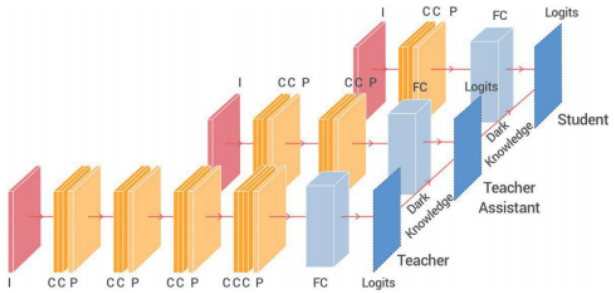
- The effect of different cross-layer parameter sharing strategies:
  - Shared attention has better performance than the other two

Model		Parameters	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg
ALBERT base E=768	all-shared	31M	88.6/81.5	79.2/76.6	82.0	90.6	63.3	79.8
	shared-attention	83M	89.9/82.7	80.0/77.2	84.0	91.4	67.7	81.6
	shared-FFN	57M	89.2/82.1	78.2/75.4	81.5	90.8	62.6	79.5
	not-shared	108M	90.4/83.2	80.4/77.6	84.5	92.8	68.2	82.3
ALBERT base E=128	all-shared	12M	89.3/82.3	80.0/77.1	82.0	90.3	64.0	80.1
	shared-attention	64M	89.9/82.8	80.7/77.9	83.4	91.9	67.6	81.7
	shared-FFN	38M	88.9/81.6	78.6/75.6	82.3	91.7	64.4	80.2
	not-shared	89M	89.9/82.8	80.3/77.3	83.2	91.5	67.9	81.6

source: "ALBERT: a lite bert for self-supervised learning of language representations", Lan et al. 2020

Knowledge distillation with teacher assistant

- Student network performance degrades when the gap between student and teacher is large
- Given a fixed student network, one cannot employ an arbitrarily large teacher
  - A teacher can effectively transfer its knowledge to students up to a certain size, not smaller
- TA models are distilled from the teacher, and the student is then only distilled from the TAs, i.e., chain distillation



source: "Improved Knowledge Distillation via Teacher Assistant", Mirzadeh et al. 2020

Reformer

- Replace dot-product attention by LSH, O(L[?]^2) -> O(LlogL), where L is the length of the sequence.

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

- **Hashing attention**. We mainly need to compute the QK^T, but note we mainly need to compute the softmax, which is dominated by the largest elements
  - For each query qi only need to focus on keys in K that are closest to qi

- Use reversible residual layers, which allows storing activations only once in the training process instead of N times (N is the number of layers)

source: "REFORMER: THE EFFICIENT TRANSFORMER", Kitaev et al. 2020

Reformer

- Performance of reformer on efficiency and accuracy

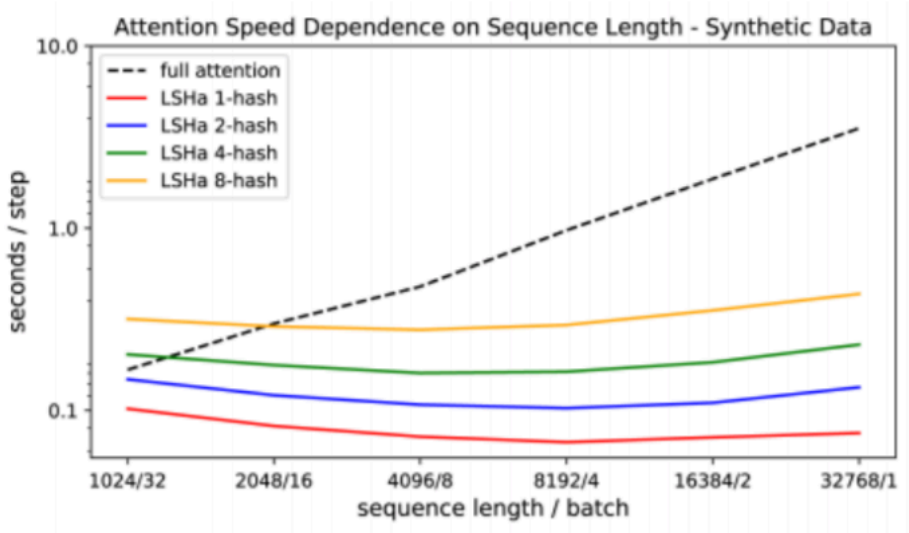


Table 4: BLEU scores on newstest2014 for WMT English-German (EnDe). We additionally report detokenized BLEU scores as computed by sacreBLEU (Post, 2018).

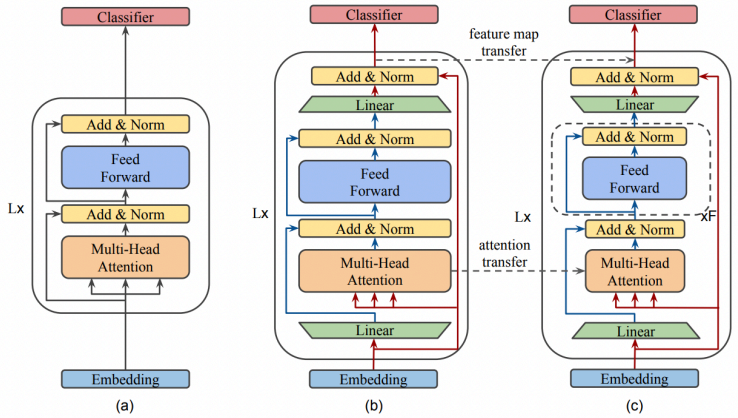
		sacreBLEU	
Model	BLEU	*Uncased*[3]	*Cased*[4]
Vaswani et al. (2017), base model	27.3		
Vaswani et al. (2017), big	28.4		
Ott et al. (2018), big	29.3		
Reversible Transformer (base, 100K steps)	27.6	*27.4*	*26.9*
Reversible Transformer (base, 500K steps, no weight sharing)	28.0	*27.9*	*27.4*
Reversible Transformer (big, 300K steps, no weight sharing)	29.1	*28.9*	*28.4*

source: "REFORMER: THE EFFICIENT TRANSFORMER", Kitaev et al. 2020

MobileBERT

- Task agnostic lightweight BERT model
  - Task-specific BERT: distill model on a specific task
  - However, this is inefficient because for another task, we need to retrain the model

- Distilling BERT to a deep, thin network
  - It is difficult to train a deep and thin network (bottleneck)
  - Teacher: BERT-large augmented with the thin network



(a) BERT; (b) Inverted-Bottleneck BERT (IB-BERT); and (c) MobileBERT

			$\text{BERT}_{\text{LARGE}}$	$\text{BERT}_{\text{BASE}}$	$\text{IB-BERT}_{\text{LARGE}}$	MobileBERT	$\text{MobileBERT}_{\text{TINY}}$
embedding		$h_{\text{embedding}}$	1024	768	128		
			no-op	no-op	3-convolution		
		h_{inter}	1024	768	512		
body	Linear	h_{input} h_{output}	$\begin{bmatrix} 1024 \\ 16 \\ 1024 \\ 1024 \\ 4096 \\ 1024 \end{bmatrix} \times 24$	$\begin{bmatrix} 768 \\ 12 \\ 768 \\ 768 \\ 3072 \\ 768 \end{bmatrix} \times 12$	$\begin{bmatrix} 512 \\ 1024 \\ 512 \\ 4 \\ 1024 \\ 1024 \\ 4096 \\ 1024 \\ 1024 \\ 512 \end{bmatrix} \times 24$	$\begin{bmatrix} 512 \\ 128 \\ 512 \\ 4 \\ 128 \\ 128 \\ 512 \\ 128 \times 4 \\ 128 \\ 512 \end{bmatrix} \times 24$	$\begin{bmatrix} 512 \\ 128 \\ 128 \\ 4 \\ 128 \\ 128 \\ 512 \\ 128 \times 2 \\ 128 \\ 512 \end{bmatrix} \times 24$
	MHA	h_{input} #Head h_{output}					
	FFN	h_{input} h_{FFN} h_{output}					
	Linear	h_{input} h_{output}					
#Params			334M	109M	293M	25.3M	15.1M

source: "MobileBERT: a Compact Task-Agnostic BERT for Resource-Limited Devices",

MobileBERT: Performance

	#Params	#FLOPS	Latency	CoLA 8.5k	SST-2 67k	MRPC 3.7k	STS-B 5.7k	QQP 364k	MNLI-m/mm 393k	QNLI 108k	RTE 2.5k	GLUE
ELMo-BiLSTM-Attn	-	-	-	33.6	90.4	84.4	72.3	63.1	74.1/74.5	79.8	58.9	70.0
OpenAI GPT	109M	-	-	47.2	93.1	87.7	84.8	70.1	80.7/80.6	87.2	69.1	76.9
$\text{BERT}_{\text{BASE}}$	109M	22.5B	342 ms	**52.1**	**93.5**	**88.9**	**85.8**	71.2	**84.6/83.4**	90.5	66.4	78.3
$\text{BERT}_{\text{BASE}}$-6L-PKD*	66.5M	11.3B	-	-	92.0	85.0	-	70.7	81.5/81.0	89.0	65.5	-
$\text{BERT}_{\text{BASE}}$-4L-PKD†*	52.2M	7.6B	-	24.8	89.4	82.6	79.8	70.2	79.9/79.3	85.1	62.3	-
$\text{BERT}_{\text{BASE}}$-3L-PKD*	45.3M	5.7B	-	-	87.5	80.7	-	68.1	76.7/76.3	84.7	58.2	-
$\text{DistilBERT}_{\text{BASE}}$-6L†	62.2M	11.3B	-	-	92.0	85.0		70.7	81.5/81.0	89.0	65.5	-
$\text{DistilBERT}_{\text{BASE}}$-4L†	52.2M	7.6B	-	32.8	91.4	82.4	76.1	68.5	78.9/78.0	85.2	54.1	-
TinyBERT*	14.5M	1.2B	-	43.3	92.6	86.4	79.9	**71.3**	82.5/81.8	87.7	62.9	75.4
$\text{MobileBERT}_{\text{TINY}}$	15.1M	3.1B	40 ms	46.7	91.7	87.9	80.1	68.9	81.5/81.6	89.5	65.1	75.8
MobileBERT	25.3M	5.7B	62 ms	50.5	92.8	88.8	84.4	70.2	83.3/82.6	90.6	66.2	77.7
MobileBERT w/o OPT	25.3M	5.7B	192 ms	51.1	92.6	88.8	84.8	70.5	84.3/**83.4**	**91.6**	**70.4**	**78.5**

