NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

Multi-perspective Optimization of Pre-trained Language Model: What Works and What's Next

Yige Xu

PhD Student, SCSE, NTU

yige002@e.ntu.edu.sg

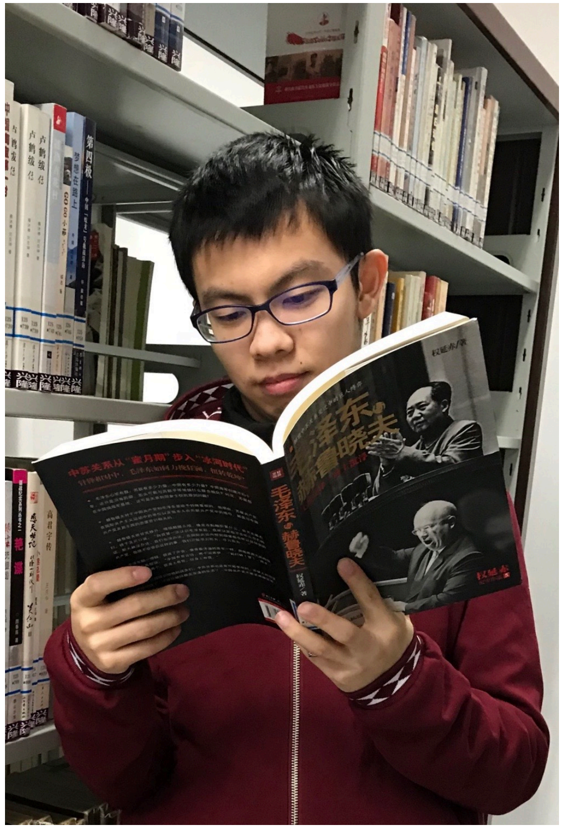
https://xuyige.github.io

Profile

- Currently, I am a first-year Ph.D. student in LILY, SCSE, NTU. My supervisor is Prof. Chunyan Miao.

- Before NTU, I obtained my Master’s degree at Fudan University in 2021, and Bachelor’s degree at Shandong University in 2018.

- My research interest mainly focus on knowledge transfer in natural language processing.



Outlines

- Background of Pre-trained Language Models
- Fine-tuning: A Simple but Effective Method of Transferring Knowledge
- Optimization of Training Objective
- Optimization of Module Architecture
- Optimization of Evaluation Metric
- Prompting: A New Paradigm of Transferring Knowledge

Background of Pre-trained Language Models

- Word Representations: Use a dense vector to represent a word
  - Convert the discrete signals to the continuous signals
  - Can represent more features

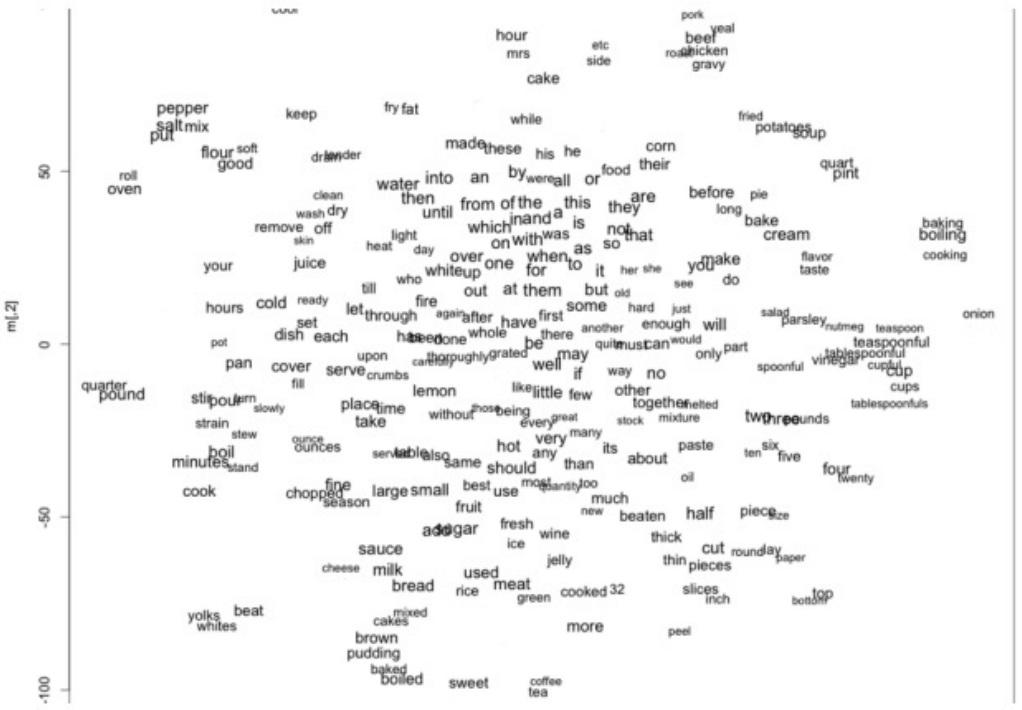


Figure is from [here](#).

