Pointing to Select: A Fast Pointer-LSTM for Long Text Classification

Jinhua Du, Yan Huang and Karo Moilanen
Investments AI, AIG (American International Group, Inc.)
{jinhua.du, yan.huang, karo.moilanen}@aig.com

Abstract

Recurrent neural networks (RNNs) suffer from well-known limitations and complications which include slow inference and vanishing gradients when processing long sequences in text classification. Recent studies have attempted to accelerate RNNs via various ad hoc mechanisms to skip irrelevant words in the input. However, word skipping approaches proposed to date effectively stop at each or a given time step to decide whether or not a given input word should be skipped, breaking the coherence of input processing in RNNs. Furthermore, current methods cannot change skip rates during inference and are consequently unable to support different skip rates in demanding real-world conditions. To overcome these limitations, we propose Pointer-LSTM, a novel LSTM framework which relies on a pointer network to select important words for target prediction. The model maintains a coherent input process for the LSTM modules and makes it possible to change the skip rate during inference. Our evaluation on four public data sets demonstrates that Pointer-LSTM (a) is 1.1x$\sim$3.5x faster than the standard LSTM architecture; (b) is more accurate than Leap-LSTM (the state-of-the-art LSTM skipping model) at high skip rates; and (c) reaches robust accuracy levels even when the skip rate is changed during inference.

1 Introduction

Recurrent neural networks (RNNs), such as Long Short-term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) and Gated Recurrent Unit (GRU) (Chung et al., 2014), are popular architectural choices for many sequential text processing tasks. Typically, RNNs read in either all or a fixed-length slice of tokens sequentially, and output a distributed representation for each token. While this paradigm is essential for some applications such as machine translation (Bahdanau et al., 2015), it may result in suboptimal or practically prohibitive processing times for long input sequences which are not uncommon in document classification (Liu et al., 2015) and complex question answering (Tan et al., 2015). Building on wider, established observations that (i) not all input tokens are equally important in many tasks (Seo et al., 2017); (ii) text can be comprehended even through partial reading due to linguistic redundancy effects (Yu et al., 2017); and (iii) attention, fixations, and saccades govern human reading in general (Hahn and Keller, 2016), recent RNN optimisation efforts have focused on mechanisms to skip irrelevant, unimportant, or redundant words in the input (Yu et al., 2017; Campos et al., 2017; Seo et al., 2017; Liu et al., 2018; Hansen et al., 2019; Huang et al., 2019). We refer to such models as *selective RNNs*.

In general, existing selective RNNs use a decision-making component to predict whether or not a word should be skipped (skimmed). In a typical selective RNN workflow, the RNN's input steps are interrupted by the decision network to make a decision to copy the previous hidden state or calculate a new hidden state for the next time step, for example. Interruptions of this kind make RNNs unnecessarily complicated from the point of view of implementation and optimisation. Furthermore, existing selective RNNs make changes to the skip rate impossible during inference. However, real-world applications frequently call for flexible skip rates in order to optimise the trade-off between inference time (i.e. latency) and accuracy, amongst others. Dynamic skip rates during inference can reduce the cost of iterative model retraining and make it easier to manage many differently parameterised models.

This paper proposes Pointer-LSTM, a novel selective RNN framework which maintains a coherent RNN input procedure (i.e. no interruptions at each time step) and supports flexible inference skip rates. We draw inspiration from the pointer network (Vinyals et al., 2015; Gulcehre et al., 2016; See et al., 2017) which was designed to select and copy tokens from the input to the output. Pointer-LSTM similarly learns to select important words from the input. However, instead of merely copying the selected words to the output, Pointer-LSTM copies a proportion of the input words to a large LSTM network for target prediction. More specifically, Pointer-LSTM uses a small bidirectional LSTM (BiLSTM) network to produce a self-attention distribution over the entire input sequence; the attention distribution is then sampled to select top-K important words for target prediction by a large BiLSTM. The value of K in top-K is controlled by a pre-set skip rate r following Huang et al. (2019).

The main contribution of this paper is as follows:

- We propose a Pointer-LSTM framework which consists of two jointly trained BiLSTMs: a small BiLSTM serves as the pointer network to select important words as the input to a large BiLSTM for text classification. Unlike existing selective RNNs, Pointer-LSTM maintains a coherent RNN input procedure which is easy to implement with standard neural network libraries.

- Pointer-LSTM offers greater control over the skip rate than existing selective RNNs: the skip rate can be changed during inference without retraining the model while the performance of the model remains robust despite variable skip rates.

