Disconnected Recurrent Neural Networks for Text Categorization

Baoxin Wang
Joint Laboratory of HIT and iFLYTEK, iFLYTEK Research, Beijing, China
bxwang2@iflytek.com

Abstract

Recurrent neural network (RNN) has achieved remarkable performance in text categorization. RNN can model the entire sequence and capture long-term dependencies, but it does not do well in extracting key patterns. In contrast, convolutional neural network (CNN) is good at extracting local and position-invariant features. In this paper, we present a novel model named disconnected recurrent neural network (DRNN), which incorporates position-invariance into RNN. By limiting the distance of information flow in RNN, the hidden state at each time step is restricted to represent words near the current position. The proposed model makes great improvements over RNN and CNN models and achieves the best performance on several benchmark datasets for text categorization.

1 Introduction

Text categorization is a fundamental and traditional task in natural language processing (NLP), which can be applied in various applications such as sentiment analysis (Tang et al., 2015), question classification (Zhang and Lee, 2003) and topic classification (Tong and Koller, 2001). Nowadays, one of the most commonly used methods to handle the task is to represent a text with a low dimensional vector, then feed the vector into a softmax function to calculate the probability of each category. Recurrent neural network (RNN) and convolutional neural network (CNN) are two kinds of neural networks usually used to represent the text.

RNN can model the whole sequence and capture long-term dependencies (Chung et al., 2014). However, modeling the entire sequence sometimes can be a burden, and it may neglect key parts for text categorization (Yin et al., 2017). In contrast, CNN is able to extract local and position-invariant features well (Scherer et al., 2010; Collobert et al., 2011). Table 1 is an example of topic classification, where both sentences should be classified as *Science and Technology*. The key phrase that determines the category is *unsolved mysteries of mathematics*, which can be well extracted by CNN due to position-invariance. RNN, however, doesn't address such issues well because the representation of the key phrase relies on all the previous terms and the representation changes as the key phrase moves.

case1: *One of the seven great* ***unsolved mysteries of mathematics*** *may have been cracked by a reclusive Russian.*
case2: *A reclusive Russian may have cracked one of the seven great* ***unsolved mysteries of mathematics.***

Table 1: Examples of topic classification

In this paper, we incorporate position-invariance into RNN and propose a novel model named Disconnected Recurrent Neural Network (DRNN). Concretely, we disconnect the information transmission of RNN and limit the maximal transmission step length as a fixed value k, so that the representation at each step only depends on the previous $k-1$ words and the current word. In this way, DRNN can also alleviate the burden of modeling the entire document. To maintain the position-invariance, we utilize max pooling to extract the important information, which has been suggested by Scherer et al. (2010).

Our proposed model can also be regarded as a special 1D CNN where convolution kernels are replaced with recurrent units. Therefore, the maximal transmission step length can also be consid-

Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Long Papers), pages 2311–2320
Melbourne, Australia, July 15 - 20, 2018.

ered as the window size in CNN. Another difference to CNN is that DRNN can increase the window size k arbitrarily without increasing the number of parameters.

We also find that there is a trade-off between position-invariance and long-term dependencies in the DRNN. When the window size is too large, the position-invariance will disappear like RNN. By contrast, when the window size is too small, we will lose the ability to model long-term dependencies just like CNN. We find that the optimal window size is related to the type of task, but affected little by training dataset sizes. Thus, we can search the optimal window size by training on a small dataset.

We conduct experiments on seven large-scale text classification datasets introduced by Zhang et al. (2015). The experimental results show that our proposed model outperforms the other models on all of these datasets.

Our contributions can be concluded as follows:

1. We propose a novel model to incorporate position-variance into RNN. Our proposed model can both capture long-term dependencies and local information well.

2. We study the effect of different recurrent units, pooling operations and window sizes on model performance. Based on this, we propose an empirical method to find the optimal window size.

3. Our proposed model outperforms the other models and achieves the best performance on seven text classification datasets.

