Adaptive Attention Network for Review Sentiment Classification

Chuantao Zong[1], Wenfeng Feng[1], Vincent W. Zheng[2], and Hankz Hankui Zhuo[1(✉)]

[1]School of Data and Computer Science, Sun Yat-Sen University, Guangzhou,China
{zongcht,fengwf}@mail2.sysu.edu.cn,zhuohank@mail.sysu.edu.cn
[2]Advanced Digital Sciences Center (ADSC), Singapore
vincent.zheng@adsc.com.sg

Abstract. Document-level sentiment classification is an important NLP task. The state of the art shows that attention mechanism is particularly effective on document-level sentiment classification. Despite the success of previous attention mechanism, it neglects the correlations among inputs (*e.g.*, words in a sentence), which can be useful for improving the classification result. In this paper, we propose a novel Adaptive Attention Network (AAN) to explicitly model the correlations among inputs. Our AAN has a two-layer attention hierarchy. It first learns an attention score for each input. Given each input's embedding and attention score, it then computes a weighted sum over all the words' embeddings. This weighted sum is seen as a "context" embedding, aggregating all the inputs. Finally, to model the correlations among inputs, it computes another attention score for each input, based on the input embedding and the context embedding. These new attention scores are our final output of AAN. In document-level sentiment classification, we apply AAN to model words in a sentence and sentences in a review. We evaluate AAN on three public data sets, and show that it outperforms state-of-the-art baselines.

1 Introduction

Sentiment classification [12] is an important task in NLP. Document-level sentiment classification attracts a lot of research interests. In general, review sentiment classification task is modeled as either a binary (*i.e.*, "positive" or "negative"), or multi-class classification (*e.g.*, ratings from "one star" to "five stars") problem.

Earlier work on sentiment classification relies on engineering useful features from the data to build the classification models. Some may also consider user features and product features [2]. With the development of neural network, recent study starts to explore using automatic feature learning for the review sentiment classification. For example, the state of the art uses various hierarchical neural networks to model the words and the sentences in a review [17, 1]. In particular, Chen *et al.*propose to model the representation of a review through a word-sentence-review hierarchy. To differentiate the importance of each word and each sentence in generating the whole review's representation, they introduce the *attention* mechanism [21] to model a weight for each word. Then they sum up the word embeddings in a sentence with the resulting weights as the sentence's embedding. Similarly, they apply attention for a weighted sum of the sentences as the review's representation.

Although the attention mechanism has shown to significantly improve the classification results [1], we notice that it assumes each input (word or sentence) as independent. Thus it overlooks the input correlations, which can be useful for the classification. Take Fig. 1(a) as an example. To The review is consist of one sentence. It shows the *word attention scores* for some salient words in a review. As we can see, "high" has a relatively large word attention score, indicating its high importance in representing the whole review. However, as "high" in the shop reviews often associate with price, it usually holds a negative polarity. Therefore, given a large attention score for "high", we tend to assign a negative polarity to the whole review. This results in a conflict between the predicted rating of "three stars", and the ground truth rating of "five stars". The fundamental reason of having such a conflict is that, the attention mechanism treats each word's attention independently. This overlooks the *context* in this specific review, *i.e.*, what this review is mainly about and what the leading polarity is. From a human's perspective, we can easily tell that this review is mainly about an endorsement of this dessert shop's yogurt, and the leading polarity is highly positive (*i.e.*, "happy", "amazing") despite the "high" price.

NSC+LA (predicted rating: 3; gold rating: 5):
always **happy** (0.074) here great yogurt and toppings brownies are **amazing** (0.076), the price is kinda **high** (0.193) or I would be here more often.

(a) Results from NSC+LA (state of the art) [1]

AAN (no U, no P) (predicted rating: 5; gold rating: 5):
always **happy** (0.109) here great yogurt and toppings brownies are **amazing** (0.123), the price is kinda **high** (0.112) or I would be here more often.

(b) Results from AAN (ours)

Fig. 1. Salient words (in boldface) with highest *word attention scores* in a sample review.