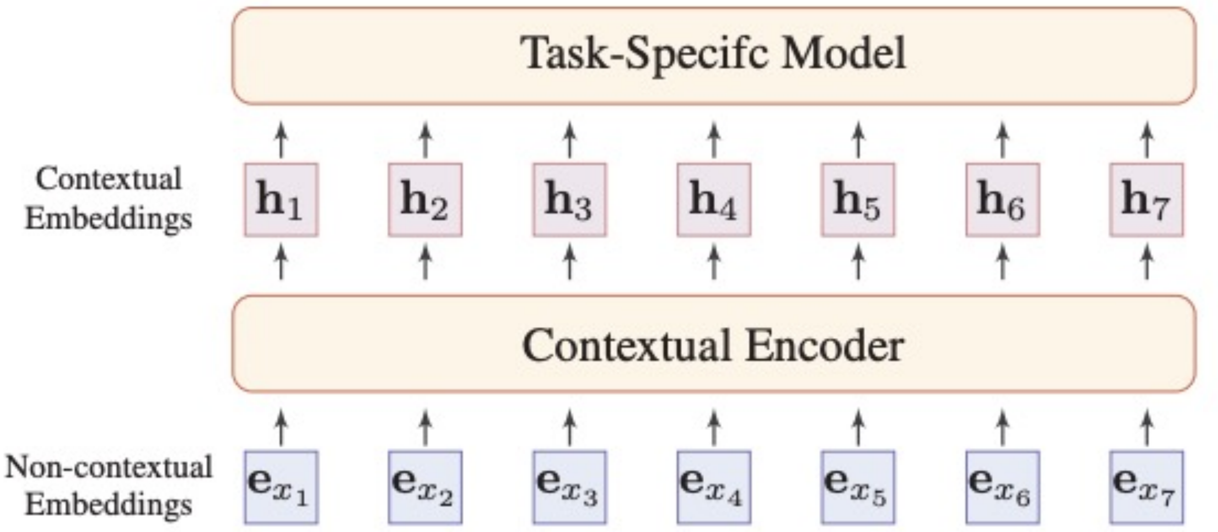
Background of Pre-trained Language Models

- Pre-trained Word Representations
  - – Provide a good initialization point
  - – Contain some semantic information

Figure is from here.

Background of Pre-trained Language Models

- Pre-trained Language Models
  - Learn universal language representations.
  - Obtain a good initialization.
  - As a regularization method.



Pre-trained Models for Natural Language Processing: A Survey

Fine-tuning

- Source domain: Pre-trained Language Model
- Target task: Downstream Task

How to transfer knowledge from source domain to target task?

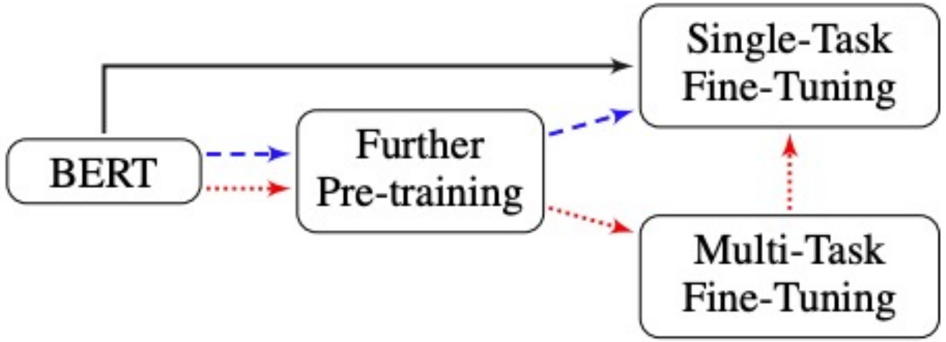


Figure 1: Three general ways for fine-tuning BERT, shown with different colors.

How to Fine-tune BERT for Text Classification?

Fine-tuning

- Single-Task Fine-Tuning
  – Simply add a classifier layer (Task-specific Model) to the bottom of PLM.
  – Jointly optimize all the parameters from PLM as well as Task-specific Model.

Layer	Test error rates(%)
Layer-0	11.07
Layer-1	9.81
Layer-2	9.29
Layer-3	8.66
Layer-4	7.83
Layer-5	6.83
Layer-6	6.83
Layer-7	6.41
Layer-8	6.04
Layer-9	5.70
Layer-10	5.46
Layer-11	**5.42**

Layer	Test error rates(%)
First 4 Layers + concat	8.69
First 4 Layers + mean	9.09
First 4 Layers + max	8.76
Last 4 Layers + concat	5.43
Last 4 Layers + mean	5.44
Last 4 Layers + max	**5.42**
All 12 Layers + concat	5.44

How to Fine-tune BERT for Text Classification?

Fine-tuning

- Further Pre-training
  - Apply pre-training task on another unlabeled corpus **U**. Then fine-tune the new checkpoints on the downstream task same as Single-Task Fine-Tuning.
    - ❑Within-task pre-training
      - ❑**U** <- corpus from the training set of a target task
    - ❑In-domain pre-training
      - ❑**U** <- corpus from the same domain of a target task
    - ❑Cross-domain pre-training
      - ❑**U** <- corpus from both the same and other different domains to a target task

- Source domain -> Target domain -> Target task

How to Fine-tune BERT for Text Classification?

Fine-tuning

- Further Pre-training
  - Within-Task Pre-training

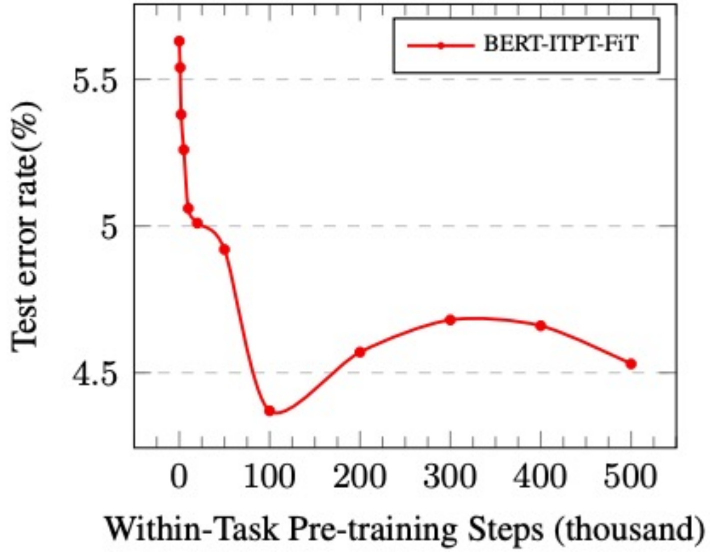


Figure 3: Benefit of different further pre-training steps on IMDb datasets. BERT-ITPT-FiT means "BERT + withIn-Task Pre-Training + Fine-Tuning".

