Supplementary Material for "Learning Local and Global Multi-Context Representations for Document Classification"

Yi Liu*, Hao Yuan† and Shuiwang Ji*
* *Texas A&M University, College Station, TX, USA* *Email: {yiliu, sji}@tamu.edu*
†*Washington State University, Pullman, WA, USA* *Email: hao.yuan@wsu.edu*

I. Local and Global Multi-Context Attention

A. Incorporation of User and Product Information

It has been shown that incorporating user preference and product characteristic can significantly improve the performance for document-level sentiment classification [1], [2]. This is reasonable since the same word may convey different emotional intensity when used by different users. For example, a critical user may use "good" to express excellent assessment towards a product while a moderate user uses "good" to evaluate an ordinary product. Similarly, the brand information of a product also influence its rating. Hence, we propose to incorporate user and product information into our proposed LGCA and MCA models.

First, we map each user and product to vector representations as $\mathbf{u} \in \mathbb{R}^{d^u}$ and $\mathbf{p} \in \mathbb{R}^{d^p}$, where d^u and d^p are the dimensionality of user and product embedding, respectively. For simplicity, we assume the user embedding, product embedding, and sentence embedding share the same dimensionality so that $d^u = d^p = d^v$. We incorporate user and product information to build the key matrix K in our models. In attention mechanism, the key matrix K is calculated base on the representations of sentences. To combine these information, we choose to replicate $\mathbf{u}$ and $\mathbf{p}$ multiple times spatially to form matrices U and P, which share the same dimensionality with V. Next, we combine U, P and V via linear transformation and concatenation, and obtain the key matrix K as

$$K = \tanh(W^k V + W^U U + W^P P) \in \mathbb{R}^{d^k \times m}, \quad (1)$$

where $W^k, W^U, W^P \in \mathbb{R}^{d^k \times d^v}$ represent the trainable weight matrices in linear transformation.

In addition, we incorporate user and product information to build the context matrix Q. The context matrix Q is a combination of the global context matrix Q_g and the local context matrix Q_l. Specifically, user and product information is used to build the local context matrix Q_l since user and product information are local information on which local context matrix should rely. Mathematically, Q_l can be computed as:

$$Q_l = \frac{1}{m} W^Q V_i \mathbf{1}_m \mathbf{h}^T + W^u \mathbf{u}\mathbf{h}_u^T + W^p \mathbf{p}\mathbf{h}_p^T, \quad (2)$$

where W^Q, W^u, W^p are trainable weight matrices and $\mathbf{h}, \mathbf{h}_u, \mathbf{h}_p \in \mathbb{R}^r$ are trainable weight vectors. In this way, both user and product information is incorporated into the document representation D^m. The matrix D^m is unfolded into a $d^v \times r$ dimensional vector, denoted as $\mathbf{d}_f$. User and product information is further incorporated to obtain the final document representation $\mathbf{d}$ as

$$\mathbf{d} = \text{ReLu}(W_f \mathbf{d}_f + W_f^U \mathbf{u} + W_f^P \mathbf{p}) \in \mathbb{R}^{d^v}, \quad (3)$$

where W_f, W_f^U, W_f^P represent the trainable weight matrices. Thus, user and product information contributes to both the attention model and the linear transformation, and finally helps document representations.

B. Document Classification

The vector $\mathbf{d} \in \mathbb{R}^{d^v}$ represents the extracted high-level features for the document D. To obtain the classification result of the document, we employ a linear layer with softmax function. It projects $\mathbf{d}$ to the target distribution of C classes. Mathematically, we have

$$\mathbf{p} = \text{softmax}(W^c \mathbf{d} + \mathbf{b}_c) \in \mathbb{R}^C, \quad (4)$$

where $W^c \in \mathbb{R}^{C \times d^v}$ denotes the weight matrix in linear layer and $\mathbf{b}_c \in \mathbb{R}^C$ denotes the bias. Then the cross-entropy loss is used to optimize the model as

$$L = -\sum_{D \in Dataset} \sum_{c=1}^{C} y^c \cdot \log(p^c), \quad (5)$$

where p^c is the computed probability of the document D to be predicted as label c. The y^c is set to 1 if the predicted label matches the ground truth. Otherwise, we set $y^c = 0$.