source: "MobileBERT: a Compact Task-Agnostic BERT for Resource-Limited Devices",

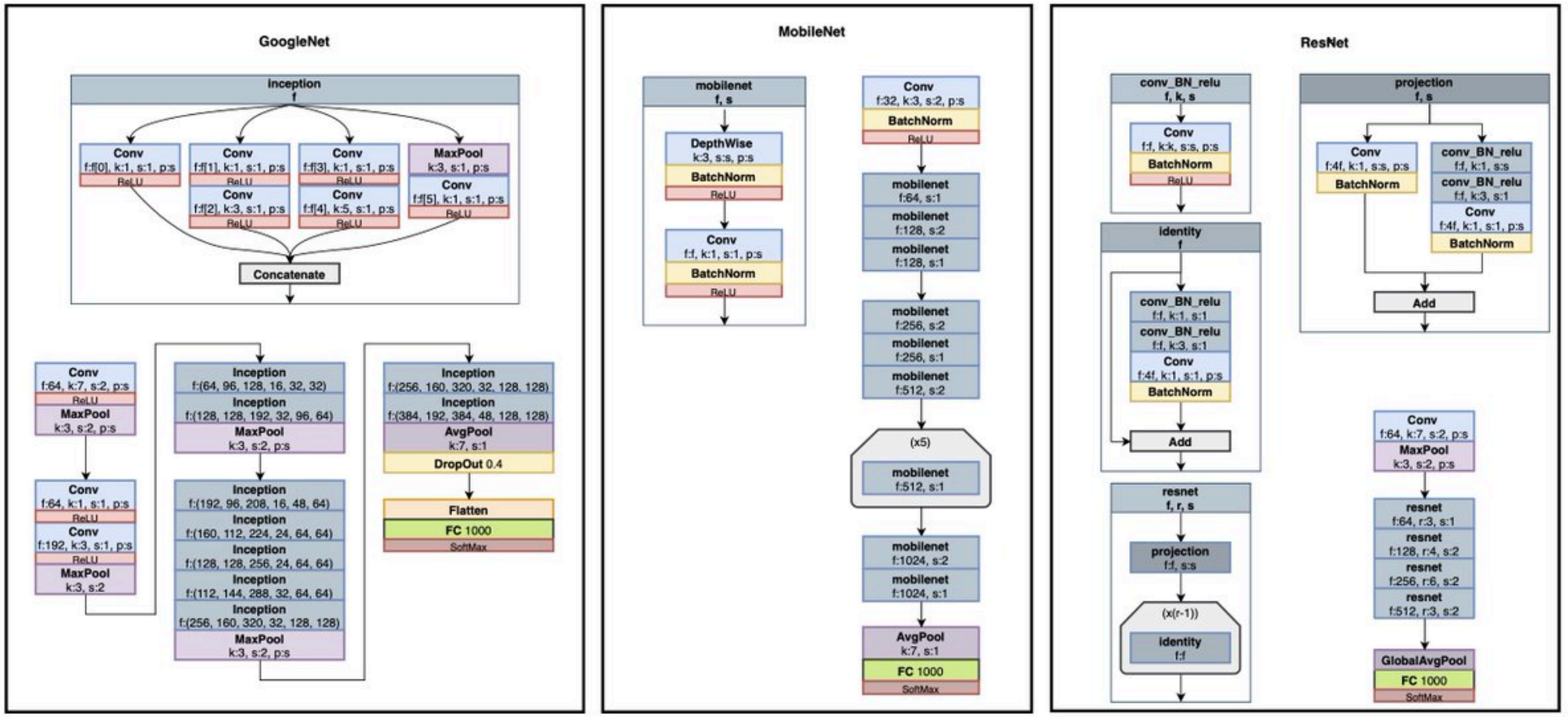
CS 589 Fall 2021 Lecture

NAS

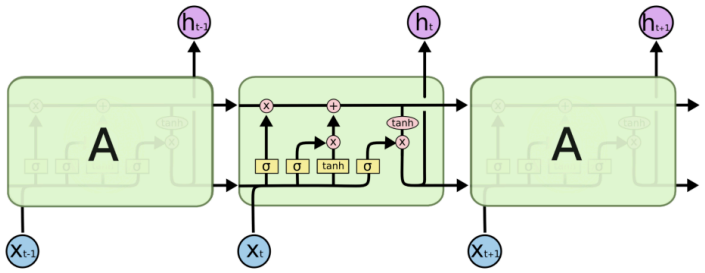
Monday 6:30-9:00
Babbio 122

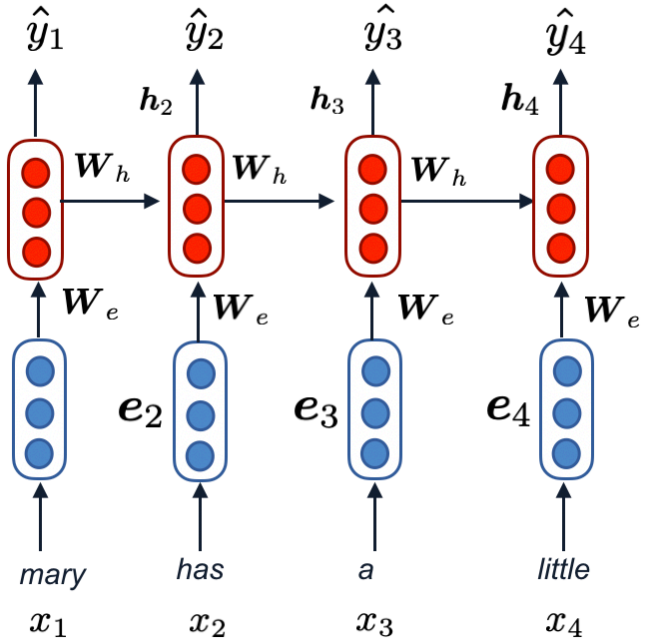
photo: https://www.scubedstudios.com/information-retrieval/

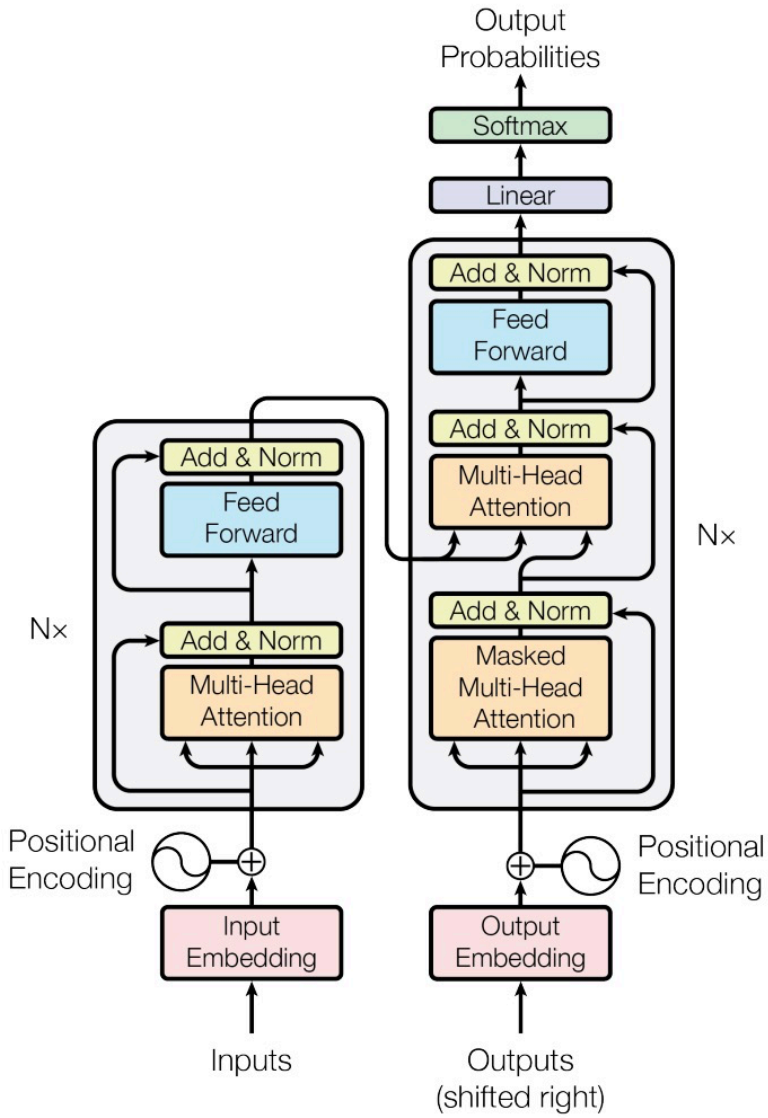
Neural network architectures



Neural network architectures

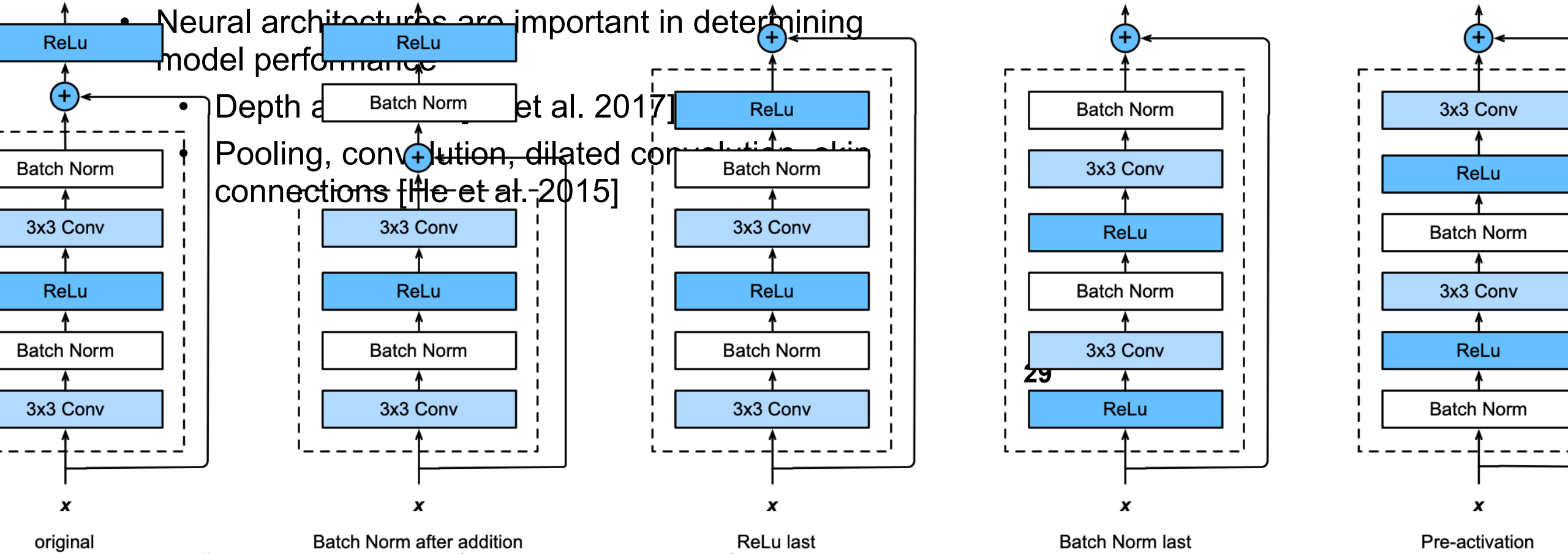






Neural network architectures

- Neural architectures are important in determining model performance
- Depth a[illegible]et al. 2017]
- Pooling, convolution, dilated co[illegible] connections [He et al. 2015]



source: "The Expressive Power of Neural Networks: A View from the Width", Lu et al. 2017

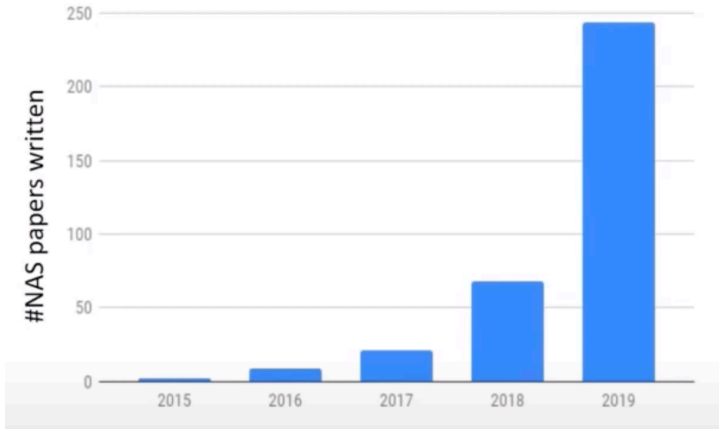
NAS: an exploding area

Neural Architecture Search: A Survey

Thomas Elsken THOMAS.ELSKEN@DE.BOSCH.COM
Bosch Center for Artificial Intelligence
71272 Renningen, Germany
and University of Freiburg

