ASCM: An Answer Space Clustered Prompting Method without Answer Engineering

Zhen Wang[1,3], **Yating Yang**[1,2,3*], **Xi Zhou**[1,2,3*], **Bo Ma**[1,2,3],
Lei Wang[1,2,3], **Rui Dong**[1,2,3], **Azmat Anwar**[1,3]
[1]Xinjiang Technical Institute of Physics & Chemistry, Chinese Academy of Sciences, Urumqi, China
[2]University of Chinese Academy of Sciences, Beijing, China
[3]Xinjiang Laboratory of Minority Speech and Language Information Processing, Urumqi, China
{wang_zhen,yangyt,zhouxi,mabo,wanglei,dongrui,azmat}@ms.xjb.ac.cn

Abstract

Prompt-based learning, which exploits knowledge from pre-trained language models by providing textual prompts and designing appropriate answer-category mapping methods, has achieved impressive successes on few-shot text classification and natural language inference (NLI). Because of the diverse linguistic expression, there exist many answer tokens for the same category. However, both manual answer design and automatic answer search constrain answer space and therefore hardly achieve ideal performance. To address this issue, we propose an answer space clustered prompting model (ASCM) together with a synonym initialization method (SI) which automatically categorizes all answer tokens in a semantic-clustered embedding space. We also propose a stable semi-supervised method named stair learning (SL) that orderly distills knowledge from better models to weaker models. Extensive experiments demonstrate that our ASCM+SL significantly outperforms existing state-of-the-art techniques in few-shot settings.[1]

1 Introduction

Pre-trained language models (PLMs, Vaswani et al., 2017; Devlin et al., 2019; Qiu et al., 2020; Lewis et al., 2020; Clark et al., 2020) have shown a great impact on natural language processing (NLP) tasks. By adding task-specific head and fine-tuning on labeled corpora, PLMs surpass conventional fully supervised learning paradigm and come into being a "pre-train, fine-tune" paradigm (Sun et al., 2019).

However, Radford et al. (2019) demonstrate PLMs can perform downstream tasks without any additional data and modification, which reveals PLMs have the potential for knowledge exploration.

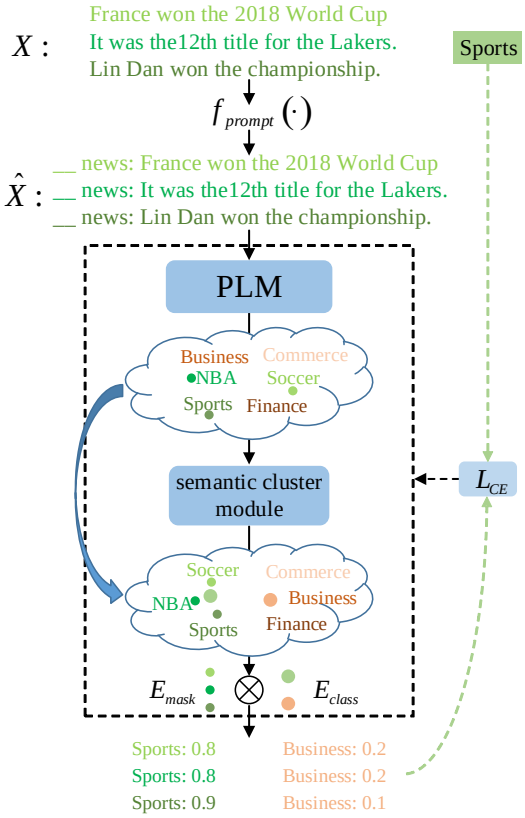

Figure 1: Illustration of ASCM. Given textual input x, ASCM adds task-specific prompt pattern to X and predicts masked token embedding by PLM encoder. Then semantic cluster module (SCM) transforms token embedding to a semantic-clustered embedding space. Finally, similarities between E_{mask} and categorized cluster centers decide the category of x.

And Petroni et al. (2019) find that BERT contains relational knowledge, factual knowledge and can be applied to QA tasks without fine-tuning.

Recently, prompting methods (Liu et al., 2021), which reformulate downstream tasks with task-specific textual prompts, is proved successful in

* Corresponding author
[1]Our implementation is publicly available at https://github.com/miaomiao1215/ASCM.

Findings of the Association for Computational Linguistics: ACL 2022, pages 2455 - 2469
May 22-27, 2022

many tasks such as few-shot text classification and natural language inference. For example, to classify textual news such as *"France won the 2018 World Cup"*, prompting methods may add the textual prompt *"__ news:"* before the news, then PLMs probably fill in the blank with token *"soccer"*, *"sports"*, *"football"*, *"match"*, *"FIFA"*, etc. Taking these tokens as the answer space of *sports* news category, PLMs may correctly predict the category of news even without fine-tuning. Nevertheless, former prompting methods bring in extra prompt engineering and answer engineering, which have significant influences on performance and need to be designed carefully. There exist various prompt engineering methods (Davison et al., 2019; Wallace et al., 2019; Haviv et al., 2021; Ben-David et al., 2021; Li and Liang, 2021), but answer engineering hasn't been researched enough. Former answer engineering can be categorized into manual answer design and automatic answer search. Both methods select limited answers space for each category and force PLMs to predict in that answer space. For example, former methods may force PLMs to fill the blank of *"__ news: It was the 12th title for the lakers."* with *"Sports"* if the answer space of *sports* category doesn't include *"NBA"*. Besides, manual answer design methods need additional expertise and automatic answer search methods may damage the performance (Schick et al., 2020; Schick and Schütze, 2021; Gao et al., 2021).

We find that answer tokens belonging to the same category get some kind of relationship. For sentiment analysis tasks and natural language inference tasks, same-category answer tokens usually get similar semantics (*glad*, *happy*). For topic classification tasks, same-category answer tokens may get relationships such as synonym (*soccer*, *football*), hyponym(*soccer*, *football*), hypernym, co-hyponym, etc. For convenience, we adopt *"synonym"* and *"semantic"* to represent all the relationships above. But the distribution of token embedding in PLMs isn't specially designed for text classification or NLI. It came to us that cluster centers of intra-class answer tokens can be used for classification if all token embedding distributes according to semantics. In that case, any tokens that are relevant to certain categories will get close to the corresponding cluster center and be automatically included in the corresponding answer space, which means no constraint on PLMs and no answer engineering. Following this idea, we propose the ASCM, as illustrated in Figure 1, which focuses on text classification and natural language inference. Our contributions can be summarized as follows:

- We propose ASCM that transforms token embeddings to a semantic-clustered embedding space and categorizes all answer tokens embeddings in that space. ASCM puts no constraint on answer space and doesn't need answer engineering or expertise.
- We propose a synonym initialization method for additional parameters introduced by ASCM, which makes ASCM competitive in few-shot settings.
- Besides, to exploit massive unlabeled data, we propose a semi-supervised method called stair learning (SL) which transfers knowledge orderly and further increases the performance.

