SALNet: Semi-Supervised Few-Shot Text Classification with Attention-based Lexicon Construction

Ju-Hyoung Lee¹, Sang-Ki Ko², Yo-Sub Han¹

¹Yonsei University, Seoul, Republic of Korea
²Kangwon National University, Kangwon, Republic of Korea

juhyounglee@yonsei.ac.kr, sangkiko@kangwon.ac.kr, emmous@yonsei.ac.kr

Abstract

We propose a semi-supervised bootstrap learning framework for few-shot text classification. From a small number of the initial data, our framework obtains a larger set of reliable training data by using the attention weights from an LSTM-based trained classifier. We first train an LSTM-based text classifier from a given labeled dataset using the attention mechanism. Then, we collect a set of words for each class called a lexicon, which is supposed to be a representative set of words for each class based on the attention weights calculated for the classification task. We bootstrap the classifier using the new data that are labeled by the combination of the classifier and the constructed lexicons to improve the prediction accuracy. As a result, our approach outperforms the previous state-of-the-art methods including semi-supervised learning algorithms and pretraining algorithms for few-shot text classification task on four publicly available benchmark datasets. Moreover, we empirically confirm that the constructed lexicons are reliable enough and substantially improve the performance of the original classifier.

Introduction

Recently, text classifiers using deep learning show great success in various NLP tasks due to lots of labeled training data. However, one critical limit is that these effective classifiers are hard to make when there is not enough labeled data. Often it is difficult and expensive to obtain a reasonable number of labeled data since it requires many well-trained human annotators. On the other hand, a traditional approach for text classification is to use a domain lexicon for classification since we can correctly decide a class using the lexicon and the size of the training data is irrelevant. The efforts to address the data sparsity problem are to make lexicons of each domain and use it for text classification (Lu and Tsou 2010; Lei et al. 2011; Hailong, Wenyan, and Bo 2014; Bandhakavi et al. 2017; Lee et al. 2018). On the other hand, it is not easy to automatically make a high-quality lexicon for a new domain (and. Binbin Chen and Bernstein 2016; Feng et al. 2018). Semi-Supervised Learning (SSL) is an approach to use both a small number of labeled data and a large number of unlabeled data in training. The traditional SSL methods based on neural network (Yarowsky 1995; Blum and Mitchell 1998; Li and Liu 2003; Zhu, Ghahramani, and Lafferty 2003; Rosenberg, Hebert, and Schneiderman 2005; Wang and Zhang 2007; Yastin and Catalpepe 2010; Jo and Cinarel 2019) use only high confidence predictions of classifiers or utilize the agreement among the different classifiers, for pseudo-labeling. These methods can lead to accumulated classification errors along the training process. Therefore, it remains challenging to train classifiers for text classification under semi-supervision, because it still requires hundreds of thousands of labeled training data to achieve satisfactory performance.

We combine the deep learning classifier with the lexicon and tackle the labeled-data sparsity problem. With respect to the attention mechanism, important words often have high attention weights for text classification (Cho et al. 2014; Sutskever, Vinyals, and Le 2014; Bahdanau, Cho, and Bengio 2015; Luong, Pham, and Manning 2015; Xu et al. 2015). We visualize the attention layer and empirically confirm that the attention mechanism indeed assigns higher weights to words that can represent the class of input data. We also observe that even though the performance of the initially trained classifier is poor, the classifier assigns higher weights to the important words of the data predicted with high confidence. For instance, the following is a visualization example of the attention mechanism on one of the IMDB review dataset. We collect these high weight words from the attention mechanism and make a set of words (called lexicon) for our semi-supervised classification.

Figure 1: Visualization of the attention mechanism using a LSTM-based classifier trained with few labeled data. Darker colors indicate higher attention weights.