Jan Hendrik Metzen JANHENDRIK.METZEN@DE.BOSCH.COM
Bosch Center for Artificial Intelligence
71272 Renningen, Germany

Frank Hutter FH@CS.UNI-FREIBURG.DE
University of Freiburg
79110 Freiburg, Germany

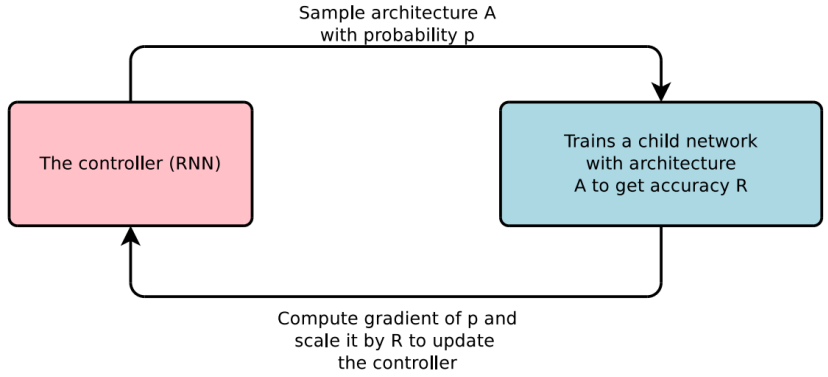


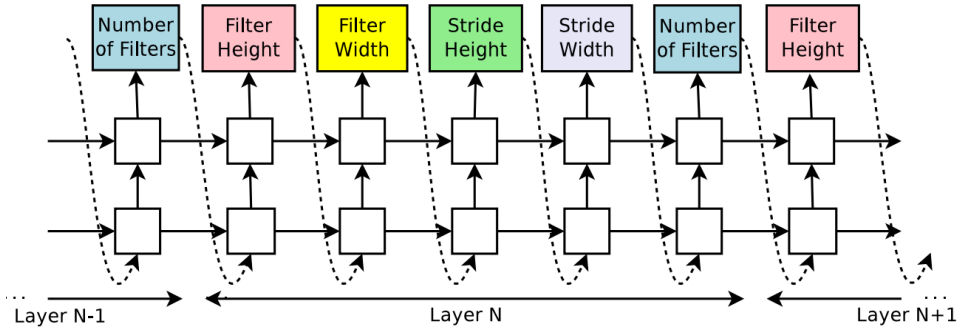
- automl.org
- “Neural architecture search with reinforcement learning” by Zoph & Le

source: Frank Hutter’s talk “Neural architecture search: coming of an age”

Neural architecture search with reinforcement learning

- Use an RNN to generate the neural architecture
- Use test accuracy as reward for controller actions with policy gradient
- On CIFAR-10, starting from scratch, can design a novel network architecture that rivals the best human-invented architecture





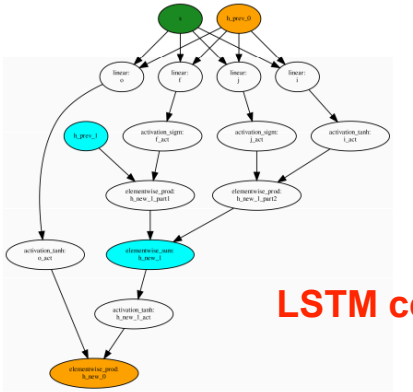
source: "Neural architecture search with reinforcement learning", So et al. 2017

DARTS: differentiable architecture search

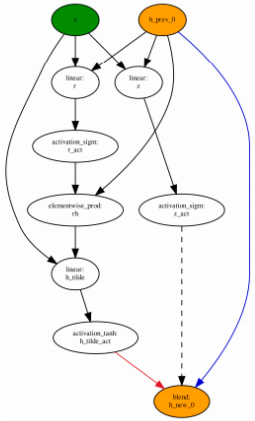
- Search space
  - The inner structure of a cell
  - Cell can be stacked into a CNN or recursively connected to an RNN

- Each intermediate node is computed based on all its predecessors

$$x^{(j)} = \sum_{i<j} o^{(i,j)}(x^{(i)})$$



LSTM cell



GRU cell

source: "DARTS: differentiable architecture search", Liu et al. 2019

DARTS: differentiable architecture search

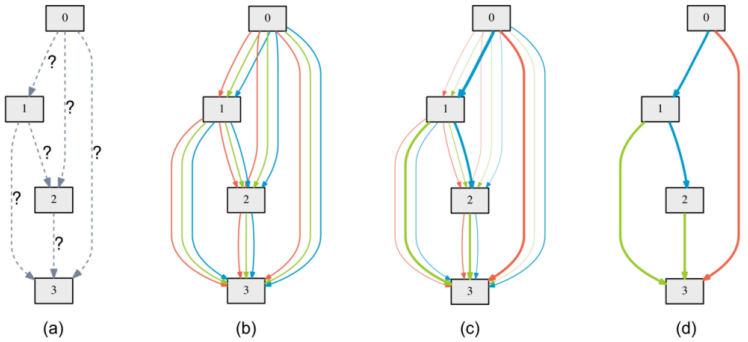
- Relax the **discrete problem** -> continuous
  - One-shot model with continuous architecture weight α (mixing weight) for each operator:

$$\bar{o}^{(i,j)}(x) = \sum_{o \in \mathcal{O}} \frac{\exp(\alpha_o^{(i,j)})}{\sum_{o' \in \mathcal{O}} \exp(\alpha_{o'}^{(i,j)})} o(x)$$

  - Solving a bi-level optimization problem

$$\begin{aligned} \min_{\alpha} \quad & \mathcal{L}_{val}(w^*(\alpha), \alpha) \\ \text{s.t.} \quad & w^*(\alpha) = \text{argmin}_w \ \mathcal{L}_{train}(w, \alpha) \end{aligned}$$

5x5 convol 3x3 convol maxpooling



Iterative updating alpha and w using gradient based methods

source: "DARTS: differentiable architecture search", Liu et al. 2019

Auto-Pytorch: Joint hyperparameter optimization and NAS

- Auto-pytorch-tabular: making deep learning easily accessible to tabular dataset for Pytorch

- Search space
  - 7 hyper parameters
  - shaped MLP + shaped ResNet

	Name	Range	log	type	cond.
Architecture	network type	[ResNet, MLPNet]	-	cat	-
	num layers (MLP)	[1, 6]	-	int	✓
	max units (MLP)	[64, 1024]	✓	int	✓
	max dropout (MLP)	[0, 1]	-	float	✓
	num groups (Res)	[1, 5]	-	int	✓
	blocks per group (Res)	[1, 3]	-	int	✓
	max units (Res)	[32, 512]	✓	int	✓
	use dropout (Res)	[F, T]	-	bool	✓
	use shake drop	[F, T]	-	bool	✓
	use shake shake	[F, T]	-	bool	✓
	max dropout (Res)	[0, 1]	-	float	✓
	max shake drop (Res)	[0, 1]	-	float	✓
Hyperparameters	batch size	[16, 512]	✓	int	-
	optimizer	[SGD, Adam]	-	cat	-
	learning rate (SGD)	[1e-4, 1e-1]	✓	float	✓
	L2 reg. (SGD)	[1e-5, 1e-1]	-	float	✓
	momentum	[0.1, 0.999]	-	float	✓
	learning rate (Adam)	[1e-4, 1e-1]	✓	float	✓
	L2 reg. (Adam)	[1e-5, 1e-1]	-	float	✓
	training technique	[standard, mixup]	-	cat	-
	mixup alpha	[0, 1]	-	float	✓
	preprocessor	[none, trunc. SVD]	-	cat	-
	SVD target dim	[10, 256]	-	int	✓

source: "Auto-PyTorch Tabular: Multi-Fidelity MetaLearning for Efficient and Robust AutoDL", Zimmer et al. 2020

Auto-Keras

- Proposes to use Bayesian optimization for neural architecture search
- Challenge applying BO to NAS as BO is designed only for the Euclidean space
  - If BO samples an architecture, it may cause input shape inconsistency
- Proposes an edit-distance neural network kernel

$$\kappa(f_a, f_b) = e^{-\rho^2(d(f_a, f_b))},$$

- Use dynamic programming-based morphing:

$$A_{i,j} = max[A_{i-1,j} + 1, A_{i,j-1} + 1, A_{i-1,j-1} + d_l(l_a^{(i)}, l_b^{(j)})],$$

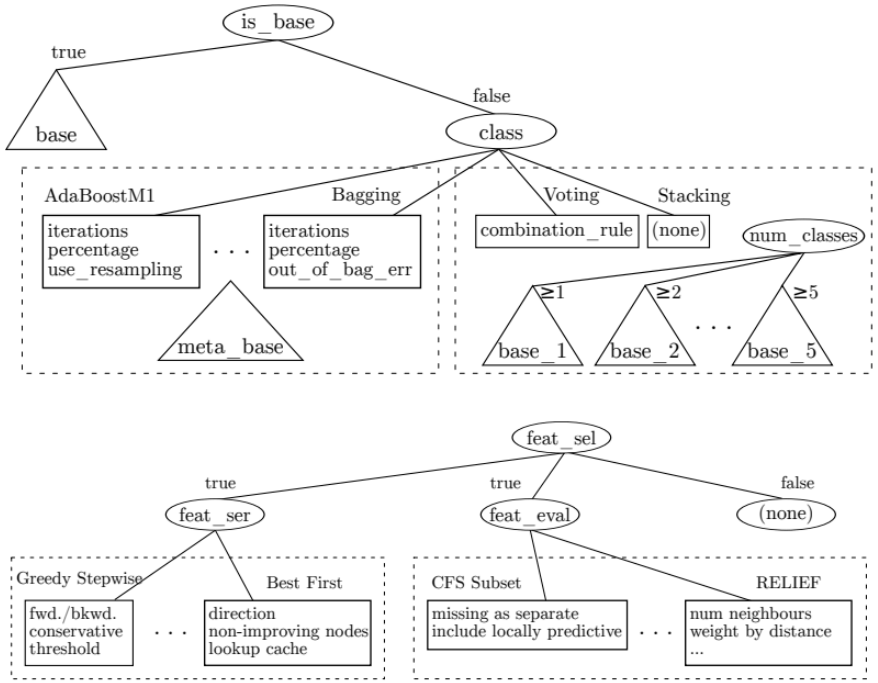
source: "Auto-Keras: An Efficient Neural Architecture Search System", Jin et al. 2019

Auto-Weka

- Combined algorithm selection and hyperparameter optimization problem (CASH) as computing

$$A^*_{\boldsymbol{\lambda}^*} \in \underset{A^{(j)} \in \mathcal{A}, \boldsymbol{\lambda} \in \boldsymbol{\Lambda}^{(j)}}{\operatorname{argmin}} \frac{1}{k} \sum_{i=1}^{k} \mathcal{L}(A^{(j)}_{\boldsymbol{\lambda}}, \mathcal{D}^{(i)}_{\text{train}}, \mathcal{D}^{(i)}_{\text{valid}})$$

- Uses two hyperparameter optimization algorithm that can handle hyperparameter search in a hierarchical space
  - Sequential model-based configuration
  - Tree-structured Parzen estimator



source: "Auto-WEKA: Combined Selection and Hyperparameter Optimization of Classification Algorithms", Thornton et al. 2013

Low-cost Proxy for NAS

- NAS is expensive
- Use a low-cost proxy for predicting and reducing the cost of NAS
  - Mean gram matrix can serve as the proxy [Yuan et al. 2016]
- Use gram matrix as the proxy to eliminate most candidate architecture and only use the top-k