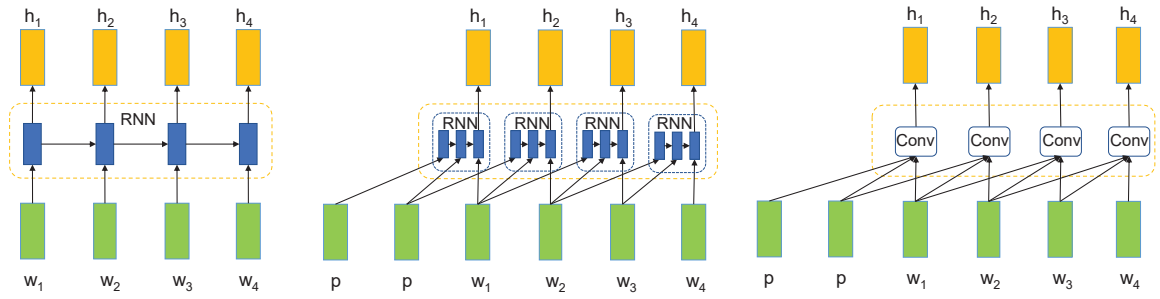
2 Related Work

Deep neural networks have shown great success in many NLP tasks such as machine translation (Bahdanau et al., 2015; Tu et al., 2016), reading comprehension (Hermann et al., 2015), sentiment classification (Tang et al., 2015), etc. Nowadays, nearly most of deep neural networks models are based on CNN or RNN. Below, we will introduce some important works about text classification based on them.

Convolutional Neural Networks CNN has been used in natural language processing because of the local correlation and position-invariance. Collobert et al. (2011) first utilize 1D CNN in part of speech (POS), named entity recognition (NER) and semantic role labeling (SRL). Kim (2014) proposes to classify sentence by encoding a sentence with multiple kinds of convolutional filters. To capture the relation between words, Kalchbrenner et al. (2014) propose a novel CNN model with a dynamic k-max pooling. Zhang et al. (2015) introduce an empirical exploration on the use of character-level CNN for text classification. Shallow CNN cannot encode long-term information well. Therefore, Conneau et al. (2017) propose to use very deep CNN in text classification and achieve good performance. Similarly, Johnson and Zhang (2017) propose a deep pyramid CNN which both achieves good performance and reduces training time.

Recurrent Neural Networks RNN is suitable for handling sequence input like natural language. Thus, many RNN variants are used in text classification. Tang et al. (2015) utilize LSTM to model the relation of sentences. Similarly, Yang et al. (2016) propose hierarchical attention model which incorporates attention mechanism into hierarchical GRU model so that the model can better capture the important information of a document. Wang and Tian (2016) incorporate the residual networks (He et al., 2016) into RNN, which makes the model handle longer sequence. Xu et al. (2016) propose a novel LSTM with a cache mechanism to capture long-range sentiment information.

Hybrid model Some researchers attempt to combine the advantages of CNN and RNN. (Xiao and Cho, 2016) extract local and global features by CNN and RNN separately. (Lai et al., 2015) firstly model sentences by RNN, and then use CNN to get the final representation. Shi et al. (2016) replace convolution filters with deep LSTM, which is similar to what is proposed in this paper. The main differences are as follows. Firstly, they regard their models as CNN and set a small window size of 3, while we propose to use a large window size. We argue that small window size makes the model lose the ability to capture long-term dependencies. Secondly, we utilize max pooling but not mean pooling, because max pooling can maintain position-invariance better (Scherer et al., 2010). Finally, our DRNN model is more general and can make use of different kinds of recurrent units. We find that using GRU as recurrent units outperforms LSTM which is utilized by Shi et al. (2016).



(a) RNN (b) DRNN (c) CNN

Figure 1: Three model architectures. In order to ensure the consistency of the hidden output, we pad $k-1$ zero vectors on the left of the input sequence for DRNN and CNN. Here window size k is 3.

3 Method

3.1 Recurrent Neural Network (RNN)

RNN is a class of neural network which models a sequence by incorporating the notion of time step (Lipton et al., 2015). Figure 1(a) shows the structure of RNN. Hidden states at each step depend on all the previous inputs, which sometimes can be a burden and neglect the key information (Yin et al., 2017).

