Attentive Pooling with Learnable Norms for Text Representation

Chuhan Wu[†], **Fangzhao Wu**[‡], **Tao Qi**[†], **Xiaohui Cui**[§], **Yongfeng Huang**[†]
[†]Department of Electronic Engineering & BNRist, Tsinghua University, Beijing 100084, China
[‡]Microsoft Research Asia, Beijing 100080, China
[§]School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China.
{wuchuhan15, wufangzhao}@gmail.com qit16@mails.tsinghua.edu.cn
xcui@whu.edu.cn yfhuang@tsinghua.edu.cn

Abstract

Pooling is an important technique for learning text representations in many neural NLP models. In conventional pooling methods such as average, max and attentive pooling, text representations are weighted summations of the $\mathcal{L}^1$ or $\mathcal{L}^\infty$ norm of input features. However, their pooling norms are always fixed and may not be optimal for learning accurate text representations in different tasks. In addition, in many popular pooling methods such as max and attentive pooling some features may be over-emphasized, while other useful ones are not fully exploited. In this paper, we propose an **A**ttentive **P**ooling with **L**earnable **N**orms (APLN) approach for text representation. Different from existing pooling methods that use a fixed pooling norm, we propose to learn the norm in an end-to-end manner to automatically find the optimal ones for text representation in different tasks. In addition, we propose two methods to ensure the numerical stability of the model training. The first one is scale limiting, which re-scales the input to ensure non-negativity and alleviate the risk of exponential explosion. The second one is re-formulation, which decomposes the exponent operation to avoid computing the real-valued powers of the input and further accelerate the pooling operation. Experimental results on four benchmark datasets show that our approach can effectively improve the performance of attentive pooling.

1 Introduction

In recent years, neural network based methods are widely used in the natural language processing (NLP) field to learn text representations (Yang et al., 2016; Peters et al., 2018). In these methods, pooling is a core technique to build the text representation vector from a collection of input feature vectors by summarizing their information (Lai et al., 2015). Thus, an effective pooling method that can select salient features accurately will facilitate many NLP methods (Ma et al., 2017).

Sentiment Classification	
Average Pooling	The movie is good, but not to my taste
Max Pooling	The movie is good, but not to my taste
Attentive Pooling	The movie is good, but not to my taste
News Topic Classification	
Average Pooling	Fire on Queensland Island Takes Heavy Toll on Wildlife
Max Pooling	Fire on Queensland Island Takes Heavy Toll on Wildlife
Attentive Pooling	Fire on Queensland Island Takes Heavy Toll on Wildlife

Figure 1: The pooling weights of several different pooling methods on the representations produced by an LSTM network. Darker colors indicate higher weights.

Among existing pooling methods, average pooling is a representative one which takes the average of the $\mathcal{L}^1$ norm of input features (Tang et al., 2014, 2015a,b). However, average pooling equally regards the input representation vector at each position and ignores their different informativeness for learning text representation, which may not be optimal (Johnson and Zhang, 2015). Thus, other pooling methods such as max pooling (Collobert et al., 2011; Kim, 2014) and attentive pooling (Yang et al., 2016; Zhou et al., 2016; Cui et al., 2017; Devlin et al., 2019; Wu et al., 2019b) are widely used in neural NLP models. For example, Kim (2014) proposed to apply max pooling to the contextual word representations learned by CNN networks to build the representations of the entire sentence. Yang et al. (2016) proposed to use attentive pooling at both word and sentence levels to learn informative sentence and document representations by selecting important words and sentences. However, these pooling methods use fixed average norms, i.e., $\mathcal{L}^1$ norm for average and attentive pooling and $\mathcal{L}^\infty$ norm for max pooling, to build text representations, which may not be optimal when handling different tasks.

Our work is motivated by the following obser-

Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 2961–2970
July 5 - 10, 2020.

vations. First, different contexts usually have different informativeness for learning text representations. For example, in Fig. 1[1], the word "but" is very important for inferring the sentiment polarity of this sentence, while "The" is uninformative. Thus, modeling the different informativeness of contexts and attending to them differently may help learn more informative text representations. Second, different tasks and even different datasets have different characteristics. For example, in Fig. 1, sentiment and negation words may be the key clues for inferring the sentiment polarity of the first sentence, while the global contexts may be useful for understanding the topic of the second sentence. Thus, using a fixed pooling norm for universal text representation learning is probably not optimal. Third, in popular pooling methods such as max pooling and attentive pooling, some contexts may be over-emphasized, and other useful contextual information is not fully-respected. For example, as shown in Fig. 1, the sentiment word "good" is highlighted, but other useful clues such as "but" and "not" do not gain sufficient attentions, which may not be optimal for learning accurate text representations. Thus, a dynamically learnable degree of "hard" or "soft" for pooling may benefit text representation learning.

