Information Aggregation via Dynamic Routing for Sequence Encoding

Jingjing Gong, Xipeng Qiu*, Shaojing Wang, Xuanjing Huang
Shanghai Key Laboratory of Intelligent Information Processing, Fudan University
School of Computer Science, Fudan University
{jjgong15, xpqiu, sjwang17, xjhuang}@fudan.edu.cn

Abstract

While much progress has been made in how to encode a text sequence into a sequence of vectors, less attention has been paid to how to aggregate these preceding vectors (outputs of RNN/CNN) into fixed-size encoding vector. Usually, a simple max or average pooling is used, which is a bottom-up and passive way of aggregation and lack of guidance by task information. In this paper, we propose an aggregation mechanism to obtain a fixed-size encoding with a dynamic routing policy. The dynamic routing policy is dynamically deciding that *what* and *how much* information need be transferred from each word to the final encoding of the text sequence. Following the work of Capsule Network, we design two dynamic routing policies to aggregate the outputs of RNN/CNN encoding layer into a final encoding vector. Compared to the other aggregation methods, dynamic routing can refine the messages according to the state of final encoding vector. Experimental results on five text classification tasks show that our method outperforms other aggregating models by a significant margin. Related source code is released on our github page[1].

1 Introduction

Learning the distributed representation of text sequences, such as sentences or documents, is crucial to a wide range of important natural language processing applications. A primary challenge is how to encode the variable-length text sequence into a fixed-size vector, which should fully capture the semantics of text.

Many successful text encoding methods usually contain three key steps: (1) converting each word in a text sequence into its embedding; (2) taking as input the sequence of word embeddings, and computing the context-aware representation for each word with a recurrent neural network (RNN) (Hochreiter and Schmidhuber, 1997; Chung et al., 2014) or convolutional neural network (CNN) (Collobert et al., 2011; Kim, 2014); (3) summarizing the sentence meaning into a fixed-size vector by an aggregation operation. Then, these models are trained by combining a downstream task in a supervised or unsupervised way.

Currently, much attention is paid to the first two steps, while the aggregation step is less emphasized on. Some simple aggregation methods, such as max (or average) pooling, is used to sum the RNN hidden states or convolved vectors, computed in the previous step, into a single vector. This kind of methods aggregate information in a bottom-up and passive way and are lack of the guide of task information. Recently, several works employ self-attention mechanism (Lin et al., 2017; Yang et al., 2016) on top of the recurrent or convolutional encoding layer to replace simple pooling. A basic assumption is that the words (or even sentences) are not equally important. One or several task-specific context vectors are used to assign a different weight to each word and select task-specific encodings. The context vectors are parameters learned jointly with other parameters during the training process. These attentive aggregation can select task-dependent information. However, the context vectors are fixed once learned.

* Corresponding Author

[1] https://github.com/FudanNLP/Capsule4TextClassification

Proceedings of the 27th International Conference on Computational Linguistics, pages 2742–2752
Santa Fe, New Mexico, USA, August 20-26, 2018.

In this paper, we regard the aggregation as a routing problem of how to deliver the messages from source nodes to target nodes. In our setting, the source nodes are the outputs of a recurrent or convolutional encoding layer, and the target nodes are one or several fixed-size encoding vectors to represent the meaning of the text sequence.

From this viewpoint, both the pooling and attentive aggregations are a *fixed* routing policy without considering the state of the final encoding vectors. For example, the final encoding vectors could receive some redundancy information from different words. The fixed routing policy cannot avoid this issue. Therefore, we wish for a new way to aggregate information according to the state of the final encoding.

In recent promising work of capsule network (Sabour et al., 2017), a dynamic routing policy is proposed and proven to be more effective than the max-pooling routing. Inspired by their idea, we introduce a text sequence encoding model with dynamic routing mechanism. Specifically, we propose two kinds of dynamic routing policies. One is the standard dynamic routing policy same as the capsule network, in which the source node decides what and how many messages are sent to different target nodes. The other is the reversed dynamic routing policy, in which the target node decides what and how many messages may be received from different source nodes.