We conduct extensive experiments which demonstrate the superiority of our method. Our ASCM+SL outperforms the previous prompt-based learning (manual answer design) by 10.3, 2.6, 2.3, and 2.1 on MNLI, Yahoo, Yelp, and AG's News with 50 labeled examples.

2 Related Work

2.1 Prompt-based Learning

Schick and Schütze (2021) propose to reformulate input examples into cloze-style phrases and show superiority in few-shot text classification and natural language inference. Gao et al. (2021) further propose to use T5 to automatically generate prompt patterns, which improve performance and makes minimal assumptions on domain expertise. Lester et al. (2021) propose prompt tuning to learn soft prompts with PLMs parameter frozen, which attain comparable performance with model tuning. Prompt-based learning has also been applied to knowledge probing(Ettinger, 2020; Jiang et al., 2020a,c), text generation (Brown et al., 2020; Schick and Schütze, 2020; Dou et al., 2021), machine translation (Radford et al., 2019), question answering (Khashabi et al., 2020; Jiang et al., 2020b) and information extraction (Shin et al., 2021; Cui et al., 2021; Chen et al., 2021).

2.2 Answer Engineering

Answer engineering aims to design appropriate answer space and map function to transform predictions of the masked token to task-specific results.

Answer space is often unconstrained in text generation and machine translation, while constrained in text classification and natural language inference tasks which this work focuses on.

Yin et al. (2019) manually select words for each label in topic classification, emotion classification, and situation classification task. Similar manual answer engineering also can be found in other works such as Schick and Schütze (2021). Manual answer engineering needs extra expert knowledge and is hardly convinced to be optimal due to limited answer space.

Jiang et al. (2020b) take back-translation (Sennrich et al., 2016) as the paraphrasing method to expand initial answer space and the prediction probability is the sum of category-specific probabilities over expanded answer space. Schick et al. (2020) propose a likelihood ratio verbalizer search which selects several proper tokens for each category according to their probability distributions. However, experiments show that handcrafted verbalizers still perform better than their automatic verbalizer search. Gao et al. (2021) automatically select top 30 tokens per class by simplifying and adding re-ranking to the method in Schick and Schütze (2021), which reach comparable performance with manual designed answer space. The aforementioned answer engineering methods can be categorized into the discrete answer search, where answer space is a small subset of the token space of PLMs. Hambardzumyan et al. (2021) explore to use continuous embedding called soft labels, which doesn't need answer engineering. However, some virtual answer embeddings lack of interpretability and this method still constrains answer space. Because tokens embeddings belonging to the same categories such as *"sports"*, *"soccer"* and *"football"* are dispersive in PLMs, virtual answer embedding belonging to *sports* category cannot fit all token embeddings above.

2.3 Semi-supervised Learning

Chen et al. (2020) create augmented training examples by interpolating text in hidden space and predict combined low-entropy labels. Xie et al. (2020) propose to combine advanced data augmentation methods such as RandAugment and back-translation with a consistency training framework. Schick and Schütze (2021) propose a semi-supervised learning method called iPET to iteratively distill knowledge and exploit unlabeled data with size gradually increasing. The iPET improves model performance further but the learning procedure is random which means the teacher model might be too weak for the student model.

3 Our Method

3.1 ASCM

Notice that the following discussions focus on text classification and natural language inference. Let M be a pre-trained language model, V its token vocabulary, $\_\_ \in V$ the mask token, $x \in X$ the token sequences to be predicted, and $y \in Y$ the corresponding ground-truth label. Prompting methods reformulate input text x to $\hat{x}$ with task-specific prompting functions $f_{prompt}(\cdot)$, in which mask token $\_\_$ is inserted. With proper prompting function, M is likely to fill $\hat{x}$ with a specific token at the masked position, which is helpful to downstream tasks. Former prompt-based learning usually designs a small answer space $\hat{V} \in V$ categorized by Y. Taking task AG's News as an example, $\hat{V}$ can be {["World"], ["Sports", "Soccer"], ["Business", "Commerce"], ["Tech"]} for labels {"World", "Sports", "Business", "Science/Technology"}. Let $E_{\hat{V}}$ be token embeddings corresponding to answer space $\hat{V}$ and E_{mask} be token embedding at mask position predicted from M. Then similarities between E_{mask} and $E_{\hat{V}}$ denote the probability distribution over labels. Hambardzumyan et al. (2021) propose to use soft continuous embedding E_{V_soft}, which are regarded as virtual answer space.

In this work, we propose ASCM that consists of PLMs encoder, a semantic cluster module (SCM, composed of a linear transformation layer, a BN layer, and a tanh activation function), and a semantic classifier (SC). The PLMs encoder predicts E_{mask} as former prompt-based methods does and then SCM and SC together classify on E_{mask}. SCM transforms E_{mask} to a virtual token embedding on another embedding space, where token embeddings are optimized to cluster according to semantics. With intra-class tokens such as {"sports", "soccer", "football" ...} converging to a cluster center and cluster centers of different categories diverging, SC predicts on this virtual token embedding according to the similarities with all categorized cluster centers. After SCM, because tokens that are relevant to certain categories will get close to the corresponding cluster center and automatically be included in the corresponding answer

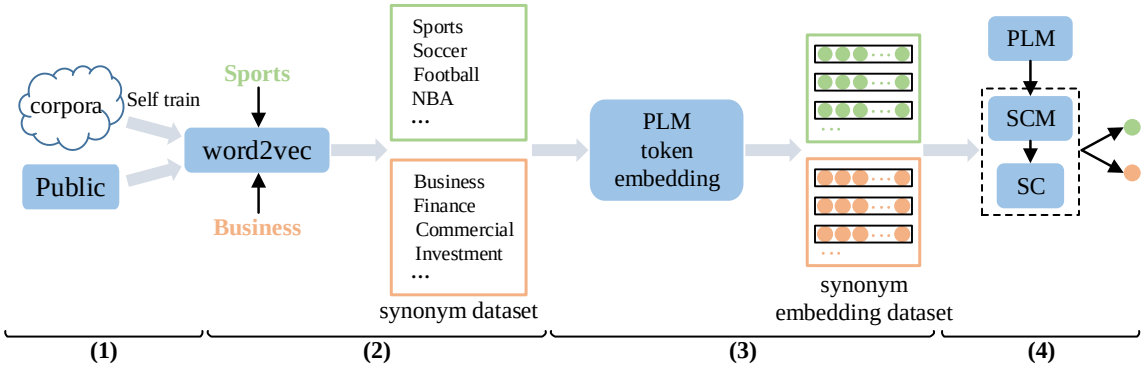

Figure 2: Illustration of synonym initialization.

space, ASCM can predict based on unconstrained answer tokens space leading to better performance.