We propose a semi-supervised few-shot text classification with attention-based lexicon construction when there is only a small number of labeled data. Our approach is closely related to neural bootstrapping methods. The bootstrapping methods for SSL use only predictions of neural-based classi-
fiers for labeling data. One major drawback of the bootstrapping is that if the initially trained classifiers have very low performance, then the new training set becomes unreliable because it may contain lots of incorrectly-labeled data. We overcome the problem by utilizing a set of words, which can explicitly predict the unlabeled data using pattern matching.

We evaluate our approach using four publicly available benchmark datasets and compare the performance with the previous state-of-the-art methods including the other semi-supervised learning algorithms (Yarowsky 1995; Jo and Cinarel 2019) and pretraining algorithms (Gururangan et al. 2019; Devlin et al. 2019). The experimental results demonstrate that our approach, Semi-supervised with Attention-based Lexicon construction Net (SALNet), outperforms the previous state-of-the-art methods on four benchmark datasets.

Our main contributions are as follows:

• We propose a semi-supervised bootstrap learning framework that utilizes lexicons constructed by attention mechanism, and our approach has improved accuracy of at least 1% to 8% from initially trained classifiers.

• We verify the effectiveness of the constructed lexicons by improving the accuracy of at least 1% to 9% from the original classifier.

• We demonstrate experimentally that our approach is an effective approach when there is an extremely small labeled dataset.

Background

Here we provide some background knowledge on neural network models discussed in our paper.

Attention mechanism. In the sequence-to-sequence translation model, Bahdanau et al. (2015) hypothesize that the fixed-length context vector $c$ is a bottleneck since the length of the input sequence can vary. They proposed the attention mechanism that computes the context vector by looking at relevant parts from the hidden states of the encoder. Indeed, the attention mechanism has proven surprisingly useful in many tasks in natural language processing. They defined each conditional probability at time $i$ depending on a dynamically computed context vector $c_i$ as follows:

$$p(y_i|y_1, y_2, \ldots, y_{i-1}, x) = \text{softmax}(g(s_i)),$$

where $s_i$ is the hidden state of the decoder RNN at time $i$ computed by $s_i = R(y_{i-1}, s_{i-1}, c_i)$.

The context vector $c_i$ is computed as a weighted sum of the hidden states from encoder: $c_i = \sum_{j=1}^{n} \alpha_{ij} h_j$, where

$$\alpha_{ij} = \frac{\exp(score(s_{i-1}, h_j))}{\sum_{k=1}^{n} \exp(score(s_{k-1}, h_j))}.$$

Here the function ‘score’ is called an alignment function that computes how well the two hidden states from the encoder and the decoder, respectively, match. For example, $score(s_i, h_j)$, where $s_i$ is the hidden state of the encoder at time $i$ and $h_j$ is the hidden state of the decoder at time $j$ implies the probability of aligning the part of the input sentence around position $i$ and the part of the output sentence around position $j$.

Transformer. Vaswani et al. (2017) propose the Transformer, a model architecture eschewing recurrence and instead relying entirely on an attention mechanism to draw global dependencies between input and output. The transformer consists of stacked multi-head attention and parameterized linear transformation layers for both the encoder and decoder. At each layer, the multi-head attention employs $h$ attention heads and performs the self-attention mechanism to capture various context of the input sentence.

Methods

The proposed method involves the following steps:

1. Create a base classifier from a given labeled data, which is very few. Importantly, we train the base classifier to over-fit the training set. The classifier must include an attention mechanism to collect crucial words for each classification.

2. Re-run the base classifier that has an attention mechanism on the unlabeled dataset $U$.

3. Obtain a set of crucial words for predicting $U$, which we call a lexicon of our method; in other words, we use $U$ and the classifier attention weights to create the lexicon for sets of crucial words.

4. Predict the labels of the data in $U$ using the trained classifier and the lexicons.

5. Add the new labels of the data in $U$ using the trained classifier and the lexicons.

6. Repeat the process above until pseudo-labeled data is no longer added to training set. Using the development set at each epoch, we do early-stopping during all the training.