Experimental results on five text classification tasks show that the dynamic routing policy outperforms other aggregation methods, such as max pooling, average pooling, and self-attention by a significant margin.

2 Background: general sequence encoding for text classification

In this section, we are going to introduce a general text classification framework. It consists of an Embedding Layer, Encoding Layer, Aggregation Layer and Prediction Layer.

2.1 Embedding Layer

Given a text sequence with words $S = w_1, w_2, \cdots, w_L$. Since the words are symbols that could not be processed directly using prominent neural architectures, so we first map each word into a d dimensional embedding vector,

$$X = [\mathbf{x}_1, \mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_L]. \tag{1}$$

In order to transfer knowledge from a vast unlabeled corpus, the embeddings can be taken from the pre-trained word embedding, such as Glove (Pennington et al., 2014).

2.2 Encoding Layer

However, each word representation in X is still independent with each other. To gain some dependency between adjacent words, we then build a bi-directional LSTM (BiLSTM) layer (Hochreiter and Schmidhuber, 1997) to incorporate forward and backward context information of a sequence. Then we can get phrase-level encoding $\mathbf{h}_t$ of a word by concatenating forward $\mathbf{h}_t^f$ and backward output vector $\mathbf{h}_t^b$ correspond to the target word.

$$\mathbf{h}_t^f = \text{LSTM}(\mathbf{h}_{t-1}^f, \mathbf{x}_t), \tag{2}$$

$$\mathbf{h}_t^b = \text{LSTM}(\mathbf{h}_{t+1}^b, \mathbf{x}_t), \tag{3}$$

$$\mathbf{h}_t = [\mathbf{h}_t^f; \mathbf{h}_t^b]. \tag{4}$$

Thus, the outputs of BiLSTM encoder are a sequence of vectors

$$H = [\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_L]. \tag{5}$$

2.3 Aggregation Layer

Encoding layer only models dependency between adjacent words, but the final prediction of the text requires a fix-length vector. Therefore we need aggregate information from variable length sequence to a single fix-length vector. There are several different ways of aggregation such as max or average pooling, and context-attention.

Max or Average Pooling Max or Average pooling is a simple way of aggregating information, which does not require extra parameters and is computationally efficient (Kim, 2014; Zhao et al., 2015; Lin et al., 2017). In the process of modeling natural language, max or average pooling is performed along the time dimension.

$$\mathbf{e}^{max} = \max([\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_L]), \tag{6}$$

$$\mathbf{e}^{avg} = \frac{1}{L}\sum_{i=1}^{L}\mathbf{h}_i, \tag{7}$$

For example,in Equation 6 the $\max$ operation is performed on each dimension of $\mathbf{h}$ along time dimension. And in Equation 7 the average operation is performed along time dimension.

Max pooling is empirically better at aggregating long sentences than average pooling. We assume it's because that, the actual word that contributes to the classification problem is far less than the number of words that contain in a long sentence. Information from important words is weakened by a large population of "boring" words.

Self-Attention As has been stated previously, average pooling is prone to weaken important words when the sentence is longer. Self-Attention assigns each word a weight to indicate the importance of a word depending on the task on hand. A few words that are crucial to the task will be emphasized while the "boring" words are ignored. The self-attention process is formulated as follows:

$$u_i = \mathbf{q}^T\mathbf{h}_i, \tag{8}$$

$$a_i = \frac{\exp(u_i)}{\sum_k \exp(u_k)}, \tag{9}$$

$$\mathbf{e}^{attn} = \sum_{i=1}^{L} a_i \cdot \mathbf{h}_i \tag{10}$$

First, we need a task-specific trainable query $\mathbf{q} \in \mathbb{R}^d$ to calculate similarity weight between query and each contextually encoded word. Then the corresponding weights are normalized across time dimension using softmax normalization function Eq. 9, after that the aggregated vector is simply a weighted sum of the input sequence in Eq. 10.