In this paper, we propose an Attentive Pooling with Learnable Norms (APLN) approach to enhance the learning of text representations[2]. Instead of manually setting a fixed pooling norm, we propose to automatically learn it in a unified framework, which can find the optimal values to learn text representations for different tasks in an end-to-end manner. In addition, since the learning of pooling norm may be numerically unstable in some cases due to the exponent operation, we propose two methods to improve its computational stability. The first one is limiting the scale of input features, which aims to ensure their non-negativity and avoid exponential explosion. The second one is a re-formulation method, which aims to avoid computing the real-valued power of input features by decomposing the exponent operation into three safe and fast atomic operations. We conducted experiments on four benchmark datasets, and the results show that our approach can effectively improve the learning of text representation.

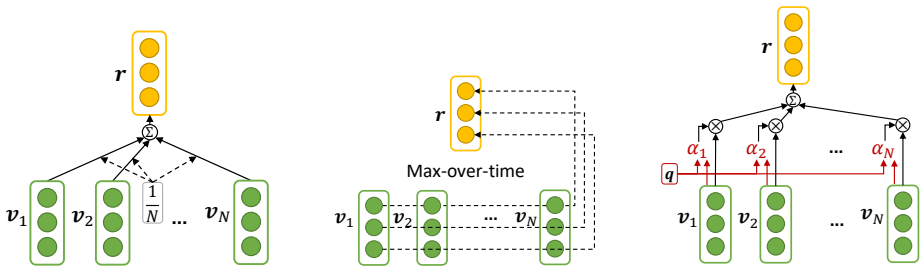
[1]The visualized weights of max pooling are summations of the maximum elements over time for each word.

[2]https://github.com/wuch15/ACL2020-APLN

2 Related Work

Neural networks are widely used to learn text representations from contexts (Peng et al., 2018). Pooling is usually an essential step in these methods to build contextual representations by summarizing the information of input features (LeCun et al., 2015). The simplest pooling method is average pooling, which is used in many approaches to construct text representations (Tang et al., 2014, 2015a,b). For example, Tang et al. (2015a) proposed to apply average pooling to the output of CNN filters to capture global contexts in a sentence. In addition, they also proposed to average the sentence representations learned by parallel CNN networks with different window sizes. In their another work (Tang et al., 2015b), they proposed to apply average pooling to the sequence of sentence representations to build the representations of an entire document. Although average pooling is computationally efficient, it cannot distinguish important contexts from unimportant ones, which may not be optimal for learning accurate text representations.

There are also other popular pooling methods that can select salient features to learn more informative text representations, such as max pooling (Kim, 2014; Zhang et al., 2015) and attentive pooling (Yang et al., 2016), which are employed by many neural NLP methods (Collobert et al., 2011; Kim, 2014; Huang et al., 2012; Yang et al., 2016; Chen et al., 2016; Zhou et al., 2016; Du et al., 2017; Li et al., 2018; Wu et al., 2019a; Tao et al., 2019; Devlin et al., 2019; Wu et al., 2019b). For example, Collobert et al. (2011) proposed to learn representations of contexts within each window using feed forward neural networks, and used max pooling to build final text representations. Kim (2014) proposed to apply max pooling over time to the contextual word representations learned by multiple CNN filters. Huang et al. (2012) proposed to build representations of the entire document using the summation of word representations weighted by their TF-IDF scores. Yang et al. (2016) proposed a hierarchical attention network to first learn sentence representations from words and then learn document representations from sentences. They proposed to apply attentive pooling at both word and sentence levels to select informative words and sentences for more informative representation learning. Wu et al. (2019b) proposed a hierarchical user and



(a) Average pooling. (b) Max pooling. (c) Attentive pooling.

Figure 2: Comparisons of several popular pooling methods.

item representation model with three-tier attention, which applies attentive pooling to simultaneously select important words, sentences and reviews. However, the pooling norms of max and attentive pooling are always fixed, which may not be optimal for universal text representation learning since the characteristics of different tasks may be different. In addition, both pooling methods may over-emphasize the most salient features, and other useful contextual information is not fully exploited, which may also be sub-optimal. There are a few methods to adapt the pooling norms in different tasks. For example, Gulcehre et al. (2014) explored the influence of selecting different pooling norms on the performance of different image classification tasks. However, the norms in their method are manually tuned, which are usually very time-consuming and may not be optimal. Different from all aforementioned methods, our approach can automatically optimize pooling norms in an end-to-end manner, and can effectively select important contexts to learn informative text representations. Extensive experiments on four datasets with different characteristics validate the effectiveness of our approach.

3 Preliminaries

In this section, we will first present a brief introduction to several popular pooling methods, i.e., average, max and attentive pooling. To make it easier to understand, we present an intuitive comparison of the mechanisms of these different pooling methods in Fig. 2.