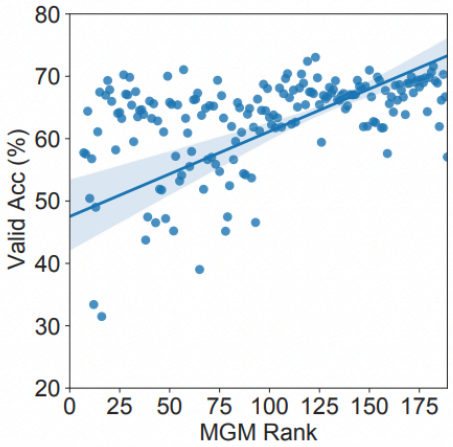
Algorithm 1 KNAS Algorithm

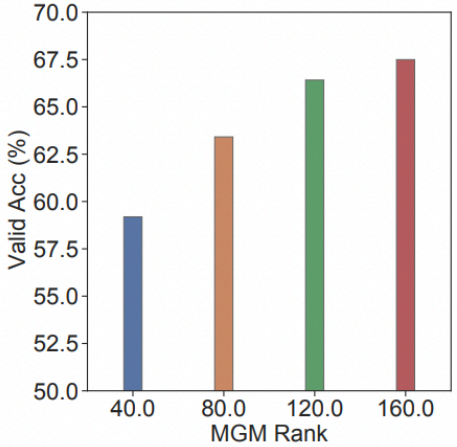
Require: Search space $\mathcal{S}$, training set $\mathcal{D}_t$, validation set $\mathcal{D}_v$
Initialize: max_iteration = M
Initialize candidate set $\mathcal{C}$ = []
for search_iteration in 1, 2, ..., max_iteration **do**
  Randomly sample an architecture s from $\mathcal{S}$
  Compute MGM based on Eq. 15
  $\mathcal{C}$.append(s, MGM)
  update $\mathcal{C}$ to ensure only top-k architectures are kept
end for
A^* = top_1($\mathcal{C}$,$\mathcal{D}_t$,$\mathcal{D}_v$) # Choose the best architecture based on validation performance.
return A^*

source: "KNAS: Green Neural Architecture Search", Xu et al. 2019

Low-cost Proxy for NAS

- NAS is expensive
- Use a low-cost proxy for predicting and reducing the cost of NAS
  - Mean gram matrix can serve as the proxy [Yuan et al. 2016]
- Use gram matrix as the proxy to eliminate most candidate architecture and only use the top-k





source: "KNAS: Green Neural Architecture Search", Xu et al. 2019

Using supernet to reduct the cost

- NAS is expensive
- We can train a supernet, and search within the subset of the supernet for NAS
- Only train one time, weight is shared across all subsets

source: “HOW DOES WEIGHT SHARING HELP IN NEURAL ARCHITECTURE SEARCH?”, Zhang et al. 2019

CS 589 Fall 2021 Lecture

Other language models

Monday 6:30-9:00
Babbio 122

photo: https://www.scubedstudios.com/information-retrieval/

RoBERTa [Liu et al. 2019]

- RoBERTa: A Robustly Optimized BERT Pre-training Approach (Liu et al, University of Washington and Facebook, 2019)

- An ablation study on BERT for design choices:
  - **Static vs. dynamic masking**
  - **Sentence completeness**:
    - segment pair + NSP, sentence pair + NSP, full sentences, doc sentences
  - **Larger batch sizes**
    - batch size = 256, 2K, 8K
  - **Larger BPE vocab**
    - roberta -> “ro” “bert” “a##”
    - 50K subword units

RoBERTa fine tuning on GLUE

- Two settings for fine-tuning GLUE

- Single task on dev
  - batch size {16, 32}
  - lr = {1e-5, 2e-5, 3e-5}
  - linear warmup for the first 6% of steps

- Ensemble on test
  - For RTE, STS-B & MRPC, helpful to fine tune starting from model fine-tuned on MNLI
  - Run 15 different seeds for the selected hyperparam, and ensemble top 7 model based on dev set metrics

RoBERTa results

Model	SQuAD 1.1		SQuAD 2.0	
	EM	F1	EM	F1
Single models on dev, w/o data augmentation				
$\text{BERT}_{\text{LARGE}}$	84.1	90.9	79.0	81.8
$\text{XLNet}_{\text{LARGE}}$	**89.0**	94.5	86.1	88.8
RoBERTa	88.9	**94.6**	**86.5**	**89.4**
Single models on test (as of July 25, 2019)				
$\text{XLNet}_{\text{LARGE}}$			86.3†	89.1†
RoBERTa			86.8	89.8
XLNet + SG-Net Verifier			**87.0**†	**89.9**†

	MNLI	QNLI	QQP	RTE	SST	MRPC	CoLA	STS	WNLI	Avg
Single-task single models on dev										
$\text{BERT}_{\text{LARGE}}$	86.6/-	92.3	91.3	70.4	93.2	88.0	60.6	90.0	-	-
$\text{XLNet}_{\text{LARGE}}$	89.8/-	93.9	91.8	83.8	95.6	89.2	63.6	91.8	-	-
RoBERTa	**90.2/90.2**	**94.7**	**92.2**	**86.6**	**96.4**	**90.9**	**68.0**	**92.4**	**91.3**	-

XLNet [Yang et al. 2019, CMU and Google]

- Innovation #1: Relative position embeddings
  - Sentence: John ate a hot dog
  - Relative attention: “How much should dog attend to hot (in any position) and how much should dog attend to the previous word?”
- Innovation #2: Randomly permute the training sequences
- Training with more data and larger models..

Model	MNLI	QNLI	QQP	RTE	SST-2	MRPC	CoLA	STS-B
Single-task single models on dev								
BERT [2]	86.6/-	92.3	91.3	70.4	93.2	88.0	60.6	90.0
RoBERTa [21]	90.2/90.2	94.7	92.2	**86.6**	96.4	**90.9**	68.0	92.4
XLNet	**90.8/90.8**	**94.9**	**92.3**	85.9	**97.0**	90.8	**69.0**	**92.5**

CS 589 Fall 2021 Lecture

Language model for Seq2seq

Monday 6:30-9:00
Babbio 122

photo: https://www.scubedstudios.com/information-retrieval/

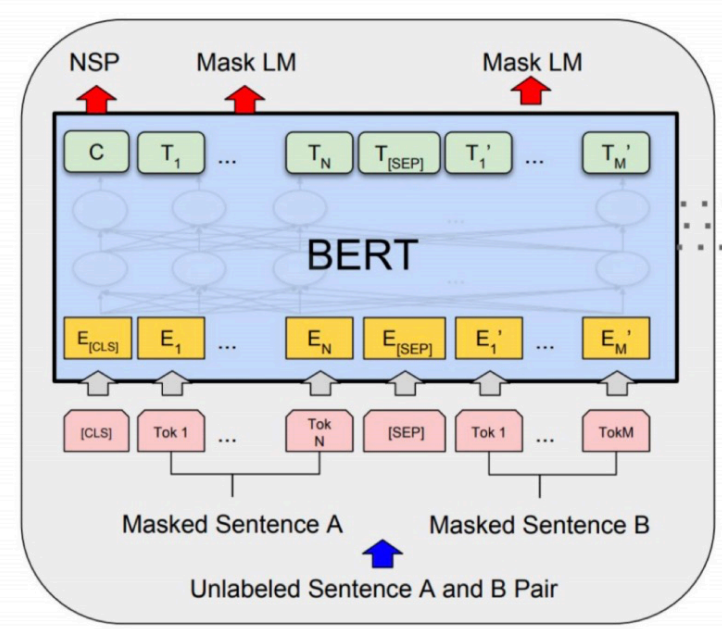
BERT: Bidirectional encoder using Transformers

- Architecture:
  - 12 layers of Transformer architecture

- Training objective:
  - Masked language model (randomly masking 15% of tokens, predicting the token using **both the left and right tokens**)
  - Next sentence prediction

- Pre-training data: the BooksCorpus (800M words) (Zhu et al., 2015) and English Wikipedia (2,500M words)

- BERT architecture is not designed for seq2seq generation task: **why not?**

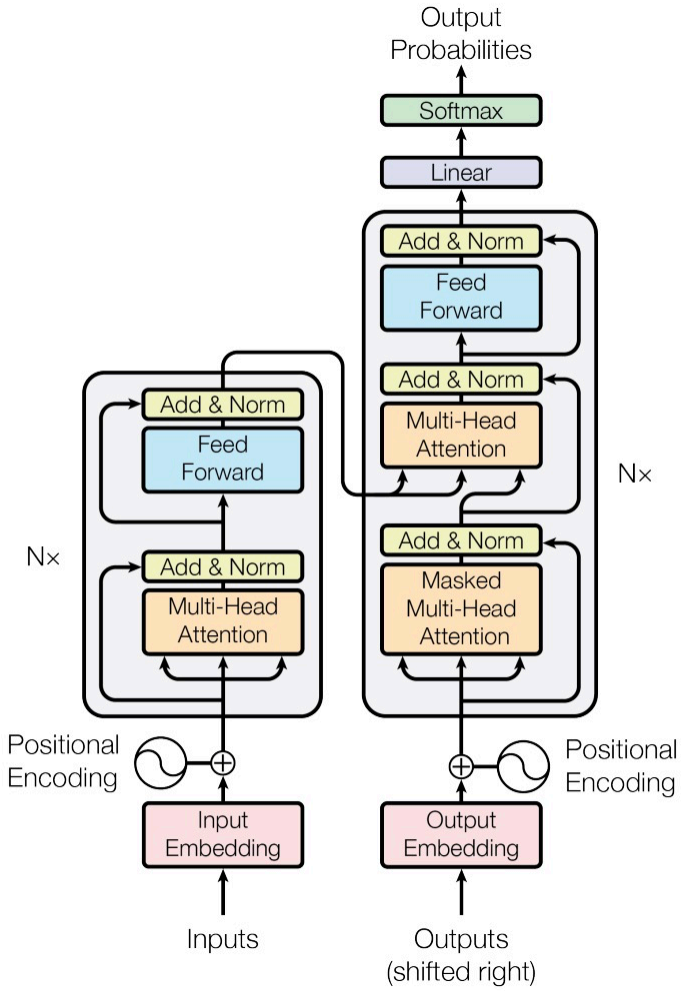


Transformer

- **Non-recurrent** sequence to sequence encoder-decoder model
- Encoder and decoder architectures
- Multi-head self attention [Lin et al. 2017]

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

- Position encoding [Gehring et al. 2017]

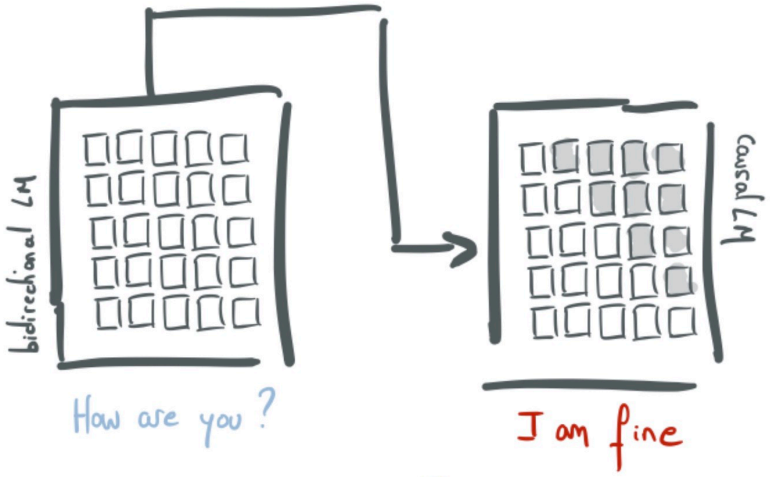


Attention is all you need. Vaswani et al. 2017

GPT: Left-to-right decoder using the Transformer arch.