2.4 Prediction Layer

Then we feed the encoding $\mathbf{e}$ to the input of a multi-layer perceptron (MLP), followed by a softmax classifier.

$$\mathbf{p}(\cdot|\mathbf{e}) = \mathrm{softmax}(\mathrm{MLP}(\mathbf{e}))$$

where $\mathbf{p}(\cdot|\mathbf{e})$ is the predicted distribution of different classes given the representation vector $\mathbf{m}$.

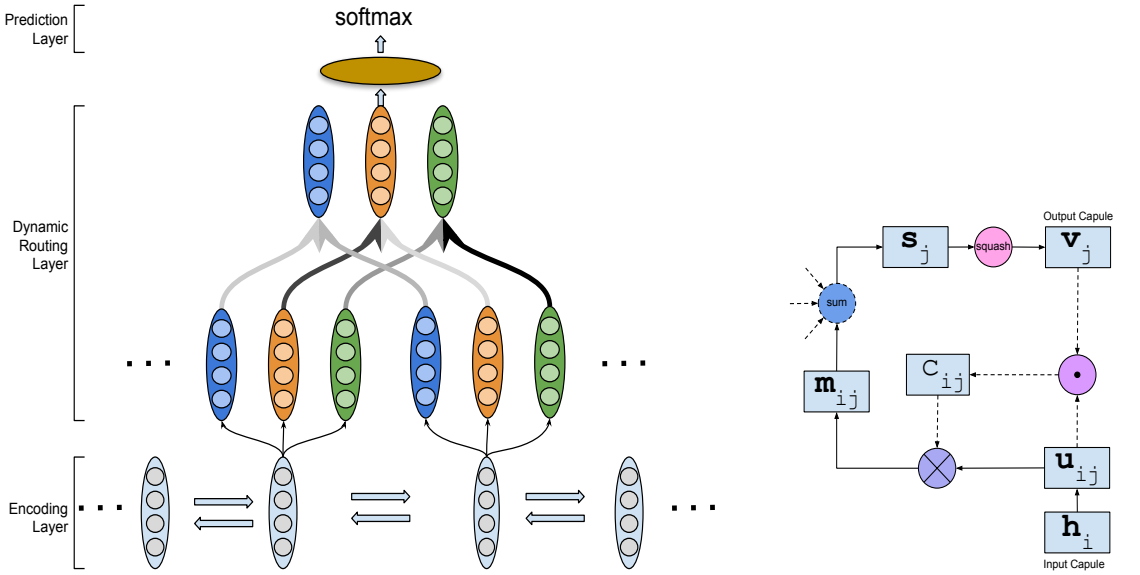
3 Aggregation via Dynamic Routing

In this section, we will formally introduce dynamic routing in detail. The goal of dynamic routing is to encode the meaning of X into M fix-length vectors

$$V = [\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_M]. \tag{11}$$

To transfer information from a variable number of representation H to a fixed number of vectors V, a key problem we need to slove is to properly design a routing policy of information transfer. In other words, *what* and *how much* information is to be transferred from $\mathbf{h}_i$ to $\mathbf{v}_j$.

Although self-attention has been applied in aggregation, the notion of summing up elements in the attention mechanism is still very primitive. Inspired by the capsule networks (Sabour et al., 2017), we propose a dynamic routing aggregation (DR-AGG) mechanism to compute the final encoding of text sequence.



(a) Aggregation via Dynamic Routing

(b) Detialed dynamic routing process

Figure 1: Diagram of dynamic routing for sequence encoding. (a) is the overall dynamic routing diagram, the width of edges between capsules indicate the amount of information transfered, which is refined iteratively. (b) is a detailed iterative process of transferring information from capsule $\mathbf{h}_i$ to capsule $\mathbf{v}_j$, where $\odot$ is a inner product operation and $\otimes$ is a element-wise product.

Following the definition of capsule networks, we call each encoding vector, or a group of neurons, as a capsule. Thus, H denotes the input capsules, and V denotes the output capsules.

