From Dense to Sparse: Contrastive Pruning for Better Pre-trained Language Model Compression

Runxin Xu[1*], Fuli Luo[2*], Chengyu Wang[2], Baobao Chang[1†], Jun Huang[2], Songfang Huang[2†], Fei Huang[2]

[1]Key Laboratory of Computational Linguistics, Peking University, MOE, China
[2]Alibaba Group
runxinxu@gmail.com, chbb@pku.edu.cn
{lfl259702,chengyu.wcy,huangjun.hj, songfang.hsf,f.huang}@alibaba-inc.com

Abstract

Pre-trained Language Models (PLMs) have achieved great success in various Natural Language Processing (NLP) tasks under the pre-training and fine-tuning paradigm. With large quantities of parameters, PLMs are computation-intensive and resource-hungry. Hence, model pruning has been introduced to compress large-scale PLMs. However, most prior approaches only consider *task-specific* knowledge towards downstream tasks, but ignore the essential *task-agnostic* knowledge during pruning, which may cause catastrophic forgetting problem and lead to poor generalization ability. To maintain both task-agnostic and task-specific knowledge in our pruned model, we propose **C**ontr**A**stive **P**runing (**CAP**) under the paradigm of pre-training and fine-tuning. It is designed as a general framework, compatible with both structured and unstructured pruning. Unified in contrastive learning, **CAP** enables the pruned model to learn from the pre-trained model for task-agnostic knowledge, and fine-tuned model for task-specific knowledge. Besides, to better retain the performance of the pruned model, the snapshots (i.e., the intermediate models at each pruning iteration) also serve as effective supervisions for pruning. Our extensive experiments show that adopting **CAP** consistently yields significant improvements, especially in extremely high sparsity scenarios. With only 3% model parameters reserved (i.e., 97% sparsity), **CAP** successfully achieves 99.2% and 96.3% of the original BERT performance in QQP and MNLI tasks. In addition, our probing experiments demonstrate that the model pruned by **CAP** tends to achieve better generalization ability.

Introduction

Pre-trained Language Models (PLMs), such as BERT (Devlin et al. 2019), have achieved great success in a variety of Natural Language Processing (NLP) tasks. PLMs are pre-trained in a self-supervised way, and then adapted to the downstream tasks through fine-tuning. Despite the success, PLMs are usually resource-hungry with a large number of parameters, ranging from millions (e.g., BERT) to billions (e.g., GPT-3), which leads to high memory consumption and computational overhead in practice.

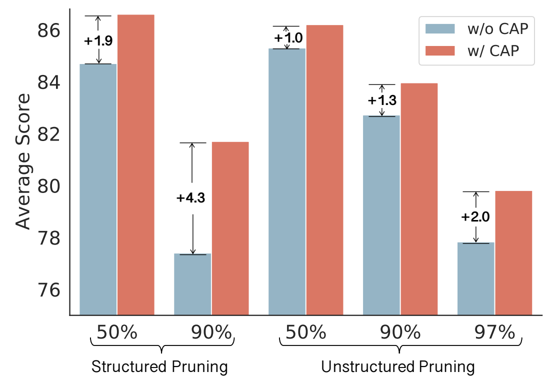


Figure 1: Comparison between BERT pruning with and without **CAP**. We report the average score across MNLI, QQP, and SQuAD tasks with different model sparsity (50%, 90%, and 97%). **CAP** consistently yield improvements for different pruning criterions, with larger gains in higher sparsity $(1.0 \rightarrow 1.3 \rightarrow 2.0)$. Please refer to Table 2 for details.

In fact, recent studies have observed that PLMs are over-parameterized with many redundant weights (Frankle and Carbin 2019; Prasanna, Rogers, and Rumshisky 2020). Motivated by this, one major line of works to compress large-scale PLMs and speed up the inference is model pruning, which focuses on identifying and removing those unimportant parameters. However, when adapting the pre-trained models to downstream tasks, most studies simply adopt the vanilla pruning methods, but do not make full use of the paradigm of pre-training and fine-tuning. Specifically, most works only pay attention to the task-specific knowledge towards the downstream task during pruning, but ignore whether the task-agnostic knowledge of the origin PLM is well maintained in the pruned model. Losing such task-agnostic knowledge can cause severe catastrophic forgetting problem (Lee, Cho, and Kang 2020; Chen et al. 2020a), which further damages the generalization ability of the pruned model. Moreover, when facing extremely high sparsity scenarios (e.g., 97% sparsity with only 3% parameters reserved), the performance of the pruned model decreases sharply compared with the original dense model.

* Equal Contribution. Joint work between Alibaba and Peking University.

† Corresponding authors.

In this paper, we propose **C**ontr**A**stive **P**runing (**CAP**), a general pruning framework under the pre-training and fine-tuning paradigm. The core of **CAP** is to encourage the pruned model to learn from multiple perspectives to reserve different types of knowledge, even in extremely high sparsity scenarios. We adopt contrastive learning (He et al. 2020; Chen et al. 2020b) to achieve the above objective with three modules: *PrC*, *SnC*, and *FiC*. These modules contrast sentence representations derived from the pruned model with those from other models, so that the pruned model is able to learn from others and reserve corresponding representation ability. Specifically, *PrC* and *FiC* strive to pull the representation from pruned model towards that from the origin pre-trained model and fine-tuned model, to learn the task-agnostic and task-specific knowledge, respectively. As a bridging mechanism, *SnC* further strives to pull the representation from pruned model towards that from the intermediate models during pruning (called snapshots), to acquire historical and diversified knowledge, so that the highly sparse model can still maintain comparable performance.

Our **CAP** framework has the following advantages: 1) **CAP** maintains both task-agnostic and task-specific knowledge in the pruned model, which helps alleviate catastrophic forgetting problem and maintain model performance during pruning, especially in extremely high sparsity cases; 2) **CAP** is based on contrastive learning that is proven to be a powerful representation learning technique; 3) **CAP** is a framework rather than a specific pruning method. Hence, it is orthogonal to various pruning criteria, including both structured and unstructured pruning, and can be flexibly integrated with them to offer improvements.