- Experiments on four public data sets show that Pointer-LSTM speeds up the inference by 1.1x$\sim$$3.5$x compared to the standard LSTM architecture, and achieves better performance than Leap-LSTM, the state-of-the-art LSTM skipping model (Huang et al., 2019), at the skip rate of 0.9.

2 Related Work

In this section, we first review some representative selective RNN frameworks, and then introduce the pointer network and its applications in NLP.

Selective RNNs

Selective RNNs started with LSTM-Jump (Yu et al., 2017) which relies on previous hidden states to predict the number of time steps that should be skipped after reading at least one input token, resulting in generally faster inference. Skip-RNN (Campos et al., 2017) improves on LSTM-Jump in the form of a binary gate to decide whether the state of the RNN is updated or copied from the previous time step. Unlike LSTM-Jump, which stops at certain time steps to make jumping decisions, Skip-RNN interrupts the input procedure of RNN at each time step. Structural-Jump-LSTM (Hansen et al., 2019) combines the advantages of LSTM-Jump and Skip-RNN to perform joint text skipping and jumping during inference. More specifically, Structural-Jump-LSTM consists of two agents: one to skip single words, and another to exploit punctuation markers (i.e. sub-sentential separators (,:), sentence-final characters (.!?), and other end-of-text markers) before jumping forward.

JUMPER (Liu et al., 2018), inspired by the cognitive process of text reading, scans a piece of text sequentially and makes classification decisions only at certain steps triggered by sufficient evidence, reducing total text reading by up to 30-40%. Even though JUMPER interrupts the LSTM's input procedure less frequently, it carries two risks: (1) if the critical cue word signalling that reading can stop is located at the end of the input sequence, JUMPER has to read in the entire input including any irrelevant information; (2) if reading is stopped prematurely due to sufficient evidence detected, the method may miss even more important information present in the remaining text not yet read, resulting in suboptimal performance. Unlike approaches which skip tokens, Skim-RNN *skims* unimportant tokens with the help of a small RNN to update only a fraction of the hidden state, as opposed to using a large RNN to update the entire hidden state when the words are important. However, Skim-RNN has to stop reading at each time step before it can make a decision.

None of the above models offer control over the skip rate. The ratio of words to skip or skim is typically determined automatically during training. Furthermore, these selective RNNs exploit preceding information (e.g. the hidden state at time step $t-1$) only to make skip decisions which not only makes them unstable in practice and but also harder to converge during training. The most recent skipping model, Leap-LSTM (Huang et al., 2019), introduces a skip rate which is controllable during training – but not inference – by adding a related penalty term to its cost function. Even though Leap-LSTM includes in its skip decisions information both from the following text and the current word, Leap-LSTM still predicts skipping behaviour at each time step, thus interrupting the LSTM input procedure. Leap-LSTM is as undesirable as other existing selective RNNs in this regard.

Pointer Network

The first pointer network, proposed by Vinyals et al. (2015), utilises an attention mechanism as a pointer to select some item in the input sequence as the output, and makes an attempt to solve the problem of variable size output dictionaries where the number of target classes in each step of the output depends on the length of the input which is variable. Gulcehre et al. (2016) use the pointer network in neural machine translation to deal with rare and unseen words: in their approach, the pointer network learns to either copy a word from the source sentence or generate a word from the target vocabulary for output. The pointer network has also been used in text summarisation (See et al., 2017) to copy words which should be maintained in the output. These methods jointly demonstrate that the attention mechanism can be used as a pointer to copy important information from the input for language generation tasks.

Our approach is similar to two existing proposals, namely 1) the coarse-to-fine question answering framework for long documents by (Choi et al., 2017) and 2) the Pointer-Generator network for text summarisation by (See et al., 2017). The first method uses a fast model to select sentences relevant to a question and feeds them to a slow RNN to produce the final answer. We take a more direct approach by employing a pointer network to select information at the word level, rather than the sentence level, and introduce a skip rate to control the percentage of information to be copied in the output. The second method uses a Pointer-Generator network to decide whether to copy a word from the input or generate a word from the vocabulary for text summarisation. Although the Pointer-Generator network is used in an encoder-decoder architecture for language generation, our Pointer-LSTM is designed to select important words for text classification and hence has a more concise structure.

3 Our Approach

With faster long text classification as our goal, we investigate a selective LSTM architecture which uses the attention mechanism as a pointer to select a controllable portion of important words. We regard word selection as a generation process and take the Pointer-Generator network for text summarisation (See et al., 2017) as our starting point. We first introduce the Pointer-Generator network and then describe our modifications to it suitable for text classification (henceforth referred to as Pointer-LSTM).