A variant of RNN has been introduced by Cho et al. (2014) with the name of gated recurrent unit (GRU). GRU is a special type of RNN, capable of learning potential long-term dependencies by using gates. The gating units can control the flow of information and mitigate the vanishing gradients problem. GRU has two types of gates: reset gate $\mathbf{r}_t$ and update gate $\mathbf{z}_t$. The hidden state $\mathbf{h}_t$ of GRU is computed as

$$\mathbf{h}_t = (\mathbf{1} - \mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \tilde{\mathbf{h}}_t \quad (1)$$

where $\mathbf{h}_{t-1}$ is the previous state, $\tilde{\mathbf{h}}_t$ is the candidate state computed with new input information and $\odot$ is the element-wise multiplication. The update gate $\mathbf{z}_t$ decides how much new information is updated. $\mathbf{z}_t$ is computed as follows:

$$\mathbf{z}_t = \sigma(\mathbf{W}_z \mathbf{x}_t + \mathbf{U}_z \mathbf{h}_{t-1}) \quad (2)$$

here $\mathbf{x}_t$ is the input vector at step t. The candidate state $\tilde{\mathbf{h}}_t$ is computed by

$$\tilde{\mathbf{h}}_t = tanh(\mathbf{W}\mathbf{x}_t + \mathbf{U}(\mathbf{r}_t \odot \mathbf{h}_{t-1})) \quad (3)$$

where $\mathbf{r}_t$ is the reset gate which controls the flow of previous information. Similarly to the update gate, the reset gate $\mathbf{r}_t$ is computed as:

$$\mathbf{r}_t = \sigma(\mathbf{W}_r \mathbf{x}_t + \mathbf{U}_r \mathbf{h}_{t-1}) \quad (4)$$

We can see that the representation of step t depends upon all the previous input vectors. Thus, we can also express the tth step state shown in Equation (5).

$$\mathbf{h}_t = GRU(\mathbf{x}_t, \mathbf{x}_{t-1}, \mathbf{x}_{t-2}, ..., \mathbf{x}_1) \quad (5)$$

3.2 Disconntected Recurrent Neural Networks (DRNN)

To reduce the burden of modeling the entire sentence, we limit the distance of information flow in RNN. Like other RNN variants, we feed the input sequence into an RNN model and generate an output vector at each step. One important difference from RNN is that the state of our model at each step is only related to the previous $k-1$ words but not all the previous words. Here k is a hyperparameter called window size that we need to set.

Our proposed model DRNN is illustrated in Figure 1(b). Since the output at each step only depends on the previous $k-1$ words and current word, the output can also be regarded as a representation of a phrase with k words. Phrases with the same k words will always have the same representation no matter where they are. That is, we incorporate the position-invariance into RNN by disconnecting the information flow of RNN.

Similarly, we can get the state $\mathbf{h}_t$ as follows:

$$\mathbf{h}_t = RNN(\mathbf{x}_t, \mathbf{x}_{t-1}, \mathbf{x}_{t-2}, ..., \mathbf{x}_{t-k+1}) \quad (6)$$

Here k is the window size, and RNN can be naive RNN, LSTM (Hochreiter and Schmidhuber, 1997), GRU or any other kinds of recurrent units.

3.3 Comparison with Convolutional Neural Network (CNN)

DRNN can be considered as a special 1D CNN which replace the convolution filters with recur-

