LONG TEXT ANALYSIS USING SLICED RECURRENT NEURAL NETWORKS WITH BREAKING POINT INFORMATION ENRICHMENT

Bo Li1,2,*, Zehua Cheng1,*, Zhenghua Xu1,†, Wei Ye2,†, Thomas Lukasiewicz3, Shikun Zhang2

1State Key Laboratory of Reliability and Intelligence of Electrical Equipment, Hebei University of Technology, Tianjin, China
2National Engineering Research Center for Software Engineering, Peking University, Beijing, China
3Department of Computer Science, University of Oxford, Oxford, United Kingdom
zhenghua.xu@hebut.edu.cn, wye@pku.edu.cn

ABSTRACT

Sliced recurrent neural networks (SRNNs) are the state-of-the-art efficient solution for long text analysis tasks; however, their slicing operations inevitably result in long-term dependency loss in lower-level networks and thus limit their accuracy. Therefore, we propose a breaking point information enrichment mechanism to strengthen dependencies between sliced subsequences without hindering parallelization. Then, the resulting BPIE-SRNN model is further extended to a bidirectional model, BPIE-BiSRNN, to utilize the dependency information in not only the previous but also the following contexts. Experiments on four large public real-world datasets demonstrate that the BPIE-SRNN and BPIE-BiSRNN models always achieve a much better accuracy than SRNNs and BiSRNNs, while maintaining a superior training efficiency.

Index Terms— Long Text Analysis, RNN, SRNN

1. INTRODUCTION

Recurrent neural networks (RNNs) have been witnessed to achieve a superior performance in many natural language processing (NLP) tasks, such as machine translation [1], question answering [2, 3], and natural language inference [4]. The success of applying RNNs in NLP tasks mainly comes from their capability to pass the information in one step of the networks to the next, making it possible to process each word based on the textual information of previous words. However, RNNs suffer from the vanishing (and exploding) gradient problem, which makes it difficult to discover long-distance dependencies between words [5]. Long short-term memory networks (LSTMs) [6, 7] and gated recurrent units (GRUs) [8] are proposed to remedy this problem using a gate mechanism.

However, traditional RNNs, LSTMs, and GRUs all suffer from a training efficiency problem in long text analysis tasks: with increasing text length, the training time of these models also increases dramatically, which greatly limits their application in real-world scenarios. One existing solution is to skip some information, simulating a human’s daily reading habits, such as fixation and saccades [9, 10]; however, this will result in the loss of textual information. Another solution is to introduce convolutional components into the recurrent structure [11, 12, 13]; but the convolutional components make it difficult to fully utilize the order information of sequential data. Thus, their performances are usually unsatisfactory.

Sliced recurrent neural networks (SRNNs) [14] are the state-of-the-art efficient solution for long text analysis tasks, where input sequences are sliced into multiple subsequences, and GRUs process these subsequences in a parallelized way. Although SRNNs can still discover dependencies between subsequences in higher levels of their networks, the slicing operations still inevitably weaken the models’ capability in capturing the long-term dependencies in lower levels.

Consequently, if more dependencies between subsequences can be maintained in lower levels without hindering parallelization, it will not only make lower-level networks capable to get more dependency information, but also help higher-level networks to get more accurate long-term dependencies, which will consequently enhance the models’ performance. One natural choice is to use the last few words of the previous subsequence to calculate a more accurate starting state for a subsequence, instead of using a fixed or random starting state. Furthermore, adding previous words to a given subsequence will increase the probability of containing sentence separators in this subsequence; since separators contain dependency information between sentences, they will further improve the representation modeling of given subsequences.

Motivated by this, we first propose a breaking point information enrichment (BPIE) mechanism, which strengthens the semantic dependencies between sliced subsequences in SRNNs by adding to each subsequence the last few words of the previous subsequence. The resulting model is called BPIE-SRNN. In addition, since people can better understand a given sentence by considering not only the information in previous sentences but also that in following sentences, we further propose to extend the BPIE-SRNN model to a bidirectional model, called BPIE-BiSRNN, to use the dependency information in not only the previous but also the following contexts. In BPIE-BiSRNNs, subsequences are processed
both forward and backward in GRUs, and the last (resp., first) few words of the previous (resp., next) subsequence are added to given subsequences in forward (resp., backward) processes for information enrichment. Extensive experimental studies on four large public real-world datasets demonstrate the superior performance of BPIE-SRNNs and BPIE-BiSRNNs on long text analysis tasks: BPIE-SRNNs and BPIE-BiSRNNs always achieve a much better accuracy than SRNNs and BiSRNNs, while maintaining a superior training efficiency.