3.1 Pointer-Generator Network

The Pointer-Generator network (See et al., 2017) is used in an encoder-decoder architecture for language generation. The encoder converts an input sequence into a weighted, fixed-dimensional context vector, and the decoder generates an output sequence based on the decision of the Pointer-Generator network over two choices: (i) generating a word from the vocabulary or (ii) copying a word from the input sequence. The Pointer-Generator network produces a probability indicator p_{gen}

$$p_{gen} = \sigma(w_{h^*} h_t^* + w_s s_t + w_o o_{t-1} + b_{ptr}) \tag{1}$$

where the vectors w_{h^*}, w_s, w_o and the scalar b_{ptr} are learnable parameters; σ is the sigmoid function; h_t^*, s_t, and o_{t-1} are the encoder context vector, decoder hidden state, and preceding decoder output for the decoder at time step t, respectively; and b_{ptr} is a bias term for the pointer network. $p_{gen} \in [0, 1]$ represents the probability of generating a word from the vocabulary versus copying a word from the source text.

While it is well suited to text summarisation, the Pointer-Generator network is cumbersome when it comes to selecting words for text classification: the autoregressive pointer network is slow and the generation of a word from the vocabulary is unnecessary. We accordingly propose Pointer-LSTM, a simplified yet effective pointer network architecture dedicated to word selection in text classification.

3.2 Pointer-LSTM: A Simplified Pointer-Generator Network

Pointer-LSTM consists of two BiLSTM networks, namely the Small LSTM and the Main LSTM, as shown in Figure 1. Small LSTM reads in the entire input sequence and produces an attention distribution over the input tokens. We set a small hidden state dimension (e.g. 20) for Small LSTM for faster processing. Gumbel-Softmax (Jang et al., 2017) is then used to sample the attention distribution to produce an importance distribution over the input tokens. We do not take the attention distribution as the importance distribution directly. Because there is no explicit supervision for the attention to select important words, the use of Gumble-Softmax can introduce a random sampling process and helps the network converge better in training. Based on a pre-set skip rate r, we select top-K words, concatenate their word embeddings and the hidden states from Small LSTM, and feed them to Main LSTM which has a larger hidden state dimension (e.g. 200). A softmax function is applied to the final state of Main LSTM to produce a probability distribution over all classes for text classification.

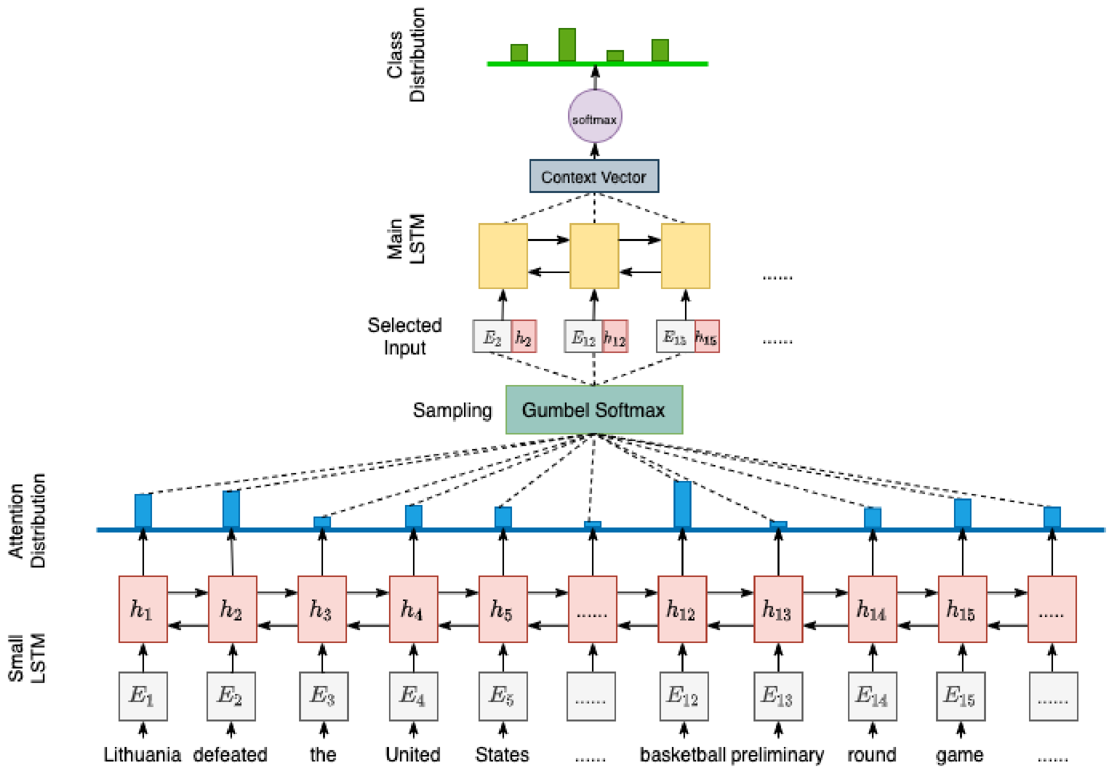
Class Distribution
softmax
Context Vector
Main LSTM
......
Selected Input
E_2 h_2
E_{12} h_{12}
E_{15} h_{15}
......
Sampling
Gumbel Softmax
Attention Distribution
Small LSTM
h_1 h_2 h_3 h_4 h_5 ...... h_{12} h_{13} h_{14} h_{15} .....
E_1 E_2 E_3 E_4 E_5 ...... E_{12} E_{13} E_{14} E_{15} ......
Lithuania defeated the United States ...... basketball preliminary round game ......

