Preserving Commonsense Knowledge from Pre-trained Language Models via Causal Inference

Junhao Zheng, Qianli Ma*, Shengjie Qiu, Yue Wu, Peitian Ma,
Junlong Liu, Huawen Feng, Xichen Shang and **Haibin Chen**
School of Computer Science and Engineering,
South China University of Technology, Guangzhou, China
junhaozheng47@outlook.com, qianlima@scut.edu.cn*

Abstract

Fine-tuning has been proven to be a simple and effective technique to transfer the learned knowledge of Pre-trained Language Models (PLMs) to downstream tasks. However, vanilla fine-tuning easily overfits the target data and degrades the generalization ability. Most existing studies attribute it to catastrophic forgetting, and they retain the pre-trained knowledge indiscriminately without identifying what knowledge is transferable. Motivated by this, we frame fine-tuning into a causal graph and discover that the crux of catastrophic forgetting lies in the missing causal effects from the pre-trained data. Based on the causal view, we propose a unified objective for fine-tuning to retrieve the causality back. Intriguingly, the unified objective can be seen as the sum of the vanilla fine-tuning objective, which learns new knowledge from target data, and the causal objective, which preserves old knowledge from PLMs. Therefore, our method is flexible and can mitigate negative transfer while preserving knowledge. Since endowing models with commonsense is a long-standing challenge, we implement our method on commonsense QA with a proposed heuristic estimation to verify its effectiveness. In the experiments, our method outperforms state-of-the-art fine-tuning methods on all six commonsense QA datasets and can be implemented as a plug-in module to inflate the performance of existing QA models. [1]

1 Introduction

Deep Pre-trained Language Models (PLMs) such as RoBERTa (Liu et al., 2019b) and T5 (Raffel et al., 2020)) are inherently knowledge bases since they are exposed to a tremendous amount of data (e.g., the C4 dataset (Raffel et al., 2020)) in the pre-training stage (Petroni et al., 2019; AlKhamissi et al., 2022). Unfortunately, transferring the intrinsic knowledge in PLMs to downstream tasks is non-trivial. In practice, fine-tuning is adopted widely due to its flexibility (Chen et al., 2020) and numerous improved methods (Lee et al., 2019; Chen et al., 2020, 2019; Mosbach et al., 2020; Zhang et al., 2020b; Xu et al., 2021a; Aghajanyan et al., 2020; Wu et al., 2022) are proposed in recent years. However, fine-tuning faces two challenges when adapting models to new domains (Chen et al., 2019), including catastrophic forgetting (Kirkpatrick et al., 2017) and negative transfer (Torrey and Shavlik, 2010). More specifically, catastrophic forgetting refers to models losing previously learned knowledge and overfitting the target domain data. Negative transfer occurs because not all pre-trained knowledge is transferable across domains. Obviously, catastrophic forgetting and negative transfer constitute a dilemma where the crux lies in identifying and utilizing transferable knowledge.

A large body of previous work has been conducted to solve this problem. Existing fine-tuning methods for mitigating catastrophic forgetting can be summarized as preventing the fine-tuned models from deviating too far from the pre-trained weights. For example, *RecAdam* (Chen et al., 2020) and *Child-Tuning* (Xu et al., 2021a) utilize the Fisher Information Matrix estimated by the pre-trained model to constraint the update in the fine-tuned model. *Mixout* (Lee et al., 2019) randomly replaces the model parameters with their pre-trained weights. These methods constrain the update of models' parameters indiscriminately without identifying what knowledge is transferable and thus susceptible to negative transfer. Chen et al. (2019) proposed *BSS*, which focuses on mitigating negative transfer by penalizing the small singular values of the feature matrix. However, when only negative transfer is concerned, *BSS* may not fully utilize the pre-trained knowledge.

*Corresponding author

[1] Our codes are publicly available at https://github.com/zzz47zzz/CET and https://github.com/qianlima-lab/CET

In this paper, we propose a novel method called ***C**ausal **E**ffect **T**uning* (**CET**) for mining the pre-trained knowledge in PLMs. Unlike the previous fine-tuning method, our method is rooted in the theory of causal inference. It delves into the causalities between data, models, and features instead of merely statistical association. First, we frame vanilla fine-tuning into a causal graph (Glymour et al., 2016) and find out that the cause of catastrophic forgetting is the vanishing causal effects of pre-trained data. Therefore, preventing forgetting is to maximize the causal effect. Then, we approximate the causal effect with the likelihood of the joint prediction of K-Nearest-Neighbor (KNN) samples. Since equipping models with commonsense knowledge is still challenging, we implement the proposed causal graph with a heuristic approximation on commonsense QA. We measure the distance with the similarity between gold answers (i.e., ground-truth answers) instead of questions for retrieving KNNs. The rationale is that the questions with the same gold answer share the same commonsense knowledge in PLMs. Finally, we apply our method to RoBERTa (Liu et al., 2019b) and T5 (Raffel et al., 2020) and conduct extensive experiments on six commonsense datasets. The experimental results show that our method outperforms state-of-the-art fine-tuning methods and can be plugged into the state-of-the-art QA models to improve performance.

More importantly, our method is lightweight and flexible since it requires no learnable parameter except PLMs and has fewer hyper-parameters to tune. It is worth noting that our method readily controls the strength of knowledge preservation by a single hyper-parameter, enabling a good balance between preserving pre-trained knowledge and absorbing new knowledge from downstream tasks. In summary, our contributions are three-fold:

- We present a causal graph for fine-tuning with less forgetting by identifying the root cause of catastrophic forgetting as the missing causal effects of pre-trained data.
- Based on the proposed causal graph, we design a lightweight and flexible fine-tuning method called ***C**ausal **E**ffect **T**uning* for preserving knowledge in PLMs.
- For commonsense QA, we estimate the causal effect with a heuristic approximation. And we verify the effectiveness and versatility of our method through extensive experiments on six commonsense QA datasets.

2 Related Work

2.1 Fine-tuning Methods

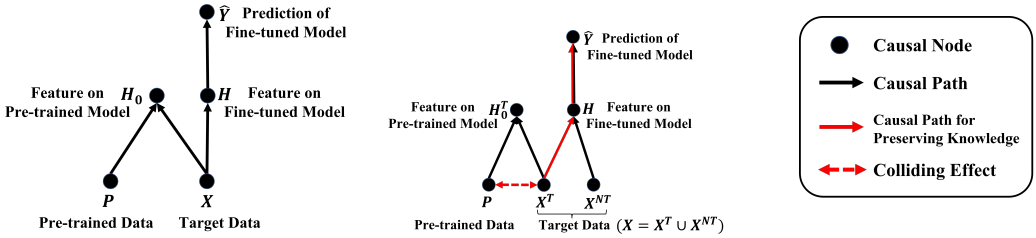
Apart from the methods mentioned above, some approaches improve downstream performances from the perspective of robustness. Aghajanyan et al. (2020) proposed *R3F*, which regularizes the symmetric KL divergence between the classifications of the original samples and the perturbed ones. Wu et al. (2022) proposed *Noisytune*, which adds uniform distribution noise to pre-trained parameters before fine-tuning to reduce the risk of overfitting the pre-training tasks and data. Besides, Mosbach et al. (2020); Zhang et al. (2020b) increased the stability of fine-tuning BERT (Devlin et al., 2019) in the low-data regime. Mosbach et al. (2020) advocated fine-tuning for a long time and choosing good optimizers and hyper-parameters. Zhang et al. (2020b) verified that re-initialized the top layers of BERT helps pre-trained knowledge transfer to downstream tasks.

2.2 Causal Inference

Causal inference (Glymour et al., 2016; Schölkopf, 2022) has been recently introduced to various computer vision tasks such as image classification (Hu et al., 2021), semantic segmentation (Zhang et al., 2020a) and long-tailed classification (Tang et al., 2020; Nan et al., 2021), and NLP tasks such as distantly supervised NER (Zhang et al., 2021), neural dialogue generation (Zhu et al., 2020) and continual named entity recognition (Zheng et al., 2022). To our best knowledge, we are the first to apply causal inference to fine-tuning.

2.3 Continual Learning

Although catastrophic forgetting happens in both continual learning (Rebuffi et al., 2017; Hu et al., 2021) and fine-tuning, the targets of these two tasks are fundamentally different. Continual learning aims to learn a growing number of tasks sequentially and maximize the performance on all recognized tasks. In contrast, fine-tuning maximize only the performance of target tasks. The recent advance in continual learning (Hu et al., 2021; Zheng et al., 2022) partially inspires this work.



(a) Vanilla Fine-Tuning (b) Fine-Tuning with Less Forgetting

Figure 1: The causal graphs of vanilla fine-tuning and our method. (a): The knowledge is forgotten during vanilla fine-tuning since the causal effect of the pre-trained data is missing; (b): When conditioning on H_0^T, the causal effect of the pre-trained data is retained through the causal path $P \leftrightarrow X^T \rightarrow H \rightarrow \hat{Y}$. In addition, the model absorbs new knowledge from X^{NT} through the causal path $X^{NT} \rightarrow H \rightarrow \hat{Y}$.