- The GPT architecture also uses the transformer architecture
  - But it uses the decoder (masked multi-head attention) to allow autoregressive generation for seq2seq tasks
  - Masked multi-head attention prevents decoder from attending to right tokens

- Framework:
  - Unsupervised pre-training: using the BooksCorpus + 1B word benchmark
  - Fine tuning: qa, common sense reasoning, text entailment



Improving Language Understanding by Generative Pre-Training. Radford et al. 2018

GPT2: Towards zero-shot learning for NLP tasks

- GPT2 framework:
  - Uses the similar model as GPT [Radford et al. 2018], expanding the model size and vocabulary size
  - Engineering the dataset (for pre-training): scraped all outbound links from Reddit, didn't use CommonCrawl (1 trillion word) due to quality issue
  - Data processing: 8 million documents for a total of 40 GB of text w/ deduplication and simple processing, removal of Wiki content

- Performance:
  - Achieves state of the art results on 7 out of 8 tested language modeling datasets in a zero-shot setting
  - On CoQA dataset, outperforms 3/4 baselines wo supervised training (GPT2: 55 F1, BERT: 89 F1)

Language Models are Unsupervised Multitask Learners. Radford et al. 2018

GPT3: Achieving few-shot learning

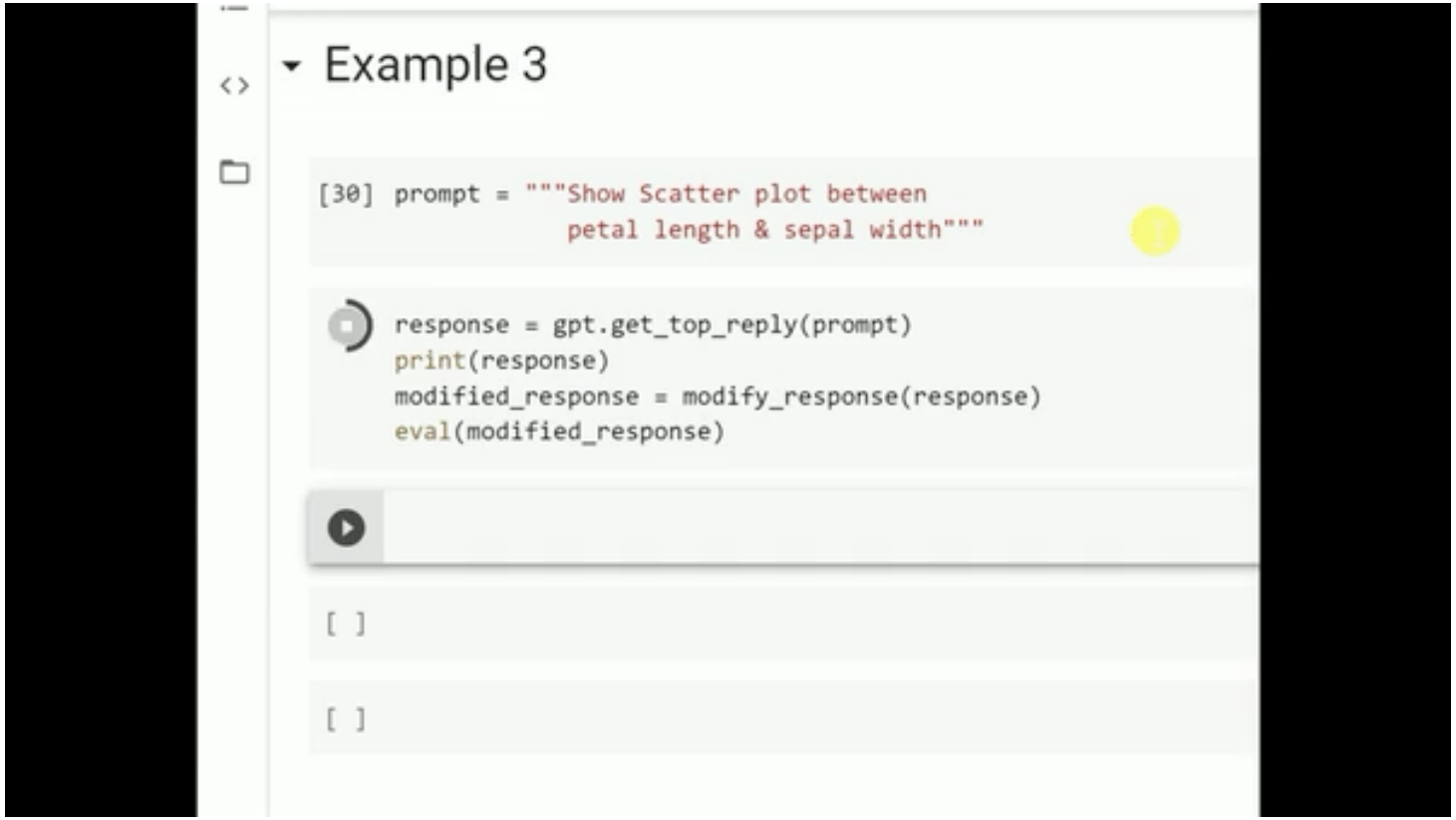
- Few-shot learning:
  - Human can learn a task with as small as a few examples, e.g., teaching babies to recognize shape
  - Can also achieve few-shot performance when we only provide a few examples?

- GPT3 training data:
  - Upon identifying the quality issue of CommonCrawl in GPT-2 [Radford et al. 2018], performed cleaning, deduplication (45TB before, 570GB after)
  - WebText, English Wiki, two internet-based books

- GPT3 performance:
  - Achieves strong performance on many NLP datasets, including translation, question-answering, and cloze tasks

Language Models are Few-shot Learners. Brown et al. 2018

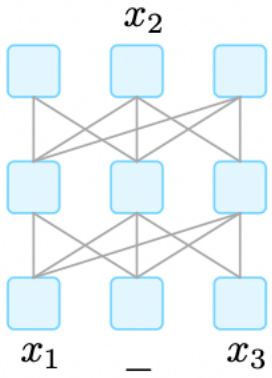
Example: Teaching GPT3 to write python program



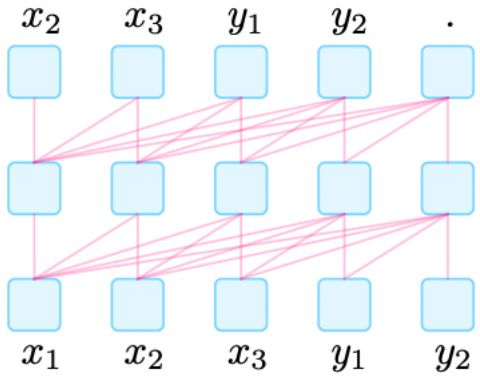
https://beta.openai.com/playground/p/default-sql-request

BART: From auto-regressive to the Encoder-decoder architecture

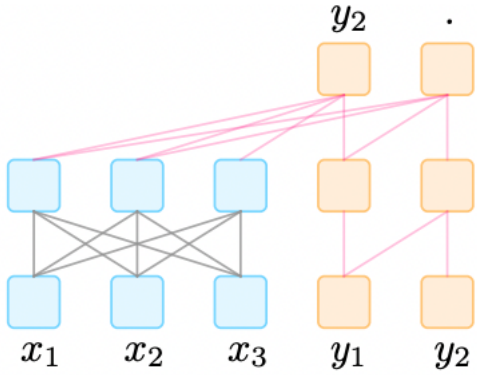
- Combining the encoder and the decoder transformer architecture:



MLM



Left-to-right (autoregressive)

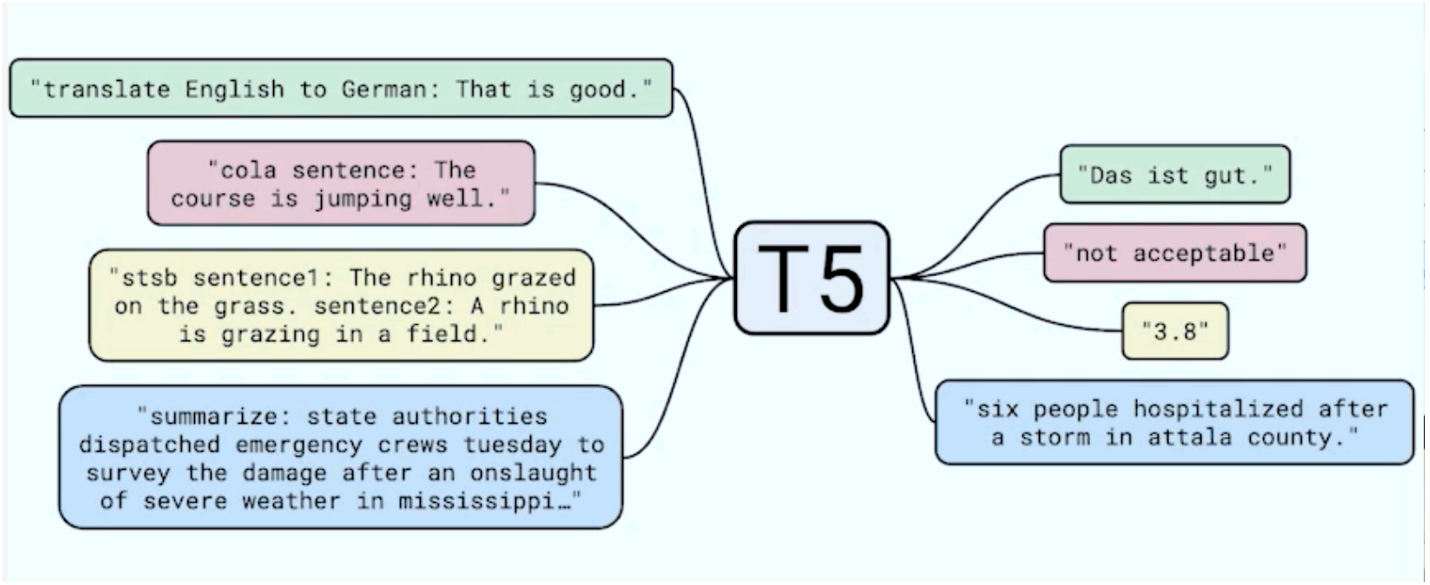


Encoder-decoder

BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. Lewis et al. 2020

T5: A unified seq-to-seq framework

- Downstream tasks are reformulated to look like language modeling



T5: Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. Raffel et al. 2019

T5: A unified seq-to-seq framework

- How does T5 work?
  - Converting every task as a seq2seq task
  - e.g., when classifying a sentence into two classes "acceptable" and "not acceptable", instead of predicting the 0/1 labels, train the language model to generate the sequence as ["acceptable"] and ["not", "acceptable"]

- Why does T5 work?
  - When fine-tuning the language model for classification task, we have to stack one additional layer and train it from scratch
  - On the other hand, when predicting "acceptable", we already learned v("acceptable") so we are not starting from scratch

An summary of pre-trained language models

- Training objective:
  - GPT: regular language modeling (no corrupted tokens)
  - BERT: bidirectional MLM, NSP
  - Electra: replaced token detection (RTD)
  - BART: deleted token detection

- LM direction:
  - GPT: left-to-right, autoregressive
  - BERT: bidirectional
  - BART: encoder decoder

- Application:
  - NLU: BERT, RoBERTa, Electra
  - NLG: BART, GPT, T5, Pegasus

CS 589 Fall 2021 Lecture

Prompt-based learning

Monday 6:30-9:00
Babbio 122

source: "Pre-train, prompt and predict: A systematic survey of prompting methods in NLP", Liu et al. 2021

photo: https://www.scubedstudios.com/information-retrieval/

Paradigm shifts in NLP

- **Fully supervised learning**:
  - A task specific model is trained solely on **labelled** training examples
  - The focus is **architecture engineering**, e.g., LSTM cell [Hochereiter and Schmidhuber 1997], attention [Bahdanau et al. 2014]

2017 →

- **Pre-training -> fine-tuning**:
  - A model with **fixed architecture** is pre-trained on massive **unlabelled data** using self-supervised learning
  - Focus is **objective engineering**, e.g., MLM (BERT), Replaced token detection (Electra), ensemble (Roberta), deleted token detection (BART)

Second paradigm shifts in NLP

- **Pre-training -> fine-tuning**:
  - A model with **fixed architecture** is pre-trained on massive **unlabelled data** using self-supervised learning
  - Focus is **objective engineering**, e.g., MLM (BERT), Replaced token detection (Electra), ensemble (Roberta), deleted token detection (BART)

2021 →

- **Pre-train, prompt, and predict:**
  - Downstream tasks are reformulated to look more like the original task via the help of a **textual prompt**
  - The focus is **prompt engineering**, e.g., finding the optimal prompt to allow the language model to solve the task easily

What is prompting?