As discussed above, PLM predicts masked token embeddings for SCM, and therefore keeping the performance for cloze question is important. We compute both cross-entropy loss L_{CE} for classification and auxiliary language modeling loss L_{MLM} (Chronopoulou et al., 2019). The final loss is as

$$L = \alpha \cdot L_{CE} + (1 - \alpha) \cdot L_{MLM} \quad (1)$$

3.2 Synonym Initialization

Previous prompt-based learning methods require answer space engineering for $E_{\hat{V}}$, which is one kind of model initialization. ASCM needs no answer space engineering but introduces additional SCM and SC to be learned and therefore is especially hard to be fine-tuned in a few-shot task.

As discussed in Section 3.1, SCM and SC classify E_{mask} according to semantics. Therefore, establishing a synonym embedding dataset and pre-training SCM and SC on this dataset shall be a reasonable solution. In this section, a synonym initialization method for SCM and SC will be introduced to address the issue above.

The synonym initialization method can be divided into four steps(as shown in Figure 2).

1) We need a words classification method or model, such as Glove (Pennington et al., 2014) and word2vec (Mikolov et al., 2013a,b). Word2vec can explore the semantics and potential relationships of words and therefore is adopted in this work. In the first step, a word2vec model trained on a task-specific dataset by self-supervised learning or a public pre-trained word2vec model is adopted.

2) We use the similarity scores of word2vec word embeddings to select the top-100 synonyms for each category and filter those with scores lower than 0.6. If a word belongs to multiple categorized synonym sets, then it will be classified to the category with the highest similarity score.

3) All words in the synonym dataset are tokenized and the first token of multi-token words is reserved. Then the token decoder (embedding) layer of PLMs maps the synonym dataset to the synonym embedding dataset.

4) Finally, SCM and SC are pre-trained on the categorized synonym embedding dataset and the parameters will be used to initialize the ASCM.

It is notable that the synonym initialization method needs no expertise.

3.3 Stair Learning

Given different prompt patterns, PLMs usually result in different performances on corpora. Knowledge distilling (Hinton et al., 2015) is a common solution for model compression, which can transfer knowledge from a teacher model to another smaller model. It gives us a hint that we can transfer knowledge from better ASCMs to weaker ASCMs. Accordingly, we propose an orderly stair learning method (SL) to transfer knowledge between ASCMs with different prompt patterns. In each retraining round k, SL exploits the unlabeled dataset and gradually multiplies the size of unlabeled examples by a constant d_0.

Let n be number of prompt patterns, T be labeled dataset, D be unlabeled dataset, M^0 =

$\left\{M_1^0, M_2^0 \cdots M_n^0\right\}$ be initial ASCMs trained on T, $M^j = \left\{M_1^j, M_2^j \cdots M_n^j\right\}$ be ASCMs in the retraining round j, M_i^j be ASCM with prompt pattern i in round j and d_j be the number of unlabeled examples in round j.

$$d_j = \begin{cases} |T| * d_0, & j = 1 \\ \\ d_{j-1} * d_0, & j \neq 1 \end{cases} \tag{2}$$

All ASCMs are retrained with several rounds in SL and each ASCM will be retrained the same times per round. The procedure for each retraining rounds in SL can be divided into five steps as follows.

1) We select the worst and no-retrained model of M^{j-1} as student model $M_{i'}^j$.

2) Best Model of last rounds $M^{j-1} = \left\{M_1^{j-1}, M_2^{j-1} \cdots M_n^{j-1}\right\}$ together with models that have been fine-tuned in this round $\left\{M_{1'}^j \cdots\right\}$ will be formed together as teacher model set M_t^j. The best model of M_t^j is selected as the teacher model $M_{t\_i'}$. If prompt pattern of $M_{t\_i'}$ is the same as $M_{i'}^j$, then the second best model of M_t^j will be selected as teacher model.

3) We evaluate $M_{t\_i'}$ on D and categorize D by the predicted labels. Then we randomly sample d_j examples from D with labels distributing uniformly. To reduce the mislabeled examples, we sample examples according to the confidence of predicted labels (Guo et al., 2017; Schick and Schütze, 2021).

4) We retrain student model $M_{i'}^j$ on L and unlabeled dataset from (3) with cross-entropy loss. After fine-tuning, $M_{i'}^j$ is added to M_t^j, which makes it possible to transfer its knowledge to other model in this round if $M_{i'}^j$ outperform $M_{t\_i'}$.

5) We repeat steps (1)-(4) until all ASCMs are fine-tuned in this round and then restart the next retraining round.

After retraining all ASCMs with the same rounds, unlabeled dataset D will be annotated by final-round ASCMs and the average probability distribution forms the soft-labeled dataset. Finally, using KL divergence loss with a temperature of 2, we fine-tune a PLM with a standard sequence classification head on this soft-labeled dataset.

4 Experiments

Following prior work, we evaluate our work on four tasks Yelp Reviews, AG's News, Yahoo Questions (Zhang et al., 2015), and MNLI (Williams et al., 2018). For comparison, we adopt RoBERTa large (Liu et al., 2019) as the pre-trained language model in all experiments except for Table 2.

To evaluate the few-shot performances, we randomly sample $|T|$ (10, 50, 100, and 1000) examples as the labeled dataset with labels distributing uniformly. And we randomly sample 10000 examples for each label to form the unlabeled dataset D for SL.

We choose the Adam optimizer with a slanted triangular schedule, an initial learning rate of 1e-5, and a maximum sequence length of 256. The batch size is set to 16 for $|T|$ equals to 50, 100, 1000 and 8 when $|T|$ equals to 10. For each training step, we randomly sample the same number of examples from D to compute auxiliary language modeling loss and the loss weight α is set to 0.5. For supervised training and individual SL, training steps are set to 300. For the final PLM classifier, training steps are set to 5000. For SL, we set $d = 5$, $k = \log_d(1000/|T|)$ and only train once for each SL round to reduce computing time. Notably, iPET trains three times for each model and the ensemble of them will improve the performance.

Training details of synonym initialization can be found in appendix A.

4.1 Prompt Pattern

In this work, we take the manual prompt engineering method to design prompt patterns for each task. Two vertical bars (‖) are used to mark boundaries between text segments.