A message vector $\mathbf{m}_{i\to j}$ denotes the information to be transferred from $\mathbf{h}_i$ to $\mathbf{v}_j$.

$$\mathbf{m}_{i\to j} = c_{ij} f(\mathbf{h}_i, \theta_j), \tag{12}$$

where c_{ij} indicates proportionally how much information is to be transferred, and $f(\mathbf{h}_i, \theta_j)$ is a one-layer fully-connected network parameterized by θ_j, indicating which aspect of information is to be transferred.

The output capsule $\mathbf{v}_j$ first aggregates all the incoming messages

$$\mathbf{s}_j = \sum_{i=1}^{L} \mathbf{m}_{i\to j}, \tag{13}$$

and then squashes $\mathbf{s}_j$ to confine $|\mathbf{s}_j| \in (0, 1)$ to a probability,

$$\mathbf{v}_j = \frac{\|\mathbf{s}_j\|^2}{1 + \|\mathbf{s}_j\|^2} \frac{\mathbf{s}_j}{\|\mathbf{s}_j\|} \tag{14}$$

Dynamic Routing Process The dynamic routing process is implemented by an iterative process of refining the coupling coefficient c_{ij}, which define proportionally how much information is to be transferred from $\mathbf{h}_i$ to $\mathbf{v}_j$.

The coupling coefficient c_{ij} is computed by

$$c_{ij} = \frac{\exp(b_{ij})}{\sum_k \exp(b_{ik})}, \tag{15}$$

$$b_{ij} \leftarrow b_{ij} + \mathbf{v}_j^{\mathrm{T}} f(\mathbf{h}_i, \theta_j), \tag{16}$$

where b_{ij} is the log probabilities, initialized with 0.

The coefficients c_{ij} is computed using a softmax function, and $\sum_{j=1}^{M} c_{ij} = 1$. Therefore, the total amount of information transferred from capsule h_i is proportionally summed to one.

When an output capsule $\mathbf{v}_j$ receives the incoming messages, its state will be updated and the coefficient c_{ij} is also re-computed for each input capsule. Thus, we iteratively refine the route of information passing, towards an instance dependent and context aware encoding of a sequence. After the text sequence is encoded into M capsules, We map these capsules into vector representation by simply concatenating all capsules,

$$\mathbf{e} = [\mathbf{v}_1; \ldots; \mathbf{v}_M]. \tag{17}$$

Figure 1 gives an illustration of the dynamic routing mechanism. The detailed dynamic routing algorithm is further described in detail in Algorithm 1.

Algorithm 1: Dynamic Routing Algorithm

Data: Input Capsules: $\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_L$, Maximum number of Iterations: T
Result: Output Capsules: $\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_M$
Initialize $b_{ij} \leftarrow 0$;
for $t = 1$ **to** T **do**
  Compute the routing coefficients c_{ij} for all $i \in [1, L], j \in [1, M]$; /* Eq.15 */
  Update all the output capsule $\mathbf{v}_j, j \in [1, M]$; /* Eq. 13 and 14 */
  Update b_{ij} for all $i \in [1, L], j \in [1, M]$; /* Eq. 16 */
end for
return $\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_M$

Reversed Dynamic Routing Process In standard DR-AGG, an input capsule decides what proportion of information can be transferred to an output capsule. We also explore a reversed dynamic routing, in which the output capsule decides what proportion of information should be received from an input capsule. The only difference between reversed dynamic routing and standard dynamic routing is how the softmax function was applied to the log probabilities $[b_{ij}]_{L \times M}$. Instead of normalizing each row of $[b_{ij}]_{L \times M}$ as is done in standard DR-AGG, reverse dynamic routing normalizes each column of $[b_{ij}]_{L \times M}$,

$$c_{ij} = \frac{\exp(b_{ij})}{\sum_k \exp(b_{kj})} \tag{18}$$

Other detail of reversed dynamic routing is the same as the standard dynamic routing.