3 Methodology

In this section, we first use causal graphs (Pearl, 2009) to analyze how pre-trained knowledge is forgotten in fine-tuning. Then, we present a causal graph for anti-forgetting based on previous analysis. Next, we estimate the causal effect through derivations and propose a unified learning objective for fine-tuning with less forgetting. At last, we provide a heuristic approximation for estimating the causal effect on a challenging downstream task, commonsense QA. Note that the proposed causal graph and the fine-tuning method are generic to all downstream tasks.

3.1 Vanilla Fine-Tuning

In a causal graph, nodes represent variables, and directed edges are causalities between nodes. Fig.(1a) delineates the process of vanilla fine-tuning. We denote the pre-trained data (i.e., pre-trained knowledge) as P; the data in target tasks as X; the feature of X extracted by the pre-trained model and fine-tuned model as H_0 and H, respectively; the prediction of the fine-tuned model on target tasks as $\hat{Y}$ (i.e., the probability over categories). The causality between nodes (i.e., directed edges) is as follows: (1) $X \rightarrow H \rightarrow \hat{Y}$: $X \rightarrow H$ represents that the feature H is extracted by the backbone model such as RoBERTa, and $H \rightarrow \hat{Y}$ represents a classifier compute the prediction $\hat{Y}$ according to the extracted feature H; (2) $X \rightarrow H_0 \leftarrow P$: H_0 is determined by both P and X because H_0 is extracted by the pre-trained model, which is trained on P [2].

[2]Here, we ignore the effect of initial parameters initialized from the pre-trained model since it will be exponentially decayed towards zero during fine-tuning (Kirkpatrick et al., 2017).

Then, the effect of pre-trained data P on predictions $\hat{Y}$ can be calculated as:

$$
\begin{aligned}
\textit{Effect}_P &= \mathbb{P}(\hat{Y} = \hat{y}|do(P = p)) \\
&\quad - \mathbb{P}(\hat{Y} = \hat{y}|do(P = 0)) && (1) \\
&= \mathbb{P}(\hat{Y} = \hat{y}|P = p) - \mathbb{P}(\hat{Y} = \hat{y}|P = 0) && (2) \\
&= \mathbb{P}(\hat{Y} = \hat{y}) - \mathbb{P}(\hat{Y} = \hat{y}) && (3) \\
&= 0, && (4)
\end{aligned}
$$

In Eq.(1), $do(P = 0)$ represents that no pre-trained data is used for pre-training, and $do(P = p)$ represents a standard pre-training is performed. Then, $\mathbb{P}(\hat{Y} = \hat{y}|do(P = p))$ is the prediction given by a **pre-trained-then-fine-tuned** model and $\mathbb{P}(\hat{Y} = \hat{y}|do(P = 0))$ is the prediction given by a **randomly-initialized-then-fine-tuned** model. Eq.(1) defines $\textit{Effect}_P$ as the difference between the two predictions. Eq.(2) holds because P has no parent nodes. Eq.(3) holds because collider H_0 blocks all causal paths from P to Y.

Eq.(1)-(4) shows that a vanilla fine-tuned model will eventually forget all pre-trained knowledge when no constraints are imposed. In practice, fine-tuned models will not forget all learned knowledge because the learning rate and training time are considerably lower and shorter than those in pre-training. However, fine-tuned models likely forget partial pre-trained knowledge, overfit the target data, and fall into sub-optimal states since the amount of target data is usually considerably less than that of pre-trained data.

3.2 Fine-Tuning with Less Forgetting

The causal graph in Fig.(1a) necessitates the retrieval of the causality between P and $\hat{Y}$ back. A straightforward solution is utilizing the pre-trained

data to constrain model behaviors in new tasks. However, it is often obstructed by time, space, and financial constraints.

Thanks to causal inference, we can build a causal path between P and X without storing P. In the causal graph Fig.(1a), H_0 is the joint outcome of the independent causes P and X. Intriguingly, once the common effect H_0 is observed, the causes P and X become dependent. The causal effect is called **colliding effect** in Hu et al. (2021); Zheng et al. (2022)[3]. We'd like to provide a vivid example (Pearl, 2009) for understanding this pattern in causal inference: If the admission criteria to a certain school require either high grades or special musical talents, then these two attributes will be found to be correlated (negatively) in that school's student population, even if these attributes are uncorrelated in the population at large. By conditioning on H_0, the causal effect of pre-trained data is preserved during fine-tuning (i.e., $\textit{Effect}_P > 0$), and thus the pre-trained knowledge is preserved.

Except for preserving old knowledge, assimilating new knowledge from target data is critical. In addition, negative transfer may occur if we preserve pre-trained knowledge overly. Motivated by this, we split the target data into two nodes X^T and X^{NT}. X^T represents the samples where we calculate colliding effects, and their knowledge should be transferred from PLMs. X^{NT} is the samples where we do not calculate colliding effects, and their knowledge is domain-specific and should be absorbed into fine-tuned models. Consequently, the causal graph for our method is in Fig.(1b), and the rationale is as follows: The fine-tuned model preserves pre-trained knowledge by utilizing colliding effects ($P \leftrightarrow X^T$) while learning domain-specific knowledge (X^{NT}). The final prediction depends on both **pre-trained knowledge** and **domain-specific knowledge** from causal paths $P \leftrightarrow X^T \rightarrow H \rightarrow \hat{Y}$ and $X^{NT} \rightarrow H \rightarrow \hat{Y}$, respectively.

3.3 Estimating Colliding Effects

Next, we need to estimate the colliding effect between P and X^T. When conditioning on H_0, $\textit{Effect}_P$ can be calculated as:

$$\textit{Effect}_P = \sum_{i=1}^{N} \textit{Effect}_P^{(i)} \tag{5}$$

$$\approx \sum_{i=1}^{N}\sum_{k=0}^{K} \mathbb{P}(\hat{Y}^{(i)}|X = x^{(i,k)}) W_P(x^{(i)}, x^{(i,k)}), \tag{6}$$

where $\sum_{k=0}^{K} W_P(x^{(i)}, x^{(i,k)}) = 1$. N is the number of samples in the target data and $x^{(i)}$ is the i-th sample. $\textit{Effect}_P^{(i)}$ is the colliding effect of P on the prediction $\hat{Y}^{(i)}$. $W_P(\cdot,\cdot)$ is a function determined by the pre-trained model and measures the similarity between two samples in the hidden space of the pre-trained model. In this case, we denote $W_P(x^{(i)}, x^{(i,k)})$ as $W_{i,k}$ for brevity. $x^{(i,k)}$ is the k-th nearest neighbor of $x^{(i)}$ in the hidden space. Since $x^{(i)}$ always has the largest similarity with itself, we let $x^{(i,0)} = x^{(i)}$ and call $x^{(i)}$ the anchor sample. Besides, we assume that the K Nearest Neighbours (KNNs) are sorted in descending order according to the similarity. Therefore, we have $W_{i,0} \geq W_{i,1} \geq W_{i,2} \geq \cdots \geq W_{i,K}$. K is a hyper-parameter representing the number of neighbors for estimating $\hat{Y}^{(i)}$. We provide a detailed derivation and further explanation in Appendix A.

Eq.(5) re-writes the total causal effect as the sum of the causal effect on the prediction of each target sample (i.e.,$\textit{Effect}_P^{(i)}$). In Eq.(6), $\mathbb{P}(\hat{Y}^{(i)}|X = x^{(i,k)})$ represents the likelihood of $\hat{Y}^{(i)}$ when $x^{(i,k)}$ is the model input. Eq.(6) shows that $\textit{Effect}_P^{(i)}$ can be approximated by the weighted sum of the likelihood when the model input is the anchor sample $x^{(i)}$ and its KNNs. Since we expect to maximize $\mathbb{P}(\hat{Y}^{(i)} = y^{(i)}|X = x^{(i)})$, maximizing $\textit{Effect}_P^{(i)}$ equals to maximizing the likelihood of the **joint prediction** on the ground-truth label $y^{(i)}$.

3.4 Overall Objective

In Eq. 6, the total causal effect $\textit{Effect}_P$ is broken down into the causal effect of each sample $\textit{Effect}_P^{(i)}$. In this case, maximizing $\textit{Effect}_P$ is to preserve the related knowledge of all samples. As we mentioned before, indiscriminately preserving knowledge may lead to negative transfer. To address this problem, we introduce a similarity threshold θ to select the number of nearest neighbors for each sample automatically. Specifically, for the i-th sample, we truncate the k_i ($K \geq k_i \geq 0$) nearest neighbors whose similarity is greater or equal

[3]This phenomenon is also known as *Berkson's paradox* in (Berkson, 1946) and as the *explaining away effect* in (Peari and Kim, 1983).