How to Fine-tune BERT for Text Classification?

Fine-tuning

- Further Pre-training
  - In-Domain and Cross-Domain Further Pre-training

Domain	sentiment			question		topic	
Dataset	IMDb	Yelp P.	Yelp F.	TREC	Yah. A.	AG's News	DBPedia
IMDb	**4.37**	2.18	29.60	2.60	22.39	5.24	0.68
Yelp P.	5.24	1.92	29.37	2.00	22.38	5.14	**0.65**
Yelp F.	5.18	1.94	29.42	2.40	22.33	5.43	**0.65**
all sentiment	4.88	**1.87**	29.25	3.00	22.35	5.34	0.67
TREC	5.65	2.09	29.35	3.20	22.17	5.12	0.66
Yah. A.	5.52	2.08	29.31	**1.80**	22.38	5.16	0.67
all question	5.68	2.14	29.52	2.20	**21.86**	5.21	0.68
AG's News	5.97	2.15	29.38	2.00	22.32	**4.80**	0.68
DBPedia	5.80	2.13	29.47	2.60	22.30	5.13	0.68
all topic	5.85	2.20	29.68	2.60	22.28	4.88	**0.65**
all	5.18	1.97	**29.20**	2.80	21.94	5.08	0.67
w/o pretrain	5.40	2.28	30.06	2.80	22.42	5.25	0.71

How to Fine-tune BERT for Text Classification?

Fine-tuning

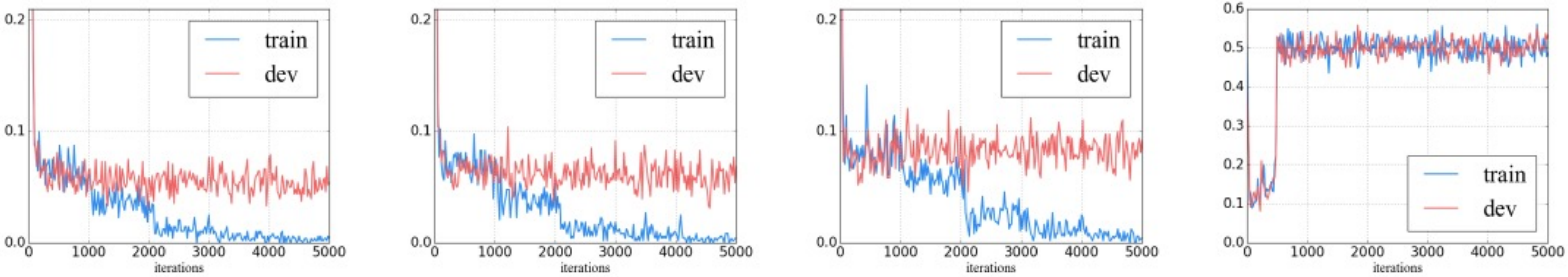
- Comparison with Models before PLM

Model	IMDb	Yelp P.	Yelp F.	TREC	Yah. A.	AG	DBP	Sogou	Avg. Δ
Char-level CNN(Zhang et al., 2015)	/	4.88	37.95	/	28.80	9.51	1.55	3.80*	/
VDCNN (Conneau et al., 2016)	/	4.28	35.28	/	26.57	8.67	1.29	3.28	/
DPCNN (Johnson and Zhang, 2017)	/	2.64	30.58	/	23.90	6.87	0.88	3.48*	/
D-LSTM (Yogatama et al., 2017)	/	7.40	40.40	/	26.30	7.90	1.30	5.10	/
Standard LSTM (Seo et al., 2017)	8.90	/	/	/	/	6.50	/	/	/
Skim-LSTM (Seo et al., 2017)	8.80	/	/	/	/	6.40	/	/	/
HAN (Yang et al., 2016)	/	/	/	/	24.20	/	/	/	/
Region Emb. (Qiao et al., 2018)	/	3.60	35.10	/	26.30	7.20	1.10	2.40	/
CoVe (McCann et al., 2017)	8.20	/	/	4.20	/	/	/	/	/
ULMFiT (Howard and Ruder, 2018)	4.60	2.16	29.98	3.60	/	5.01	0.80	/	/
BERT-Feat	6.79	2.39	30.47	4.20	22.72	5.92	0.70	2.50	-
BERT-FiT	5.40	2.28	30.06	2.80	22.42	5.25	0.71	2.43	9.22%
BERT-ITPT-FiT	**4.37**	1.92	29.42	3.20	22.38	**4.80**	0.68	**1.93**	16.07%
BERT-IDPT-FiT	4.88	**1.87**	29.25	**2.20**	**21.86**	4.88	**0.65**	/	**18.57%**
BERT-CDPT-FiT	5.18	1.97	**29.20**	2.80	21.94	5.08	0.67	/	14.38%

How to Fine-tune BERT for Text Classification?

Fine-tuning

- Learning Rate Tuning



(a) lr=2e-5 (b) lr=5e-5 (c) lr=1e-4 (d) lr=4e-4

– A lower learning rate such as 2e-5 is necessary to make PLM (BERT) overcome the catastrophic forgetting problem.

How to Fine-tune BERT for Text Classification?

