ENSEMBLE MODELS INSTEAD OF PROMPTS: A SAMPLE-SPECIFIC KNOWLEDGE TRANSFER METHOD FOR FEW-SHOT PROMPT TUNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Prompt tuning approaches, which freeze pre-trained language models and learn task-specific soft prompts conditioning on the frozen pre-trained model during downstream fine-tuning, have attracted growing interest due to its parameter efficiency. We work on improving the few-shot performance of prompt tuning via knowledge transfer from trained soft prompts of source tasks. We first empirically find that in few-shot settings, a simple ensemble of model outputs from source tasks can already outperform existing multi-source knowledge transfer approaches, such as initializing the target prompt with the best source prompt, or source prompt fusion methods. Motivated by this observation, we further look into ensemble approaches and propose our solution, a sample-specific ensemble of source models (SESOM). Source models refer to the trained soft prompts of source tasks, together with the pre-trained language model that the source soft prompts are trained with. As the name suggests, SESOM learns from the few-shot target samples to adaptively decide how much each source model should contribute during ensembling for different target samples. Therefore, SESOM not only inherits the good generalization of ensemble methods, but also captures sample-specific preferences given different target samples. We conduct few-shot experiments across diverse NLP tasks on different model scales. SESOM outperforms existing methods by a large margin in every scenario tested.

1 INTRODUCTION

Recent few years have witnessed the great success of large pre-trained language models (Kenton & Toutanova, 2019; Liu et al., 2019; Radford et al., 2019; Raffel et al., 2020; Brown et al., 2020). The size of such pre-trained models which can easily go to billions of parameters (Brown et al., 2020; Raffel et al., 2020), however, hinder their real-world deployments and applications. Because the huge size of pre-trained language models can make model fine-tuning for downstream NLP tasks computationally expensive and memory-inefficient. To alleviate this problem, many parameter-efficient fine-tuning methods are proposed (Li & Liang, 2021; Houlsby et al., 2019; Zhang et al., 2021; Lester et al., 2021; Liu et al., 2021b). Among them, *prompt tuning* (Lester et al., 2021) is one of the most widely adopted methods. Given a downstream task, prompt tuning methods keep the entire pre-trained model frozen. Only the newly added task-specific *soft prompts* are updated given training data of the target task, conditioned on the original pre-trained model. Compared to traditional fine-tuning methods that update the entire pre-trained model, prompt tuning consumes significantly less memory and less training time per iteration (). [Prafulla can you add citations here? :) – CX]

Despite prompt tuning's advantages in practice and its continuously improved performances on various NLP tasks (Liu et al., 2021a; Vu et al., 2022), its performances in few-shot settings where labeled training data is limited, still have large space for improvements (Gu et al., 2022). In such low-data scenarios, one of the most widely applied approaches to alleviate the data shortage of the *target task*, is to seek help from *source tasks* where labeled training data is abundant. Although such knowledge transfer approaches from multiple source tasks to the target task are analyzed in some domains (Chen, 2021; li2, 2020; Sasso et al., 2022; Lee et al., 2019), relevant research in the few-shot prompt-tuning setting still seems vacant. Therefore in this paper we ask: for prompt tuning

approaches, given the trained soft prompts of source tasks, what is the most effective way to employ them to benefit few-shot learning of the target task?

With training data or trained soft prompts from source tasks, there are multiple lines of approaches that we could adopt. For example, in full-data instead of few-shot setting, existing methods either find the most suitable source soft prompt to initialize the soft prompt of the target task (Vu et al., 2022), or fuse multiple source soft prompts together to facilitate the prompt tuning on the target task (Asai et al., 2022). While in few-shot settings, our attention is attracted by model ensemble methods due to their reputation of providing good generalization and low variance (). [Praffulla can you also add the ensemble citations here :) – CX] Simple ensemble methods are observed to outperform complicated approaches in few-shot settings of other domains (Dvornik et al., 2019; Liu et al., 2020). Therefore we wonder, for few-shot prompt tuning, would an ensemble of model outputs from source tasks achieve better performance compared to existing approaches mentioned above? If so, what is the most effective model ensemble approach for the knowledge transfer from source tasks for few-shot prompt tuning?

To answer these questions, we conduct empirical analysis and find that a simple logit-averaging ensemble method can already outperform existing multi-source knowledge transfer approaches for few-shot prompt tuning. Motivated by this observation, we further look into ensemble approaches and propose our solution, a sample-specific ensemble of source models (SESOM). Source models refer to the trained soft prompts of source tasks, together with the pre-trained language model that the source soft prompts are trained with. As the name suggests, SESOM learns from the few-shot target samples to adaptively decide how much each source task should contribute given different target samples. Specifically, our method trains an attention-style network to generate weights to ensemble the outputs of different source models, in order to make the prediction given each target sample. Through this way, our model is able capture the sample-specific preferences to ensemble different source models given the few-shot labelled target data. Compared to existing knowledge transfer approaches for prompt tuning that provide a fixed knowledge transfer strategy for all target samples, SESOM is more effective due to its sample-specific strategy. Compared to existing approaches that conduct sample-specific soft prompt fusion, SESOM is more effective in few-shot settings because the model-level ensemble provides better robustness and generalization to unseen samples.