Figure 1: The proposed Pointer-LSTM architecture.

More specifically, given an input sequence $X = \{x_1, x_2, \ldots, x_T\}$ where x_i is the i^{th} word of the sequence, let E_i denote the pre-trained word embedding of x_i for Small LSTM, and h_t denote the concatenation of forward and backward LSTM hidden states at time step t. Small LSTM calculates a self-attention distribution over the input sequence as follows:

$$\alpha = \texttt{softmax}(\omega \texttt{tanh}(\boldsymbol{H})) \tag{2}$$

where $\boldsymbol{H} = [h_1, h_2, \ldots, h_T]$; ω is a learnable weight vector; and α is the attention distribution.

The Gumbel-Softmax for sampling the attention distribution is defined as follows:

$$d_i = \frac{exp((\log(\alpha_i) + g_i)/\tau}{\sum_{j=1}^{T} exp((\log(\alpha_j) + g_j)/\tau)} \tag{3}$$

where $i = \{1, \ldots, T\}$; α_i is the i^{th} weight in the attention distribution α; τ is the softmax temperature; and $\boldsymbol{d} = \{d_1, d_2, \ldots, d_T\}$ is the importance distribution over the input sequence.

We rank the importance distribution $\boldsymbol{d}$ and select top-K important words based on the skip rate r via

$$K = \lceil T * (1 - r) \rceil \tag{4}$$

where $\lceil \rceil$ is the ceiling operation.

Figure 1 illustrates the Pointer-LSTM workflow on a sentence from the AGNews data set (*"Lithuania defeated the United States 94-90 in an Olympic men 's basketball preliminary round game , only the fourth loss in 115 Olympic starts for the defending champions"*) (see Section 4.1). The topic label of this sample sentence with 28 words is *Sports*. The small attentive LSTM encodes the entire input sequence into an attention distribution which is then sampled by Gumbel-Softmax to select important words such as *"defeated"* and *"basketball"*. If the skip rate is set to $r = 0.9$, then $K = \lceil 28 * (1 - 0.9) \rceil = 3$ as a result of which only three words with the highest probability scores in the importance distribution $\boldsymbol{d}$ are selected for class prediction.

3.3 Reinforcement Learning

The selection of top-K words unfortunately makes the entire framework non-differentiable because Small LSTM is separated from Main LSTM in training. Another complication arises from the fact that there exists no ground truth for the selection of important words in text classification. In order to train Small LSTM, we use a reinforcement learning method after Williams (1992). We regard word selection as actions taken to maximise a reward related to the classification result. Similar to Choi et al. (2017), we define the reward to be the log probability of the corrected class given a word in K selected words:

$$R(x_l) = \log p(y = y^* | x_l) \tag{5}$$

where x_l is an important word in the set S_K which contains the selected top-K words; y^* is the true class; y is the predicted class; and $R_\theta(S_K)$ is the reward. According to this equation, the reward is close to zero when the prediction is correct; if the prediction is incorrect, the reward becomes negative.

The learning objective is to maximise the expected reward which is equivalent to minimising the negative expected reward via

$$L_r = - \sum_{x_l \in S_K} d(x = x_l | X) \cdot \log p(y = y^* | x_l) \tag{6}$$

where X is the input sequence; x_l denotes an important word; and $d_\theta(x = x_l | X)$ is the importance probability from Eq. (3).