Yelp The Yelp reviews full star task is to estimate the restaurant rating (1 to 5 stars) of customers based on their review's text. We define 4 prompt patterns for an input text x:
$f_p^1 = It\ was\ \_\_.\ x$ $\quad$ $f_p^2 = Just\ \_\_!\ \|\ x$
$f_p^3 = x.\ All\ in\ all,\ it\ was\ \_\_.$
$f_p^4 = x\ \|\ In\ summary,\ the\ restaurant\ is\ \_\_.$

AG's News The AG's News task is to classify textual news into one of the four categories *World*,

Examples	Method	Yelp	AG's	Yahoo	MNLI(m/mm)
$\|T\| = 10$	supervised	21.1 ±1.6	25.0 ±1.6	10.1 ±0.1	34.2 ±2.1 / 34.1 ±2.0
	PET	52.9 ±0.1	87.5 ±0.0	63.8 ±0.2	41.8 ±0.1 / 41.5 ±0.2
	iPET	57.6 ±0.0	89.3 ±0.1	**70.7** ±0.1	43.2 ±0.0 / 45.7 ±0.1
	ASCM+PET	56.7 ±2.6	83.9 ±3.4	64.7 ±1.1	51.3 ±5.2 / 54.9 ±8.0
	ASCM+SL	**62.9** ±0.7	**90.3** ±0.3	70.4 ±3.3	**64.6** ±6.2 **/ 65.0** ±11.9
$\|T\| = 50$	supervised	44.8 ±2.7	82.1 ±2.5	52.5 ±3.1	45.6 ±2.1 / 47.6 ±2.0
	PET	60.0 ±0.1	86.3 ±0.0	66.2 ±0.1	63.9 ±0.0 / 64.2 ±0.0
	iPET	60.7 ±0.1	88.4 ±0.1	69.7 ±0.1	67.4 ±0.3 / 68.3 ±0.3
	ASCM+PET	62.7 ±1.2	89.0 ±0.3	69.9 ±0.6	72.9 ±2.3 / 74.5 ±0.7
	ASCM+SL	**63.0** ±1.0	**90.5** ±0.3	**72.3** ±0.4	**77.6** ±0.8 **/ 78.6** ±0.5
$\|T\| = 100$	supervised	53.0 ±3.1	86.0 ±0.7	62.9 ±0.9	47.9 ±2.8 / 51.2 ±2.6
	PET	61.9 ±0.0	88.3 ±0.1	69.2 ±0.0	74.7 ±0.3 / 75.9 ±0.4
	iPET	62.9 ±0.0	89.6 ±0.1	71.2 ±0.1	78.4 ±0.7 / 78.6 ±0.5
	ASCM+PET	**64.2** ±0.5	89.5 ±0.6	69.2 ±1.2	75.2 ±5.4 / 76.1 ±5.0
	ASCM+SL	63.8 ±0.1	**90.7** ±0.4	**72.0** ±0.5	**80.7** ±0.8 **/ 81.5** ±0.9
$\|T\| = 1000$	supervised	63.0 ±0.5	86.9 ±0.4	70.5 ±0.3	73.1 ±0.2 / 74.8 ±0.3
	PET	64.8 ±0.1	86.9 ±0.2	72.7 ±0.0	**85.3** ±0.2 **/ 85.5** ±0.4
	ASCM+PET	**65.7** ±0.1	**91.4** ±0.1	**73.9** ±0.2	83.2 ±2.7 / 83.7 ±2.8

Table 1: Average accuracies and standard deviation of different methods on Yelp, AG's News, Yahoo, and MNLI (m: matched/mm: mismatched) for four training set sizes $|T|$.

Sports, *Business* and *Science/Technology*. Each news contains a headline a and a text body b. We define 6 prompt patterns for an input text x:

$f_p^1 = \_\_ : \ a \ b$ $\quad$ $f_p^2 = a \ (\_\_) \ b$

$f_p^3 = \_\_ \ - \ a \ b$ $\quad$ $f_p^4 = a \ b \ (\_\_)$

$f_p^5 = \_\_ \ News : \ a \ b$

$f_p^6 = [Category : \ \_\_] \ a \ b$

Yahoo The Yahoo Questions task is to classify text to one of the ten categories *Society*, *Science*, *Health*, *Education*, *Computer*, *Sports*, *Business*, *Entertainment*, *Relationship* and *Politics*. Each news contains a question a and an answer b. We use the same prompt patterns as for AG's News.

MNLI The MNLI task is a natural language inference task that is to estimate the relationships of text pairs $(a, \ b)$. MNLI contains three categories *contradiction*, *entailment* and *Neutral*. We define 2 prompt patterns:

$f_p^1 = "a"? \ \| \ \_\_, "b"$ $\quad$ $f_p^2 = a? \ \| \ \_\_, \ b$

4.2 Results

Table 1 shows the results of our method on different tasks. We also include the supervised method, current state-of-the-art method PET, and iPET for comparison. Mean accuracy and standard deviation for three training runs are adopted as measurements. Notably, results of the supervised, PET, and iPET method in Table 1 come from Schick and Schütze (2021).

ASCM significantly outperforms the supervised method on all configurations, especially on smaller $|T|$. The difference between ASCM+PET and PET is the base model. ASCM+PET surpasses PET on most tasks because ASCM gets better performance than conventional prompt-based learning. What's more, on several tasks, ASCM+PET even performs better than iPET, which additionally retrains models on unlabeled dataset iteratively. For example, ASCM+PET outperforms iPET by 8.1 on MNLI with $|T| = 10$, by 5.5 on MNLI with $|T| = 50$, and by 2.0 on Yelp with $|T| = 50$.

By retraining ASCM with SL, especially on smaller $|T|$, ASCM+SL gives further consistent improvements compared to ASCM+PET. ASCM+SL attains the state-of-the-art on most tasks. On MNLI, Yelp, AG's, and Yahoo, the average increments of accuracy come to 8.0, 2.4, 2.2, and 1.1. On MNLI with $|T| = 10$, ASCM+SL even surpasses iPET by 21.4. We also find that the standard deviations of ASCM+PET and ASCM+SL are much bigger than PET and iPET. It is because that Schick and Schütze (2021) train each model three times and train the final PLM classifier on $3n$ models (n prompt patterns) for three rounds. This ensemble

Ex.	Method	Yelp	AG's	Yahoo	MNLI
10	UDA	27.3	72.6	36.7	34.7
	MixText	20.4	81.1	20.6	32.9
	iPET	52.9	87.5	67.0	42.1
	Ours	**55.6**	**89.0**	**70.3**	**42.8**
50	UDA	46.6	83.0	60.2	40.8
	MixText	61.3	84.8	61.5	34.8
	iPET	56.7	87.3	66.4	56.3
	Ours	**59.1**	**89.9**	**70.1**	**61.3**

Table 2: Accuracy comparison of ASCM+SL with other semi-supervised methods using RoBERTa (base).

learning eventually will improve stability and performance. In this work, we create three different datasets for each task (dataset and $|T|$) and retrain ASCM once in each SL round. If we retrain ASCM three times in each SL round, ASCM+SL shall get better results.

