BAG OF REGION EMBEDDINGS VIA LOCAL CONTEXT UNITS FOR TEXT CLASSIFICATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Contextual information and word orders are proved valuable for text classification task. To make use of local word order information, n-grams are commonly used features in several models, such as linear models. However, these models commonly suffer the data sparsity problem and are difficult to represent large size region. The discrete or distributed representations of n-grams can be regarded as region embeddings, which are representations of fixed size regions. In this paper, we propose two novel text classification models that learn task specific region embeddings without hand crafted features, hence the drawbacks of n-grams can be overcome. In our model, each word has two attributes, a commonly used word embedding, and an additional local context unit which is used to interact with the word's local context. Both the units and word embeddings are used to generate representations of regions, and are learned as model parameters. Finally, bag of region embeddings of a document is fed to a linear classifier. Experimental results show that our proposed methods achieve state-of-the-art performance on several benchmark datasets. We provide visualizations and analysis illustrating that our proposed local context unit can capture the syntactic and semantic information.

1 INTRODUCTION

Text classification is an important task in natural language processing, which has been studied for years with applications such as topic categorization, web search, sentiment classification. A simple yet effective approach for text classification is to represent documents as bag-of-words, and train a linear classifier, e.g., a logistic regression, a support vector machines (Joachims, 1998; Fan et al., 2008) or a naive bayes (McCallum et al., 1998). Although bag-of-words methods are efficient and popular, they lose the local ordering information of words which has been proved useful particularly on sentiment classification (Pang et al., 2002).

To benefit from local word order for text classification, n-grams which is a simple statistics of ordered word combinations usually performs good (Pang et al., 2002; Wang & Manning, 2012; Joulin et al., 2016). With a n-gram vocabulary, Wang & Manning (2012) use one-hot vectors as discrete embedding of n-grams. Recently, FastText (Joulin et al., 2016) directly learns distributed embeddings of n-grams and words for the task of interest. In FastText, learning the embedding of n-gram can be regarded as learning low-dimensional vector representation of a small fixed size region, i.e, region embedding. Although the n-grams was widely used, two caveats come with n-grams: 1) the vocabulary size increases exponentially with n, and yielding a high-dimensional feature space, which makes it difficult to apply large region size(e.g., $n > 4$), such as *People 's Republic of China* 2) Since the representation of each n-gram is looked up by an unique id, the data sparsity problem cannot be avoided, for example, the similar n-grams *a very good movie* and *an excellent movie* are trained separately.

In this paper, we focus on learning task specific region embedding for text classification, while Johnson & Zhang (2015) learn the task independent region embedding from unlabeled data. Instead of using hand crafted n-grams features, we directly learn the region embedding based on word sequence, thus the limitations of n-grams can be avoided. Intuitively, given a word sequence in a fixed size region, both words and words' relative positions are the key information to generate the semantic of the region. In this work, each word has two components, a commonly used word embedding, and an additional local context unit. To utilize the information of words' relative positions, the local context unit of a word is parameterized by a matrix, and each column is used to interact with the local context word in corresponding relative position, which can be also regarded as a linear projection function on the embedding of corresponding word. The local context units and word embeddings are used to generate representations of regions from two perspectives.

For text classification task, our models just use bag of region embeddings to represent the document, and feed the document representation to a fully connected linear classifier. The word embeddings and local context units are jointly trained on the classification task.

Recently, several neural models are proposed to make use of local ordered information in word sequence, and convolutional neural networks(CNN) (Kim, 2014) (Johnson & Zhang, 2014) are most similar with our works. The essence of CNN is to learn embeddings for small fixed size regions, and each kernel of the convolutional layer tries to capture a specific semantic or structural feature. The kernel of CNN is shared globally, and usually several kernels are needed to preserve task related features, while the proposed local context unit is word dependent and can be applied to capture the particular semantic of the word to its local context. Moreover, the convolution kernels extract the predictive feature by applying convolution operation on the word sequence, while our local context units are used as distinct linear projection functions on context words in corresponding relative positions.