CAP is conceptually general and empirically powerful. As shown in Figure 1, our experiments show that by equipping different pruning criteria with **CAP**, the average scores across several tasks are consistently improved by up to 4.3 point, achieving the state-of-the-art performance among different pruning mechanisms. The improvement even grows larger in higher sparsity. Our experiments also demonstrate that **CAP** succeeds to achieve 99.2% and 96.3% of the original BERT performance, with only 3% model parameters in QQP and MNLI tasks. Through task transferring probing experiments, we also find that the generalization ability of the pruned model is significantly enhanced with **CAP**.

Background

Model Compression

Pre-trained Language Models (PLMs) have achieved remarkable success in NLP community, but the demanding memory and latency also greatly increase. Different compression methods, such as model pruning (Han et al. 2015; Molchanov et al. 2017), knowledge distillation (Jiao et al. 2020; Wang et al. 2020), quantization (Shen et al. 2020), and matrix decomposition (Lan et al. 2020), have been proposed.

In this paper, we mainly focus on model pruning, which identifies and removes unimportant weights of the model. It can be divided into two categories, that is, unstructured pruning that prunes individual weights, and structured pruning that prunes structured blocks of weights.

For unstructured pruning, magnitude-based methods prunes weights according to their absolute values (Han et al. 2015; Xu et al. 2021), while movement-based methods consider the change of weights during fine-tuning (Sanh, Wolf, and Rush 2020). In addition, Louizos, Welling, and Kingma (2018) use a hard-concrete distribution to exert L_0-norm regularization, and Guo et al. (2019) introduce reweighted L_1-norm regularization instead.

For structured pruning, some studies use the first-order Taylor expansion to calculate the importance scores of different heads and feed-forward networks based on the variation in the loss if we remove them (Molchanov et al. 2017; Michel, Levy, and Neubig 2019; Prasanna, Rogers, and Rumshisky 2020; Liang et al. 2021). Lin et al. (2020) prune modules whose outputs are very small. Although the above structured pruning methods are matrix-wise, there are also some studies focusing on layer-wise (Fan, Grave, and Joulin 2020; Sajjad et al. 2020), and row/column-wise (Khetan and Karnin 2020; Li et al. 2020).

Different pruning methods can be applied in a one-shot (prune for once) way, or iteratively (prune step by step) that we use in this paper. However, most of the prior methods only consider task-specific knowledge of downstream tasks, but neglect to reserve task-agnostic knowledge in the pruned model, which leads to catastrophic forgetting problem.

Contrastive Learning

Contrastive learning serves as an effective mechanism for representation learning. With similar instances considered as positive examples, and dissimilar instances as negative ones, contrastive learning aims at pulling positive examples close together and pushing negative examples apart, which usually uses InfoNCE loss (van den Oord, Li, and Vinyals 2018). He et al. (2020) and Chen et al. (2020b) propose self-supervised contrastive learning in computer vision, with different views of the figure being positive examples, and different figures being negative examples. It is also successfully introduced to NLP community, such as sentence representation (Wu et al. 2020; Gao, Yao, and Chen 2021), text summarization (Liu and Liu 2021), and so on. In order to take advantage of annotated labels of the data, some studies extend the contrastive learning in a supervised way with an arbitrary number of positive examples (Khosla et al. 2020; Gunel et al. 2021).

Formally, suppose that we have an example x_i and it is encoded into a vector representation $z_i = \phi(x_i) \in \mathbb{R}^d$ by model ϕ. Besides, there are also N examples being encoded into $\mathcal{S} = \{\hat{z}_j\}_{j=1}^{N}$, which are used to contrast with z_i. Suppose there is one or multiple positive examples $\hat{z}_p \in \mathcal{S}$ and the others $\mathcal{S}\setminus\{\hat{z}_p\}$ are negative examples towards z_i. Following Khosla et al. (2020), the contrastive training objective for example x_i is defined as follows:

$$\mathcal{L}_i = -\frac{1}{\|P(i)\|} \sum_{\hat{z}_j \in P(i)} \log \frac{e^{\mathrm{sim}(z_i,\hat{z}_j)/\tau}}{\sum_{k=1}^{N} e^{\mathrm{sim}(z_i,\hat{z}_k)/\tau}} \tag{1}$$

where $P(i) \subset \mathcal{S}$ refers to the positive examples set for z_i, $\mathrm{sim}(z_i, z_j) = \frac{z_i^\top z_j}{\|z_i\|\|z_j\|}$ refers to the cosine similarity function, and τ denotes the temperature hyperparameter.

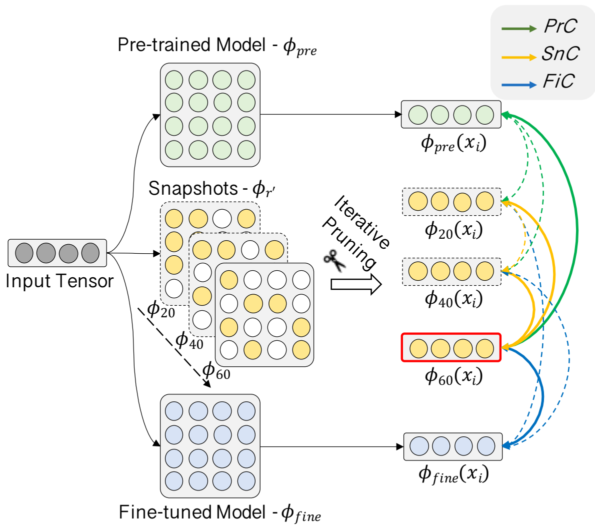


Figure 2: Overview of **CAP** framework, which prunes model step by step ($\phi_{pre} \rightarrow \phi_{20} \rightarrow \phi_{40} \rightarrow \phi_{60}$), where the number denotes the sparsity ratio (%). Overall, **CAP** consists of three contrastive modules: *PrC*, *SnC*, and *FiC*. *PrC* (green lines): contrastive learning with the pre-trained model ϕ_{pre} to maintain task-agnostic knowledge. *SnC* (yellow lines): contrastive learning with snapshots $\phi_{r'}$ to bridge the gap between pre-trained model and current pruned model, and gain historic and diversified knowledge. *FiC* (blue lines): contrastive learning with the fine-tuned model ϕ_{fine} to gain the task-specific knowledge. The solid lines indicate the learning of the current pruned model ϕ_{60}, while the dashed lines denote the learning of previous snapshots, ϕ_{20} and ϕ_{40}.