II. Experimental Studies

A. Datasets

We evaluate our methods on three datasets, including the IMDB, Yelp 2013 and Yelp 2014. The datasets are constructed by [2], and they all contain user and product information. The statistics of these datasets are summarized in Table I. These datasets are originally split into three parts, including training (80%), validation (10%) and testing (10%) sets.

Table I
SUMMARY OF STATISTICS FOR THE 3 DATASETS.

Datasets	#c	#docs	#sens/doc	#words/sen	#users	#products	#docs/user	#docs/product
IMDB	10	84,919	16.08	24.54	1,310	1,635	64.82	51.94
Yelp 2013	5	78,966	10.89	17.38	1,631	1,633	48.42	48.36
Yelp 2014	5	231,163	11.41	17.26	4,818	4,194	47.97	51.11

IMDB: The IMDB dataset is a collection of movie reviews from online movie database. Each review contains a rating between 0 to 10. In this dataset, each movie is reviewed by multiple users and each user can make comment on multiple movies.
Y. 13: The Yelp 2013 dataset is obtained from The Yelp Dataset Challenge in 2013. Each review includes information of multiple components, such as business and service. The rating scale is between 1 and 5.
Y. 14: This dataset is obtained from Yelp Challenge in 2014. The contents in this dataset are similar to those of the Yelp 2013 dataset but the number of reviews is much larger.

B. Baseline Methods

We compare our models with several baseline approaches for document-level sentiment classification tasks. These baselines include both traditional methods and deep learning methods. All of these models can be easily extended to incorporate user and product information.
Trigram method first extracts the unigrams, bigrams and trigrams of the document as features, and then employs SVM classifiers to perform document sentiment classification [3].
Textfeature extracts rich text features such as n-grams, bag-of-words, and cluster features to train SVM classifiers [4].
SSWE first learns the sentiment-specific word embeddings (SSWE) [5], then uses pooling methods to obtain document representation. Finally the achieved vector features are used as input for a SVM classifier for sentiment classification.
RNN employs recursive neural tensor network (RNTN) [6] to obtain vector representation for each sentence in the document. Then the RNN is used to generate the overall representation for the whole document based on these sentence-level representations.
SSA is the structured self-attention for sentence embedding [7]. It employs the globally shared contexts, but does not consider multiple local contexts to capture document-specific information. In addition, it builds a one-level architecture where documents are treated as long sentences and represented from words directly.
NSC employs a hierarchical architecture, which includes LSTM networks at both word and sentence levels to obtain text representations for sentiment classification.
NSC+LA implements hierarchical attention based on NSC. It's essentially the re-implementation of the idea in hierarchical attention networks [8].
UPF incorporates the user leniency and product popularity features [9] to text features in **Trigram** and **Textfeatue**, and then train SVM classifiers using these features.
RNN+UPF first generates the document representation via the model **RNN**, then combines the user information, product information and the overall document representation as final features to perform classification.
UPNN (CNN) first constructs vector representations for the user and product information. Then it incorporates these representations into the word level embedding of input texts [2]. This method does not employ the hierarchical architecture but directly generate the document representation based on the enriched word vectors via CNN layers.
UPNN (NSC) employs hierarchical CNNs, which incorporates user and product information at both the word level and sentence level. The remaining parts are the same as **UPNN (CNN)**.
SSA+UPA incorporates user and product information to the structured self-attention model to improve the performance for document-level sentiment classification.
NSC+UPA uses the hierarchical attention architecture to build representations for document vectors [1]. It incorporates user and product information at both word and sentence levels.

For the **RNN+UPF** approach, there is no available implementation that incorporates user and product information. Hence, we re-implement the model report the best performance. As the **SSA** model is not applied to the three used datasets, we apply the publicly released code[1] to our datasets using the same settings as what used for sentiment analysis described in the work [7]. Particularly, we also set the number of contexts to be 30 and use the penalty term as these settings are proved to achieve best performance. The same settings are used in the **SSA+UPA** model. For other approaches, we directly report the results from [1], [10].