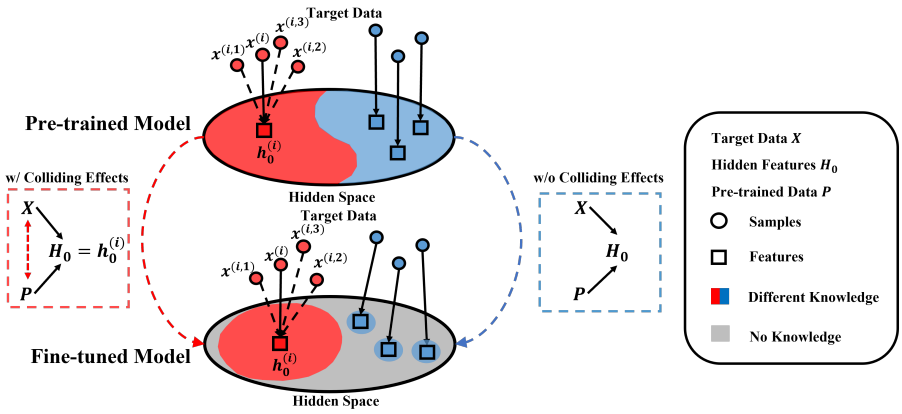


Figure 2: An illustration of Causal Effect Tuning. $x^{(i)}$ is the anchor sample and $h_0^{(i)}$ is the hidden feature extracted by the pre-trained model. $x^{(i,1)}, x^{(i,2)}, x^{(i,3)}$ are the KNNs of $x^{(i)}$. We apply colliding effects on $x^{(i)}$ to preserve the old knowledge. After fine-tuning, the "Red" knowledge is preserved with colliding effects, and "blue" knowledge is forgotten without colliding effects. A specific instance is as follows: $x^{(i)}$= "What is a fast but expensive way to send small cargo? (answer: airplane)"; $x^{(i,1)}$="Where could you find a seat that sometimes vibrates?"(answer: airplane); $x^{(i,2)}$="What has metal wings?"(answer: airplane); $x^{(i,3)}$= "It was important precious cargo, so it was delivered as quickly as possible by means of what?"(answer: aeroplane). The "red" knowledge represents the commonsense about "**airplane**".

than θ. In this way, we differentiate the strength of knowledge preservation on each sample by selecting the neighbors with small distances to their anchor sample. More interestingly, when $k_i = 0$, *i.e.*, a sample has no neighbors, the $\mathit{Effect}_P^{(i)}$ amounts to $\mathbb{P}(\hat{Y}^{(i)} = y^{(i)}|X = x^{(i)})$, which is exactly the objective of each sample in vanilla fine-tuning. Fig. 2 provides an illustration for our method, where the samples with no neighbors can be seen as a special case of our method. Formally, we define the overall objective as follows:

$$\max \quad \mathit{Effect}_P = \sum_{i=1}^{N} \mathit{Effect}_P^{(i)} \tag{7}$$

$$= \underbrace{\sum_{i \in \mathcal{S}^T} \mathit{Effect}_P^{(i)}}_{\text{Colliding Effects}} + \underbrace{\sum_{i \in \mathcal{S}^{NT}} \mathit{Effect}_P^{(i)}}_{\text{Vanilla Fine-Tuning}}, \tag{8}$$

$$= \underbrace{\sum_{i \in \mathcal{S}^T} \sum_{k=0}^{k_i} \mathbb{P}(\hat{Y}^{(i)}|X = x^{(i,k)}) W_{i,k}}_{\text{Colliding Effects}} + \underbrace{\sum_{i \in \mathcal{S}^{NT}} \mathbb{P}(\hat{Y}^{(i)}|X = x^{(i)})}_{\text{Vanilla Fine-Tuning}}, \tag{9}$$

where $\sum_k W_{i,k} = 1, \mathcal{S}^T = \{i|k_i > 0\}, \mathcal{S}^{NT} = \{i|k_i = 0\}$. Considering the distances between KNNs and their anchor sample are approximated and thus inaccurate, we set $W_{i,0} = W_0$ and $W_{i,1} = W_{i,2} = \cdots = W_{i,k_i} = \frac{1-W_0}{k_i}$ when $k_i > 0$ for implementation. W_0 is a hyper-parameter for controlling the strength of colliding effects. When $W_0 = 0$, the overall target degenerates to the vanilla fine-tuning target. When $W_0 = 1$, the overall target retains knowledge indiscriminately on all samples. In Eq.(9), the second term amounts to the vanilla fine-tuning objective since only the anchor sample's prediction is computed. In other words, we preserve knowledge for the samples with KNNs and learn new knowledge for the samples without KNNs. The rationale is that the knowledge should be preserved when more samples require it to answer the question. In the proposed causal graph in Fig.(1b), the first and the second term of Eq.(9) correspond to the two causal paths through X^T and X^{NT} respectively. We summarized the proposed method in Fig. 2 and Alg. 1 in Appendix A.

3.5 An Implementation on Commonsense QA

In this subsection, we provide an implementation for the causal graph in Fig.(1b) on commonsense QA. We note that the overall objective in Eq. 9 is agnostic to specific downstream tasks and model architectures. The implementation can be different in various tasks or model architectures, and the key is to find proper KNNs. This paper provides an implementation on commonsense QA since PLMs may be endowed with commonsense knowledge in pre-training (Petroni et al., 2019; AlKhamissi et al., 2022), and it is still challenging for models to

capitalize on commonsense (Talmor et al., 2018).

We first formulate the commonsense QA as follows: Given a dataset with N samples $\{(q^{(i)}, a^{(i)}, \{o_j^{(i)}\}_j)\}_i^N$, we train the best model for choosing the gold answer $a^{(i)}$ among options $\{o_j^{(i)}\}$ given a question $q^{(i)}$. More specifically, the input of the i-th sample can be $x^{(i)} = q^{(i)}||o_1^{(i)}||\cdots||o_j^{(i)}$ or $\{x^{(i)}\}_j = \{q^{(i)}||o_j^{(i)}\}_j$ [4] where $||$ is the string-level concatenation.

Then, we define a metric to search KNNs. A simple solution is to compute the euclidean distance or cosine similarity between the average last hidden states of PLMs. However, this method struggles to capture accurate semantic meanings, and measuring sentence similarity remains challenging. In this regard, we provide a simple heuristic approximation. In most cases, the questions with the same gold answers share the same knowledge. For example, "airplane" is the gold answer to the following questions, and we can use the knowledge about "airplane" to answer them: "*What is a fast but expensive way to send small cargo?*"; "*Where could you find a seat that sometimes vibrates?*"; "*What has metal wings?*". Therefore, we estimate the similarity between gold answers to cope with the difficulty of evaluating sentence similarity. Since options are usually much shorter than questions, lightweight tools such as spaCy (Honnibal et al., 2020) can be used to retrieve gold answers with close semantic meanings (e.g., "airplane" and "aeroplane").

At last, we define the input of the i-th sample's KNNs as $x^{(i,k)} = q^{(i,k)}||o_1^{(i)}||\cdots||o_j^{(i)}$ or $\{x^{(i,k)}\}_j = \{q^{(i,k)}||o_j^{(i)}\}_j$. It alleviates the overfitting problem since the model needs to select the correct answer among the options of anchor sample when the question is from its KNNs.

4 Experiments

4.1 Settings

Datasets. We conduct experiments on 6 datasets: CommonsenseQA(CSQA) (Talmor et al., 2018), OpenBookQA(OBQA) (Mihaylov et al., 2018), ARC (Clark et al., 2018, 2016), QASC (Khot et al., 2020), SocialIQA (SIQA) (Sap et al., 2019), PIQA (Bisk et al., 2020). Since the official test sets of CSQA, QASC, SIQA, and PIQA are not available, we follow (Yasunaga et al., 2021) and use the official dev sets as test sets and split in-house dev set from the original training sets. The dataset statistics are summarized in Table 6 in Appendix B.

Training. Given its popularity, we use RoBERTa-large (Liu et al., 2019b) as the backbone model in default. We also explore T5-large (Raffel et al., 2020) since Khashabi et al. (2020) showed that it excels at answering questions in different formats. Other training details are specified in Appendix B.

Competitive Methods. We make comparisons with nine state-of-the-art fine-tuning methods: vanilla fine-tuning, BSS (Chen et al., 2019), ChildTune-F&ChildTune-D (Xu et al., 2021a), Mixout (Lee et al., 2019), NoisyTune (Wu et al., 2022), R3F (Aghajanyan et al., 2020), RecAdam (Chen et al., 2020) and ReInit (Zhang et al., 2020b). For each method, we use the recommended hyper-parameters in the paper and source code for a fair comparison. We discuss the implementation details of the fine-tuning methods in Appendix C.

