Enhancing Aspect Term Extraction with Soft Prototypes

Zhuang Chen, Tieyun Qian*
School of Computer Science, Wuhan University, China
{zhchen18, qty}@whu.edu.cn

Abstract

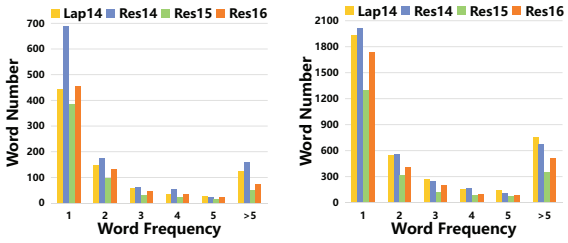
Aspect term extraction (ATE) aims to extract aspect terms from a review sentence that users have expressed opinions on. Existing studies mostly focus on designing neural sequence taggers to extract linguistic features from the token level. However, since the aspect terms and context words usually exhibit long-tail distributions, these taggers often converge to an inferior state without enough sample exposure. In this paper, we propose to tackle this problem by correlating words with each other through *soft prototypes*. These prototypes, generated by a soft retrieval process, can introduce global knowledge from internal or external data and serve as the supporting evidence for discovering the aspect terms. Our proposed model is a general framework and can be combined with almost all sequence taggers. Experiments on four SemEval datasets show that our model boosts the performance of three typical ATE methods by a large margin.

1 Introduction

Aspect term extraction (ATE) is a fundamental subtask in aspect-based sentiment analysis. Given a review sentence, ATE aims to extract all aspect terms that users have expressed opinions on. For example, from the review "*The Bombay style bhelpuri is very palatable.*", ATE aims to extract "*bhelpuri*".

ATE has been widely studied in the last twenty years. Early researches are devoted to design rule-based (Popescu and Etzioni, 2005) and feature engineering-based (Li et al., 2010) methods. With the development of deep learning techniques, recent researches mostly regard ATE as a sequence labeling task and focus on developing various types of neural models (Liu et al., 2015; Xu et al., 2018; Ma et al., 2019) to generate a tag sequence for the review.

*Corresponding author.



(a) Aspect terms. (b) Context words.

Figure 1: The distributions of aspect terms and context words in the training sets of four SemEval datasets.

Though achieving impressive progress, current sequence taggers mentioned still face a serious challenge: the taggers may converge to an inferior state due to the lack of samples for tail words. As shown in Figure 1, about 80% aspect terms and context words (i.e., non-aspect terms) appear no more than five times in the commonly-used SemEval datasets. Without enough sample exposure, neural models can hardly achieve an optimal performance (He et al., 2018; Chen and Qian, 2019).

To tackle this challenge, correlating samples with each other may offer helping hands. For example, if we correlate the rare aspect term "*bhelpuri*" with a frequent one like "*food*", there will be more abundant samples for "*bhelpuri*" than ever. The problem then becomes how to build such a token-level correlation. Retrieving synonyms is an intuitive approach to this problem, but it has two limitations. Firstly, synonyms only exist for a small number of words in the vocabulary. This will make the correlations incomplete. Though we can calculate the nearest neighbors for a certain word based on the pre-trained word embeddings, it is not guaranteed that they have a similar semantic meaning. Secondly, in ATE, the existence of an aspect term depends on whether there are opinions on it. That is to say, we need to build a dynamic correlation for a certain word based on its entire contexts rather than the word itself. Indeed, if the retrieval is con-

Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pages 2107–2117,
November 16–20, 2020.

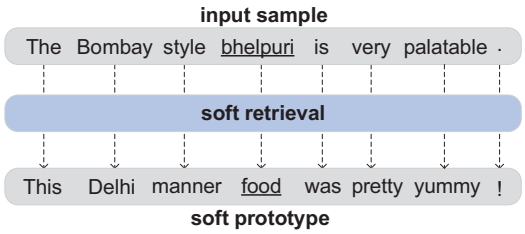


Figure 2: Process of the soft retrieval.

ducted based on an individual token, the above two limitations always exist.

In this paper, we propose a *soft* retrieval method to build the token-level correlation for both aspect terms and context words. Rather than conducting a *hard* retrieval for individual tokens, we turn to retrieve the tokens' counterparts according to their contexts. As shown in Figure 2, after conducting the soft retrieval, we can obtain a generated sample strictly corresponds to the input sample in every position. We name the generated sample "*soft prototype*" since it is actually a simplified prototype that can build a reference point for guiding the tagging process for the input sample.

We resort to the language models (LMs) to implement the soft retrieval and generate high-quality soft prototypes. As a self-supervised task, language modeling needs no extra annotations and can absorb data-specific global knowledge. Moreover, LMs tend to generate frequent outputs (Li et al., 2016), which exactly meets our needs for correlating a rare word in the input sample with a frequent one in the soft prototype. Specifically, we first pre-train bi-directional LMs using the given training samples on ATE datasets. Alternatively, we can take advantage of large-scale unlabeled data like Yelp and Amazon reviews to pre-train LMs. Then, after fixing the pre-trained LMs, we can infer each token's prototype according to its contexts for both the training and testing samples.

We regard the generated soft prototypes as the supporting evidence for tagging aspect terms, and design a simple and effective gating mechanism to fuse the knowledge embedded in both samples before sending them to a sequence tagger. The soft prototypes can be combined with almost all existing sequence taggers. To demonstrate the effectiveness of our proposed model, we conduct experiments on four SemEval datasets by adding the generated soft prototypes on three existing sequence taggers. The results prove that our soft prototypes significantly boost the performance of their original counterparts.