Figure 2 is an overview of our method. We use an attention-based LSTM (Wang et al. 2016) for constructing a set of crucial words for text classification. The attention-based LSTM extracts a set of relevant words based on their attention weights for each class. Ruder and Plank (2018) showed that the relative order of confidence is more robust than absolute confidence and thus, we select the top $n$ unlabeled data with the highest confidence from the classifier in each class. Then, we select $m$ words that have the highest attention weights from each of the selected $n$ data, and make $n$ sets that consist of $m$ words. We regard the collected word set as a lexicon for the corresponding class. Table 1 shows an example of a set of crucial words in each lexicon of the AG News dataset obtained from an initially trained classifier.

We count the number of matching words from unlabeled data with respect to the lexicon of each class. Then, we regard the number of matching words as a prediction confidence of the corresponding class and assign the class of the highest confidence as a predicted label. If there is a tie for the number of matching words between two classes, then we ignore both classes and do not predict the label since there is a
possibility of an incorrect prediction. We have two cases for pseudo-labeling, as shown in Figure 2.

- Case 1: If the classifier predicts the unlabeled data with high confidence and the lexicon has at least $t_1$ matching words, then we label the data according to the prediction of the classifier. In Case 1, we use the lexicon for selecting the correct label among the predictions of the classifiers.

- Case 2: If the lexicon has at least $t_2$ matching words, then we label the data according to the lexicon prediction. In other words, in Case 2, we use the lexicon to decide the label of unlabeled data that the classifier incorrectly predicts with low confidence.

If we use more than two classifiers in SALNet, we can add the pseudo-labeled data by repeating the process of Case 1 and Case 2 for the additional classifiers. Once we obtain a new dataset after pseudo-labeling, the new dataset may have a different number of data in each class. This imbalance may make a classifier overfit to larger classes. We avoid this problem by selecting the same number of data from each class, which is the number of data in the smallest class, for the next training step.

### Experimental Setup

We describe the experimental setup for evaluation.

### Datasets

We use four benchmark datasets to evaluate the performance of our proposed method across different domains; IMDB review (Maas et al. 2011), AG News (Zhang, Zhao, and Le-
Baselines

We compare our approach with the others to verify the effectiveness of our approach. We set 0.9 as the threshold to verify the high confidence for all bootstrapping methods.

- **Self-training (Yarowsky 1995):** Self-training is a one of the simplest approaches for SSL. Since the source code is not available in public, we implement this method using their pseudo algorithm.

- **Delta-training (Jo and Cinarel 2019):** Delta-training is a one of the simplest approaches for SSL. Since the source code is not available in public, we implement this method using their pseudo algorithm.

- **VAMPIRE (Gururangan et al. 2019):** Variational Methods for Pretraining In Resource-limited Environments (VAMPIRE) pretrained a unigram document model as a variational autoencoder (VAE) on unlabeled data and used its internal states as features in a downstream classifier. We use the same hyperparameters used in their paper for the experiments.

- **BERT (Devlin et al. 2019):** We use the pretrained BERT-based-uncased-model and fine-tuned it for the text classification.

**Experimental Results and Analysis**

We evaluate our approach and baselines to test the effectiveness of our approach for few-shot text classification task.

**Main Results**

Table 3 demonstrates that our approach outperforms all the baselines, especially when the performance of initially trained classifiers is less than 80%. The semi-supervised bootstrapping approaches such as self-training and delta-training, which uses only high confidence predictions of classifiers, show low performance when the performance of initially trained classifiers is lower than 75%. The results show the drawback of the bootstrapping that it is only effective when the initially trained classifiers perform well. On the other hand, SALNet shows relatively robust performance, compared to other bootstrapping algorithms. The performance slightly improves on Yahoo! Answer, since there is a small number of data that the classifier predicted with high confidence. On DBpedia, since the performance of the initially trained classifier is already over 94%, the improvement of performance is very small.