Hyper-Parameters. As for the hyperparameters of our methods, we fix $K = 5$ and search the best W_0 in {0.5, 0.7, 0.9, 0.95, 0.97} for each dataset. We use spaCy to estimate the similarity between gold answers. We set $\theta = 0.99$ for PIQA and $\theta = 1.00$ for other datasets (i.e., the gold answers should be matched precisely).

4.2 Results and Analyses

Comparisons with State-Of-The-Art. To demonstrate the effectiveness of our method, we re-implement several strong baselines on commonsense QA datasets using their officially released codes and hyper-parameters. The results are summarized in Table 1. Results show that our method outperforms all fine-tuning methods consistently. On QASC and OBQA, our method achieves 57.57% and 70.76% accuracy, obtaining 3.53% and 2.64% improvements on vanilla fine-tuning.

Why our method better preserves commonsense knowledge from PLMs? The reasons are two-fold. The first reason is that our method utilizes the colliding effect for transferring the "colliding" commonsense knowledge, while other methods do not. For instance, in Fig.2, our method encourages models to update $x^{(i)}$ and its KNNs $x^{(i,1)}, x^{(i,2)}, x^{(i,3)}$ simultaneously. In this way, the commonsense knowledge about "airplane" that "airplanes deliver small and precious cargo", "airplanes have metal wings" and "airplanes have seats" can be trans-

[4] Concatenating all options or each option depends on models.

Table 1: Comparison with state-of-the-art methods. The average accuracy (%) and the standard derivation are reported.

Methods	CSQA	OBQA	ARC-Easy	ARC-Challenge	QASC	PIQA	SIQA
Fine-Tuning	75.74 (0.47)	68.12 (0.32)	67.66 (0.45)	45.98 (0.53)	54.04 (1.05)	78.62 (0.53)	77.46 (0.33)
BSS	76.21 (0.63)	68.64 (1.23)	68.24 (0.31)	46.62 (0.80)	53.82 (1.20)	78.20 (0.96)	77.35 (0.18)
ChildTune-F	75.50 (0.44)	69.84 (0.88)	68.17 (0.77)	46.30 (1.67)	54.41 (1.63)	77.61 (1.06)	75.87 (0.64)
ChildTune-D	76.76 (0.81)	69.36 (0.60)	67.86 (0.73)	45.28 (0.67)	55.77 (0.52)	78.32 (0.38)	78.20 (0.35)
Mixout	76.09 (0.56)	69.70 (0.71)	67.85 (0.57)	44.87 (0.72)	57.34 (1.02)	79.22 (0.31)	77.89 (0.37)
NoisyTune	76.01 (0.61)	67.56 (0.52)	67.61 (0.58)	46.05 (0.65)	54.43 (0.60)	78.61 (0.31)	76.59 (0.36)
R3F	76.59 (0.48)	68.47 (0.26)	68.13 (0.68)	47.01 (0.58)	55.69 (0.78)	79.38 (0.60)	77.05 (0.44)
RecAdam	75.43 (0.33)	70.68 (0.89)	68.07 (0.69)	45.90 (0.59)	54.62 (1.22)	78.26 (1.25)	76.71 (0.61)
ReInit	75.51 (0.71)	69.92 (1.14)	67.63 (0.59)	46.68 (0.39)	52.12 (1.66)	78.61 (0.37)	77.79 (0.15)
CET(Ours)	**76.82 (0.33)**	**70.76 (0.33)**	**68.53 (0.53)**	**47.52 (0.38)**	**57.57 (0.44)**	**79.43 (0.27)**	**78.76 (0.31)**

Table 2: Comparisons with knowledge-graph-based methods on CSQA with different proportions of training data. We use the train-dev-test split in Jiang et al. (2022) and thus the CSQA results are inconsistent with those in other experiments. The results of RoBERTa-large, RGCN, KagNet, Relation Network, MHGRN, QAGNN, and SAFE are reported in Jiang et al. (2022). We report the average accuracy (%).

Methods	use GNN?	use KG?	Proportion of Training Data					
			5%	10%	20%	50%	80%	100%
RoBERTa-large	✗	✗	29.66	42.84	58.47	66.13	68.47	68.69
+RGCN (Schlichtkrull et al., 2018)	✓	✓	24.41	43.75	59.44	66.07	68.33	68.41
+KagNet (Lin et al., 2019)	✓	✓	21.92	49.83	60.09	66.93	69.14	68.59
+Relation Network (Santoro et al., 2017)	✓	✓	23.77	34.09	59.90	65.62	67.37	69.08
+MHGRN (Feng et al., 2020)	✓	✓	29.01	32.02	50.23	68.09	70.83	71.11
+QAGNN (Yasunaga et al., 2021)	✓	✓	32.95	37.77	50.15	69.33	70.99	73.41
+SAFE (Jiang et al., 2022)	✓	✓	36.45	56.51	65.16	70.72	73.22	74.03
+CET(Ours)	✗	✗	56.24	59.55	65.19	67.93	70.02	70.99
+CET+QAGNN	✓	✓	58.78	60.35	65.59	70.43	72.04	73.81
+CET+SAFE	✓	✓	**59.39**	**61.02**	**65.75**	**70.79**	**73.31**	**74.54**

Table 3: An CSQA example and its KNNs in our method.

	Gold Answer	Question
Anchor	pet shops	Too many people want exotic snakes. The demand is driving what to carry them?
KNNs	pet shops	Where can a person buy a snake?
	pet shop	Where might a blowfish be kept?
	pet shop	Where can you take home a hermit crab?
	pet store	Where would you get a dog if you do not have one?
	pet store	John loves animals and he hates animal abuse. Because of this, john is very careful about the places he goes. Where might he avoid going?

ferred jointly, which reduces the risk of over-fitting. We provide more examples from each dataset in Table 3 and Table 10,11, in Appendix F. The second reason is that our method does not directly constrain (e.g., ChildTune-D, Mixout, RecAdam) or modify (e.g., NoisyTune, ReInit) the parameters of fine-tuned models. Empirical results show that these methods encounter negative transfers on some of the datasets. Instead, our method builds upon the causal inference theory and utilizes the joint prediction as a soft constraint to transfer related knowledge while mitigating negative transfer.

Compared with Knowledge-Graph-Based Methods. Utilizing knowledge graphs such as ConceptNet (Speer et al., 2017) is a common practice for building commonsense QA systems. We compared our method with six knowledge-graph-based methods: Relation Network (Santoro et al., 2017), KagNet (Lin et al., 2019), RGCN(Schlichtkrull et al., 2018), MHGRN(Feng et al., 2020), QAGNN(Yasunaga et al., 2021), SAFE(Jiang et al., 2022). Detailed descriptions and other related works are given in Appendix D. Note that these methods utilize knowledge graphs (KGs) as external knowledge resources, and most of them train graph neural networks (GNNs) for extracting features from KGs. In contrast, our method does not introduce any additional learnable parameters except PLMs and the final fully-connected layer. The result in Table 2 shows that our method out-

performs RGCN, KagNet, and Relation Network by only mining the internal knowledge of PLMs. Furthermore, our method significantly outperforms all the knowledge-graph-based methods under low resource conditions ($\leq 20\%$ training data is used), which shows that our method helps PLMs adapt to downstream tasks with less data.

In addition, our method can be easily implemented as a plug-in module by simply substituting the vanilla fine-tuning objective for the causal effect in Eq.(9). We combine our method with QAGNN and SAFE, respectively. Table 2 shows that our approach consistently improves QAGNN and SAFE and achieves superior performances. Therefore, the pre-trained commonsense knowledge benefits downstream tasks even when KGs are introduced.

Fine-tuning on a Cyclic Chain of Tasks. To understand how our method preserves knowledge during fine-tuning, we follow Aghajanyan et al. (2020) and design a cyclic chain of tasks:

$$\underbrace{A \rightarrow B \rightarrow C}_{Cycle1} \rightarrow \underbrace{A \rightarrow B \rightarrow C}_{Cycle2} \rightarrow \cdots$$

In our experiment, we set A=CSQA, B=OBQA, and C=QASC for a demonstration. Specifically, we start from a PLM and fine-tune it on CSQA. Then, we use the model fine-tuned on CSQA to initialize the backbone model's parameters and continue fine-tuning it on OBQA. Table 4 shows that our method retains knowledge significantly better than vanilla fine-tuning. The performances on OBQA and QASC improve at every cycle, suggesting that our method effectively retains knowledge from the previous datasets. Unfortunately, both performances of vanilla fine-tuning and our method on CSQA degrade slightly, showing that negative transfer happens. In this case, vanilla fine-tuning will lead to more serious performance degradation. The experiment is for demonstration, and a better combination of tasks that promote each other may be found.