2 Related Work

Aspect Term Extraction Early researches for ATE mainly involve pre-defined rules (Hu and Liu, 2004; Popescu and Etzioni, 2005; Wu et al., 2009; Qiu et al., 2011) and hand-craft features (Li et al., 2010; Liu et al., 2012, 2013; Chen et al., 2014). With the development of deep learning techniques, neural methods have become the mainstream. ATE can be viewed as either a supervised or an unsupervised task. For unsupervised ATE, the commonly-used neural methods are based on topic models (He et al., 2017; Liao et al., 2019). For supervised ATE, the researchers focus on developing various types of neural sequence taggers (Liu et al., 2015; Wang et al., 2016; Yin et al., 2016; Wang et al., 2017; Li and Lam, 2017; Xu et al., 2018; Li et al., 2018; Ma et al., 2019). A recent trend is towards the unified framework (Wang et al., 2018; Li et al., 2019; Luo et al., 2019; He et al., 2019; Hu et al., 2019; Chen and Qian, 2020), where the interactive relations between ATE, opinion term extraction (OTE), or aspect-level sentiment classification (ASC) are exploited to enhance the overall performance. Xu et al. (2019) post-train BERT on domain-specific data to boost its sequence labeling performance. Li et al. (2020) propose to generate additional datasets for improving the performance of ATE.

In this paper, we focus on the supervised scenario. Different from the aforementioned supervised models, we develop a novel model to enhance ATE. By automatically generating and utilizing soft prototypes, we correlate samples with each other, which greatly enhances the learning process of sequence taggers. Moreover, the decoupling of soft prototypes from taggers makes our model flexible and general, i.e., it can be combined with almost all neural sequence taggers.

Prototypes in Neural Networks The idea of prototypes (or templates) originates from information retrieval (IR) approaches for sentence matching tasks like response generation (Ji et al., 2014; Hu et al., 2014). They aim to retrieve a related sample from the dataset as the counterpart of the input sample. More recently, several studies shed new light in this domain by deeply fusing prototypes with neural networks. Many of them use the task-dependent metrics (Guu et al., 2018; Hashimoto et al., 2018), common metrics such as Jaccard similarity (Gu et al., 2018; Cao et al., 2018; Wu et al., 2019), or existing tools like Lucene (Cao et al.,

2018) to retrieve prototypes, and then input the prototypes into a neural model for generating outputs. Wang et al. (2019) follows another line, where the prototype (the target words related to a source word in machine translation) is generated using a pre-trained Seq2Seq model.

The approach of generating words via LMs is inspired by a recent study (Kobayashi, 2018). However, the method in Kobayashi (2018) is developed for text classification and is not suitable for the ATE task here. Concretely, their method randomly replaces a small percentage (typically 10%) of original training words with the generated ones and then discard the original words. This operation may work well for text classification tasks which only require sentence-level information. For token-level tasks like ATE, the original words are however necessary for tagging each token correctly. Moreover, the small percentage of replacement implies that the generated knowledge cannot be fully incorporated into the new sample. In contrast, we generate a prototype for each word in the sentence, and then deeply fuse the original word with its corresponding prototype to make good use of their embedded knowledge for ATE.

To the best of our knowledge, we are the first to introduce the retrieval method to handle the data deficiency problem in ATE. To this end, we propose a new approach to generate and utilize soft prototypes that can build the token-level correlation for aspect terms and context words.

3 Methodology

In this section, we first illustrate the overall framework for enhancing ATE with soft prototypes. We then detail the generation and utilization of soft prototypes. Lastly, we describe the objective function and the training procedure.

3.1 The Overall SoftProto Framework

Aspect term extraction (ATE) aims to extract aspect terms from a review sentence that users have expressed opinions on. Given a sentence $S = \{w_1, w_2, ..., w_n\}$, we formulate ATE as a sequence labeling task that aims to predict a tag sequence $Y = \{y_1, y_2, ..., y_n\}$, i.e., learning the mapping $S \rightarrow Y$, where $y \in \{B, I, O\}$ denotes the *beginning of*, *inside of*, and *outside of* an aspect term.

To incorporate soft prototypes to ATE, we slightly modify the traditional learning process. Formally, rather than directly learning the mapping from S to Y, we additionally introduce a soft prototype $\mathbf{P}$ for each S and learn the new mapping $[S, \mathbf{P}] \rightarrow Y$. Given S, the soft prototype $\mathbf{P}$ is automatically generated by a soft retrieval mechanism, and can serve as the supporting evidence to discover the aspect terms. As shown in Figure 3, we summarize the above processes into the *SoftProto* framework that mainly consists of three modules:

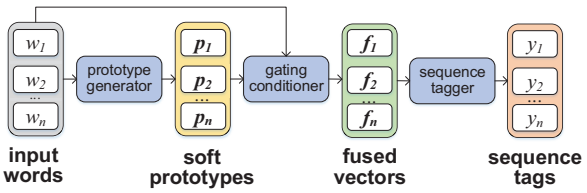


Figure 3: The SoftProto framework for enhancing ATE with soft prototypes.

- A *prototype generator* is used for conducting the soft retrieval process and generating the corresponding soft prototype $\mathbf{P}$ for S.
- A *gating conditioner* is used for merging S's representation and $\mathbf{P}$ into the fused vectors $\mathbf{F}$.
- A *sequence tagger* is used for predicting the tag sequence Y based on $\mathbf{F}$.

Next, we will illustrate each module in detail.

3.2 Prototype Generator

To efficiently implement the soft retrieval and generate high-quality soft prototypes, we resort to the language models (LMs) to build a prototype generator. Specifically, we first pre-train two LMs, where $\overrightarrow{LM}$ and $\overleftarrow{LM}$ is the *forward* and *backward* language model parameterized by $\overrightarrow{\theta_{LM}}$ and $\overleftarrow{\theta_{LM}}$, respectively. Then we infer soft prototypes based on the pre-trained LMs.

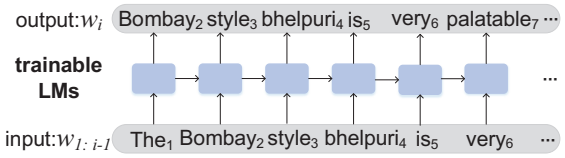
One can use either the ATE training set or other unlabeled external data like Yelp reviews to pre-train LMs, and we will examine the effects of these two types of data in the experiments. The details of pre-training LMs and inferring soft prototypes are as follows.