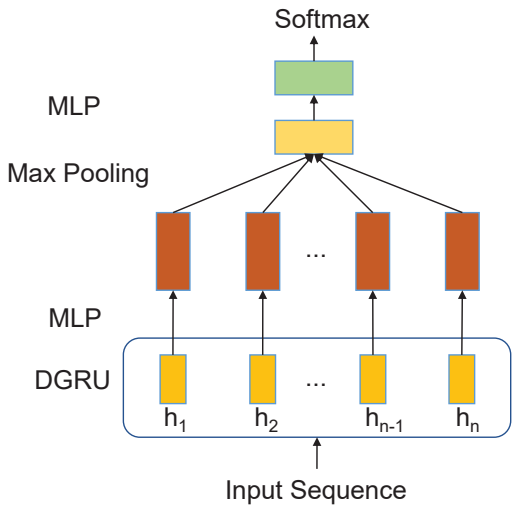


Figure 2: Model architecture

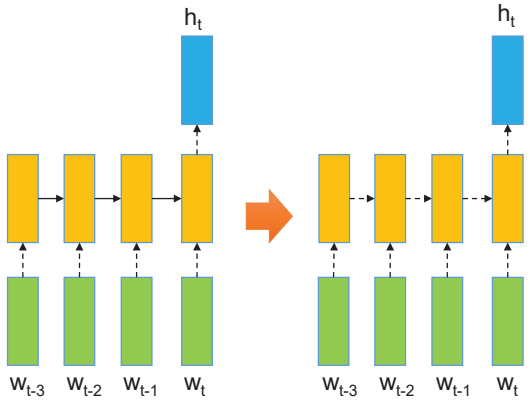


Figure 3: Dropout in DRNN. The dashed arrows indicate connections where dropout is applied. The left model only applies dropout in input and output layers, but the right model applies dropout in hidden states.

rent units. Let $\mathbf{x}_t$ denote the tth input word vector. Then for each position t we can get a window vector $\mathbf{c}_t$.

$$\mathbf{c}_t = [\mathbf{x}_t, \mathbf{x}_{t-1}, \mathbf{x}_{t-2}, ..., \mathbf{x}_{t-k+1}] \quad (7)$$

here, we concatenate k word vectors and generate vector $\mathbf{c}_t$. Then we can get the output of convolution as follows:

$$\mathbf{h}_t = \mathbf{W}\mathbf{c}_t + \mathbf{b} \quad (8)$$

where $\mathbf{W}$ is a set of convolution filters and $\mathbf{b}$ is a bias vector. Then a pooling operation can be applied after the convolutional layer and generate a fixed size vector (Kim, 2014). Similarly to RNN and DRNN, we can also represent the context vector of CNN as followings:

$$\mathbf{h}_t = Conv(\mathbf{x}_t, \mathbf{x}_{t-1}, \mathbf{x}_{t-2}, ..., \mathbf{x}_{t-k+1}) \quad (9)$$

Obviously, the parameters of convolution filters $\mathbf{W}$ increase as the window size k increases. By contrast, for DRNN the parameters do not increase with the increase of window size. Hence, DRNN can mitigate overfitting problem caused by the increase of parameters.

3.4 DRNN for Text Classification

DRNN is a general model framework, which can be used for a variety of tasks. In this paper, we only discuss how to apply DRNN in text categorization.

We utilize GRU as recurrent units of DRNN and get the context representation of each step. Every context vector can be considered as a representation of a text fragment. Then we feed the context vectors into a multi-layer perceptron (MLP) to extract high-level features as illustrated in Figure 2. Before feeding the vectors into MLP, we utilize Batch Normalization (Ioffe and Szegedy, 2015) after DRNN, so that the model can alleviate the internal covariate shift problem. To get the text representation vector, we apply max pooing after MLP layer to extract the most important information and position-invariant features (Scherer et al., 2010).

Finally, We feed the text representation vector into an MLP with rectified linear unit (ReLU) activation and send the output of MLP to a softmax function to predict the probability of each category. We use cross entropy loss function as follows:

$$H(y, \hat{y}) = \sum_i y_i \log \hat{y}_i \quad (10)$$

where $\hat{y}_i$ is the predicted probability and y_i is the true probability of class i.

To alleviate the overfitting problem, we apply dropout regularization (Srivastava et al., 2014) in DRNN model. Dropout is usually applied in the input and output layers but not the hidden states of RNN, because the number of previous states is variable (Zaremba et al., 2014). In contrast, our DRNN model has a fixed window size for output at each step, so we also apply dropout in the hidden states. In this paper, we apply dropout in the input layer, output layer, and hidden states. The Figure 3 shows the difference to apply dropout between

	AG	DBP	Yelp P.	Yelp F.	Yah. A.	Amz. F.	Amz. P.
Tasks	News	Ontology	SA	SA	QA	SA	SA
Train dataset	120k	560k	560k	650k	1.4M	3.6M	3M
Test dataset	7.6k	70k	38k	50k	60k	400k	650k
Average Lengths	45	55	153	155	112	93	91
Classes Number	4	14	2	5	10	5	2

Table 2: Dataset information. Here SA refers to sentiment analysis, and QA refers to question answering.