We further compare our works with other semi-supervised methods such as UDA and MixText. We take RoBERTa(base) as PLM and keep other hyper-parameters. Table 2 shows that ASCM+SL outperforms other methods even with smaller PLM.

5 Analysis

5.1 ASCM

ASCM needs no answer engineering and shows better performance than conventional prompt-based learning methods based on manual answer design.

As Table 3 shows, we compare ASCM with conventional prompt-based learning baseline which needs expertise to carefully design answer space. We train ASCM and baseline with the same hyper-parameters and report the average accuracies on all prompt patterns. With $|T| = 10$, ASCM significantly outperforms baseline by 7.9, 4.6, and 3.0 on Yahoo Yelp, and MNLI. With $|T| = 1000$, ASCM still attains better results on most tasks, showing the superiority of ASCM structure (semantic cluster then classification).

We also conduct ablation experiments by training ASCM without SI or SCM. Without SI, although we pre-train the SCM and SC with PLMs encoder frozen, accuracies of ASCM-noSI is lower than baseline by a large margin on Yahoo and AG's. Compared with ASCM-noSI, ASCM gets significant improvement on all datasets, which shows the necessity of synonym initialization in ASCM. Meanwhile, with size $|T|$ getting bigger, ASCM-noSI attains a comparable performance with ASCM which imply the ability of PLM to find appropriate answer space. A much larger decline in

Ex.	Method	Yelp	AG's	Yahoo	MNLI
10	Baseline	48.4	82.2	54.1	45.5
	w/o SI	47.1	74.6	53.6	44.8
	w/o SCM	45.3	68.0	31.0	36.6
	ASCM	**53.0**	**82.5**	**62.0**	**48.5**
50	Baseline	58.0	87.9	64.6	63.2
	w/o SI	59.7	87.6	62.0	63.7
	w/o SCM	60.2	84.9	68.0	63.8
	ASCM	**61.2**	**88.3**	**68.4**	**68.9**
100	Baseline	60.5	88.7	66.4	69.7
	w/o SI	62.4	89.2	67.1	71.9
	w/o SCM	60.2	85.9	68.5	36.9
	ASCM	**62.7**	**89.2**	**68.6**	**74.1**
1000	Baseline	64.2	90.8	71.6	**82.0**
	w/o SI	64.2	90.7	71.9	75.7
	w/o SCM	62.8	87.5	70.9	35.8
	ASCM	**64.8**	**91.1**	**73.3**	80.5

Table 3: Comparison of ASCM with baseline. Average accuracies on four tasks for four training set sizes $|T|$ are reported. Line w/o SI refers to ASCM trained without SI and Line w/o SCM refers to ASCM without SCM.

performance, compared with ASCM-noSI, is also found in ASCM-noSCM on most tasks, especially on smaller $|T|$ tasks and MNLI. We consider that the original distribution of PLM token embedding isn't suitable for downstream token classification and the SCM with SI eases the problem. Detailed analysis can be found in 5.2.

5.2 SCM and SI

ASCM uses SCM to transform token embeddings to a semantic-clustered embedding space and categorizes them on this space by SC. Besides, ASCM takes the synonym initialization method to initialize SCM and SC. In this section, we explore their mechanism.

We list part of the synonym dataset generated from word2vec according to similarities. As Table 4 shows, synonyms generated from word2vec mostly get similar semantics or certain relationship. However, there is also wrong word *"theworld"* and cognates such as *"sport"*, *"businesses"*, etc. These synonyms, probably because of the characteristics of word2vec, might damage the ASCM, but we still keep them to avoid additional expertise.

We also test ASCM on evaluating corpora and list the top-5 tokens predicted on masked position according to frequency. Tokens belonging to the same categories get similar semantics or certain relationship and the results are much better than that of word2vec. ASCM also finds potential relationships such as *"NFL"* and shows a different tendency of tokens such as *"Science"* even with

Model	Category	Top-1	Top-2	Top-3	Top-4	Top-5
word2vec	World	World	globe	theworld	country	continent
	Sports	Sports	sport	sporting	athletics	football
	Business	Business	businesses	business	company	entrepreneurial
	Technology	Technology	technologies	innovation	technol	innovation
ASCM	World	World	Foreign	Military	/	/
	Sports	Sports	Football	NFL	NBA	/
	Business	Business	Economic	Energy	Company	/
	Technology	Tech	Science	Space	/	/

Table 4: Top-5 synonyms of query words for each category on AG's according to word2vec embedding similarities and top-5 most frequently predicted tokens of ASCM at the masked position on AG's. Tokens with frequency less than 100 are filtered (/).

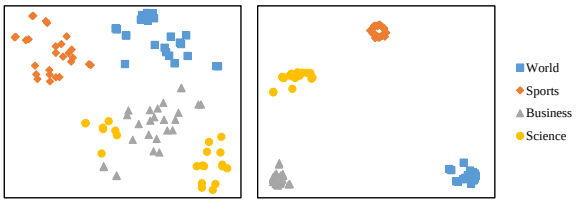

Figure 3: Distributions of original token embeddings (left) and token embeddings after SCM (Right) on AG-News.

Ex.	Method	Yelp	AG's	Yahoo	MNLI
10	w/o SI	48.4	82.2	54.1	45.5
	Skip-gram	46.0	78.1	35.4	43.8
	CBOW	50.2	**82.7**	60.4	**51.0**
	ASCM	**53.0**	82.5	**62.0**	48.5
	Union	51.8	82.5	62.2	44.6
	Intersection	50.4	80.7	59.2	40.8
50	w/o SI	59.7	87.6	62.0	63.7
	Skip-gram	59.7	87.6	62.0	63.8
	CBOW	60.3	88.5	67.6	**69.5**
	ASCM	**61.2**	88.3	**68.4**	68.9
	Union	60.5	88.1	67.4	66.1
	Intersection	59.8	**88.7**	68.2	42.2

Table 5: Comparison of ASCM with different generative approaches. Average accuracies on four tasks for $|T| = 10, 50$ are reported. ASCM refers to adopting public pre-trained word2vec model as the generative approach.

word2vec synonym initialization. As we consider, PLM we use gets different linguistic knowledge and factual knowledge with word2vec because of different pre-training task, corpora, and network. And much knowledge existing in PLM is kept successfully thanks to the ASCM.