Pre-training Language Models As shown in Figure 4(a), given S, the forward $\overrightarrow{LM}$ computes the probability of S by modeling the probability of token w_i conditioned on the history $(w_1, .., w_{i-1})$:

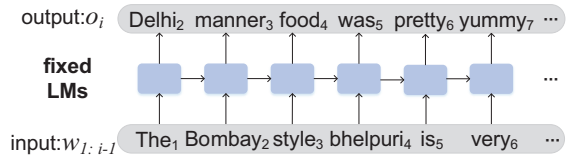
$$prob(w_1, ..., w_n) = \prod_{i=1}^{n} prob(w_i | w_1, ..., w_{i-1}). \quad (1)$$

In the pre-training process, $\overrightarrow{LM}$ tries to maximize the log likelihood of the forward direction:

$$\sum_{i=1}^{n} \log prob(w_i | w_1, w_2, ..., w_{i-1}; \overrightarrow{\theta_{LM}}). \quad (2)$$



(a) Pre-training a forward language model.



(b) Inferring oracle words.

Figure 4: Generating soft prototypes with LMs.

Similarly, the backward $\overleftarrow{LM}$ tries to maximize the log likelihood of the backward direction:

$$\sum_{i=1}^{n} \log prob(w_i | w_{i+1}, w_{i+2}, ..., w_n; \overleftarrow{\theta_{LM}}). \quad (3)$$

After the pre-training process converges, we can fix $\overrightarrow{\theta_{LM}}$ and $\overleftarrow{\theta_{LM}}$, and infer a soft prototype **P** conditioned on S, $\overrightarrow{\theta_{LM}}$, and $\overleftarrow{\theta_{LM}}$ for each sample in the training and testing sets in ATE[1].

Generating Soft Prototypes After getting $\overrightarrow{\theta_{LM}}$ and $\overleftarrow{\theta_{LM}}$, we then infer the soft prototype **P**. We still take the forward $\overrightarrow{LM}$ as the example.

As shown in Figure 4(b), for generating the forward prototype vector $\overrightarrow{\boldsymbol{p_i}}$ for word w_i, we feed the prefix sentence $\{w_1, w_2, ..., w_{i-1}\}$ to the fixed $\overrightarrow{LM}$ and collect the output probability distribution $prob(o_i^v | w_1, w_2, ..., w_{i-1}; \overrightarrow{\theta_{LM}})$, where V is the size of vocabulary and $v \in [1, V]$. To suppress noise, we do not directly select the word o_i^1 with the largest output probability. Instead, we preserve the words $\{o_i^1, o_i^2, ..., o_i^K\}$ with K-largest output probabilities, and normalize their probabilities to sum 1 as the weighted scores $\{s_i^1, s_i^2, ..., s_i^K\}$. We call the selected words as "*oracle words*" (Zhang et al., 2019). Then we map these words with a pre-trained embedding lookup table **E** and obtain their word vectors $\{\boldsymbol{o_i^1}, \boldsymbol{o_i^2}, ..., \boldsymbol{o_i^K}\}$. Finally, we aggregate the oracle words by their weighted scores to calculate $\overrightarrow{\boldsymbol{p_i}}$ for word w_i:

$$\overrightarrow{\boldsymbol{p_i}} = \sum_{k=1}^{K} \boldsymbol{o_i^k} \cdot s_i^k. \quad (4)$$

Similarly, we can calculate the backward prototype vector $\overleftarrow{\boldsymbol{p_i}}$. To consider the context information in both directions, we use the average of $\overrightarrow{\boldsymbol{p_i}}$ and $\overleftarrow{\boldsymbol{p_i}}$ as the final prototype vector $\boldsymbol{p}_i$ for word w_i. We then regard the set of prototype vectors $\{\boldsymbol{p}_1, \boldsymbol{p}_2, ..., \boldsymbol{p}_n\}$ as the soft prototype **P** for the sentence S[2].

3.3 Gating Conditioner

For better discovering the aspect terms, we need to leverage the supporting evidence embedded in the soft prototype **P**. Intuitively, we have two schemes to incorporate the soft prototypes into ATE: inside or outside the sequence tagger. We choose the latter because we want to decouple the soft prototypes from the sequence taggers, such that we can make the prototypes suitable for all types of taggers. Hence, we introduce an additional upstream module named the gating conditioner to fuse the soft prototype **P** with the original sentence S.

The soft prototype **P** provides two kinds of information : (1) **P** itself has embedded data-specific knowledge that can serve as supporting evidence. (2) **P** also helps to refine the original representation of S. Accordingly, the gating conditioner is developed to conduct two types of operations on **P**. We first map $S = \{w_1, w_2, ..., w_n\}$ with the pre-trained embedding lookup table **E** and obtain the corresponding word vectors **X**$= \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n\}$. Then, we conduct two types of operations on **X** and **P** to obtain the fused vectors **F**:

$$\boldsymbol{f_i} = \sigma(W(\boldsymbol{x}_i \oplus \boldsymbol{p}_i) + b) \odot (\boldsymbol{x}_i \oplus \boldsymbol{p}_i), \quad (5)$$

where σ is the Sigmoid function, W and b are trainable parameters, $\oplus$ and $\odot$ denotes the concatenation and element-wise multiplication operation, respectively.

In Eq. 5, the concatenation of **P** and **X** makes the representation more discriminative than before. Moreover, the gating mechanism can help select the important dimensions and further refine the representation. The generated fused vectors **F**$= \{\boldsymbol{f_1}, \boldsymbol{f_2}, ..., \boldsymbol{f_n}\}$ then act as the enhanced representation for $S = \{w_1, w_2, ..., w_n\}$.

3.4 Sequence Tagger

The sequence tagger aims to extract high-level semantic features from the low-level tokens, and predicts a tag sequence Y for the review S based on these features. In order to investigate the influence of soft prototypes, we need to control variables in SoftProto. Therefore, we choose three existing sequence taggers as our basic models, including BiLSTM (Liu et al., 2015), DECNN (Xu et al.,

[1]Note that the testing ATE samples are not used for pre-training the LMs, thus there is no data leakage in this process.