- Designing natural language templates to manipulate better results from language models for filling in the blanks
- The bold fonts serve as a hint for language model, telling the LM to generate a french translation for the english sentence before it

  ***English**: I missed the bus today. **French**: _____*
- Prompts for other tasks:

  *In my younger and more vulnerable years my father gave me some advice that I've been turning over in my mind ever since...**TL;DR**: _____*

  *Create a SQL request to find all users who live in California and have over 1000 credits? **SQL statement**: _____*

Prompts can be used for classification tasks too!

- Prompts for sentiment classification:
  - Each sentiment (positive, negative) contains a list of adj, e.g., **pos**: happy, delighted, ... **neg**: sad, bad, ..., the predicted class is the class with the higher generative probability

I missed the bus today. ***I felt so*** _____

- Named entity recognition:

Mike went to Paris. [Paris] ***is a*** *[location]* ***entity****.*

Prompt engineering

- Designing the prompting that results in the most effective performance in the downstream task

- Categories of prompt:
  - **Cloze style prompts**: prompts which ask the LM to fill in the blank, e.g., Mike went to Paris. [Paris] **is a** [location] **entity**. More suitable for text classification
  - **Prefix style prompts**: prompts which ask the LM to complete a sequence, e.g., **English**: I missed the bus today. **French**: _____, More suitable for text generation tasks.

- Prompts can be created manually
  - However, this process is an art that takes time and experience
  - Even experienced prompt engineer may fail to manually discover optimal prompt

Automatically mining of prompt templates

- Factual retrieval: retrieving knowledge such as (obama, profession, president of the United States), wow to retrieve such knowledge from language models?

- Manually created prompts can be sub-optimal
  - e.g., ""*Obama worked as a* ____" may be better of a prompt than "*Obama is a* ____ *by profession*", because the LM learned the knowledge from a different context

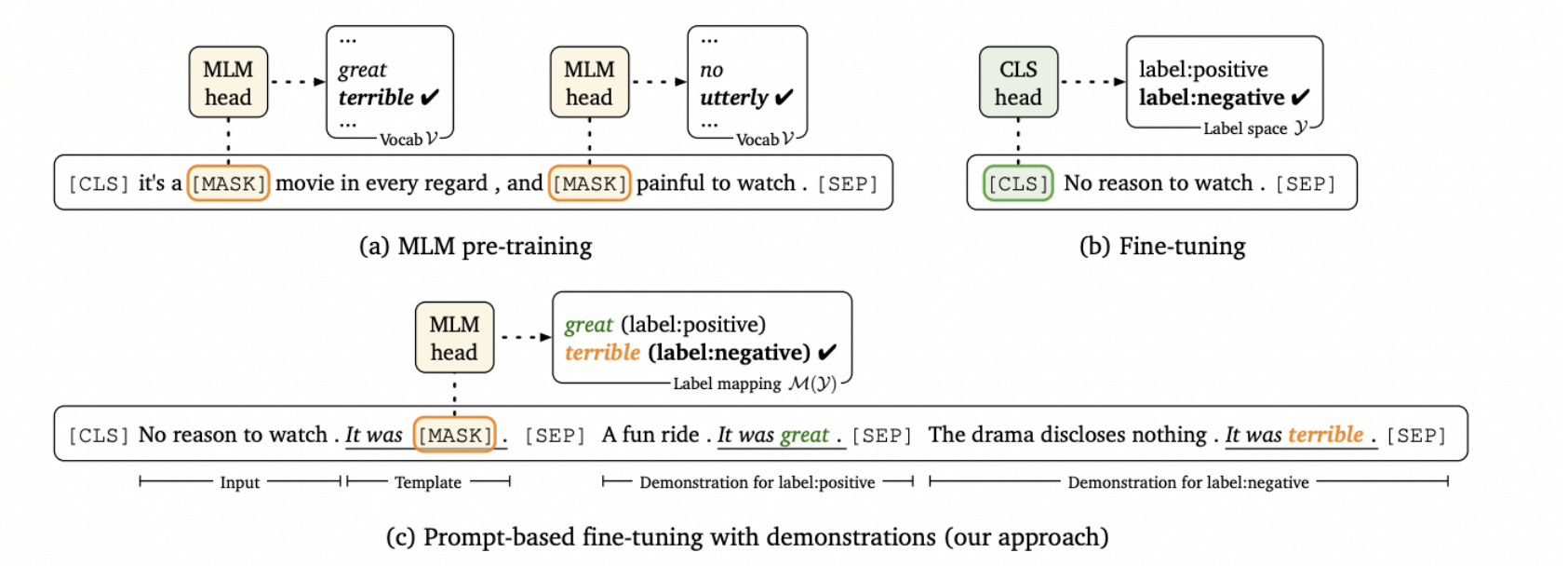
- Mining the frequent relation
  - Identify all wiki sentences containing both subjects and objects of a specific relation r
  - Leverage dependency parsing, e.g.

$$France \xleftarrow{pobj} of \xleftarrow{prep} capital \xleftarrow{nsubj} is \xrightarrow{attr} Paris"$$

How Can We Know What Language Models Know?. Jiang et al. 2019

Prompt generation

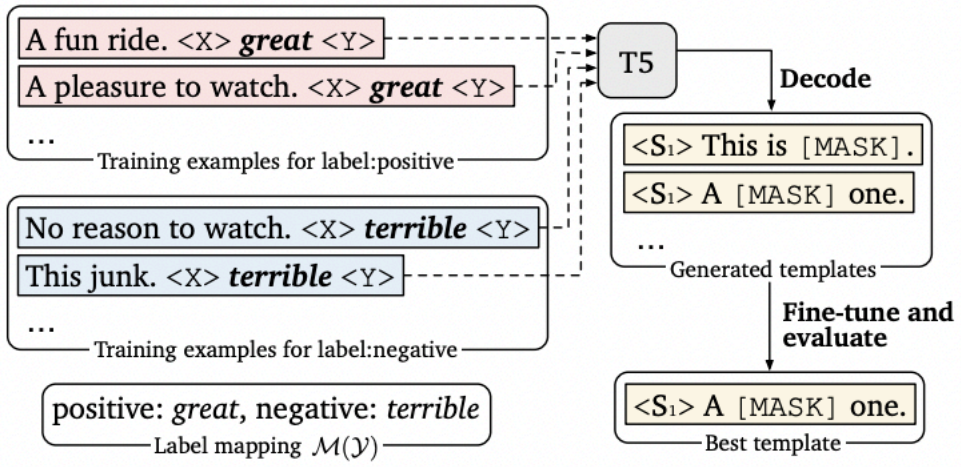
- Prompting by demonstration:



Making pre-trained language models better few-shot learners. Gao et al. 2020

Prompt generation

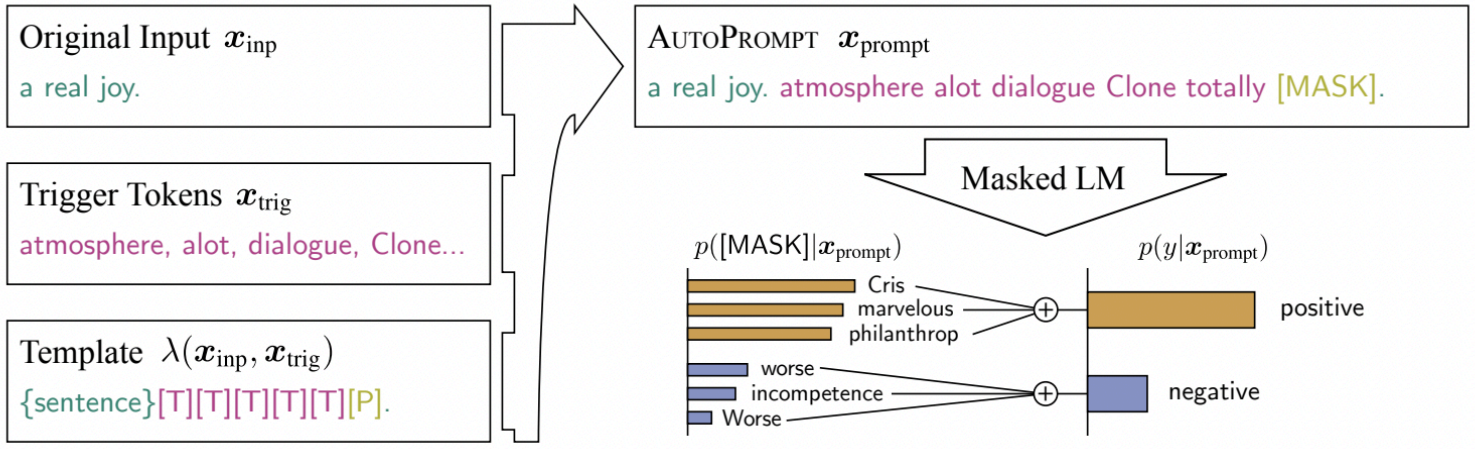
- Using T5 for prompt generation:
  - Defining an input and output template pairs, e.g., input: Thank you <X> me to your party <Y> week, output: <X> for inviting <Y> last <Z>
  - It teaches T5 that <X> is for replacing <X> in the input and <Y> is for replacing <Y> in the input
  - During decoding, replace <X> with <S1> and <Y> with [MASK], which makes [MASK] the target for generating the sentiment word



Making pre-trained language models better few-shot learners. Gao et al. 2020

AutoPrompt: Gradient based prompt generation

- Create a task-specific prompt with a collection of trigger words
- The same trigger words are used for all examples, learned through maximizing likelihood of sentiment labels in the training examples



AUTOPROMPT: Eliciting Knowledge from Language Models with Automatically Generated Prompts. Shin et al. 2020