We observe that fine-tuning the pretrained BERT achieves the best performance on Yahoo! Answer and DBpedia. Recall that BERT uses word-piece tokenization and generates a contextualized vector for each word while GloVe encodes a word into a fixed-sized vector representation. Therefore, each pretrained model may perform differently depending on the characteristics of dataset. Importantly, our approach with low resources can outperform pretrained BERT in certain domains, compared to pretrained BERT that requires a significant amount of computational resources and a large-scale dataset. Moreover, SALNet with BERT as a classifier outperforms the original BERT and the self-training with BERT on four benchmark datasets. The BERT in SALNet repeats the fine-tuning process whenever additional pseudo-labeled data is obtained.

**Experiments for few-shot learning.** We conduct experiments to demonstrate the effectiveness of SALNet when there is an extremely small number of labeled data. We randomly select only 0.2% of the original training set and assign 15% of the labeled training set to the development set. As mentioned earlier, bootstrapping methods except for SALNet lead to accumulated classification errors along the semi-supervised learning process and eventually result in low improvement of performance. On the other hand, SALNet shows a stable improvement of performance for few-shot text classification since SALNet employs reliable lexicons, as shown in Table 4. We observe that BERT exhibits the best performance on two datasets (Yahoo! Answer and DBpedia), compared to SALNet using the baseline classifiers such as attention-based LSTM and TextCNN. However, SALNet outperforms the original BERT and the self-training with BERT when we use BERT in SALNet.

Empirically, we confirm the relative robustness of our approach when utilizing 0.2% to 0.9% of the original training...
<table>
<thead>
<tr>
<th>Method</th>
<th>IMDB review</th>
<th>AG News</th>
<th>Yahoo! Answer</th>
<th>DBpedia</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline (attention-based LSTM)</td>
<td>74.35 (3.38)</td>
<td>85.78 (1.18)</td>
<td>49.77 (1.15)</td>
<td>94.41 (0.58)</td>
</tr>
<tr>
<td>Baseline (TextCNN)</td>
<td>75.39 (1.20)</td>
<td>88.08 (0.49)</td>
<td>49.66 (0.36)</td>
<td>96.10 (0.10)</td>
</tr>
<tr>
<td>Self-training (TextCNN)</td>
<td>76.43 (2.42)</td>
<td>88.71 (0.35)</td>
<td>51.35 (0.25)</td>
<td>97.05 (0.10)</td>
</tr>
<tr>
<td>Self-training (attention-based LSTM)</td>
<td>74.97 (4.05)</td>
<td>87.01 (1.19)</td>
<td>51.80 (1.50)</td>
<td>95.80 (0.47)</td>
</tr>
<tr>
<td>Self-training (BERT)</td>
<td>82.53 (4.20)</td>
<td>89.45 (0.21)</td>
<td>56.71 (3.06)</td>
<td>98.38 (0.18)</td>
</tr>
<tr>
<td>Delta-training</td>
<td>77.68 (1.49)</td>
<td>85.31 (0.56)</td>
<td>50.60 (0.46)</td>
<td>97.67 (0.14)</td>
</tr>
<tr>
<td>BERT (fine-tuning with 1% of the total labeled data)</td>
<td>79.74 (3.91)</td>
<td>88.76 (0.18)</td>
<td>57.58 (0.65)</td>
<td>98.01 (0.17)</td>
</tr>
<tr>
<td>VAMPIRE</td>
<td>64.64 (7.60)</td>
<td>85.88 (0.42)</td>
<td>50.57 (1.27)</td>
<td>92.29 (1.65)</td>
</tr>
<tr>
<td>SALNet (attention-based LSTM)</td>
<td>79.00 (2.06)</td>
<td>86.25 (0.96)</td>
<td>51.97 (1.14)</td>
<td>96.62 (0.28)</td>
</tr>
<tr>
<td>SALNet (attention-based LSTM + TextCNN)</td>
<td>80.33 (1.76)</td>
<td>89.23 (0.22)</td>
<td>53.48 (0.81)</td>
<td>97.48 (0.13)</td>
</tr>
<tr>
<td>SALNet (attention-based LSTM + BERT)</td>
<td><strong>84.87</strong> (2.76)</td>
<td><strong>90.35</strong> (0.57)</td>
<td><strong>59.08</strong> (0.76)</td>
<td><strong>98.66</strong> (0.24)</td>
</tr>
</tbody>
</table>

Table 3: Performance (test accuracy (%)) comparison with baselines. Each result is an average over five random samplings with standard deviation in parentheses, and the highest mean result shown in bold.