Methodology

In this paper, we propose a general pruning framework, ContrAstive Pruning (**CAP**), which prunes model via supervisions from pre-trained and fine-tuned models, and snapshots during pruning to gain different types of knowledge. Following iterative pruning, we compress the pre-trained model ϕ_{pre} to expected sparsity ratio $R\%$ progressively ($\phi_{pre} \rightarrow \phi_{r1} \rightarrow \phi_{r2} \rightarrow \cdots \rightarrow \phi_R$), and arbitrary pruning criteria can be used at each step. Figure 2 illustrates the overview of **CAP** that consists of three modules: *PrC*, *SnC*, and *FiC*. They are all based on contrastive learning, with different ways to construct positive examples shown in Table 1.

PrC: Contrastive Learning with Pre-trained Model

On the transfer learning towards a specific downstream task, the task-agnostic knowledge in the original PLM is inclined to be lost, which can cause catastrophic forgetting problem. Hence, in this section, we introduce a *PrC* module to maintain such general-purpose language knowledge based on contrastive learning (green lines in Figure 2).

Suppose that example x_i is encoded into $z_i = \phi_r(x_i) \in \mathbb{R}^d$ by model ϕ_r with $r\%$ sparsity ratio. The high-level idea is that we can contrast z_i with $\{\hat{z}_j = \phi_{pre}(x_j)\}_{j=1}^N$ encoded by pre-trained model ϕ_{pre} and enforce the model to correctly identify those semantically similar (positive) examples. In this way, the current pruned model ϕ_r is able to mimic the representation modeling ability of the pre-trained model, and therefore maintain task-agnostic knowledge.

Module	Supervised	Unsupervised
PrC	$\{\phi_{pre}(x_j) \mid y_j = y_i\}$	$\{\phi_{pre}(x_i)\}$
SnC	$\{\phi_{r'}(x_j) \mid y_j = y_i, r' < r\}$	$\{\phi_{r'}(x_i) \mid r' < r\}$
FiC	$\{\phi_{fine}(x_j) \mid y_j = y_i\}$	$\{\phi_{fine}(x_i)\}$

Table 1: Positive examples $P(i)$ in Eq. 1 for $\phi_r(x_i)$ in three contrastive modules of **CAP**. $\phi_r(x_i)$ refers to representation of x_i encoded by model ϕ_r with $r\%$ sparsity ratio, and y_i denotes the annotated label for x_i.

Specifically, we adopt contrastive learning in both unsupervised and supervised settings. For unsupervised *PrC*, $\phi_{pre}(x_i)$ is considered as a positive example for $\phi_r(x_i)$, and $\{\phi_{pre}(x_j)\}_{j \neq i}$ are negative examples. The loss $\mathcal{L}_{\text{unsup}}^{PrC}$ is then calculated following Eq. 1. For supervised *PrC*, we further utilize the sentence-level annotations of the data. For example, the sentences are labeled as *entailment*, *neutral*, or *contradiction* in the MNLI task. Intuitively, we treat those having the same labels with x_i as positive examples since they share similar semantic features, and the others as negative ones. Formally, we define the positive examples set as $\{\phi_{pre}(x_j) \mid y_j = y_i\}$, where y_i denotes the label of x_i. Then the supervised loss $\mathcal{L}_{\text{sup}}^{PrC}$ is calculated as Eq. 1. Therefore, the final training objective for *PrC* is $\mathcal{L}^{PrC} = \mathcal{L}_{\text{unsup}}^{PrC} + \mathcal{L}_{\text{sup}}^{PrC}$.

SnC: Contrastive Learning with Snapshots

Pruning can be applied in a one-shot or iterative way. *One-shot pruning* drops out weights and retrains the model for once. In contrast, *iterative pruning* removes weights step by step, until reaching the expected sparsity, and the intermediate models at each pruning iterations are called **snapshots**. In this paper, we adopt iterative pruning since it better suits high sparsity regimes. However, different from prior studies that simply ignore these snapshots, we propose *SnC* to enable the current pruned model to learn from these snapshots based on contrastive learning (yellow lines in Figure 2).

In detail, suppose that we prune the model to $r\%$ sparsity ratio progressively ($\phi_{pre} \rightarrow \phi_{r1} \rightarrow \phi_{r2} \rightarrow \cdots \rightarrow \phi_r$), and $\{\phi_{r'}\}_{r'<r} = \{\phi_{r1}, \phi_{r2}, \dots\}$ are snapshots. Intuitively, these snapshots can bridge the gap between the sparse model (ϕ_r) and dense models (ϕ_{pre}, ϕ_{fine}), and provide diversified supervisions with different sparse structures. Under unsupervised settings, for example x_i encoded into $\phi_r(x_i) \in \mathbb{R}^d$ by the current pruned model ϕ_r, we treat the representations encoded from the same example but by the snapshots $\{\phi_{r'}(x_i) \mid r' < r\}$ as positive examples, and $\{\phi_{r'}(x_j) \mid j \neq i, r' < r\}$ as negative ones. Under supervised settings, we utilize the annotation labels to consider instances with the same labels as positive examples. We calculate the loss for *SnC* following Eq. 1, $\mathcal{L}^{SnC} = \mathcal{L}_{\text{unsup}}^{SnC} + \mathcal{L}_{\text{sup}}^{SnC}$. Table 3 show that snapshots serve as effective guidance during pruning, with $0.58 \sim 2.47$ average gains on MNLI, QQP, and SQuAD, especially in high sparsity regimes.