[2]Since $\overrightarrow{\boldsymbol{p_1}}$ and $\overleftarrow{\boldsymbol{p_n}}$ are inferred based on the BOS token, they are less informative than other prototype vectors. In practice, we replace them with the pre-trained word vectors of w_1 and w_n, respectively.

2018), and Seq2Seq4ATE (Ma et al., 2019). Readers can refer to the original paper for more details or Section 4.2 for a quick glance. Please note that the only difference between an original sequence tagger and its variant enhanced by our proposed SoftProto is the representation of S. In other words, by comparing the performance of a sequence tagger and its enhanced variant, we can observe that how ATE benefits from soft prototypes.

For training SoftProto, we simply compute the cross-entropy loss $\mathcal{L}$:

$$\mathcal{L} = -\sum_{i=1}^{n}\sum_{j=1}^{J} \hat{y_{ij}} \cdot log(y_{ij}), \quad (6)$$

where n is the length of S, J is the category of labels, y_i and $\hat{y}_i$ are the predicted tags and ground truth labels. We then train all parameters with back propagation.

4 Experiments

4.1 Datasets and Settings

ATE Datasets To evaluate the effectiveness of SoftProto for ATE tasks, we conduct extensive experiments on four datasets from SemEval 2014 (Pontiki et al., 2014), 2015 (Pontiki et al., 2015) and 2016 (Pontiki et al., 2016). These datasets contain review sentences from the restaurant and laptop domains with annotated aspect terms. All of them have a fixed train/test split, and we further randomly hold out 150 training samples as the validation set for tuning hyper-parameters. The statistics of four ATE datasets are summarized in Table 1[3].

Table 1: The statistics of ATE datasets.

Datasets	Lap14		Res14		Res15		Res16	
Type	train	test	train	test	train	test	train	test
Sentences	3045	800	3041	800	1315	685	2000	676
Aspects	2342	650	3686	1134	1209	547	1757	622

Details for Pre-training Language Models As mentioned in section 3.2, we use two types of data to pre-train the LMs: (1) The ATE training sets. In this setting, we directly use the same training/validation samples of each SemEval dataset to pre-train its own LMs. Hence, there are four groups of pre-trained LMs (including $\overrightarrow{LM}$ and $\overleftarrow{LM}$) for four datasets, respectively. We denote this setting as **SoftProtoI** (I for internal knowledge). (2) The unlabeled external data. In this setting, we additionally collect 100,000 training and 10,000 validation samples from Yelp Review (Zhang et al., 2015) and Amazon Electronics (McAuley et al., 2015) datasets, respectively. LMs pre-trained on Yelp serve as the prototype generator when training and evaluating SoftProto on {Res14, Res15, Res16} datasets, while those pre-trained on Amazon are used for the Lap14 dataset. We denote this setting as **SoftProtoE** (E for external knowledge). For pre-training the LMs, we adopt the Fairseq[4] toolkit (Ott et al., 2019) and the basic transformer decoder LM architecture (Vaswani et al., 2017)[5].

Parameter Settings The only hyperparameter in our SoftProto is the number K of oracle words when generating soft prototypes. We use a grid search to select K in the range [1,10] based on the validation performance, and consequently set K=$\{10, 7, 10, 7\}$ for four datasets, respectively. For other parameters, including the pre-trained word embedding, epoch number, optimizer selection, learning rate, and batch size, we inherit the default settings from the original papers (Liu et al., 2015; Xu et al., 2018; Ma et al., 2019). Models achieving the maximum F1-scores on the validation set are used for evaluation on the testing set. We report the averaged F1 scores over 5 runs with random initialization. We run all methods in a single 2080Ti GPU.

4.2 Compared Methods

We choose two kinds of baselines. The first is the SemEval winners for corresponding datasets. **IHS-RD** (Chernyshevich, 2014), **DLIREC** (Toh and Wang, 2014), **EliXa** (Vicente et al., 2015), and **NLANGP** (Toh and Su, 2016) are the winners for Lap14, Res14, Res15, and Res16 datasets, respectively. The second is the recent deep learning-based methods. **RNCRF** (Wang et al., 2016), **MIN** (Li and Lam, 2017), **CMLA** (Wang et al., 2017), and **HAST** (Li et al., 2018) are frequently-used neural baselines. They all introduce the auxiliary opinion term extraction (OTE) task and exploit the relation between ATE and OTE.

In order to discern the impacts of soft prototypes on pure ATE task, we do not choose the hybrid models as the base taggers. Instead, we adapt SoftProto to three pure sequence taggers, including **BiLSTM** (Liu et al., 2015; Li et al., 2018) which is an RNN-based sequence tagger including a vanilla

[3]Our code and data are available at https://github.com/NLPWM-WHU/SoftProto.

[4]https://github.com/pytorch/fairseq.

[5]In practice, we also tried a self-constructed single-layer LSTM architecture and got a similar performance in language modeling. Since the Fairseq toolkit has already integrated the transformer architecture, we directly use it for convenience.

Table 2: Comparison of different methods in F1-scores. Results for the first eight methods are taken from Li et al. (2018), while other results are the averaged scores of 5 runs with random initialization. The best scores are in bold, and the best baselines are underlined. The subscript denotes the improvement/decrease after enhancing an ATE tagger with a certain method (e.g., BiLSTM + SoftProtoE vs. BiLSTM). * denotes the statistical significance between the orginal methods and their enhanced counterparts at $p < 0.05$ level.