We further filter the misclassified tokens in synonym token embedding datasets. Then, we use PCA and tSNE (Van der Maaten and Hinton, 2008) to visualize the distributions of token embeddings before and after SCM. As Figure 3 shows, original token embeddings distribute according to categories. However, intra-class distances are too large and there exist several fault cluster, leading to poor classification performance. And just as we designed, token embeddings after SCM cluster to several embedding centers and the inter-class distances are enlarged, which makes ASCM works better especially in a few-shot setting.

Results on other tasks can be found in Table 10, Table 11, and Figure 5.

5.3 Generative Approach of Synonyms Datasets

As shown in Figure 2, both public pre-trained models and models trained on task-specific datasets can be adopted to generate synonym datasets. In this section, we evaluate several approaches on four tasks with $|T| = 10, 50$.

As shown in Table 5, Skip-gram trained on task-specific datasets performs worse than ASCM-noSI, because of numerous misclassified words in synonym datasets. Public pre-trained word2vec (CBOW), which is better than CBOW trained on task-specific datasets on Yahoo and Yelp tasks but a bit worse on AG's and MNLI, is adopted as the basic approach (ASCM).

We also combine other methods such as public pre-trained FastText (Joulin et al., 2017) and Glove (Pennington et al., 2014) with word2vec. For similar reason to Skip-gram, the union set of the three synonyms datasets perform worse on most task. However, the intersection set with less misclassified words still gets worse, showing the necessity for the size of synonym dataset.

Ex.	Method	Yelp	AG's	Yahoo	MNLI
10	Baseline	63.0	88.3	71.1	63.3
	SL	**63.4**	**90.5**	**72.5**	**67.3**
50	Baseline	63.7	89.8	71.6	78.1
	SL	**64.2**	**90.8**	**72.5**	**78.4**
100	Baseline	**64.2**	90.0	71.3	79.7
	SL	63.9	**91.1**	**71.6**	**81.2**

Table 6: Accuracy comparison of SL and Baseline (iPET) method.

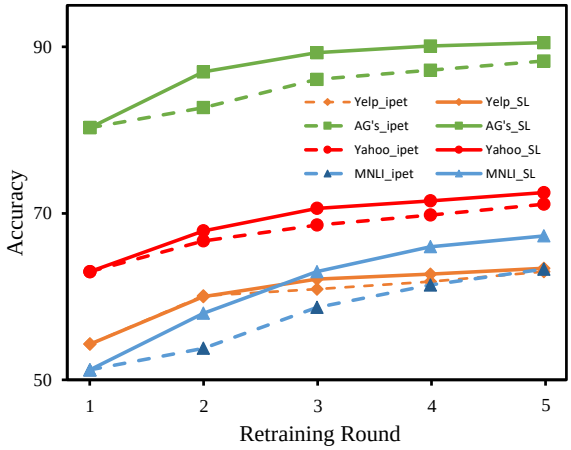

Figure 4: Average accuracy of ASCMs for each iPET and SL round with $|T| = 10$. Round 1 refers to the average accuracy of ASCMs trained on $|T|$ and round 5 refers to the training result for the final PLM classifier.

5.4 Stair Learning

We retrain ASCMs with iPET and SL on four datasets with $|T| = 10, 50, 100$. It's notable that we train ASCMs once for each iPET round. Base ASCMs and hyper-parameters are kept the same for comparison and results are reported in Table 6. ASCM+SL gets significant improvements than ASCM+iPET on most tasks especially when the size of labeled datasets is small. With $|T| = 10$, ASCM+SL outperforms ASCM+iPET by 4.0, 2.2, 1.4, and 0.4 on MNLI, AG's, Yahoo, and Yelp.

Average accuracies of all rounds with $|T| = 10$ are shown in Figure 4. The performances of iPET and SL keep improving in all rounds but the increments slow down with training rounds increasing. And SL gets larger increments because iPET distills knowledge from the randomly chosen models while SL distills knowledge from the best model of the round.

6 Conclusion

In conclusion, we propose an answer space clustered prompting model and a synonym initialization method that doesn't need answer engineering or expertise. Our method clusters token embeddings according to semantics and classifies them on unconstrained answer space. Experiments show that our method combined with a stable stair learning method outperforms the previous prompt-based learning methods based on manual answer design. Clustering multi-tokens words and phrases based on semantics is desirable for future work. In addition, research on adapting the thought of token embedding semantic-clustering to machine translation, text generation, information retrieval, and text summarization might also prove valuable.

7 Acknowledgement

This research is sponsored by the Natural Science Foundation for Distinguished Young Scholars of Xinjiang Uygur Autonomous Region (2022D01E04), the Youth Innovation Promotion Association of Chinese Academy of Sciences ([2019]26), the National Natural Science Foundation of China (U2003303), and the Tianshan Innovative Research Team of Xinjiang (2020D14045).

References

Eyal Ben-David, Nadav Oved, and Roi Reichart. 2021. PADA: A prompt-based autoregressive approach for adaptation to unseen domains. *CoRR*, abs/2102.12206.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Jiaao Chen, Zichao Yang, and Diyi Yang. 2020. Mixtext: Linguistically-informed interpolation of hidden space for semi-supervised text classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 2147–2157. Association for Computational Linguistics.

Xiang Chen, Xin Xie, Ningyu Zhang, Jiahuan Yan, Shumin Deng, Chuanqi Tan, Fei Huang, Luo Si, and

Huajun Chen. 2021. Adaprompt: Adaptive prompt-based finetuning for relation extraction. *arXiv preprint arXiv:2104.07650*.

Alexandra Chronopoulou, Christos Baziotis, and Alexandros Potamianos. 2019. An embarrassingly simple approach for transfer learning from pretrained language models. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 2089–2095. Association for Computational Linguistics.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: pre-training text encoders as discriminators rather than generators. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Leyang Cui, Yu Wu, Jian Liu, Sen Yang, and Yue Zhang. 2021. Template-based named entity recognition using BART. In *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021*, volume ACL/IJCNLP 2021 of *Findings of ACL*, pages 1835–1845. Association for Computational Linguistics.

Joe Davison, Joshua Feldman, and Alexander M. Rush. 2019. Commonsense knowledge mining from pretrained models. 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 2019, Hong Kong, China, November 3-7, 2019*, pages 1173–1178. Association for Computational Linguistics.