FiC: Contrastive Learning with Fine-tuned Model

To better adapt to the downstream task, the pruned model ϕ_r can also learn from the fine-tuned model ϕ_{fine} that contains rich task-specific knowledge. To this end, we propose a *FiC* module, which conducts contrastive learning between the current pruned model ϕ_r and the fine-tuned model ϕ_{fine}. It is almost identical to the *PrC* module, except that the target model ϕ_{pre} is replaced with the fine-tuned model ϕ_{fine} (blue lines in Figure 2). Accordingly, the training loss is calculated as $\mathcal{L}^{FiC} = \mathcal{L}^{FiC}_{\text{unsup}} + \mathcal{L}^{FiC}_{\text{sup}}$ based on Eq. 1. In addition to the contrastive supervision signal in representation space, we can also introduce distilling supervision in label space through knowledge distillation mechanism.

Pruning with CAP Framework

Putting *PrC*, *SnC*, and *FiC* together, we can reach our proposed **CAP** framework. Note that we can flexibly integrate with different pruning criteria in **CAP**. In this paper, we try out both structured and unstructured pruning criteria.

For structured pruning, a widely used structured pruning criterion is to derive the importance score of an element based on the variation towards the loss $\mathcal{L}$ if we remove it, using the first-order Taylor expansion (Molchanov et al. 2017) of the loss. We denote this method as *First-order pruning* and absorb it into **CAP**, which we call **CAP-f**.

$$I_w = |\mathcal{L}_w - \mathcal{L}_{w=0}| \approx \left| \frac{\partial \mathcal{L}}{\partial w} w \right| \tag{2}$$

For unstructured pruning, we apply the movement-based pruning methods (Sanh, Wolf, and Rush 2020), which calculates importance score for parameter w as follows:

$$I_w = -\sum_t \frac{\partial \mathcal{L}^{(t)}}{\partial w} w^{(t)} \tag{3}$$

where t is the training step. Based on it, Sanh, Wolf, and Rush (2020) reserve parameters using Top-K selection strategy or a pre-defined threshold, called *Movement pruning* or *Soft-movement pruning*, respectively. We absorb these methods into **CAP**, and denote them as **CAP-m** and **CAP-soft**.

Finally, we prune and train the model using the final objective $\mathcal{L} = \lambda_1 \mathcal{L}^{CE} + \lambda_2 \mathcal{L}^{PrC} + \lambda_3 \mathcal{L}^{SnC} + \lambda_4 \mathcal{L}^{FiC}$, where $\mathcal{L}^{CE}$ is the cross-entropy loss towards the downstream task.

Extra Memory Overhead

In our proposed **CAP** framework, the pruned model learns from the pre-trained, snapshots, and fine-tuned models. However, there is unnecessary to load all of these models in GPU, which can lead to large GPU memory overhead. In fact, because only the sentence representations of examples are required for Eq. 1 in contrastive learning, and they also do not back-propagate the gradients, we can simply pre-encode the examples and store them in CPU. When a normal input batch arrives, we fetch N pre-encoded examples for contrastive learning. In our paper, we use $N = 4096$ by default, and the dimensions of the sentence representation for $\text{BERT}_{\text{base}}$ are 768. Therefore, the extra GPU memory overhead is $4096 \times 768 = 3.15\text{M}$ in total, and only takes up 3.15M / 110M = 2.86% of the memory consumption of $\text{BERT}_{\text{base}}$, which is low enough and acceptable.

Experiments

Datasets

We conduct experiments on various tasks to illustrate the effectiveness of **CAP**, including a) **MNLI**, the Multi-Genre Natural Language Inference Corpus (Williams, Nangia, and Bowman 2018), a natural language inference task with in-domain test set (MNLI-m), and cross-domain one (MNLI-mm). b) **QQP**, the Quora Question Pairs dataset (Wang et al. 2019), a pairwise semantic equivalence task. c) **SST-2**, the Stanford Sentiment Treebank (Socher et al. 2013), a sentiment classification task for an individual sentence. d) **SQuAD v1.1**, the Stanford Question Answering Dataset (Rajpurkar et al. 2016), an extractive question answering task with crowdsourced question-answer pairs. Following most prior works (Lin et al. 2020; Sanh, Wolf, and Rush 2020), we report results for the dev sets. The detailed statistics and the metrics are provided in Appendix.

Experiment Setups

We conduct experiments based on $\text{BERT}_{\text{base}}$ (Devlin et al. 2019) with 110M parameters, and follow their settings unless noted otherwise. Following Sanh, Wolf, and Rush (2020), we prune and report the sparsity ratio based on the weights of the encoder. For **CAP-f**, we prune 10% parameters each step and retrain the model to recover the performance until reaching the expected sparsity. For **CAP-m** and **CAP-soft**, we follow the cubic sparsity scheduling with cool-down strategy and hyperparameter settings the same as Sanh, Wolf, and Rush (2020). The number of examples for contrastive learning is $N = 4096$. We use the final hidden state of $[CLS]$ as the sentence representation, which is shown to be slightly better than mean pooling in our exploration experiments. We search the temperature from $\tau = \{0.05, 0.1, 0.2, 0.3\}$. [1]

Main Results

In this section, we compare **CAP** with the following model compression methods: 1) *knowledge distillation*: DistillBERT (Sanh et al. 2019), BERT-PKD (Sun et al. 2019), TinyBERT (Jiao et al. 2020), and MiniLM (Wang et al. 2020). Note that we report the results of TinyBERT without data augmentation mechanism to ensure fairness. 2) *structured pruning*: the most standard First-order pruning (Molchanov et al. 2017) that **CAP-f** is based, Top-drop (Sajjad et al. 2020), SNIP (Lin et al. 2020), and schuBERT (Khetan and Karnin 2020). 3) *unstructured pruning*: Magnitude pruning (Han et al. 2015), L_0-regularization (Louizos, Welling, and Kingma 2018), and the state-of-the-art Movement pruning and Soft-movement pruning (Sanh, Wolf, and Rush 2020) that our **CAP-m** and **CAP-soft** are based on. Please refer to the Appendix for more details about the baselines. Table 2 illustrates the main results, from which we have some important observations.