Is it possible for us to take the context of a specific review into account, and *adaptively* assign the attention for each salient word (*e.g.*, discount the weight of "high")? Our answer is yes! In this paper, we propose a novel *Adaptive Attention Network* (AAN) to explicitly model the correlation between the inputs (*e.g.*, words and sentences in the review domain) in the attention definition. Our AAN has a deeper two-layer attention hierarchy. Take the word attention in a sentence as an example. AAN first computes an attention score for each word in the sentence, by employing the outstanding attention mechanism of Chen *et al.*[1]. Then it introduces a *context embedding*, which aggregates all the words' embedding vectors with the attention scores by a weighted sum. To model the correlation, it computes another attention score for each word, based on how much the word matches the context embedding. These new attention scores are the final outputs of AAN for the words in a sentence. Similarly, we also apply AAN to model the sentence attention in the review. By utilizing the input correlations among words and sentences, AAN is able to improve the review comprehension and thus the classification results. As shown in Fig. 1(b), under our AAN mechanism, the word attention scores of salient words "happy" and "amazing" all significantly increase, whereas that of "high" decreases, which eventually helps us generate a perfect rating prediction of "five stars". Note that in this sample review we only have one sentence, thus the sentence attention score from AAN is one.

We summarize our contributions as follows.

• We identify an important limitation of the existing attention mechanism, and develop an adaptive attention network to model the input correlations in attention modeling.

• We evaluate AAN with three public, real-world review sentiment datasets. We show that AAN outperforms state-of-the-art baselines.

2 Related Work

Sentiment classification is usually seen as a special case of text classification. As the performance of text classifiers heavily relies on the extracted features, early work on sentiment classification mostly focuses on designing useful features from text content [12], sentiment lexicons [4], social network [2] and so on. With the development of neural network, some recent studies start to explore the application of deep learning in sentiment classification to avoid engineering the features. For example, to model the text's syntactic structure, Socher *et al.*explored a set of recursive neural networks models such as Recursive Auto-Encoder [13] and Recursive Neural Tensor Networks [14]. To leverage the dependency parsing information, Tai *et al.*proposed a tree-structured Long Short-Term Memory (LSTM) [16] in learning the representation of a document. To model the n-gram patterns, Lai *et al.*[8] and Kalchbrenner *et al.*[6] both explored using Convolutional Neural Networks (CNN) over the words in a sentence. To model the word-sentence-document hierarchy, Tang *et al.*proposed to first use CNN over the words to embed each sentence, then aggregate all the sentences by either simple pooling or Gated Recurrent Neural Network to embed the whole document [18]. Compared with our AAN model, these above neural network methods do not study the attention mechanism.

To incorporate the different importances of the words in each sentence, as well as the sentences in each document, attention mechanism [21] was introduced into text representation learning. For example, Yang *et al.*[22] proposed a hierarchical attention mechanism, which leverages the local semantic information in both word and sentence levels. In document-level sentiment classification, Chen *et al.*further extended the hierarchical attention mechanism to incorporate the user and product information with the attention design for words and sentences [1]. Tang *et al.*explored modeling attentions for different types of signals, including text content and text location [19]. Compared with AAN, these above attention methods often assume the inputs as independent. As a result, their attention definitions only have one single layer, which is from the inputs directly to the attention score output. Unlike these works, our AAN exploits the correlations among the inputs (to our knowledge this is the first work).

3 Adaptive Attention Network

We first formulate the problem of review sentiment classification. As inputs, we have a set of training documents $\mathcal{D} = \{(d_1, y_1), ..., (d_n, y_n)\}$, where each d_i is a document and $y_i \in \mathcal{Y}$ is the sentiment class (*e.g.*, $\mathcal{Y} = \{1, ..., 5\}$ indicating the ratings from "one star" to "five stars"). As output, we want to build a model $\mathcal{M}$, which can take a test document d as inputs and predict a rating class in $\mathcal{Y}$. Inspired by the pioneer work [17, 1], we choose to model each document as a sequence of sentences and each sentence as a sequence of words. Formally, we denote a document as $d_i = \{s_{i,1}, ..., s_{i,m_i}\}$, where each $s_{i,j}$ is a sentence and m_i is the number of sentences in d_i. We denote each sentence $s_{i,j} = \{w_{i,j,1}, ..., w_{i,j,m'_{ij}}\}$, where $w_{i,j,k} \in \mathcal{V}$ is a word from the vocabulary $\mathcal{V}$ and m'_{ij} is the number of words in $s_{i,j}$.