Average Pooling. Average pooling is used to build contextual representations by taking the arithmetic mean of input features, as shown in Fig. 2(a). It uses the $\mathcal{L}^1$ norm of the input. Denote the input sequence of hidden representations as $[\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_N]$, where N is the sequence length. The output representation is computed as:

$$\mathbf{r} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{h}_i. \tag{1}$$

Max Pooling. Max pooling aims to build contextual representations by selecting the most salient features via max-over-time operations, as shown in Fig. 2(b). It utilizes the $\mathcal{L}^\infty$ norm at the time dimension of input features. Denote $\mathbf{r}^j$ as the j-th value in the vector $\mathbf{r}$, which is computed as:

$$\mathbf{r}^j = \max(\mathbf{h}_1^j, \mathbf{h}_2^j, ..., \mathbf{h}_N^j), \tag{2}$$

where $\mathbf{h}_i^j$ represents the j-th value in the feature vector $\mathbf{h}_i$.

Attentive Pooling. As shown in Fig. 2(c), attentive pooling usually builds contextual representations by selecting important input features, which can also be regarded as a kind of $\mathcal{L}^1$ norm average. It computes an attention weight α_i for the input at each position to indicate its informativeness, which is formulated as follows:

$$\alpha_i = \frac{\exp[\mathbf{q}^T f(\mathbf{h}_i)]}{\sum_{j=1}^{N}\exp[\mathbf{q}^T f(\mathbf{h}_j)]}, \tag{3}$$

where $f(\cdot)$ is a non-linear function, $\mathbf{q}$ is the attention query vector. Following Yang et al. (2016), we apply the $\tanh$ operation to the linear transformation of $\mathbf{h}_i$ to form the function $f(\cdot)$. The final contextual representation $\mathbf{r}$ is the summation of input representation vectors weighted by their attention weight as follows:

$$\mathbf{r} = \sum_{i=1}^{N}\alpha_i\mathbf{h}_i. \tag{4}$$

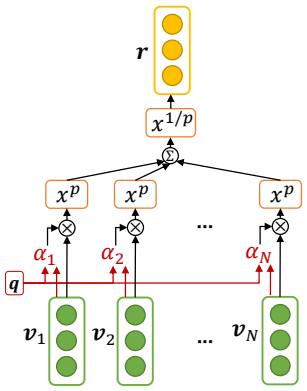


Figure 3: Architecture of our Attentive Pooling with Learnable Norms (APLN) approach.

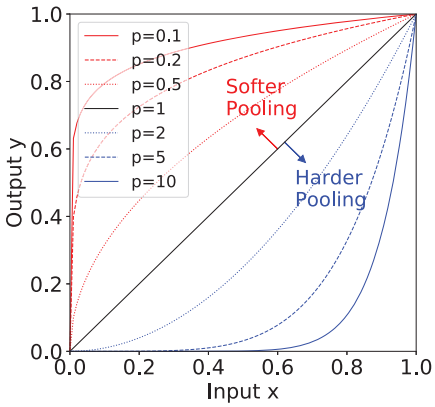


Figure 4: Illustration of the influence of p on the shape of the function $y = x^p$.

4 Attentive Pooling with Learnable Norms

In this section, we will introduce the details of our Attentive Pooling with Learnable Norms (APLN) approach. In the aforementioned pooling methods, the pooling norm is always fixed (i.e., $\mathcal{L}^1$ or $\mathcal{L}^\infty$). However, the characteristics of different NLP tasks and even different datasets should have some differences, and it may not be optimal to use a fixed pooling norm for universal text representation learning. In addition, tuning the pooling norm manually is usually very time-consuming, and it may also be sub-optimal. Thus, it is an intuitive idea to automatically learn the pooling norm in an end-to-end manner to alleviate the efforts on hyperparameter searching and learn more informative text representations. The architecture of our *APLN* approach is shown in Fig. 3. We will introduce its details as follows.

Since different contexts usually have different importance, modeling their informativeness may help learn more informative text representations. Thus, similar to the vanilla attentive pooling, in our *APLN* approach, we also compute an attention score for the input at each position. However, instead of using the simple weighted summation to build the contextual representation $\mathbf{r}$, we propose to compute the $\mathcal{L}^p$ norm[3] average of the input feature vectors weighted their attention weights, which is formulated as follows:

$$\mathbf{r} = [\frac{1}{\sum_{i=1}^{N} \alpha_i^p} \sum_{i=1}^{N} (\alpha_i \mathbf{h}_i)^p]^{\frac{1}{p}}, \quad (5)$$

[3]It should be noticed that when $p < 1$, this definition is not a norm since it does not obey the triangle inequality. But we still call it "norm" for consistency.

where p is a learnable parameter. In this way, our model will automatically find appropriate values of pooling norms for learning text representations in different tasks.