2. BPIE-BiSRNN

In this section, we introduce the details of the BPIE-BiSRNN model: BPIE-SRNNs are a simplified version of BPIE-BiSRNNs, where all backward operations are removed. The overall working of BPIE-BiSRNNs is shown as pseudocode in Algorithm 1 and also depicted in Fig. 1.

In BPIE-BiSRNNs, given a long text sequence \( S \) (after padding) consisting of \( L \) words (i.e., \( S = [x_0, x_1, \ldots, x_{L-1}] \)), we first slice \( S \) into \( n \) subsequences \( S = [s_0, s_1, \ldots, s_n-1] \) with an equal length \( z = L/n \) (line 1 in Algorithm 1).

Then, for each subsequence \( s_k \), we add the last \( m \) words of the previous subsequence \( s_{k-1} \) to the front of \( s_k \), resulting in an information-enriched subsequence for the forward processes, denoted \( s_{k,\text{forward}} \); similarly, \( s_{k,\text{backward}} \) is obtained for the backward processes by adding the first \( m \) words of the next subsequence \( s_{k+1} \) (lines 4–5). Formally,

\[
\begin{align*}
    s_{k,\text{forward}} &= \text{Concat}([s_{k-1}[m:]; s_k]), \\
    s_{k,\text{backward}} &= \text{Concat}([s_k, s_{k+1}[1:m]]),
\end{align*}
\]

(1)

(2)

where zero-padded subsequences \( s_{-1} \) and \( s_n \) are created for the boundary cases, when \( k = 0 \) and \( k = n-1 \) (line 2), and \( m \) is a suitable hyperparameter with value less than \( z \).

2. BPIE-BiSRNN

In this section, we introduce the details of the BPIE-BiSRNN model: BPIE-SRNNs are a simplified version of BPIE-BiSRNNs, where all backward operations are removed. The overall working of BPIE-BiSRNNs is shown as pseudocode in Algorithm 1 and also depicted in Fig. 1.

In BPIE-BiSRNNs, given a long text sequence \( S \) (after padding) consisting of \( L \) words (i.e., \( S = [x_0, x_1, \ldots, x_{L-1}] \)), we first slice \( S \) into \( n \) subsequences \( S = [s_0, s_1, \ldots, s_n-1] \) with an equal length \( z = L/n \) (line 1 in Algorithm 1).

Then, for each subsequence \( s_k \), we add the last \( m \) words of the previous subsequence \( s_{k-1} \) to the front of \( s_k \), resulting in an information-enriched subsequence for the forward processes, denoted \( s_{k,\text{forward}} \); similarly, \( s_{k,\text{backward}} \) is obtained for the backward processes by adding the first \( m \) words of the next subsequence \( s_{k+1} \) (lines 4–5). Formally,

\[
\begin{align*}
    s_{k,\text{forward}} &= \text{Concat}([s_{k-1}[m:]; s_k]), \\
    s_{k,\text{backward}} &= \text{Concat}([s_k, s_{k+1}[1:m]]),
\end{align*}
\]

(1)

(2)

where zero-padded subsequences \( s_{-1} \) and \( s_n \) are created for the boundary cases, when \( k = 0 \) and \( k = n-1 \) (line 2), and \( m \) is a suitable hyperparameter with value less than \( z \).