The final cost function of the Pointer-LSTM networks is defined as

$$\mathcal{L} = \mathcal{L}_c + \mathcal{L}_r \tag{7}$$

where $\mathcal{L}_c$ is classification loss; and $\mathcal{L}_r$ is the reward loss.

4 Experiments

We evaluate Pointer-LSTM on four benchmark data sets for long text classification. The state-of-the-art selective RNN - Leap-LSTM (Huang et al., 2019) - is used as our baseline[1]. We first compare the accuracy of our models and their speed-up over the standard LSTM architecture[2]. We then investigate the robustness of Pointer-LSTM in cases where the skip rate is changed during inference, focusing on accuracy at different skip rates.

[1]The source code for Leap-LSTM (on Tensorflow) can be found at: `https://github.com/ht1221/leap-lstm`. We re-implemented their work on PyTorch.

[2]Leap-LSTM is a unidirectional LSTM framework which skips words from left to right. Therefore, to perform a fair comparison, we follow the baseline of standard LSTM in (Huang et al., 2019). We will compare our work with more diverse baselines (such as BiLSTM and Transformers) in terms of accuracy and speed in future work.

Data Set	Task	#Train	#Validation	#Test	Classes	Avg. #Words
AGNews	News classification	110K	10K	76K	4	45
DBPedia	Ontology classification	540K	20K	70K	14	55
Yelp Polarity	Binary sentiment analysis	540K	20K	38K	2	154
Yelp Full	Multi-class sentiment analysis	600K	50K	50K	5	156

Table 1: Summary statistics of our four benchmark data sets. NB. Because the original data sets do not come with separate validation sets, we split the original training sets by 90%-10% to obtain new training vs. validation partitions within each data set.

4.1 Data Sets

Following Zhang et al. (2015), we use the large-scale AG's News, DBPedia Ontology, Yelp Review Full, and Yelp Review Polarity data sets to represent challenging long text classification conditions. Table 1 provides overall summary statistics of these data sets. As we can see, sentences in the AGNews and DBPedia data sets contain ca. 50 words on average while those in the Yelp Review data sets are considerably longer, with more than 150 words on average. The maximum length of sentences in the DBPedia, Yelp Full, and Yelp Polarity data sets is over 1,000 words each.

4.2 Model Settings

Our neural networks were implemented on PyTorch[3]. We initialise word embeddings with GloVe 100-D 6B word vectors (Pennington et al., 2014). For word embeddings not included in GloVe word vectors, we use random initialisation. The word embeddings are updated during training. The hyper-parameters for Pointer-LSTM are summarised in Table 2.

Parameter	**Value**
optimiser	Adam
dimension of pre-trained GloVe word embeddings	100
size of hidden states in Small LSTM	20
size of hidden states in Main LSTM	200
direction of two LSTMs	bidirectional
learning rate	0.001
number of layers in LSTM	1
epochs	5
batch size	64
dropout rate	0.3
lowercase	True
temperature τ	0.1

Table 2: Pointer-LSTM model settings and hyper-parameters.

We trained Pointer-LSTM at two skip rates that have been tested on the Leap-LSTM baseline: (i) a low skip rate at $r = 0.25$, and (ii) a high skip rate at $r = 0.9$. Note that the skip-rate of Leap-LSTM is fixed after training: a model trained with a skip rate of $r = 0.25$ cannot be changed to any other skip rate during inference, for example. As a result, multiple Leap-LSTM models need to be trained to actualise different skip rates. By contrast, Pointer-LSTM allows changes to be made to the skip rate during inference which we illustrate in our experiments.

4.3 Accuracy and Speed-up

Table 3 compares the accuracy, actual skip rate, and speed of Pointer-LSTM and the Leap-LSTM. The *LSTM* and *Leap-LSTM* scores were obtained from (Huang et al., 2019). *Acc.* denotes the accuracy

[3]`https://pytorch.org/`

Model	r	AGNews			DBPedia			Yelp Full			Yelp Polarity		
		Acc.	Skip	Speed	Acc.	Skip	Speed	Acc.	Skip	Speed	Acc.	Skip	Speed
Leap-LSTM	0.25	**93.93**	24.93	1.1x	**99.10**	27.94	1.2x	65.91	22.71	1.1x	96.20	23.57	1.1x
	0.9	92.62	86.33	2.3x	98.87	87.58	2.8x	61.70	80.03	1.9x	94.34	86.17	2.0x
Pointer-LSTM	0.25	93.86	23.59	1.1x	99.01	23.85	1.1x	65.63	24.60	1.1x	**96.44**	24.59	1.2x
	0.9	**93.30**	88.76	2.6x	**98.89**	88.99	3.5x	**63.18**	89.65	3.3x	**95.82**	89.64	2.2x
LSTM		93.23	–	1.0x	99.01	–	1.0x	**65.93**	–	1.0x	95.92	–	1.0x

Table 3: Results on four benchmark data sets.

metric; r is the pre-set skip rate for training; *Skip* denotes the actual skip rate during inference; and *Speed* is the speed-up in runtime on the test set compared to LSTM.