3.1 Two-layer AAN Architecture

Next we develop the AAN model. We begin with reviewing existing attention mechanism in [1]. As shown in Fig. 2(a), the existing attention mechanism generally takes a set of vectors $\{\mathbf{h}_1, ..., \mathbf{h}_m\}$ as inputs, and tries to compute an attention score α_i for each vector $\mathbf{h}_i \in \mathbb{R}^{K_1}$ by

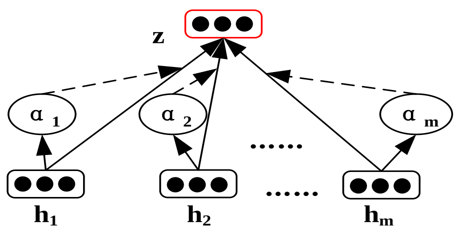
$$f_i = \mathbf{v}^\top \tanh(W\mathbf{h}_i + \mathbf{b}), \tag{1}$$

$$\alpha_i = \frac{\exp(f_i)}{\sum_{j=1}^{m} \exp(f_j)}, \tag{2}$$

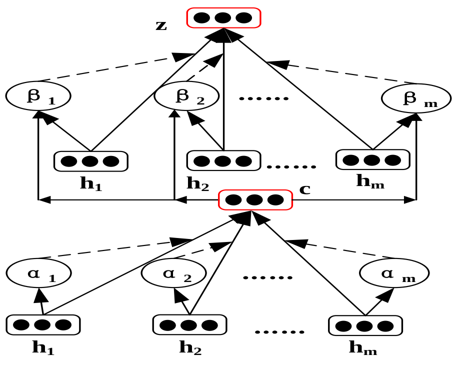
where $\mathbf{v} \in \mathbb{R}^{K_2}$, $W \in \mathbb{R}^{K_2 \times K_1}$ and $\mathbf{b} \in \mathbb{R}^{K_2}$ are learnable parameters. Based on the attention scores, the output $\mathbf{z} \in \mathbb{R}^{K_2}$ is

$$\mathbf{z} = \textstyle\sum_{i=1}^{m} \alpha_i \mathbf{h}_i. \tag{3}$$

As we can see, the above attention definition treats each input $\mathbf{h}_i$ independently. In practice, the correlation between the inputs can be useful. For example, in Fig. 1(a), we can see the necessity to consider the correlation between the salient word “high” with the other words, so as to ensure its attention score to be fully aware of the context in this specific review.



(a) Existing one-layer attention architecture



(b) Our two-layer AAN attention architecture

Fig. 2. Comparing our two-layer AAN with the existing one-layer attention architecture.

In order to take the input correlation into account, we develop a two-layer attention hierarchy as shown in Fig. 2(b). Let us use the example in Fig. 1(a) again to illustrate how we design such a hierarchy. For word attention, we denote each input $\mathbf{h}_i$ as an embedding for the i-th word in a sentence. To assign an appropriate attention score to the salient word "high", we first need to understand its context in the sentence. We represent such a sentence context by a *context embedding*, and we compute it by Eq. 3 as

$$\mathbf{c} = \textstyle\sum_{i=1}^{m} \alpha_i \mathbf{h}_i, \tag{4}$$

where α_i is estimated by Eq. 2 and $\mathbf{c} \in \mathbb{R}^{K_2}$. This context embedding estimation corresponds to our first layer of AAN in Fig. 2(b), where the α_i's are the first-layer attention scores. Ideally, we want to prompt those salient words that "match" the context. Therefore, we introduce a second layer of attention in Fig. 2(b), which measures how well $\mathbf{h}_i$ matches the context embedding $\mathbf{c}$ by a score b_i and then outputs a final attention score β_i:

$$b_i = \mathbf{h}_i^\top \mathbf{c}, \tag{5}$$

$$\beta_i = \frac{\exp(b_i)}{\sum_{j=1}^{m} \exp(b_j)}. \tag{6}$$

The β_i's are the second-layer, and the final, attention scores for AAN. Once having the final attention scores of AAN, we compute the representation of the whole sentence as

$$\mathbf{z} = \textstyle\sum_{i=1}^{m} \beta_i \mathbf{h}_i. \tag{7}$$

Remark: to help understand why mathematically Eq. 7 models the correlation among the inputs, we can do some simple expansion:

$$\begin{aligned} \mathbf{z} &\overset{1}{=} \textstyle\sum_{i=1}^{m} \frac{\exp(b_i)}{\sum_{j=1}^{m} \exp(b_j)} \mathbf{h}_i, \\ &\overset{2}{=} \textstyle\sum_{i=1}^{m} \frac{\exp(\mathbf{h}_i^\top \mathbf{c})}{\sum_{j=1}^{m} \exp(\mathbf{h}_j^\top \mathbf{c})} \mathbf{h}_i, \\ &\overset{3}{=} \textstyle\sum_{i=1}^{m} \frac{\exp(\sum_{k=1}^{m} \alpha_k \mathbf{h}_i^\top \mathbf{h}_k)}{\sum_{j=1}^{m} \exp(\sum_{k=1}^{m} \alpha_k \mathbf{h}_j^\top \mathbf{h}_k)} \mathbf{h}_i, \end{aligned} \tag{8}$$

where at step 1, we plug in Eq. 6; at step 2, we plug in Eq. 5; at step 3, we plug in Eq. 4. As we can see in Eq. 8, the attention score for each input $\mathbf{h}_i$ now becomes aware of the correlation between $\mathbf{h}_i$ and the other $\mathbf{h}_k$'s.