The reversed DR-AGG works like the multi-hop memory network in iteratively aggregating information (Sukhbaatar et al., 2015; Kumar et al., 2015).

3.1 Analysis

The DR-AGG is somewhat like attention mechanism (Bahdanau et al., 2014; Vaswani et al., 2017).however, there are differences.

In standard DR-AGG, each input capsule (encoding of each word) is employed as query vector to assign a proportion weight to each output capsule, and then sends messages to the output capsules in proportion. Thus, for all input capsules the total amount of messages sent from an input capsule are the same.

In reversed DR-AGG, each output capsule is used as query vector to assign a proportion weight to each input capsule and then receives messages from the input capsules in proportion. Thus, for all output capsules the total amount of message received by an output capsule is same.

The major difference between DR-AGG and self-attention (Lin et al., 2017; Yang et al., 2016) is that the query vector of self-attention is task dependent trainable parameters learned during the training phase, while the query vector of DR-AGG is each input or output capsule which is instance dependent and dynamically updated.

Dataset	Type	Train Size	Dev. Size	Test size	Classes	Averaged Length	Vocabulary Size
Yelp 2013	Document	62522	7773	8671	5	189	29.3k
Yelp 2014	Document	183019	22745	25399	5	197	49.6k
IMDB	Document	67426	8381	9112	10	395	61.1k
SST-1	Sentence	8544	1101	2201	5	18	16.3k
SST-2	Sentence	6920	872	1821	2	19	14.8k

Table 1: Statistics of the five datasets used in this paper

Additionally, the self-attention aggregation collects information in a bottom-up way, without considering the state of the final encoding. It is hard to avoid the problems of information redundancy and information loss. While in the standard DR-AGG, each word can iteratively decide *what* and *how much* information is to be sent to the final encoding.

4 Hierarchical Dynamic Routing for Long Text

The dynamic routing mechanism can aggregate the text sequence with any length, therefore it is able to handle long texts directly, such as the whole paragraphs or documents.

To further enhance the efficiency and scalability of information aggregation, we adopt a hierarchical dynamic routing mechanism to handle the long text. The hierarchical routing strategy can exploit more parallelization and speed up training and inference process. A similar strategy is also used in (Yang et al., 2016).

Concretely, we split a document into sentences, and apply the proposed dynamic routing mechanism on word and sentence levels separately. We first encode each sentence into a fixed-length vector, then convert the sentence encodings into document encoding.

5 Experiment

We test the empirical performance of our proposed model on 5 benchmark datasets for document and sentence level classification and compare our proposed model to other competitor models.

5.1 Datasets

To evaluate the effectiveness of our proposed aggregation method, we have conducted experiments on 5 datasets, the statistics of experimented datasets are shown in Table 1. As shown in the table, Yelp-2013, Yelp-2014, and IMDB are document level datasets, while SST-1 and SST-2 are sentence level datasets. Note that we use the same document level datasets provided in (Tang et al., 2015).

Yelp reviews Yelp-2013 and Yelp-2014 are reviews from Yelp, each example consists of several review sentences and a rating score range from 1 to 5 (higher is better).

IMDB is a movie review dataset extracted from IMDB website. It is a multi-sentence dataset that for each example there are several review sentences. A rating score range from 1 to 10 is also associated with each example.

SST-1 Stanford Sentiment Treebank is a movie review dataset which has been parsed and further splited to train/dev/test set (Socher et al., 2013). For each example in the dataset, there exists only one sentence and a label associated with it. And the labels can be one of {negative, somewhat negative, neutral, somewhat positive, positive}.

SST-2 This dataset is a binary-class version of SST-1, with neutral reviews removed and the remaining reviews categorized to either negative or positive.