To show the influence of p on the inputs of the *APLN* module, we vary the value of p and illustrate the shape of the function $y = x^p$ in Fig. 4. According to Fig. 4, we can see when p is larger, the attention of *APLN* is sharper and sparser since small values of $\alpha_i \mathbf{h}_i$ will be suppressed, which indicates the attentive pooling is "harder". In contrast, if p is smaller, the attentions are more distributed, which indicates the attentive pooling is "softer". Thus, in this manner, our *APLN* model can automatically explore how "hard/soft" the attention should be when constructing text representations, which may help recognize important contexts and avoid the problem of over-emphasizing some features and not fully respecting other useful ones, both of which are important for learning accurate text representations.

Unfortunately, in most cases the training of *APLN* is unstable if we directly use it for pooling. Thus, we propose two methods to ensure the numerical stability of the model training. The first one is *scale limiting*, which is used to limit the range of the elements of $\alpha_i \mathbf{h}_i$. The second one is *Re-formulation*, which is used to avoid the direct computation of the real-valued powers of the input features and accelerate the pooling operation. We will introduce the two methods as follows.

4.1 Scale Limiting

According to Eq. (5), to ensure the values of $\mathbf{r}$ are real, the elements of $\alpha_i \mathbf{h}_i$ must be non-negative. Thus, we apply a ReLU function to $\alpha_i \mathbf{h}_i$ to keep

$\alpha_i \mathbf{h}_i \geq 0$. However, there are still some risks if there exist elements with $\alpha_i \mathbf{h}_i^j > 1$, since the gradients may explode when $p > 1$ due to the amplification of the exponent, which are also observed in our practice. To solve this problem, we propose to clip the values of $\alpha_i \mathbf{h}_i$ as follows:

$$0 \leq \alpha_i \mathbf{h}_i^j \leq 1. \quad (6)$$

In this way, the input features is re-scaled to a "safe" range. We also explored other kinds of re-scaling methods such as normalization, but we find there are no significant differences in the model performance. Thus, we simply use the clipping operation for its efficiency.

4.2 Re-formulation

However, there are still some problems in our approach. We find the training of our approach is not numerically stable (e.g., NAN problem) when implemented by several popular deep learning frameworks such as Tensorflow. In addition, computing the real-value powers of input features is quite time-consuming. Thus, we propose a re-formulation strategy by converting the exponent computation in Eq. (5). For instance, the exponent x^p is re-formulated as follows:

$$x^p = e^{\log(x^p)} = e^{p \log(x)} \approx e^{p \log(x+\epsilon)}, \quad (7)$$

where $\epsilon = 10^{-7}$ is a protection value. In this way, the computation of the power of x is divided into three atomic operations, i.e., logarithm, multiplication and exponent, all of them are fast[4] and numerically stable in our approach. Thus, using the re-formulation strategy can enhance the numerical stability and accelerate the pooling operation.

5 Experiments

5.1 Datasets and Experimental Settings

Our experiments are widely conducted on four benchmark datasets with different characteristics. The first one is *AG's News*[5], which is a news topic classification dataset. Following (Zhang et al., 2015), we only use the title and description fields in this dataset. The second one is *IMDB*[6] (Diao et al., 2014), which is a dataset with movie reviews and ratings. The third one is *Amazon_Electronics* (denoted as *Amazon*) (He and McAuley, 2016), which contains reviews on electronics. The fourth one is *Yelp 2015* (denoted as *Yelp*), which is a restaurant review dataset. The latter three datasets are all for sentiment classification. Since the original *Amazon* and *Yelp* datasets are too large, we sampled 50,000 reviews to form each dataset. The detailed statistics are shown in Table 1. The class distributions of the *AG's News* and *Yelp* are balanced, but are imbalanced on *IMDB* and *Amazon*, as shown in Fig. 5. In addition, *AG's News* is a sentence-level classification dataset, while the others are document-level. Since the *AG's News* dataset only contains the training and test sets, we randomly sampled 10% of news in the training set for validation. For the other three datasets, we used 80% of samples for training, 10% for validation and the rest 10% for test.

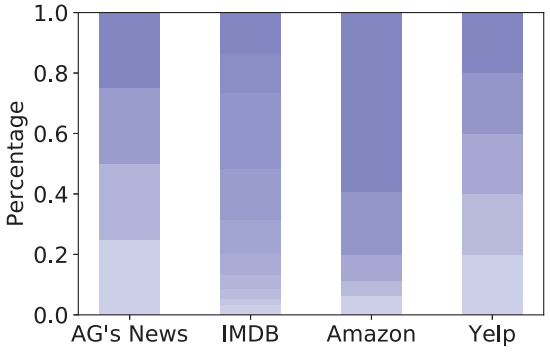


Figure 5: Class distributions of datasets. For *IMDB*, *Amazon* and *Yelp*, darker colors indicate higher ratings.