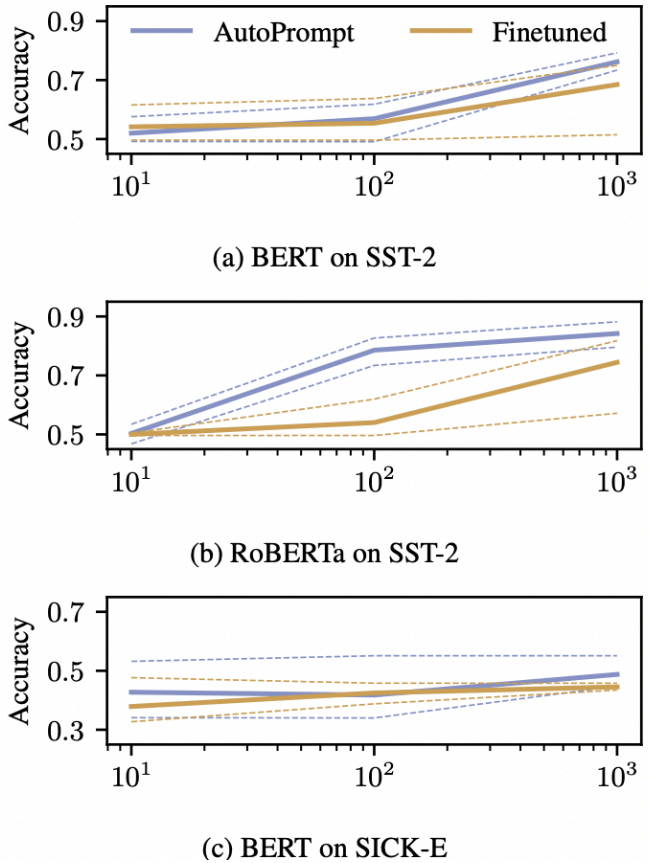
AutoPrompt: Gradient based prompt generation

- How to select words for the prompt:
  - Step 1: train a classifier to predict the class label using the contextualized embedding of the [MASK] as input:

$$\boldsymbol{h} = \text{Transformer}_{\text{enc}}(\tilde{\boldsymbol{x}})$$

  - Step 2: substitute h_i with the MLM's output word embeddings to obtain a score s(y, w), and the set of labelled tokens are constructed from the k-highest scoring words

$$\mathcal{V}_y = \underset{w \in \mathcal{V}}{\text{top-}k} \left[s(y, w) \right]$$

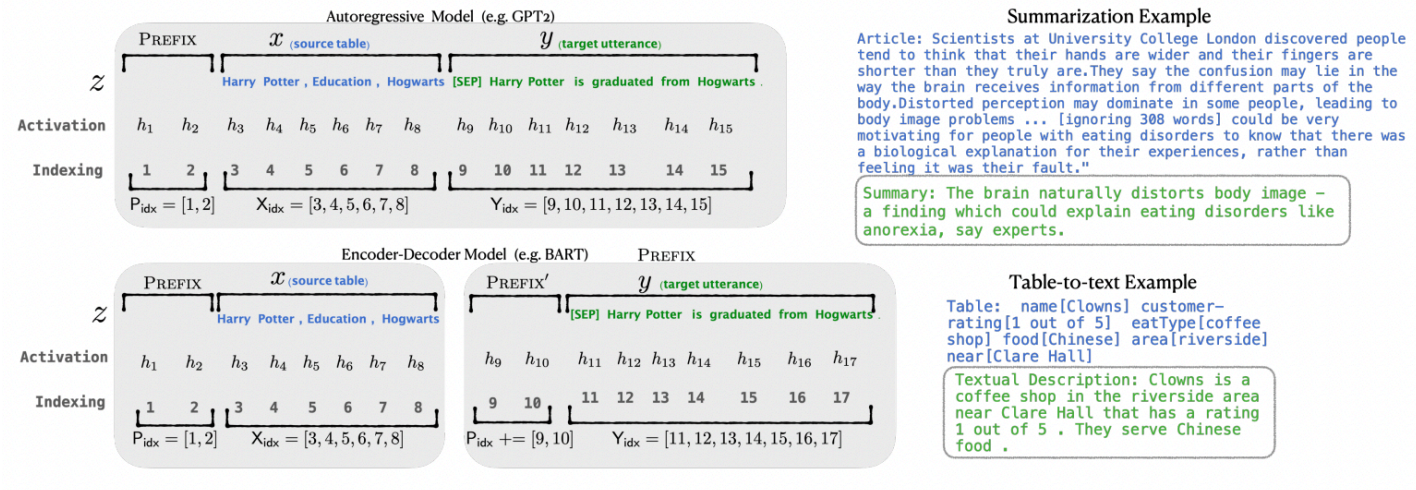


AUTOPROMPT: Eliciting Knowledge from Language Models with Automatically Generated Prompts. Shin et al. 2020

Continuous prompting

- Prompt does not have to be discrete
  - Because the goal of prompting is to elicit the content in the blank
  - Therefore it is OK if they are not interpretable, or not even being natural language

- Continuous prompts
  - Prepend the input and output pairs with continuous vectors as the continous prompt

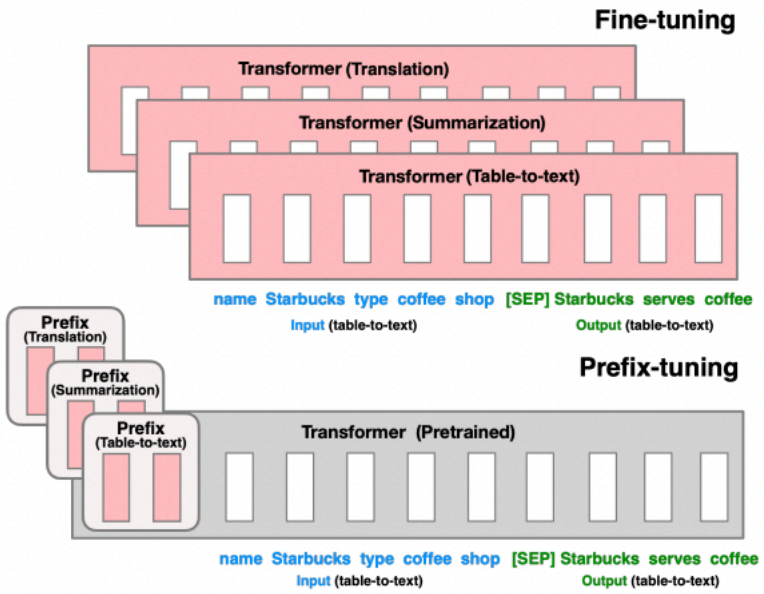


Prefix-tuning: optimizing continuous prompts for generation. Li et al. 2020

Prefix-tuning: lightweight fine-tuning

- Prefix tuning also helps with lightweight fine tuning
  - Fine tuning large language models is costly!
  - gpt-j-6b is 22gb!

- It was shown in [Li et al. 2020] that with pre-fix tuning, we only need to tune the prefix for each task, which significantly reduces the parameters that need to be tuned
  - with 0.1% parameters, can obtain comparable performance with GPT-2 and BART full parameter

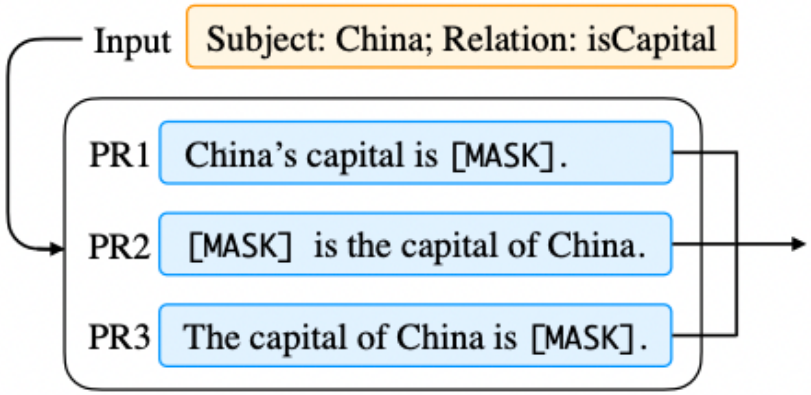


Prefix-tuning: optimizing continuous prompts for generation. Li et al. 2020

Prompt ensemble

- Using multiple unanswered prompts for an input at inference time to make predictions

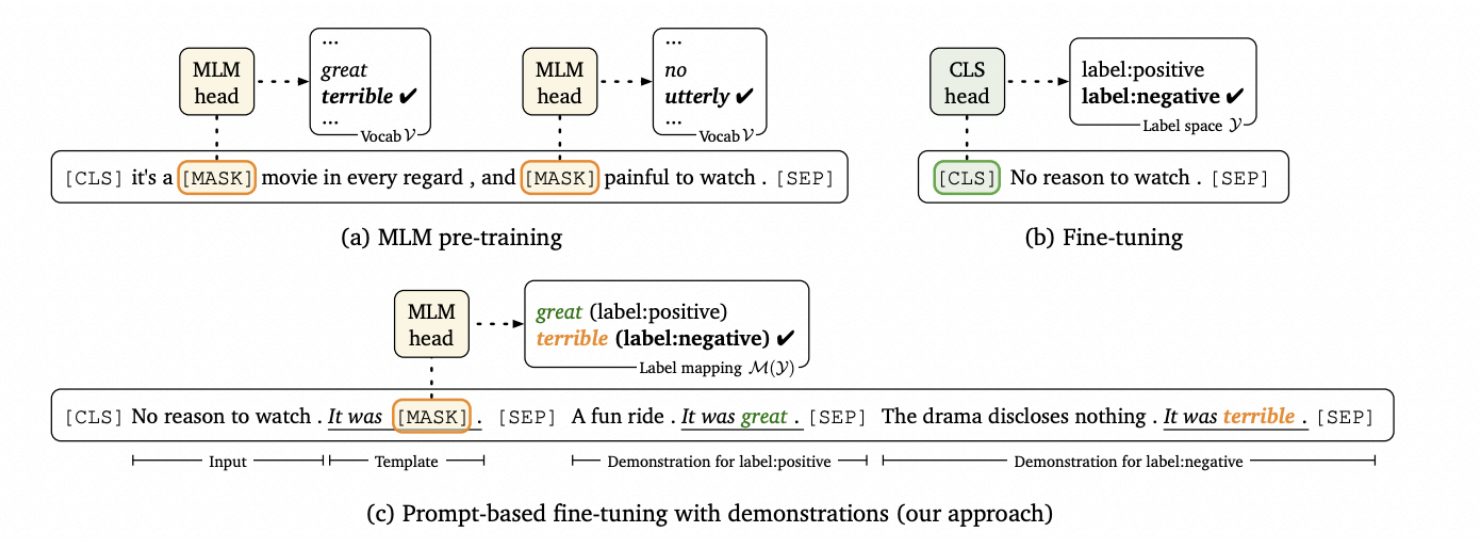
- How to ensemble the prediction:
  - Uniform averaging
  - Weighted averaging
  - Majority voting



Making pre-trained language models better few-shot learners. Gao et al. 2020

Prompt augmentation

- Providing a few additional answer prompts that can be used to demonstrate how the language model could provide the answer to the actual prompt instatiated with the input



