Analyzing the Learnability and Representability of Recurrent Architectures

A Thesis

submitted by

PRATYUSH MAINI

in partial fulfilment of the requirements
for the award of the degree of

BACHELOR OF TECHNOLOGY

Department of Computer Science and Engineering
Indian Institute of Technology Delhi

July 2020

CERTIFICATE

This is to certify that the thesis (or project report) titled **Analyzing the Learnability and Representability of Recurrent Architectures**, submitted by **Pratyush Maini**, to the Indian Institute of Technology Delhi, for the award of the degree of **Bachelor of Technology**, is a bona fide record of the research work done by him under my (our) supervision. The contents of this thesis (or project report), in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Place: New Delhi

Date: 14th July 2020

Dr. Mausam
Deptartment of Computer Science
& Engineering
IIT Delhi, 110 016

ACKNOWLEDGEMENTS

I extend my sincere acknowledgement to my advisor Dr. Mausam, who invigorated my interest in the beautiful field of artificial intelligence and natural language processing. I am indebted to him for mentoring me ever since I was an undergraduate in my third year, and for showing faith in me right throughout. I thank him for all the wonderful and thought-provoking discussions we had, that often left me in excitement and appreciation of the field.

I thank Keshav for guiding me into the lab, from the times when I did not how to manage working over large clusters and code-bases. I am left in awe of the immense patience he has shown in helping me learn the basics of NLP during my initial days. I also thank him for making me the benefactor of his assortment of wonderful sweets and savouries. I thank Danish for teaching me how to organize my thoughts, how to write papers, and most importantly how to balance research with fun and engagement. I have massively benefitted from our discussions on and off-research and have learnt a lot through our engagement. I am thankful to Sankalan for collaborating with me and patiently answering all the questions and the doubts that I had with regards to the theoretical aspects of our work. I am also grateful to HPC IIT Delhi for providing high-speed GPU clusters which could enable my research.

I enjoyed my discussions over research topics with fellow members of the DAIR lab. I had a wonderful time talking with them about the varied projects they engaged in, and thank all of them for the time we spent together during my work at the lab, especially when we all sat together and supported each other before paper deadlines.

During my undergraduate years, I have been fortunate to have had the opportunity to adapt and learn from the research styles of my amazing external collaborators: Prof. Zico Kolter, Eric Wong at CMU, Prof. Bo Li at UIUC, Prof. Dawn Song, Xinyun Chen at UCB, Prof. Nicolas Papernot at UofT/Vector and Prof. James Larus, Stuart

Byma at EPFL. While we worked on project ideas that do not overlap with my thesis at IIT Delhi, I thank all of them for teaching me research and widening my horizon of knowledge which proved to be immensely beneficial for me in my work towards this thesis.

I stay grateful to all my friends who would teach me contents of nearly every course I took during my undergraduate journey, discuss intellectually challenging problems and work on projects that would make us think out of the box. But more importantly, for all the amazing times we spent together that allowed us to switch off from work and relax. These remain my most cherishable memories from the last few years. Lastly, but most importantly, I am grateful to my parents and family for their constant emotional and mental support throughout these years, and for always being a phone call away in times of need.

ABSTRACT

KEYWORDS: Learnability; Represntability; LSTM; Analysis; Recurrent Architectures; Vanishing Gradients

LSTMs were introduced to mitigate the problem of vanishing gradients in standard recurrent architectures. Pooling-based recurrent neural architectures consistently outperform their counterparts without pooling. However, the reasons for their enhanced performance are largely unexamined. In this work, we examine three commonly used pooling techniques (mean-pooling, max-pooling, and attention), and propose max-attention, a novel variant that effectively captures interactions among predictive tokens in a sentence. We find that pooling-based architectures substantially differ from their non-pooling equivalents in their learning ability and positional biases–which elucidate their performance benefits. By analyzing the gradient propagation, we discover that pooling facilitates better gradient flow compared to BiLSTMs. Further, we expose how BiLSTMs are positionally biased towards tokens in the beginning and the end of a sequence. Pooling alleviates such biases. Consequently, we identify settings where pooling offers large benefits: (i) in low resource scenarios, and (ii) when important words lie towards the middle of the sentence. Among the pooling techniques studied, max-attention is the most effective, resulting in significant performance gains on several text classification tasks.

Next, we attempt to distinguish the effect of learnability and representability in model expressivity. To these ends, first we investigate the representative power of LSTMs under different types of pooling and attempt to explore problems that pooled LSTMs can not solve. We theoretically examine how the gradients of LSTMs vanish independent of the task and show that LSTMs have universal representation power. After establishing the learning issues faced by LSTMs, we attempt to mitigate them using four different training time modifications (and no architectural changes). While we are able

to show substantial gains in the performance of LSTMs in tasks where it was shown to fail in past works, we observe that despite our best efforts, standard LSTM performance is generally succeeded by LSTMs in the presence of advanced pooling techniques. This opens interest for further exploring aspects other than gradient propagation that limit LSTMs, but not their pooling-based counterparts.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

RNN	Recurrent Neural Networks
LSTM	Long Short Term Memory Networks
MAXATT	Max-attention pooled LSTM
MAXPOOL	Max-pooled LSTM
MEANPOOL	Mean-pooled LSTM
ATT	Attention pooled LSTM

CHAPTER 1

Why and When should you Pool?

1.1 Introduction

Pooling mechanisms are ubiquitous components in Recurrent Neural Networks (RNNs) used for natural language tasks. Pooling operations consolidate hidden representations from RNNs into a single sentence representation. Various pooling techniques, like mean-pooling, max-pooling, and attention, have been shown to improve the performance of RNNs on text classification tasks (Lai et al., 2015; Conneau et al., 2017). Despite widespread adoption, precisely how and when pooling benefits the models is largely under-explored.

In this work, we perform an in-depth analysis comparing popular pooling methods, and proposed max-attention, with standard BiLSTMs for several text classification tasks. We identify two key factors that explain the benefits of pooling techniques: learnability, and positional invariance.

First, we analyze the flow of gradients for different classification tasks to assess the learning ability of BiLSTMs (§ 1.5). We observe that in the initial epochs, the gradients corresponding to hidden representations in the middle of the sequence vanish. On training for more examples, these gradients slowly recover, suggesting that the gates of standard BiLSTMs require many examples to learn. In contrast, we find the gradient norms in pooling-based architectures to be free from this problem. Pooling enables a fraction of the gradients to directly reach any hidden state instead of having to backpropagate through a long series of recurrent cells. Thus we hypothesize, and subsequently confirm, that pooling is particularly beneficial for tasks with long input sequences.

Second, we explore the positional biases of BiLSTMs, with and without pooling (§ 1.6). Across several classification tasks, and various novel experimental setups, we

expose that BiLSTMs are less responsive to tokens towards the middle of the sequence, when compared to tokens at the beginning or the end of the sequence. However, we find that this bias is largely absent in pooling-based architectures, indicating their ability to respond to salient tokens regardless of their position.

Third, we propose max-attention, a novel pooling technique, which combines the advantages of max-pooling, and attention (§ 1.3.2). Max-attention uses the max-pooled representation as its query vector to compute the attention weights for each hidden state. Max-pooled representations are extensively used in the literature to capture prominent tokens (or objects) in a sentence (or an image) (Zhang and Wallace, 2015; Boureau et al., 2010b). Therefore, using them as a query vector effectively captures interactions among salient portions in the input. Max-attention is simple to use, and yields performance gains over other pooling methods on several classification setups.

The contents of this chapter describe work done in Maini et al. (2020).

1.2 Related Work

Pooling: A wide body of work compares the performance of different pooling techniques in object recognition tasks (Boureau et al., 2010a,b, 2011) and finds max-pooling to generally outperform mean-pooling. However, pooling in natural language tasks is relatively understudied. For some text classification tasks, pooled recurrent architectures (Lai et al., 2015; Zhang and Wallace, 2015; Johnson and Zhang, 2016; Jacovi et al., 2018; Yang et al., 2016a), outperform CNNs and BiLSTMs. Additionally, for textual entailment tasks, Conneau et al. (2017) find that max-pooled representations better capture salient words in a sentence. Our work extends the analysis and examines several pooling techniques, including attention, for BiLSTMs applied to natural language tasks. While past approaches assess the ability of pooling in capturing linguistic phenomena, to the best of our knowledge, we are the first to systematically study the training advantages of various pooling techniques.

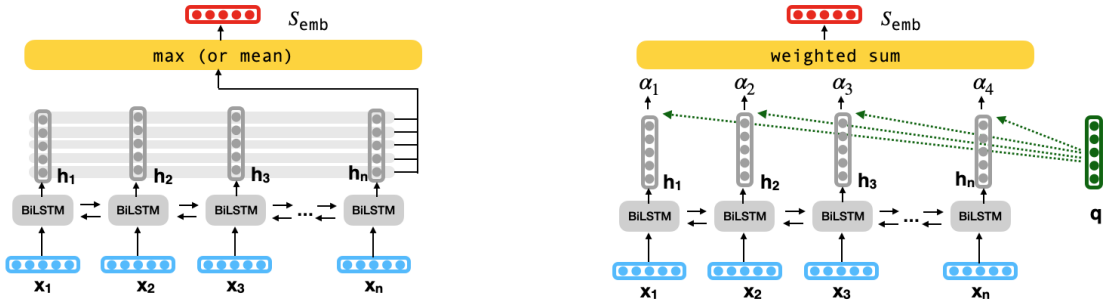


Figure 1.1: A pictorial overview of the pooling techniques. Left: element-wise mean and max pooling operations aggregate hidden representations. Right: attention scores (α) are computed using the similarity between hidden representations (h) and query vector (q), which are subsequently used to weight hidden representations. Our proposed max-attention uses the sentence embedding from max-pooling as a query to attend over hidden states.

Attention: First proposed as a way to align target tokens to the source tokens in translation (Bahdanau et al., 2014), the core idea behind attention—learning a weighted sum of the hidden states—has been widely adopted. As attention aggregates hidden representations, we consider it under the umbrella of pooling. Recently, Pruthi et al. (2020) conjecture that attention offers benefits during training; our work explains, and provides empirical evidence to support the speculation.

Gradient Propagation: Vanilla RNNs are known to suffer from the problem of vanishing and exploding gradients (Hochreiter, 1991; Bengio et al., 1994). In response, Hochreiter and Schmidhuber (1997) invented LSTMs, which provide a direct connection passage through all the cells in order to remember new inputs without forgetting prior history. However, recent work suggests that LSTMs do not solve this problem completely (Arjovsky et al., 2015a; Chandar et al., 2019a). Our work quantitatively investigates this phenomenon, exposing scenarios where the effect is pronounced, and demonstrating how pooling techniques mitigate the problem, leading to better sample efficiency, and generalization.

1.3 Methods

1.3.1 Background and Notation

Let $s = \{x_1, x_2, \ldots, x_n\}$ be an input sentence, where x_t is a representation of the input word at position t. A recurrent neural network such as an LSTM produces a hidden state h_t, and a cell state c_t for each input word x_t, where $h_t, c_t = \phi(h_{t-1}, c_{t-1}, x_t)$.

Basic RNN

Recurrent Neural Networks use a series of input sequence x_t and pass it sequentially over a network of hidden states where each each hidden state leads to the next. Mathematically, this is given by:

$$h_t = \sigma(Ux_t + Wh_{t-1} + b)$$
$$y_t = \text{softmax}(Vh_t + c)$$

where x_t refers to the input sequence at time step t, and W, U, V are weights for the RNN cell, and σ is a non-linearity of choice.

LSTM

The forward propagation of information in a basic LSTM are governed by the following equations:

$$i_t = \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi})$$
$$f_t = \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf})$$
$$g_t = \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg})$$
$$o_t = \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho})$$
$$c_t = f_t * c_{t-1} + i_t * g_t$$
$$h_t = o_t * \tanh(c_t)$$

where at time t, h_t is the hidden state, c_t is the cell state, x_t is the input, and i_t, f_t, g_t, o_t are the input, forget, cell, and output gates, respectively. σ is the sigmoid function, and * is the Hadamard product.

Standard BiLSTMs concatenate the first hidden state of the backward LSTM, and the last hidden state of the forward LSTM for the final sentence representation: $s_{\text{emb}} = [\overrightarrow{h_n}, \overleftarrow{h_1}]$. The sentence embedding (s_{emb}) is further fed to a downstream text classifier. For training BiLSTMs, multiple works have emphasized the importance of initializing the bias for forget gates to a high value (between 1-2) to prevent the model from forgetting information before it learns what to forget (Gers et al., 2000; van der Westhuizen and Lasenby, 2018). Hence, in our analysis, we experiment with both a high and low value of bias for the forget gate. For the non-pooled BiLSTM, we initialize the forget gate bias to 1, unless specified. For brevity, from hereon we would use h_t to mean $[\overrightarrow{h_t}, \overleftarrow{h_t}]$. Below, we formally discuss popular pooling techniques:

Max-pooling

For a max-pooled BiLSTM (MAXPOOL), the sentence embedding s_{emb}, is:

$$s^i_{\text{emb}} = \max_{t \in (1,n)} (h^i_t)$$

where h^i_t represents the i^{th} dimension of the hidden state corresponding to the word at position t. This implies that while backpropagating the loss, we find a direct pathway to the t^{th} hidden state as:

$$\frac{\partial s^i_{\text{emb}}}{\partial h^i_t} = \begin{cases} 1, & \text{if } t = \text{argmax}_{\text{t} \in (1,\text{n})}\, \text{h}^\text{i}_\text{t} \\ \frac{\partial h^i_k}{\partial h^i_t}, & \text{if } k = \text{argmax}_{\text{t} \in (1,\text{n})}\, \text{h}^\text{i}_\text{t}, \text{k} \neq \text{t} \end{cases}$$

Mean-pooling

For a mean-pooled LSTM, while the forward propagation remains the same as BiLSTM, the output embedding is given by:

$$s^i_{emb} = \frac{\sum_{t \in (1,n)} h^i_t}{n}$$

where h^i_t represents the i^{th} dimension of the hidden state at time step = t, and s_{emb} represents the final output embedding returned by the recurrent structure. This implies that during backpropagation we find a direct influence of the t^{th} hidden state as:

$$\frac{\partial s^i_{emb}}{\partial h^i_t} = \frac{\sum_{k \in (1,n)} \frac{\partial h^i_k}{\partial h^i_t}}{n}$$

Attention

Attention (ATT) works by calculating a non-negative weight for each hidden state that together sum to 1. Hidden representations are then multiplied with these weights and summed, resulting in a fixed-length vector (Bahdanau et al., 2014; Luong et al., 2015):

$$\alpha_t = \frac{\exp(h_t^\top q)}{\sum_{j=1}^{n} \exp(h_j^\top q)}; \quad s_{\text{emb}} = \sum_{t=1}^{n} \alpha_t h_t$$

where q is a learnable query vector. Several variations like hierarchical attention (Yang et al., 2016b), self-attention (Madasu and Rao, 2019) have been proposed for text classification. However, the above formulation (referred in literature as "Luong attention") is most widely used in text classification tasks (Jain and Wallace, 2019; Wiegreffe and Pinter, 2019; Pruthi et al., 2020).