Ablation Study. To verify the effectiveness of our method, we consider the following ablated version of our method: (1) replacing the KNNs (*Large*,Ours) with randomly selected samples (*Rand*) or samples with the smallest similarity (*Small*); (2) searching the KNNs according to the similarity of average last hidden states (*Avg*) instead of gold answers (*Gold*, Ours). The result in Table 5 shows that the model learns commonsense knowledge better when the KNNs share the gold answer with close meaning.

Table 4: The results of cyclical sequential fine-tuning for three cycles. The average accuracy (%) is reported.

	Dataset	Fine-Tuning	CET(Ours)
Cycle1	CSQA	75.74	**76.82**
	OBQA	68.80	**70.89**
	QASC	54.31	**57.49**
Cycle 2	CSQA	75.52	**76.69**
	OBQA	69.95	**71.18**
	QASC	55.06	**57.64**
Cycle 3	CSQA	75.44	**76.75**
	OBQA	70.28	**71.45**
	QASC	55.12	**57.78**

Table 5: The ablation study of our method. *Gold*/*Avg*: searching the KNNs according to the similarity of gold answers or the average last hidden states. *Large*/*Small*/*Rand*: searching the KNNs with the largest or smallest similarity, or randomly. The average accuracy (%) is reported.

Methods	CSQA	OBQA	QASC
Gold+Large(Ours)	**76.82**	**70.76**	**57.57**
Gold+Rand	74.61	68.53	55.77
Gold+Small	74.04	64.67	53.13
Avg+Large	76.17	69.64	55.62
Avg+Rand	74.12	68.54	54.54
Avg+Small	74.20	68.07	53.46
Fine-Tuning	75.74	68.12	54.04

Additional Experiments. Due to space constraints, we present the experiments on T5, the hyper-parameter analysis, the experiments on Named Entity Recognition, and further discussions in Appendix E.

5 Conclusion

We propose a novel fine-tuning technique rooted in causal inference for preserving pre-trained knowledge from PLMs. Although many fine-tuning methods have been proposed in recent years, most of them overlooked one or both hidden issues of fine-tuning, catastrophic forgetting and negative transfer, which result in a dilemma. In this paper, we provide an answer to the dilemma from the casual lens. Impressively, we empirically find that the proposed method achieves the best performance on six commonsense QA datasets and is flexible to be applied to various QA systems and model architectures.

Limitations

There are three limitations on our method. First, we did not verify our method on more generic tasks, such as text classification, yet it is not limited to commonsense QA. Extending our method to other downstream tasks is our future work. Second, our method requires a longer training time and a larger GPU memory since the KNNs require forward and backward propagation additionally. Third, we do not consider the ambiguity of gold answers, which may affect the quality of KNNs. For example, "apple" may refer to a kind of fruit or a technology company.

Acknowledgements

The work described in this paper was partially funded by the National Natural Science Foundation of China (Grant Nos. 62272173, 61872148), the Natural Science Foundation of Guangdong Province (Grant Nos. 2022A1515010179, 2019A1515010768).

References

Armen Aghajanyan, Akshat Shrivastava, Anchit Gupta, Naman Goyal, Luke Zettlemoyer, and Sonal Gupta. 2020. Better fine-tuning by reducing representational collapse. In *International Conference on Learning Representations*.

Badr AlKhamissi, Millicent Li, Asli Celikyilmaz, Mona Diab, and Marjan Ghazvininejad. 2022. A review on language models as knowledge bases. *arXiv preprint arXiv:2204.06031*.

Joseph Berkson. 1946. Limitations of the application of fourfold table analysis to hospital data. *Biometrics Bulletin*, 2(3):47–53.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Sanyuan Chen, Yutai Hou, Yiming Cui, Wanxiang Che, Ting Liu, and Xiangzhan Yu. 2020. 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)*, pages 7870–7881, Online. Association for Computational Linguistics.

Xinyang Chen, Sinan Wang, Bo Fu, Mingsheng Long, and Jianmin Wang. 2019. Catastrophic forgetting meets negative transfer: Batch spectral shrinkage for safe transfer learning. *Advances in Neural Information Processing Systems*, 32.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Peter Clark, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Turney, and Daniel Khashabi. 2016. Combining retrieval, statistics, and inference to answer elementary science questions. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Yanlin Feng, Xinyue Chen, Bill Yuchen Lin, Peifeng Wang, Jun Yan, and Xiang Ren. 2020. Scalable multi-hop relational reasoning for knowledge-aware question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1295–1309, Online. Association for Computational Linguistics.

Madelyn Glymour, Judea Pearl, and Nicholas P Jewell. 2016. *Causal inference in statistics: A primer*. John Wiley & Sons.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python.

Eduard Hovy, Mitch Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. Ontonotes: the 90% solution. In *Proceedings of the human language technology conference of the NAACL, Companion Volume: Short Papers*, pages 57–60.

Xinting Hu, Kaihua Tang, Chunyan Miao, Xian-Sheng Hua, and Hanwang Zhang. 2021. Distilling causal effect of data in class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3957–3966.

Jinhao Jiang, Kun Zhou, Wayne Xin Zhao, and Ji-Rong Wen. 2022. Great truths are always simple: A rather simple knowledge encoder for enhancing the commonsense reasoning capacity of pre-trained models. *arXiv preprint arXiv:2205.01841*.

Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. 2020. UNIFIEDQA: Crossing format boundaries with a single QA system. In *Findings of the Association for Computational Linguistics:*

EMNLP 2020, pages 1896–1907, Online. Association for Computational Linguistics.

Tushar Khot, Peter Clark, Michal Guerquin, Peter Jansen, and Ashish Sabharwal. 2020. Qasc: A dataset for question answering via sentence composition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8082–8090.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526.

Cheolhyoung Lee, Kyunghyun Cho, and Wanmo Kang. 2019. Mixout: Effective regularization to finetune large-scale pretrained language models. In *International Conference on Learning Representations*.

Bill Yuchen Lin, Xinyue Chen, Jamin Chen, and Xiang Ren. 2019. KagNet: Knowledge-aware graph networks for commonsense reasoning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2829–2839, Hong Kong, China. Association for Computational Linguistics.

Jiacheng Liu, Skyler Hallinan, Ximing Lu, Pengfei He, Sean Welleck, Hannaneh Hajishirzi, and Yejin Choi. 2022a. Rainier: Reinforced knowledge introspector for commonsense question answering. *arXiv preprint arXiv:2210.03078*.

Jiacheng Liu, Alisa Liu, Ximing Lu, Sean Welleck, Peter West, Ronan Le Bras, Yejin Choi, and Hannaneh Hajishirzi. 2022b. Generated knowledge prompting for commonsense reasoning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3154–3169, Dublin, Ireland. Association for Computational Linguistics.

Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. 2019a. On the variance of the adaptive learning rate and beyond. In *International Conference on Learning Representations*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391, Brussels, Belgium. Association for Computational Linguistics.

Marius Mosbach, Maksym Andriushchenko, and Dietrich Klakow. 2020. On the stability of fine-tuning bert: Misconceptions, explanations, and strong baselines. In *International Conference on Learning Representations*.

Shawn N Murphy, Griffin Weber, Michael Mendis, Vivian Gainer, Henry C Chueh, Susanne Churchill, and Isaac Kohane. 2010. Serving the enterprise and beyond with informatics for integrating biology and the bedside (i2b2). *Journal of the American Medical Informatics Association*, 17(2):124–130.

Guoshun Nan, Jiaqi Zeng, Rui Qiao, Zhijiang Guo, and Wei Lu. 2021. Uncovering main causalities for long-tailed information extraction. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9683–9695.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

J Peari and J Kim. 1983. A computationai modei for combined causal and diagnostic reasoning in inference systems. In *Proceeding of the 8th International Joint Conference on Artificial Intelligence*.

Judea Pearl. 2009. *Causality*. Cambridge university press.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67.

Nazneen Fatema Rajani, Bryan McCann, Caiming Xiong, and Richard Socher. 2019. Explain yourself! leveraging language models for commonsense reasoning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4932–4942, Florence, Italy. Association for Computational Linguistics.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. 2017. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010.

Erik Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. 2017. A simple neural network module for relational reasoning. *Advances in neural information processing systems*, 30.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. 2019. Social IQa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4463–4473, Hong Kong, China. Association for Computational Linguistics.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European semantic web conference*, pages 593–607. Springer.

Bernhard Schölkopf. 2022. Causality for machine learning. In *Probabilistic and Causal Inference: The Works of Judea Pearl*, pages 765–804.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Thirty-first AAAI conference on artificial intelligence*.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2018. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *arXiv preprint arXiv:1811.00937*.

Kaihua Tang, Jianqiang Huang, and Hanwang Zhang. 2020. Long-tailed classification by keeping the good and removing the bad momentum causal effect. *Advances in Neural Information Processing Systems*, 33:1513–1524.