In terms of accuracy, Pointer-LSTM and Leap-LSTM both outperform LSTM in both skip rate conditions (except on the Yelp Full data set). Nevertheless, even on the Yelp Full data set, Pointer-LSTM and Leap-LSTM can outperform LSTM when the skip rate is lowered (e.g. to 0.15), as verified in our experiments and discussed in (Huang et al., 2019). This provides further evidence that the reduction of unimportant words in the input can lead to an increase in accuracy.

Furthermore, Pointer-LSTM is more accurate than Leap-LSTM when the skip rate is high. Leap-LSTM performs marginally better than Pointer-LSTM on three data sets (NB. not on Yelp Polarity) when the skip rate r is set to 0.25 (i.e. when only a quarter of the words are skipped). Pointer-LSTM in turn achieves better results than Leap-LSTM on all four data sets when the skip rate r is set to 0.9 (i.e. when ca. 90% of words are skipped). Moreover, Pointer-LSTM performs particularly well on long sentences at very high skip rates. Table 3 shows that the accuracy gain of Pointer-LSTM over Leap-LSTM at $r = 0.9$ is more significant on Yelp Full and Yelp Polarity data where, on average, sentences are approximately three times longer than sentences in the other two data sets.

We believe that the advantage brought by Pointer-LSTM in the high skip rate setting reflects the fact that Small LSTM was used to learn a complete context representation of the entire input sequence: without ground truth information, word selection or skipping in selective RNNs is not explicitly supervised during training and may hence be more error-prone, leading to the absence of some important words in the input. The ramifications of such errors become markedly prominent when the skip rate is high while the input sequence is long. Unlike Leap-LSTM and other existing selective RNNs which decide whether or not a word should be skipped at each time step and which use only the selective words as their input, Pointer-LSTM learns a context representation over the entire input sequence for each time step, and feeds the context representations of selected words into the main LSTM. Pointer-LSTM can effectively capture some global context in the input sequence, mitigating information loss stemming from word selection.

In terms of inference time[4], Pointer-LSTM performs significantly faster than Leap-LSTM in the high skip rate setting at $r = 0.9$, achieving up to 2.2x$\sim$3.5x speed-ups over LSTM. At $r = 0.25$, Pointer-LSTM is 1.1x$\sim$1.2x faster than LSTM, suggesting a speed-up rate similar to that of Leap-LSTM. These results indicate that Pointer-LSTM is an effective approach towards faster long text classification. The higher speed of Pointer-LSTM over Leap-LSTM at high skip rates may correlate with the maintenance of a coherent input procedure for the LSTM components.

4.4 Dynamic Skip Rate: Flexibility and Robustness of Pointer-LSTM

Leap-LSTM has an advantage over other existing selective RNNs in that it enables greater control over the skip rate for training. However, the skip rate of Leap-LSTM cannot be trained during inference. As a result, distinct skip rate-specific Leap-LSTM models need to be trained which is inconvenient or simply impossible in some real-world conditions, especially when computational resources are limited. Pointer-LSTM addresses this issue by offering control over the skip rate during inference. The skip rate can be set to any value between 0 and 1. To investigate the robustness of Pointer-LSTM when the skip rate is changed during inference, we evaluate the accuracy and actual skip rate of Pointer-LSTM in such scenarios. We use *tuned* to refer to a scenario in which Pointer-LSTM is trained with a fixed skip rate

[4]To evaluate the speed-up, we implemented a standard LSTM network on PyTorch, ran it on all four data sets, and compared its inference times with Pointer-LSTM on a CPU-only machine.

r during training and r for inference. We use *dynamic* to refer to a scenario whereby the skip rate is changed to a different value $r*$ during inference while all other model parameters stay unchanged.