	Yelp-2013	Yelp-2014	IMDB	SST-1	SST-2
Embedding size	300	300	300	300	300
LSTM hidden unit	200	200	200	200	200
Capsule dimension	200	200	200	200	200
Capsule number	5	5	5	5	5
Iteration number	3	3	3	3	3
Regularization rate	1e-5	1e-5	1e-6	1e-6	1e-5
Initial learning rate	0.0001	0.0002	0.0001	0.0001	0.0003
learning rate decay	0.9	0.9	0.95	0.95	0.95
learning rate decay steps	1000	1000	1000	500	500
Initial Batch size	32	32	32	64	64
Batch size low bound	32	32	32	16	16
Dropout rate	0.2	0.2	0.2	0.2	0.5

Table 2: Detailed hyper-parameter settings

5.2 Training

Given a training set $\{x^{(i)}, t^{(i)}\}_{i=1}^{N}$, where $x^{(i)}$ is an example of the training set and $t^{(i)}$ is the corresponding label, the goal is to minimize the cross-entropy loss $\mathcal{J}(\theta)$:

$$\mathcal{J}(\theta) = -\frac{1}{N}\sum_{i} \log p(t^{(i)}|x^{(i)};\theta) + \lambda||\theta||_2^2, \tag{19}$$

where θ represents all of the parameters.

The Adam optimizer is applied to update the parameters (Kingma and Ba, 2014). Table 2 displays the detailed hyper-parameter settings. To prevent overfitting, the L2 regularization term is introduced to our loss function. We also adopt early stop strategy, The training process will be stopped after seven epochs of no improvement on development set is observed.To further avoid overfitting, dropout is applied before the biLSTM encoder and hidden layer of classifier **MLP**.

The mini-batch size is set to 32 for document level dataset, 64 for sentence level dataset, examples are sampled from a sliding bucket to speed up the training process. Data is sorted by the length of sentence, and we first sample a window on the sorted data, we call the window "sliding bucket" and then sample a batch of examples from the sliding bucket, we double the window size after an epoch of no improvement on development set, through such a strategy, we are able to considerably speed up training while retaining randomness. Also, batch size is halved after an epoch of no improvement on development set until it reaches the low bound batch size. We also utilize a data preparation queue to parallelize data preparation and training.

Word embedding is initialized from pre-trained Glove (Pennington et al., 2014). We randomly initialize word vectors for words that doesn't appear in Glove. Network weights are initialized with Xavier Normalization (Glorot and Bengio, 2010). A more detailed hyper-parameter setting can be referred to hyper-parameter Table 2. And hyper-parameters are determined using grid search strategy.

5.3 Experimental Results

We evaluate several aggregation methods on five text classification datasets, in which Yelp-2013, Yelp-2014 and IMDB are document-level datasets, and SST-1 and SST-2 are sentence-level datasets. Since max pooling, average pooling and self-attention are most related to our proposed DR-AGG, we mainly compare DR-AGG to these three methods.

Table 3 gives the results for different methods, the last two rows are our model (standard DR-AGG and reversed DR-AGG), the table shows that our proposed dynamic routing performed the best on all datasets. In document-level text classification, specifically Yelp 2013 Yelp 2014 and IMDB, DR-AGG outperforms previous models best results by 2.5%, 3.0% and 1.6% respectively. In sentence-level text

	Yelp-2013	Yelp-2014	IMDB	SST-1	SST-2
RNTN+Recurrent (Socher et al., 2013)	57.4	58.2	40.0	-	-
CNN-non-static (Kim, 2014)	-	-	-	48.0	87.2
Paragraph-Vec (Le and Mikolov, 2014)	-	-	-	48.7	**87.8**
MT-LSTM (F2S) (Liu et al., 2015)	-	-	-	49.1	87.2
UPNN(np UP) (Tang et al., 2015)	57.7	58.5	40.5	-	-
UPNN(full) (Tang et al., 2015)	59.6	60.8	43.5	-	-
Cached LSTM (Xu et al., 2016)	59.4	59.2	42.1	-	-
Max pooling	61.1	61.2	41.1	48.0	87.0
Average pooling	60.7	60.6	39.1	46.2	85.2
Self-attention	61.0	61.5	43.3	48.2	86.4
Standard DR-AGG	**62.1**	**63.0**	**45.1**	**50.5**	87.6
Reverse DR-AGG	61.6	62.5	44.5	49.3	87.2

Table 3: Experimental result comparison on five datasets. For the document-level datasets, hierarchical aggregation is used for both self-attention and DR-AGGs.