Fine-tuning

- Learning Rate Tuning
  - Utilize a layer-specific learning rate

$$\theta_t^l = \theta_{t-1}^l - \eta^l \cdot \nabla_{\theta^l} J(\theta)$$

$$\eta^{k-1} = \xi \cdot \eta^k$$

Learning rate	Decay factor ξ	Test error rates(%)
2.5e-5	1.00	5.52
2.5e-5	0.95	5.46
2.5e-5	0.90	**5.44**
2.5e-5	0.85	5.58
2.0e-5	1.00	5.42
2.0e-5	0.95	**5.40**
2.0e-5	0.90	5.52
2.0e-5	0.85	5.65

Table 4: Decreasing layer-wise layer rate.

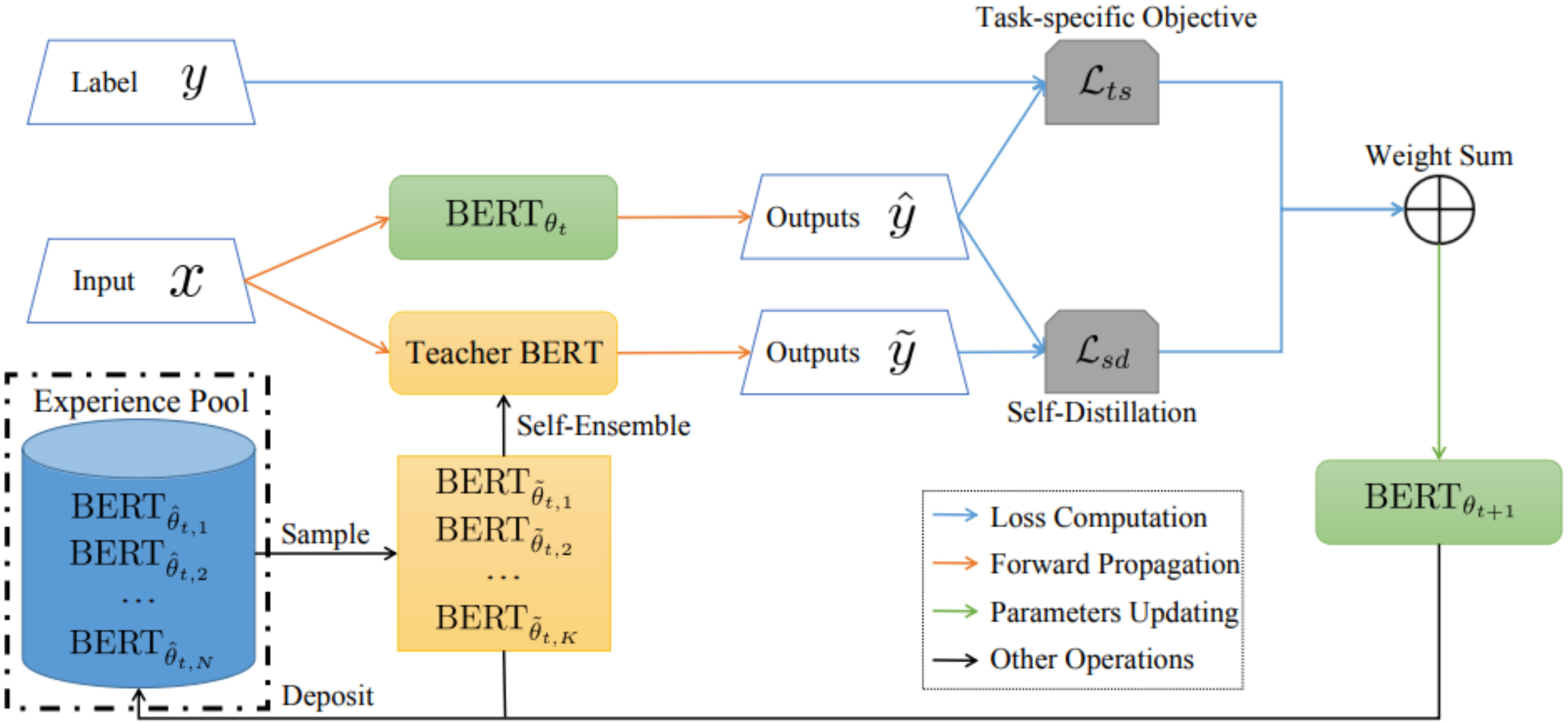
How to Fine-tune BERT for Text Classification?

Optimize the Training Objective

- Fine-tuning usually achieves better results than feature extraction
- Fine-tuning strategy itself is simple and has yet to be fully explored
- How can we maximize the utilization of PLM without introducing external data or knowledge?

Improving BERT Fine-Tuning via Self-Ensemble and Self-Distillation

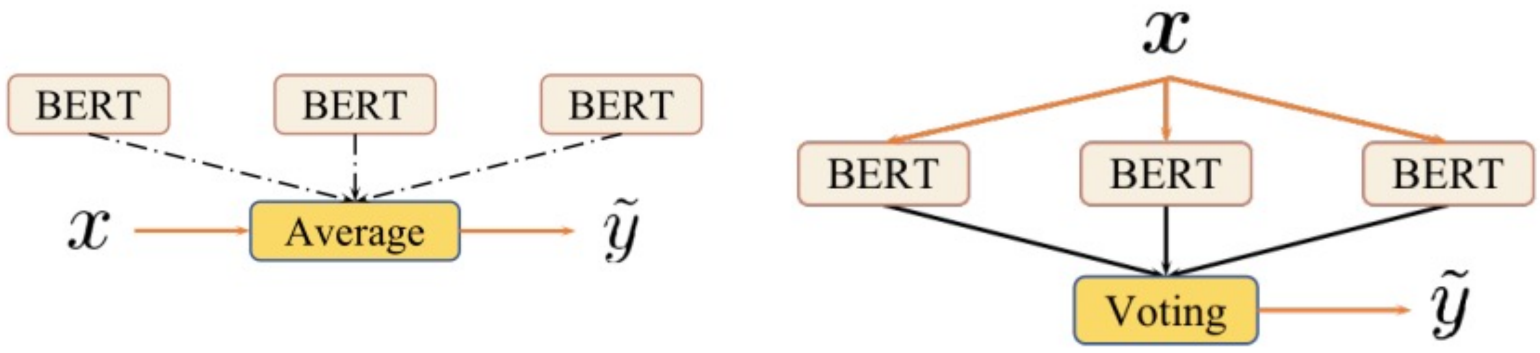
Optimize the Training Objective



Improving BERT Fine-Tuning via Self-Ensemble and Self-Distillation

Optimize the Training Objective

- Self-Ensemble
  - Sample checkpoints from the experience pool
  - Use parameter averaging or logits voting to compute the output of teacher models