3.2 AAN for Review Modeling

We customize AAN for document-level sentiment classification. As suggested by [17, 1], we model each review as a hierarchy from words to sentences and finally a document. Therefore, we can assign attention to both the words in the sentence level and the sentences in the document level. Next, we illustrate how to take a review's content, as well as its user (who publishes this review) and product (which this review is about), as inputs, and finally predict a sentiment class as output. We summarize our deep neural network architecture of using AAN for document-level sentiment classification in Fig. 3. The architecture consists of three parts, as we shall introduce one by one next.

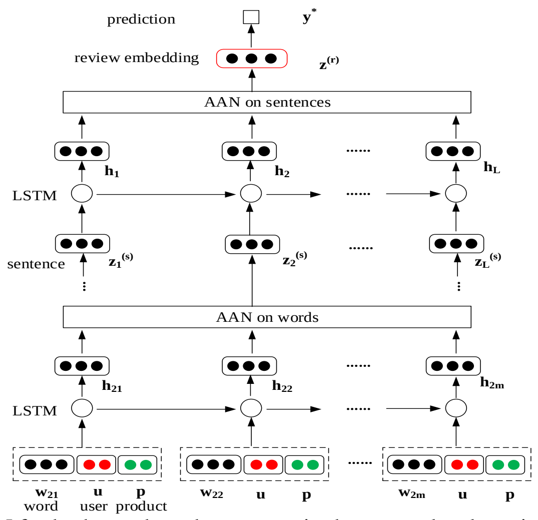


Fig. 3. Using AAN for both words and sentences in document-level sentiment classification.

• **Embedding from word to sentence**. For each sentence, we aim to generate a sentence embedding vector, from its words. First of all, for each word $w_{j,k}$ $(k = 1, ..., m_j)$ in sentence s_j, we assign an embedding vector $\mathbf{w}_{j,k} \in \mathbb{R}^{K_0}$. We pre-train these word embedding vectors by word2vec [11]. Secondly, in order to incorporate the user and product information, we also introduce an embedding vector $\mathbf{u} \in \mathbb{R}^{K_3}$ for each user u and an embedding vector[1] $\mathbf{p} \in \mathbb{R}^{K_3}$ for each product p. We choose to concatenate each word embedding $\mathbf{w}_{j,k}$ with the user embedding $\mathbf{u}$ and the product embedding $\mathbf{p}$ as the inputs for sentence embedding. It is worth noting that, such a concatenation design is significantly different from the previous designs. For example, in [17], each user (product) is represented with a matrix, which is multiplied with each $\mathbf{w}_{j,k}$ to get the input for sentence embedding. This matrix representation is likely to suffer from the data insufficiency for those users (products) with limited reviews. In [1], neither user nor product is used as input for sentence embedding, thus missing the opportunity of enriching the sentence semantics with user and product information. Thirdly, given the embedding concatenation for each word in the sentence, we employ a LSTM [15] to generate a hidden output $\mathbf{h}_{j,k} \in \mathbb{R}^{K_1}$ for each word.

Finally, we apply AAN to assign an attention score to each word. To incorporate the user and product information, we start with extending the first layer attention score definition as

$$\begin{aligned} f_{j,k}^{(w)} &= \mathbf{v}^\top \tanh(W_1^{(w)}\mathbf{h}_{j,k} + \mathbf{b}_1) + g_{j,k}^{(w)}, \\ g_{j,k}^{(w)} &= \mathbf{h}_{j,k} W_1^{(u)}\mathbf{u} + \mathbf{h}_{j,k} W_1^{(p)}\mathbf{p}, \\ \alpha_{j,k} &= \frac{\exp(f_{j,k}^{(w)})}{\sum_{l=1}^{m} \exp(f_{j,l}^{(w)})}, \end{aligned} \tag{9}$$

[1] Generally, user and product can have different dimensions, but we set them as the same to control the number of hyperparameters.

where $g_{j,k}^{(w)}$ is an extra term we introduce to indicate the interactions between word and user, as well as word and product. $W_1^{(w)} \in \mathbb{R}^{K_2 \times K_1}$, $W_1^{(u)} \in \mathbb{R}^{K_2 \times K_3}$, $W_1^{(p)} \in \mathbb{R}^{K_2 \times K_3}$ and $\mathbf{b}_1 \in \mathbb{R}^{K_2}$ are parameters. Then, we compute the second layer attention score $\beta_{j,k}$ for each word in the same way as Eq. 6. In the end, we aggregate all the words with their embedding vectors and the second-layer attentions to output a sentence embedding $\mathbf{z}_j^{(s)} \in \mathbb{R}^{K_2}$ by Eq. 7.