We conduct experiments across 6 source tasks (MNLI, QNLI, QQP, SST-2, SQuAD and ReCoRD) and 8 target tasks (WNLI, MRPC, BOOlQ, MultiRC, RTE, WiC, WSC and CB) on 3 model scales, T5-Base, T5-Large and T5-XL. Experiment results show that SESOM outperforms existing methods by large margins in every scenario tested. We also show in experiments that ensemble methods in general achieve better performances compared to other existing knowledge transfer approaches in few-shot prompt tuning. Moreover, we also find that SESOM can achieve better performances given more source tasks, although the newly added tasks are less suitable sources for knowledge transfer in general given the target task. It implies that SESOM's sample-specific strategy is able to set proper sample-specific weights for source tasks when the generally less favourable source tasks can actually perform well. [Is it safe to emphasize this here – CX]

2 Related Work

Knowledge transfer approaches in the context of prompt tuning. Since the emergence of prompt-tuning methods, a lot of research has focused on improving the performance of prompt-tuning methods in the setting of full-data fine-tuning. Compared to traditional fine-tuning method, people find that prompt-tuning approaches are less data efficient, given the fact that they need more training data/training iterations to achieve comparable performances with traditional fine-tuning (). Therefore, some research focuses on transferring knowledge from other tasks similar to the target task, to facilitate prompt tuning of the target task. Among them, SPOT () first learns a prompt on one or more source tasks and then uses it to initialize the prompt for a target task. SPOT significantly boosts the performance of prompt-tuning across many tasks. Similarly, PPT () pre-trains the soft prompt of the target task with data formulated similarly with target data. These two methods provide a fixed knowledge transfer strategy for all target samples, given that they both provide initialization before few-shot fine-tuning of the target task. Different from them, our method provides sample-specific knowledge transfer from source models to each target samples, leading to better performance on few-shot fine-tuning. More recently, Asai et al. (2022) proposes a sample-specific

knowledge fusion method for full-data prompt tuning. In such method, the soft prompts of source tasks are fused together to construct a new prompt for each target sample, instead of ensembling the source models' outputs given each target sample in our method. We empirically find that such prompt-level fusion methods perform significantly worse than our model-level ensemble method in few-shot settings. It indicates that when training data is limited, model-level ensemble might provides better generalization compared to prompt-level knowledge fusion.

Ensemble learning has been a popular approach to obtain a low-variance and generalizable model. The most commonly used ensembling techniques include unweighted averaging (), majority voting () and maximum over logits (), [just a placeholder, TODO look at prior NLP work and group them and cite them.. – PKC]. Within the paradigm of prompt-based learning, Schick & Schütze (2021) explored unweighted average of logits corresponding to different human-selected verbalizers for zero-shot classification, and Lester et al. (2021) used majority voting over logits from five prompts trained on the same task. They all employ identical ensembling strategy for all test samples thereby under-utilizing the capability (better word?) of using many verbalizers or prompts. For instance, Schick & Schütze (2021) found that the oracle that selects the best verbalizer for every test sample significantly outperforms the unweighted-averaging model. SESOM fills that gap by taking over the role of oracle and train a separate attention module to generate sample-specific importance weight for every source prompt. [should I discuss Dynamic Classifier selection ()?, might question the novelty of this work, and I have never seen this type of model in NLP papers? – PKC]

3 METHOD

3.1 PRELIMINARIES

Given that our proposed method SESOM is based on prompt tuning, we introduce the basics of prompt tuning in this subsection. In SESOM, given the training data of source tasks and a pre-trained language model, we train the soft prompts $[\mathbf{P}_1, ..., \mathbf{P}_t]$ for source tasks $T_1, ..., T_t$ by running prompt tuning (Lester et al., 2021).

In prompt tuning (Lester et al., 2021), all downstream tasks are solved in a text-to-text format. In such a text-to-text format, each label of a training sample is represented by a *verbalizer* (), or a verbalizer together with a task-specific *template* optionally (). Formally, each instance in the source task T_j where $j \in [1, t]$ is represented as (X, y), where X is a sequence of tokens, and y is a classification label. Then, we represent the verbalizer or the verbalizer-template sequence of class label y as Y.

Before prompt tuning, for the task T_j, a task specific soft prompt $\mathbf{P}_j = [\mathbf{p}_1, ..., \mathbf{p}_m] \in \mathbb{R}^{m \times d}$ is randomly initialized. m is the number of soft prompt embeddings for the task and d is the embedding dimension of each prompt embedding. We represent the sequence of token embeddings of the input tokens X as $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_l] \in \mathbb{R}^{l \times d}$ where l is the length of the token sequence. Prompt tuning prepends the task-specific soft prompt $\mathbf{P}_j$ to the token embedding sequence $\mathbf{X}$ as the final input $[\mathbf{P}_j; \mathbf{X}]$. Then, $[\mathbf{P}_j; \mathbf{X}]$ is fed into the frozen language model to make the prediction. The target task is modeled as $\Pr_\theta(\mathbf{Y} \mid [\mathbf{P}_j; \mathbf{X}])$ where θ is the parameters of the frozen pre-trained model. During prompt tuning, θ is frozen and only the soft prompt $\mathbf{P}_j$ is updated by maximizing $\Pr_\theta(\mathbf{Y} \mid [\mathbf{P}_j; \mathbf{X}])$.