2 Related Works

Text classification has been studied for years, traditional approaches focused on feature engineering and using different types of machine learning algorithms. For feature engineering, bag-of-words features are efficient and popular. In addition, the hand-crafted n-grams or phrases are added to make use of word order in text data, which has been shown effective in Wang & Manning (2012). For machine learning algorithms, linear classifiers are widely used, such as naive bayes (McCallum et al., 1998), logistic regression and support vector machines (Joachims, 1998; Fan et al., 2008). However, these models commonly suffer the data sparsity problem.

Recently, several neural models have been proposed, the pre-trained word embeddings of *word2vec* (Mikolov et al., 2013) have been widely used as the inputs to deep neural models such as recursive tensor networks (Socher et al., 2013). On the other hand, models that can directly learn task specific word embeddings have been proposed recently, such as FastText (Joulin et al., 2016) and CNN (Kim, 2014), which both have been successfully applied on text classification tasks. In the rest of this section, we briefly introduce FastText and CNN, the two approaches that are most similar to ours.

FastText FastText averages the word embeddings to represent a document, and uses a full connected linear layer as the classifier. The word embeddings are trained for each task specifically. To utilize the local word order information of small regions, FastText uses hand-crafted n-grams as features in addition to single words. With the simple architecture, FastText has been proved to be effective and highly efficient on text classification tasks. Similarly, our models use bag of region embeddings to represent a document, and use the same linear classifier. Differently, our models directly learn the semantics of regions based on word sequence, hand-crafted features is not required.

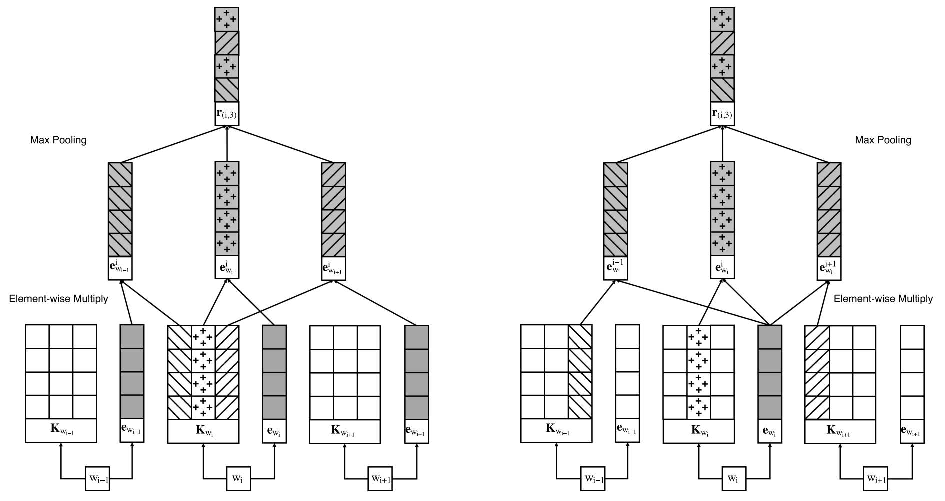
CNN CNN is a feed-forward network with convolution layers interleaved with pooling layers, which are originally used for image processing tasks. For natural language processing, words are commonly converted to vectors. CNN directly applies convolutional layer on word vectors, both word vectors and the shared(word independent) kernels are the parameters of CNN, which can be learned to capture the predictive structures of small regions. Our purpose is similar with CNN, which tries to learn task specific representations for regions. Unlike CNN, we apply local context units on word vectors, our units are word dependent.

3 Method

Several previous works have been proposed to show the effectiveness of word order on text classification task, such as statistical n-gram features, or apply CNNs to learn an embedding of a fixed size region.