(a) (b)

Improving BERT Fine-Tuning via Self-Ensemble and Self-Distillation

Optimize the Training Objective

- Self-Distillation
  - Self-Distillation-Averaged (SDA)

$$\bar{\theta}_t = \frac{1}{K}\sum_{k=1}^{K} \tilde{\theta}_{t,k},$$

$$\mathcal{L}_{sd}(x) = MSE\Big(BERT_{\vartheta_t}(x), BERT_{\bar{\theta}_t}(x)\Big),$$

  - Self-Distillation-Voted (SDV)

$$\mathcal{L}_{sd}(x) = MSE\Big(BERT_{\theta_t}(x), \frac{1}{K}\sum_{k=1}^{K} BERT_{\tilde{\theta}_{t,k}}(x)\Big).$$

Improving BERT Fine-Tuning via Self-Ensemble and Self-Distillation

Optimize the Training Objective

- Main Results

Table 6. Model Comparison on the Test Set of the GLUE Benchmark

Model	CoLA	SST-2	MRPC	STS-B	QQP	MNLI-m/mm	QNLI	RTE	Avg. Score
	Mcc	Acc	Acc/F1	P/S Corr	Acc/F1	Acc	Acc	Acc	
$\text{BERT}_{\text{BASE}}$ [1]	52.1	93.5	88.9/84.8	87.1/85.8	71.2/89.2	84.6/83.4	90.5	66.4	79.7
$\text{BERT}_{\text{BASE}}$-ReImp	52.2	93.4	88.3/84.8	86.7/85.6	71.0/89.2	84.3/83.4	90.5	66.5	79.6
BERT_{SDA} (ours)	**53.1**	94.4	**88.7/84.5**	**87.0/86.0**	72.4/89.6	85.0/84.3	91.3	68.8	**80.6**
BERT_{SDV} (ours)	52.6	**94.6**	88.4/84.4	86.9/85.7	**72.5/89.7**	**85.3/84.3**	**91.4**	**68.9**	80.5

Improving BERT Fine-Tuning via Self-Ensemble and Self-Distillation

Optimize the Training Objective

- Main Results

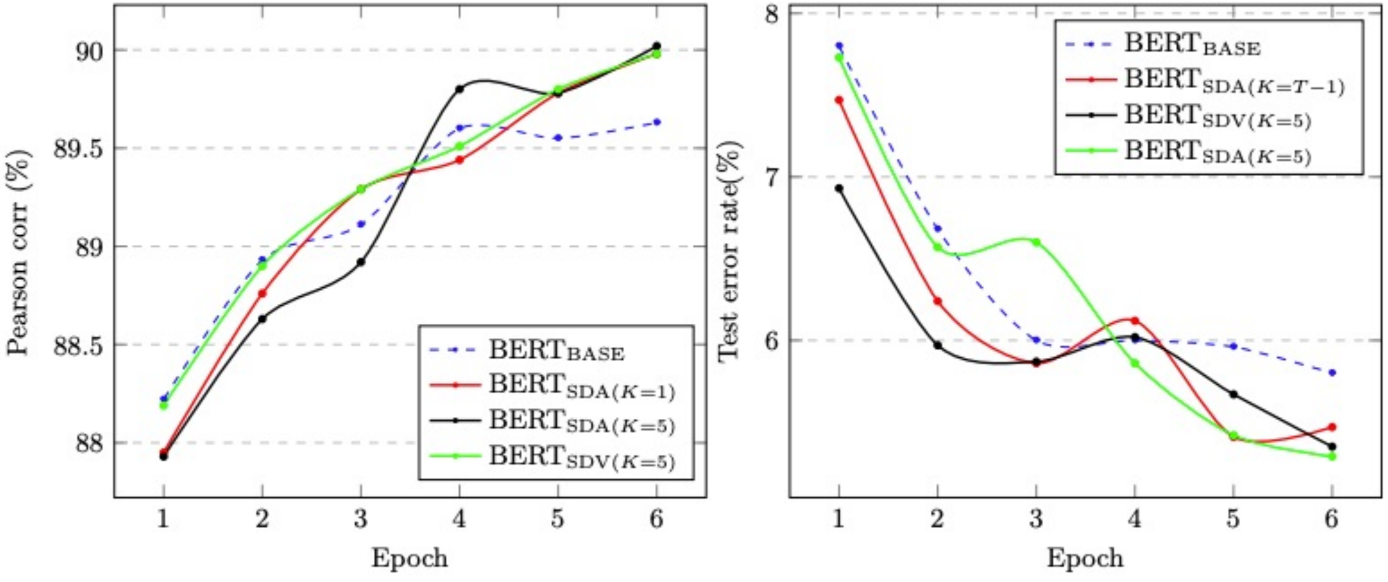
Table 7. Model Comparison of Different 24-Layer Model