3.2 SAMPLE-SPECIFIC ENSEMBLE OF SOURCE MODELS

Given a collection of source tasks $T_1, ..., T_t$, their corresponding soft prompts $[\mathbf{P}_1, ..., \mathbf{P}_t]$ trained from previous sub-section, and the pre-trained language model θ that these soft prompts are conditioned on, SESOM aims to transfer knowledge from these source resources under few-shot settings. We represent each trained source model for source task T_j as $[\mathbf{P}_j; \theta]$. In SESOM, we firstly fine-tune each source model $[\mathbf{P}_j; \theta]$ with the few-shot labeled data from the target task in a prompt-tuning manner. This is to enforce source models to adapt to the target task's templates and verbalizers.

Then, given a labeled instance (X, y) from target task T_{target}, same as prompt tuning, SESOM feeds $[\mathbf{P}_j; \mathbf{X}]$ into its corresponding source model $[\mathbf{P}_j; \theta]$. We then get the *source logit* $\mathbf{l}_{x,j}$ as source model $[\mathbf{P}_j; \theta]$'s output given target input sample X. $\mathbf{l}_{x,j}$ is the pre-softmax logit from [I

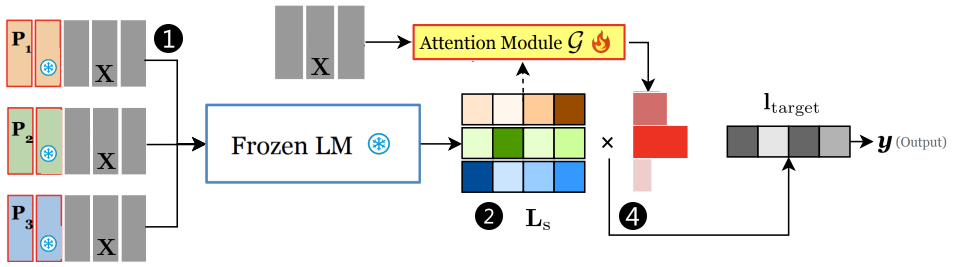

Figure 1: Overview of SESOM. The part with 🔥 is updated during training while those with ❄ are frozen. ❶ Pre-trained soft prompts are prepended to the embedded input. ❷ Frozen LM computes the logits of next sequence. ❸ Input-logit attention is calculated between embedded input and output logits. ❹ Attention coefficients are used to compute a linear combination of the output logits, which serve as the final output logit.

think we need to be more specific here about what this is and maybe add an equation. – CX] SESOM then trains our attention-style network $\mathcal{G}$ to compute sample-specific weights for all source logits $[\mathbf{l}_1, ..., \mathbf{l}_t]$ to ensemble them to make the prediction for $\mathbf{X}$. Specifically, we denote the logits from t source models with $\mathbf{X}$ as input, as $\mathbf{L}_x = [\mathbf{l}_1, ..., \mathbf{l}_t] \in \mathbb{R}^{t \times v}$, where v is the vocabulary size of the pre-trained model. SESOM then computes the contribution weights of *source logits* $\mathbf{L}_x$ with respect to the current target sample $\mathbf{X}$ by our attention-style network $\mathcal{G}$.

Attention module $\mathcal{G}$. SESOM firstly performs max-pool operation for the embedding input $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_l] \in \mathbb{R}^{l \times d}$ to obtain $\hat{\mathbf{x}} \in \mathbb{R}^{1 \times d}$.[Becky: embedding;][is X the output representation of languages or word embedding? Reword this sentence – CX]

$$\hat{\mathbf{x}} = [max(\mathbf{x}_1), ..., max(\mathbf{x}_l)] \tag{1}$$

where l is the length of the token sequence, and d is the embedding dimension of each prompt embedding. [Becky: replicated with Preliminary section.]

Given the representation $\hat{\mathbf{x}}$ of the input target input $\mathbf{X}$, we apply a feed-forward network to project $\hat{\mathbf{x}}$ non-linearly to a new space,

$$\mathbf{h}_x = \text{layer_norm}(W_{u,x}^T \cdot \gamma(W_{d,x}^T \cdot \hat{\mathbf{x}})) \tag{2}$$

where $\gamma(\cdot)$ is a non-linear activation function, $W_{d,x} \in \mathbb{R}^{d \times d'_x}$ and $W_{u,x} \in \mathbb{R}^{d'_x \times d'}$ are trainable weights. Then we apply Layer Norm Ba et al. (2016) to get the final projected representation of target input $\mathbf{X}$, $\mathbf{h}_x \in \mathbb{R}^{d'}$.

Similarly, we project logit $\mathbf{l}_j \in \mathbb{R}^v$ of each source model given $\mathbf{X}$ into the same space,

$$\mathbf{h}_{l,j} = \text{layer_norm}(W_{u,l}^T \cdot \gamma(W_{d,l}^T \cdot \mathbf{l}_j)), \tag{3}$$

where $W_{d,l} \in \mathbb{R}^{v \times d'_l}$ and $W_{u,l} \in \mathbb{R}^{d'_l \times d'}$ are trainable weights. Then, given $\mathbf{h}_x$ and the projected representation of all source logits $\{\mathbf{h}_{l,j}\}_{j=1}^t$, we compute the attention score a_j for the source logit $\mathbf{l}_j$ by

$$a_j = \frac{e^{(\mathbf{h}_{l,j} \cdot \mathbf{h}_x)}}{\sum_{k=1}^t e^{\mathbf{h}_{l,k} \cdot \mathbf{h}_x}}. \tag{4}$$

SESOM produces the final output logit $\mathbf{l}_{\text{target}} \in \mathbb{R}^{1 \times v}$ by computing a linear combination of $\mathbf{l}_s$ given the computed input-logit attention,

$$\begin{aligned}\mathbf{l}_{\text{target}} &= \mathcal{G}(\mathbf{X}, [\mathbf{l}_1, ..., \mathbf{l}_t]) \\ &= \sum_{j=1}^{t} a_j \mathbf{l}_j \end{aligned} \tag{5}$$

Then similar with prompt tuning in Section 3.1, we keep all the parameters θ in pre-trained language model and source soft prompts ($[\mathbf{P}_1, ..., \mathbf{P}_t]$) frozen. Attention module $\mathcal{G}$ is trained by minimizing its cross-entropy loss between output logits $\mathbf{l}_{\text{target}}$ and the label $\mathbf{Y}$.