Algorithm 1: BPIE-BiSRNN(S)

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( s_0, s_1, \ldots, s_{n-1} \leftarrow \text{slice}(S) )</td>
</tr>
<tr>
<td>2</td>
<td>( s_{-1} \leftarrow s_n \leftarrow [0 \ast m] )</td>
</tr>
<tr>
<td>3</td>
<td>for ( k ) in range 0 to ( n-1 ) do</td>
</tr>
<tr>
<td>4</td>
<td>( s_{k,\text{forward}} \leftarrow \text{Concat}([s_{k-1}[m:]; s_k]) )</td>
</tr>
<tr>
<td>5</td>
<td>( s_{k,\text{backward}} \leftarrow \text{Concat}([s_k, s_{k+1}[1:m]]) )</td>
</tr>
<tr>
<td>6</td>
<td>( O_{k,\text{forward}} \leftarrow \text{GRU}(s_{k,\text{forward}}) )</td>
</tr>
<tr>
<td>7</td>
<td>( O_{k,\text{backward}} \leftarrow \text{GRU}(s_{k,\text{backward}}) )</td>
</tr>
<tr>
<td>8</td>
<td>( O'<em>{k,\text{forward}} \leftarrow O</em>{k,\text{forward}}[m:] )</td>
</tr>
<tr>
<td>9</td>
<td>( O'<em>{k,\text{backward}} \leftarrow O</em>{k,\text{backward}}[-m] )</td>
</tr>
<tr>
<td>10</td>
<td>( r_{k,\text{forward}} \leftarrow \text{Concat}([\text{max}(O'<em>{k,\text{forward}}), \text{avg}(O'</em>{k,\text{forward}}), \text{last}(O'_{k,\text{forward}})]) )</td>
</tr>
<tr>
<td>11</td>
<td>( r_{k,\text{backward}} \leftarrow \text{Concat}([\text{max}(O'<em>{k,\text{backward}}), \text{avg}(O'</em>{k,\text{backward}}), \text{last}(O'_{k,\text{backward}})]) )</td>
</tr>
<tr>
<td>12</td>
<td>( r_k \leftarrow \text{Concat}([r_{k,\text{forward}}, r_{k,\text{backward}}]) )</td>
</tr>
<tr>
<td>13</td>
<td>( R \leftarrow [r_0, r_1, \ldots, r_{n-1}] )</td>
</tr>
<tr>
<td>14</td>
<td>( R' \leftarrow \text{BiGRU}(R) )</td>
</tr>
<tr>
<td>15</td>
<td>( r' \leftarrow \text{Concat}([\text{max}(R'), \text{avg}(R'), \text{last}(R')]) )</td>
</tr>
<tr>
<td>16</td>
<td>( p \leftarrow \text{Fully_Connected}(r') )</td>
</tr>
<tr>
<td>17</td>
<td>return Softmax(p)</td>
</tr>
</tbody>
</table>

\( s_{k,\text{forward}} \) (resp., \( s_{k,\text{backward}} \)) is then used as the input of a forward (resp., backward) GRU to obtain a forward (resp., backward) output tensor \( O_{k,\text{forward}} \) (resp., \( O_{k,\text{backward}} \)) (lines 6–7). Formally,

\[
\begin{align*}
    O_{k,\text{forward}} &= \text{GRU}(s_{k,\text{forward}}), \\
    O_{k,\text{backward}} &= \text{GRU}(s_{k,\text{backward}}),
\end{align*}
\]

(3)

(4)
where the size of $O_{k, \text{forward}}$ and $O_{k, \text{backward}}$ is $(z + m) \times h$, with $h$ being the number of hidden states in the GRU.

Furthermore, to avoid duplicate computations in the following steps, we drop the outputs originating from the $m$ enriched words, resulting in a target output tensor $O'_{k, \text{forward}}$ and $O'_{k, \text{backward}}$ with a size $z \times h$ (lines 8–9). This operation is called get target tensor and formally written as

$$O'_{k, \text{forward}} = O_{k, \text{forward}}[m:];$$

$$O'_{k, \text{backward}} = O_{k, \text{backward}}[: -m].$$

Then, for each pair of $O_{k, \text{forward}}$ and $O'_{k, \text{backward}}$, we conduct maximum, average, and last tensor operations on their first dimension and concatenate the resulting $1 \times h$ vectors to obtain a 1st-layer feature representation of $s_k$, denoted $r_k$, whose size is $1 \times (6 \times h)$ (lines 10–12). Furthermore, we merge the representations of all subsequences to a full 1st-layer feature representation of the input long text sequence $S$, denoted $R$, whose size is $n \times (6 \times h)$ (line 13).