• **Embedding from sentence to review**. Given the embedding of each sentence, we aim to generate a review embedding vector, from its sentences. We take a similar procedure as embedding from word to sentence. Specifically, we take all the sentence embeddings as input, and employ a LSTM to generate a hidden output $\mathbf{h}_j \in \mathbb{R}^{K_1}$ for each sentence s_j (for $j = 1, ..., L$). Then, we apply AAN to assign an attention score to each sentence embedding vector. We use the similar extension in Eq. 9 to the first layer attention score definition for sentence. Denote $W_2^{(s)} \in \mathbb{R}^{K_2 \times K_1}$, $W_2^{(u)} \in \mathbb{R}^{K_2 \times K_3}$, $W_2^{(p)} \in \mathbb{R}^{K_2 \times K_3}$ and $\mathbf{b}_2 \in \mathbb{R}^{K_2}$ as parameters. Thus we compute the first-layer attention score for each sentence as

$$\begin{aligned} f_j^{(s)} &= \mathbf{v}^\top \tanh(W_2^{(s)}\mathbf{h}_j + \mathbf{b}_2) + g_j^{(s)}, \\ g_j^{(s)} &= \mathbf{h}_j W_2^{(u)}\mathbf{u} + \mathbf{h}_j W_2^{(p)}\mathbf{p}. \\ \alpha_j &= \frac{\exp(f_j^{(s)})}{\sum_{l=1}^{L}\exp(f_l^{(s)})}. \end{aligned} \tag{10}$$

Then, we compute the second-layer attention score β_j for each sentence in the same way as Eq. 6. Finally, we aggregate all the sentences with their embedding vectors and the second-layer attentions in Eq. 7 to output a review embedding $\mathbf{z}_j^{(r)} \in \mathbb{R}^{K_2}$.

• **Sentiment class prediction**. Given the review embedding, we aim to generate a prediction of the review's sentiment class. In our design, we feed the review embedding $\mathbf{z}_j^{(r)}$ into a multi-layer perceptron (MLP), which outputs the probabilities of this review belonging to each class in $\mathcal{Y}$:

$$\mathbf{y}^* = \mathrm{softmax}(W'\mathbf{z}_j^{(r)} + \mathbf{b}'), \tag{11}$$

where $W' \in \mathbb{R}^{|\mathcal{Y}| \times K_2}$ are $b_c \in \mathbb{R}^{\mathcal{Y}}$ are parameters.

We can link $\mathbf{y}^*$ with the ground truth label of this review, so as to supervise the model training. Denote Θ as the set of parameters, including the AAN parameters on words, the AAN parameters on sentences, the LSTM parameters on words, the LSTM parameters on sentences and the sentiment class prediction MLP parameters. For the training data set $\mathcal{D}$, we design the objective function as

$$\mathcal{L} = -\sum_{i=1}^{n} \log P(y_i|\mathcal{D}) + \lambda\Omega(\Theta), \tag{12}$$

where $P(y_i|\mathcal{D})$ is the probability of predicting review d_i as class y_i, and it can be computed by Eq. 11. $\Omega(\cdot)$ is a regularization function, *e.g.*, it sums up the ℓ_2-norm of each parameter in Θ. $\lambda > 0$ is a trade-off parameter.

4 Experiments

We evaluate AAN on three public benchmark data sets, including IMDB, Yelp 2013 and Yelp 2014 which are review texts including user/product information developed by[17]. Each record in the data sets is composed of a user ID, a product ID, a review and a rating. Table 1 list the statistics of the datasets including number of users, number of products, number of sentiment categories, number of documents, average number of documents per user, average number of documents per product, average length of document, average length of sentence, size of vocabulary in a data set.

Table 1. Statistics of three public data sets.

Data	user	product	class	doc	doc/user	doc/product	sen/doc	word/sen	voc
IMDB	1,310	1,635	10	84,919	64.82	51.94	16.08	24.54	105,373
Yelp 2013	1,631	1,633	5	78,966	48.42	48.36	10.89	17.38	48957
Yelp 2014	4,818	4,194	5	231,163	47.97	55.11	11.41	17.26	93197

We follow [17] to employ two evaluation metrics: 1) accuracy $Acc = \frac{T}{N}$, where T is the number of ratings predicted correctly and N is the size of the testing set; 2) root mean square error $RMSE = \sqrt{\frac{\sum_i (gd_i - pr_i)^2}{N}}$, where gd_i and pr_i are the gold rating and the predicted rating for document i, respectively.