1.3.2 Max-attention

We introduce a novel pooling variant called max-attention (MAXATT) to capture inter-word dependencies. It uses the max-pooled hidden representation as the query vector

for attention. Formally:

$$q^i = \max_{t \in (1,n)} (h_t^i); \qquad \hat{h_t} = h_t / \|h_t\|$$

$$\alpha_t = \frac{\exp(\hat{h_t}^\top q)}{\sum_{j=1}^{n} \exp(\hat{h_j}^\top q)}; \quad s_{\text{emb}} = \sum_{t=1}^{n} \alpha_t h_t$$

It is worth noting that the learnable query vector in Luong attention is the same for the entire corpus, whereas in max-attention each sentence has a unique locally-informed query. Previous literature extensively uses max-pooling to capture the prominent tokens (or objects) in a sentence (or image). Hence, using max-pooled representation as a query for attention allows for a second round of aggregation among important hidden representations.

1.3.3 Transformers

We briefly experiment with transformer architectures (Vaswani et al., 2017; Devlin et al., 2018a), and observe that purely attention-based architectures perform poorly on text-classification without significant pre-training. Further, the memory footprint for transformers is $O(n^2)$ vs $O(n)$ for LSTMs. Thus, for long examples used in some of our experiments (~ 4000 words), XL-Net (Yang et al., 2019) runs out of memory even for a batch size of 1 on a 32GB GPU. Therefore, we limit the scope of this work to recurrent architectures.

1.4 Datasets & Experimental Setup

We experiment with four different text classification tasks: (1) The **IMDb** dataset (Maas et al., 2011) contains movie reviews and their associated sentiment label; (2) **Yahoo! Answers** (Zhang et al., 2015) dataset comprises 1.4 million question and answer pairs, spread across 10 topics, where the task is to predict the topic of the answer, using the answer text; (3) **Amazon** reviews (Ni et al., 2019) contain product reviews from the Amazon website, filtered by their category. We construct a 20-class classification task

using these reviews; (4) **Yelp** Reviews (Zhang et al., 2015) is another sentiment polarity classification task.

1.4.1 Dataset Extraction

Amazon Reviews The Amazon Reviews Dataset (Ni et al., 2019) includes reviews (ratings, text, helpfulness votes) and product metadata (descriptions, category etc.) pertaining to products on the Amazon website. We extract the product category and review text corresponding to 2500 reviews from to each of the following 20 classes:

- Automotive
- Books
- Clothing Shoes and Jewelry
- Electronics
- Movies and TV
- Arts Crafts and Sewing
- Toys and Games
- Pet Supplies
- Sports and Outdoors
- Grocery and Gourmet Food
- CDs and Vinyl
- Tools and Home Improvement
- Software
- Office Products
- Patio Lawn and Garden
- Home and Kitchen
- Industrial and Scientific
- Luxury Beauty
- Musical Instruments
- Kindle Store

In the standard setting, we ensure that all reviews have lengths between 100 and 500 words.

IMDb The IMDb Movie Reviews Dataset (Maas et al., 2011) is a popular binary sentiment classification task. We take a subset of 20000 reviews that have length greater than 100 words for the purposes of experimentation in this paper.

Yahoo Yahoo! Answers (Zhang et al., 2015) has over 1,400,000 question and answer pairs spread across 10 classes. We do not use information such as question, title, date and location for the purpose of classification. As in the case of Amazon reviews, in the standard setting, we ensure that all answers have lengths between 100 and 1000 words, while in the short sentence setting, the maximum answer length in the filtered dataset is 100 words.

Yelp Reviews Yelp Reviews (Zhang et al., 2015) is a sentiment analysis task with 5 matching classes. For the purposes of experimentation, we create a subset which is filtered to contain sentences in the range 100 to 1000 tokens. Further, all reviews with a score of 4 or 5 are maked positive, while those with a score of 1 or 2 are marked negative for the binary classification task.

For these datasets, we only use the text and labels, ignoring any auxiliary information (like title or location). We select subsets of the datasets with sequences having greater than 100 words to better understand the impact of vanishing gradients and positional bias in recurrent architectures. A summary of statistics is presented in Table 1.1.

Dataset	Classes	Avg. Length	Max Length	Train Size	Test Size
IMDb	2	240.4	2470	20K	9.8K
Yahoo! Answers	10	206.2	998	25K	4.8K
Amazon Reviews	20	185.6	500	25K	12.5K
Yelp Reviews	2	202.4	1000	25K	9.5K

Table 1.1: Corpus statistics for classification tasks.

1.4.2 Experimental Settings and Reproducibility

In all the experiments, we use a single-layered BiLSTM with hidden dimension size of 256 and embedding dimension size of 100 (initialized with GloVe vectors (Pennington et al., 2014) trained on a 6 billion word corpus). The sentence embeddings generated by the BiLSTM are passed to a final classification layer to obtain per-class probability

distributions. We train our models using Adam optimizer (Kingma and Ba, 2014), with a learning rate of 2×10^{-3}. The batch size is set to 32 for all the experiments. We train for 20 epochs and select the model with the best validation accuracy. All experiments are repeated over 5 random seeds using a single GPU (Tesla K40).

Computing Infrastructure For all the experiments described in the paper, we use a Tesla K40 GPUs supporting a maximum of 10GB of GPU memory. All experiments can be performed on a single GPU. The brief experimentation done on transformer models was done using Tesla V100s that support 32 GB of GPU memory.

Run Time The average run-time for each epoch varies linearly with the amount of training data and average sentence length. For the mode with 25K training data in standard setting (sentences with greater than 100 words, and no wikipedia words) the average training time for 1 epoch is under 2 minutes. Further, across all pooling techniques, the run time varies only marginally.

Number of Parameters The number of parameters in the model varies with the vocabulary size. We cap the maximum vocabulary size to 25,000 words. However, in the 1K training data setting, the actual vocabulary size is lesser (depending on the training data). The majority of the parameters of the model are accounted for in the model's embedding matrix $=$ (vocabulary size)$\times$(embedding size). The number of parameters for the main LSTM model are around 70,000, with the ATT model having a few more parameters than other methods due to a learnable query vector.

Validation Scores We provide validation results in Table 1.2 for the standard setting. However, in interest of brevity, we only detail the test scores in all subsequent tables. Note that we always select the model based on the best validation accuracy during the training process (among all the epochs).

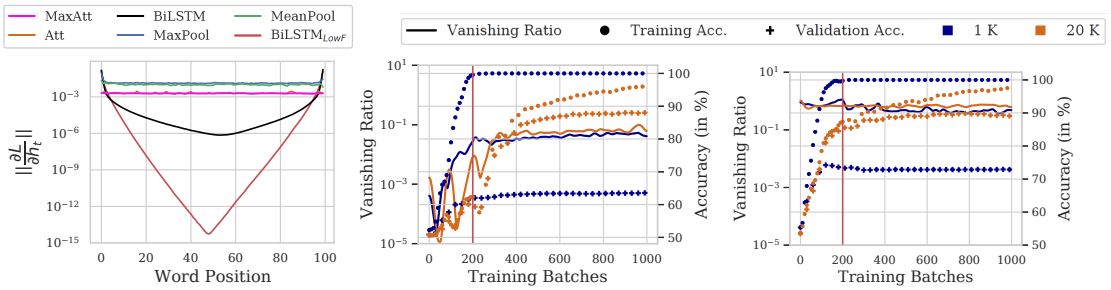
Evaluation Metric The evaluation metric used is the model's accuracy on the test set and is reported as an average over 5 different seeds. All the classes are nearly balanced in the datasets chosen, hence standard accuracy metric serves as an accurate indicator.

Hyperparameters search An explicit hyperparameter search is not performed for each model in each training setting over all seeds, since the purpose of the paper is not to beat the state of art, but rather to analyze the effect of pooling in recurrent architectures. We do note that, in the manual search performed on the learning rates of $\{1 \times 10^{-3}, 2 \times 10^{-3}, 5 \times 10^{-3}\}$ on the IMDb and Yahoo datasets, we find that for all the pooling and non-pooling methods discussed, models trained on learning rate equal to 2×10^{-3} showed the best validation accuracy. Thus, we use that for all the following results. However, we do perform a hyperparameter search for the best regularization parameters as described inthe next paragraph. We keep the embedding dimension and hidden dimension fixed for all experiments.

On using regularization For the experiments in the work, we do not regularize trained LSTMs. This has two analytical advantages (1) we can examine the benefits of pooling without having to account for the the effect of regularization; and (2) training to 100% accuracy acts as an indicator of training the models *adequately*. However, for validation, we also performed our experiments on the IMDb dataset with 2 different types of regularization schemes, following best practices used in previous works (Merity et al., 2017). We use DropConnect (Wan et al., 2013) [1] and Weight Decay [2] for regularization of all the models. We observe that the effect of regularization consistently improves the final accuracies by 1-2% across the board. However, even after sustained training (up to 50 epochs), BiLSTM still suffers from the learning issues outlined in the paper. The goal of this paper is not to study the effect of various regularization schemes, but to merely understand the effect pooling in improving the performance of BiLSTM.

[1] grid search over mask rate: {0.1,0.3,0.5}
[2] grid search over decay value: $\{10^{-3}, 10^{-4}, 10^{-6}, 10^{-8}\}$

1.5 Gradient Propagation



(a) Gradient Norms (b) BiLSTM (c) MAXATT

Figure 1.2: (a): The gradient norm $(\|\frac{\partial L}{\partial h_t}\|)$ across different word positions. BiLSTM$_{\text{LowF}}$ suffers from extreme vanishing of gradients, with the gradient norm in the middle nearly 10^{-15} times that at the ends. In contrast, pooling methods result in gradients of nearly the same value, irrespective of the word position. (b), (c): The vanishing ratio $(\|\frac{\partial L}{\partial h_{\text{mid}}}\|/\|\frac{\partial L}{\partial h_{\text{end}}}\|)$ over training batches for BiLSTM and MAXATT, using 1K, 20K unique training examples from the IMDb dataset. The respective training and validation accuracies are also depicted.

(a) ATT (b) MAXPOOL (c) MEANPOOL

Figure 1.3: The vanishing ratio $(\|\frac{\partial L}{\partial h_{\text{end}}}\|/\|\frac{\partial L}{\partial h_{\text{mid}}}\|)$ over training steps for ATT, MAXPOOL, MEANPOOL using 1K, 20K training examples from the IMDb dataset. The respective training and validation accuracies are also depicted.

In this section, we study the flow of gradients in different architectures and training regimes. Pooling techniques used in conjunction with BiLSTMs provide a direct gradient pathway to intermediate hidden states. However for BiLSTMs without pooling, it is crucial that the parameters for the input, output, and forget gates are appropriately learned so that the loss backpropagates across long input sequences, without the gradients vanishing.

Experimental Setup: In order to quantify the extent to which the gradients vanish across different word positions, we compute the gradient of the loss function w.r.t the hidden state at every word position t, and study their ℓ_2 norm ($\|\frac{\partial L}{\partial h_t}\|$). To aggregate the gradients across multiple training examples (of different lengths), we linearly interpolate the distribution of gradient values for each example to a fixed length between 1 and 100. The gradient values at each (normalized) position are averaged across all the training examples. We plot these values (on a $\log$ scale) after training on the first 500 IMDb reviews to study the effect of gradient vanishing at the beginning of training (Figure 1.2a).

To understand how the distribution of gradients (across word positions) changes with the number of training batches, we compute the ratio of the gradient norm corresponding to the word at the middle and word at the end: $\|\frac{\partial L}{\partial h_{\text{mid}}}\| \ / \ \|\frac{\partial L}{\partial h_{\text{end}}}\|$.[3] We call this the *vanishing ratio* and use it as a measure to quantify the extent of vanishing (where lower values indicate severe vanishing). Each training batch on the x-axis in Figures 1.2b, 1.2c corresponds to 64 training examples.

Results It is evident from Figure 1.2a that the gradients vanish significantly for BiLSTM, with $\|\frac{\partial L}{\partial h_t}\|$ falling to the order of 10^{-6} as we approach the middle positions in the sequence. This effect is even more pronounced for the case of $\text{BiLSTM}_{\text{LowF}}$, which uses the Xavier initialization (Glorot and Bengio, 2010) for the bias of the forget-gate. The plot suggests that specific initialization of the gates with best practices (such as setting the bias of forget-gate to a high value) helps to reduce the extent of the issue, but the problem still persists. In contrast, none of the pooling techniques face this issue, resulting in an almost straight line.

Additionally, from Figure 1.2b we note that the problem of vanishing gradients is more pronounced at the beginning of training, when the gates are still untrained. The problem continues to persist, albeit to a lesser degree, until later in the training process. This specifically limits the performance of BiLSTM in resource-constrained settings, with fewer training examples. For instance, in the 1K training data setting, BiLSTM

[3]Implementation detail: we choose the left end, as some sequences in a batch might be padded with zeros on the right.

	Vanishing ratio			Validation acc.		
	1K	5K	20K	1K	5K	20K
BiLSTM	3×10^{-4}	0.03	0.06	64.9	82.8	88.4
MEANPOOL	1.09	0.84	1.02	78.4	82.6	88.5
MAXPOOL	0.43	0.46	0.50	78.0	84.7	89.6
ATT	0.58	0.61	0.91	77.1	84.6	90.0
MAXATT	0.68	0.56	0.72	78.1	86.0	90.2

Table 1.2: Values of vanishing ratio as computed when different models achieve 95% training accuracy, along with the best validation accuracy for that run.

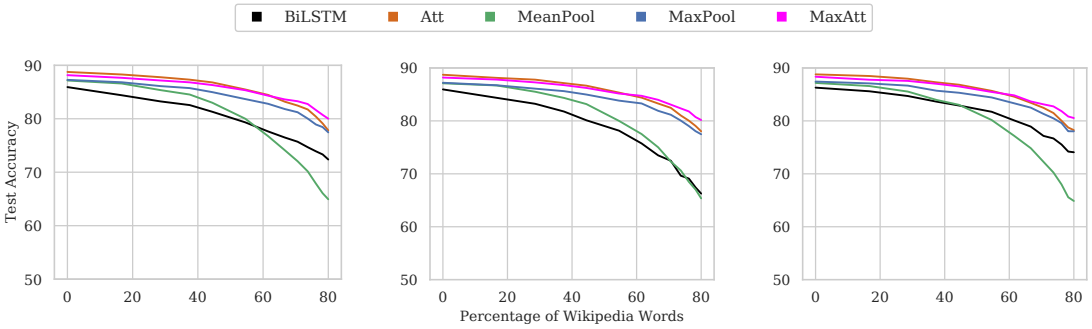
has an extremely low value of vanishing ratio ($\sim 10^{-3}$) at the 200^{th} training batch (denoted by red vertical line in the plot), when it achieves nearly 100% accuracy on the training data.