RNN and DRNN.

4 Experiments

4.1 Experimental Settings

Datasets Introduction We use 7 large-scale text classification datasets which are proposed by Zhang et al. (2015). We summarize the datasets in Table 2. AG corpus is news and DBPedia is an ontology which comes from the Wikipedia. Yelp and Amazon corpus are reviews for which we should predict the sentiment. Here *P.* means that we only need to predict the polarities of the dataset, while *F.* indicates that we need predict the star number of the review. *Yahoo! Answers* (Yah. A.) is a question answering dataset. We can see that these datasets contain various domains and sizes, which would be credible to validate our models.

Implementation Details We tokenize all the corpus with NLTK's tokenizer (Bird and Loper, 2004). We limit the vocabulary size of each dataset as shown in Table 3. The words not in vocabulary are replaced with a special token *UNK*. Table 3 also shows the window sizes that we set for these datasets.

We utilize the 300D GloVe 840B vectors (Pennington et al., 2014) as our pre-trained word embeddings. For words that do not appear in GloVe, we average the vector representations of 8 words around the word in training dataset as its word vector, which has been applied by Wang and Jiang (2016). When training our model, word embeddings are updated along with other parameters.

We use Adadelta (Zeiler, 2012) to optimize all the trainable parameters. The hyperparameter of Adadelta is set as Zeiler (2012) suggest that ϵ is $1e-6$ and ρ is 0.95. To avoid the gradient explosion problem, we apply gradient norm clipping (Pascanu et al., 2013). The batch size is set to 128 and all the dimensions of input vectors and hidden states are set to 300.

Corpus	Window size	Vocabulary size
AG	15	100k
DBP.	15	500k
Yelp P.	20	200k
Yelp F.	20	200k
Yah. A.	20	500k
Amz. F.	15	500k
Amz. P.	15	500k

Table 3: Experimental settings

4.2 Experimental Results

Table 4 shows that our proposed model significantly outperforms all the other models in 7 datasets. DRNN does not have too many hyperparameters. The main hyperparameter is the window size which can be determined by an empirical method.

The top block shows the traditional methods and some other neural networks which are not based on RNN or CNN. The linear model (Zhang et al., 2015) achieves a strong baseline in small datasets, but performs not well in large data. FastText (Joulin et al., 2017) and region embedding methods (Qiao et al., 2018) achieve comparable performance with other CNN and RNN based models.

The RNN based models are listed in the second block and CNN based models are in the third block. The D-LSTM (Yogatama et al., 2017) is a discriminative LSTM model. Hierarchical attention network (HAN) (Yang et al., 2016) is a hierarchical GRU model with attentive pooling. We can see that very deep CNN (VDCNN) (Conneau et al., 2017) performs well in large datasets. However, VDCNN is a CNN model with 29 convolutional layers, which needs to be tuned more carefully. By contrast, our proposed model can achieve

Models	AG	DBP.	Yelp P.	Yelp F.	Yah. A.	Amz. F.	Amz. P.
Linear model (Zhang et al., 2015)	7.64	1.31	4.36	40.14	28.96	44.74	7.98
FastText (Joulin et al., 2017)	7.5	1.4	4.3	36.1	27.7	39.8	5.4
Region.emb (Qiao et al., 2018)	7.2	1.1	4.7	35.1	26.3	39.1	4.7
D-LSTM (Yogatama et al., 2017)	7.9	1.3	7.4	40.4	26.3	-	-
HAN (Yang et al., 2016)	-	-	-	-	24.2	36.4	-
char-CNN (Zhang et al., 2015)	9.51	1.55	4.88	37.95	28.80	40.43	4.93
word-CNN (Zhang et al., 2015)	8.55	1.37	4.60	39.58	28.84	42.39	5.51
VDCNN (Conneau et al., 2017)	8.67	1.29	4.28	35.28	26.57	37.00	4.28
char-CRNN (Xiao and Cho, 2016)	8.64	1.43	5.51	38.18	28.26	40.77	5.87
DRNN	**5.53**	**0.81**	**2.73**	**30.85**	**23.74**	**35.57**	**3.51**