$R$ is further passed to a BiGRU to get a full 2nd-layer output of $S$, denoted $R'$, and then maximum, average, and last tensor operations are conducted again on $R'$ to get the full 2nd-layer feature representation $r'$ (lines 14–15). Finally, we send $r'$ into a fully-connected layer and apply softmax operation to get results (lines 16–17). Please note that BPIE-SRNNs are set to three layers in this work for clarity; however, by stacking multiple GRUs between the first two layers, BPIE-BiSRNNs can be easily extended to deeper networks.

3. EXPERIMENTS

To show the strength of our proposed BPIE-SRNN model in handling long text analysis tasks and offering a superior performance in both accuracy and efficiency, standard RNNs with GRUs [8] as the recurrent unit and the state-of-the-art solution, SRNNs [14], are selected as the baselines. In addition, the performance of the bidirectional models of GRUs, SRNNs, and BPIE-SRNNs (denoted BiGRUs, BiSRNNs, and BPIE-BiSRNNs, respectively) is further evaluated to show the advantage of applying the bidirectional structure to further enhance the classification accuracies of models. For a fair comparison, the experiments are performed on the same public large online real-world datasets used in [14]. The statistic information of the datasets is shown in Table 1, and the detailed descriptions are as follows:

Rating sentiment datasets: Three subsets of Yelp reviews released by the Yelp Dataset Challenge1 in the years 2013, 2014, and 2015 are extracted by Yu an Liu [14] to construct three Yelp rating sentiment datasets, denoted Yelp 2013, Yelp 2014, and Yelp 2015, respectively. These datasets contain 468, 608 documents, 670, 440 documents, and 897, 835 documents, respectively, and each document is associated with a rating sentiment label, ranging from 1 to 5 stars (the more, the better). Then, 10% of the documents in each dataset are randomly selected using the same random seed as in [14] to form a test set, and the rest is used as the training set.

Polarity sentiment dataset: Yelp reviews are also used by Zhang et al. [15] to construct a polarity sentiment dataset, denoted Yelp_P, where each document is associated with a polarity sentiment label (either positive or negative). This dataset is also used in [14], and the same division of training and test sets as in [15] is adopted.

3.1. Training Details and Code Release

All models are implemented using Keras [16], and trained on a GPU server with an NVIDIA GTX 1080Ti GPU. The training details here are all similar to that in [14], except for the length of subsequences. Specifically, in [14], the input long text sequence is sliced into several subsequences with 32 words in each of them; however, in BPIE-SRNNs, we add to each subsequence the last 5 words of the previous subsequence to enrich the semantic dependency information between sliced subsequences, making the length of each subsequence become 37; thus, to keep a fair comparison in training efficiency, the length of subsequences in SRNNs is also set to 37. Similarly, the length of each subsequence in BiSRNNs is also the same as that in BPIE-BiSRNNs.

To facilitate future research and clarify the details, easy-to-run codes have been released online.² Some important training details are as follows: (i) the pre-trained Glove embeddings in [17] are used to initialize the word embeddings; (ii) the batch size is 512; (iii) the recurrent unit is set as GRU, and all its hidden states and fully-connected layers have 64 dimensions; (iv) all models are learned using the Adam optimizer [18] with learning rate $\alpha = 0.001$, the first momentum $\beta_1 = 0.9$, and the second momentum $\beta_2 = 0.999$; and (v) the dropout rate is 0.2.

3.2. Main Results

Table 2 depicts the performance of the proposed models, BPIE-SRNNs and BPIE-BiSRNNs, and their corresponding baselines, GRUs, SRNNs, BiGRUs, and BiSRNNs, on four sentiment datasets, Yelp 2013, Yelp 2014, Yelp 2015, and Yelp_P, in terms of classification accuracy and training efficiency. Despite some slight differences, due to different subsequence lengths (37 vs. 32) and different kinds of GPUs used (1080Ti vs. 1080), the performance results of GRUs and SRNNs in Table 2 are highly consistent with the results reported in [14], showing the correctness of our experiments.