In this paper we also focus on learning the representations of small text regions which preserve the local internal structure information for text classification. The regions in document can be treat as fixed length continues subsequence of the document. More specifically, with w_i stands for the i-th(starting from 0) word of the document, we use $region(i, c)$ to denote the region with middle word w_i, where c is the region size. For instance, given a sentence such as *The food is not very good in this hotel*, $region(3, 5)$ means the subsequence *food is not very good*.

In this work, each word has two components, a word embedding and an additional local context unit. With the units and embeddings of words in a fixed size region, the representation of the region can be generated from two perspectives. In the rest of this section, we will introduce the local context units firstly, and two new architectures to generate the region embeddings with local context units will be introduced, finally we will introduce how we use the region embeddings on text classification.



(a) Word-Context Region Embedding (b) Context-Word Region Embedding

Figure 1: Architectures of region embedding using local context units in different perspectives

3.1 Local context units

In natural language processing tasks, words are commonly converted to low dimensional vectors as the inputs to neural networks. More formally, the embedding $\mathbf{e}_w$ of word w is represented by a column in a matrix $\mathbf{E} \in \mathcal{R}^{h \times v}$ with a look up layer, where v is the size of vocabulary, h is the embedding size.

To utilize information of words' relative positions, we learn a local context unit for each word in addition to word embedding, which both the unit and word embedding can be learned as model parameters. Formally, we define the local context unit $\mathbf{K}_{w_i} \in \mathcal{R}^{h \times c}$ of w_i as a matrix which can be looked up in the matrix $\mathbf{U} \in \mathcal{R}^{h \times c \times v}$ by w_i's index in vocabulary.

Each column in $\mathbf{K}_{w_i}$ can be used to interact with the local context word in corresponding relative position of w_i. In fact, the columns of a unit matrix can be regarded as distinction linear projection functions on the embeddings of words in the local context. The parameters of these projection functions(i.e., columns of each unit matrix) can be learned to capture the semantic and syntactic influence from the word to its context. Word embeddings are used as inputs to the projection functions, and we call the outputs projected word embeddings.

Formally, given the unit $\mathbf{K}_{w_i}$ of w_i and the embedding $\mathbf{e}_{w_{i+t}}$ of w_{i+t}, the $(\frac{c}{2} + t)$-th column in $\mathbf{K}_{w_i}$ is denoted by $\mathbf{K}_{w_i,t}$, where $-\frac{c}{2} <= t <= \frac{c}{2}$. We apply a element-wise multiplication(denoted by $\odot$) to compute the projected word embedding $\mathbf{e}^i_{w_{i+t}}$ of w_{i+t} as:

$$\mathbf{e}^i_{w_{i+t}} = \mathbf{K}_{w_i,t} \odot \mathbf{e}_{w_{i+t}}$$

Note that the middle column $\mathbf{K}_{w_i,\frac{c}{2}}$ of $\mathbf{K}_{w_i}$ can be regarded as a linear projection function on $\mathbf{e}_{w_i}$ itself, which transforms $\mathbf{e}^i_{w_i}$ to the same feature space as other projected embeddings'. For a local context word in a particular relative position of w_i, there is a corresponding linear projection function(a particular column of $\mathbf{K}_{w_i}$), thus our proposed local context units can utilize the local ordered word information in a novel way.

3.2 WORD-CONTEXT REGION EMBEDDING

We proposed two architectures to perform the region embedding from different perspectives. In the first proposed architecture, we focus on addressing the influence from the middle word to the context words. For example, in the sentence *The food is not very good in this hotel*, the occurrence of word *not* might bring a semantic reversal to the local region. Hence we compose the region embedding with word embeddings of all words in the region as well as the local context unit of the central word. we call this model word-context region embedding. Formally, for a $region(i, c)$, given the context unit $\mathbf{K}_{w_i}$ of center word w_i and the embeddings of words in the region, the projected embeddings are computed as the following matrix:

$$[\mathbf{e}^i_{w_{i-\frac{c}{2}}} \quad \mathbf{e}^i_{w_{i-\frac{c}{2}+1}} \quad ... \quad \mathbf{e}^i_{w_{i+\frac{c}{2}-1}} \quad \mathbf{e}^i_{w_{i+\frac{c}{2}}}]$$