(1) so relentlessly wholesome it made me want to swipe something .
(2) so relentlessly wholesome it made me want to swipe something .
(3) so relentlessly wholesome it made me want to swipe something .

Table 4: A visualization to show the perspective of a sentence from 3 different upper level capsule. A deeper color indicates more information of the associated word is routed to the corresponding capsule.

classification, such as SST-1 SST-2, our model also achieves better results. Compared to max pooling, average pooling and self-attention, which are closely related to our model, DR-AGGs significantly improves the performance. For example the standard DR-AGG outperforms the max pooling approach by 1%, 1.8%, 4%,2.5% and 0.4% on Yelp 2013,Yelp 2014, IMDB, SST-1 and SST-2. It empirically shows that our proposed dynamic routing policy is the most effective method on aggregating information.

It is worth to note the reversed DR-AGG is inferior to the standard DR-AGG by a small margin, although it has also achieved better results than the other aggregation methods and SOTA approaches. As discussion before, the reversed DR-AGG have much resemblance with the attention using output capsule as query vector. Not all of the input capsules would be selected by the reversed DR-AGG, while in the standard DR-AGG, the information of all the input capsules need be sent to the output capsules.

Effects of Iterative Routing We also study how the iteration number affect the performance of aggregation on the SST-2 dataset. Figure 2 shows the comparison of 1 - 5 iterations in the standard DR-AGG. The capsule number is set to 1, 2, 3 and 4 for each comparison respectively. We found that the performances on several different capsule number setting reach the best when iteration is set to 3. The results indicate the dynamic routing is contributing to improve the performance.

Visualization Additionally, we visualize how much information each input capsule sends to the output capsules. As shown in Table 4, the visualization experiment was conducted with the setting on three output capsules. The i-th column represents the i-th input capsule, while the j-th row is the j-th output capsule. The color density of each word denotes the proportion c_{ij} in equation 15. A deeper color indicates more information of the concerned word is routed to the output capsule.

Intuitively, the different part of the sentence is routed to three different capsules. In another word, each capsule has a different perspective or focus of the sequence. Therefore, DR-AGG can avoid the problem of information redundancy and information missing.

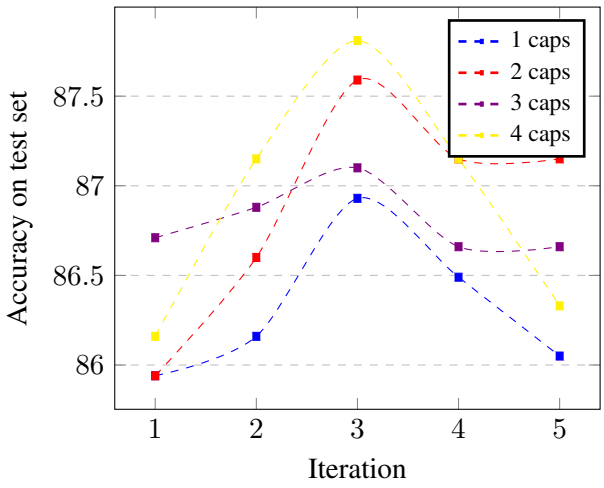


Figure 2: Relationship between test accuracy and routing iteration, where the vertical axis denotes test accuracy and the horizontal axis denotes routing iteration. When the iteration is set to 3 performance peaks on several different capsule number setting

6 Related Work

Currently, much attention has been paid to how developing a sophisticated encoding models to capture the long and short term dependency information in a sequence. Specific to text classification task, most of the models cannot deal with the texts of several sentences (paragraphs, documents), such as MV-RNN (Socher et al., 2012), RNTN (Socher et al., 2013), CNN (Kim, 2014), AdaSent (Zhao et al., 2015), and so on. The simple neural bag-of-words model can deal with long texts, but it loses the word order information. PV (Le and Mikolov, 2014) works in an unsupervised way, and the learned vector cannot be fine-tuned on the specific task. There are also many works (Liu et al., 2015; Xu et al., 2016; Cheng et al., 2016) to improve LSTM's ability to carrying information for a long distance.