Model	IMDb	AG's News	Avg. Δ	SNLI	Δ
XLNet [2]	3.79	4.49	/	/	/
MT-DNN [11]	/	/	/	91.6	/
CA-MTL [42]	/	/	/	92.1	/
BERT – L (our implementation)	4.98	5.45	-	90.9	-
RoBERTa – L (our implementation)	3.88	5.33	-	91.8	-
BERT – L_{SDV}	4.66	5.21	5.62%	**91.5**	**6.59%**
BERT – L_{SDA}	**4.58**	**5.15**	**7.02%**	91.4	5.49%
RoBERTa – L_{SDV}	3.58	5.03	5.62%	**92.6**	**9.76%**
RoBERTa – L_{SDA}	**3.48**	**5.02**	**5.81%**	92.5	8.54%

Improving BERT Fine-Tuning via Self-Ensemble and Self-Distillation

Optimize the Training Objective

- Convergence Curves



(a) (b)

Improving BERT Fine-Tuning via Self-Ensemble and Self-Distillation

Optimize the Training Objective

- Convergence Curves

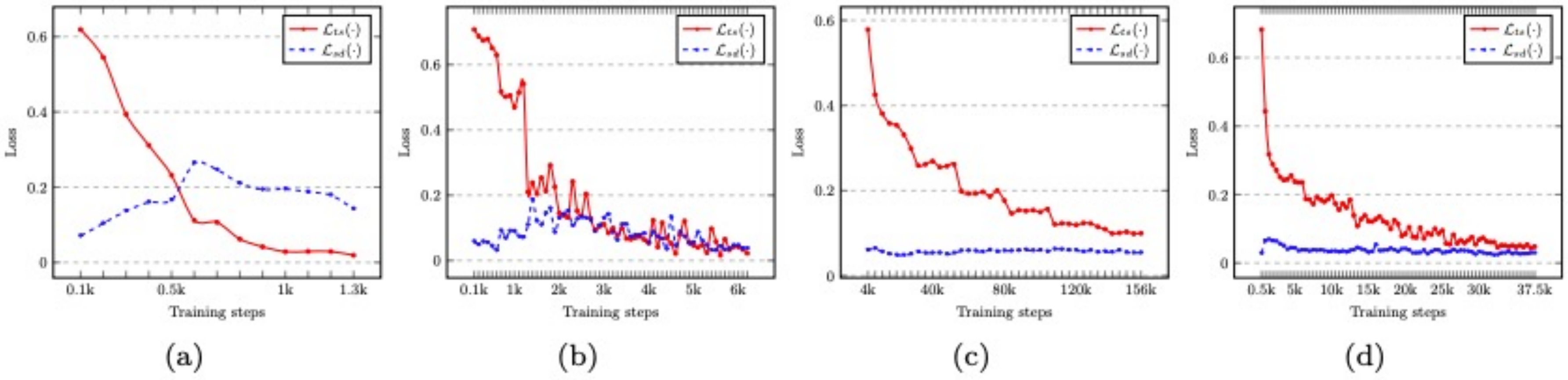


Fig.5. Loss curve of $\text{BERT}_{\text{SDA}(K=1)}$ on four datasets: (a) MRPC, (b) RTE, (c) QNLI, and (d) IMDb.

Improving BERT Fine-Tuning via Self-Ensemble and Self-Distillation

Optimize the Training Objective

- Model Comparison

Table 9. Model Comparison of Fine-Tuning the BERT-Base ($\mathrm{BERT_{BASE}}$) Model

Model	Test Error Rate (%)						Accuracy (%)	
	IMDb	AG's News	DBPedia	Yelp P.	Yelp F.	Avg. Δ	SNLI	Δ
ULMFiT [43]	4.60	5.01	0.80	2.16	29.98	/	/	/
$\mathrm{BERT_{BASE}}$ [13]*	5.40	5.25	0.71	2.28	30.06	/	/	/
$\mathrm{BERT_{BASE}}$	5.80	5.71	0.71	2.25	30.37	-	90.7	-
$\mathrm{BERT_{VOTE}}$ $(K = 4)$	5.60	5.41	0.67	2.03	29.44	5.44%	91.2	5.50%
$\mathrm{BERT_{AVG}}$ $(K = 4)$	5.68	5.53	0.68	2.03	30.03	4.07%	90.8	1.07%
$\mathrm{BERT_{SE}}$ (ours)	5.82	5.59	0.65	2.19	30.48	2.50%	90.8	1.07%
$\mathrm{BERT_{SDV}}$ (ours)	5.35	5.38	**0.68**	2.05	**29.88**	5.65%	**91.2**	**5.38%**
$\mathrm{BERT_{SDA}}$ (ours)	**5.29**	**5.29**	**0.68**	**2.04**	**29.88**	**6.26%**	91.2	5.38%

Improving BERT Fine-Tuning via Self-Ensemble and Self-Distillation

Optimize the Training Objective

- Model Comparison

Table 10. Comparison with Distillation-based Methods on the Development Set of the GLUE Benchmark

Model	CoLA (Mcc)	SST-2 (Acc)	QQP (Acc/F1)	MNLI-m/mm (Acc)	QNLI (Acc)
BERT$_{\text{LARGE}}$	61.8	93.5	91.1/88.0	86.3/86.2	92.4
MT-DNN	63.5	94.3	91.9/89.2	87.1/86.7	92.9
MT-DNN$_{\text{KD}}$	64.5	94.3	91.9/89.4	87.3/87.3	93.2
BERT$_{\text{SDA}}$ (ours)	63.4	94.4	91.8/88.9	87.0/86.6	92.6
BERT$_{\text{SDV}}$ (ours)	63.1	94.3	92.0/89.1	87.2/86.8	92.8

Improving BERT Fine-Tuning via Self-Ensemble and Self-Distillation

Optimization on other Perspectives

- Optimization of the Module Architecture
  - PLM not always performs well, in some challenging task such as Keyphrase Generation, Transformer even performs worse than RNNs
  - Keyphrase Generation: Given an input document X, the task aims to predict a sequence of keyphrases that contain the core idea of the input document
  - In KG tasks, uninformative content abounds in documents while salient information is diluted in the global context.