The plots of the change in vanishing ratios for ATT, MAXPOOL and MEANPOOL are shown in Figure 1.3. This completes the representative analysis for BiLSTM and MAXATT shown in Figure 1.2. It can be seen that for all the different pooling types discussed in this paper, the vanishing ratios are small right from the beginning of training. This motivates future research to further formally analyze and discover other learning advantages (apart from vanishing ratios) that distinguish the performance of one pooling technique from the other.

Consequently, the BiLSTM model (prematurely) achieves a high training accuracy, solely based on the starting and ending few words, well before the gates can learn to allow the gradients to pass through (and mitigate the vanishing gradients problem). Further reduction in vanishing ratio is unable to improve validation accuracy, due to saturation in training. To examine this more closely, we tabulate the vanishing ratios at the point where the model reaches 95% accuracy on the training data in Table 1.2. A low value at this point indicates that the gradients are still skewed towards the ends, even as the model begins to overfit on the training data. The vanishing ratio is low for BiLSTM, especially in low-data settings. This results in a 13-14% lower test accuracy in the 1K data setting, compared to other pooling techniques. We conclude that the phenomenon of vanishing gradients results in poorer performance of BiLSTMs. Encouragingly, pooling methods do not exhibit low vanishing ratios, right from the beginning of training,

leading to performance gains as demonstrated in the next section.

1.6 Positional Biases



(a) Left (b) Mid (c) Right

Figure 1.4: For models trained on 10K examples, varying amounts of random Wikipedia sentences are appended to the original IMDb reviews *at test time*. Original review is preserved on the (a) left; (b) middle; and (c) right of the modified input. Performance degrades significantly for BiLSTM and MEANPOOL, whereas ATT, MAXPOOL and MAXATT are more resilient.

Analyzing the gradient propagation in BiLSTMs suggests that standard recurrent networks are biased towards the end tokens, as the overall contribution of distant hidden states is extremely low in the gradient of the loss. This implies that the weights of various parameters in an LSTM cell (all cells of an LSTM have tied weights) are hardly influenced by the middle words of the sentence. In this light, we aim to evaluate positional biases of recurrent architectures with different pooling techniques.

1.7 Evaluating Natural Positional Biases

Can organically trained recurrent models skip over unimportant words on either ends of the sentence?

	IMDb			IMDb (mid) + Wiki			IMDb (right) + Wiki		
	1K	2K	10K	1K	2K	10K	1K	2K	10K
BiLSTM	64.7 ± 2.3	75.0 ± 0.4	86.6 ± 0.8	49.6 ± 0.7	49.9 ± 0.5	50.3 ± 0.3	53.5 ± 2.5	64.7 ± 2.8	85.9 ± 0.5
MEANPOOL	73.0 ± 3.0	81.7 ± 0.7	87.1 ± 0.6	69.8 ± 2.1	76.2 ± 1.0	84.1 ± 0.7	70.0 ± 1.1	76.8 ± 1.0	84.8 ± 0.9
MAXPOOL	69.0 ± 3.9	80.1 ± 0.5	87.8 ± 0.6	64.5 ± 1.8	77.2 ± 2.0	86.0 ± 0.8	65.9 ± 4.6	77.8 ± 0.9	**87.2** ± 0.6
ATT	75.7 ± 2.6	**82.8** ± 0.8	**89.0** ± 0.3	75.0 ± 0.8	79.4 ± 0.8	86.7 ± 1.4	74.7 ± 1.4	80.2 ± 1.8	87.1 ± 1.0
MAXATT	**75.9** ± 2.2	82.5 ± 0.4	88.5 ± 0.5	**75.4** ± 2.4	**80.9** ± 1.8	**86.8** ± 0.5	**77.9** ± 0.9	**81.9** ± 0.5	**87.2** ± 0.5
	Yahoo			Yahoo (mid) + Wiki			Yahoo (right) + Wiki		
	1K	2K	10K	1K	2K	10K	1K	2K	10K
BiLSTM	38.3 ± 4.8	51.4 ± 2.1	63.5 ± 0.6	12.7 ± 1.1	12.7 ± 1.1	11.4 ± 0.8	18.8 ± 2.5	37.3 ± 0.9	60.1 ± 1.5
MEANPOOL	48.2 ± 2.3	56.6 ± 0.5	64.7 ± 0.6	31.9 ± 2.3	43.1 ± 2.0	58.5 ± 0.6	33.9 ± 2.1	43.2 ± 1.0	58.6 ± 0.4
MAXPOOL	50.2 ± 2.1	56.3 ± 1.8	63.9 ± 1.1	33.0 ± 1.0	40.1 ± 1.4	58.4 ± 1.2	33.1 ± 2.5	41.2 ± 0.9	60.9 ± 1.0
ATT	47.3 ± 2.2	54.2 ± 1.1	**65.1** ± 1.5	39.4 ± 0.5	45.1 ± 1.8	61.5 ± 1.7	37.9 ± 1.4	47.6 ± 2.3	62.2 ± 0.9
MAXATT	**51.8** ± 1.1	**57.0** ± 1.1	**65.1** ± 1.1	**39.6** ± 0.9	**48.5** ± 0.6	**62.2** ± 1.6	**40.3** ± 1.5	**50.1** ± 1.6	**63.1** ± 0.7
	Amazon			Amazon (mid) + Wiki			Amazon (right) + Wiki		
	1K	2K	10K	1K	2K	10K	1K	2K	10K
BiLSTM	38.5 ± 4.2	52.7 ± 7.7	76.2 ± 0.7	5.3 ± 0.3	5.4 ± 0.3	5.1 ± 0.4	7.9 ± 0.6	27.9 ± 9.9	70.8 ± 1.5
MEANPOOL	44.8 ± 9.8	55.6 ± 6.4	76.9 ± 0.4	34.4 ± 3.5	52.7 ± 3.5	70.3 ± 1.7	33.3 ± 1.0	48.2 ± 3.4	71.9 ± 0.8
MAXPOOL	49.6 ± 3.9	61.6 ± 2.6	**79.1** ± 0.4	17.0 ± 0.7	34.5 ± 2.0	72.8 ± 0.6	17.0 ± 1.7	36.5 ± 3.0	72.4 ± 0.3
ATT	54.1 ± 5.2	61.2 ± 2.9	77.0 ± 0.3	48.0 ± 1.7	59.1 ± 1.8	**75.3** ± 0.5	48.9 ± 1.5	58.9 ± 1.3	**75.7** ± 0.3
MAXATT	**58.2** ± 3.8	**65.6** ± 0.9	77.3 ± 0.2	**57.7** ± 0.5	**63.0** ± 0.8	74.8 ± 0.5	**57.8** ± 0.8	**63.7** ± 0.8	75.3 ± 0.3
	Yelp			Yelp (mid) + Wiki			Yelp (right) + Wiki		
	1K	2K	10K	1K	2K	10K	1K	2K	10K
BiLSTM	80.7 ± 4.1	84.9 ± 8.0	93.1 ± 0.1	50.2 ± 0.4	51.1 ± 0.9	51.4 ± 0.7	59.4 ± 3.7	79.6 ± 6.2	92.7 ± 0.4
MEANPOOL	**87.1** ± 1.2	**87.9** ± 1.7	93.4 ± 0.3	79.2 ± 1.1	86.7 ± 1.0	92.7 ± 0.2	79.4 ± 0.9	87.1 ± 0.6	92.3 ± 0.4
MAXPOOL	84.4 ± 2.0	86.4 ± 5.1	93.4 ± 0.2	81.1 ± 1.5	85.6 ± 0.6	92.5 ± 0.4	80.6 ± 0.8	86.7 ± 0.9	**93.2** ± 0.2
ATT	82.5 ± 3.7	85.6 ± 6.5	**93.7** ± 0.2	84.4 ± 1.0	89.3 ± 1.0	92.5 ± 0.6	**84.8** ± 0.7	89.1 ± 0.9	92.8 ± 0.4
MAXATT	81.3 ± 5.1	86.0 ± 6.3	**93.7** ± 0.3	**85.1** ± 0.8	**89.4** ± 0.5	**92.9** ± 0.3	84.1 ± 2.5	**89.5** ± 0.7	93.0 ± 0.4

Table 1.3: Mean test accuracy (± std) (in %) across 5 random seeds. In low-resource settings, MAXATT consistently outpeforms other pooling variants. The performance of different pooling methods converges with increase in data.

1.7.1 Experimental Setup

We append randomly chosen Wikipedia sentences to the input examples of two text classification tasks, based on IMDb and Amazon Reviews, *only at test time*, keeping the training datasets unchanged. Wikipedia sentences are declarative statements of fact, and should not influence the sentiment of movie reviews, and given the diverse nature of the Wikipedia sentences it is unlikely that they would interfere with the few categories (i.e. the labels) of Amazon product reviews. Therefore, it is not unreasonable to expect the models to be robust to such random noise, even though they were not trained for the same. We perform this experiment in three configurations, such that original input is preserved on the (a) left, (b) middle, and (c) right of the modified input. For these configurations, we vary the length of added Wikipedia text in proportion to the length of the original sentence. Figure 1.5 illustrates the setup when 66% of the total words

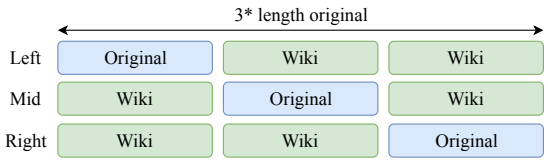


Figure 1.5: Explaining Wikipedia sentence addition.

come from Wikipedia.

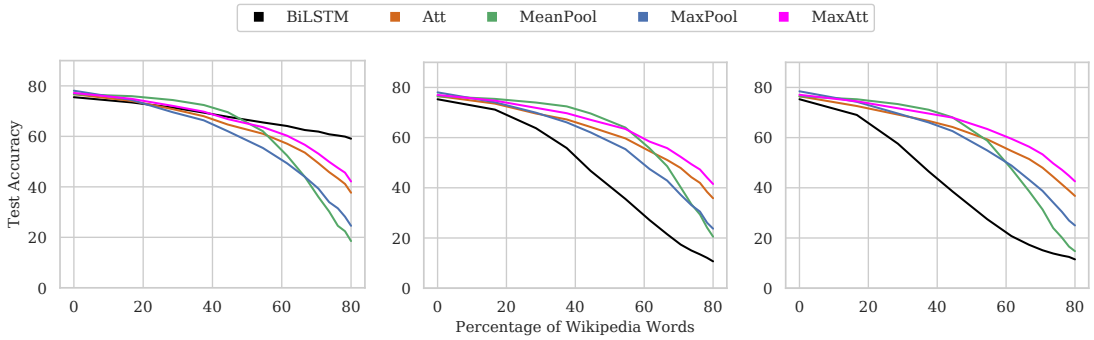
1.7.2 Results

The effect of adding random words can be seen in Figure 1.4. We draw two conclusions: (1) Adding random sentences on both ends is more detrimental to the performance of BiLSTM as compared to the scenario where the input is appended to only one end.[4] This corroborates our previous findings that these models largely account for information at the ends for their predictions. (2) We speculate that paying equal importance to all hidden states prevents MEANPOOL from distilling out important information effectively, making it more susceptible to random noise addition. On the contrary, both max-pooling and attention based architectures like MAXPOOL, ATT and MAXATT are significantly more robust in all the settings. This indicates that max-pooling and attention can help account for salient words and ignore unrelated ones, regardless of their position. Lastly, we provide concurring results on the Amazon dataset, and examine the robustness of different models given lesser training data in § 1.7.3.

1.7.3 Deeper dive into Natural Positional Biases

In line with our results in § 1.7, we further evaluate models trained on the Amazon dataset in the same settings to re-validate our results. The effect of appending random Wikipedia sentences to input examples on models trained on the Amazon dataset can be found in Figure 1.6. We use the model trained on 10K examples to perform this experiment. The graphs show similar findings as in Figure 1.4, and further supports the

[4]One practical implication of this finding is that adversaries can easily attack middle portions of the input text.



(a) Left (b) Mid (c) Right

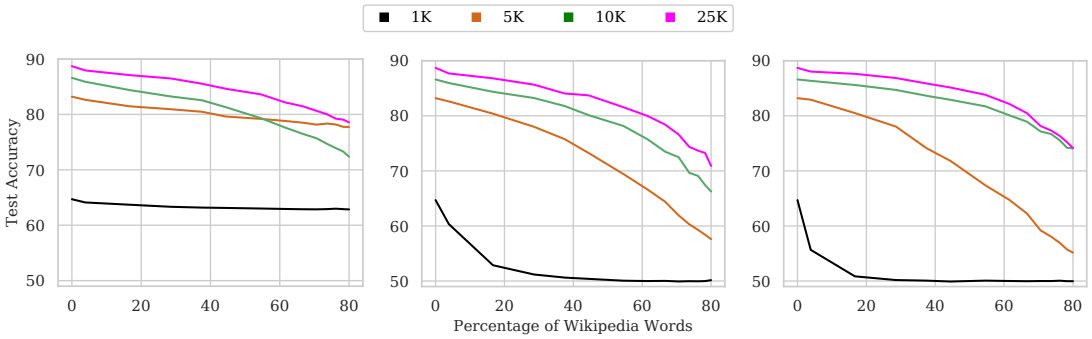
Figure 1.6: Amazon Dataset (10K setting): Random Wikipedia sentences are appended to the original input paragraphs. Original input is preserved on the (a) left, (b) middle, and (c) right of the new input. Test accuracies are reported by varying the percentage of total Wikipedia words in the new input.

hypothesis that BiLSTM gives a strong emphasis on extreme words when trained on standard datasets, which is why its performance significantly deteriorates when random Wikipedia sentences are appended on both ends.

1.7.4 Effect of Amount of Training Data

Figure 1.4 suggests that BiLSTM is equally responsive to the effect of appending random words to the left or right. However, in case of the Amazon Reviews dataset (Figure 1.6), we notice that the BiLSTM is more resilient when the text to the left is preserved. This indicates a learning bias, where the BiLSTM pays greater emphasis to outputs of one chain of the bidirectional LSTM. It is interesting to note that on reducing the training data, this bias increases significantly in the case of IMDb dataset as well.