Table 4: Error rates (%) on seven datasets

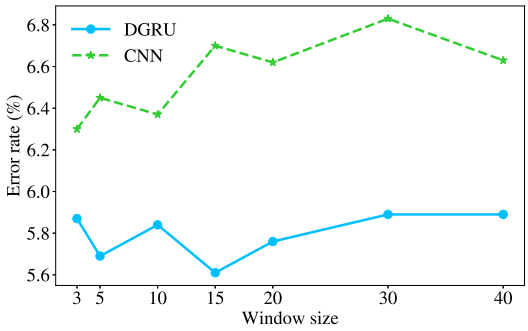


Figure 4: DGRU compared with CNN

Models	AG	DBP.	Yelp P.
CNN	6.30	1.13	4.08
GRU	6.25	0.96	3.41
LSTM	6.20	0.90	3.20
DRNN	**5.53**	**0.81**	**2.73**

Table 5: Comparison with RNN and CNN. Table shows the error rate (%) on three datasets.

better performance in these datasets by simply setting a large window size.

Char-CRNN (Xiao and Cho, 2016) in the fourth block is a model which combines position-invariance of CNN and long-term dependencies of RNN. Nevertheless, they do not achieve great improvements over other models. They first utilize convolution operation to extract position-invariant features, and then use RNN to capture long-term dependencies. Here, modeling the whole sequence with RNN leads to a loss of position-invariance. Compared with their model, our model can better maintain the position-invariance by max pooling (Scherer et al., 2010). Table 4 shows that our model achieves 10-50% relative error reduction compared with char-CRNN in these datasets.

4.3 Comparison with RNN and CNN

In this section, we compare DRNN with CNN, GRU and LSTM (Hochreiter and Schmidhuber, 1997). To make these models comparable, we implement these models with the same architecture shown in Figure 2. We just replace the DRNN with CNN or RNN.

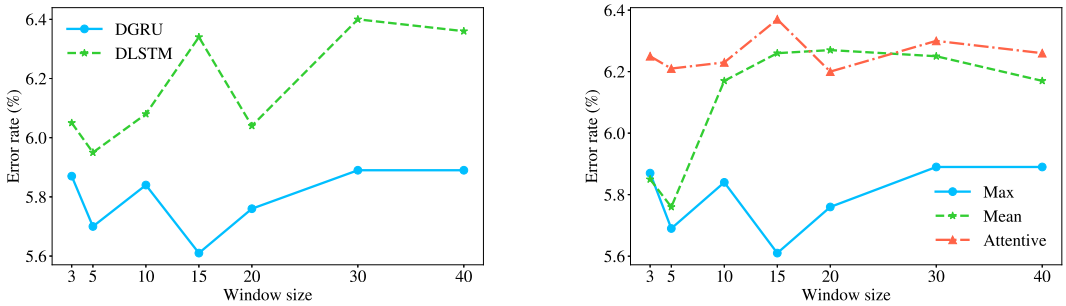
we firstly compare DRNN with CNN on AG dataset. Figure 4 shows that DRNN performs far better than CNN. In addition, the optimal window size of CNN is 3, while for DRNN the optimal window size is 15. It indicates that DRNN can model longer sequence as window size increases. By contrast, simply increasing the window size of CNN only results in overfitting. That is also why Conneau et al. (2017) design complex CNN models to learn long-term dependencies other than simply increase the window size of convolution filters.

In addition, we also compare our model with GRU and LSTM. The experimental results are shown in Table 5. Our model DRNN achieves much better performance than GRU and LSTM.

Qualitative Analysis To investigate why DGRU performs better than CNN and GRU, we do some error analysis on Yelp P. dataset. Table 6 shows two examples which have been both

case1: ***I love*** *Hampton Inn* ***but*** *this location is in serious need of remodeling and some deep cleaning. Musty smell everywhere.*
case2: ***Pretty good service***, ***but*** *really busy and noisy!! It gets a little overwhelming because the sales people are very knowledgeable and bombard you with useless techy information to I guess impress you?? Anyways I bought the Ipad 3 and it is* ***freaking awesome*** *and makes up for the store. I would give the Ipad 3 a gazillion stars if I could. I left it at home today and got really sad when I was driving away. Boo Hoo!!*

Table 6: Examples of error analysis. The case 1 is a negative review and case 2 is a positive review. The first example is misclassified by CNN and classified correctly by GRU. The second one is just the contrary. DGRU classify both examples correctly.