(1) **CAP** *removes a large proportion of BERT parameters while still maintaining comparable performance*. With 50%

[1] Our code is available at https://github.com/alibaba/AliceMind/tree/main/ContrastivePruning and https://github.com/PKUnlp-icler/ContrastivePruning.

Methods	Sparsity	MNLI-m/-mm	$\mathrm{QQP}_{\mathrm{ACC/F1}}$	SST-2	$\mathrm{SQuAD}_{\mathrm{EM/F1}}$
$\mathrm{BERT}_{\mathrm{base}}$	0%	84.5 / 84.4	90.9 / 88.0	92.9	80.7 / 88.4
Knowledge Distillation					
DistillBERT (Sanh et al. 2019)	50%	82.2 / -	88.5 / -	91.3	78.1 / 86.2
BERT-PKD (Sun et al. 2019)	50%	- / 81.0	88.9 / -	91.5	77.1 / 85.3
TinyBERT (Jiao et al. 2020)	50%	83.5 / -	90.6 / -	91.6	79.7 / 87.5
MiniLM (Wang et al. 2020)	50%	84.0 / -	91.0 / -	92.0	- / -
TinyBERT (Jiao et al. 2020)	66.7%	80.5 / 81.0	- / -	-	72.7 / 82.1
Structured Pruning					
First-order (Molchanov et al. 2017)	50%	83.2 / 83.6	90.8 / 87.5	90.6	77.2 / 86.0
Top-drop (Sajjad et al. 2020)	50%	81.1 / -	90.4 / -	90.3	- / -
SNIP (Lin et al. 2020)	50%	- / 82.8	88.9 / -	91.8	- / -
schuBERT (Khetan and Karnin 2020)	50%	83.8 / -	- / -	91.7	80.7 / 88.1
CAP-f (Ours)	50%	**84.5 / 85.0**	**91.6 / 88.6**	**92.7**	**81.4 / 88.7**
SNIP (Lin et al. 2020)	75%	- / 78.3	87.8 / -	88.4	- / -
First-order (Molchanov et al. 2017)	90%	79.1 / 79.5	88.7 / 84.9	86.9	59.8 / 72.3
CAP-f (Ours)	90%	**81.0 / 81.2**	**90.2 / 86.9**	**89.7**	**70.2 / 80.6**
Unstructured Pruning					
Movement (Sanh, Wolf, and Rush 2020)	50%	82.5 / 82.9	91.0 / 87.8	-	79.8 / 87.6
CAP-m (Ours)	50%	**83.8 / 84.2**	**91.6 / 88.6**	-	**80.9 / 88.2**
Magnitude (Han et al. 2015)	90%	78.3 / 79.3	79.8 / 65.0	-	70.2 / 80.1
L_0-regularization (Louizos, Welling, and Kingma 2018)	90%	78.7 / 79.7	88.1 / 82.8	-	72.4 / 81.9
Movement (Sanh, Wolf, and Rush 2020)	90%	80.1 / 80.4	89.7 / 86.2	-	75.6 / 84.3
Soft-Movement (Sanh, Wolf, and Rush 2020)	90%	81.2 / 81.8	90.2 / 86.8	-	76.6 / 84.9
CAP-m (Ours)	90%	81.1 / 81.8	91.6 / 87.7	-	76.5 / 85.1
CAP-soft (Ours)	90%	**82.0 / 82.9**	**90.7 / 87.4**	-	**77.1 / 85.6**
Movement (Sanh, Wolf, and Rush 2020)	97%	76.5 / 77.4	86.1 / 81.5	-	67.5 / 78.0
Soft-Movement (Sanh, Wolf, and Rush 2020)	97%	79.5 / 80.1	89.1 / 85.5	-	72.7 / 82.3
CAP-m (Ours)	97%	77.5 / 78.4	88.8 / 85.0	-	69.5 / 79.7
CAP-soft (Ours)	97%	**80.1 / 81.3**	**90.2 / 86.7**	-	**73.8 / 83.0**

Table 2: Comparison between **CAP** with other model compression methods without data augmentation. **CAP** consistently achieve the best performance under the same sparsity ratio across different tasks. With only 3% of the encoder's parameter (i.e., 97% sparsity), **CAP-soft** still reaches 99.2% and 96.3% of the original BERT performance in QQP and MNLI task, respectively.

sparsity ratio, **CAP-f** achieves an equal score in MNLI-m compared with origin BERT, and even improves by $0.3 \sim 0.7$ score for MNLI-mm, QQP, and SQuAD tasks. More importantly, with only 3% of the encoder's parameters (i.e., 97% sparsity ratio), our **CAP-soft** successfully achieves 99.2% and 96.3% of the original BERT performance in QQP ($90.9 \rightarrow 90.2$) and MNLI-mm ($84.4 \rightarrow 81.3$).

(2) **CAP** *consistently yields improvements for different pruning criteria, along with larger gains in higher sparsity*. Compared with structured First-order pruning, **CAP-f** improves by $1.9(85.6 \rightarrow 87.5)$ and $4.1(78.7 \rightarrow 82.8)$ average score over all tasks under 50% and 90% sparsity. Similarly, compared with unstructured Movement pruning, as the sparsity grows by $50\% \rightarrow 90\% \rightarrow 97\%$, **CAP-m** improves $1.0(85.2 \rightarrow 86.2)$, $1.3(82.7 \rightarrow 84.0)$, and $2.0(77.8 \rightarrow 79.8)$ scores, which also increases accordingly.

(3) **CAP** *consistently outperforms other pruning methods*. For example, **CAP-f** surpasses SNIP by 2.2 and 2.7 score in MNLI-mm and QQP tasks under 50% sparsity. Besides, with higher sparsity, the 90%-sparsified **CAP-f** can even beat SNIP with 75% sparsity, with 2.9 and 2.4 higher score in the MNLI-mm and QQP tasks.

(4) **CAP** *also outperforms knowledge distillation methods*. For example, compared with TinyBERT under 66.7% sparsity ratio, **CAP-f** that is under 90% sparsity ratio can still surpass it by 0.5 accuracy in the MNLI-m task.