We hypothesize that such a phenomenon may have resulted due to an artifact of the training process itself, that is, the model is able to find 'easily identifiable' important sentiment at the beginning of the reviews during training (speculatively due to the added effects of padding to the right). Therefore, given less training data, BiLSTMs prematurely learn to use features from only one of the two LSTM chains and (in this case) the left $\rightarrow$ right chain of the dominates the final prediction. We confirm from Figure 1.7 that with a decrease in training data (such as in the 1K IMDb data setting), the bias



(a) Left (b) Mid (c) Right

Figure 1.7: IMDb Dataset (BiLSTM): Random Wikipedia sentences are appended to the original input paragraphs for the standard BiLSTM models trained on 1K, 5K, 10K and 20K examples. Original input is preserved on the (a) left, (b) middle, and (c) right of the new input. Test accuracies are reported by varying the percentage of total Wikipedia words in the new input. BiLSTM is unrepsonsive to any appended tokens as long as the 'left' text is preserved in the 1K and 5K setting. But this bias dilutes with more training samples. Given sufficient data (more than 10K unique examples) the effect of appending random words on both ends is more detrimental than that on appending at only one end.

towards one end substantially increases, that is, BiLSTM is extremely insensitive to random sentence addition, as long as the left end is preserved.

1.7.5 Practical Implications

We observe that MEANPOOL and BiLSTM can be susceptible to *changes in test-time data distribution*. This questions the use of such models in real word settings. We speculate that paying equal importance to all hidden states handicaps MEANPOOL from being able to distil out important information effectively, while the preceding discussion on the effect of size of training data highlights the possible cause of this occurrence in BiLSTM. We observe that other pooling methods like MAXATT are able to circumvent this issue as they are only mildly affected by the added Wikipedia sentences.

1.8 Training to Skip Unimportant Words

How well can different models be trained to skip unrelated words?

1.8.1 Experimental setup

We create new training datasets by appending random Wikipedia sentences to the original input examples of the datasets described in § 1.4, such that 66% of the text of each new training example comes from Wikipedia sentences (see Figure 1.5). We experiment with a varying number of training examples, however, the test set remains the same for fair comparisons.

1.8.2 Results

The results are presented in Table 1.3. First, we note that BiLSTM severely suffers when random sentences are appended at both ends. In fact, the accuracy of BiLSTM in mid settings drops to 50%, 12%, 5%, 50% on IMDb, Yahoo, Amazon, Yelp datasets respectively, which is equal to the majority class baseline. However, the performance drop (while large) is not as drastic when sentences are added to only one end of the text. We speculate that this is because a BiLSTM is composed of a forward and a backward LSTM, and when random sentences are appended to the left, the backward LSTM is able to capture information about the original sentence on the right and vice versa.

Second, while accuracies of all pooling techniques begin to converge given sufficient data, the differences in low training data regime are substantial. Further, the poor performance of BiLSTM re-validates the findings of § 1.5, where we hypothesize that the model's training saturates before the gradients can learn to reach the middle tokens.[5]

Evaluation on Short Sentences Finally, we re-evaluate this experiment on (new) datasets with short sentences (< 100 words). Results for the standard and 'mid' set-

[5]Results on more dataset sizes, and the 'left' setting are in § 1.8.4. Conclusions drawn from the 'right' setting are in line with the observations from the 'left'.

tings are presented in Table 1.4. Unlike long sequences, where BiLSTM model was no better than majority classifier (see Table 1.3), with shorter sequences, the BiLSTM model performs better. This result supports our hypothesis that the effect of vanishing gradients is prominent in longer sequences.[6] Overall, among all the scenarios discussed in tables 1.3 and 1.4, on comparing all pooling methods (and BiLSTM) on the basis of their mean test accuracy, **MAXATT emerges as the best performing model in about 80% cases, ATT in 18% cases.**

	Datasets with Short Sentences			
	Yahoo		Yahoo (mid) + Wiki	
	1K	10K	1K	10K
BiLSTM	20.5 ± 2.9	42.4 ± 0.2	9.9 ± 0.7	24.2 ± 0.9
MEANPOOL	23.1 ± 1.8	43.0 ± 0.3	14.9 ± 2.2	32.8 ± 0.8
MAXPOOL	23.0 ± 2.8	**43.3** ± 0.4	14.1 ± 2.6	33.8 ± 1.2
ATT	24.3 ± 1.1	43.1 ± 0.2	16.9 ± 3.0	37.6 ± 0.5
MAXATT	**25.1** ± 2.2	**43.3** ± 0.3	**18.2** ± 2.4	**37.8** ± 0.8
	Amazon		Amazon (Mid) + Wiki	
	1K	10K	1K	10K
BiLSTM	26.6 ± 4.4	54.0 ± 2.6	5.6 ± 0.4	37.9 ± 0.9
MEANPOOL	29.4 ± 4.0	54.4 ± 2.6	10.8 ± 1.9	46.5 ± 0.5
MAXPOOL	33.5 ± 4.5	55.9 ± 2.0	10.6 ± 1.8	47.0 ± 0.9
ATT	36.4 ± 3.7	55.6 ± 0.6	17.4 ± 3.2	**49.7** ± 0.3
MAXATT	**37.4** ± 3.8	**56.2** ± 0.8	**17.8** ± 4.6	**49.7** ± 0.5

Table 1.4: Mean test accuracy (± std) (in %) on standard, 'mid' settings across 5 random seeds on Yahoo, Amazon datasets with **short** sentences (< 100 words).

[6]Refer to § 1.8.5 for full evaluation.

Datasets with Long Sentences										
	Yahoo Dataset					Amazon Dataset				
	1K	2K	5K	10K	25K	1K	2K	5K	10K	25K
BiLSTM	38.3 ± 4.8	51.4 ± 2.1	57.4 ± 0.6	63.5 ± 0.6	67.5 ± 0.8	38.5 ± 4.2	52.7 ± 7.7	70.0 ± 0.9	76.2 ± 0.7	81.8 ± 0.3
MEANPOOL	48.2 ± 2.3	56.6 ± 0.5	60.8 ± 0.5	64.7 ± 0.6	68.7 ± 0.6	44.8 ± 9.8	55.6 ± 6.4	71.2 ± 0.9	76.9 ± 0.4	82.0 ± 0.3
MAXPOOL	50.2 ± 2.1	56.3 ± 1.8	61.3 ± 0.9	63.9 ± 1.1	67.0 ± 1.1	49.6 ± 3.9	61.6 ± 2.6	73.9 ± 0.2	79.1 ± 0.4	84.2 ± 0.2
ATT	47.3 ± 2.2	54.2 ± 1.1	61.0 ± 0.5	65.1 ± 1.5	68.2 ± 0.7	54.1 ± 5.2	61.2 ± 2.9	72.0 ± 0.2	77.0 ± 0.3	82.6 ± 0.1
MAXATT	51.8 ± 1.1	57.0 ± 1.1	63.2 ± 0.4	65.1 ± 1.1	68.4 ± 0.6	58.2 ± 3.8	65.6 ± 0.9	72.8 ± 0.5	77.3 ± 0.2	82.4 ± 0.2
	Yahoo (left) + Wiki					Amazon (left) + Wiki				
	1K	2K	5K	10K	25K	1K	2K	5K	10K	25K
BiLSTM	41.4 ± 2.9	51.0 ± 0.5	56.2 ± 1.2	60.9 ± 0.7	64.6 ± 2.0	44.9 ± 0.7	57.0 ± 0.8	68.3 ± 0.9	73.5 ± 0.4	79.6 ± 0.2
MEANPOOL	31.9 ± 1.5	43.3 ± 1.7	51.4 ± 0.9	58.8 ± 0.7	65.1 ± 0.3	31.0 ± 2.1	48.1 ± 1.4	65.0 ± 1.4	70.8 ± 1.2	79.1 ± 0.8
MAXPOOL	33.6 ± 0.9	42.3 ± 1.4	52.7 ± 2.0	60.7 ± 0.9	66.0 ± 1.0	19.2 ± 1.9	42.5 ± 3.5	68.5 ± 2.6	76.8 ± 0.6	82.1 ± 0.5
ATT	37.3 ± 0.5	47.2 ± 2.2	57.6 ± 1.6	62.5 ± 1.0	67.6 ± 0.3	47.6 ± 2.0	59.3 ± 1.1	70.8 ± 0.9	75.6 ± 0.3	81.3 ± 0.3
MAXATT	40.0 ± 0.6	48.7 ± 0.5	59.6 ± 1.4	63.0 ± 1.4	67.2 ± 0.9	56.1 ± 1.3	63.8 ± 1.3	70.3 ± 0.3	75.6 ± 0.2	80.7 ± 0.5
	Yahoo (mid) + Wiki					Amazon (mid) + Wiki				
	1K	2K	5K	10K	25K	1K	2K	5K	10K	25K
BiLSTM	12.7 ± 1.1	12.7 ± 1.1	12.0 ± 0.9	11.4 ± 0.8	13.2 ± 2.2	5.3 ± 0.3	5.4 ± 0.3	5.0 ± 0.1	5.1 ± 0.4	7.8 ± 5.2
MEANPOOL	31.9 ± 2.3	43.1 ± 2.0	50.1 ± 1.6	58.5 ± 0.6	64.9 ± 0.7	34.4 ± 3.5	52.7 ± 3.5	63.4 ± 2.0	70.3 ± 1.7	79.0 ± 0.6
MAXPOOL	33.0 ± 1.0	40.1 ± 1.4	51.0 ± 1.2	58.4 ± 1.2	65.5 ± 0.7	17.0 ± 0.7	34.5 ± 2.0	58.8 ± 0.4	72.8 ± 0.6	80.4 ± 0.3
ATT	39.4 ± 0.5	45.1 ± 1.8	57.0 ± 2.0	61.5 ± 1.7	66.5 ± 0.6	48.0 ± 1.7	59.1 ± 1.8	69.5 ± 0.6	75.3 ± 0.5	81.1 ± 0.2
MAXATT	39.6 ± 0.9	48.5 ± 0.6	58.7 ± 1.5	62.2 ± 1.6	66.5 ± 0.7	57.7 ± 0.5	63.0 ± 0.8	69.8 ± 0.6	74.8 ± 0.5	80.3 ± 0.4
	Yahoo (right) + Wiki					Amazon (right) + Wiki				
	1K	2K	5K	10K	25K	1K	2K	5K	10K	25K
BiLSTM	18.8 ± 2.5	37.3 ± 0.9	52.9 ± 2.1	60.1 ± 1.5	65.4 ± 0.6	7.9 ± 0.6	27.9 ± 9.9	45.8 ± 16.2	70.8 ± 1.5	78.7 ± 0.8
MEANPOOL	33.9 ± 2.1	43.2 ± 1.0	50.6 ± 0.8	58.6 ± 0.4	64.6 ± 0.5	33.3 ± 1.0	48.2 ± 3.4	64.1 ± 0.7	71.9 ± 0.8	78.8 ± 0.2
MAXPOOL	33.1 ± 2.5	41.2 ± 0.9	53.0 ± 3.6	60.9 ± 1.0	66.0 ± 0.7	17.0 ± 1.7	36.5 ± 3.0	64.3 ± 1.5	72.4 ± 0.3	80.2 ± 0.9
ATT	37.9 ± 1.4	47.6 ± 2.3	58.1 ± 1.4	62.2 ± 0.9	67.0 ± 0.3	48.9 ± 1.5	58.9 ± 1.3	69.7 ± 0.6	75.7 ± 0.3	81.1 ± 0.3
MAXATT	40.3 ± 1.5	50.1 ± 1.6	59.3 ± 1.2	63.1 ± 0.7	66.8 ± 0.3	57.8 ± 0.8	63.7 ± 0.8	71.1 ± 0.6	75.3 ± 0.3	80.7 ± 0.5

Table 1.5: Mean test accuracy (± std) (in %) on different manipulated settings across 5 random seeds on the Yahoo, Amazon datasets with long sentences (greater than 100 words).

1.8.3 Further Analyses

We demonstrate in § 1.8 that the ability of BiLSTM, and its different pooling variants, to learn to skip unrelated words can be greatly diminished in challenging datasets especially given less amount of input data. In this section, we aim to (a) provide a complete evaluation on all positions of data modification and dataset size settings (including those which were skipped in the main paper for brevity); (b) evaluate the same experiment in a setting where input examples are shorter in length.