To compose the representation of a region $region(i, c)$ noted as $\mathbf{r}_{i,c}$, we apply a max pooling on the row dimension of the projected embedding matrix. Figure 1a shows the details of the first model architecture. For instance, in the sentence *The food is not very good in this hotel*, the projected word embeddings in the $region(3, 5)$ are composed by the element-wise multiplications between columns in the local context unit of *not* and word embeddings of *food*, *is*, *not*, *very* and *good*. The embedding $\mathbf{r}_{3,5}$ of $region(3, 5)$ can be obtained by max pooling on projected word embedding matrix.

3.3 CONTEXT-WORD REGION EMBEDDING

The second architecture takes as a different view, which address the local context words' influence on the center word in the region, and we call this model context-word region embedding. Similarly, for a $region(i, c)$, the projected embeddings are represent as the following matrix:

$$[\mathbf{e}^{i-\frac{c}{2}}_{w_i} \quad \mathbf{e}^{i-\frac{c}{2}+1}_{w_i} \quad ... \quad \mathbf{e}^{i+\frac{c}{2}-1}_{w_i} \quad \mathbf{e}^{i+\frac{c}{2}}_{w_i}]$$

Figure 1b shows the details of the second model architecture, the difference between our two models is that inputs of max pooling layer are different, the Word-Context model uses the projected embeddings produced by the context unit of central word and word embeddings of context words, while the Context-Word model just the opposite.

3.4 REGION EMBEDDING FOR TEXT CLASSIFICATION

On text classification task, the documents commonly are of variable-length, which need to be represented as fixed size vectors. In order to show the effectiveness of our proposed region embedding models, we just sum up the embeddings of all regions to represent a document, and feed it an upper Full-Connected layer for text classification task.

Formally, the model can be represented as following:

$$f(\mathbf{x}; \mathbf{E}, \mathbf{U}, \mathbf{W}, \mathbf{b}) = g(\mathbf{W}^\top \sigma(\sum_{i=0}^{n} \mathbf{r}_{i,c}) + \mathbf{b})$$

where $\mathbf{x}$ denotes the input text sequence, $\mathbf{W}$ and $\mathbf{b}$ denote the weight matrix and bias of the fully connected layer respectively, g denotes the *softmax* function of the output layer, σ denotes the *softsign* function and n denotes the number of regions in the document, $\mathbf{r}$ is the region embedding which can be computed by the two proposed methods. $\mathbf{E}$, $\mathbf{U}$, $\mathbf{W}$ and $\mathbf{b}$ can be updated in the training period.

4 EXPERIMENTS

We report the experiments with proposed models in comparison with previous models. The Code is publicly available on the Internet.

4.1 Datasets

We use publicly available datasets from Zhang et al. (2015) to evaluate our models. There are total 8 text classification datasets, corresponding to sentiment analysis, news topics, question-answer, ontology extraction, respectively. Table 1 shows the descriptive statistics of datasets used in our experiments. To guarantee comparable indications, same evaluation protocol of Zhang et al. (2015) is employed.