We learn the word embedding by word2vec [10] and set the embedding dimension as $K_0 = 200$. We set the dimensions of hidden states and cell memory states in LSTM, sentence embedding and review embedding, user and product embeddings as $K_1 = 100$, $K_2 = 100$, $K_3 = 50$ respectively, which fit well to our GPU memory. We organize the reviews into batches for training. For varying length of sentences and reviews in each batch, we do zero padding in using LSTM. We set the batch size as 32. We set the regularization weight as $\lambda =$ 1E-5. We use the data splits provided by [17], which separate each data set into training, development and testing sets with a 80/10/10 split. We adadelta [23] for stochastic gradient descent.

4.1 Comparison with Baselines

We compare AAN with the state-of-the-art baselines, as listed below. **Majority:** it assigns the majority sentiment category in the training set to each test review. **Trigram:** it uses trigrams as features to train a Support Vector Machine (SVM) [12] for review classification. **TextFeature:** it extract several text features, such as word and character n-grams, sentiment lexicons to train a SVM [7]. **UPF:** it was introduced by [17]. It extracts user and product features like [5], and concatenates them with the features in **Trigram** and **TextFeature** for SVM. **AvgWordvec:** it learns 200-dimensional word embeddings by word2vec [11] and uses the average word embeddings of each review for SVM training. **SSWE:** it learns sentiment-specific word embeddings and thus the review embedding for SVM classification [20]. **RNTN + Recurrent:** its learns a RNTN

[14] for sentence embedding, and a recurrent neural network (RNN) for review embedding. **Paragraph Vector:** it uses the paragraph structure to learn the embedding for varying-length sentences and documents [9]. **JMARS:** it combines user and review aspects by collaborative filtering and topic modeling for review sentiment classification [3]. **UPNN:** it takes user-text and product-text consistency matrices as additional inputs, to assist the embedding for words, sentences and reviews [17]. **NSC & NSC + LA & NSC + UPA:** these three neural network models all explore the word-sentence-review hierarchy [1]. NSC uses a mean pooling in sentence and review embedding. NSC + LA improves NSC with a local semantic attention (LA) [22]. NSC + UPA improves NSC by considering user and product in the attention definitions.

Table 2. Results of all the approaches on IMDB, Yelp2013 and Yelp2014 datasets. *Acc* (the higher, the better) and *RMSE* (the lower, the better) are two evaluation criteria. The best performances in each group are in boldface.

Settings	Models	IMDB		Yelp13		Yelp14	
		Acc	RMSE	Acc	RMSE	Acc	RMSE
no U, no P	Majority	0.196	2.495	0.411	1.060	0.392	1.097
	Trigram [12]	0.399	1.783	0.569	0.814	0.577	0.804
	TextFeature [12]	0.402	1.793	0.556	0.845	0.572	0.800
	AvgWordvec + SVM [11]	0.304	1.985	0.526	0.898	0.530	0.893
	SSWE + SVM [20]	0.312	1.973	0.549	0.849	0.557	0.851
	Paragraph Vector [9]	0.341	1.814	0.554	0.832	0.564	0.802
	RNTN + Recurrent [14]	0.400	1.764	0.574	0.804	0.582	0.821
	UPNN (no U, no P) [17]	0.405	1.629	0.577	0.812	0.585	0.808
	NSC [1]	0.438	1.495	0.628	0.703	0.635	0.687
	NSC + LA [1]	0.474	1.391	0.631	0.708	0.641	0.683
	AAN (no U, no P)	**0.483**	**1.385**	**0.636**	**0.694**	**0.643**	**0.681**
with U and P	Trigram + UPF [17]	0.404	1.764	0.570	0.803	0.576	0.789
	TextFeature + UPF [17]	0.402	1.774	0.561	1.822	0.579	0.791
	JMARS [3]	N/A	1.773	N/A	0.985	N/A	0.999
	UPNN (U + P) [17]	0.435	1.602	0.596	0.784	0.608	0.764
	NSC + UPA [1]	0.513	1.299	0.645	0.689	0.666	0.655
	AAN (U + P)	**0.538**	**1.243**	**0.662**	**0.663**	**0.670**	**0.646**

Table 2 shows the performance. We test all the methods in two settings: 1) with user and product information, denoted by “with U and P”; 2) without them, denoted by “no U, no P”. Note that, because we use exactly the same data set splits and experimental settings with [17], we can directly borrow some of their results in the “no U, no P” setting, including “Majority”, “Trigram”, “TextFeature”, “AvgWordvec + SVM”, “SSWE + SVM”, “Paragraph Vector”, “RNTN + Recurrent” and “UPNN (no U, no P)”.