1.8.4 Full Evaluation

For completeness, we perform the evaluation in § 1.8 on each of {1K, 2K, 5K, 10K, 25K} dataset size settings, and also report the results when Wikipedia words are ap-

Datasets with Long Sentences										
	IMDb Dataset					Yelp Dataset				
	1K	2K	5K	10K	20K	1K	2K	5K	10K	25K
BiLSTM	64.7 ± 2.3	75.0 ± 0.4	83.2 ± 0.4	86.6 ± 0.8	88.7 ± 0.6	80.7 ± 4.1	84.9 ± 8.0	92.2 ± 0.3	93.1 ± 0.1	94.1 ± 0.3
MEANPOOL	73.0 ± 3.0	81.7 ± 0.7	85.4 ± 0.1	87.1 ± 0.6	88.6 ± 0.3	87.1 ± 1.2	87.9 ± 1.7	92.2 ± 0.4	93.4 ± 0.3	94.4 ± 0.1
MAXPOOL	69.0 ± 3.9	80.1 ± 0.5	85.7 ± 0.2	87.8 ± 0.6	89.9 ± 0.3	84.4 ± 2.0	86.4 ± 5.1	92.2 ± 0.3	93.4 ± 0.2	94.7 ± 0.2
ATT	75.7 ± 2.6	82.8 ± 0.8	86.9 ± 0.7	89.0 ± 0.3	90.3 ± 0.2	82.5 ± 3.7	85.6 ± 6.5	92.6 ± 0.4	93.7 ± 0.2	94.9 ± 0.1
MAXATT	75.9 ± 2.2	82.5 ± 0.4	86.1 ± 0.8	88.5 ± 0.5	89.9 ± 0.2	81.3 ± 5.1	86.0 ± 6.3	92.6 ± 0.2	93.7 ± 0.3	94.8 ± 0.1
	IMDb (left) + Wiki					Yelp (left) + Wiki				
	1K	2K	5K	10K	20K	1K	2K	5K	10K	25K
BiLSTM	67.6 ± 1.1	74.7 ± 1.2	80.6 ± 0.3	84.5 ± 0.4	87.2 ± 0.4	81.7 ± 0.5	87.5 ± 0.5	90.7 ± 0.5	92.0 ± 0.3	93.8 ± 0.2
MEANPOOL	69.7 ± 3.4	76.6 ± 0.9	81.7 ± 0.7	83.6 ± 1.0	86.5 ± 0.8	78.1 ± 1.3	87.0 ± 1.1	90.9 ± 0.3	92.5 ± 0.1	93.8 ± 0.2
MAXPOOL	68.8 ± 1.2	76.8 ± 1.7	82.2 ± 0.8	86.9 ± 0.9	88.4 ± 0.5	80.2 ± 1.5	87.5 ± 1.0	91.4 ± 0.2	93.0 ± 0.4	94.3 ± 0.1
ATT	76.5 ± 1.5	79.6 ± 1.1	82.6 ± 0.6	86.9 ± 0.8	88.9 ± 0.5	84.7 ± 1.6	89.5 ± 0.7	92.0 ± 0.2	92.9 ± 0.4	94.4 ± 0.2
MAXATT	75.8 ± 1.5	80.6 ± 1.0	84.1 ± 1.5	87.1 ± 0.6	89.1 ± 0.2	84.7 ± 1.3	89.7 ± 0.6	92.1 ± 0.1	93.1 ± 0.4	94.2 ± 0.4
	IMDb (mid) + Wiki					Yelp (mid) + Wiki				
	1K	2K	5K	10K	20K	1K	2K	5K	10K	25K
BiLSTM	49.6 ± 0.7	49.9 ± 0.5	50.2 ± 0.3	50.3 ± 0.3	50.1 ± 0.3	50.2 ± 0.4	51.1 ± 0.9	51.2 ± 0.8	51.4 ± 0.7	51.5 ± 0.5
MEANPOOL	69.8 ± 2.1	76.2 ± 1.0	82.2 ± 0.7	84.1 ± 0.7	86.5 ± 0.8	79.2 ± 1.1	86.7 ± 1.0	90.7 ± 0.3	92.7 ± 0.2	94.0 ± 0.1
MAXPOOL	64.5 ± 1.8	77.2 ± 2.0	82.9 ± 1.2	86.0 ± 0.8	88.4 ± 0.6	81.1 ± 1.5	85.6 ± 0.6	90.7 ± 0.4	92.5 ± 0.4	94.1 ± 0.2
ATT	75.0 ± 0.8	79.4 ± 0.8	83.4 ± 1.0	86.7 ± 1.4	88.8 ± 0.2	84.4 ± 1.0	89.3 ± 1.0	91.8 ± 0.6	92.5 ± 0.6	94.4 ± 0.2
MAXATT	75.4 ± 2.4	80.9 ± 1.8	84.7 ± 1.3	86.8 ± 0.5	88.7 ± 0.4	85.1 ± 0.8	89.4 ± 0.5	91.7 ± 0.7	92.9 ± 0.3	94.3 ± 0.2
	IMDb (right) + Wiki					Yelp (right) + Wiki				
	1K	2K	5K	10K	20K	1K	2K	5K	10K	25K
BiLSTM	53.5 ± 2.5	64.7 ± 2.8	79.7 ± 4.3	85.9 ± 0.5	88.5 ± 0.2	59.4 ± 3.7	79.6 ± 6.2	91.7 ± 0.3	92.7 ± 0.4	93.7 ± 0.4
MEANPOOL	70.0 ± 1.1	76.8 ± 1.0	81.8 ± 0.5	84.8 ± 0.9	87.1 ± 0.3	79.4 ± 0.9	87.1 ± 0.6	90.9 ± 0.7	92.3 ± 0.4	93.8 ± 0.3
MAXPOOL	65.9 ± 4.6	77.8 ± 0.9	84.9 ± 0.8	87.2 ± 0.6	89.3 ± 0.3	80.6 ± 0.8	86.7 ± 0.9	91.9 ± 0.5	93.2 ± 0.2	94.5 ± 0.3
ATT	74.7 ± 1.4	80.2 ± 1.8	84.7 ± 1.1	87.1 ± 1.0	89.4 ± 0.3	84.8 ± 0.7	89.1 ± 0.9	92.0 ± 0.4	92.8 ± 0.4	94.3 ± 0.1
MAXATT	77.9 ± 0.9	81.9 ± 0.5	85.2 ± 0.8	87.2 ± 0.5	89.4 ± 0.3	84.1 ± 2.5	89.5 ± 0.7	91.7 ± 0.9	93.0 ± 0.4	94.3 ± 0.1

Table 1.6: Mean test accuracy (± std) (in %) on different manipulated settings across 5 random seeds on the IMDb, Yelp Reviews datasets with long sentences (less than 100 words).

pended on the right, preserving the original input to the left. We report results for the Yahoo and Amazon datasets in Table 1.5 and the IMDb and Yelp Reviews datasets in Table 1.6. It can be noted that the advantages of MAXATT over other pooling and non-pooling techniques significantly increase in the three Wikipedia settings in each of the tables. This suggests that MAXATT performs better in more challenging scenarios where the important signals are hidden in the input data. Further, the performance advantages of MAXATT are more when amount of training data is less.

1.8.5 Short Sentences

Dataset	Classes	Avg. Length	Max Length	Train Size	Test Size
Yahoo! Answers	10	30.1	95	25K	25K
Amazon Reviews	20	29.1	100	25K	12.5K

Table 1.7: Corpus statistics for classification tasks (short datasets).

Datasets with Short Sentences										
	Yahoo Dataset					Amazon Dataset				
	1K	2K	5K	10K	25K	1K	2K	5K	10K	25K
BiLSTM	20.5 ± 2.9	25.8 ± 3.7	33.1 ± 2.4	42.4 ± 0.2	46.0 ± 0.4	26.6 ± 4.4	37.7 ± 3.4	48.6 ± 2.2	54.0 ± 2.6	61.7 ± 0.2
MEANPOOL	23.1 ± 1.8	28.4 ± 1.5	35.3 ± 1.8	43.0 ± 0.3	46.5 ± 0.4	29.4 ± 4.0	38.2 ± 3.4	49.0 ± 1.8	54.4 ± 2.6	62.0 ± 0.2
MAXPOOL	23.0 ± 2.8	31.2 ± 1.4	37.3 ± 1.9	43.3 ± 0.4	46.8 ± 0.8	33.5 ± 4.5	41.4 ± 3.3	50.8 ± 1.7	55.9 ± 2.0	62.8 ± 0.1
ATT	24.3 ± 1.1	30.7 ± 2.5	36.3 ± 2.0	43.1 ± 0.2	46.4 ± 0.6	36.4 ± 3.7	43.3 ± 1.7	50.9 ± 0.6	55.6 ± 0.6	61.9 ± 0.2
MAXATT	25.1 ± 2.2	30.8 ± 2.6	37.9 ± 1.1	43.3 ± 0.3	46.8 ± 0.7	37.4 ± 3.8	44.6 ± 1.2	51.6 ± 0.8	56.2 ± 0.8	62.4 ± 0.4
	Yahoo (left) + Wiki					Amazon (left) + Wiki				
	1K	2K	5K	10K	25K	1K	2K	5K	10K	25K
BiLSTM	19.6 ± 1.7	28.5 ± 0.8	34.5 ± 0.3	38.2 ± 0.7	43.0 ± 0.4	20.0 ± 1.8	30.7 ± 1.9	43.4 ± 0.4	49.9 ± 0.4	56.1 ± 0.3
MEANPOOL	17.0 ± 2.9	20.3 ± 0.3	29.1 ± 1.1	34.8 ± 1.1	42.0 ± 0.3	10.4 ± 2.1	18.0 ± 2.4	34.3 ± 2.4	46.2 ± 1.1	55.0 ± 0.5
MAXPOOL	15.7 ± 0.8	24.0 ± 1.2	33.5 ± 0.4	37.5 ± 1.0	43.7 ± 0.1	12.4 ± 1.9	26.0 ± 0.6	44.5 ± 1.0	51.4 ± 0.3	57.5 ± 0.2
ATT	19.8 ± 3.1	26.0 ± 0.5	35.5 ± 0.9	40.1 ± 0.5	43.8 ± 0.2	21.3 ± 4.3	37.1 ± 0.7	46.1 ± 0.6	51.3 ± 0.8	57.2 ± 0.2
MAXATT	19.7 ± 3.5	27.0 ± 0.9	36.2 ± 1.3	40.0 ± 0.6	43.7 ± 0.3	22.1 ± 5.9	36.7 ± 1.3	46.7 ± 0.1	52.2 ± 0.1	57.5 ± 0.2
	Yahoo (mid) + Wiki					Amazon (mid) + Wiki				
	1K	2K	5K	10K	25K	1K	2K	5K	10K	25K
BiLSTM	9.9 ± 0.7	12.3 ± 0.8	17.4 ± 1.1	24.2 ± 0.9	36.3 ± 0.5	5.6 ± 0.4	6.9 ± 0.5	20.3 ± 1.0	37.9 ± 0.9	51.5 ± 1.0
MEANPOOL	14.9 ± 2.2	22.1 ± 1.3	28.3 ± 0.4	32.8 ± 0.8	39.2 ± 0.4	10.8 ± 1.9	20.8 ± 1.3	39.0 ± 0.6	46.5 ± 0.5	54.8 ± 0.1
MAXPOOL	14.1 ± 2.6	22.6 ± 0.3	28.6 ± 0.5	33.8 ± 1.2	40.1 ± 0.5	10.6 ± 1.8	21.3 ± 1.7	37.1 ± 1.4	47.0 ± 0.9	55.3 ± 0.4
ATT	16.9 ± 3.0	24.8 ± 1.1	31.4 ± 0.9	37.6 ± 0.5	42.1 ± 0.4	17.4 ± 3.2	33.2 ± 1.0	43.9 ± 0.5	49.7 ± 0.3	55.4 ± 0.1
MAXATT	18.2 ± 2.4	25.7 ± 0.5	32.6 ± 0.6	37.8 ± 0.8	42.1 ± 0.4	17.8 ± 4.6	35.0 ± 1.2	44.7 ± 0.3	49.7 ± 0.5	55.8 ± 0.4
	Yahoo (right) + Wiki					Amazon (right) + Wiki				
	1K	2K	5K	10K	25K	1K	2K	5K	10K	25K
BiLSTM	12.3 ± 0.5	23.8 ± 1.2	33.4 ± 0.6	38.2 ± 0.2	43.8 ± 0.3	7.4 ± 0.8	15.3 ± 3.2	40.8 ± 0.5	50.4 ± 0.7	58.4 ± 0.4
MEANPOOL	15.7 ± 1.9	22.7 ± 0.4	27.7 ± 0.9	34.2 ± 0.6	41.3 ± 0.1	14.8 ± 2.0	20.4 ± 3.3	40.1 ± 1.2	48.6 ± 0.5	56.9 ± 0.3
MAXPOOL	14.7 ± 0.6	22.5 ± 1.5	33.6 ± 0.5	38.5 ± 0.4	43.4 ± 0.5	11.1 ± 2.3	24.0 ± 1.9	45.6 ± 0.5	52.0 ± 0.4	58.4 ± 0.3
ATT	19.7 ± 0.2	27.4 ± 1.5	35.9 ± 0.2	40.0 ± 0.4	43.8 ± 0.7	22.4 ± 5.7	36.6 ± 1.3	46.7 ± 0.4	52.5 ± 0.4	59.1 ± 0.3
MAXATT	20.3 ± 1.3	28.1 ± 0.9	35.4 ± 0.8	40.3 ± 0.4	43.8 ± 0.4	20.8 ± 6.8	37.3 ± 0.9	47.8 ± 0.4	53.1 ± 0.3	59.0 ± 0.2

Table 1.8: Mean test accuracy (± std) (in %) on different manipulated settings across 5 random seeds on the Yahoo, Amazon datasets with short sentences (less than 100 words).

For shorter sequences, we reuse two of our text classification tasks: (1) **Yahoo! Answers**; and (2) **Amazon** Reviews. Similar to the setting with long sentences in the main paper, we use only the text and labels, ignoring any auxiliary information (like title or location). We select subsets of the datasets with sequences having a length (number of space separated words) less than 100. A summary of statistics with respect

to sentence length and corpus size is given in Table 1.7.

The results for the performance of the trained models can be found in Table 1.8. In the 'Mid' setting, we observe that BiLSTM performs significantly better on shorter sequences as opposed to the long sequences. For instance, in case of Amazon Dataset (Mid), under the 25K data setting, the classification accuracy increases from 7.8% in Table 1.5 to 51.5% in Table 1.8, which is a significant improvement from only doing as well as majority guessing in the former. We note that most of the learning issues of BiLSTM in long sentence setting are largely absent when sentence lengths are short, with BiLSTM also emerging as the best-performing model in a few cases. This corroborates the effect of gradients vanishing with longer time steps.

1.9 Fine-grained Positional Biases

How does the position of a word affect its contribution to a model's prediction?