During the few-shot training, we update the attention module $\mathcal{G}$ with the few-shot labeled target samples. Through this way, $\mathcal{G}$ is trained to capture the sample-specific preference of different source models. At inference, we also use $\mathcal{G}$ to calculate the sample specific ensemble weight of logits of different source models, and calculate the weighted average of them as the final logit to make the prediction.

Implementation Details [Here I want to add the description of our verbalizer mapping. – CX][Becky: Appendix A.1, will see whether we need to move it here later]

4 Experiments

In this section, we first describe our experimental setup, then our main empirical results compared to baselines. Finally we conduct various ablation studies and case study to show the effectiveness of different components of SOSEM and reasonings behind our design choices.

4.1 Experimental Setup

Hyper-parameter set up for few-shot prompt tuning. [The pre-trained langauge models we used, the lr and other hyper-parameters we use to train them. the dimensions of our attention network. the few-shot details: how much random seeds we use for each exps, how many labelled data we use, and the data we evaluate on. @becky – CX] [Becky: —] We randomly initialize attention module $\mathcal{G}$ based on Section 3.2. The detail of hyper-parameters of SESOM such as the dimensions of $\mathcal{G}$ and learning rate can be found in Appendix A.4. We use T5-{base,large,XL} as language model θ and $[\mathbf{P}_1, ..., \mathbf{P}_t]$ trained in Section 3.1 and Appendix A.2 as source prompts. For each target dataset, D_{train} and D_{dev} of each task are both 32 samples with 20 random seeds from the original training data[1]. The original validation set is used to evaluate the model as test set.

Source and target tasks. Following previous work(Asai et al., 2022; **?**), we use 6 tasks that have been considered generally helpful to other NLP tasks as our source tasks. They are MNLI (Williams et al., 2018), QNLI (Demszky et al., 2018), QQP (Wang et al., 2018), SST2 (Socher et al., 2013), SQuAD (Rajpurkar et al., 2016), and ReCoRD (Zhang et al., 2018). Given each pre-trained model tested (T5-Base, T5-Large, T5-XL), we train soft prompts for these source tasks separately using prompt tuning. [The hyper-parameters for training the soft prompts come here – CX][Becky: see right] Details can be found in Appendix A.2.[learning rate and weight decay and batch size and other optimizer pls. – CX][Becky: added.]

We then evaluate our model on the following GLUE (Wang et al., 2018) and SuperGLUE (Wang et al., 2019) tasks: WNLI (Wang et al., 2018), MRPC (Dolan et al., 2005), BoolQ (Clark et al., 2019), MultiRC (Khashabi et al., 2018), RTE (Giampiccolo et al., 2007), WiC (Pilehvar & Camacho-Collados, 2019), WSC (Levesque et al., 2012), and CB (De Marneffe et al., 2019). They cover a wide range of NLP tasks including question answering, sentiment analysis and textual entailment.

Baselines. We compare SESOM not only with existing knowledge transfer approaches in the context of prompt tuning, but also with multiple existing ensemble strategies. The knowledge transfer methods for prompt tuning are,

[1]https://huggingface.co/datasets

Model	Method	**WNLI**	**MRPC**	**RTE**	**MultiRC**	**BoolQ**	**WiC**	**WSC**	**CB**	**Avg**
T5-Base	PT	56.27	34.01	48.5	53.05	49.98	52.19	50.58	49.46	$49.26_{5.85}$
	SPoT-t	54.44	53.32	54.15	50.61	56.87	54.51	47.98	43.48	$51.92_{10.64}$
	PPT	47.89	68.97	**67.51**	60.32	51.81	57.25	**63.46**	50.00	$58.40_{1.03}$
	ATTEMPT	52.95	75.35	52.63	59.52	54.28	54.48	43.89	43.74	$54.61_{6.48}$
	Uni Ensemble	**56.34**	75.00	57.40	65.92	70.12	62.69	45.19	57.14	$61.22_{3.87}$
	FW Ensemble	50.07	77.57	65.03	66.32	72.28	63.14	51.39	43.74	$62.28_{4.58}$
	SESOM	**56.34**	**84.34**	66.66	**66.91**	**74.80**	**63.68**	52.68	**74.02**	$\mathbf{67.43}_{2.12}$
T5-Large	PT	56.34	51.02	68.56	75.78	75.78	56.25	60.34	55.0	$62.38_{7.14}$
	SPoT-t	54.93	72.46	65.14	64.83	53.66	56.26	48.76	53.7	$58.72_{14.23}$
	PPT	43.66	79.41	75.81	65.5	67.68	53.61	61.54	80.36	$65.95_{1.00}$
	ATTEMPT	42.25	67.96	62.63	70.11	68.42	54.33	60.04	69.64	$61.92_{6.37}$
	Uni Ensemble	**56.34**	86.76	67.14	74.36	78.83	64.42	60.42	58.78	$68.38_{3.35}$
	FW Ensemble	**56.34**	83.74	77.97	73.63	**81.28**	61.44	56.73	62.85	$69.24_{3.19}$
	SESOM	**56.34**	**87.84**	**80.63**	**74.40**	80.99	**67.06**	**63.08**	**84.73**	$\mathbf{74.38}_{1.09}$
T5-XL	PT	56.34	87.33	72.71	73.47	78.46	65.58	61.25	75.98	$71.39_{4.73}$
	SPoT-t	56.34	77.65		58.76	57.84				
	PPT	50.07	76.22	70.03	67.57	69.89	58.62	59.73	78.92	$66.38_{0.85}$
	ATTEMPT	49.37	62.46	84.24	71.26	60.39	42.5	45.38	80.18	$61.97_{6.65}$
	Uni Ensemble	56.34	75.73	89.16	76.56	81.07	58.77	63.40	78.57	$72.45_{3.61}$
	FW Ensemble	56.27	85.62	89.58	76.20	82.83	66.79	57.55	73.48	$73.54_{2.41}$
	SESOM	**56.34**	**87.70**	89.04	**76.90**	82.34	**68.05**	**63.41**	**84.15**	$\mathbf{76.00}_{1.44}$