Making pre-trained language models better few-shot learners. Gao et al. 2020

Prompting performance

- With appropriate prompts, few shot learning can be achieved

	SST-2 (acc)	SST-5 (acc)	MR (acc)	CR (acc)	MPQA (acc)	Subj (acc)	TREC (acc)	CoLA (Matt.)
Majority†	*50.9*	*23.1*	*50.0*	*50.0*	*50.0*	*50.0*	*18.8*	*0.0*
Prompt-based zero-shot‡	83.6	35.0	80.8	79.5	67.6	51.4	32.0	2.0
"GPT-3" in-context learning	84.8 (1.3)	30.6 (0.9)	80.5 (1.7)	87.4 (0.8)	63.8 (2.1)	53.6 (1.0)	26.2 (2.4)	-1.5 (2.4)
Fine-tuning	81.4 (3.8)	43.9 (2.0)	76.9 (5.9)	75.8 (3.2)	72.0 (3.8)	90.8 (1.8)	88.8 (2.1)	**33.9** (14.3)
Prompt-based FT (man)	92.7 (0.9)	47.4 (2.5)	87.0 (1.2)	90.3 (1.0)	84.7 (2.2)	91.2 (1.1)	84.8 (5.1)	9.3 (7.3)
+ demonstrations	92.6 (0.5)	**50.6** (1.4)	86.6 (2.2)	90.2 (1.2)	**87.0** (1.1)	**92.3** (0.8)	87.5 (3.2)	18.7 (8.8)
Prompt-based FT (auto)	92.3 (1.0)	49.2 (1.6)	85.5 (2.8)	89.0 (1.4)	85.8 (1.9)	91.2 (1.1)	88.2 (2.0)	14.0 (14.1)
+ demonstrations	**93.0** (0.6)	49.5 (1.7)	**87.7** (1.4)	**91.0** (0.9)	86.5 (2.6)	91.4 (1.8)	**89.4** (1.7)	21.8 (15.9)
Fine-tuning (full)†	*95.0*	*58.7*	*90.8*	*89.4*	*87.8*	*97.0*	*97.4*	*62.6*

	MNLI (acc)	MNLI-mm (acc)	SNLI (acc)	QNLI (acc)	RTE (acc)	MRPC (F1)	QQP (F1)	STS-B (Pear.)
Majority†	*32.7*	*33.0*	*33.8*	*49.5*	*52.7*	*81.2*	*0.0*	-
Prompt-based zero-shot‡	50.8	51.7	49.5	50.8	51.3	61.9	49.7	-3.2
"GPT-3" in-context learning	52.0 (0.7)	53.4 (0.6)	47.1 (0.6)	53.8 (0.4)	60.4 (1.4)	45.7 (6.0)	36.1 (5.2)	14.3 (2.8)
Fine-tuning	45.8 (6.4)	47.8 (6.8)	48.4 (4.8)	60.2 (6.5)	54.4 (3.9)	76.6 (2.5)	60.7 (4.3)	53.5 (8.5)
Prompt-based FT (man)	68.3 (2.3)	70.5 (1.9)	77.2 (3.7)	64.5 (4.2)	69.1 (3.6)	74.5 (5.3)	65.5 (5.3)	71.0 (7.0)
+ demonstrations	**70.7** (1.3)	**72.0** (1.2)	**79.7** (1.5)	**69.2** (1.9)	68.7 (2.3)	77.8 (2.0)	**69.8** (1.8)	73.5 (5.1)
Prompt-based FT (auto)	68.3 (2.5)	70.1 (2.6)	77.1 (2.1)	68.3 (7.4)	**73.9** (2.2)	76.2 (2.3)	67.0 (3.0)	75.0 (3.3)
+ demonstrations	70.0 (3.6)	**72.0** (3.1)	77.5 (3.5)	68.5 (5.4)	71.1 (5.3)	**78.1** (3.4)	67.7 (5.8)	**76.4** (6.2)
Fine-tuning (full)†	*89.8*	*89.5*	*92.6*	*93.3*	*80.9*	*91.4*	*81.7*	*91.9*

Making pre-trained language models better few-shot learners. Gao et al. 2020

Controlling the generation quality of prompt based generation

- GPT-3 is a very powerful generation tool, however, the quality of generated sentences still vary
- Traditional generation task optimizes the conditional probability of the generated text:
- To control the quality of the generated text, we can apply inverse prompts:

$$f(c_g|c_p) = \log p(c'_p|c'_g),$$

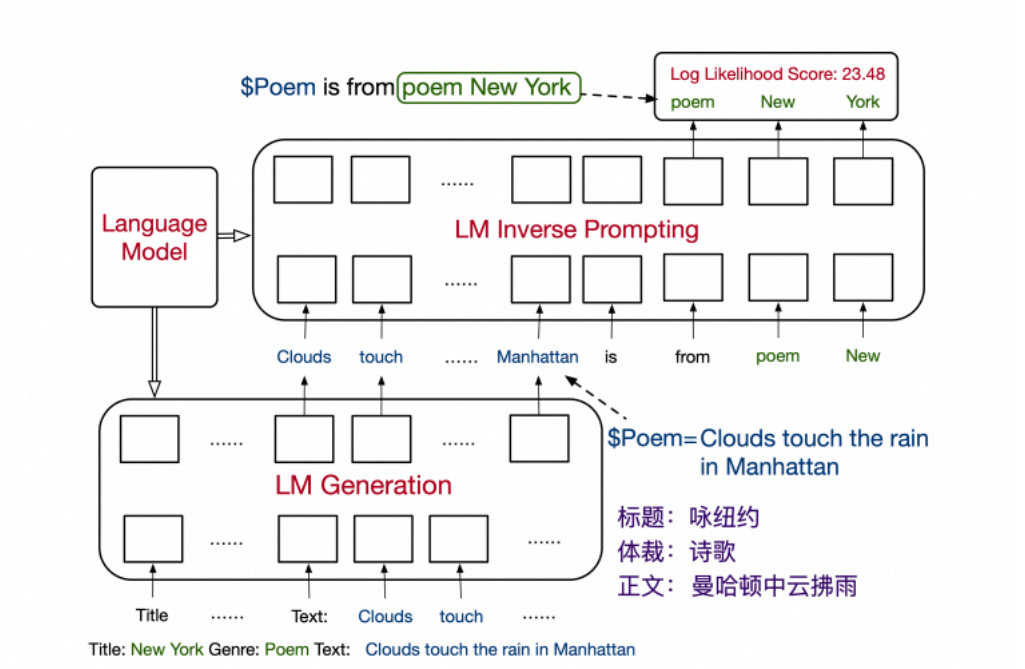
Question	什么瞬间让你想留住那一刻? Which moment did you want to live in forever?
Prompting Baseline (relevance:2.83 overall:5.40)	我和男朋友在一起两年。我去国外留学, 他留在国内。一人在国外, 寂寞心酸都是真的。念书的时候, 觉得常在一起也没什么, 两个人一起努力, 吵吵闹闹就过来了。我一直这样想, 以至于到了高中, 到了最后几个月, 我都觉得他就是我行走的小弟儿。其实最快乐的时候, 就是一起在外头的“孤儿院”里。 I have been in a relationship with my boyfriend for two years. I went to study abroad and he stayed in our home country. Being alone in a foreign country, I felt really lonely and sad. When I was in school, I felt that it was ok to be together often. We worked hard together and we also quarrelled with each other. I kept thinking in this way. In the last few months of our time at the high school, I felt that he was my walking little brother. In fact, the happiest moment was when we were together in the “orphanage”.
Inverse Prompting (relevance:4.53 overall:8.60)	一瞬间想到高二的时候吧, 那天下午在操场晒了一中午太阳, 和同学一起在楼下打球, 下午的夕阳徐徐洒下来, 我们三个人一起站在走廊上, 倚着栏杆可以面对着夕阳。一直想留下那一瞬间, 可惜我以后再也没遇到过那样的时光。就是想留一下那一刻吧。 In an instant I recall the second year of high school. That afternoon, I took a sunbath on the playground for a long time. I then played basketball downstairs with my classmates. As the sun set slowly, the three of us stood together on the corridor, leaning on the railing to face the sunset. I’ve always wanted to live in that moment forever, but unfortunately I never encountered a similar moment again. I just want to live in that moment.

Controllable Generation from Pre-trained Language Models via Inverse Prompting. Zou et al. 2021

Controlling the generation quality of prompt based generation

- GPT-3 is a very powerful generation tool, however, the quality of generated sentences still vary
- Traditional generation task optimizes the conditional probability of the generated text:
- To control the quality of the generated text, we can apply inverse prompts:

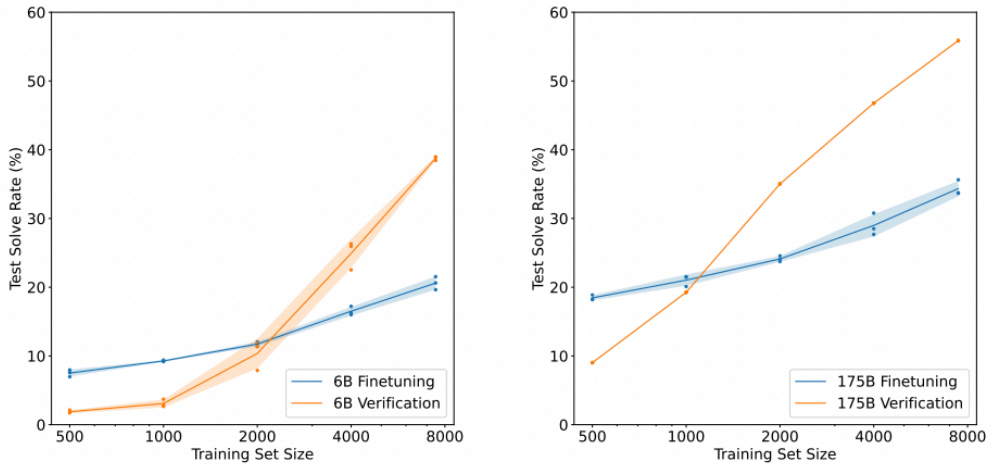
$$f(c_g|c_p) = \log p(c'_p|c'_g),$$



Controllable Generation from Pre-trained Language Models via Inverse Prompting. Zou et al. 2021

Using GPT for math word problem

- The temperature in GPT controls the diversity of generation
- Investigate two methods:
  - Fine-tuning
  - Verification



Training Verifiers to Solve Math Word Problems. Cobbe et al. 2021

Problem: Beth bakes 4, 2 dozen batches of cookies in a week. If these cookies are shared amongst 16 people equally, how many cookies does each person consume?
Solution: Beth bakes 4 2 dozen batches of cookies for a total of 4*2 = <<4*2=8>>8 dozen cookies
There are 12 cookies in a dozen and she makes 8 dozen cookies for a total of 12*8 = <<12*8=96>>96 cookies
She splits the 96 cookies equally amongst 16 people so they each eat 96/16 = <<96/16=6>>6 cookies
Final Answer: 6

Problem: Mrs. Lim milks her cows twice a day. Yesterday morning, she got 68 gallons of milk and in the evening, she got 82 gallons. This morning, she got 18 gallons fewer than she had yesterday morning. After selling some gallons of milk in the afternoon, Mrs. Lim has only 24 gallons left. How much was her revenue for the milk if each gallon costs $3.50?
Mrs. Lim got 68 gallons - 18 gallons = <<68-18=50>>50 gallons this morning.
So she was able to get a total of 68 gallons + 82 gallons + 50 gallons = <<68+82+50=200>>200 gallons.
She was able to sell 200 gallons - 24 gallons = <<200-24=176>>176 gallons.
Thus, her total revenue for the milk is $3.50/gallon x 176 gallons = $<<3.50*176=616>>616.
Final Answer: 616

Problem: Tina buys 3 12-packs of soda for a party. Including Tina, 6 people are at the party. Half of the people at the party have 3 sodas each, 2 of the people have 4, and 1 person has 5. How many sodas are left over when the party is over?
Solution: Tina buys 3 12-packs of soda, for 3*12= <<3*12=36>>36 sodas
6 people attend the party, so half of them is 6/2= <<6/2=3>>3 people
Each of those people drinks 3 sodas, so they drink 3*3=<<3*3=9>>9 sodas
Two people drink 4 sodas, which means they drink 2*4=<<4*2=8>>8 sodas
With one person drinking 5, that brings the total drank to 5+9+8+3= <<5+9+8+3=25>>25 sodas
As Tina started off with 36 sodas, that means there are 36-25=<<36-25=11>>11 sodas left
Final Answer: 11

Training Verifiers to Solve Math Word Problems. Cobbe et al. 2021