<table>
<thead>
<tr>
<th>Method</th>
<th>IMDB review</th>
<th>AG News</th>
<th>Yahoo! Answer</th>
<th>DBpedia</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training set (0.2% of total labeled data)</td>
<td>42</td>
<td>204</td>
<td>340</td>
<td>238</td>
</tr>
<tr>
<td>Dev set</td>
<td>8</td>
<td>36</td>
<td>60</td>
<td>42</td>
</tr>
<tr>
<td>Baseline (attention-based LSTM)</td>
<td>66.16 (1.18)</td>
<td>80.26 (3.81)</td>
<td>38.95 (0.87)</td>
<td>82.45 (3.14)</td>
</tr>
<tr>
<td>Baseline (CNN)</td>
<td>63.46 (3.16)</td>
<td>85.67 (0.78)</td>
<td>44.18 (0.67)</td>
<td>91.77 (0.44)</td>
</tr>
<tr>
<td>Self-training (TextCNN)</td>
<td>56.98 (3.87)</td>
<td>86.25 (0.96)</td>
<td>45.57 (1.57)</td>
<td>93.79 (0.50)</td>
</tr>
<tr>
<td>Self-training (attention-based LSTM)</td>
<td>59.65 (6.46)</td>
<td>81.09 (2.54)</td>
<td>41.20 (2.79)</td>
<td>84.30 (3.19)</td>
</tr>
<tr>
<td>Self-training (BERT)</td>
<td>55.25 (4.07)</td>
<td>87.68 (0.64)</td>
<td>46.47 (6.28)</td>
<td>93.69 (0.24)</td>
</tr>
<tr>
<td>Delta-training</td>
<td>55.79 (4.01)</td>
<td>86.18 (0.83)</td>
<td>46.44 (0.59)</td>
<td>92.85 (0.58)</td>
</tr>
<tr>
<td>BERT (fine-tuning with 0.2% of the total labeled data)</td>
<td>59.11 (2.54)</td>
<td>86.51 (0.75)</td>
<td>47.82 (1.36)</td>
<td>97.55 (0.51)</td>
</tr>
<tr>
<td>VAMPIRE</td>
<td>53.58 (7.38)</td>
<td>76.64 (1.68)</td>
<td>41.66 (2.13)</td>
<td>84.47 (1.86)</td>
</tr>
<tr>
<td>SALNet (attention-based LSTM)</td>
<td>69.94 (1.91)</td>
<td>85.59 (1.18)</td>
<td>43.08 (2.27)</td>
<td>92.76 (1.10)</td>
</tr>
<tr>
<td>SALNet (attention-based LSTM + TextCNN)</td>
<td>71.34 (3.22)</td>
<td>87.68 (0.57)</td>
<td>46.59 (1.26)</td>
<td>95.38 (0.71)</td>
</tr>
<tr>
<td>SALNet (attention-based LSTM + BERT)</td>
<td><strong>75.77</strong> (1.08)</td>
<td><strong>88.59</strong> (0.41)</td>
<td><strong>53.65</strong> (0.95)</td>
<td><strong>98.23</strong> (0.12)</td>
</tr>
</tbody>
</table>

Table 4: Experimental results using an extremely small number of labeled data. Each result is an average over five random samplings with standard deviation in parentheses.

On the other hand, if we take less than 0.2% of the original training set, the size of data is too small to contain words that represent each class. Therefore, SALNet constructs the unreliable lexicons, and it can lead to accumulated errors along the training process.