Model	Lap14	Res14	Res15	Res16
IHS-RD	74.55	79.62	-	-
DLIREC	73.78	84.01	-	-
EliXa	-	-	70.04	-
NLANGP	-	-	67.12	72.34
RNCRF	78.42	84.93	67.74	69.72
MIN	77.58	-	-	73.44
CMLA	77.80	85.29	70.73	72.77
HAST	79.52	85.61	<u>71.46</u>	73.61
Selected ATE taggers and their enhanced variants				
BiLSTM	73.69	82.02	64.66	67.95
+ Synonym	$74.34_{(+0.65)}$	$81.93_{(-0.09)}$	$65.33_{(+0.67)}$	$68.49_{(+0.54)}$
+ Replacement	$73.47_{(-0.22)}$	$81.78_{(-0.24)}$	$65.72_{(+1.06)}$	$67.29_{(-0.66)}$
+ SoftProtoI	$73.97_{(+0.28)}$	$82.82_{(+0.80)}$	$65.77_{(+1.11)}$	$68.04_{(+0.09)}$
+ SoftProtoE	$74.75^{*}_{(+1.06)}$	$84.27^{*}_{(+2.25)}$	$66.06_{(+1.40)}$	$69.65_{(+1.70)}$
Seq2Seq4ATE	79.02	84.08	69.89	72.82
+ Synonym	$79.21_{(+0.19)}$	$84.34^{*}_{(+0.26)}$	$70.12_{(+0.23)}$	$73.99^{*}_{(+1.17)}$
+ Replacement	$78.97_{(-0.05)}$	$84.56_{(+0.48)}$	$70.17_{(+0.28)}$	$73.92_{(+1.10)}$
+ SoftProtoI	$79.13_{(+0.11)}$	$85.16^{*}_{(+1.08)}$	$71.92_{(+2.03)}$	$74.80^{*}_{(+1.98)}$
+ SoftProtoE	$80.46^{*}_{(+1.44)}$	$87.38^{*}_{(+3.30)}$	$71.99_{(+2.10)}$	$76.06^{*}_{(+3.24)}$
DECNN	<u>81.39</u>	<u>86.04</u>	71.18	<u>74.39</u>
+ Synonym	$81.93^{*}_{(+0.54)}$	$85.45_{(-0.59)}$	$70.80_{(-0.38)}$	$75.78^{*}_{(+1.39)}$
+ Replacement	$80.19^{*}_{(-1.20)}$	$85.54_{(-0.50)}$	$72.08_{(+0.90)}$	$74.96_{(+0.57)}$
+ SoftProtoI	$82.67^{*}_{(+1.28)}$	$86.35_{(+0.31)}$	$72.01^{*}_{(+0.83)}$	$75.88^{*}_{(+1.49)}$
+ SoftProtoE	$\mathbf{83.19}^{*}_{(+1.80)}$	$\mathbf{87.39}^{*}_{(+1.35)}$	$\mathbf{73.27}^{*}_{(+2.09)}$	$\mathbf{76.98}^{*}_{(+2.59)}$

BiLSTM architecture, **DECNN** (Xu et al., 2018) which is a CNN-based sequence tagger which uses two types of pre-trained embeddings and stacked convolutional layers to extract context features for tagging aspect terms, and **Seq2Seq4ATE** (Ma et al., 2019) which is an attention-based sequence tagger and uses a modified encoder-decoder framework to extract aspect terms.

We further compare SoftProto with two simple enhancing methods, namely **Synonym** and **Replacement**. For Synonym, we substitute the top-K oracle words with top-K nearest synonyms measured by the cosine distance of word vectors while keeping the other settings unchanged. For Replacement, we use the prototype generated by our language models, but replace the training words with the method in Kobayashi (2018). The modified samples are sent to the sequence tagger directly[6].

4.3 Main Results

The comparison results for all methods are shown in Table 2 . Obviously, SoftProto greatly boosts all basic sequence taggers. For example, DECNN achieves an overall best performance among baselines, while SoftProtoI and SoftProtoE further achieve {1.28%,0.31%,0.83%,1.49%} and {1.80%,1.35%, 2.09%,2.59%} absolute gains for DECNN on four datasets, respectively. There even exists an amazing 3.30% gain after incorporating SoftProtoE to Seq2Seq4ATE on the Res14 dataset. This strongly demonstrates the effectiveness of proposed soft prototypes for the ATE task. By correlating samples through the soft prototypes, the training of sequence taggers can easily converge to a better state than before.

We also find that the improvements brought by the SoftProto are more remarkable on small datasets (Res15 and Res16) than those on large ones (Res14 and Lap14). This is because there are not enough samples on small datasets to train a well-performed sequence tagger, and the discovery of aspect terms largely relies on the knowledge embedded in the soft prototypes. Moreover, SoftProtoE performs much better than SoftProtoI. The reason is that the external unlabeled data from Yelp and Amazon is much bigger and more informative than the original ATE datasets. Accordingly, the pre-trained LMs in SoftProtoE contain more knowledge than those in SoftProtoI and can generate more discriminative soft prototypes.

[6]We use a grid search to select the replacement probability and present the best results. Prototype tokens are generated using the LMs pre-trained on the Yelp/Amazon data.

The performances of Synonym and Replacement are far from satisfactory, and they even result in decreases in some cases. Synonym generates noisy prototypes by only considering the individual tokens, and can hardly handle the unknown (UNK) words. The ineffectiveness of Replacement lies in two issues. Firstly, it simply replaces the original words with the generated ones, which incurs information loss. Secondly, the generated knowledge cannot be fully utilized due to the small percentage of replacement. The inferior results demonstrate that these two methods are not qualified for enhancing the ATE task.

5 Analysis

5.1 Perplexities of Language Models

In this section, we present the perplexities of language models pre-trained on different datasets. As shown in Table 3, the perplexity is linearly related to the size of datasets. The larger the dataset, the lower the perplexity.

Table 3: Perplexities of pre-trained language models.

Source	Dataset	Forward LM	Backward LM
Internal	Lap14	147.00	146.79
	Res14	164.31	169.67
	Res15	238.68	236.07
	Res16	199.76	200.69
External	Yelp	48.53	49.63
	Amazon	56.45	57.16

Clearly, LMs trained on external Yelp/Amazon datasets have much lower perplexities than original SemEval datasets. Among the SemEval datasets, Lap14 and Res14 have relatively more samples than Res15 and Res16, resulting in relatively lower perplexities. Moreover, language models in forward and backward directions have no significant differences in performance. We will release all pre-trained language models in time for encouraging further studies on soft prototypes.

5.2 Ablation Study

Without loss of generality, we choose two DECNN +SoftProto models and conduct the ablation study to investigate the effects of different modules in SoftProto. We sequentially remove the forward LM, the backward LM, the concatenation operation, and the gating operation to obtain four simplified variants.