Lisa Torrey and Jude Shavlik. 2010. Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pages 242–264. IGI global.

Peifeng Wang, Nanyun Peng, Filip Ilievski, Pedro Szekely, and Xiang Ren. 2020a. Connecting the dots: A knowledgeable path generator for commonsense question answering. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4129–4140, Online. Association for Computational Linguistics.

Tan Wang, Jianqiang Huang, Hanwang Zhang, and Qianru Sun. 2020b. Visual commonsense r-cnn. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10760–10770.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Chuhan Wu, Fangzhao Wu, Tao Qi, and Yongfeng Huang. 2022. NoisyTune: A little noise can help you finetune pretrained language models better. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 680–685, Dublin, Ireland. Association for Computational Linguistics.

Runxin Xu, Fuli Luo, Zhiyuan Zhang, Chuanqi Tan, Baobao Chang, Songfang Huang, and Fei Huang. 2021a. Raise a child in large language model: Towards effective and generalizable fine-tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9514–9528.

Yichong Xu, Chenguang Zhu, Ruochen Xu, Yang Liu, Michael Zeng, and Xuedong Huang. 2021b. Fusing context into knowledge graph for commonsense question answering. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1201–1207, Online. Association for Computational Linguistics.

Yiben Yang, Chaitanya Malaviya, Jared Fernandez, Swabha Swayamdipta, Ronan Le Bras, Ji-Ping Wang, Chandra Bhagavatula, Yejin Choi, and Doug Downey. 2020. Generative data augmentation for commonsense reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1008–1025, Online. Association for Computational Linguistics.

Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. 2021. QA-GNN: Reasoning with language models and knowledge graphs for question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 535–546, Online. Association for Computational Linguistics.

Dong Zhang, Hanwang Zhang, Jinhui Tang, Xian-Sheng Hua, and Qianru Sun. 2020a. Causal intervention for weakly-supervised semantic segmentation. *Advances in Neural Information Processing Systems*, 33:655–666.

Tianyi Zhang, Felix Wu, Arzoo Katiyar, Kilian Q Weinberger, and Yoav Artzi. 2020b. Revisiting few-sample bert fine-tuning. In *International Conference on Learning Representations*.

Wenkai Zhang, Hongyu Lin, Xianpei Han, and Le Sun. 2021. De-biasing distantly supervised named entity recognition via causal intervention. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International*

Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 4803–4813.

Junhao Zheng, Zhanxian Liang, Haibin Chen, and Qianli Ma. 2022. Distilling causal effect from miscellaneous other-class for continual named entity recognition. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3602–3615.

Qingfu Zhu, Weinan Zhang, Ting Liu, and William Yang Wang. 2020. Counterfactual off-policy training for neural dialogue generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3438–3448.

A A Detailed Derivation for the Colliding Effect

Algorithm 1: Causal Effect Tuning

Input: $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$: a training set with N samples; $\mathcal{F}_0$: a pre-trained model

Output: $\mathcal{F}$: a fine-tuned model

Initialize $\mathcal{F} \leftarrow \mathcal{F}_0$;

Compute the KNNs for each sample $x^{(i)}$: $x^{(i,1)}, \cdots, x^{(i,k_i)}$;

while *not converge* **do**

  Compute *Effect*$_P$ according to Eq.(9);

  $\mathcal{F} \leftarrow \arg\max_{\mathcal{F}}$ *Effect*$_P$;

end

return $\mathcal{F}$;

Without loss of generality, we first define the fine-tuning process formally as follows: Given a pre-trained model $\mathcal{F}_0$ and a dataset with N samples $\{(x^{(i)}, y^{(i)}\}_{i=1}^{N}$, we aim to learn a model $\mathcal{F}$ which has the best performance on predicting the label $y^{(i)}$. Recall that in Eq.(5), we re-write *Effect*$_P$ as the sum of the causal effect on each prediction $\hat{Y}^{(i)}$. Now, the outcome node $\hat{Y}$ in the causal graph becomes $\hat{Y}^{(i)}$. Then, we need to condition on H_0 to utilize colliding effects. Considering when predicting $\hat{Y}^{(i)}$, $x^{(i)}$ should play an important role. Furthermore, when $X = x^{(i)}$, its hidden feature is simply calculated as $h_0^{(i)} = \mathcal{F}_0(x^{(i)})$. Therefore, it is natural to choose $h_0^{(i)}$ as the hidden feature we condition on.

After controlling $H_0 = h_0^{(i)}$, the meaning of the input node X in the causal graph becomes all samples whose hidden feature is $h_0^{(i)}$. Unfortunately, due to the sparsity in high dimensional spaces, only $x^{(i)}$ satisfies this constraint. Intuitively, if we loosen this constraint a bit, the colliding effect will not disappear instantly. Instead, the colliding effect will vanish gradually when the hidden feature becomes farther and farther away from $h_0^{(i)}$. Put differently, colliding effects still exist when samples bear a resemblance to each other in the hidden space of the pre-trained model.

Now, we provide a derivation as follows:

$$
\begin{aligned}
&\textit{Effect}_P^{(i)} \\
&= \mathbb{P}(\hat{Y}^{(i)}|H_0 = h_0^{(i)}, P = p) - \mathbb{P}(\hat{Y}^{(i)}|H_0 = h_0^{(i)}, P = 0) && (10)\\
&= \sum_{k=1}^{N} (\mathbb{P}(\hat{Y}^{(i)}|X = x^{(k)}, H_0 = h_0^{(i)}) && (11)\\
&\qquad \mathbb{P}(X = x^{(k)}|H_0 = h_0^{(i)}, P = p) \\
&\qquad - \mathbb{P}(\hat{Y}^{(i)}|X = x^{(k)}, H_0 = h_0^{(i)}) \\
&\qquad \mathbb{P}(X = x^{(k)}|H_0 = h_0^{(i)}, P = 0)) \\
&= \sum_{k=1}^{N} \mathbb{P}(\hat{Y}^{(i)}|X = x^{(k)}) (\mathbb{P}(X = x^{(k)}|H_0 = h_0^{(i)}, P = p) && (12)\\
&\qquad - \mathbb{P}(X = x^{(k)}|H_0 = h_0^{(i)}, P = 0)) \\
&\approx \sum_{k=1}^{N} \mathbb{P}(\hat{Y}^{(i)}|X = x^{(k)}) \mathbb{P}(X = x^{(k)}|H_0 = h_0^{(i)}, P = p) && (13)\\
&= \sum_{k=1}^{N} \mathbb{P}(\hat{Y}^{(i)}|X = x^{(k)}) && (14)\\
&\frac{\mathbb{P}(H_0 = h_0^{(i)}|X = x^{(k)}, P = p)\mathbb{P}(X = x^{(k)}|P = p)}{\mathbb{P}(H_0 = h_0^{(i)}|P = p)} \\
&= \sum_{k=1}^{N} \mathbb{P}(\hat{Y}^{(i)}|X = x^{(k)}) W_P(x^{(i)}, x^{(k)}) && (15)\\
&\approx \sum_{k=0}^{K} \mathbb{P}(\hat{Y}^{(i)}|X = x^{(i,k)}) W_P(x^{(i)}, x^{(i,k)}) && (16)
\end{aligned}
$$

Eq.(10) is deduced from Eq.(2) and the condition of $H_0 = h_0^{(i)}$. Eq.(11) expands Eq.(10) as the sum of all N samples. In Eq.(12), $\mathbb{P}(\hat{Y}^{(i)}|X, H_0) = \mathbb{P}(\hat{Y}^{(i)}|X)$ because X is the only mediator (Pearl, 2009) from P to $\hat{Y}^{(i)}$. In Eq.(13), we approximate $\mathbb{P}(X = x^{(k)}|H_0 = h_0^{(i)}, P = 0)$ as zero because the likelihood of $X = x^{(k)}$ is small when the model is randomly initialized. Eq.(14) is obtained by applying Bayes formula to Eq.(13). In Eq.(14), $\mathbb{P}(H_0 = h_0^{(i)}|P = p)$ and $\mathbb{P}(X = x^{(k)}|P = p)$ are intractable and can be seen as constants. We note that the likelihood term $\mathbb{P}(H_0 = h_0^{(i)}|X = x^{(k)}, P = p)$ represents how likely the hidden feature is $h_0^{(i)}$ when the input sample is $x^{(k)}$. Obviously, the likelihood is the largest when $k = i$ and becomes smaller when the hidden feature of

$x^{(k)}$ become farther away from $h_0^{(i)}$. Therefore, the fractional term of Eq. 14 can be regarded as a **scaling factor** of the likelihood $\mathbb{P}(\hat{Y}^{(i)}|X = x^{(k)})$. In Eq.(15), we re-write the fractional term of Eq.(14) as a function of $x^{(i)}$ and $x^{(k)}$ since $h_0^{(i)} = \mathcal{F}_0(x^{(i)})$. In Eq.(15), we truncate the top K samples, which are closest to $x^{(i)}$, in the hidden space of the pre-trained model. Besides, we let $x^{(i,0)} = x^{(i)}$ since $x^{(i)}$ has the largest similarity with itself. Additionally, we let $\sum_{k=0}^{K} W_P(x^{(i)}, x^{(i,k)}) = 1$ to ensure that the joint prediction is a probability distribution over categories.