Dataset	# Train	# Val.	# Test	# Classes	Balanced
AG's News	108,000	12,000	7,600	4	✓
IMDB	108,535	13,567	13,567	10	×
Amazon	40,000	5,000	5,000	5	×
Yelp	40,000	5,000	5,000	5	✓

Table 1: Statistics of our datasets.

In our experiments, the word embeddings were 300-dimensional and initialized by Glove (Pennington et al., 2014)[7]. In our comparative experiments, the CNN networks had 400 filters, and their window size was 3. The dimension of LSTM hidden states was 200. The attention query vectors were 200-dimensional. The initial pooling norm p was set to 1, which is consistent with the vanilla attentive pooling. Adam (Kingma and Ba, 2014) was used as the optimizer, and the

[4] In experiments on a machine with a GTX1080ti GPU, the computation of x^p is accelerated by more than 10 times.

[5] https://www.di.unipi.it/en/

[6] https://github.com/nihalb/JMARS

[7] We do not use language models such as ELMo and BERT since our work focuses on facilitating the pooling technique rather than boosting the performance of our approach against the state-of-the-art methods.

Methods	AG's News		IMDB		Amazon		Yelp	
	Accuracy	Macro-F	Accuracy	Macro-F	Accuracy	Macro-F	Accuracy	Macro-F
CNN-Avg	91.55	91.52	49.96	38.88	64.73	36.68	55.41	54.78
CNN-Max	92.10	92.07	50.53	40.96	66.24	43.80	59.19	59.14
CNN-Att	92.32	92.30	51.24	42.24	66.79	44.01	59.22	59.19
CNN-APLN	**92.48**	**92.45**	51.63	43.57	66.86	45.80	59.97	59.95
LSTM-Last	91.65	91.62	48.96	38.32	64.55	39.62	55.20	54.88
LSTM-Avg	91.10	91.07	48.65	38.67	62.09	40.09	55.76	54.92
LSTM-Max	92.01	91.99	50.94	40.94	66.80	43.63	59.63	59.26
LSTM-Att	92.20	92.18	51.12	41.83	67.07	43.70	59.87	59.44
LSTM-APLN	92.45	92.43	51.77	43.65	67.39	45.55	60.21	60.01
HAN	-	-	52.05	42.81	67.22	45.01	60.18	59.72
HAN-APLN	-	-	**52.59**	**44.01**	**67.95**	**46.01**	**60.55**	**60.35**

Table 2: The performance of different methods on the four benchmark datasets.

batch size was 64. We applied dropout (Srivastava et al., 2014) techniques to the word embeddings, CNN networks or LSTMs to mitigate overfitting, and the dropout ratio was 0.2. These hyperparameters were tuned on the validation set. In classification tasks the metrics were accuracy and macro-F scores, and in regression tasks the performance was evaluated by rooted mean squared error (RMSE). We reported the average results of 10 independently repeated experiments.

5.2 Performance Evaluation

We compare the performance of different neural text classification models with different pooling methods to evaluate the performance of our approach. The methods to be compared include: (1) *CNN-Avg* (Tang et al., 2015b), applying average pooling to the representations learned by CNN to build contextual text representations; (2) *CNN-Max* (Kim, 2014), using a combination of CNN and max pooling; (3) *CNN-Att* (Gong and Zhang, 2016), using a combination of CNN and vanilla attentive pooling; (4) *CNN-APLN*, combining CNN with our *APLN* approach; (5) *LSTM-Last* (Hochreiter and Schmidhuber, 1997), using the last hidden state in an LSTM network; (6) *LSTM-Avg* (Zhao et al., 2016), using average pooling after LSTM; (7) *LSTM-Max* (Johnson and Zhang, 2016), using max pooling after LSTM; (8) *LSTM-Att* (Zhou et al., 2016), using attentive pooling after LSTM; (9) *LSTM-APLN*, combining LSTM with *APLN*; (10) *HAN* (Yang et al., 2016), a hierarchical LSTM network with both word-level and sentence-level attentive pooling; (11) *HAN-APLN*, using *APLN* at both word and sentence levels. In methods based on LSTM, we used two parallel LSTMs to scan the input in both directions. The results of these methods are summarized in Table 2, which reveal several findings.