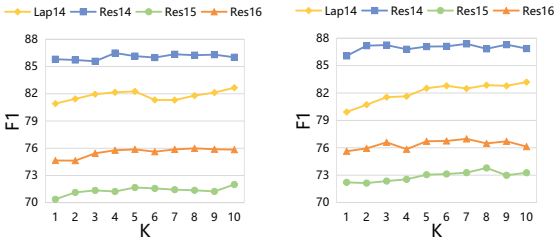
As shown in Table 4, all variants have a performance decrease of the F1-score. The results demonstrate that : (1) Considering both directions in language modeling can generate better soft prototypes. (2) Both kinds of conditioning operations (i.e., gating and concatenation) can contribute to the utilization of the soft prototypes.

Table 4: Ablation study. The scores denote the performance decreases of SoftProtoI/SoftProtoE after removing the component.

	Lap14	Res14	Res15	Res16
- $\overrightarrow{LM}$	2.44 / 0.83	1.71 / 1.59	1.42 / 1.11	0.16 / 0.41
- $\overleftarrow{LM}$	1.56 / 0.24	0.24 / 0.78	1.31 / 1.22	0.66 / 0.17
- concat	1.65 / 2.07	0.14 / 1.94	0.28 / 0.25	1.00 / 1.04
- gate	2.48 / 1.44	0.04 / 1.31	2.97 / 0.07	2.99 / 2.78

5.3 Impacts of Oracle Words

In the prototype generator, the hyper-parameter K controls how many oracle words are taken into account when generating soft prototypes. To investigate the impacts of the oracle words on different datasets, we vary K in the range of [1,10] stepped by 1, and present the results of two DECNN+SoftProto models in Figure 5.



(a) +SoftProtoI. (b) +SoftProtoE.

Figure 5: F1-scores under different settings of K.

Generally, the F1-scores of DECNN have an overall upward trend when more oracle words are introduced. This is explainable since the oracle words actually provide the data-specific knowledge that can be aggregated into the soft prototypes. Moreover, owing to the high confidence of language models trained on Yelp/Amazon datasets, the curves of SoftProtoE are smoother than those of SoftProtoI. The reason is that language models with high perplexities almost inevitably output noisy oracle words and bring about the high variance when generating soft prototypes.

5.4 Case Study

To have a close look, we further select six samples from the testing sets for a case study. Due to the space limitation, we only present the results of the best baseline DECNN and its two variants enhanced by SoftProto in Table 5.

S1∼S2 are in similar circumstances. DECNN only extracts a single word as the aspect term and

Table 5: Case study. The left column presents the selected examples, and the words in red with brackets denote individual aspect terms. The three columns on the right denote the extraction results of corresponding models.

Examples	DECNN	+SoftProtoI	+SoftProtoE
S1.My favs here are the [Tacos Pastor] and the [Tostada de Tinga] .	Tacos, Tostada de Tinga ✗	Tacos Pastor, Tostada de Tinga	Tacos Pastor, Tostada de Tinga
S2.Fine if you have a [touch screen] .	screen ✗	touch screen	touch screen
S3.[Web surfuring] is smooth and seamless .	Web ✗	Web surfuring	Web surfuring
S4.My one complaint is that there was no [internal CD drive] .	CD drive ✗	internal CD drive	internal CD drive
S5.They are [served] on [Focacchia bread] and are to die for .	served ✗	served ✗	served, Focacchia bread
S6.The [food] is great and they make a mean [bloody mary] .	food ✗	food ✗	food, bloody mary

neglects the integrality of phrases. Since the aspect phrases [*Tacos Pastor*] and [*touch screen*] have not occurred on the corresponding training sets on Res14 and Lap14 datasets, their linguistic features are not strong enough to become aspect terms. In contrast, SoftProto variants can make correct extractions. We go deep into the prototype generator and investigate the oracle words generated by the language models. For [*Pastor*], LMs introduce words like [*nachos*, *burrito*, *salsa*, *food*]. And for [*touch*], LMs introduce words like [*DSLR*, *cable*, *camera*, *projector*]. Obviously, these oracle words are strong indicative words for [*Pastor*] and [*touch*], and hence SoftProto is able to tag them correctly.

S3 is another interesting example. Since [*surfuring*] is an rare variant of [*surfing*], DECNN only extracts [*Web*] as the aspect term. For SoftProto, LMs introduce oracle words like [*browsing*, *management*, *interface*, *search*]. Owing to the knowledge embedded in the oracle words, recognizing [*Web surfuring*] as a complete aspect term becomes much easier than before.

S4 shows another ability of SoftProto, i.e., judging whether an adjective is sentimental or descriptive. [*internal*] is a descriptive adjective for [*CD drive*] without polarity, and should be included in the aspect term. DECNN regards [*internal*] as an opinion word and neglects it. For SoftProto, LMs condition [*internal*] with oracle words like [*AC*, *on/off*, *wire*, *cable*]. The nominal information contained in the soft prototype helps the sequence tagger extract [*internal*] correctly.

S5 and S6 verify the superiority of SoftProtoE over SoftProtoI. Since the perplexities of LMs in SoftProtoI are much higher than those in SoftProtoE, the oracles words in SoftProtoI correspondingly have lower qualities. For [*Focacchia*], SoftProtoI introduces meaningless oracle words like [*the*, *a*, *my*, *our*]. In contrast, SoftProtoE produces [*pumpkin*, *homemade*, *garlic*, *baked*], which are closely related with [*Focacchia*]. Similarly, for [*bloody mary*], SoftProtoE introduces [*garlic*, *martini*] to [*bloody*] and [*mojito*, *beer*] to [*mary*], while the oracle words generated by SoftProtoI are less informative.

5.5 Performance on Tail Aspect Terms

To prove that SoftProto are indeed beneficial for identifying the tail aspect terms, we keep the training sentences unchanged and only preserve the testing sentences containing the tail aspect terms (appearing no more than 3 times in training sentences). We present the performance of DECNN and its two variants enhanced by SoftProto on these sentences in Table 6. Clearly, SoftProto enhances the ability of DECNN in recognizing the tail aspect terms by a large margin.