C. Experimental Setup

The evaluation metrics include Accuracy and RMSE in our study. Basically, Accuracy measures the overall document classification performance of our model, and RSME measures the differences between predicted labels and ground truth labels. They are defined as

$$\text{Accuracy} = \frac{C}{N}, \qquad \text{RSME} = \frac{1}{N}\sqrt{\sum_{k=1}^{N}(p_k - g_k)^2}, \qquad (6)$$

[1]https://github.com/kaushalshetty/Structured-Self-Attention/

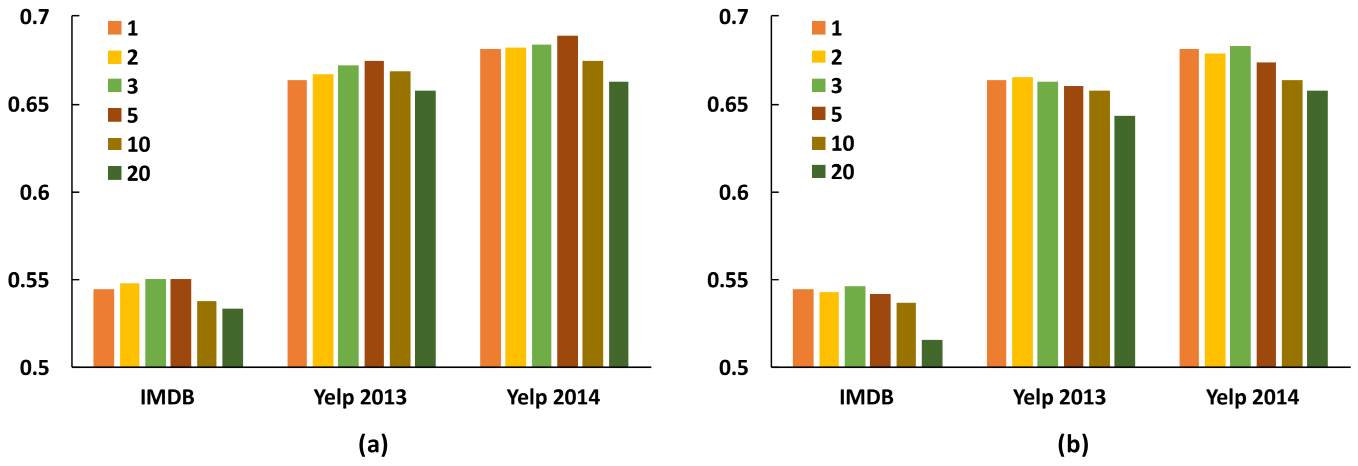

Figure 1. Effect of the context vector numbers at two levels in terms of accuracy. We denote the context vector number at sentence level as r_1, and the context vector number at document level as r_2. In Figure (a), r_2 is set to different values while r_1 is fixed to 1. In Figure (b), r_1 is set to different values while r_2 is fixed to 1.

Table II
COMPARISON BETWEEN FOUR MODELS IN TERMS OF ACCURACY AND NUMBER OF PARAMETERS ON THE YELP 2014 DATASET WITH INCORPORATING USER AND PRODUCT INFORMATION. "RATIO" INDICATES THE CHANGE IN NUMBER OF PARAMETERS COMPARED WITH THE STATE-OF-THE-ART NSC+UPA MODEL.

Models	ACC	* of Parameters	Ratio
SSA+UPA	0.644	2,428,400	23.5%
NSC+UPA	0.667	1,966,210	0.0%
LGCA$_{neat}$+UPA	0.679	1,966,210	0.0%
MCA$_{neat}$+UPA	0.683	2,086,219	6.1%
LGCA+UPA	0.681	2,086,210	6.1%
MCA+UPA	0.684	2,206,210	12.2%

where C is the number of documents that are correctly predicted, N is the total number of documents. p_k and g_k denote the predicted label and the ground truth label for the k^{th} document, respectively.