Table 1: Statistics of Datasets

Dataset	Classes	Train Samples	Test Samples	Tasks
Yelp Review Polarity	2	560,000	38,000	Sentiment Analysis
Yelp Review Full	5	650,000	50,000	
Amazon Review Polarity	2	3,000,000	650,000	
Amazon Review Full	5	3,600,000	400,000	
AG's News	4	120,000	7,600	News Topics
Sogou News	5	450,000	60,000	
Yahoo! Answers	10	1,400,000	60,000	Question Answer
DBPedia	14	560,000	70,000	Ontology Extraction

4.2 Baselines

Our models are compared with several widely used supervised text classification models. We report the n-grams and TFIDF baselines from Zhang et al. (2015), as well as the character level convolutional model (char-CNN) of Zhang & LeCun (2015), the word level convolutional model (word-CNN) of Zhang & LeCun (2015), the character based convolution recurrent network (char-CRNN) of Xiao & Cho (2016), the very deep convolutional network (VDCNN) of Conneau et al. (2016), the Discriminative LSTM (D-LSTM) of Yogatama et al. (2017) and the bigram FastText (bigram-FastText) of Joulin et al. (2016).

4.3 Implementation Details

For data preprocessing, all the texts of datasets are tokenized by Stanford tokenizer and all words are converted to lower case. Words that appear only once are treated as out-of-vocabulary (OOV) items, and any stop words or symbols are kept. Additionally, padding length of $\frac{c}{2}$ are added to both the head and tail of each document.

Optimal hyperparameters are tuned with 10% of the training set on Yelp Review Full dataset, and identical hyperparameters are applied to all datasets: the dimension of word embedding is 128, the region size is 7 which means the shape of local context unit matrix of each word is 128×7, the learning rate is set to 1×10^{-4}, and the batch size is 16. For optimization, the embeddings of words and the units are randomly initialized with Gaussian Distribution. Adam (Kingma & Ba, 2014) are used as the optimizer. We do not use any extra regularization methods, such as L2 normalization or dropout. Algorithms are entirely implemented with TensorFlow and trained on NVIDIA Tesla P40 GPUs.

4.4 Results

Table 2 is the summary of our experimental results. performance summary of compared methods on listed datasets. We use underscores to represent the best known published results, and bold the best records. On six datasets of eight, our models beat or match the state-of-the-art with a performance gain highest to $0.7(\%)$. We beat all the previous models on all datasets except VDCNN, while the latter performs almost best on all classification tasks before. As a result, we slightly win VDCNN on six datasets and lost in two of Amazon datasets.

Although the slight advantage, our methods are remarkable on account of its simplicity along with divergence. Particularly, the upper layer structure of our models only uses a summing up operation,

Table 2: Test Set Accuracy [%] Compared to other Methods on several Datasets

Model	Yelp P.	Yelp F.	Amz. P.	Amz. F.	AG	Sogou	Yah. A.	DBP
BoW	92.2	58.0	90.4	54.6	88.8	92.9	68.9	96.6
ngrams	95.6	56.3	92.0	54.3	92.0	97.1	68.5	98.6
ngrams TFIDF	95.4	54.8	91.5	52.4	92.4	97.2	68.5	98.7
char-CNN	94.7	62.0	94.5	59.6	87.2	95.1	71.2	98.3
char-CRNN	94.5	61.8	94.1	59.2	91.4	95.2	71.7	98.6
bigram-FastText	95.7	63.9	94.6	60.2	92.5	96.8	72.3	98.6
VDCNN	95.7	64.7	**95.7**	**63.0**	91.3	96.8	73.4	98.7
D-LSTM	92.6	59.6	-	-	92.1	94.9	73.7	98.7
W.C.region.emb	**96.4**	**64.9**	95.1	60.9	**92.8**	**97.6**	**73.7**	**98.9**
C.W.region.emb	96.2	64.5	95.3	60.8	**92.8**	97.3	73.4	**98.9**

which is more concise and robust than any other deep or complex models. In fact, both of our two proposed models are effective.

4.5 Exploratory Experiments

In this subsection we are going to do a set of exploratory experiments to study the relative effect of each component in our model. Typical cases will be analyzed to validate properties of various aspects of our models. With the limitation of paper space, we only analyzed the Word-Context region embedding model in our exploratory experiments.