Table 1: Few-shot performance of all methods. All the scores are the average of 20 runs. The best scores are in bold. Uni Ensemble is the acronym for Uniform Ensemble. FW Ensemble is the acronym for Fixed-weight Ensemble.

- **SPoT-t** (Vu et al., 2022): The original SPoT-t first trains soft prompt with the full training set of the target task for a few epochs, and then retrieve the source soft prompt which shares the highest cosine similarity with this target prompt to re-initialize the target prompt. Then, it re-train the target prompt starting from this new initialization. To adapt SPoT-t to our few-shot setting, we keep everything unchanged except that we train the target prompt with the few-shot labelled data, instead of with the full target training set.
- **PPT**(Gu et al., 2022): PPT pre-trains the target prompt with additional data by pre-processing the additional data to the format of the target task. For fair comparison with other approaches, we use the training data of all 6 source tasks to pre-train the target soft prompt. We then update the target soft prompt with the few-shot target data as in prompt tuning.
- **ATTEMPT**(Asai et al., 2022): ATTEMPT is a sample-specific knowledge fusion method for full-data prompt tuning. In this method, the soft prompts of source tasks are fused together to construct a new prompt for each target sample, instead of ensembling the source models' outputs given each target sample in our method. Similar with SPoT-t, in our few-shot setting, we keep everything unchanged for ATTEMPT except that we train their model with the few-shot labelled data, instead of with the full target training set.

The ensemble baselines are,

- **Uniform-Ensemble**: the simplest ensemble method of the source models. Same as SESOM, we first fine-tune each source model with the target few-shot labelled data. Then we take the average of logits of all source model's output given a target sample to make the prediction. The only difference of Uniform-Ensemble from SESOM is that Uniform-Ensemble simply averages the source logits for all target samples instead of conducting a sample-specific weighted average.
- **Fixed-weight Ensemble**: Instead of simply averaging logits of source models as Uniform-Ensemble, Fixed-weight Ensemble takes a weighted average of all source logits to make predictions. The voting weight of each source model is fixed for all target samples. The voting weight is calculated according to the F1 score of the source models given the few-shot target data. The better a source model performs given the labelled few-shot data, larger voting weight it would have for all target samples.

More details about the training of all baseline methods above can be found in Appendix A.3.

Model	# of s.	**WNLI**	**MRPC**	**RTE**	**MultiRC**	**BoolQ**	**WiC**	**WSC**	**CB**	**Avg**
T5-Base	1	54.44	53.32	54.15	50.61	56.87	54.51	47.98	43.48	$51.92_{10.64}$
	3	56.97	72.59	62.49	61.59	67.48	61.43	48.08	64.11	$61.84_{7.48}$
	5	56.41	81.91	64.01	66.08	72.96	61.54	49.04	69.46	$65.18_{3.40}$
T5-Large	1	54.93	72.46	65.14	64.83	53.66	56.26	48.76	53.7	$58.72_{14.23}$
	3	56.34	86.27	72.17	73.55	76.36	62.24	56.30	68.04	$68.91_{6.93}$
	5	56.34	87.43	77.60	74.09	80.28	66.05	60.19	80.98	$72.87_{3.41}$

Table 2: Results of SESOM with top k source prompts under few-shot setting over 20 random seeds. # of s. denotes the number of source prompts used in SESOM.

4.2 Main Results

The few-shot performance of SESOM and other methods are shown in Table 4.4. First, on every pre-trained model (T5-Base, T5-Large and T5-XL), SESOM outperforms all baselines on the average score by a large margin. SESOM also outperforms baselines on a definite majority of single target tasks on pre-trained models with different sizes. Specifically, compared to approaches that provide a fixed knowledge transfer strategy for all target samples (SPoT-t, PPT, Uni Ensemble, FW Ensemble), SESOM's superior performance indicates the effectiveness of its sample-specific strategy. Compared to existing approaches that conduct sample-specific soft prompt fusion (ATTEMPT), SESOM's superior performance indicates the effectiveness of its model ensemble strategy in few-shot settings.