Table 4 demonstrates accuracy and actual skip rates when we change the pre-set skip rate from the default $r = 0.25$ and $r = 0.9$ for training (see Table 3) to some other values during inference. As we can see, Pointer-LSTM remains robust at dynamic skip rates during inference. When the skip rate increases from 0.25 during inference to $r = \{0.3, 0.4, 0.5\}$, the accuracy of Pointer-LSTM decreases only marginally on all four data sets; correspondingly, when the skip rate decreases from 0.9 to $r = \{0.8, 0.7, 0.6\}$, Pointer-LSTM either maintains comparable accuracy levels or indeed achieves better performance in many scenarios apart from the AGNews data set which led to a significant drop when the skip rate was set dynamically to $r = 0.6$. Because Pointer-LSTM relies heavily on global contexts to select words, the significant drop on this very data set may correlate with the larger number of shorter sentences in the AGNews data compared to other data sets. We will explore this topic on additional data sets which contain either short or long sentences only in future work.

	r	AGNews		DBPedia		Yelp Full		Yelp Polarity	
		Acc.	Skip	Acc.	Skip	Acc.	Skip	Acc.	Skip
Tuned	*0.25*	93.86	23.59	99.01	23.85	65.63	24.60	96.44	24.59
Dynamic	0.3	93.67	28.77	99.00	28.99	65.36	29.65	96.27	29.64
	0.4	93.21	38.66	98.91	38.90	64.66	39.62	96.12	39.61
	0.5	91.82	48.31	98.79	48.64	63.45	49.52	95.69	49.51
Tuned	*0.9*	93.30	88.76	98.89	88.99	63.18	89.65	95.82	89.64
Dynamic	0.8	91.25	78.65	98.89	78.90	63.23	79.62	95.93	79.61
	0.7	80.47	68.76	98.87	69.00	63.17	69.65	95.92	69.64
	0.6	67.58	58.65	98.87	58.90	63.13	59.62	95.92	59.61

Table 4: Impact of dynamic skip rates on Pointer-LSTM performance.

Furthermore, the actual skip rate of Pointer-LSTM is (i) close to the pre-set skip rate of Leap-LSTM and (ii) more stable than the actual skip rate of Leap-LSTM, as can can seen in Tables 3 and 4. We observe that:

- at $r = 0.25$, the mean and unbiased variance of the actual skip rate for Pointer-LSTM are 24.16 and 0.27, respectively, compared to 24.76 and 5.34 for Leap-LSTM. Although the actual skip rate of Leap-LSTM is on average slightly closer to the pre-set skip rate in comparison with Pointer-LSTM, our variance is much lower, indicating a more stable skip rate over different data sets;
- at $r = 0.9$, the mean and unbiased variance of Pointer-LSTM are 89.26 and 0.21, respectively, while those of Leap-LSTM are only 85.03 and 11.5. In this setting, the average actual skip rate of Pointer-LSTM is much closer to the pre-set skip rate of Leap-LSTM and it is also much more stable; and
- when the skip rate of Pointer-LSTM is dynamically set to $r = \{0.3, 0.4, 0.5, 0.6, 0.7, 0.8\}$ during inference, the mean and unbiased variance values of the actual skip rates are $\{29.26, 39.20, 49.0, 59.19, 69.26, 79.20\}$ and $\{0.2, 0.24, 0.38, 0.25, 0.2, 0.25\}$, respectively. These average skip rates are close to the pre-set values at low variance levels, showing that Pointer-LSTM offers good control of the skip rate even in dynamic settings.

4.5 Sample Output

To further compare Pointer-LSTM with the baseline Leap-LSTM, we use the examples in Huang et al. (2019) to illustrate word selection for target prediction.

Table 5 compares the outputs of Pointer-LSTM and Leap-LSTM on sample *Business* and *Sports* snippets. Words in red are kept while the rest are skipped. In these two examples, Leap-LSTM kept 10/40 vs. 12/31 words while our Pointer-LSTM kept 5/40 vs. 4/31 words, respectively, calculated by Eq. (4). Accordingly, on these two specific snippets, the actual skip rates of Leap-LSTM are 75.0% and 61.3% which are far lower than the pre-set skip rate of 0.9; ours are 87.5% and 87.1% both of which are significantly closer to $r = 0.9$. With respect to actual skip rates, Pointer-LSTM offers greater controllability and stability compared to Leap-LSTM.

System	Example 1 (Business)	Example 2 (Sports)
Leap-LSTM	new york (reuters) - u.s. treasury prices paused for breath on tuesday after a blistering two-session rally ran out of steam , though analysts still saw room to the upside given the large short-base in the market .	afp - lithuania defeated the united states 94-90 in an olympic men 's basketball preliminary round game , only the fourth loss in 115 olympic starts for the defending champions .
Pointer-LSTM	new york (reuters) - u.s. treasury prices paused for breath on tuesday after a blistering two-session rally ran out of steam , though analysts still saw room to the upside given the large short-base in the market .	afp - lithuania defeated the united states 94-90 in an olympic men 's basketball preliminary round game , only the fourth loss in 115 olympic starts for the defending champions .