4.5.1 Effect of region size and embedding size

Our method uses a fixed size of region as contextual information just like CNN. So the selection of region size really matters. A small region may lose some long-distance patterns, whereas large regions will bring into more noises. Luckily, our models seem to be fairly insensitive towards kinds of datasets. Actually, we just select consistent configuration of size 7 for entire datasets and it is able to outperform best published results ever.

Figure 2 describes the performance on Yelp Review Full with different region sizes, and when the size equals to 1, the result is quite close to unigram FastText(Accuracy 60.7%), but still get 0.6(%) promotion. Intuitively, the central word can not influence other words except itself when the size equals to 1. The performance increases with the growth of region size up to 7.

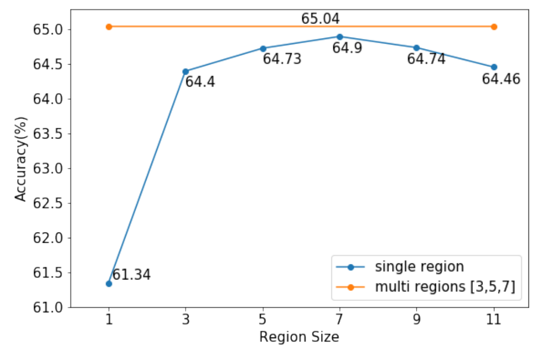


Figure 2: Effect of the hyperparameters ($region\ size$) on Yelp Review Full dataset. This Figure shows the comparison of single fixed region size 7 and multi sizes combination [3,5,7].

To further study the effectiveness of region sizes, we experiment our models with the combination of multi region sizes. In figure 2, the combination of multi region sizes *3,5,7* is slightly better than the best single region size 7. The effectiveness of multi-size combination can be explained by the difference of influence ranges between words, for example, word *very* only emphasizes the next word while *however* may lay stress on a wide range of the following words.

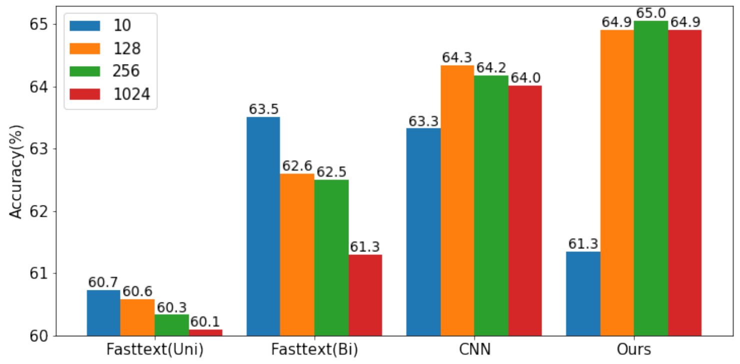


Figure 3: Effect of the hyperparameters ($embedding\ size$) on Yelp Review Full dataset. This figure shows the effect of different settings of embedding size among four kinds of models, unigram FastText, bigram FastText, CNN and ours. We unified use region size 7 for CNN and ours.

In addition to the analysis of region sizes, we further study the influence of word embedding dimensions. Figure 3 lists the comparative results on Yelp Review Full with different embedding dimensions. The result shows that our model is less prone to be over-fitting than FastText and CNN with the word embedding dimension increasing. In fact, the amount of parameters in our models is relatively large. Since we learn a specific unit for each word, under the same word embedding dimension, our parameter size has been expanded by *region size* times. Notice that the sizes of parameters are relatively consistent among *1024 in FastText*, *1024 in CNN* and *128 in ours*.

4.5.2 Effect of Context unit

In this section, we explore some comparative experiments to show the effectiveness of our proposed word specific context unit. The experiments are employed based on unigram FastText baseline, which has similar upper layer structure with our models. Table 3 illustrates the interrelated results.

Table 3: Comparative decomposition results on Yelp Review Full dataset. For FastText(Unigram), embedding dimension is 10. For FastText(Win-pool), W.C.region.emb(Scalar) and W.C.region.emb(our model), region size is 7 and embedding dimension is 128

.