Second, if we compare the performances of ensemble methods (Uniform Ensemble, Fixed-weight Ensemble and our proposed method SESOM) versus the rest methods, ensemble methods have superior performance against the rest methods in the few-shot setting. The bigger the pre-trained model is, more apparent the performance advantages of ensemble methods are in the few-shot setting. It is likely due to the widely observed strengths of ensemble methods that provide a better generalization and lower variance during inference, as illustrated in Section 1 as our motivation. Finally, compared with all methods, ATTEMPT seems to have the most unsatisfactory results in few-shot settings. ATTEMPT uses the few-shot target data to train a network to output weights to fuse source soft prompts for each target sample. Although it achieves great performances in full-data fine-tuning, few-shot training is not an easy scenario for it. Because it uses few-shot data to train a neural network, which could easily cause overfitting. While our proposed SESOM takes the logits of source models as input and outputs weights for model ensemble during few-shot training, which takes advantage of the superior generalization and robustness of model ensemble approaches.

4.3 Analysis

Would more "ordinary" source tasks help SESOM? In our main experiments, we use 6 source tasks as mentioned earlier, regardless of their actual transferability given the target task. If a source task is quite "different" from the target task, it is possible that this source task wouldn't effect the target task positively. In order to investigate how SESOM performs when less preferable source tasks are added, we conduct experiments as follows. First, given each target task, we pick top 1 and top 3 most helpful source tasks following SPoT-t () given few-shot target samples. Then we apply SESOM with the top 1 and top 3 source tasks separately, and compare with SESOM's main results with 6 default source tasks. Results are shown in Table 2. From Table 2 we can see that SESOM can achieve better performances when the number of source tasks increases, although the newly added source tasks might not be preferable to the target task in general. It indicates that for the target samples on which the generally less favourable source tasks can actually perform well, SESOM is able to set proper sample-specific weights for the source tasks, which implies the effectiveness of SESOM sample-specific strategy.

The effect of the number of shots. In main experiments, the number of few-shot labeled data we use for each target task is 32 following existing work. We wonder how SESOM would perform when we vary the number of shots compared to other methods. Therefore we have tried to set the number of shots as 64, and blablabla. Results are presented in Table 2. We can find that SESOM consistently outperforms baseline method on every number of shots tested. It suggests that SESOM can be applied to a wide range of few-shot settings.

Verification of other design choices. Before the current SESOM, we have also tried some related methods that turned out to be less effective. For example, instead of using source logits as our attention module's other input given input sample X in SESOM, we have tried using the source prompts as inputs, i.e., replacing $\mathbf{l}_j$ in Eq.2 with $\mathbf{P}_j$. Other components of this method stay the same as SESOM. We refer to this method as Ensemble acc. source prompts. We have also tried self-train approaches. Specifically, we prompt tune six source prompts on the target dataset under fewshot setting. Then each prompt-tuned source prompts with language model are used to predict the pseudo labels for the entire target datasets. These pseudo labelled samples are used to train ATTEMPT(Asai et al., 2022) before training on target datasets under fewshot setting.

4.4 Case Study

Method	**WNLI**	**MRPC**	**RTE**	**MultiRC**	**BoolQ**	**WiC**	**WSC**	**CB**	**Avg**
PPT	50.70	87.50	80.87	71.39	79.57	66.61	40.38	78.57	$69.45_{0.01}$
ATTEMPT	47.61	88.14	75.31	70.86	79.37	68.50	59.23	75.36	$70.55_{1.98}$
FW Ensemble	54.93						63.46	76.79	
SESOM	57.75							82.14	

Table 3: Results on GLUE and SuperGLUE on full dataset. All the scores are the average of 5 runs. "PPT", "ATTEMPT" denote prompt tuning, and **?** with T5-base. (TODO: FW Ensemble)

Method	**WNLI**	**MRPC**	**RTE**	**MultiRC**	**BoolQ**	**WiC**	**WSC**	**CB**	**Avg**
Att. prompt									
Majority Vote	49.75	60.30	58.38	**80.81**	66.89	**70.81**	50.8	60.69	$62.30_{0.28}$
Self-Train	50.07	77.57	65.03	65.32	72.98	63.14	51.39	43.74	$61.15_{6.37}$
SESOM	**56.34**	**84.34**	**66.66**	66.91	**74.80**	63.68	**52.68**	**74.02**	$\mathbf{67.43}_{2.12}$

Table 4: Results on GLUE and SuperGLUE under fewshot setting. "ATTEMPT w/o $\mathbf{P}_{\text{target}}$" and "ATTEMPT" denote **?** without and with randomly initialized target prompt with T5-base. "Self-Train" indicates the training "ATTEMPT" with pseudo labelled dataset. (TODO: replace ATTEMPT with attention.)

5 Closing Remarks

References

GradMix: Multi-source transfer across domains and tasks, 2020.

Akari Asai, Peters-Matthew E Salehi, Mohammadreza, and Hannaneh Hajishirzi. Attentional mixtures of soft prompt tuning for parameter-efficient multi-task knowledge sharing. *arXiv preprint*, 2022.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

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

Yifei Chen. A transfer learning model with multi-source domains for biomedical event trigger extraction. *BMC genomics*, 22(1):1–18, 2021.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. 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)*, pp. 2924–2936, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

Marie-Catherine De Marneffe, Mandy Simons, and Judith Tonhauser. The commitmentbank: Investigating projection in naturally occurring discourse. In *proceedings of Sinn und Bedeutung*, volume 23, pp. 107–124, 2019.