In Table 2, BPIE-SRNNs (resp., BPIE-BiSRNNs) significantly outperform their corresponding baselines, GRUs and SRNNs (resp., BiGRUs and BiSRNNs), on all four datasets in terms of classification accuracy. This demonstrates that BPIE-SRNNs and BPIE-BiSRNNs successfully overcome the long-term dependency loss problem in the lower levels of SRNNs and BiSRNNs by applying breaking point information enrichment to strengthen the semantic dependencies between sliced subsequences. Furthermore, due to breaking point information enrichment, the training time costs of BPIE-SRNNs and BPIE-BiSRNNs on all datasets are inevitably higher than those of SRNNs and BiSRNNs; however, the increases are always less than 20%, which is minor and acceptable in practice, and the time costs are still much less.

1https://www.yelp.com/dataset/challenge

2Code release link: https://github.com/limberc/BPIE-BiSRNN
than those of GRUs and BiGRUs. These results show that BPIE-SRNNs and BPIE-BiSRNNs can achieve a much better accuracy than SRNNs and BiSRNNs in long text analysis tasks, while maintaining a superior training efficiency.

Table 2 also exhibits that the bidirectional models, BiGRUs, BiSRNNs, and BPIE-BiSRNNs, always outperform the single directional models, GRUs, SRNNs, and BiSRNNs, respectively, in terms of classification accuracy on all datasets, and BPIE-BiSRNNs achieve the best accuracy among all six models. This finding proves that besides the dependency information in the previous context, the information in the following context is also beneficial for long text analysis tasks; thus, when the accuracy instead of the time cost is the first priority, it is reasonable to extend BPIE-SRNNs to BPIE-BiSRNNs, to further enhance the accuracy.

Notice also that the accuracy improvements of BPIE-BiSRNNs relative to BPIE-SRNNs are higher than those of BiSRNNs relative to SRNNs on all datasets. It proves that, by using breaking point information enrichment, BPIE-BiSRNNs successfully overcome the long-term dependency loss problem in lower levels of BiSRNNs, making it capable to obtain more backward dependency information in lower levels and to get more accurate long-term backward dependencies in higher levels. This thus results in higher accuracy improvements in BPIE-BiSRNNs compared to BiSRNNs.

3.3. Effect of Increasing Text Length

We also investigate the effect of increasing the text length on the accuracy improvement of BPIE-BiSRNNs (resp., BPIE-SRNNs) relative to BiSRNNs (resp., SRNNs). The experiments are on five subsets of text sequences selected from Yelp 2013, where the sequence lengths are 100-200, 300-400, 500-600, 700-800, and 900-1000 words, respectively.

As shown in Table 3, the accuracy improvement of BPIE-BiSRNNs relative to BiSRNNs greatly rises up with the increase of the text length: when the text length is 100-200 words, the accuracy improvement is only 1.40%; it then rises up to 3.66% and 3.75%, when the text length increases to 300–400 and 500–600 words, respectively; the accuracy improvement finally reaches up to 8.40% and 8.06%, when the text lengths are 700–800 and 900–1000 words, respectively. In addition, the comparative results of BPIE-SRNNs and SRNNs are similar and exhibit the same finding; so they are omitted due to space limit. This observation further demonstrates the capability of BPIE on discovering long-term semantic dependencies, and proves that BPIE-BiSRNNs and BPIE-SRNNs are better choices than BiSRNNs and SRNNs in handling long text analysis tasks.

4. CONCLUSION

We have proposed a breaking point information enrichment mechanism to strengthen dependencies between sliced subsequences without hindering parallelization. The resulting BPIE-SRNN model has also been extended to a bidirectional model, called BPIE-BiSRNN. In experiments, BPIE-SRNNs and BPIE-BiSRNNs always achieve a much better accuracy than SRNNs and BiSRNNs, while having a superior training efficiency. In future works, we will apply BPIE-SRNNs and BPIE-BiSRNNs to improve more NLP applications. Moreover, we will also investigate more techniques, e.g., distance masking [19], to further enhance the models’ performances.
5. REFERENCES