(a) Comparison of recurrent units

(b) Comparison of pooling methods

Figure 5: Component comparison

classified correctly by DRNN. The first example is misclassified by CNN and classified correctly by GRU. It is just contrary to the second example. Considering the first example, CNN may extract some key phrases such as *I love* and misclassifies the example as $Positive$, while GRU can model long sequence and capture the information after *but*. For the second example, however, GRU still captures the information after *but* and neglects the key phrases such as *pretty good service* and *freaking awesome*, which leads to the wrong classification.

DGRU can both extract the local key features such as *pretty good service* and capture long-term information such as the sentence after *but*, which makes it perform better than GRU and CNN.

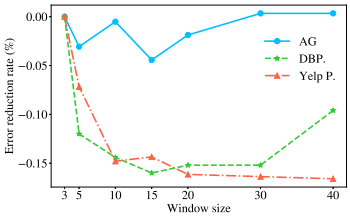
4.4 Component Analysis

Recurrent Unit In this part, we study the impact of different recurrent units on the effectiveness of DRNN. We choose three types of recurrent units: naive RNN, LSTM and GRU which have been compared by Chung et al. (2014). We carry out the experiments with different window sizes to eliminate the impact of window sizes. All the experiments in this part are conducted on the AG dataset.

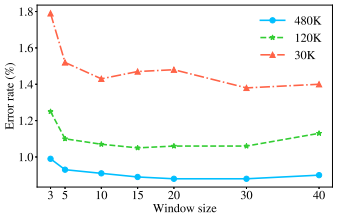
We find that the disconnected naive RNN performs just a little worse than disconnected LSTM (DLSTM) and disconnected GRU (DGRU) when the window size is lower than 5. However, when the window size is more than 10, its performance decreases rapidly and the error rate becomes even more than 20%. We believe that it is due to vanishing gradient problem of naive RNN.

From Figure 5(a), we can see that window sizes affect the performance of DGRU and DLSTM. DGRU achieves the best performance when the window size is 15, while the best window size for DLSTM is 5. The performance of DGRU is always better than DLSTM no matter what the window size is. We also find that the DGRU model converges faster than DLSTM in the process of training. Therefore, we apply GRU as recurrent units of DRNN in this paper for all the other experiments.

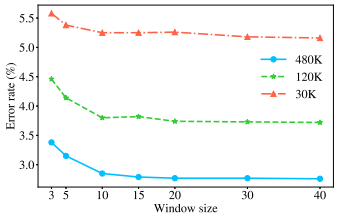
Pooling Method Pooling is a kind of method to subsample the values to capture more important information. In NLP, pooling can also convert a variable-length tensor or vector into a fixed-length



(a) Different tasks



(b) Different training sets of DBP.



(c) Different training sets of Yelp P.

Figure 6: Window size analysis. For better comparing the trends of different tasks, (a) shows the error reduction rates with different window sizes. (b) and (c) show the error rates of DBP. and Yelp P. with different training set numbers.

one, so that it can be dealt with more easily. There're several kinds of pooling methods such as max pooling, mean pooling and attentive pooling (dos Santos et al., 2016).

We still conduct the experiments on AG dataset. Figure 5(b) shows the experimental results of three pooling methods along with different window sizes. From Figure 5(b), we can see that the DRNN model with max pooling performs better than the others. This may be because that max pooling can capture position-invariant features better (Scherer et al., 2010). We find attentive pooling is not significantly affected by window sizes. However, the performance of mean pooling becomes worse as the window becomes larger.

4.5 Window size analysis

In this section, we mainly study what factors affect the optimal window size. In addition to the recurrent units and pooling methods discussed above, we believe the optimal window size may be also related to the amount of training data and the type of task.