Table 5: Two examples from the AGNews test set with skip rate at $r = 0.9$.

From the selected words' perspective, we can see that most of the words selected by Pointer-LSTM represent a subset of the words selected by Leap-LSTM. For example, $\{$treasury, analysts, market$\}$ in Example 1 were selected by both systems, indicating that they are recognised as important words for the *Business* topic. However, Pointer-LSTM also selected the comma $\{,\}$ in Example 1 which is counter-intuitive to humans. We infer that, although the attention mechanism can learn a probability distribution to distinguish between important and unimportant words, it cannot be guaranteed that the probability distribution is accurate due to the fact that important word selection is not a supervised learning process. The ability to learn an accurate attention distribution by the small LSTM in Poiner-LSTM framework also deteriorates when the input sequence is long. These observations guide our future work towards more advanced mechanisms to score important words, accordingly.

5 Conclusion and Future Work

In this paper, we propose a novel selective RNN framework for long text classification – Pointer-LSTM – which can select important words from the input sequence more effectively than recent state-of-the-art methods. Pointer-LSTM uses a small attentive LSTM network to produce an importance distribution over all input tokens using a pre-set skip rate to select important words as the input to a large LSTM network for target prediction. This architectural design offers two advantages over existing selective RNN frameworks in that (i) Pointer-LSTM maintains a coherent, uninterrupted input process for the LSTM components, and (ii) the skip rate can be adjusted during inference. Our experiments on four long text classification data sets demonstrate that Pointer-LSTM offers higher accuracy and faster inference than the state-of-the-art selective RNN (Leap-LSTM) at a high skip rate. The Pointer-LSTM's accuracy also remains high when the skip rate is changed during inference. Compared to Leap-LSTM, Pointer-LSTM offers a more stable control over the skip rate during training.

In many industry scenarios under heavy resource constraints (such as memory-limited or CPU-only hardware, limited training data, and latency requirements), not all large-scale deep architectures can be deployed as-is. Our skipping model, which can scan any long sequence and keep important words while reducing information loss, does not suffer from the inherent maximum input length limitation which hamstrings many a deep architecture.

We will explore alternative attention mechanisms to select important words in future work. Owing to the fact that the input into the main model can be condensed significantly after word selection, we will explore non-RNN networks (for example, convolutional neural networks and Transformer variants) as the main model for target classification.

Acknowledgements

The authors would like to thank the reviewers for their insightful comments, Guruprasad Sethurathinam for infrastructure support, and internal AIG Investments AI reviewers for their feedback.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Victor Campos, Brendan Jou, Xavier Gir'o i Nieto, Jordi Torres, and Shih-Fu Chang. 2017. Skip RNN: learning to skip state updates in recurrent neural networks. *CoRR*, abs/1708.06834.

Eunsol Choi, Daniel Hewlett, Jakob Uszkoreit, Illia Polosukhin, Alexandre Lacoste, and Jonathan Berant. 2017. Coarse-to-fine question answering for long documents. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 209–220.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning, December 2014*.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 140–149, aug.

Michael Hahn and Frank Keller. 2016. Modeling human reading with neural attention. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 85–95, November.

Christian Hansen, Casper Hansen, Stephen Alstrup, Jakob Grue Simonsen, and Christina Lioma. 2019. Neural speed reading with structural-jump-lstm. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.

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

Ting Huang, Gehui Shen, and Zhi-Hong Deng. 2019. Leap-lstm: Enhancing long short-term memory for text categorization. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 5017–5023.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

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*, pages 2326–2335, Lisbon, Portugal.

Xianggen Liu, Lili Mou, Haotian Cui, Zhengdong Lu, and Sen Song. 2018. Jumper: Learning when to make classification decision in reading. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 4237–4243.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, jul.

Min Joon Seo, Sewon Min, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Neural speed reading via skim-rnn. *CoRR*, abs/1711.02085.

Ming Tan, Bing Xiang, and Bowen Zhou. 2015. Lstm-based deep learning models for non-factoid answer selection. *CoRR*, abs/1511.04108.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems 28*, pages 2692–2700.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256.

Adams Wei Yu, Hongrae Lee, and Quoc V. Le. 2017. Learning to skim text. *CoRR*, abs/1704.06877.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems 28*, pages 649–657.