First, the methods based on average pooling are usually inferior to those using other pooling methods in our experiments. This is probably because average pooling equally regards different features and cannot distinguish their informativeness. Thus, modeling the importance of different features has the potential to improve text representation learning. Second, the methods based on attentive pooling outperform their variants based on max pooling. This may be because attentive pooling can model the informativeness of contexts for text representation, while max pooling only selects the most salient features, which may be sub-optimal. Third, our *APLN* approach can consistently outperform other pooling methods, and further hypothesis test results show that the improvement brought by our approach is significant ($p < 0.01$). This may be because vanilla max pooling and attentive pooling methods use a fixed pooling norm for universal text representation learning, and the differences in the characteristics of different tasks and datasets are not considered, which may also be sub-optimal. Our approach can dynamically adapt the pooling norm in different scenarios, which may facilitate text representation learning. In addition, we find the advantage in Macro-F score of our approach over other methods is more significant on the datasets with imbalanced class distributions. This may be because our approach can build text representation in a softer manner, which may help neural models avoid focusing on the clues of major classes only and alleviate their dominance. Fourth, we find hierarchical models (*HAN* and *HAN-APLN*) outperform flatten models (e.g., *LSTM-APLN*) for doc-

Methods	IMDB	Amazon	Yelp
CNN-Avg	1.388	0.920	0.847
CNN-Max	1.322	0.908	0.834
CNN-Att	1.292	0.899	0.824
CNN-APLN	1.271	0.886	0.801
LSTM-Last	1.316	0.896	0.822
LSTM-Avg	1.343	0.911	0.830
LSTM-Max	1.269	0.890	0.815
LSTM-Att	1.257	0.878	0.799
LSTM-APLN	1.233	0.865	0.784
HAN	1.230	0.866	0.789
HAN-APLN	**1.214**	**0.858**	**0.776**

Table 3: The performance of different methods on rating regression. Lower RMSE scores indicate better performance.

ument representation learning. This may be because modeling documents in a hierarchical manner can better utilize the structure of documents. In addition, since our approach can be applied at both word and sentence levels in *HAN*, text representation may be learned more accurately. These results validate the effectiveness of our approach.

To further validate the generality of our approach in regression tasks[8], we also conduct experiments on the *IMDB*, *Amazon* and *Yelp* datasets by formulating the task as a rating regression problem, and the results in terms of RMSE are shown in Table 3. From the results, we find our *APLN* approach can also bring consistent improvements to many existing methods in the regression task.

5.3 Influence of Scale Limiting and Re-formulation

In this section, we will explore the influence of the scale limiting and re-formulation techniques on the stability and relative pooling speed of our approach. The results are summarized in Table 4. From these results, if the limitation of non-negativity is removed, the model training is usually unstable, which is intuitive. In addition, if the scale limitation (≤ 1) is removed, our model occasionally does not converge. This may be because when $p > 1$, our model has the risk of gradient explosion. Thus, the scale of input features should be limited. Besides, the re-formulation method also has critical impacts on our approach. This is probably because directly computing the real-valued exponents of input features may be numerically unstable. In our approach we decompose the exponents into three stable operations, which is robust to numerical errors. In addition, the pooling speed can be effectively improved, since the computational costs of these atomic operations are usually small. These results validate the effectiveness of our approach.

[8]We find that the regression labels need to be normalized, or the performance may be sub-optimal.

	Stability	Speed
-SL (≥ 0)	$\times$	1.001
-SL (≤ 1)	$\circ$	1.001
-RF	$\times$	0.116
APLN	✓	1.000

Table 4: Influence of the scale limiting (abbreviated as *SL*) and re-formulation (abbreviated as *RF*) on the stability and relative pooling speed of *APLN*. The symbol $\circ$ represents the model training is unstable on occasion.

5.4 Influence of Norm Initialization

In this section, we study the influence of a small but very important step, i.e., the initialization of the trainable pooling norm p, on the performance of our approach. We compare the performance of *LSTM-APLN* by varying the initialized values of p. The results are shown in Fig. 6. From Fig. 6, we find the performance of our approach increases when the initialized value of p increases. This is intuitive because when p is too small, the attention network may not be capable of recognizing important contexts effectively, which is not optimal for learning accurate text representations. In addition, when p is initialized with a too large value, the performance will start to decline. This is probably because a large value of p will lead to sharp attentions on critical contexts, and other useful information is not fully exploited. Thus, the performance is also not optimal. These results show that a moderate value (e.g., 1.0) is the most appropriate for initializing the pooling norm p, which is also consistent with standard attentive pooling.

5.5 Parameter Analysis

In this section, we analyze a critical parameter learned by our model, i.e., the pooling norm p in the *APLN* module. The evolution of the values of p learned by *LSTM-APLN* on the four benchmark datasets during model training is portrayed in Fig. 7. From the results, we have several interesting observations. First, the pooling norms learned by our model are consistently less than

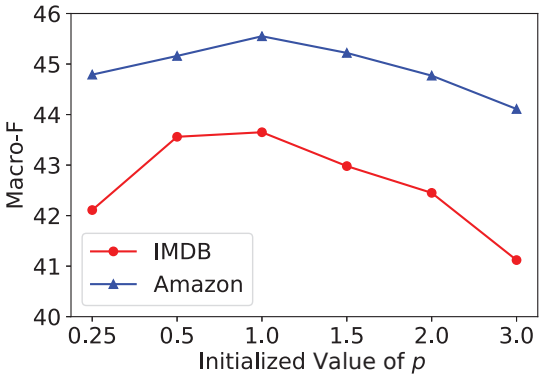


Figure 6: The influence of the initialization of the pooling norm p on our approach.