The above observations support our claim that **CAP** helps the pruned model to maintain both task-agnostic and task-specific knowledge and hence benefits the pruning, especially under extremely high sparsity scenarios.

Generalization Ability

Different from prior methods, **CAP** can maintain task-agnostic knowledge in the pruned model, and therefore strengthen its generalization ability. To justify our claim, we follow the task transfer probing experiments from Aghajanyan et al. (2021). In detail, we freeze the representations derived from the pruned model trained on MNLI or QQP,

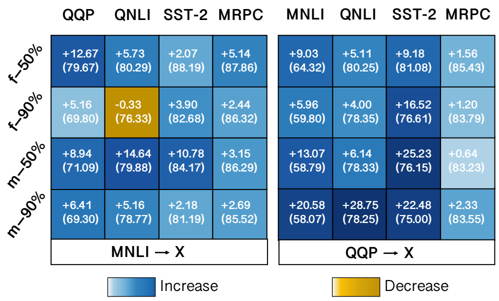


Figure 3: Generalization ability probing. We transfer the pruned model trained on MNLI (left) and QQP (right) to target tasks under 50% and 90% sparsity ratio. We report the improvement brought by **CAP** compared with its basic pruning method, and the absolute performance of **CAP** (in bracket). **CAP** yields improvement in most cases, suggesting better generalization ability of the model pruned by **CAP**.

and then only train a linear classifier on top of the model for another task. Besides MNLI, QQP, and SST-2 tasks we have used, we also include QNLI (Wang et al. 2019) and MRPC (Dolan and Brockett 2005) as our target tasks[2].

The results under 50% and 90% sparsity are shown in Figure 3, where the first two rows correspond to the improvement of **CAP-f** over First-order pruning, and the last two rows correspond to the improvement of **CAP-m** over Movement pruning. Improvements are shown at each cell, with the performance score of **CAP** in the bracket. As is shown, **CAP** yields improvements in an overwhelming majority of cases. For example, **CAP-m** outperforms Movement pruning by a large margin, with up to 28.75 higher score in QNLI transferred from QQP task, under 90% sparsity. The significant improvements in task transferring experiments suggest **CAP** can better maintain the task-agnostic knowledge and strengthen the generalization ability of the pruned model.

Discussions

Understanding Different Contrastive Modules

CAP is mainly comprised of three major contrastive modules, *PrC* for learning from the pre-trained model, *SnC* for learning from snapshots, and *FiC* for learning from the fine-tuned model. To better explore their effects, we remove one of them from **CAP** at a time, and prunes model with **CAP-f** and **CAP-m** methods under various sparsity ratios.

We report the accuracy for MNLI-m, F1 for QQP, and Exact Match score for SQuAD, along with the average score decrease Δ. As shown in Table 3, removing any contrastive module would cause degradation of the model. For example, with 90% sparsity ratio, removing *PrC*, *SnC*, or *FiC* leads to $1.83 \sim 2.59$ average score reduction for **CAP-f**, and $0.58 \sim 1.07$ for **CAP-m**. Besides, the performance degradation gets larger as the sparsity gets higher in general. This is especially evident for **CAP-f**. For instance, using **CAP-f** without *PrC* module, when the sparsity ratio varies from

Methods	Ratio	MNLI-m	QQP	SQuAD	Δ
CAP-f	50%	84.48	88.49	81.37	-
- w/o *PrC*		-0.47	-0.50	-0.67	-0.55
- w/o *SnC*		-0.47	-0.57	-1.10	-0.71
- w/o *FiC*		-0.42	-0.63	-0.58	-0.54
CAP-f	90%	80.98	86.92	70.16	-
- w/o *PrC*		-1.78	-0.87	-5.11	-2.59
- w/o *SnC*		-1.57	-1.72	-4.11	-2.47
- w/o *FiC*		-1.53	-1.31	-2.64	-1.83
CAP-m	50%	83.29	88.28	80.40	-
- w/o *PrC*		-0.84	-0.22	-0.74	-0.60
- w/o *SnC*		-0.74	-0.36	-0.66	-0.59
- w/o *FiC*		-1.31	-0.77	-1.07	-1.05
CAP-m	90%	80.53	87.12	76.20	-
- w/o *PrC*		-0.58	-0.47	-0.68	-0.58
- w/o *SnC*		-0.45	-0.78	-0.66	-0.58
- w/o *FiC*		-1.54	-0.73	-0.94	-1.07
CAP-m	97%	77.30	84.70	69.47	-
- w/o *PrC*		-0.27	-0.23	-1.87	-0.79
- w/o *SnC*		-0.29	-0.52	-1.67	-0.83
- w/o *FiC*		-1.31	-0.34	-2.11	-1.25

Table 3: Ablation study of different contrastive modules. Δ refers to the average score reduction across all tasks. Removing any contrastive module would cause degradation of the pruned model, especially in highly sparse regimes.

50% to 90%, the degradation increases from 0.55 to 2.59, The results demonstrate that all contrastive modules play important roles in **CAP**, and have complementary advantages with each other, especially in highly sparse regimes.

Understanding Supervised and Unsupervised Contrastive Objectives

In **CAP**, the same example encoded by different models are considered as positive examples for unsupervised contrastive learning (unsup). If the sentence-level label annotations are available, we can also conduct supervised contrastive learning (sup) by considering examples with the same labels as positive examples. To explore their effects, we remove either of them for the ablation study.

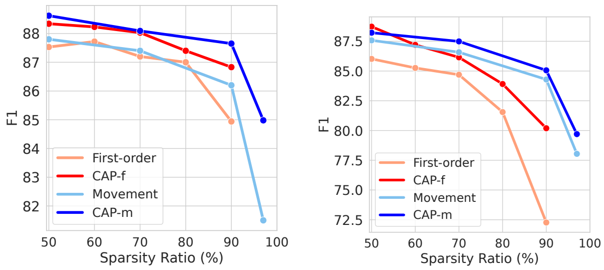
Our experiments [3] demonstrates that without either supervised or unsupervised contrastive learning objectives, the performance of the pruned model would markedly decline. Specifically, removing the supervised contrastive learning objective leads to $0.53 \sim 0.91$ average score decrease for **CAP**, while abandoning the unsupervised one also causes $0.49 \sim 1.25$ decrease. It suggests that both supervised and unsupervised objectives are essential and necessary for **CAP**, and their advantages are orthogonal to each other.