D. Parameter Study

Since LGCA and MCA both involve extra parameters, it maybe be argued that the performance improvement is due to the use of extra parameters. In this section, we conduct experiments to investigate the contributions of local contexts and multiple contexts by removing the extra parameters in original LGCA and MCA. These parameters are essentially used for linear transformation. Specifically, we remove the additional trainable weight vectors and weight matrices in Eq. 2 and obtain the model which we denote as LGCA$_{neat}$. If we remove the additional trainable weight matrices in Eq. 2 and Eq. 3, we can achieve the model which we denote as MCA$_{neat}$. The results on the large-scale datasets Yelp 2014 with incorporating user and product information are provided in Table II. We can find that both LGCA$_{neat}$ and MCA$_{neat}$ use less parameters than SSA, but achieve a large grid of 3.5% and 3.9% improvement in accuracy, respectively. When comparing with the state-of-the-art model NSC+UPA, LGCA$_{neat}$ does not involve extra parameters and MCA$_{neat}$ induces 6.1% additional parameters. However, LGCA$_{neat}$ results in 1.2% improvement and MCA$_{neat}$ results in 1.6% improvement in accuracy. In addition, LGCA$_{neat}$ and LGCA achieve similar performance, and MCA$_{neat}$ and MCA also achieve similar performance for classification. These observations demonstrate the improvement in performance is mainly caused by encoding local context information and considering multiple contexts within a document. Also, the promising results in the main paper indicate there is no over-fitting, thereby demonstrating the effectivenesses and efficiency of our proposed LGCA and MCA.

E. Analysis of Context Vector Number

Our proposed MCA approach incorporates different components in the context of a sentence and a document to compute multiple context vectors. The number of context vectors is denoted as r. In this section, we investigate the effect of different values for r. We study such r choices for both sentence level and document level. We denote the numbers of sentence-level and document-level contexts as r_1 and r_2, respectively. If we set $r_1 = 1$ and $r_2 = 1$, the MCA method is equivalent to the LGCA approach. As models considering user and product information achieve great improvements compared with the corresponding models without extra information, we focus on the MCA approach incorporating user and product information. The experimental results are reported in Figure 1.

When r_1 is set to 1, there is only one context vector for a sentence, which means we only explore one aspect within a sentence. As shown in Figure 1, a reasonable choice of r_2 results in improvements in accuracy. We observe that when r_2 is between 2 and 5, the performance is better than the case when $r_2 = 1$. However, when we set r_2 to 10 and 20, the performance drops significantly. The choices of $r_1 = 3, 5, 5$ leads to the best performance for the dataset IMDB, Yelp 2014 and Yelp 2014, respectively. These results indicate that multiple context vectors can help capture local context information, and they can improve the performance. However, when too much context information is captured, the over-fitting problem occurs.

Similarly, we investigate the choice of r_1, and we set r_2 to 1. We can observe from the results on the right side of Figure 1 that smaller value for r_1 tends to yield better results. The choice of $r_1 = 3$ leads to the best performance for the dataset IMDB and Yelp 2014.

By comparing the above two cases, we can see that the increase of r_2 tends to have larger impact on the performance of our models. This is reasonable since most documents contains multiple sentences, and we need a larger r_2 to capture the context information for multiple sentences. On the other hand, the context within a sentence is limited since most sentences only cover one or two topics. Hence, we should choose a smaller value for r_1.

References

[1] H. Chen, M. Sun, C. Tu, Y. Lin, and Z. Liu, "Neural sentiment classification with user and product attention," in *EMNLP*, 2016.

[2] D. Tang, B. Qin, and T. Liu, "Learning semantic representations of users and products for document level sentiment classification," in *ACL*, 2015.

[3] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "Liblinear: A library for large linear classification," *Journal of machine learning research*, vol. 9, no. Aug, pp. 1871–1874, 2008.

[4] S. Kiritchenko, X. Zhu, and S. M. Mohammad, "Sentiment analysis of short informal texts," *Journal of Artificial Intelligence Research*, vol. 50, pp. 723–762, 2014.

[5] D. Tang, F. Wei, N. Yang, M. Zhou, T. Liu, and B. Qin, "Learning sentiment-specific word embedding for twitter sentiment classification," in *ACL*, 2014.

[6] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *EMNLP*, 2013.

[7] Z. Lin, M. Feng, C. N. d. Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio, "A structured self-attentive sentence embedding," *arXiv preprint arXiv:1703.03130*, 2017.

[8] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, "Hierarchical attention networks for document classification," in *NAACL*, 2016.

[9] W. Gao, N. Yoshinaga, N. Kaji, and M. Kitsuregawa, "Modeling user leniency and product popularity for sentiment classification," in *IJCNLP*, 2013.

[10] Z. Wu, X.-Y. Dai, C. Yin, S. Huang, and J. Chen, "Improving review representations with user attention and product attention for sentiment classification," in *AAAI*, 2018.