Table 6: The performance of DECNN and its SoftProto variants on recognizing the tail aspect terms.

	Lap14	Res14	Res15	Res16
Tail Percentage	30.50	30.38	30.66	28.11
DECNN	74.37	77.61	70.00	70.68
+SoftProtoI	78.88	79.96	75.04	71.80
+SoftProtoE	79.85	82.22	76.80	70.93

5.6 Prototypes Generation with BERT

Since BERT (Devlin et al., 2019) is pre-trained as a masked language model (MLM), we wonder if it can serve as the prototype generator. Hence, we regard the generation of prototypes as a cloze test. We sequentially mask each word and collect the top-K output words of the MLM as the oracle words. We name this variant SoftProtoB. The setting of K and the usage of the oracle words remain the same as those in SoftProtoI and SoftProtoE, thus the only difference among all these SoftProto variants is the way of pre-training language models.

We conduct experiments on two pre-trained BERT models, where SoftProtoB (BASE) is the officially released BERT-Base-Uncased model, and SoftProtoB (PT) is further post-trained on domain-specific data and released by Xu et al. (2019). Since both SoftProtoB and SoftProtoE make use of the external data, they are fair competitors and we list the results of these two variants in Table 7.

Table 7: Comparison between two pre-trained BERT models and one pre-trained traditional language model.

	Lap14	Res14	Res15	Res16
DECNN	81.39	86.04	71.18	74.39
+SoftProtoB (BASE)	82.15	86.84	71.20	75.09
+SoftProtoB (PT)	82.30	87.70	72.69	76.43
+SoftProtoE	83.19	87.39	73.27	76.98

From the results in Table 7, we can see that the BERT-based models are also qualified for generating the soft prototypes. In general, SoftProtoB (BASE) generates domain-independent oracle words and achieves limited improvements over the base model, while SoftProtoB (PT) can generate domain-specific oracle words and achieves a comparable performance with SoftProtoE.

5.7 Analysis on Computational Cost

Since we use the pre-trained language models, the cost for generating soft prototypes can almost be ignored. To demonstrate that SoftProto does not incur the high computational cost in utilizing soft prototypes, we run three sequence taggers on the Laptop 2014 dataset, and present the trainable parameter number and running time per epoch of each method before and after introducing SoftProto in Table 8.

Table 8: Computational cost of each method.

	Parameter Number	Runtime
BiLSTM	903,903	3s
+ SoftProto	1,263,903	3s
Seq2Seq4ATE	4,738,353	87s
+ SoftProto	5,638,953	91s
DECNN	1,394,435	2s
+ SoftProto	2,444,835	3s

From Table 8, we can conclude that SoftProto is a lightweight framework and does not add much cost on the original sequence taggers.

6 Conclusion

In this paper, we present a general SoftProto framework to enhance the ATE task. Rather than designing elaborated sequence taggers, we turn to correlate samples with each other through soft prototypes. For this purpose, we resort to the language models for automatically generating soft prototypes and then design a gating conditioner for utilizing them. The performance of SoftProto can be further improved after introducing the large-scale external unlabeled data like Yelp and Amazon reviews. Extensive experiments on four SemEval datasets demonstrate that SoftProto greatly boosts the performance of the typical ATE methods and introduces small computational cost.

Acknowledgments

We thank the anonymous reviewers for their valuable comments. The work described in this paper is supported by the NSFC projects (61572376, 91646206), and the 111 project (B07037).

References

Ziqiang Cao, Wenjie Li, Sujian Li, and Furu Wei. 2018. Retrieve, rerank and rewrite: Soft template based neural summarization. In *ACL*, pages 152–161.

Zhiyuan Chen, Arjun Mukherjee, and Bing Liu. 2014. Aspect extraction with automated prior knowledge learning. In *ACL*, pages 347–358.

Zhuang Chen and Tieyun Qian. 2019. Transfer capsule network for aspect level sentiment classification. In *ACL*, pages 547–556.

Zhuang Chen and Tieyun Qian. 2020. Relation-aware collaborative learning for unified aspect-based sentiment analysis. In *ACL*, pages 3685–3694.

Maryna Chernyshevich. 2014. IHS r&d belarus: Cross-domain extraction of product features using CRF. In *SemEval@COLING*, pages 309–313.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL*, pages 4171–4186.

Jiatao Gu, Yong Wang, Kyunghyun Cho, and Victor O. K. Li. 2018. Search engine guided neural machine translation. In *AAAI*, pages 5133–5140.

Kelvin Guu, Tatsunori B. Hashimoto, Yonatan Oren, and Percy Liang. 2018. Generating sentences by editing prototypes. *Trans. Assoc. Comput. Linguistics*, 6:437–450.

Tatsunori B. Hashimoto, Kelvin Guu, Yonatan Oren, and Percy Liang. 2018. A retrieve-and-edit framework for predicting structured outputs. In *NIPS*, pages 10073–10083.

Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. 2017. An unsupervised neural attention model for aspect extraction. In *ACL*, pages 388–397.

Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. 2018. Exploiting document knowledge for aspect-level sentiment classification. In *ACL*, pages 579–585.

Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. 2019. An interactive multi-task learning network for end-to-end aspect-based sentiment analysis. In *ACL*, pages 504–515.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *NeurIPS*, pages 2042–2050.

Minghao Hu, Yuxing Peng, Zhen Huang, Dongsheng Li, and Yiwei Lv. 2019. Open-domain targeted sentiment analysis via span-based extraction and classification. In *ACL*, pages 537–546.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *KDD*, pages 168–177.

Zongcheng Ji, Zhengdong Lu, and Hang Li. 2014. An information retrieval approach to short text conversation. *CoRR*, abs/1408.6988.

Sosuke Kobayashi. 2018. Contextual augmentation: Data augmentation by words with paradigmatic relations. In *NAACL*, pages 452–457.

Fangtao Li, Chao Han, Minlie Huang, Xiaoyan Zhu, Yingju Xia, Shu Zhang, and Hao Yu. 2010. Structure-aware review mining and summarization. In *COLING*, pages 653–661.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *NAACL*, pages 110–119.