Performance under Various Sparsity Ratios

In this section, we compare **CAP-f** and **CAP-m** with their basic pruning methods, First-order pruning and Movement

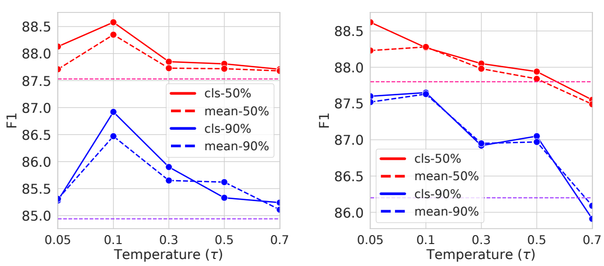
[2]We report F1 for QQP/MRPC and accuracy for other tasks.

[3]Due to limited space, you can find the experimental results in our arXiv version https://arxiv.org/abs/2112.07198.



(a) QQP (b) SQuAD

Figure 4: Comparison between **Cap** and its basic pruning methods. **Cap-f** consistently outperforms First-order pruning, and **Cap-m** also surpasses Movement pruning under various sparsity ratios.



(a) **Cap-f** (b) **Cap-m**

Figure 5: Performance of different pooling methods and temperatures. The horizontal dashed line denotes the performance of First-order and Movement pruning under 50% and 90% sparsity. In general, using $[CLS]$ as sentence representations slightly outperforms mean pooling, and temperature $\tau = 0.1$ tends to achieve the best results.

pruning, with sparsity ratios varying from 50% to 97%. The evolution of the performance in QQP and SQuAD task is shown in Figure 4. The performance of the pruned model for all methods decreases as the sparsity ratio increases. However, we can observe that **Cap-f** consistently outperforms First-order pruning, and the improvement is even larger in higher sparsity situations. A similar tendency also exists between **Cap-m** and Movement pruning. These results suggest that with three core contrastive modules, **Cap** can better maintain the model performance during the pruning.

Exploration of Pooling Methods and Temperatures

The contrastive loss function in Eq. 1 involves two important points, the sentence representation z_i and the temperature τ. For sentence representation z_i, we explore two pooling methods of the hidden states encoded by the model, using the vector representation of the $[CLS]$ or the mean pooling of all representations of the whole sentence. For temperature τ, we also explore different values ranging from 0.05 to 0.7. We conduct experiments on QQP. As shown in Figure 5, using $[CLS]$ as sentence representations slightly outperforms the mean pooling method. Besides, **Cap** can achieve better performance than its basic pruning method under most temperatures, and setting $\tau = 0.1$ tends to achieve the best performance in most cases.

Methods	**QQP**		**SQuAD**	
	50%	90%	50%	90%
Movement	87.80	86.20	87.58	84.29
- w/o KD	87.30	83.20	83.16	81.72
Cap-f	88.58	86.92	88.73	80.59
- w/o KD	88.59	86.88	86.52	77.76
Cap-m	88.62	87.65	88.22	85.06
- w/o KD	88.60	87.67	85.94	82.46

Table 4: Exploring of learning from the fine-tuned model. For **Cap**, KD brings improvements for token-level task (SQuAD), but has little effect on sentence-level task (QQP).

Exploration of Learning From Fine-tuned Model

To gain task-specific knowledge, we propose to perform contrastive learning on the sentence representations from the fine-tuned model (*FiC*). Another common way is to perform the knowledge distillation (KD) on the soft label that has already been shown effective in pruning (Sanh, Wolf, and Rush 2020; Hou et al. 2020). To explore the effect of KD, we conduct further experiments in Table 4. It shows that KD boosts the performance of CAP in token-level task (SQuAD) while has little effect on sentence-level task (QQP). The reason can be that the contrastive learning on the sentence representations is sufficient to capture the features of the sentence-level task, while the information still incurs losses on token-level tasks. Thus, for token-level tasks, it is better to perform **Cap** with KD.

Conclusion

In this paper, we propose a general pruning framework, **C**ontr**A**stive **P**runing (**Cap**), under the paradigm of pre-training and fine-tuning. Based on contrastive learning, we enhance the pruned model to maintain both task-agnostic and task-specific knowledge via pulling it towards the representations from the pre-trained model ϕ_{pre}, and fine-tuned model ϕ_{fine}. Furthermore, the snapshots during the pruning process are also fully utilized to provide historic and diversified supervisions to retain the performance of the pruned model, especially in high sparsity regimes. **Cap** consistently yields significant improvements to different pruning criteria, and achieves the state-of-the-art performance among different pruning mechanisms. Experiments also show that **Cap** strengthen the generalization ability of the pruned model.

Acknowledgments

This paper is supported by the National Key Research and Development Program of China under Grant No. 2020AAA0106700, the National Science Foundation of China under Grant No.61936012 and 61876004.

References

Aghajanyan, A.; Shrivastava, A.; Gupta, A.; Goyal, N.; Zettlemoyer, L.; and Gupta, S. 2021. Better Fine-Tuning by Reducing Representational Collapse. In *International Conference on Learning Representations (ICLR)*.

Chen, S.; Hou, Y.; Cui, Y.; Che, W.; Liu, T.; and Yu, X. 2020a. Recall and Learn: Fine-tuning Deep Pretrained Language Models with Less Forgetting. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Chen, T.; Kornblith, S.; Norouzi, M.; and Hinton, G. 2020b. A Simple Framework for Contrastive Learning of Visual Representations. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Dolan, W. B.; and Brockett, C. 2005. Automatically Constructing a Corpus of Sentential Paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP)*.