**Analysis**

**Attention mechanism.** Figure 3 visualizes the attention layer of the initially trained attention-based classifier for identifying lexicon in our model: the darker the color, the higher the score. We notice that the initially trained classifier assigns a higher score to important words of the data predicted with high confidence; in other words, the attention mechanism successfully identifies relevant words for text classification using attention scores. This is why the attention mechanism is crucial in our method for effective classification.

**Effectiveness of the constructed lexicons.** In order to label the unlabeled data for semi-supervised learning, we pre-
dict the class of data when the data contains a word that is contained in the lexicon for the class. Table 5 shows the ratio of data predicted by lexicons and the accuracy of the lexicons. We also conduct experiments to verify the effectiveness of the constructed lexicons. Since BERT shows the best performance among the baselines, we use BERT to demonstrate the effectiveness of lexicons. We obtain a new dataset predicted by the lexicons, and select the same number of data from each class to avoid overfitting to a larger class. Then, we update the training dataset with the new dataset and train BERT with the updated dataset. Table 6 demonstrates that the constructed lexicons effectively improve the performance of BERT. We remark that this experiment uses BERT to verify the effectiveness of the constructed lexicons, while the previous experiment uses BERT for bootstrapping learning of SALNet.

Ablation study. We perform ablation studies to show the effectiveness of each component in SALNet. We exclude the lexicon in SALNet to verify the effectiveness of the lexicon. Table 7 shows that the combination of the lexicons and the attention-based LSTM outperforms the model only with attention-based LSTM on four datasets. The effectiveness is especially prominent when labeled data is extremely limited. Since SALNet constructs the lexicons using the data predicted by the classifier with high confidence and the constructed lexicons explicitly predict the unlabeled data using pattern matching, the lexicons improve the reliability of pseudo-labeling. We conduct further experiments to explore the performance according to the number of classifiers in SALNet, and the results are shown in Table 7. Since we can obtain an increased training set that consists of many predicted labels with high confidence if we use two classifiers rather than one for pseudo-labeling, then using two classifiers outperforms only using one classifier in SALNet. Also, there is a difference in the performance of two classifiers since the initial performance of TextCNN outperforms attention-based LSTM on four datasets.

Analysis of failure cases. We manually categorize the resulting errors into two types: type-1, involving short sentences of fewer than ten words, and type-2, which involve ambiguous crucial words from each lexicon. Table 8 shows a few examples of types and errors. From type-1, we can see that the short-sentences examples do not contain crucial words for text classification. In this case, the lexicon contains words that do not belong to the core of the class. For the type-1 errors, we plan to exclude data of less than ten words in the training set. Type-2 errors are caused by ambiguous words of each class. For example, if there are classes such as “artist” and “album”, the lexicon of each class can include “singer”, “sing”, “fan” and “guitar”. For type-2 errors, we plan to count the number of the ambiguous words in each class and assign a weight to keywords according to the confidence value of the classifier.

Related Work

Most of the prior works on SSL for text classification has been covered the type of bootstrapping, adversarial training as well as the type of pretraining. In this section, we introduce several studies for semi-supervised learning discussed in our paper.

Bootstrapping Algorithms

The bootstrapping methods such as self-training (Yarowsky 1995), co-training (Blum and Mitchell 1998), tri-training (Zhu 2005) leverage the predictions of the neural network model on unlabeled data to obtain additional information that can be used during training.
Table 8: Examples of errors by the constructed lexicons. N: None, E: Electronic, O: Office, Ar: Artist, Al: Album, M: Music, C: Culture, S: Society, T: Transportation, At: Athlete