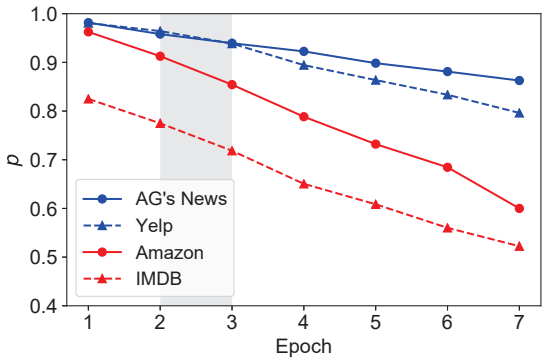
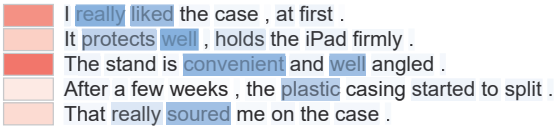
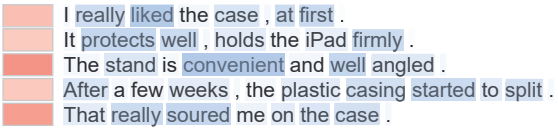


Figure 7: The evolution of the pooling norm p learned by our model on different datasets. The required training epochs to achieve the best performance are marked as the grey region.

1, which indicates that our norm-wise attention is "softer" than vanilla attention. This may be because L1 norm is not optimal for attentive pooling, and a softer attention manner may be more suitable for learning accurate text representations. Second, we find it is interesting that the norm p consistently decreases when the training epoch increases. This may be because the model may tend to take the global contexts into consideration rather than focus on important ones. Third, a moderate norm p is more appropriate for our approach. This may be because when p is too large, the attentions may be too sparse and useful contextual information is not fully exploited. When p is too small, the attention networks cannot effectively distinguish informative contexts from uninformative ones, which may also be sub-optimal for learning text representations. Fourth, we observe that the norm p learned on datasets with imbalanced class distributions is lower than those with balanced distributions. This may be because on imbalanced dataset, if p is too large, the clues



(a) Attention weights in *HAN*. Predicted rating is 4.



(b) Attention weights in *HAN-APLN*. Predicted rating is 3.

Figure 8: Visualization of the word-level and sentence-level attention weights in *HAN* and *HAN-APLN* on a randomly selected review in the *Amazon* dataset, whose gold rating score is 3. Darker colors indicate higher attention weights. The visualized attention weights of *APLN* are α_i^p of words and sentences, under $p = 0.885$ at the word level and $p = 0.892$ at the sentence level.

of the majority classes may be over-emphasized, and other useful information is not fully respected. Thus, the performance of our *APLN* approach is better when it learns a moderate pooling norm.

5.6 Case Study

In this section, we conducted several case studies to further explore the effectiveness of our *APLN* approach. We visualize the word-level and sentence-level attention weights in *HAN* and *HAN-APLN* of a randomly selected review to compare their differences, and the results are portrayed in Fig. 8. According to the results, we have several observations. First, both *HAN* and *HAN-APLN* can recognize important words and sentences. For example, the word "liked" and the sentence "I really liked the case, at first." are highlighted since they are important for modeling the opinions condensed by this review. Second, the attentions of *HAN* are sparse, which indicates that *HAN* tends to focus more on some contexts in a review such as the first and the third sentence, and pays little attentions to the useful information in other contexts such as the fourth and fifth sentences. In addition, *HAN* wrongly classifies the rating of this review. This is probably because the rating of a review is usually a synthesis of all opinions conveyed by it. Thus, it may not be optimal for learning accurate text representations if only salient contexts are considered. Third, different from *HAN*, the attentions of *HAN-APLN* are smoother. This is probably because the pooling norm learned by our approach is less than 1, which encourages our model

to attend to important contexts in a softer manner. In addition, *HAN-APLN* can classify this review correctly. This is probably because our approach can effectively take global contextual information into consideration, and does not over-emphasize critical contexts. Thus, our *APLN* approach can learn more accurate text representations than the methods based on vanilla attentive pooling. These results show the effectiveness of our approach.

6 Conclusion and Future Work

In this paper, we propose an Attentive Pooling with Learnable Norms (APLN) approach for text representation. Instead of using a fixed pooling norm for universal text representation learning, we propose to learn the norm in an end-to-end framework to automatically find the optimal ones for learning text representations in different tasks. In addition, we propose two methods to ensure the numerical stability of the model training. The first one is scale limiting, which limits the scale of input representations to ensure their non-negativity and avoid potential exponential explosion. The second one is re-formulation, which decomposes the exponent operation into several safe atomic operations to avoid computing the real-valued powers of input features with less computational cost. Extensive experiments on four benchmark datasets validate the effectiveness of our approach.