Decomposition	Performance(%)
FastText(Unigram)	60.73
FastText(Win-pool)	61.01(+**0.28**)
W.C.region.emb(Scalar)	63.18(+**2.45**)
W.C.region.emb(Our model)	64.9(+**4.17**)

Firstly, we remove the entire context units from our model, which means it is just a variant version of unigram FastText, we call it FastText(Win-pool). The difference is that FastText sums up the word embeddings directly while FastText(Win-pool) sums up the window pooled embeddings in a stride of 1. It yields a slightly accuracy gain of (0.28%) than unigram FastText.

Secondly, we apply a simplified scalar version of context units to FastText(Win-pool). Distinguishable, the context unit of each word has the shape with $1 \times c$, hence it can be regarded as a broadcasting operation on corresponding word embeddings of its local context. We name this method W.C.region.emb(Scalar). Compared to the non-scalar method, it yields a huge parameter size reduction. but it already yields a significant gain of 2.45(%).

Furthermore, W.C.region.emb(our model) is the variant version of W.C.region.emb(Scalar) where the each column of scalar context unit is expanded to a dense vector. Each word's context unit has a shape with $h \times c$. Adding the low dimensional dense context unit improves performance by 4.17(%).

We can sense much from the procedure of decomposition, with the help of context unit, even a simpler scalar version promotes a lot. To have a better understanding of what context unit actually capture, heat maps are plotted for chosen word samples. Representative adversarial conjunctions like *however*, *but* and modifiers like *very*, *good*, *bad* are listed in Figure 4.

For each row of the figure, the intensity of the surrounding color box reflects the emphasis degree in the view of the central word. Qualitative but not fully rigorous, a normalized L2-norm is used to render the shade. Window size 7 is adopted default, annotation l_i(i$\leq$ 3) is denoted as left columns of the specific context unit, while r_i(i$\leq$ 3) denoted as the right part.

What the figure reflects are consistent with intuitive priors of human beings. In the perspective of *however*, right contexts play the key role for classification polarity because of the emotional turnaround, the color is indeed deeper in r_i than l_i, so does *but*. For word *very*, r_1 is more prominent than the rest of all, which capture some modified patterns like *very happy* or *very sad*. For word *good*, tendencies will be completely different for pattern *not good*, *very good* and *not that good*, which are intensive negative, intensive positive and slightly hesitated, separately, the position of l_1 will be strengthened as a result, so does word *bad*.

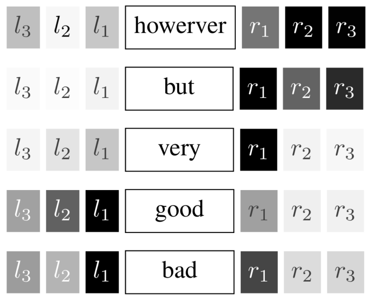


Figure 4: Heat maps of chosen words trained on Yelp Review Polarity, which belongs to a binary classification task of sentiment analysis. Each row represent the context unit of the central word. Region size is 7 and embedding size is 128.

Actually, from the motivation of word specified context units, we would like to believe this feature help capture syntactic and semantic influence on the relative position of its contexts. It can be regarded as a way to make use of word order in context unit covered regions.

4.5.3 VISUALIZATION

In this subsection, we will try to visualize the contribution of each word and selected phrase to the classification. Detailed visualization techniques have been introduced in Li et al. (2015). Here we generalize it the to the color rendering of multi-category version. Notice that not the self embedding acts here, but the projected embedding of self one on surrounding words' context unit.

Table 4: Visualization of chosen samples on Yelp Review Polarity dataset. Green denotes positive contribution while read denotes negative. Two methods are compared about without context unit(No C-unit) and with context unit(With C-unit).