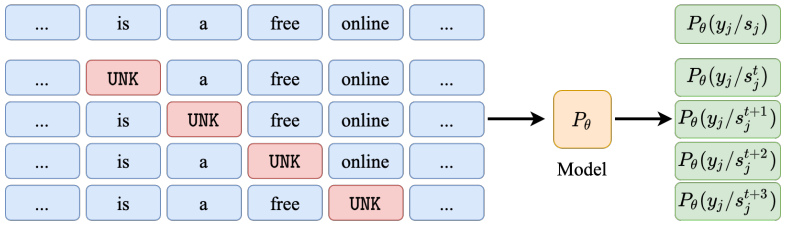
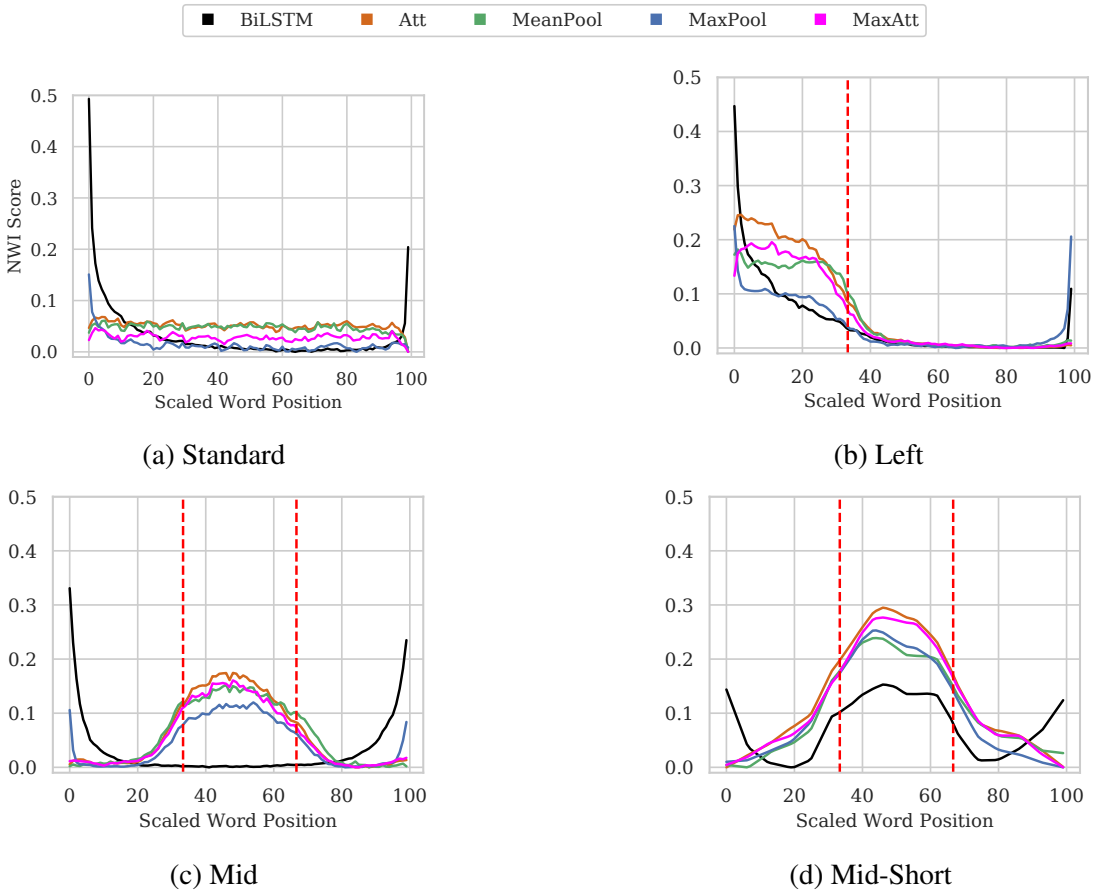


Figure 1.8: Explaining NWI evaluation.

1.9.1 Experimental Setup

We aim to achieve a fine-grained understanding of model biases w.r.t. each word position, as opposed to evaluating the same at a coarse level (between left, mid and right) as in the previous experiment (§ 1.8). To this end, we define Normalized Word Importance (NWI), a metric to determine the per-position importance of words as attributed by the model. It measures the importance of a particular word (or a set of words) on a model's prediction by calculating the change in the model's confidence in the prediction after



(a) Standard

(b) Left

(c) Mid

(d) Mid-Short

Figure 1.9: Normalized Word Importance w.r.t. word position averaged over examples of length between 400-500 on the Yahoo (25K) dataset in (a,b,c) using $k = 5$; and NWI for examples of length between 50-60 on the Yahoo Short (25K) dataset in (d) with $k. = 3$. Results shown for 'standard', 'left' & 'mid' training settings described in § 1.8. The vertical red line represents a separator between relevant and irrelevant information (by construction).

replacing it with UNK. (Figure 1.8). The evaluation is further extended by removing a sequence of k consecutive words to get a smoother metric. The metric is adapted from past efforts to assign word importance, with some differences (Khandelwal et al., 2018; Verwimp et al., 2018; Jain and Wallace, 2019). We believe that this is a more versatile evaluation metric as opposed to previous work (Khandelwal et al., 2018; Verwimp et al., 2018) as it has a direct correlation with the final model prediction, thus being more semantically relevant. Khandelwal et al. remove all words beyond a certain context and thus answer the question of how important are *all* the removed words, and not one particular word. It may be unfair to label models to be biased because they are able to perform well by seeing only a small amount of information (at the ends), and in fact

this may be a desirable property in some scenarios. On the other hand, Verwimp et al. check the transfer of gradient directions to determine how much of the word is remembered. This mechanism cannot compute the importance for multiple words at a time (say phrase). Finally, recent work has used a related approach (leave-one-out) to calculate word importance (Jain and Wallace, 2019), however our method is able to capture positional importance better, since it substitutes the prior word with an irrelevant word rather than removing that position altogether. Note that leave-one-out approach, deletes a given word rather than replacing it with `UNK`, effectively shifting positions of words by one, and thus can not evaluate positional importance.

1.9.2 Results

The results from this experiment are presented in Figure 1.9 (on the Yahoo dataset). The NWI for architectures with pooling indicate no bias w.r.t. word position, however for BiLSTM there exists a clear bias towards the extreme words on either ends (c.f. Figure 1.9a). The word importance plots in Figure 1.9b & 1.9c demonstrate how pooling is able to *learn* to disambiguate between words that are important for sentence classification significantly better as opposed to BiLSTM. There is a clear peak in the middle in case of 'mid' setting, and on the left in case of 'left' setting for all the pooling architectures. BiLSTM is unable to respond to middle words in Figure 1.9c. However, they show reasonably higher importance to the left tokens in Figure 1.9b which is justified by their good performance in the 'left' experimental setting in Table 1.3. Results for NWI evaluation on all datasets and modified settings (left, mid and right) are available in § 1.9.3, and are consistent with the representative graphs in Figure 1.9. We also perform such an analysis on models that are trained on datasets with shorter sentences. Interestingly, the NWI analysis for the Yahoo short dataset in Figure 1.9d shows that while BiLSTM can better respond to middle words for shorter sentences, it still remains heavily biased towards the ends. We detail these findings in § 1.9.5

1.9.3 Detailed Description

We now provide a complete description of the algorithm to compute NWI, along with further evaluation on IMDb and Amazon datasets.

We detail the method for calculating the Normalized Word Importance (NWI) score in Algorithm 1. The parameter k can be adjusted according to the average sentence

Algorithm 1 *NWI* evaluation

Input: softmax classifier P_θ, test set D
Parameters: k
for $s_j = \{x_j^1, \ldots, x_j^n\}, y_j$ in D **do**
  $p_j = \log\{P_\theta(y_j|s_j)\}$
  for $t = 0 \ldots \frac{n}{k}$ **do**
    $s_j^t = \{x_j^1, \ldots, x_j^{k.t}, \underbrace{\texttt{UNK}, \ldots, \texttt{UNK}}_{\text{k words}}, \ldots, x_j^n\}$
    $p_j^t = \log\{P_\theta(y_j|s_j^t)\}$
    $\delta_j^t = |p_j^t - p_j|$
  end for
  $\text{nwi}_j = \frac{\delta_j}{\max_{t\in(1,\frac{n}{k})}\delta_j^t}$
  $\text{nwi}_j = \text{nwi}_j - \min_{t\in(1,\frac{n}{k})}\delta_j^t$
  $\text{nwi}_j = \texttt{LinInterp}(\text{nwi}_j, \frac{n}{k}, 100)$
end for
return $\frac{1}{|D|}\sum_{j=1}^{|D|}\text{nwi}_j$
*$\texttt{LinInterp}(x, n, l)$ linearly interpolates input distribution x of n discrete steps to l steps.

length. For a sentence of length 100, setting an extremely low value of k (say 1) may have very little impact of the model's prediction $\log\{P_\theta(y_j|s_j^t)\}$ for all positions t. On the other hand, setting an extremely high value of k (say 20) may provide only few data points, and also change the model prediction drastically at all values of t.

Complete graphs for the positional importance (as perceived by the model) of words are detailed in this section. The trends observed for the remaining datasets are similar to the representative graphs shown in the main paper. We show graphs for the IMDb, Yahoo and Amazon datasets in Figure 1.10.

1.9.4 Practical Implications

Our findings suggest that adversaries can easily replace the middle portion of texts with racist or abusive sentences, and still stay undetected by BiLSTM based detection systems. This is because BiLSTM attributes little or no importance to words in the middle of the input. Pooling based models are able to circumvent this issue by being able to attribute importance to words irrespective of their position.

1.9.5 NWI for Short sentences

We repeat our experiments of NWI evaluation on the datasets with short sentences (<100 words) as described in § 1.8.5. It is interesting to observe the graphs on the Yahoo and Amazon short datasets in Figure 1.11, where due to the short sentence length, even BiLSTM is able to show the desired importance characteristic in case of mid setting. This supports the fact that the test time accuracies in the mid setting are no longer as bad as a majority class predictor. Interestingly, in case of short sentences in the mid setting (Figures 1.11c,1.11f), we observe three peaks in the NWI graph. The one in the middle is expected given the data distribution. However, the two peaks in NWI at the extreme ends help establish that while BiLSTM is able to propagate gradients to the middle given the short sentences, it is still not able to forego the extreme bias towards the end tokens.

1.10 Learning Under Limited Data

Recent works have proposed the use of self-supervised learning to enhance model performance in downstream tasks providing very little training data. More specifically, in the most extreme setting, Chen et al. (2020) study the learning trade-offs of various model architectures when only 10 training examples corresponding to each of the class labels are available for the model. The work compares the performance of various pre-trained models and semi-supervised learning algorithms. In this section, we compare our proposed MAXATT with these methods.

1.10.1 Architectural modifications

We do not make any modifications in the LSTM architecture for this problem. However, we replace the GloVe embeddings with embeddings generated from pre-trained BERT models. Such utilization of pre-trained language models has also been utilized in prior work (Liu et al., 2020; Devlin et al., 2018b). Since BERT has the limitation of only being able to use a maximum sentence length of 512 tokens, our model only considers the first 512 tokens of any given input sentence. We use the MAXATT model on top of these embeddings to report our results.

1.10.2 Baselines

We compare our models with the following five baselines, in line with previous work (Chen et al., 2020).

VAMPIRE VAriational Methods for Pretraining In Resource-limited Environments (Gururangan et al., 2019) propose a pretraining framework that aims to capture input (textual) data representations by training a unigram document model as a VAE on unlabeled data, and finally using the extracted features for downstream text classification problems.

BERT BERT (Devlin et al., 2018b) is one of the most popular pre-trained model used for various down stream natural language tasks, and was the first paper to show the effectiveness of transformer based models in training large language models. For this evaluation, we report results from finetuning the BERT-based-uncased model, in line with Chen et al. (2020).

UDA Unsupervised Data Augmentation for Consistency Training (Xie et al., 2019) is a semi-supervised training algorithm that introduces advanced data augmentation techniques to improve performance on tasks with as few as 20 labeled inputs.

TMix Chen et al. (2020) utilize TMix as a data augmentation strategy that attempts to interpolate different pieces of text within their hidden representation space to create an infinite amount of augmentable data for training. The method is used for augmenting data in the MixText model, but also reported as a baseline by the authors, in spirit of an ablation to the other architectural modifications in MixText.

MixText Proposed by Chen et al. (2020), MixText aims to specifically solve the problem of text classification under limited labeled datapoints. By using a mixture of labeled, unlabeled and augmented data, MixText achieves state of the art performance on different semi-supervised and pre-trained and fine-tuned models.

1.10.3 Results

Results for the comparison on IMDb and Yahoo datasets are shown in Table 1.9. We observe that MAXATT outperforms past approaches such as VAMPIRE, TMix and BERT by over 5% accuracy points on both the Yahoo and IMDb datasets. When compared with UDA and MixText, we note that both of these algorithms use extra unlabeled data for training, whereas our MAXATT model shows competitive performance against these models as well. Note that the approach was not specifically designed to learn in extremely low-resourced settings that require unsupervised data augmentation, but still generalizes well, indicating its ease of training.

Once again, we observe that MAXATT has superior performance in settings with extremely limited learning data. We speculate that this is primarily because a local attention over the 'max' embedding helps the model to easily assign blame to individual tokens of a sentence that are responsible for a given prediction.

1.11 Discussion

We hypothesize that for pooling to show its significant superiority, the problems must have some important features. First, the number of tokens in the input should be high

Model	Yahoo	IMDb
VAMPIRE	50.1 %	61 %
BERT	56.2 %	67.5 %
TMix	58.6 %	69.3 %
MAXATT	65.7 %	75.6 %
UDA	63.2 %	78.2 %
MixText	67.6 %	78.7 %

Table 1.9: Test set accuracy (in %) for different models on the Yahoo and IMDb datasets. The models were trained using only 10 labeled data points for each class. While UDA and MixText make use of unlabeled data during trainingl VAMPIRE, BERT, TMIX and MAXATT do not. MAXATT is the best performing model among the approaches that do not utilize any unlabeled data.

to require the LSTM to learn long-range dependencies. Second, the training dataset size should be small. Typically, models overfit quickly on small datasets, and our experiments show that it takes several epochs for BiLSTM models to start propagating gradients towards the middle. Alternatively, the setting where important words for the task are in the middle of the input (in conjunction with long input sequences) creates a pathological case for BiLSTM models.

However, it is not clear whether these features occur that often in real-world datasets, since often the domain or sentiment of text can be quickly deciphered from just a few words. That said, our analysis suggests an easy way for adversaries to fool BiLSTMs, by embedding important text (e.g., racist or offensive texts) in the middle of their narrative. Further, in Appendix 1.7.3 we discuss how MEANPOOL, ATT perform poorly when the test distribution of words is changed. A shift in the distribution of words at test time can be commonly observed in real-world scenarios, making the practical utility of such pooling methods questionable.

Finally, we show that not only is MAXATT robust to these dangers, but it consistently gives small improvements on regular datasets. We argue that MAXATT should be the pooling of choice for computing sentence embeddings for classification tasks.