In our future work, we will explore several potential directions. First, we plan to explore why the model prefers "soft" attentions rather than "hard" ones, which is different from the findings in several prior works based on hard attention. Second, we plan to study how to model the differences on the characteristics of different samples and use different pooling norms, which may have the potential to further improve our approach. Third, we will explore how to generalize our approach to other modalities, such as images, audios and videos, to see whether it can facilitate more attention-based methods.

Acknowledgments

This work was supported by the National Key Research and Development Program of China under Grant number 2018YFC1604002, the National Natural Science Foundation of China under Grant numbers U1936208, U1936216, U1836204, and U1705261.

References

Huimin Chen, Maosong Sun, Cunchao Tu, Yankai Lin, and Zhiyuan Liu. 2016. Neural sentiment classification with user and product attention. In *EMNLP*, pages 1650–1659.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(Aug):2493–2537.

Yiming Cui, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu. 2017. Attention-over-attention neural networks for reading comprehension. In *ACL*, pages 593–602.

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

Qiming Diao, Minghui Qiu, Chao-Yuan Wu, Alexander J Smola, Jing Jiang, and Chong Wang. 2014. Jointly modeling aspects, ratings and sentiments for movie recommendation (jmars). In *KDD*, pages 193–202. ACM.

Jiachen Du, Lin Gui, Ruifeng Xu, and Yulan He. 2017. A convolutional attention model for text classification. In *NLPCC*, pages 183–195. Springer.

Yuyun Gong and Qi Zhang. 2016. Hashtag recommendation using attention-based convolutional neural network. In *IJCAI*, pages 2782–2788.

Caglar Gulcehre, Kyunghyun Cho, Razvan Pascanu, and Yoshua Bengio. 2014. Learned-norm pooling for deep feedforward and recurrent neural networks. In *ECML-PKDD*, pages 530–546. Springer.

Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *WWW*, pages 507–517.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. 2012. Improving word representations via global context and multiple word prototypes. In *ACL*, pages 873–882.

Rie Johnson and Tong Zhang. 2015. Semi-supervised convolutional neural networks for text categorization via region embedding. In *NIPS*, pages 919–927.

Rie Johnson and Tong Zhang. 2016. Supervised and semi-supervised text categorization using lstm for region embeddings. In *ICML*, pages 526–534.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*, pages 1746–1751.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *AAAI*.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature*, 521(7553):436.

Zheng Li, Ying Wei, Yu Zhang, and Qiang Yang. 2018. Hierarchical attention transfer network for cross-domain sentiment classification. In *AAAI*.

Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng Wang. 2017. Interactive attention networks for aspect-level sentiment classification. In *AAAI*, pages 4068–4074.

Hao Peng, Jianxin Li, Yu He, Yaopeng Liu, Mengjiao Bao, Lihong Wang, Yangqiu Song, and Qiang Yang. 2018. Large-scale hierarchical text classification with recursively regularized deep graph-cnn. In *WWW*, pages 1063–1072.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL-HLT*, pages 2227–2237.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, 15(1):1929–1958.

Duyu Tang, Bing Qin, and Ting Liu. 2015a. Document modeling with gated recurrent neural network for sentiment classification. In *EMNLP*, pages 1422–1432.

Duyu Tang, Bing Qin, and Ting Liu. 2015b. Learning semantic representations of users and products for document level sentiment classification. In *ACL-IJCNLP*, pages 1014–1023.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *ACL*, pages 1555–1565.

Hanqing Tao, Shiwei Tong, Hongke Zhao, Tong Xu, Binbin Jin, and Qi Liu. 2019. A radical-aware attention-based model for chinese text classification. In *AAAI*.

Chuhan Wu, Fangzhao Wu, Junxin Liu, Shaojian He, Yongfeng Huang, and Xing Xie. 2019a. Neural demographic prediction using search query. In *WSDM*, pages 654–662.

Chuhan Wu, Fangzhao Wu, Junxin Liu, and Yongfeng Huang. 2019b. Hierarchical user and item representation with three-tier attention for recommendation. In *NAACL-HLT*, pages 1818–1826.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *NAACL-HLT*, pages 1480–1489.

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

Xue Zhao, Chao Wang, Zhifan Yang, Ying Zhang, and Xiaojie Yuan. 2016. Online news emotion prediction with bidirectional lstm. In *International Conference on Web-Age Information Management*, pages 238–250. Springer.

Xinjie Zhou, Xiaojun Wan, and Jianguo Xiao. 2016. Attention-based lstm network for cross-lingual sentiment classification. In *EMNLP*, pages 247–256.