Kun Li, Chengbo Chen, Xiaojun Quan, Qing Ling, and Yan Song. 2020. Conditional augmentation for aspect term extraction via masked sequence-to-sequence generation. *CoRR*, abs/2004.14769.

Xin Li, Lidong Bing, Piji Li, and Wai Lam. 2019. A unified model for opinion target extraction and target sentiment prediction. In *AAAI*, pages 6714–6721.

Xin Li, Lidong Bing, Piji Li, Wai Lam, and Zhimou Yang. 2018. Aspect term extraction with history attention and selective transformation. In *IJCAI*, pages 4194–4200.

Xin Li and Wai Lam. 2017. Deep multi-task learning for aspect term extraction with memory interaction. In *EMNLP*, pages 2886–2892.

Ming Liao, Jing Li, Haisong Zhang, Lingzhi Wang, Xixin Wu, and Kam-Fai Wong. 2019. Coupling global and local context for unsupervised aspect extraction. In *EMNLP*, pages 4578–4588.

Kang Liu, Heng Li Xu, Yang Liu, and Jun Zhao. 2013. Opinion target extraction using partially-supervised word alignment model. In *IJCAI*, pages 2134–2140.

Kang Liu, Liheng Xu, and Jun Zhao. 2012. Opinion target extraction using word-based translation model. In *EMNLP*, pages 1346–1356.

Pengfei Liu, Shafiq R. Joty, and Helen M. Meng. 2015. Fine-grained opinion mining with recurrent neural networks and word embeddings. In *EMNLP*, pages 1433–1443.

Huaishao Luo, Tianrui Li, Bing Liu, and Junbo Zhang. 2019. DOER: dual cross-shared RNN for aspect term-polarity co-extraction. In *ACL*, pages 591–601.

Dehong Ma, Sujian Li, Fangzhao Wu, Xing Xie, and Houfeng Wang. 2019. Exploring sequence-to-sequence learning in aspect term extraction. In *ACL*, pages 3538–3547.

Julian J. McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-based recommendations on styles and substitutes. In *SIGIR*, pages 43–52.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *NAACL-HLT*, pages 48–53.

Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Ion Androutsopoulos, Suresh Manandhar, Mohammad Al-Smadi, Mahmoud Al-Ayyoub, Yanyan Zhao, Bing Qin, Orphée De Clercq, Véronique Hoste, Marianna Apidianaki, Xavier Tannier, Natalia V. Loukachevitch, Evgeniy V. Kotelnikov, Núria Bel, Salud María Jiménez Zafra, and Gülsen Eryigit. 2016. Semeval-2016 task 5: Aspect based sentiment analysis. In *SemEval@NAACL-HLT*, pages 19–30.

Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Suresh Manandhar, and Ion Androutsopoulos. 2015. Semeval-2015 task 12: Aspect based sentiment analysis. In *SemEval*, pages 486–495.

Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. Semeval-2014 task 4: Aspect based sentiment analysis. In *SemEval*, pages 27–35.

Ana-Maria Popescu and Oren Etzioni. 2005. Extracting product features and opinions from reviews. In *EMNLP*, pages 339–346.

Guang Qiu, Bing Liu, Jiajun Bu, and Chun Chen. 2011. Opinion word expansion and target extraction through double propagation. *Computational Linguistics*, 37(1):9–27.

Zhiqiang Toh and Jian Su. 2016. NLANGP at semeval-2016 task 5: Improving aspect based sentiment analysis using neural network features. In *SemEval@NAACL-HLT*, pages 282–288.

Zhiqiang Toh and Wenting Wang. 2014. DLIREC: aspect term extraction and term polarity classification system. In *SemEval@COLING*, pages 235–240.

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 *NIPS*, pages 5998–6008.

Iñaki San Vicente, Xabier Saralegi, and Rodrigo Agerri. 2015. Elixa: A modular and flexible ABSA platform. In *SemEval@NAACL-HLT*, pages 748–752.

Feixiang Wang, Man Lan, and Wenting Wang. 2018. Towards a one-stop solution to both aspect extraction and sentiment analysis tasks with neural multi-task learning. In *IJCNN*, pages 1–8.

Wenya Wang, Sinno Jialin Pan, Daniel Dahlmeier, and Xiaokui Xiao. 2016. Recursive neural conditional random fields for aspect-based sentiment analysis. In *EMNLP*, pages 616–626.

Wenya Wang, Sinno Jialin Pan, Daniel Dahlmeier, and Xiaokui Xiao. 2017. Coupled multi-layer attentions for co-extraction of aspect and opinion terms. In *AAAI*, pages 3316–3322.

Yiren Wang, Yingce Xia, Fei Tian, Fei Gao, Tao Qin, ChengXiang Zhai, and Tie-Yan Liu. 2019. Neural machine translation with soft prototype. In *NIPS*, pages 6313–6322.

Yu Wu, Furu Wei, Shaohan Huang, Yunli Wang, Zhoujun Li, and Ming Zhou. 2019. Response generation by context-aware prototype editing. In *AAAI*, pages 7281–7288.

Yuanbin Wu, Qi Zhang, Xuanjing Huang, and Lide Wu. 2009. Phrase dependency parsing for opinion mining. In *EMNLP*, pages 1533–1541.

Hu Xu, Bing Liu, Lei Shu, and Philip S. Yu. 2018. Double embeddings and cnn-based sequence labeling for aspect extraction. In *ACL*, pages 592–598.

Hu Xu, Bing Liu, Lei Shu, and Philip S. Yu. 2019. BERT post-training for review reading comprehension and aspect-based sentiment analysis. In *NAACL*, pages 2324–2335.

Yichun Yin, Furu Wei, Li Dong, Kaimeng Xu, Ming Zhang, and Ming Zhou. 2016. Unsupervised word and dependency path embeddings for aspect term extraction. In *IJCAI*, pages 2979–2985.

Wen Zhang, Yang Feng, Fandong Meng, Di You, and Qun Liu. 2019. Bridging the gap between training and inference for neural machine translation. In *ACL*, pages 4334–4343.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *NIPS*, pages 649–657.