1.11.1 Tranformers

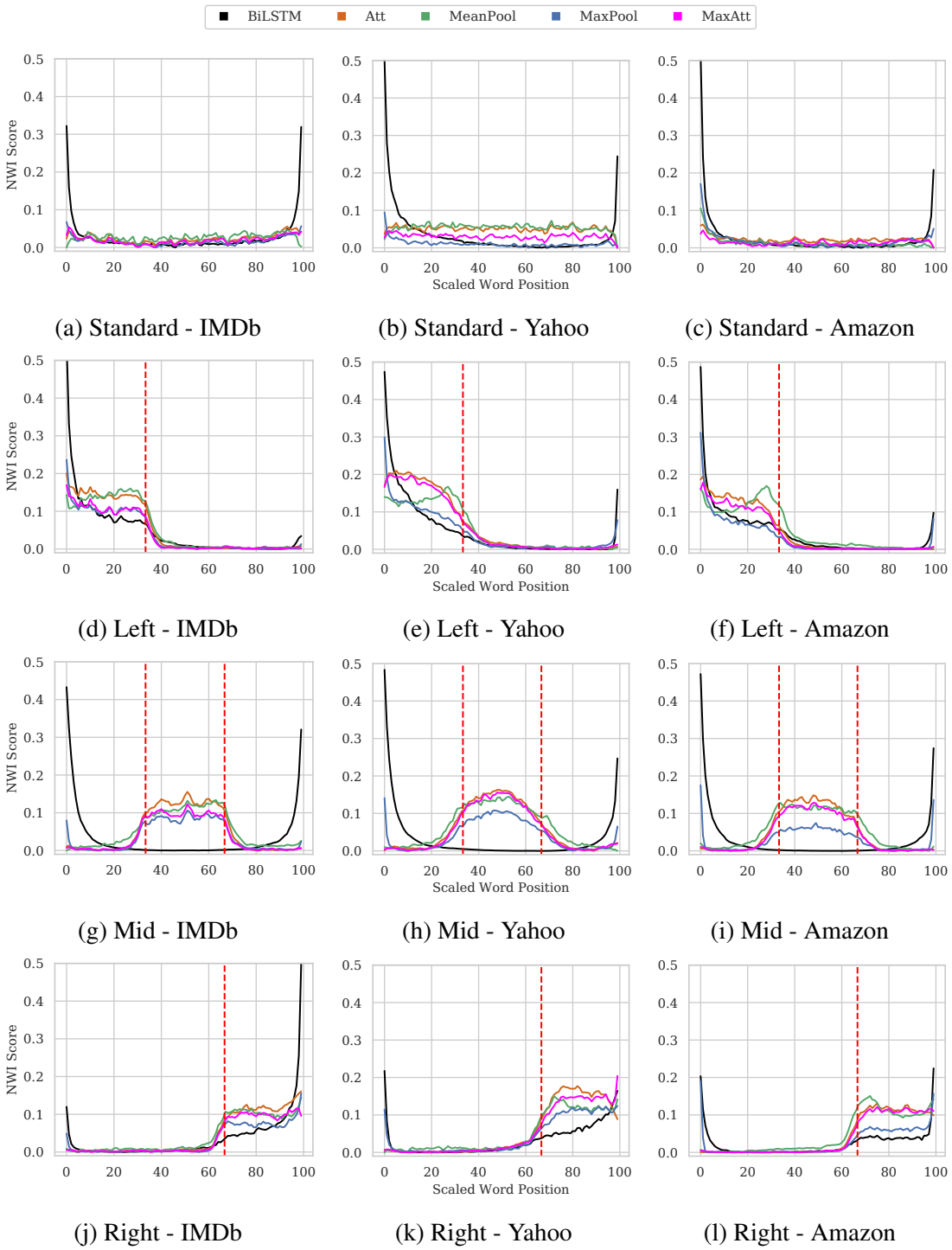
We briefly perform our experiments on the more recent transformer architectures (Vaswani et al., 2017; Devlin et al., 2018a). Purely attention-based architectures (such as transformers), without significant pre-training, perform poorly on text-classification. Thus they are most valuable for resource-rich languages. Randomly-initialized transformers show an average drop of 6.4% compared to MAXATT on our datasets. Further, transformers explode in memory usage for longer sentences. The memory scaling is $O(n^2)$ for transformers vs $O(n)$ for LSTM. On XL-Net (Yang et al., 2019), long examples, like the ones used in our wiki experiments (sentences up to 4000 words), make the transformer untrainable even for a batch-size of 1 on a 32GB GPU. Thus, we strongly believe in the continued relevance of recurrent architectures in the presence of both computational and language-based resource constraints.

Future Work on Transformers We also note that MAXATT is a simple and versatile pooling technique that has a natural extension to transformer based networks. We hope our work further motivates the development of informed attentive networks in the recently successful transformer-based models. This may potentially help remedy the failure of transformer models in low data settings, since in their present form they require a large amount of data to satisfactorily learn the query, key and value vectors. However, using locally informed query, key vectors can help overcome the need for large training data.

1.12 Conclusion

Through detailed analysis we identify *why* and *when* pooling representations are beneficial in RNNs. We attribute the performance benefits of pooling techniques to their learning ability (pooling mitigates the problem of vanishing gradients), and positional invariance (pooling eliminates positional biases). Our findings suggest that pooling offers large gains when the training examples are few and long, or when salient words lie towards the middle of the sequence. Lastly, we introduce a novel pooling technique

called max-attention, which often outperforms other pooling variants. Most of our insights are derived for sequence classification tasks using RNNs. While our proposed pooling method and analyses are broadly applicable, it remains a part of the future work to evaluate its impact on other tasks and architectures.



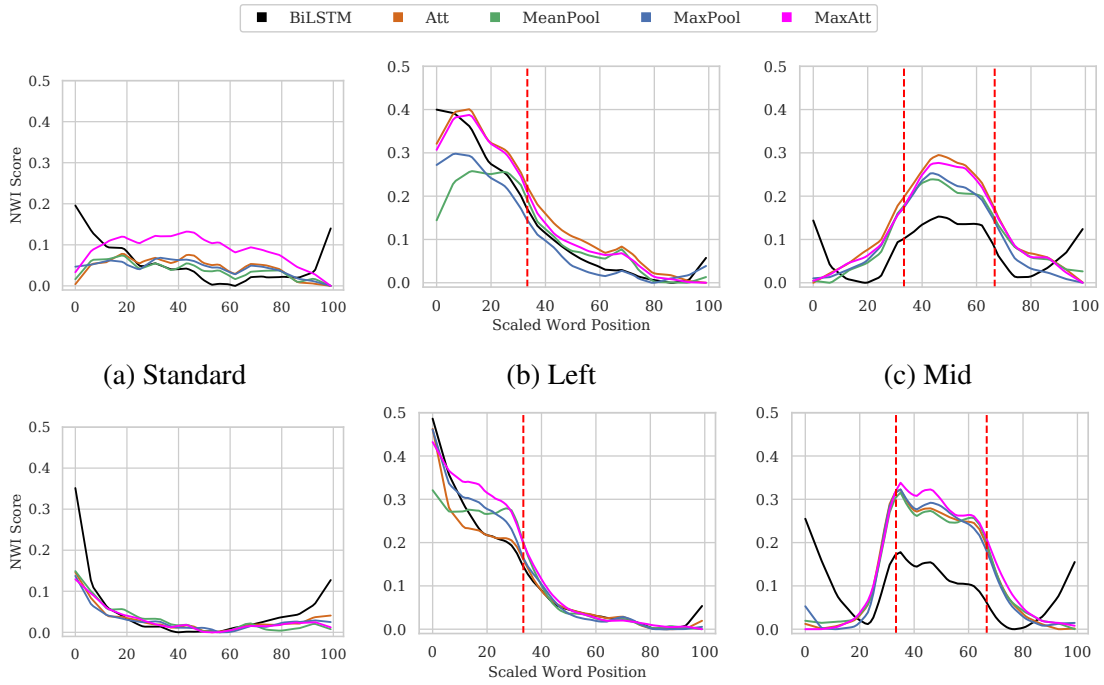
(a) Standard - IMDb (b) Standard - Yahoo (c) Standard - Amazon

(d) Left - IMDb (e) Left - Yahoo (f) Left - Amazon

(g) Mid - IMDb (h) Mid - Yahoo (i) Mid - Amazon

(j) Right - IMDb (k) Right - Yahoo (l) Right - Amazon

Figure 1.10: Normalized Word Importance w.r.t. word position for k = 5; averaged over sentences of length between 400-500 on the IMDb, Yahoo, Amazon (10K) Datasets. Results shown for the 'standard', 'left', 'mid' and 'right' training settings described in § 1.8. The vertical red line represents an approximate separator between relevant and irrelevant information (by construction). For instance, The word positions to the 'left' of the vertical line in graphs in the second row of the Figure contain data from true input examples, while those to the right contain Wikipedia sentences.



(a) Standard (b) Left (c) Mid

(d) Standard (e) Left (f) Mid

Figure 1.11: Normalized Word Importance w.r.t. word position for k = 3; averaged over sentences of length between 50-60 on the Yahoo, Amazon (10K) Datasets. Results shown for the 'standard', 'left' and 'mid' training settings described in § 1.8.5. The vertical red line represents an approximate separator between relevant and irrelevant information (by construction). For instance, The word positions to the 'left' of the vertical line in (b), (e) contain data from true input examples, while those to the right contain Wikipedia sentences.

CHAPTER 2

Disentangling Learnability & Representability of LSTMs

2.1 Theoretical Analysis

2.1.1 Introduction

LSTMs were introduced to mitigate the problem of vanishing gradients in standard recurrent architectures. In the previous chapter we show that despite these efforts, the performance of LSTMs is limited by their recurrent nature and they suffer from vanishing gradients in the early part of training, leading to an inherent bias towards the end tokens. However, this does not distinguish the effect of learnability from representability in these experimental findings. We attempt to bridge this gap in our understanding of recurrent architectures. To these ends, in this chapter we investigate the representative power of LSTMs under different types of pooling and attempt to explore problems that pooled LSTMs can not solve. Next, we theoretically examine how the gradients of LSTMs vanish independent of the task. We show that LSTMs have universal representation power, but still suffer from learning issues.

2.1.2 Related Work

Multiple works have attempted at theoretically quantifying the expressivity and limitations of recurrent networks. Schäfer and Zimmermann (2006) show that recurrent neural networks are universal approximators by unrolling a long chain of recurrence into its constituent functions, each of which are universal approximators by the universal approximation theorem for two-layer infinite width neural networks (Cybenko,

1989). While RNNs are Turing complete in the limit of infinite precision, Weiss et al. (2018) study the computation power of different RNNs under a finite precision whose computation time grows linearly with the length of the input. Khrulkov et al. (2017) show that for certain tasks RNNs can be exponentially more efficient and expressive than shallow convolutional networks with one hidden layer. The expressive capacity of different RNN architectures like LSTMs and QRNN is further formalized in the form of a formal hierarchy by Merrill et al. (2020). Finally, Hahn (2019) mathematically investigate the computation capabilities of self-attention networks when deprived of recurrence, and show that in the absence of recurrence, purely attention based networks can not model hierarchical structures, unless the model size grows linearly with the input length.

2.1.3 Methods

The contents of this section can be referred to in Chowdhury (2020) and contain theoretical methods at understanding the prevalence of vanishing gradients in LSTMs and the representative power of MAXPOOL and LSTM.

Vanishing Gradients In § 2.2 Chowdhury (2020) show how under certain relaxed assumptions, it is indeed possible to theoretically show that for subsequent time steps away from the last hidden state, the expected value of the norm of gradients with respect to that hidden state decreases by a factor less than 1.

LSTM equivalence In § 2.4 Chowdhury (2020) develop various scenarios under which the representative power of an LSTM and MAXPOOL model can be equivalent. Further, they also show that under limit of finite precision, it is possible to prove that MAXPOOL can not approximate all functions.

The interested reader is referred to Chapter 2 (Chowdhury, 2020) for a deeper take on the theoretical extensions and analyses of the problems discussed in this report.

2.2 Enhancing the Learnability of LSTMs

2.2.1 Introduction

In the previous section, we showed that while LSTMs do have universal approximation power, they still have limitations in learning these representations. Now, we attempt to mitigate its learning issues using three different training time modifications (and no architectural changes). While we are able to show substantial gains in the performance of LSTMs in tasks where it was shown to fail in past works, we observe that despite our best efforts, standard LSTM performance is generally succeeded by LSTMs in the presence of pooling. This opens interest for further exploring aspects other than gradient propagation that limit LSTMs but not their pooling-based counterparts.

2.2.2 Related Work

Improving the ability of recurrent networks in capturing both long and short term dependencies had been a long standing problem. Recent works such as the Ordered Neurons LSTM (ON-LSTM) (Shen et al., 2018) have shown considerable promise in being able to model hierarchical dependencies by integrating tree structures into the RNN. Tallec and Ollivier (2018) introduce a new way of initializing gradients of an LSTM, called chrono-initialization. They experimentally show that this new initialization strategy greatly helps improve the ability of LSTMs to capture long term dependencies.

Other approaches have aimed at minimizing the number of LSTM gates and simplifying the architecture in a more principled way. This includes approaches such as utilization of diagonal weight matrices or orthogonal initialization (Arjovsky et al., 2015b; Le et al., 2015; Henaff et al., 2016; Gulcehre et al., 2017; Li et al., 2018; Chandar et al., 2019b). In another line of work Bhatt et al. (2020) develop biologically inspired RNNs that can capture long term dependencies by incorporating a decay term inside the architecture.

While most of these approaches suggest architectural changes to the recurrent ar-

chitecture itself, our work is focused at utilizing algorithmic changes to enhance the learnability of standard LSTMs and can be augmented on top of any of these methods.

2.2.3 Experimental Setup

We perform our experiments on the IMDb dataset in two of the settings introduced in § 1.6, namely the standard and Wiki-mid setting. For each of the settings we aim to reduce the performance gap between standard LSTMs and pooled LSTMs using four alternate training time modifications, with no architectural change at the evaluation time.

Teacher Forcing Given a standard LSTM and a max-pooled LSTM, we first train MAXPOOL to the given task. To train the LSTM, we make use of soft sentence embedding vectors from MAXPOOL model and convert the classification task into a regression task, where we ask the final hidden state of an LSTM to learn to mimic the max-pooled sentence embedding.

Penalty Attention Pruthi et al. (2020) conjecture that attention aids model training even when it is not used at inference time. The objective of this approach is to perform a curriculum based training, starting with an attention based model, and penalizing any attention mass given to any hidden state apart from the final hidden state. Given a sufficiently high penalty factor, the only way for the model to learn to perform well on the task is by using the last hidden state as the sentence embedding.

Dropout Attention Dropout attention considers a small modification to the idea of penalty attention. Rather than penalizing attention weights at non-final positions, we aim to create redundancies within the attention modules. For a model to perform well, it will have to learn to propagate gradients within the recurrent chain as the randomly dropped attention values make the attention module unreliable. Once again we follow a curriculum training approach and increase the dropout ratio as training proceeds.

Gradient-directed gradients One of the benefits of pooling, as outlined in Maini et al. (2020), is that it allows the flow of gradient to any hidden state without propagating through a long recurrent chain. In this method, we aim to provide artificial paths for gradient propagation without any architectural modification (at train time) as in the previous two approaches. The key idea is to add a loss term for correcting imbalanced gradients, which directly have attribution to individual hidden states.

2.2.4 Teacher Forcing

Method Given a standard LSTM and a max-pooled LSTM, we first train MAXPOOL to the given task. To train the LSTM, we make use of soft sentence embedding vectors from MAXPOOL model and convert the classification task into a regression task, where we ask the final hidden state of an LSTM to learn to mimic the max-pooled sentence embedding.