<table>
<thead>
<tr>
<th>Type</th>
<th>Example</th>
<th>Predicted</th>
<th>Ground Truth</th>
<th>Matching words</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type-1</td>
<td>where can sony ericsson? Antoinette Spaak Antoinette M.</td>
<td>N</td>
<td>E</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>Ronald Isley Ronald Isley is an American recording artist songwriter record producer and occasional actor.Isley is better known as the lead singer and founding member of the family music group the Brothers.</td>
<td>Ar</td>
<td>Al</td>
<td>artists, songwriter, record, producer</td>
</tr>
<tr>
<td>Type-2</td>
<td>doswell is acity in virginia? Yep, it’s a city in Virginia. It is a small, unincorporated town that’s doesn’t have too much, but it is located close to Richmond. You can find out all about King’s Dominion on their website about to host King’s Fest, a large Christian music concert featuring artists like the David Crowder Band.</td>
<td>M</td>
<td>C</td>
<td>concert, band, music, artists</td>
</tr>
</tbody>
</table>

One major weakness of the bootstrapping is that if the initially trained models have low performance, then it can lead to accumulated errors along the process. Several attempts (Abney 2008; Sogaard 2010) have been made to overcome the limitations. Recently, Jo et al. (2019) propose a variation of self-training framework for semi-supervised text classification. The method stems from the hypothesis that a classifier with pretrained word embedding always outperforms the same classifier with randomly initialized word embedding. Our approach and delta-training are based on a self-training framework. Therefore, we use delta-training and self-training as baselines to verify the excellence of our framework.

**Adversarial Training Algorithms**

The adversarial training is a technique of improving model performance by augmenting adversarial examples in the training process. Miyato et al. (2017) proposed a virtual adversarial training approach to smooth the output distributions of the neural networks on straight-forward classification tasks. They extended adversarial and virtual adversarial training to the text domain by applying perturbations to the word embeddings. Gururangan et al. (2019) introduced a lightweight pretraining framework for effective text classification when data and computing resources are limited. They pretrained a uni-gram document model as a variational autoencoder (VAE) on unlabeled data and used it for the downstream classifier. They demonstrated the effectiveness of the model for limited resource settings, without the need for computationally demanding. The adversarial training is not related to our framework. However, the adversarial training has the same goal of solving the data-sparcity problem for text classification. Therefore, we use VAMPIRE, a state-of-the-art model of the adversarial training for the experimental comparison.

**Pretraining Algorithms**

The pretraining algorithms transfer knowledge from rich-resource pretraining task to the low downstream tasks. Language Models (LMs) such as OpenAI Transformer (Vaswani et al. 2017) and BERT (Devlin et al. 2019) have achieved state-of-the-art performance on many classification tasks. The methods show excellent performance even with small number of labeled data. However, LMs require significant computational resources to train at a high scale. Unlike the two previous algorithms, pretraining algorithms are more related to unsupervised learning than semi-supervised learning. The pretraining model applies to other semi-supervised methods to improve performance. One of the latest adversarial learning, VAMPIRE, utilizes the pretraining model and has improved performance. Since the pretraining model can also be applied to the bootstrapping algorithms, we use the representative LMs, BERT, as a baseline, and show the improvement of performance when BERT is integrated into our framework, SALNet.

**Conclusions and Future Work**

We propose a simple, yet effective semi-supervised bootstrap learning framework for few-shot text classification, which takes full advantage of both only a small number of labeled data and a large number of unlabeled data. Our framework generates a lexicon using the attention mechanism, and we use the constructed lexicons for pseudo-labeling. The lexicon can select the correct label among the predictions of the classifier and correctly predict the unlabeled data that a model incorrectly predicts. Extensive experimental results demonstrate that our method achieves superior performance to the previous state-of-the-art methods.

We plan to develop methods that generate lexicons using the pretrained language model such as XLNet (Yang et al. 2019) and ALBERT (Lan et al. 2020). We also plan to study assigning weights to crucial words to enhance lexicon performance.

**Acknowledgments**

This research was supported by the NRF grant funded by MIST (NRF-2020R1A4A3079947) and the AI Graduate.
School Program (2020-0-01361). Han is a corresponding author.

References