A line of orthogonal researches (Lin et al., 2017; Yang et al., 2016; Shen et al., 2018a; Shen et al., 2018b) is to introduce attention mechanism (Vaswani et al., 2017) to weighted average the outputs of CNN/RNN layer. The attention mechanism can effectively reduce the burden of CNN/RNN. The CNN/RNN encoding layer is only expected to extract local context information for each word, while the global semantics of text sequence can be aggregated from the local encoding vectors.

The attention based aggregation collects information in a bottom-up way, without considering the state of the final encoding. It is hard to avoid the problems of information redundancy or information lost. An improved idea is to use multi-hop attention, like memory network (Sukhbaatar et al., 2015; Kumar et al., 2015), to iterative aggregate information. This idea is equivalent to our proposed reversed dynamic routing mechanism.

Different from the attention based aggregation methods, aggregation via dynamic routing is iteratively deciding that *what* and *how much* information need be transfer to the final encoding of each word.

7 Conclusion

In this paper, we focus on how to obtain a fixed-size encoding of text sequence by aggregating the encodings of each word. Although we use LSTM hidden states as word encoding in this paper, the other word encodings, such as convolved n-gram, could be alternatively used. We introduced a fixed-size encoding of text sequence with dynamic routing mechanism. Experimental results of five text classification tasks show that the model outperforms other encoding models by a significant margin.

In the future, we would like to investigate more sophisticated routing policy for better encoding the text sequence. Besides, dynamic routing should also be useful to improve the encoder in the sequence-to-sequence tasks (Sutskever et al., 2014).

Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments. The research work is supported by the National Key Research and Development Program of China (No. 2017YFB1002104), Shanghai Municipal Science and Technology Commission (No. 17JC1404100 and 16JC1420401), and National Natural Science Foundation of China (No. 61672162 and 61751201).

References

D. Bahdanau, K. Cho, and Y. Bengio. 2014. Neural machine translation by jointly learning to align and translate. *ArXiv e-prints*, September.

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 551–561.

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. *The Journal of Machine Learning Research*, 12:2493–2537.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *International conference on artificial intelligence and statistics*, pages 249–256.

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

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

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

Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. 2015. Ask me anything: Dynamic memory networks for natural language processing. *arXiv preprint arXiv:1506.07285*.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of ICML*.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*.

Pengfei Liu, Xipeng Qiu, Xinchi Chen, Shiyu Wu, and Xuanjing Huang. 2015. Multi-timescale long short-term memory neural network for modelling sentences and documents. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 2326–2335.

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.

Sara Sabour, Nicholas Frosst, and Geoffrey E. Hinton. 2017. Dynamic routing between capsules. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 3859–3869.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2018a. DISAN: Directional self-attention network for RNN/CNN-free language understanding. In *AAAI Conference on Artificial Intelligence*.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, and Chengqi Zhang. 2018b. Bi-directional block self-attention for fast and memory-efficient sequence modeling.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of EMNLP*, pages 1201–1211.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of EMNLP*.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems*, pages 2431–2439.

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Learning semantic representations of users and products for document level sentiment classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1014–1023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 6000–6010.

Jiacheng Xu, Danlu Chen, Xipeng Qiu, and Xuanjing Huang. 2016. Cached long short-term memory neural networks for document-level sentiment classification. *CoRR*, abs/1610.04989.

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.

Han Zhao, Zhengdong Lu, and Pascal Poupart. 2015. Self-adaptive hierarchical sentence model. *arXiv preprint arXiv:1504.05070*.