Searching Effective Transformer for Seq2Seq Keyphrase Generation

Optimization on other Perspectives

- Optimization of the Module Architecture
  - Chunking: Separate the input document manually
  - Sparse the Matrix of Attention Mask:

$$\bar{\mathbf{M}}_{i,j} = (\alpha \mathbf{M}_{i,j}^{lead}) \circ (\beta \mathbf{M}_{i,j}^{neigh}) \circ (\gamma \mathbf{M}_{i,j}^{topk})$$

  - Apply Relative Multi-head Attention:

$$\begin{aligned} \mathbf{A}_{i,j}^{abs} &= \mathbf{Q}_i \mathbf{K}_j^T \\ &= \mathbf{H}_i \mathbf{W}_q (\mathbf{H}_j \mathbf{W}_k)^T + \mathbf{H}_i \mathbf{W}_q (\mathbf{R}_{i-j} \mathbf{W}_k)^T \\ &+ \mathbf{u} (\mathbf{H}_j \mathbf{W}_k)^T + \mathbf{v} (\mathbf{R}_{i-j} \mathbf{W}_k)^T . \end{aligned}$$

Searching Effective Transformer for Seq2Seq Keyphrase Generation

Optimization on other Perspectives

- Optimization of the Module Architecture

Model	Inspec		Krapivin		SemEval		KP20k	
	$F_1@M$	$F_1@5$	$F_1@M$	$F_1@5$	$F_1@M$	$F_1@5$	$F_1@M$	$F_1@5$
ExHiRD (Chen et al., 2020)	0.291	0.253	0.347	**0.286**	0.335	0.284	0.374	0.311
ExHiRD with RNN (Our Implementation)	0.288	0.248	0.344	0.281	0.326	0.274	0.374	0.311
(Ours) ExHiRD with TF	0.278	0.232	0.329	0.272	0.310	0.258	0.364	0.300
+ SM only	0.280	0.235	0.334	0.275	0.319	0.266	0.372	0.304
+ RMHA only	0.289	0.244	0.336	0.277	0.325	0.278	0.372	0.313
+ SM + RMHA	**0.293**	**0.254**	**0.351**	**0.286**	**0.337**	**0.289**	**0.375**	**0.316**

Searching Effective Transformer for Seq2Seq Keyphrase Generation

Optimization on other Perspectives

- Optimization of the Module Architecture

N	α	β	γ	$F_1@M$	$F_1@5$	$C@M$	$C@5$	#Avg. Len
4	baseline			0.364	0.300	24,154	23,827	3.96
4	1	1	1	0.372	0.304	24,812	24,042	3.85
			0	0.367	0.302	24,905	23,929	3.95
		0		0.363	0.298	25,051	23,662	4.05
	0			0.370	0.298	24,315	23,632	3.73
6	1	1	1	0.372	0.304	24,562	23,979	3.82
			0	0.364	0.306	25,112	24,208	4.13
		0		0.366	0.296	24,177	23,347	3.90
	0			0.368	0.302	24,468	23,770	3.87

Searching Effective Transformer for Seq2Seq Keyphrase Generation

Optimization on other Perspectives

- Optimization of the Evaluation Metric
  - Traditional F1 score only considers the exact match predictions

Score(“natural language processing”, “language understanding”) = Score(“natural language processing”, “apple tree”) = 0

  - Keyphrases are short, therefore it is not suitable for n-gram-based metric

Is there any fine-grained metric for a smooth evaluation?