Dorottya Demszky, Kelvin Guu, and Percy Liang. Transforming question answering datasets into natural language inference datasets. *arXiv preprint arXiv:1809.02922*, 2018.

Bill Dolan, Chris Brockett, and Chris Quirk. Microsoft research paraphrase corpus. *Retrieved March*, 29(2008):63, 2005.

Nikita Dvornik, Cordelia Schmid, and Julien Mairal. Diversity with cooperation: Ensemble methods for few-shot classification. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 3723–3731, 2019.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and William B Dolan. The third pascal recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*, pp. 1–9, 2007.

Yuxian Gu, Xu Han, Zhiyuan Liu, and Minlie Huang. PPT: Pre-trained prompt tuning for few-shot learning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8410–8423, Dublin, Ireland, May 2022. Association for Computational Linguistics.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pp. 2790–2799. PMLR, 2019.

Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pp. 4171–4186, 2019.

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 252–262, 2018.

Joshua Lee, Prasanna Sattigeri, and Gregory Wornell. Learning new tricks from old dogs: Multi-source transfer learning from pre-trained networks. *Advances in neural information processing systems*, 32, 2019.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 3045–3059, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

Hector Levesque, Ernest Davis, and Leora Morgenstern. The winograd schema challenge. In *Thirteenth international conference on the principles of knowledge representation and reasoning*, 2012.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. 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)*, pp. 4582–4597, 2021.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *arXiv preprint arXiv:2110.07602*, 2021a.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. Gpt understands, too. *arXiv preprint arXiv:2103.10385*, 2021b.

Yaoyao Liu, Bernt Schiele, and Qianru Sun. An ensemble of epoch-wise empirical bayes for few-shot learning. In *European Conference on Computer Vision*, pp. 404–421. Springer, 2020.

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

Mohammad Taher Pilehvar and Jose Camacho-Collados. WiC: the word-in-context dataset for evaluating context-sensitive meaning representations. 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)*, pp. 1267–1273, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

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

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392, Austin, Texas, November 2016. Association for Computational Linguistics.

Remo Sasso, Matthia Sabatelli, and Marco A Wiering. Multi-source transfer learning for deep model-based reinforcement learning. *arXiv preprint arXiv:2205.14410*, 2022.

Timo Schick and Hinrich Schütze. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pp. 255–269, Online, April 2021. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 1631–1642, 2013.

Tu Vu, Brian Lester, Noah Constant, Rami Al-Rfou', and Daniel Cer. SPoT: Better frozen model adaptation through soft prompt transfer. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5039–5059, Dublin, Ireland, May 2022. Association for Computational Linguistics.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 353–355, Brussels, Belgium, November 2018. Association for Computational Linguistics.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in neural information processing systems*, 32, 2019.

Adina Williams, Nikita Nangia, and Samuel Bowman. 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: Human Language Technologies, Volume 1 (Long Papers)*, pp. 1112–1122, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

Ningyu Zhang, Luoqiu Li, Xiang Chen, Shumin Deng, Zhen Bi, Chuanqi Tan, Fei Huang, and Huajun Chen. Differentiable prompt makes pre-trained language models better few-shot learners. In *International Conference on Learning Representations*, 2021.

Sheng Zhang, Xiaodong Liu, Jingjing Liu, Jianfeng Gao, Kevin Duh, and Benjamin Van Durme. Record: Bridging the gap between human and machine commonsense reading comprehension. *arXiv preprint arXiv:1810.12885*, 2018.

A IMPLEMENTATION DETAILS

A.1 MAPPING

Source prompts are trained on six source datasets in a text-to-text format. Different source datasets have different verbalizers. For example, MNLI (**?**) is classified as "entailment", "neutral" and "contradiction"; and QNLI (Demszky et al., 2018) is classified as "entailment" and "not entailment".

In order to conduct a correct weighted average of logits for SESOM, Uniform and Eval-Vote, we conduct a verbalizer mapping for logits. Based on original labels in huggingface dataset, we develop a mapping dictionary $\mathbf{M}$ to map text to the token. For example, we map "True" to "1", and "False" to "0" for the prediction of Squad and Record. More specifically, $\mathbf{M}[10998] = 209$ and $\mathbf{M}[10747] = 3$, where 10998 and 10747 are the first token representing "True" and $False$; 209 and 3 are the first token representing "1" and "0".

Formally, for the ith source prompt, the logit $l_i = [l_{i,1}, ..., l_{i,v}]$ obtained from language model is transformed to $\hat{l}_i$ by Algorithm 1, where v is the size of tokens.

Algorithm 1: Mapping Algorithm

Data: $\mathbf{M}$, l_i
Result: $\hat{l}_i$
$\hat{l}_i \leftarrow l_i$;
for $k \in$ *keys of* $\mathbf{M}$ **do**
  $\hat{l}_{i,k} \leftarrow min(l_{i,k}, l_{i,\mathbf{M}[k]})$;
  $\hat{l}_{i,\mathbf{M}[k]} \leftarrow max(l_{i,k}, l_{i,\mathbf{M}[k]})$;
end

A.2 SOURCE PROMPTS TRAINING

We use the source prompts trained by (Asai et al., 2022)[2]. These source prompts are prompt-tuned(**?**) respectively on MNLI (**?**), QNLI (Demszky et al., 2018), QQP (**?**), SST2 (Socher et al., 2013), SQuAD (**?**), and ReCoRD (Zhang et al., 2018). Soft tokens size (m) is 100 and embedding dimensions (d) are 768, 1024 and 1024 respectively for T5-{base, large, 3b}.