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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Zi-Yi Dou, Pengfei Liu, Hiroaki Hayashi, Zhengbao Jiang, and Graham Neubig. 2021. Gsum: A general framework for guided neural abstractive summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 4830–4842. Association for Computational Linguistics.

Allyson Ettinger. 2020. What BERT is not: Lessons from a new suite of psycholinguistic diagnostics for language models. *Trans. Assoc. Comput. Linguistics*, 8:34–48.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 3816–3830. Association for Computational Linguistics.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. 2017. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1321–1330. PMLR.

Karen Hambardzumyan, Hrant Khachatrian, and Jonathan May. 2021. WARP: word-level adversarial reprogramming. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 4921–4933. Association for Computational Linguistics.

Adi Haviv, Jonathan Berant, and Amir Globerson. 2021. Bertese: Learning to speak to BERT. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021*, pages 3618–3623. Association for Computational Linguistics.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531.

Zhengbao Jiang, Antonios Anastasopoulos, Jun Araki, Haibo Ding, and Graham Neubig. 2020a. X-FACTR: multilingual factual knowledge retrieval from pretrained language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 5943–5959. Association for Computational Linguistics.

Zhengbao Jiang, Jun Araki, Haibo Ding, and Graham Neubig. 2020b. How can we know when language models know? *CoRR*, abs/2012.00955.

Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020c. How can we know what language models know. *Trans. Assoc. Comput. Linguistics*, 8:423–438.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomás Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 2: Short Papers*, pages 427–431. Association for Computational Linguistics.

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, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pages 1896–1907. Association for Computational Linguistics.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 3045–3059. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7871–7880. Association for Computational Linguistics.

Xiang Lisa Li and Percy Liang. 2021. 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, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 4582–4597. Association for Computational Linguistics.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *CoRR*, abs/2107.13586.

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

Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*.

Tomás Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 3111–3119.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543. ACL.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick S. H. Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander H. 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 2019, Hong Kong, China, November 3-7, 2019*, pages 2463–2473. Association for Computational Linguistics.

Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. 2020. Pre-trained models for natural language processing: A survey. *CoRR*, abs/2003.08271.

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

Timo Schick, Helmut Schmid, and Hinrich Schütze. 2020. Automatically identifying words that can serve as labels for few-shot text classification. In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 5569–5578. International Committee on Computational Linguistics.

Timo Schick and Hinrich Schütze. 2020. Few-shot text generation with pattern-exploiting training. *CoRR*, abs/2012.11926.

Timo Schick and Hinrich Schütze. 2021. 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, EACL 2021, Online, April 19 - 23, 2021*, pages 255–269. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics.

Richard Shin, Christopher H. Lin, Sam Thomson, Charles Chen, Subhro Roy, Emmanouil Antonios Platanios, Adam Pauls, Dan Klein, Jason Eisner, and Benjamin Van Durme. 2021. Constrained language models yield few-shot semantic parsers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 7699–7715. Association for Computational Linguistics.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune BERT for text classification? In *Chinese Computational Linguistics - 18th China National Conference, CCL 2019, Kunming, China, October 18-20, 2019, Proceedings*, volume 11856 of *Lecture Notes in Computer Science*, pages 194–206. Springer.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing NLP. 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 2019, Hong Kong, China, November 3-7, 2019*, pages 2153–2162. Association for Computational Linguistics.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics.

Qizhe Xie, Zihang Dai, Eduard H. Hovy, Thang Luong, and Quoc Le. 2020. Unsupervised data augmentation for consistency training. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Wenpeng Yin, Jamaal Hay, and Dan Roth. 2019. Benchmarking zero-shot text classification: Datasets, evaluation and entailment approach. 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 2019, Hong Kong, China, November 3-7, 2019*, pages 3912–3921. Association for Computational Linguistics.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 649–657.

Ex.	Yelp	AG's	Yahoo	MNLI
10	91.2	88.1	92.8	50.0
50	91.7	87.6	92.5	51.3
100	92.1	90.6	92.9	49.7
1000	92.1	88.7	93.4	51.3

Table 7: Average word classification accuracy for all ASCMs on categorized token embedding datasets.

Category	Synonyms
Contradiction	No, But, However, Instead, Yet, Actually, Whereas, Nevertheless
Entailment	Yes, Uh, yep, So, Therefore, consequently
Neutral	Maybe, probably, Further, Also, Neutral, perhaps, possibly

Table 8: Manually designed synonym dataset for MNLI.

A Synonym Initialization Details

For the synonyms generation models trained on task-specific datasets, we adopt the Genism library and the default training setting. Part of the synonym datasets generated by public pre-trained word2vec is listed in Table 10.

For fine-tuning of SCM and SC, we choose the Adam optimizer with a slanted triangular schedule with an initial learning rate of 1e-5, and a weight decay of 0.01. The batch size is set to 16 and the training epochs are set to 40. And we choose the model with the highest classification accuracy on the training synonym dataset to initialize ASCM.

B SCM and SI

We list the top-5 predicted tokens on masked position according to frequency by testing ASCM on evaluation corpora (Table 11). Compared to Table 10, there are big changes in both words (tokens) and order of words (tokens), which is similar to the analysis in section 5.2.

For the distribution visualization of token embeddings, misclassified words are removed from the synonym token embedding dataset. Besides, if a word gets multiple tokens by tokenization and the first token occurs in other words, we also filter that kind of words. For example, token "base" is the first token of "base" and "baseball", which means ambiguity.

We further take categorized token embedding datasets as testing datasets and show the token classification accuracy of SCM and SC (Table 7). In

Method	10	50	100	1000
Auto	48.5	68.9	**74.1**	80.5
Manual	**58.7**	**71.1**	73.5	**81.4**

Table 9: ASCM accuracy on MNLI.

conformity to Figure 5, SCM and SC get high accuracy on Yelp, AG's, and Yahoo. For the MNLI task, ASCM only gets about 50% accuracy, because of the size (50) and poor quality of the MNLI synonym dataset.

Therefore, we manually designed a synonym dataset for MNLI task as shown in Table 8 and train ASCM based on it as shown in Table 9. Compared to automatically designed synonym dataset, there is a significantly increment with $|T| = 10$. It's encouraged to automatically generate a big synonym dataset at first and then do manually data cleaning.