B Training Details

The dataset statistics is in Table 6. All models are implemented based on Pytorch (Paszke et al., 2019) and Huggingface (Wolf et al., 2019). We use the default hyper-parameters of RoBERTa and T5 according to the Huggingface implementation. Following Yasunaga et al. (2021); Khashabi et al. (2020), we concatenate all options as input when the backbone is T5 and concatenate each option respectively when the backbone is RoBERTa. We tuned the batch size in {64, 128}, the learning rate of the backbone model in {5e-5, 2e-5, 1e-5}. Before fine-tuning RoBERTa, a randomly initialized fully connected (FC) layer is added on top of RoBERTa, and the learning rate of the FC layer is 1e-2. We use RAdam (Liu et al., 2019a) as the optimizer and use a constant learning rate scheduler. The weight decay is 1e-2, and the maximum gradient norm is 1.0. For each dataset, the training hyper-parameters are the same for all methods for a fair comparison. We select the best model according to the performance on the dev set and report the test accuracy of the chosen model. The experiments are run on GeForce RTX 3090 GPU. Each experiment is repeated five times. Since we do not introduce any learnable parameters except PLMs, the total number of parameters of our method is the same as PLMs (RoBERTa-large and T5-large have 355M and 770M parameters, respectively).

C Details of the Competitive Fine-tuning Methods

The details of the competitive fine-tuning methods are as follows. Note that we use recommended hyper-parameters in the paper or the source code for a fair comparison.

- vanilla fine-tuning: fine-tuning has been proven to be a simple and effective method of adapting large PLMs to downstream tasks.
- BSS (Chen et al., 2019) [5]: BSS focuses on mitigating negative transfer by penalizing the small singular values of the feature matrix. We penalize the smallest singular value, and the weight of the regularization term is set as 1e-3 as recommended.
- ChildTune-F&ChildTune-D (Xu et al., 2021a) [6]: ChildTune-F&ChildTune-D update a subset of parameters (called child network) of large PLMs in the backward process. ChildTune-D utilizes the Fisher Information Matrix estimated by the pre-trained model to determine the child network. ChildTune-F uses Bernoulli distribution to determine the child network.
- Mixout [7] (Lee et al., 2019): Mixout randomly mixes the parameters of the pre-trained and the fine-tuned model to regularize the fine-tuning process. In the experiments, the mixing probability p is set as 0.9.
- NoisyTune (Wu et al., 2022): NoisyTune adds uniform noises to the parameter of the pre-trained model based on their standard deviations. The scaling factor λ, which controls the relative noise intensity, is set as 0.15.
- R3F [8] (Aghajanyan et al., 2020): R3F alleviates representational collapse by introducing parametric noise. R3F generates noise from either a normal or uniform distribution.
- RecAdam [9] (Chen et al., 2020): RecAdam optimizes a multi-task objective and utilize an annealing coefficient to gradually shift the objective from pre-training to downstream tasks.
- ReInit (Zhang et al., 2020b): Zhang et al. (2020b) verified that transferring the top pre-trained layers slows down learning and hurts performance. ReInit re-initializes the top layers of PLMs when adapting to new tasks. In our experiments, we re-initialize the top 3 transformer block.

[5] https://github.com/thuml/Batch-Spectral-Shrinkage

[6] https://github.com/alibaba/AliceMind/tree/main/ChildTuning

[7] https://github.com/bloodwass/mixout

[8] https://github.com/facebookresearch/fairseq/tree/main/examples/rxf

[9] https://github.com/Sanyuan-Chen/RecAdam

Table 6: The dataset statistics.

	Train	Dev	Test	Option Number	Question Length	Option Length
CommonsenseQA	8.5k	1.2k	1.2k	5	13.4	1.5
OpenBookQA	5.0k	0.5k	0.5k	4	10.7	2.9
ARC-Easy	2.2k	0.6k	2.4k	4	19.4	3.7
ARC-Challenge	1.1k	0.3k	1.2k	4	22.3	4.9
QASC	7.3k	0.8k	0.9k	8	8.1	1.6
PIQA	14k	1.8k	1.8k	2	7.1	19.4
SocialIQA	31k	1.9k	1.9k	3	20.1	3.6

D Related Works of Commonsense QA

Commonsense reasoning is a key pillar of human cognition and intelligence, but it is still a long-standing challenge for deep learning systems (Xu et al., 2021b; Wang et al., 2020b; Talmor et al., 2018). Current question and answering (QA) systems rely on external sources such as knowledge graphs (e.g., ConceptNet) (Yasunaga et al., 2021; Feng et al., 2020; Wang et al., 2020a; Lin et al., 2019), knowledge bases (e.g., Wiktionary) (Xu et al., 2021b) and generative pre-trained language models (e.g., GPT3 (Brown et al., 2020)) (Liu et al., 2022b; Yang et al., 2020; Rajani et al., 2019; Liu et al., 2022a), and achieve remarkable success. Despite the remarkable success, collecting high-quality external knowledge is usually expensive, and noisy knowledge is easily introduced (Liu et al., 2022b). In this paper, we present a novel fine-tuning method that retains commonsense knowledge from PLMs since they are exposed to a colossal amount of data in pre-training and inherently knowledge bases (Petroni et al., 2019; AlKhamissi et al., 2022). Different from the existing commonsense QA models, our method does not rely on KGs or GNNs. Moreover, our method can be a plug-in module to enhance the performance of commonsense QA models. We compared six commonsense QA methods in the experiments:

- Relation Network (Santoro et al., 2017) utilizes a relational reasoning structure over the knowledge graph;
- KagNet (Lin et al., 2019) aggregates information with graph convolutional networks and LSTMs, and a hierarchical path-based attention mechanism;
- RGCN (Schlichtkrull et al., 2018) extends the graph convolutional network with relation-specific weights;

Table 7: The average accuracy (%) of fine-tuning and our method when T5-large is used as the backbone model.

Methods	Fine-Tuning	CET(Ours)
CSQA	76.33 (0.55)	**76.85 (0.30)**
OBQA	68.04 (0.62)	**69.14 (0.35)**
ARC-Easy	70.96 (0.48)	**71.63 (0.34)**
ARC-Challenge	46.68 (0.53)	**48.55 (0.58)**
QASC	60.69 (0.78)	**61.79 (0.81)**
PIQA	78.96 (0.42)	**81.58 (0.55)**
SIQA	78.25 (0.38)	**79.40 (0.44)**

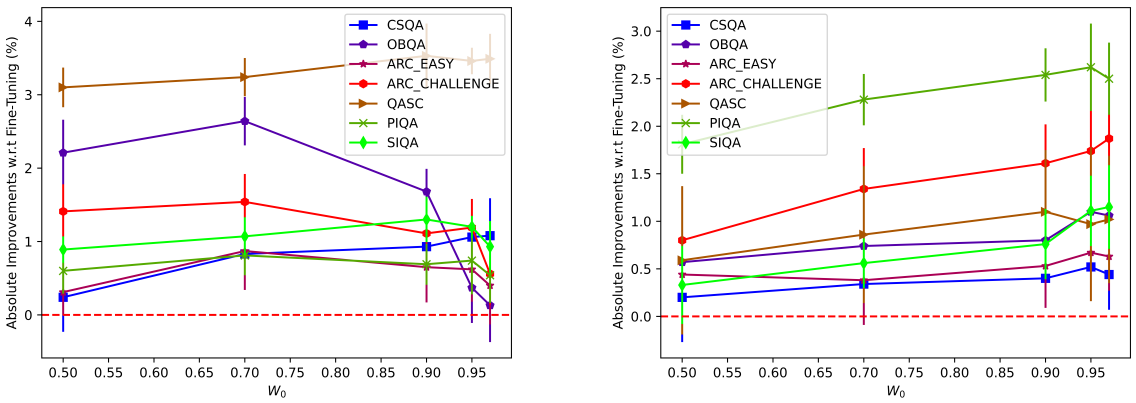
Table 8: The average accuracy (%) of our method when different K is selected.

	K=3	K=5
CSQA	76.74	**76.82**
OBQA	**70.88**	70.76
ARC-EASY	**68.59**	68.53
ARC-CHALLENGE	47.40	**47.52**
QASC	57.42	**57.57**
PIQA	79.13	**79.43**
SIQA	78.61	**78.76**

- MHGRN (Feng et al., 2020) utilizes both GNNs and path-based models for commonsense QA;
- QAGNN (Yasunaga et al., 2021) models the QA context and the knowledge graph in a joint graph and extracts their representations through a GNN.
- SAFE (Jiang et al., 2022) designs a simple MLP-based knowledge encoder that utilizes statistical relation paths as features.