Parameters	**Value**
max source length	512
learning rate	3e-1
train epoch	10
optimizer	AdamW
batch size	16

A.3 BASELINES

A.3.1 FINE-TUNED (FT) MODELS

We fine-tune T5-{base, large, 3b} on target datasets. D_{train} and D_{dev} are both 32 samples. The following hyperparameters are used in training FT.

Parameters	**Value**
max source length	216
learning rate	5e-4
train epoch	20

[2]https://github.com/AkariAsai/ATTEMPT

A.3.2 SPoT-TARGET (SPoT-T)

We first trains a randomly initialized soft prompt on a target task, which is referred as *target prompt*. D_{train} and D_{dev} are the same with the ones used in training FT. Then we calculate the cosine similarity between the target prompt and the source prompts. The source prompt which shares the highest similarity is used to initialize the soft prompt. We then prompt-tune this soft prompt on the same D_{train} and D_{dev}.

Parameters	**Value**
max source length	256
learning rate	3e-1
batch size	16
train epoch	30

A.3.3 PPT

For a fair comparison, we replicate **?** only with pre-training a soft prompt on the datasets we used for training source prompts. Following **?**, we firstly unify six source tasks to a single format——multiple-choice classification. We then prompt-tune a randomly initialized soft prompt on this unified dataset over 20 random seeds. We use the same hyperparametes as training SPoT-t.

A.3.4 UNIFORM-ENSEMBLE MODELS

Same as SESOM, pre-trained language model (T5-{base, large, 3b}) with six source prompts, $[\mathbf{P}_1, ..., \mathbf{P}_t]$, are firstly used to computes six logits. Different verbalizers are used during training soft prompts. So we also conduct verbalizer mapping for the logits of tokens which share the same meaning (Appendix A.1). Then we use the uniformly weighted average of these logits as the final logit to produce predictions.

A.3.5 FIXED-WEIGHT ENSEMBLE MODEL

Fixed-weight Ensemble model takes a weighted average of all source logits to make predictions. We firstly prompt-tune six source prompts $[\mathbf{P}_1, ..., \mathbf{P}_t]$ on D_{train} and then obtain the F1 scores $S = [s_1, s_2, ..., s_6]$ of D_{dev} which are used as the fixed weight. Same with Uniform-Ensemble models, we also conduct verbalizer mapping for $[\mathbf{l}_1, ..., \mathbf{l}_t]$. Then the weighted average of all source logits is used as $\mathbf{l}_{\text{target}}$.

A.4 SESOM

We first randomly initialize attention module $\mathcal{G}$ based on Section 3.2. The dimensions of $\mathcal{G}$ is shown as follows,

Model	Method	**WNLI**	**MRPC**	**RTE**	**MultiRC**	**BoolQ**	**WiC**	**WSC**	**CB**
T5-base	d'_x	32	32	100	100	64	100	100	64
	d'	32	128	100	100	256	100	100	64
	dropout %	50	0	0	0	0	0	0	0
T5-large	d'_x	100	64	64	32	100	100	32	64
	d'	100	64	512	128	100	100	64	256
	dropout %	50	0	0	50	0	0	50	0
T5-XL	d'_x	100	64	64	100	128	64	64	64
	d'	100	64	64	100	512	512	64	512
	dropout %	50	0	50	0	0	0	50	0

We then get the source logits $\mathbf{l}_x = [\mathbf{l}_{x,1}, ..., \mathbf{l}_{x,t}]$ as source models $\{[\mathbf{P}_1; \theta], ..., [\mathbf{P}_t; \theta]\}$'s output given target input sample X. Attention module $\mathcal{G}$ takes $\hat{\mathbf{x}}$ as key and $\mathbf{l}_x$ as value to calculate attention scores. We then conduct verbalizer mapping for $\mathbf{l}_x$ based on Appendix A.1. The final output logit $\mathbf{l}_{\text{target}}$ is calculated according to Equation 5. The training hyper-parameters is shown as follows,

Parameters	Value
max source length	256
learning rate	3e-1
batch size	32 (T5-base); 16 (T5-large); 4 (T5-XL)
train epoch	20

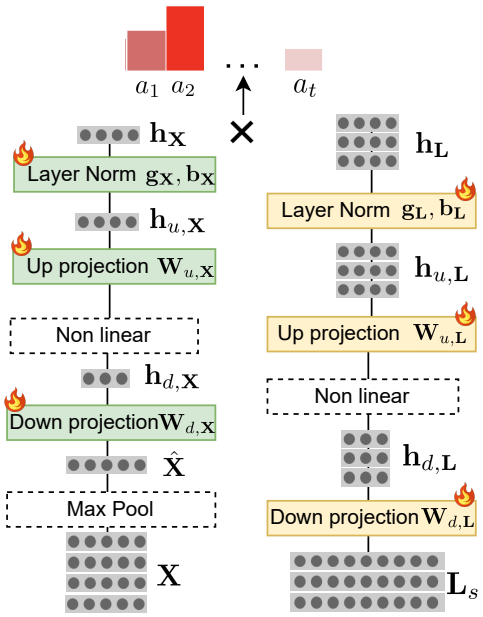

Figure 2: The overview of our attention module $\mathcal{G}$ for calculating input-logit attentions.[Becky: change later]