Fan, A.; Grave, E.; and Joulin, A. 2020. Reducing Transformer Depth on Demand with Structured Dropout. In *International Conference on Learning Representations (ICLR)*.

Frankle, J.; and Carbin, M. 2019. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. In *International Conference on Learning Representations (ICLR)*.

Gao, T.; Yao, X.; and Chen, D. 2021. SimCSE: Simple Contrastive Learning of Sentence Embeddings. *arXiv*, arXiv:2104.08821.

Gunel, B.; Du, J.; Conneau, A.; and Stoyanov, V. 2021. Supervised Contrastive Learning for Pre-trained Language Model Fine-tuning. In *International Conference on Learning Representations (ICLR)*.

Guo, F.; Liu, S.; Mungall, F. S.; Lin, X.; and Wang, Y. 2019. Reweighted Proximal Pruning for Large-Scale Language Representation. *arXiv*, arXiv:1909.12486.

Han, S.; Pool, J.; Tran, J.; and Dally, W. 2015. Learning both Weights and Connections for Efficient Neural Network. In *Advances in Neural Information Processing Systems (NeurIPS)*.

He, K.; Fan, H.; Wu, Y.; Xie, S.; and Girshick, R. 2020. Momentum Contrast for Unsupervised Visual Representation Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Hou, L.; Huang, Z.; Shang, L.; Jiang, X.; Chen, X.; and Liu, Q. 2020. DynaBERT: Dynamic BERT with Adaptive Width and Depth. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Jiao, X.; Yin, Y.; Shang, L.; Jiang, X.; Chen, X.; Li, L.; Wang, F.; and Liu, Q. 2020. TinyBERT: Distilling BERT for Natural Language Understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*.

Khetan, A.; and Karnin, Z. 2020. schuBERT: Optimizing Elements of BERT. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Khosla, P.; Teterwak, P.; Wang, C.; Sarna, A.; Tian, Y.; Isola, P.; Maschinot, A.; Liu, C.; and Krishnan, D. 2020. Supervised Contrastive Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Lan, Z.; Chen, M.; Goodman, S.; Gimpel, K.; Sharma, P.; and Soricut, R. 2020. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. In *International Conference on Learning Representations (ICLR)*.

Lee, C.; Cho, K.; and Kang, W. 2020. Mixout: Effective Regularization to Finetune Large-scale Pretrained Language Models. In *International Conference on Learning Representations (ICLR)*.

Li, B.; Kong, Z.; Zhang, T.; Li, J.; Li, Z.; Liu, H.; and Ding, C. 2020. Efficient Transformer-based Large Scale Language Representations using Hardware-friendly Block Structured Pruning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*.

Liang, C.; Zuo, S.; Chen, M.; Jiang, H.; Liu, X.; He, P.; Zhao, T.; and Chen, W. 2021. Super Tickets in Pre-Trained Language Models: From Model Compression to Improving Generalization. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL)*.

Lin, Z.; Liu, J.; Yang, Z.; Hua, N.; and Roth, D. 2020. Pruning Redundant Mappings in Transformer Models via Spectral-Normalized Identity Prior. In *Findings of the Association for Computational Linguistics: EMNLP 2020*.

Liu, Y.; and Liu, P. 2021. SimCLS: A Simple Framework for Contrastive Learning of Abstractive Summarization. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL)*.

Louizos, C.; Welling, M.; and Kingma, D. P. 2018. Learning Sparse Neural Networks through L_0 Regularization. In *International Conference on Learning Representations (ICLR)*.

Michel, P.; Levy, O.; and Neubig, G. 2019. Are Sixteen Heads Really Better than One? In *Advances in Neural Information Processing Systems (NeurIPS)*.

Molchanov, P.; Tyree, S.; Karras, T.; Aila, T.; and Kautz, J. 2017. Pruning Convolutional Neural Networks for Resource Efficient Inference. In *International Conference on Learning Representations (ICLR)*.

Prasanna, S.; Rogers, A.; and Rumshisky, A. 2020. When BERT Plays the Lottery, All Tickets Are Winning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Rajpurkar, P.; Zhang, J.; Lopyrev, K.; and Liang, P. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2383–2392.

Sajjad, H.; Dalvi, F.; Durrani, N.; and Nakov, P. 2020. Poor Man's BERT: Smaller and Faster Transformer Models. *arXiv*, abXiv:2004.03844.

Sanh, V.; Debut, L.; Chaumond, J.; and Wolf, T. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv*, arXiv:1910.01108.

Sanh, V.; Wolf, T.; and Rush, A. 2020. Movement Pruning: Adaptive Sparsity by Fine-Tuning. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Shen, S.; Dong, Z.; Ye, J.; Ma, L.; Yao, Z.; Gholami, A.; Mahoney, M. W.; and Keutzer, K. 2020. Q-BERT: Hessian Based Ultra Low Precision Quantization of BERT. *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

Socher, R.; Perelygin, A.; Wu, J.; Chuang, J.; Manning, C. D.; Ng, A.; and Potts, C. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Sun, S.; Cheng, Y.; Gan, Z.; and Liu, J. 2019. Patient Knowledge Distillation for BERT Model Compression. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

van den Oord, A.; Li, Y.; and Vinyals, O. 2018. Representation Learning with Contrastive Predictive Coding. *arXiv*, arXiv:1807.03748.

Wang, A.; Singh, A.; Michael, J.; Hill, F.; Levy, O.; and Bowman, S. R. 2019. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *International Conference on Learning Representations (ICLR)*.

Wang, W.; Wei, F.; Dong, L.; Bao, H.; Yang, N.; and Zhou, M. 2020. MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Williams, A.; Nangia, N.; and Bowman, S. 2018. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Wu, Z.; Wang, S.; Gu, J.; Khabsa, M.; Sun, F.; and Ma, H. 2020. CLEAR: Contrastive Learning for Sentence Representation. *arXiv*, arXiv:2012.15466.

Xu, D.; Yen, I. E.-H.; Zhao, J.; and Xiao, Z. 2021. Rethinking Network Pruning – under the Pre-train and Fine-tune Paradigm. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.