In order to study the factors that affect the optimal window size, we conduct experiments on three datasets: AG, DBP and Yelp Polarity. To eliminate the influence of differrnt training data sizes, we conduct experiments with the same training data size. From Figure 6(a) we can see that the type of task has a great impact on the optimal window size. For AG and DBPedia, the optimal window size is 15. However, for Yelp P. the optimal window size is 40 or even larger. The result is intuitive, because sentiment analysis such as Yelp often involves long-term dependencies (Tang et al., 2015), while topic classification such as AG and DBPedia relys more on the key phrases.

From Figure 6(b) and Figure 6(c) we can see the effect of different training data sizes on the optimal window size. Surprisingly, the effect of different training data sizes on the optimal window size seems little. We can see that for both DBPedia and Yelp corpus, the trend of error rate with the window size is similar. This shows that the number of training data has little effect on the choice of the optimal window size. It also provides a good empirical way for us to choose the optimal window size. That is, conducting experiments on a small dataset first to select the optimal window size.

5 Conclusion

In this paper, we incorporate position-invariance into RNN, so that our proposed model DRNN can both capture key phrases and long-term dependencies. We conduct experiments to compare the effects of different recurrent units and pooling operations. In addition, We also analyze what factors affect the optimal window size of DRNN and present an empirical method to search it. The experimental results show that our proposed model outperforms CNN and RNN models, and achieve the best performance in seven large-scale text classification datasets.

Acknowledgments

This work was supported by the National Key Research and Development Program of China (No. 2016YFC0800806). I would like to thank Jianfeng Li, Shijin Wang, Ting liu, Guoping Hu, Shangmin Guo, Ziyue Wang, Xiaoxue Wang and the anonymous reviewers for their insightful comments and suggestions.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*.

Steven Bird and Edward Loper. 2004. Nltk: the natural language toolkit. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*. Association for Computational Linguistics, page 31.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* .

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* .

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* 12:2493–2537.

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2017. Very deep convolutional networks for text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. volume 1, pages 1107–1116.

Cıcero Nogueira dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive pooling networks. *CoRR, abs/1602.03609* 2(3):4.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. pages 770–778.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*. pages 1693–1701.

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

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*. pages 448–456.

Rie Johnson and Tong Zhang. 2017. Deep pyramid convolutional neural networks for text categorization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 562–570.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. volume 2, pages 427–431.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188* .

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pages 1746–1751.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *AAAI*. volume 333, pages 2267–2273.

Zachary C Lipton, John Berkowitz, and Charles Elkan. 2015. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019* .

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*. pages 1310–1318.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. pages 1532–1543.

Chao Qiao, Bo Huang, Guocheng Niu, Daren Li, Daxiang Dong, Wei He, Dianhai Yu, and Hua Wu. 2018. A new method of region embedding for text classification. In *International Conference on Learning Representations*.

Dominik Scherer, Andreas Müller, and Sven Behnke. 2010. Evaluation of pooling operations in convolutional architectures for object recognition. In *International conference on artificial neural networks*. Springer, pages 92–101.

Yangyang Shi, Kaisheng Yao, Le Tian, and Daxin Jiang. 2016. Deep lstm based feature mapping for query classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 1501–1511.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15(1):1929–1958.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 conference on empirical methods in natural language processing*. pages 1422–1432.

Simon Tong and Daphne Koller. 2001. Support vector machine active learning with applications to text classification. *Journal of machine learning research* 2(Nov):45–66.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. *arXiv preprint arXiv:1601.04811* .

Shuohang Wang and Jing Jiang. 2016. Learning natural language inference with lstm. In *Proceedings of NAACL-HLT*. pages 1442–1451.

Yiren Wang and Fei Tian. 2016. Recurrent residual learning for sequence classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. pages 938–943.

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

Jiacheng Xu, Danlu Chen, Xipeng Qiu, and Xuanjing Huang. 2016. Cached long short-term memory neural networks for document-level sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. pages 1660–1669.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 1480–1489.

Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. 2017. Comparative study of cnn and rnn for natural language processing. *arXiv preprint arXiv:1702.01923* .

Dani Yogatama, Chris Dyer, Wang Ling, and Phil Blunsom. 2017. Generative and discriminative text classification with recurrent neural networks. *arXiv preprint arXiv:1703.01898* .

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329* .

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701* .

Dell Zhang and Wee Sun Lee. 2003. Question classification using support vector machines. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*. ACM, pages 26–32.

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