C Training Details

All of our experiments are conducted using a single GPU with 32GB/16GB RAM (NVIDIA Tesla V100). Training a single PET with auxiliary language modeling for 300 steps on one GPU took approximately 20 minutes; retraining a single PET with auxiliary language modeling in SL for 300 steps on one GPU took approximately 20 minutes; Training a final PLM classifier for 5000 steps on one GPU took approximately 90 minutes. Labeling 10000 examples (per label) from D took approximately 13 minutes.

dataset	Category	Top-1	Top-2	Top-3	Top-4	Top-5
Yelp	☆	terrible	horrible	horrendous	dreadful	awful
	☆☆	bad	lousy	crummy	stupid	nasty
	☆☆☆	okay	alright	ok	OK	yeah
	☆☆☆☆	good	tough	Good	decent	nice
	☆☆☆☆☆	great	unbelievable	terrific	really	fantastic
Yahoo	Society	Society	societies	societal	polity	culture
	Science	Science	sciences	biology	scientific	mathematics
	Health	Health	Health	healthcare	wellness	wellbeing
	Education	Education	educational	curriculum	schooling	literacy
	Computer	Computer	computers	laptop	PC	laptops
	Sports	Sports	Sport	sporting	athletics	football
	Business	Business	businesses	business	businesss	company
	Entertainment	Entertainment	entertainment	music	amusements	multimedia
	Relationship	Relationship	relationships	friendship	ties	partnership
	Politics	Politics	discourse	political	politics	partisanship
MNLI	Contradiction	No	whatsoever	any	there	nothing
	Entailment	Yes	Uh	nope	/	/
	Neutal	Maybe	yeah	probably	suppose	hey

Table 10: Top-5 synonyms for each category in synonym dataset and query words is in "Top-1" column.

dataset	Category	Top-1	Top-2	Top-3	Top-4	Top-5
Yelp	☆	horrible	disgusting	terrible	HELL	disappointed
	☆☆	disappointing	blah	OK	bad	disappointed
	☆☆☆	OK	okay	ok	/	/
	☆☆☆☆	good	great	amazing	excellent	/
	☆☆☆☆☆	amazing	great	incredible	wonderful	fantastic
Yahoo	Society	Religion	Faith	Christianity	/	/
	Science	Science	Mathematics	Biology	Physics	Chemistry
	Health	Health	Sex	Nutrition	/	/
	Education	Education	History	Language	English	
	Computer	Computer	Internet	Software	IT	Technology
	Sports	Sports	Soccer	Football	Basketball	Baseball
	Business	Business	Finance	Money	Work	Employment
	Entertainment	Music	Entertainment	Movies	TV	/
	Relationship	Relationship	Dating	Sex	Family	Marriage
	Politics	Politics	Law	Military	History	Crime
MNLI	Contradiction	No	But	However	Or	Except
	Entailment	Yes	Indeed	/	/	/
	Neutal	Adding	Further	But	Or	/

Table 11: Top-5 most frequently predicted tokens of ASCM at masked position. Tokens with frequency less than 100 are filtered.

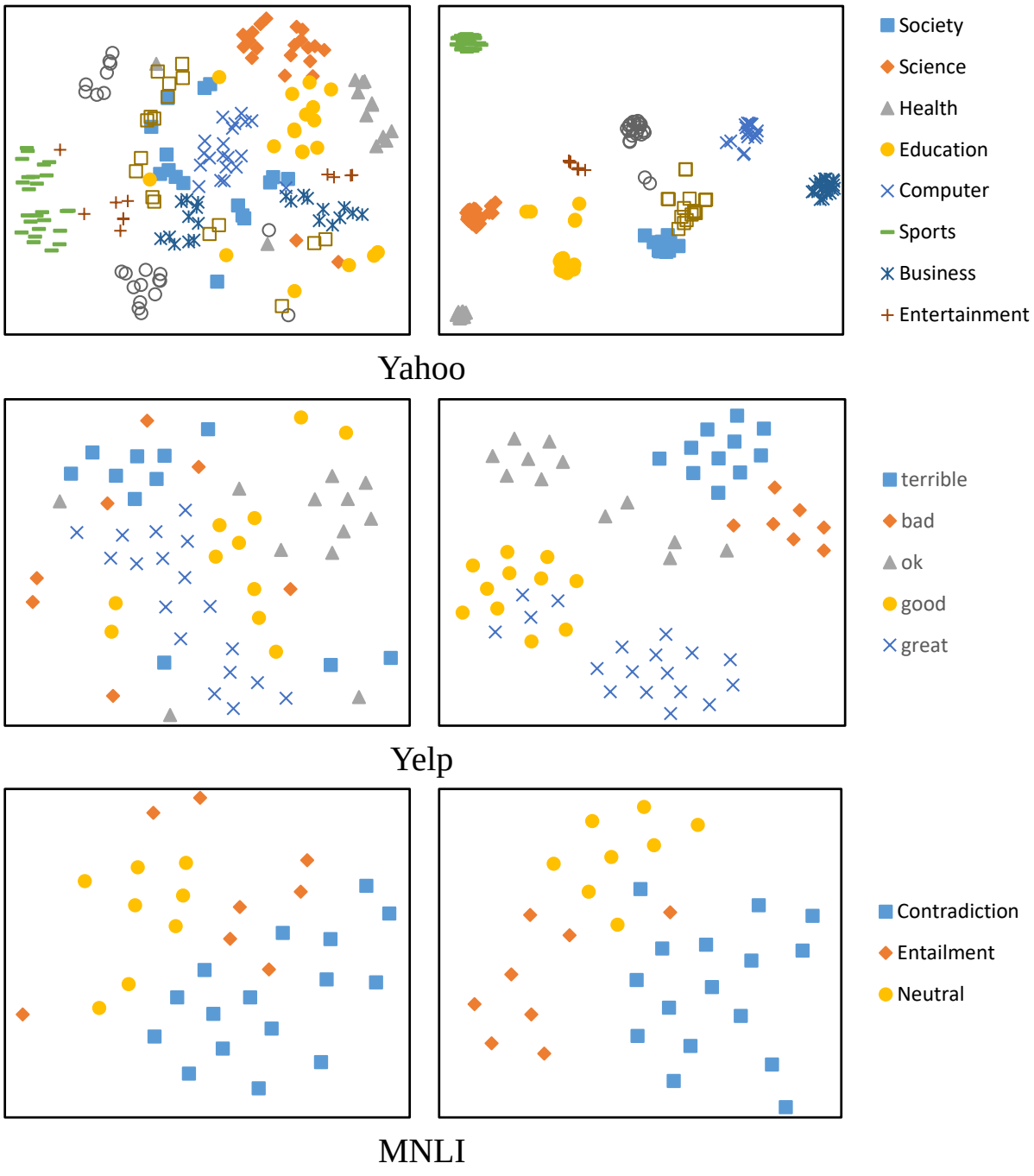

Figure 5: Distributions of original token embeddings (left) and token embeddings after SCM (Right) on Yahoo, Yelp, and MNLI.