E Additional Experimental Results

Experiments on T5. Our method is model-agnostic since it only requires computing the joint prediction. Different from discriminant models such as RoBERTa, T5 is a generative model whose



(a) The backbone is RoBERTa-large

(b) The backbone is T5-large

Figure 3: The absolute improvements (%) of our method w.r.t fine-tuning when $W_0 = \{0.50, 0.70, 0.90, 0.95, 0.97\}$. The backbone model is RoBERTa-large (a) and T5-large (b), respectively.

output is in text format. Following Khashabi et al. (2020), we concatenate a question and its all options with prefixes $(a), (b), (c), ...$ as the input, and expect the model to output the ground-truth option in text format. To adapt our model to T5, we substitute the prediction from the probability distribution over options to the probability distribution over vocabulary. In this way, we encourage T5 to generate the same gold answer when the input is the question of the anchor sample and its KNNs.

The experimental result is in Table 7. From the result, we find that our method still improves vanilla fine-tuning consistently, which demonstrates that our approach can be applied to various architectures. Besides, we also apply ReInit on T5 as in RoBERTa. Unfortunately, T5 fails to adapt to downstream tasks when only a few parameters are re-initiated (e.g., the self-attention layer or the cross-attention layer in the topmost transformer block). We conjecture that the final language modeling head (LM head), which maps the last hidden states to the vocabulary space, hinders the knowledge of the bottom layers to transfer to new tasks. Different from ReInit, our method is also applicable to T5 because it has no assumptions about the model architecture.

Hyper-parameter Analysis. We consider two hyper-parameters that may influence the effectiveness of our method: the number of neighbors K and the weight for controlling the strength of colliding effects W_0. Fig. 3a and 3b show that our method is robust when various W_0 are chosen. When the backbone is RoBERTa-large, our method achieves the best performance when $W_0 = 0.7$ on OBQA, ARC-Easy, and ARC-Challenge; when $W_0 = 0.9$ on QASC and SIQA; and when $W_0 = 0.97$ on CSQA. When the backbone is T5-large, our method achieves the best performance when $W_0 = 0.9$ on QASC; when $W_0 = 0.95$ on CSQA, OBQA, ARC-Easy, and PIQA; and when $W_0 = 0.97$ on ARC-Challenge and SIQA. In addition, we find that some datasets, such as CSQA, require more domain-specific knowledge while some datasets, such as OBQA, require more pre-trained knowledge. The result of K in Table 8 shows that a larger K is beneficial. Our method is also robust to K because the similarity threshold θ truncates the number of nearest neighbors for each sample.

Differences between Our Method and Data Augmentation. Our method recombines the KNN questions with the options of the anchor sample. A reasonable conjecture is that our method "adds" KNN samples to enhance generalization ability. We do the following experiment to test the hypothesis: We add the same KNN samples generated by our method into the original training set for fine-tuning. The result shows that its improvement is not statistically significant. The reason may be as follows: Recall that we set $\theta = 1.0$ on five out of six datasets where the gold answer of the anchor sample and its KNNs should be matched precisely. Therefore, on most datasets, the KNN samples recombine with the options containing their original gold answer, suggesting that they provide no additional information. Besides, the newly added samples change the data distribution of the original

Table 9: Comparison between CET and vanilla fine-tuning on NER.

Method	CoNLL2003		OntoNotes5		I2B2	
	Micro F1	**Macro F1**	**Micro F1**	**Macro F1**	**Micro F1**	**Macro F1**
Vanilla Fine-Tuning	92.52	91.09	89.35	80.42	92.81	85.61
CET (Ours)	**92.94**	**91.52**	**90.09**	**81.67**	**94.07**	**88.46**

training set.

Experiments on Named Entity Recognition. To demonstrate that CET has the potential to improve more generic tasks, we apply CET to another task, Named Entity Recognition (NER), which is a fundamental task in NLP. First, NER can be formulated as a word-level classification task. Therefore, both "anchor" and KNNs refer to a specific word. Then, we use the Euclidean distance as a metric to find the KNNs in the space of the last hidden states of PLMs. Considering NER focuses on recognizing entities, we only compute the causal effects on entity words. During training, both the sentences containing anchor and KNN words are fed into the model. And then, we compute the joint prediction as in Eq.6 by combining the score prediction of the anchor word and the corresponding KNN words. Finally, we jointly optimize the causal effects of entity words and the vanilla fine-tuning objective of non-entity words as in Eq.9.

We choose three widely used datasets for experiments: CoNLL2003 (Sang and De Meulder, 2003), Ontonotes5 (Hovy et al., 2006), I2B2 (Murphy et al., 2010). Following previous experiments, we use RoBERTa-large as the backbone. The result in Table 9 indicates that CET outperforms vanilla fine-tuning consistently.

To better understand CET, here is an example from CoNLL2003: The anchor is a Location entity "California" in the sentence "... Marine Laboratories in California say ...". Its three nearest neighbours are 1. "California" in the sentence "At California, Tim ..."; 2. "Oakland" in the sentence "OAKLAND AT NEW YORK"; 3. "Florida" in the sentence "At Florida, ...". As shown, the anchor and KNN words share the related prior knowledge of PLMs, which can also be illustrated in Figure 2.

F More examples of Colliding Effects

Table 10: Examples from PIQA and QASC.

PIQA	Gold Answer	Question
Anchor	throw it away	how do you dispose of a cutip?
KNNs	throw it away	how do you dispose of something?
	throw it away	how do you scrap metal?
QASC	**Gold Answer**	**Question**
Anchor	bacteria	What causes botulism?
KNNs	bacteria	what may die if it becomes too hot?
	bacteria	what causes serious illness?
	bacteria	What causes food to spoil?
	bacteria	What can cause people to die?
	bacteria	what feed on dead organisms?

Table 11: Examples from CSQA, OBQA, ARC-Easy, ARC-Challenge, and SIQA.

CSQA	Gold Answer	Question
Anchor	television	To prevent any glare during the big football game he made sure to clean the dust of his what?
KNNs	television	Where do you watch garbage?
	television	What home entertainment equipment requires cable?
	television	What does one watch garbage reality shows on?
	television	Where might I hear and see information on current events?
	television	James wasn't a repair person, but even he knew that he didn't need a freon coin in a what?

OBQA	Gold Answer	Question
Anchor	sun	The leaves of a plant benefit from?
KNNs	sun	The moon orbits an object that orbits the
	sun	Which of these items is required for a deer to live
	sun	What is larger then the human planet and causes cycles of day and night?
	the sun	Despite what some think, instead around themselves, our planet spins around

ARC-Easy	Gold Answer	Question
Anchor	line graph	A student wants to find the relationship between the diameter of several plastic disks and the circumference of each disk. Which of these types of graphs should be constructed to determine this relationship?
KNNs	line graph	The number of squirrels in a certain ecosystem changes over time. These changes can be represented as a number of connected data points. Which method would a student most likely use to show this information?
	line graph	In a city, the daily high and low 16 temperatures for a month are best represented by which of the following?
	line graph	A student measures the growth of a group of plants given different amounts of fertilizer. Which data display should the student use to compare the growth of the plants?
	line graph	Scientists recorded the hourly temperature at a weather station for the month of July and want to quickly measure a trend over time in temperature changes. Which of these formats would be the most appropriate representation of the temperature data to quickly measure any trend?
	line graph	The most effective way to show a change happening over time is to display your results using a

ARC-Challenge	Gold Answer	Question
Anchor	air	Four materials are put into small containers. These materials are then moved from the small containers into larger containers. Which material will spread out to completely fill a larger container?
KNNs	air	When you make soap bubbles, what is inside the bubbles?
	air	When a tadpole grows, its gills change into lungs. What does it now need to survive?
	air	How are green plants an important part of the carbon dioxide-oxygen cycle?
	air	Which of the following substances can be separated into several elements?

SIQA	Gold Answer	Question
Anchor	compassionate	Jan had always wanted a puppy, but decided to adopt an older shelter dog instead. How would you describe Jan?
KNNs	compassionate	Jan gave Kai's husband a hug after hearing the good news about Kai's recovery. How would Kai feel as a result?
	compassionate	Quinn ran over a squirrel on the road. They felt a little guilty. How would you describe Quinn?
	compassionate	Cameron was volunteering at a soup kitchen and provided assistance to individuals. How would Cameron feel afterwards?
	compassionate	Bailey found out that the local fire department lacked funding. Bailey decided to do something about it. How would you describe Bailey?
	compassionate	Ash let the dog inside as it was getting too hot for dog to be outside. How would you describe Ash?