In the setting of “no U, no P”, AAN outperforms AvgWordvec + SVM, SSWE + SVM, Paragraph Vector and RNTN + Recurrent, which shows the necessity to differentiate the words and sentences in the review representations for sentiment classification. Besides, AAN outperforms both NSC and NSC + LA, which justifies our motivation to capture correlations among words or sentences, since NSC and NSC + LA both only model the words and sentences individually in the attention estimation.

In the setting of "with U and P", AAN still outperforms all the baselines in all of the three datasets. This confirms the superiority of our AAN model in exploring the attention mechanism, as well as the user and product information. It is worth noting that, the improvement of our AAN (U + P) model over NSC + UPA is larger than those of AAN (no U, no P) over UPNN (U + P) and NSC + LA. This means, in addition to the benefit we obtained from our novel two-layer attention mechanism, our new design of incorporating the user and product information (*i.e.*, concatenating each word embedding with the user and product embeddings) can indeed help improve the classification results. Such an observation justifies our conjecture of how to model the user and product information in Sect. 3.2.

4.2 Impact of User and Product Embeddings

We further study the impact of using the user and product embeddings in our AAN model. We perform AAN in four different settings, depending on whether we use the user or product embedding or not. We summarize the results in Table 3. As we can see, AAN (P + U) outperforms the other three models, which suggests both user and product information can help improve the sentiment classification results. This is consistent with our intuition that, the more information we exploit, the better result we will generally have. We also observe that, AAN (only U) outperforms AAN (only P), which implies that user preferences seems to be more important than the product properties in sentiment classification. We may understand it as that the reviews are subjective, thus the user preferences play a more important role in sentiment classification. Finally, we also see that, both AAN (only U) and AAN (only P) outperform AAN (no U, no P), which suggests both user and product are useful for sentiment classification.

Table 3. Impacts of using user and product information in AAN.

Models	IMDB		Yelp13		Yelp14	
	Acc	RMSE	Acc	RMSE	Acc	RMSE
AAN (P + U)	**0.538**	**1.243**	**0.662**	**0.663**	**0.670**	**0.646**
AAN (only U)	0.527	1.264	0.653	0.674	0.666	0.652
AAN (only P)	0.485	1.373	0.632	0.695	0.641	0.670
AAN (no U, no P)	0.483	1.385	0.636	0.694	0.643	0.681

4.3 Impact of Adaptive Attention Mechanism

We also study the impact of using AAN in both word and sentence levels. In Table 4, the first column denoted by "AA" indicates that we exploit our attention mechanism in the word level, and "MP" indicates that we exploit a simple mean pooling without our attention mechanism in the word level. From Table 4, we can see that employing our attention mechanism in both word and sentence levels (*i.e.*, AA + AA in the third row of Table 4) outperforms all the other settings. This means: 1) it is important to use adaptive attention in both the word and sentence level; 2) adaptive attention is more effective than the simple mean pooling, since it tries to differentiate the importances of different words and sentences. We also make another interesting observation, by comparing the fourth row (AA + MP) and the fifth row (MP + AA) in Table 4. The results show that, AA + MP seems to slightly outperform MP + AA; *i.e.*, using AAN in the word level seems to be better than in the sentence level. The possible reason is that, the number of words is much larger than that of sentences, thus the noise is bigger in the word level. Besides, since there are more words than sentences, we may have more data to better learn the attentions in the word level than in the sentence level.

Table 4. Impact of using AAN in both word and sentence levels. "AA" indicates using AAN. "MP" indicates a simple mean pooling without attention. The best results are in boldface.

Attention mechanisms		IMDB		Yelp13		Yelp14	
Word-level	Sentence-level	Acc	RMSE	Acc	RMSE	Acc	RMSE
AA	AA	**0.538**	**1.243**	**0.662**	**0.663**	**0.670**	**0.646**
AA	MP	0.519	1.276	0.650	0.682	0.664	0.650
MP	AA	0.513	1.281	0.648	0.690	0.665	0.656
MP	MP	0.496	1.339	0.643	0.685	0.661	0.659

5 Conclusion

In this paper, we identify that the existing attention mechanisms often suffer from a significant limitation, which assumes the inputs as independent. Therefore, we propose a novel Adaptive Attention Network to model the correlation among the words and the sentences in their attention definitions. We also customize AAN for document-level sentiment classification, especially incorporating the user and product information. We evaluate AAN on three public benchmark data sets and show that it outperforms the state-of-the-art baselines. In the future, we plan to extend AAN with syntactic structure of the text, such as dependency trees, so as to further improve the classification.