Keyphrase Generation with Fine-Grained Evaluation-Guided Reinforcement Learning

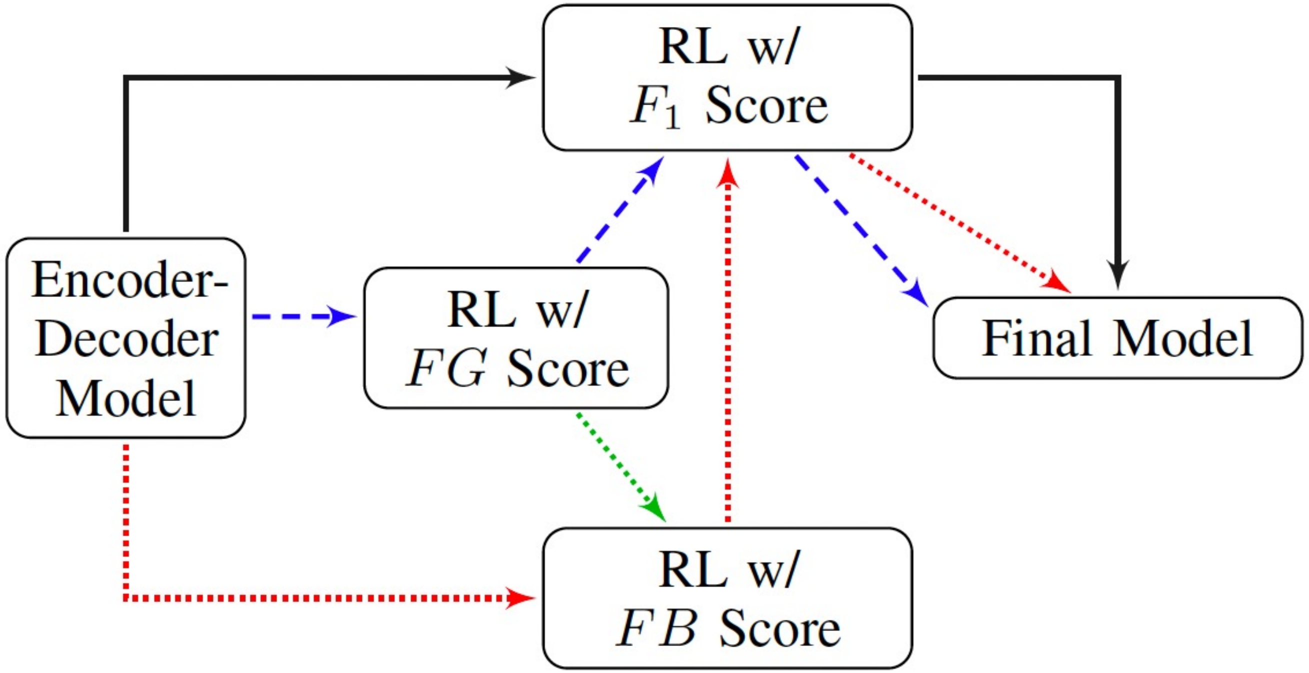
Optimization on other Perspectives

- Fine-Grained Score (FG-Score)
  - Token-level F1 Score
    - For the predicted keyphrase and the ground truth, compute the F1 score in token level
  - Token-level Edit Distance
    - Use dynamic programming to compute the edit distance in token level and then re-normed by the target length
  - Repetition Rate Penalty
    - Prevent from generating similar keyphrases
    - Penalize when the predicted words appear more times than that in the ground truth
  - Generation Quantity Penalty
    - Prevent from generating keyphrases only with high confidence
    - Penalize when the number of the predicted keyphrases is not equal to the number of the ground truth

Keyphrase Generation with Fine-Grained Evaluation-Guided Reinforcement Learning

Optimization on other Perspectives

- Optimization of the Evaluation Metric
  - Two-stage Reinforcement Learning Framework



Keyphrase Generation with Fine-Grained Evaluation-Guided Reinforcement Learning

Optimization on other Perspectives

- Optimization of the Evaluation Metric

Model	Inspec $F_1@M$	Inspec $F_1@5$	Inspec FG	Krapivin $F_1@M$	Krapivin $F_1@5$	Krapivin FG	KP20k $F_1@M$	KP20k $F_1@5$	KP20k FG
catSeq(Yuan et al., 2020)	0.262	0.225	0.381	0.354	0.269	0.352	0.367	0.291	0.371
catSeqD(Yuan et al., 2020)	0.263	0.219	0.385	0.349	0.264	0.350	0.363	0.285	0.369
catSeqCorr(Chen et al., 2018)	0.269	0.227	0.391	0.349	0.265	0.360	0.365	0.289	0.374
catSeqTG(Chen et al., 2019)	0.270	0.229	0.391	0.366	0.282	0.344	0.366	0.292	0.369
SenSeNet(Luo et al., 2020)	0.284	0.242	0.393	0.354	0.279	0.355	0.370	0.296	0.373
ExHiRD-h(Chen et al., 2020)	0.291	0.253	0.395	0.347	0.286	0.354	0.374	0.311	0.375
Utilizing RL (Chan et al., 2019)									
catSeq+RL(F_1)	0.300	0.250	0.382	0.362	0.287	0.360	0.383	0.310	0.369
catSeqD+RL(F_1)	0.292	0.242	0.380	0.360	0.282	0.357	0.379	0.305	0.377
catSeqCorr+RL(F_1)	0.291	0.240	0.392	0.369	0.286	0.376	0.382	0.308	0.377
catSeqTG+RL(F_1)	0.301	0.253	0.389	0.369	0.300	0.344	0.386	0.321	0.370
Ours									
catSeq*+RL(FG)	0.252	0.201	0.460	0.359	0.228	0.413	0.365	0.290	0.440
catSeq*+RL(FB)	0.254	0.200	**0.463**	0.354	0.230	**0.416**	0.366	0.291	**0.444**
catSeq*+2RL(FG)	0.308	0.266	0.425	**0.375**	0.304	0.389	0.391	0.327	0.381
catSeq*+2RL(FB)	**0.310**	**0.267**	0.430	0.374	**0.305**	0.390	**0.392**	**0.330**	0.383

Keyphrase Generation with Fine-Grained Evaluation-Guided Reinforcement Learning

Prompt

- Definition
  - Prompt engineering is the process to create a prompting function $f_{\text{prompt}}(x)$ that helps the PLM predicts the answer.

Type	Task	Input ([X])	Template	Answer ([Z])
Text CLS	Sentiment	I love this movie.	[X] The movie is [Z].	great fantastic ...
	Topics	He prompted the LM.	[X] The text is about [Z].	sports science ...
	Intention	What is taxi fare to Denver?	[X] The question is about [Z].	quantity city ...
Text-span CLS	Aspect Sentiment	Poor service but good food.	[X] What about service? [Z].	Bad Terrible ...
Text-pair CLS	NLI	[X1]: An old man with ... [X2]: A man walks ...	[X1]? [Z], [X2]	Yes No ...
Tagging	NER	[X1]: Mike went to Paris. [X2]: Paris	[X1] [X2] is a [Z] entity.	organization location ...
Text Generation	Summarization	Las Vegas police ...	[X] TL;DR: [Z]	The victim ... A woman ... ...
	Translation	Je vous aime.	French: [X] English: [Z]	I love you. I fancy you. ...

Prompt

- Advantages
  - Better explore the potential of PLM
  - Avoid the gap between pre-training and fine-tuning
  - Effective in many source-limited scenarios such as few-shot settings
  - Make all the tasks consistent in the same approaches

Prompt

- Challenges
  - – Prompts require carefully tuning in specific domain
  - – The interpretability of prompt is limited
  - – Fine-tuning usually has better performance in large-scaled supervised scenarios
  - – Transferability prompts have yet to be fully explored

Thank you