Method	Sentence Samples	Phrase
No C-unit	get your wallet ready , the prices are crazy high	prices are crazy high
With C-unit	get your wallet ready , the prices are crazy high	prices are crazy high
No C-unit	nothing remarkable , but not bad either	but not bad either
With C-unit	nothing remarkable , but not bad either	but not bad either

For clarity, we choose a binary classification task of sentiment analysis. In Table 4, we list two real case samples in Yelp Review Polarity dataset, in which our model behave as expected. Words or artificially selected phrases are highlighted green if they are positive factors, red if they are negative. The intensity of the color indicates the extends of the polarity.

To have a better comparison, the methods of with and without context unit are both visualized. We abbreviate them as *With C-unit* and *Without C-unit*, respectively. For sentence *get your wallet ready, the prices are crazy high*, if no context unit is adopted, the word color will be reflected by its only embedding. The word *crazy* is colored positive, and *high* colored negative. Because the intensity of *crazy* is higher than *high*, the entire polarity of phrase *prices are crazy high* is totally positive, which is a complete mistake. But with context unit, things have changed quite a bit. Because of the exist of *high* as the context, the positive polarity of *crazy* vanishes. The negative polarity of *high* is stronger, phrase *prices are crazy high* performs positive overall. For another case *nothing remarkable, but not bad either*, things seem more interesting. If without context-unit , *remarkable* will be positive, while *nothing, not, bad* perform negative, respectively. But with context-unit, the polarity of the part ahead of *but* is reduced, meanwhile the polarity of *not* and *bad* flip attractively. As a result, phrase *but not bad either* performs positive overall.

5 CONCLUSION

This paper propose two novel architectures for text classification tasks that learn task specific region embeddings without hand crafted features. To utilize the information of words' relative positions, we learn a local context unit for each word in addition to word embedding. Our models achieve state-of-the-art performances on six benchmark text classification datasets, and the visualization experiments show that our proposed local context unit can capture the semantic and synthetic information for each word.

Noticed that the power of the local context unit on learning task related region embeddings, we are interested in its ability to semi-supervised learning. At the same time, we are also curious about whether we can do more natural language processing tasks by adding more complex and deep upper layers.

REFERENCES

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. Very deep convolutional networks for natural language processing. *arXiv preprint arXiv:1606.01781*, 2016.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *Journal of machine learning research*, 9(Aug):1871–1874, 2008.

Thorsten Joachims. Making large-scale svm learning practical. Technical report, Technical Report, SFB 475: Komplexitätsreduktion in Multivariaten Datenstrukturen, Universität Dortmund, 1998.

Rie Johnson and Tong Zhang. Effective use of word order for text categorization with convolutional neural networks. *arXiv preprint arXiv:1412.1058*, 2014.

Rie Johnson and Tong Zhang. Semi-supervised convolutional neural networks for text categorization via region embedding. In *Advances in neural information processing systems*, pp. 919–927, 2015.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.

Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

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

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. Visualizing and understanding neural models in nlp. *arXiv preprint arXiv:1506.01066*, 2015.

Andrew McCallum, Kamal Nigam, et al. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, volume 752, pp. 41–48. Madison, WI, 1998.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pp. 3111–3119, 2013.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pp. 79–86. Association for Computational Linguistics, 2002.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 1631–1642, 2013.

Sida Wang and Christopher D Manning. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pp. 90–94. Association for Computational Linguistics, 2012.

Yijun Xiao and Kyunghyun Cho. Efficient character-level document classification by combining convolution and recurrent layers. *arXiv preprint arXiv:1602.00367*, 2016.

Dani Yogatama, Chris Dyer, Wang Ling, and Phil Blunsom. Generative and discriminative text classification with recurrent neural networks. *stat*, 1050:6, 2017.

Xiang Zhang and Yann LeCun. Text understanding from scratch. *arXiv preprint arXiv:1502.01710*, 2015.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pp. 649–657, 2015.