Acknowledgments

We thank the National Key Research and Development Program of China (2016YFB020 1900), National Natural Science Foundation of China (U1611262), Guangdong Natural Science Funds for Distinguished Young Scholar (2017A030306028), Pearl River Science and Technology New Star of Guangzhou, and Guangdong Province Key Laboratory of Big Data Analysis and Processing for the support of this research. Zheng thanks the support of the National Research Foundation, Prime Ministers Office, Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) programme.

References

1. Chen, H., Sun, M., Tu, C., Lin, Y., Liu, Z.: Neural sentiment classification with user and product attention. In: EMNLP 2006, pp. 1650–1659 (2016)
2. Cheng, K., Li, J., Tang, J., Liu, H.: Unsupervised sentiment analysis with signed social networks. In: AAAI, pp. 3429–3435 (2017)
3. Diao, Q., Qiu, M., Wu, C.Y., Smola, A.J., Jiang, J., Wang, C.: Jointly modeling aspects, ratings and sentiments for movie recommendation (JMARS). In: SIGKDD, pp. 193–202 (2014)
4. Ding, X., Liu, B., Yu, P.S.: A holistic lexicon-based approach to opinion mining. In: WSDM, pp. 231–240 (2008)
5. Gao, W., Yoshinaga, N., Kaji, N., Kitsuregawa, M.: Modeling user leniency and product popularity for sentiment classification. In: IJCNLP, pp. 1107–1111 (2013)
6. Kalchbrenner, N., Grefenstette, E., Blunsom, P.: A convolutional neural network for modelling sentences. In: ACL, pp. 655–665 (2014)
7. Kiritchenko, S., Zhu, X., Mohammad, S.M.: Sentiment analysis of short informal texts. J. Artif. Intell. Res. 50, 723–762 (2014)
8. Lai, S., Xu, L., Liu, K., Zhao, J.: Recurrent convolutional neural networks for text classification. In: AAAI, pp. 2267–2273 (2015)
9. Le, Q.V., Mikolov, T.: Distributed representations of sentences and documents. CoRR. abs/1405.4053 (2014)
10. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. CoRR. abs/1301.3781 (2013)
11. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: NIPS, pp. 3111–3119 (2013)
12. Pang, B., Lee, L., Vaithyanathan, S.: Thumbs up?: Sentiment classification using machine learning techniques. In: EMNLP, pp. 79–86 (2002)
13. Socher, R., Pennington, J., Huang, E.H., Ng, A.Y., Manning, C.D.: Semi-supervised recursive autoencoders for predicting sentiment distributions. In: EMNLP, pp. 151–161 (2011)
14. Socher, R., Perelygin, A., Wu, J.Y., Chuang, J., Manning, C.D., Ng, A.Y., Potts, C.: Recursive deep models for semantic compositionality over a sentiment treebank. In: EMNLP, 1642 (2013)
15. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: NIPS, pp. 3104–3112 (2014)
16. Tai, K.S., Socher, R., Manning, C.D.: Improved semantic representations from tree-structured long short-term memory networks. In: ACL, pp. 1556–1566 (2015)
17. Tang, D., Qin, B., Liu, T.: Learning semantic representations of users and products for document level sentiment classification. In: ACL, pp. 1014–1023 (2015)
18. Tang, D., Qin, B., Liu, T.: Document modeling with gated recurrent neural network for sentiment classification. In: EMNLP, pp. 1422–1432 (2015)
19. Tang, D., Qin, B., Liu, T.: Aspect level sentiment classification with deep memory network. In: EMNLP, pp. 214–224 (2016)
20. Tang, D., Wei, F., Yang, N., Zhou, M., Liu, T., Qin, B.: Learning sentiment-specific word embedding for twitter sentiment classification. In: ACL, pp. 1555–1565 (2014)
21. Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A.C., Salakhutdinov, R., Zemel, R.S., Bengio, Y.: Show, attend and tell: Neural image caption generation with visual attention. In: ICML, pp. 2048–2057 (2015)
22. Yang, Z., Yang, D., Dyer, C., He, X., Smola, A.J., Hovy, E.H.: Hierarchical attention networks for document classification. In: NAACL, pp. 1480–1489 (2016)
23. Zeiler, M.D.: ADADELTA: an adaptive learning rate method. CoRR. abs/1212.5701 (2012)