Loss formulation While standard training utilizes the cross-entropy loss, since our new objective is governed by soft-vectors, we consider a mean-squared error between the final sentence embedding of the LSTM and that of MAXPOOL (which we use as gold labels). Formally, for an input of length n, the loss can be defined as:

$$
\begin{aligned}
s_{\text{emb}} &= h_n \\
s^{m,i}_{\text{emb}} &= \max_{t \in (1,n)} (h^{m,i}_t) \\
\ell(x, y) &= \| s_{\text{emb}}(x) - s^m_{\text{emb}}(x) \|
\end{aligned}
$$

where s_{emb} represents the final hidden state of the LSTM model and s^m_{emb} represents the max-pooled sentence embedding of the trained MAXPOOL model, and $\|.\|$ represents the ℓ_2 norm.

Training procedure and Observations We use the Adam optimizer to train our models and train for 20 epochs. No special curriculum setup is required for these experiments. In the standard training settings, LSTMs can mimic MAXPOOL sentence em-

bedding with a very mean squared error of the order 10^{-6}. The procedure also helps improves test accuracy by 1-5% (depending on data size setting) on the IMDb dataset.

Limitations In the Wiki mid setting, the LSTM still fails to achieve any reasonable accuracy above 50% on the IMDb dataset. For this experiment we train a MAXPOOL model on the same Wiki mid dataset and use its sentence embedding as gold vectors. However, the LSTM is unable to mimic the MAXPOOL embedding in this scenario and suffers from high mean squared error of the order 10^{-1}. This further suggests that the learning issue faced by LSTMs in challenging scenarios where the middle words are important, is not a caveat of the task (classification), but a consequence of the recurrent architecture itself.

2.2.5 Penalty Attention

Method Pruthi et al. (2020) conjecture that attention aids model training even when it is not used at inference time. The objective of this approach is to perform a curriculum based training, starting with an attention based model, and penalizing any attention mass given to any hidden state apart from the final hidden state. Given a sufficiently high penalty factor, the only way for the model to learn to perform well on the task is by using the last hidden state as the sentence embedding.

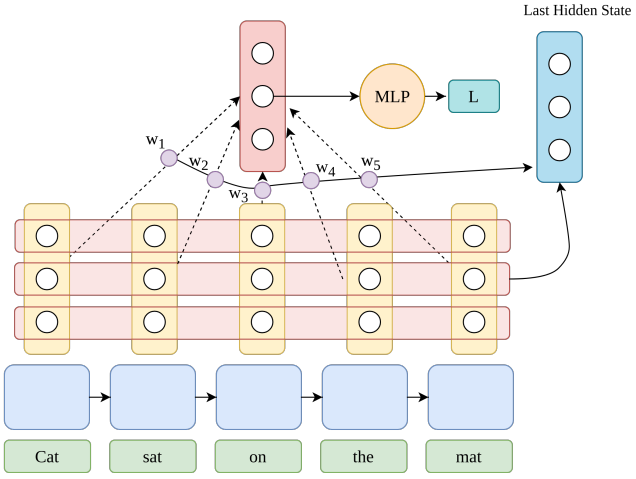


Figure 2.1: Explaining the last-attention model.

We modify the MAXATT architecture proposed in (Maini et al., 2020) to formulate

a new form of attention called last-attention. An illustration of the same is provided in Figure 2.1. The key idea is that each hidden state attends over the last hidden state as query, and our objective is to finally reduce all attention mass at the end state alone. This would mean the last hidden state taking a dot product with itself and producing the last hidden-state as the sentence representation as well. For a sequence of maximum length n this is given by:

$$
\begin{aligned}
q &= h_n \\
\hat{h_t} &= h_t / \|h_t\| \\
\alpha_t &= \frac{\exp(\hat{h_t}^\top q)}{\sum_{j=1}^{n} \exp(\hat{h_j}^\top q)} \\
s_{\text{emb}} &= \sum_{t=1}^{n} \alpha_t h_t
\end{aligned}
$$

where α_t represent the attention mass attributed to the hidden state at time step t.

Loss formulation The training loss is a sum of the standard crossentropy loss and a penalty loss that penalizes attention mass given to any hidden state apart from the last one.

$$
\begin{aligned}
y_p &= f_\theta(x) \\
\ell(x, y; \theta) &= \text{CrossEntropyLoss}(y_p, y) + \lambda \frac{\sum_{t=1}^{n-1} \alpha_i}{\alpha_n} \\
&= -y_p^y + \log\left(\sum_j \exp(y_p^j)\right) + \lambda \frac{\sum_{t=1}^{n-1} \alpha_i}{\alpha_n}
\end{aligned}
$$

Training procedure and Observations The penalty factor λ undergoes a curriculum procedure where we modify λ from 1 to 1 as the number of epochs increase. More specifically, we train the model for a total of 20 epochs, with the last 10 trained at $\lambda = 1$ that is complete drop of attention module. The dropout value is given by:

$$
\lambda = \left(\frac{\text{epoch}}{\text{max epochs}}\right)^\eta
$$

where max epochs is set to 10 in our case, and epoch represents the current epoch number. The parameter η is used to slow down the increase of dropout value in the initial epochs and is set to 4 for our experiments.

Limitations We observe from the results in Table 2.1 that the use of a penalty attention indeed does help improve the flow of gradients in the LSTMs and allows the model to learn from tokens that are present in the middle of a long sequence as in our Wiki mid setting. However, we note that the performance is still sub-optimal when compared with MAXPOOL by over 10% points on the IMDb Wiki mid setting.

2.2.6 Dropout Attention

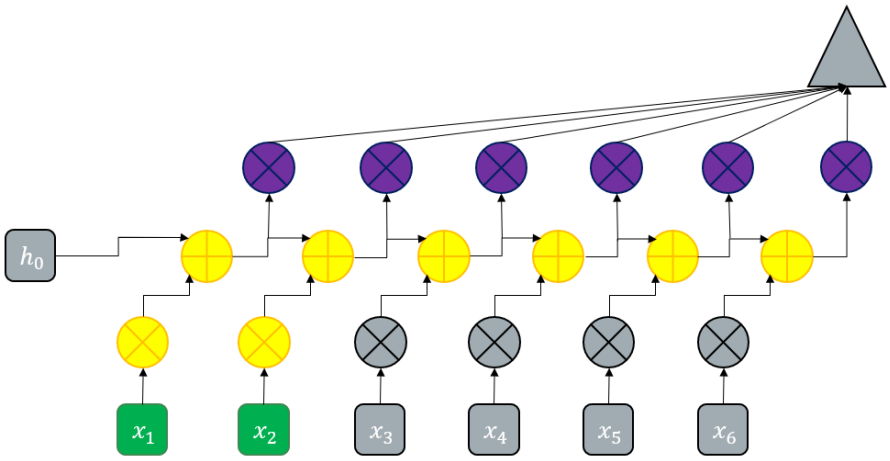


Figure 2.2: Explaining dropout-attention. Gates in yellow represent the relevant gates at the end of curriculum training, the ones in purple denote those that are probabilistically forced to 0 and green suggests the input under consideration. For inputs x_1, x_2 in the initial phase of curriculum training, the purple gates probabilistically turn on or off. As a result, the model is forced to learn to propagate gradients through the long chain denoted in yellow.

Method Dropout attention considers a small modification to the idea of penalty attention. Rather than penalizing attention weights at non-final positions, we aim to create redundancies within the attention modules. An illustration of the same is provided in Figure 2.2. For a model to perform well, it will have to learn to propagate gradients within the recurrent chain as the randomly dropped attention values make the attention

module unreliable. Once again we follow a curriculum training approach and increase the dropout ratio as training proceeds.

Loss formulation There is no change in the training time loss and it is standard cross-entropy loss given by:

$$\begin{aligned} y_p &= f_\theta(x) \\ \ell(x, y; \theta) &= \text{CrossEntropyLoss}(y_p, y) \\ &= -y_p^y + \log\left(\sum_j \exp(y_p^j)\right) \end{aligned}$$

Training procedure The dropout ratio p undergoes a curriculum procedure where we modify p from 0 to 1 as the number of epochs increase. More specifically, we train the model for a total of 20 epochs, with the last 10 trained at $p = 1$ that is complete drop of attention module. The dropout value is given by:

$$p = \min\left\{1, \left(\frac{\text{epoch}}{\text{max epochs}}\right)^\eta\right\}$$

where max epochs is set to 10 in our case, and epoch represents the current epoch number. The parameter η is used to slow down the increase of dropout value in the initial epochs and is set to 2 for our experiments.

Limitations A more thorough analysis and the intuition behind Dropout attention is deferred to § 2.3 (Chowdhury, 2020). We note that Dropout attention is a simple curriculum method to improve gradient flow without the side-effect of having to change the model objective function during training. As a result, we also find performance gains when compared with penalty attention on both the IMDb standard and Wiki mid settings (10k data points) in Table 2.1. However, we find that in the final epoch when dropout ratio p becomes equal to 1, there is a sudden performance drop of the LSTM. This indicates that even when the attention mass on the non-final hidden states is extremely low (order 10^{-4}), there is a significant advantage of using attention. These

results are also supported by observations in Pruthi et al. (2020).

2.2.7 Gradient-directed gradients

Method One of the benefits of pooling, as outlined in Maini et al. (2020), is that it allows the flow of gradient to any hidden state without propagating through a long recurrent chain. In this method, we aim to provide artificial paths for gradient propagation without any architectural modification (at train time) as in the previous two approaches. The key idea is to add a loss term for correcting imbalanced gradients, which directly have attribution to individual hidden states.

Loss formulation

$$
\begin{aligned}
y_p &= f_\theta(x) \\
\ell_1(x, y; \theta) &= \text{CrossEntropyLoss}(y_p, y) \\
x_g &= \frac{\partial \ell_1}{\partial x} \\
\ell_2 &= KL(x_g | \mathbf{1}) \\
(x, y; \theta) &= \ell_1(x, y; \theta) + \lambda \cdot \ell_2
\end{aligned}
$$

where the LSTM is given by f_θ, the input text is represnted by x and the true label is given by y. x_g represents the change in the standard cross-entropy loss with change in the input at each position. ℓ_2 is formulated in such a way that it calculates the KL divergence between the gradient distribution and a normalized constant distribution. The objective is to ensure that the divergence is minimized and the gradient flow at all positions is equivalent in the beginning.

Training procedure The training method once again follows a simple curriculum setup where the penalty parameter λ is reduced with time. While we want an initial flow of gradients to all the hidden states, as training proceeds the LSTM should be able to attribute importance to any hidden state irrespective of the penalty term.

Limitations While we find that in practice, gradient-directed gradients help in significantly boosting the performance of LSTMs without pooling, and even help them surpass the MAXPOOL model on the Wiki-mid setting in Table 2.1, the training procedure is itself very unstable, requires large amount of tuning of the parameter λ and shows sub-optimal performance when compared with recent approaches such as MAXATT (Maini et al., 2020).

2.2.8 Results

Method	IMDb Standard	IMDb Wiki Mid
LSTM	86.6 %	50.1 %
MAXPOOL	87.2 %	80.1 %
Teacher Forcing	87.3 %	49.8%
Penalty Attention	86.4 %	70.7 %
Dropout Attention	86.7 %	73.6 %
Gradient-directed LSTM	**87.4 %**	**80.3 %**

Table 2.1: Test set accuracy (in %) for different models and training methods is depicted on the IMDb dataset under two settings – standddard and Wiki mid. The models were trained using 10k labeled data points. The gradient directed LSTM training outperforms all the approaches discussed and also exceeds the test set accuracy of MAXPOOL by a small margin on the more challenging Wiki mid setting.

The results for each of the training methods discussed in this section are detailed in Table 2.1. We also present the results of standard LSTM and MAXPOOL models for reference. All the results presented are for the 10k data setting. It may be noted that in the Wiki mid setting, standard LSTM models were not able to perform any better than a majority classifier. However, an augmenting with the specified training algorithms to enrich gradient propagation in the the recurrent chain, we see substantial improvements in the Penalty Attention, Dropout Attention and the Gradient directed gradients approach. Finally, one may observe that the Gradient directed LSTM training approach is also able to outperform MAXPOOL method on both the standard and Wiki mid data setting.

2.2.9 Conclusions and Future Work

In this chapter, we attempted to enhance our understanding of the various factors that may lead to learning or representative issues for recurrent networks. We start by establishing that LSTMs are universal approximators and should be able to represent any function to a near approximation. On the contrary, we are not able to find a proof that justifies the universal approximation capabilities of other pooled architectures like MAXPOOL. We then attempt at distinguishing the learning aspects of LSTMs. First, we show that in expectation, (under certain relaxed constraints) the gradienst of an LSTM are bound to vanish with subsequent hidden states away from the end. Using these insights, coupled with the understanding that this is not a represntation issue, but only a learning problem, we then moved towards developing new training methods that can alleviate these problems. In our work, we developed four different training algorithms, that do not modify the LSTM architecture at inference time. We find that our final approach of adding a uniform gradient encouragement term within the optimization objective of the LSTM is the most useful and is able to succeed simple pooling variants like MAXPOOL. However, we also note that these training methods come with their own limitations of training instability and it remains a part of future work to examine the scope of these works specifically attempted at enhancing LSTM learnability.

REFERENCES

Martín Arjovsky, Amar Shah, and Yoshua Bengio. 2015a. Unitary evolution recurrent neural networks. In *ICML*.

Martin Arjovsky, Amar Shah, and Yoshua Bengio. 2015b. Unitary evolution recurrent neural networks.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate.

Yoshua Bengio, Patrice Y. Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5 2:157–66.

Gantavya Bhatt, Hritik Bansal, Rishubh Singh, and Sumeet Agarwal. 2020. How much complexity does an RNN architecture need to learn syntax-sensitive dependencies? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 244–254, Online. Association for Computational Linguistics.

Y-Lan Boureau, Francis Bach, Yann LeCun, and Jean Ponce. 2010a. Learning mid-level features for recognition. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2559–2566. IEEE.

Y-Lan Boureau, Nicolas Le Roux, Francis Bach, Jean Ponce, and Yann LeCun. 2011. Ask the locals: multi-way local pooling for image recognition. In *2011 International Conference on Computer Vision*, pages 2651–2658. IEEE.

Y-Lan Boureau, Jean Ponce, and Yann LeCun. 2010b. A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 111–118.

A. P. Sarath Chandar, Chinnadhurai Sankar, Eugene Vorontsov, Samira Ebrahimi Kahou, and Yoshua Bengio. 2019a. Towards non-saturating recurrent units for modelling long-term dependencies. In *AAAI*.

Sarath Chandar, Chinnadhurai Sankar, Eugene Vorontsov, Samira Ebrahimi Kahou, and Yoshua Bengio. 2019b. Towards non-saturating recurrent units for modelling long-term dependencies.

Jiaao Chen, Zichao Yang, and Diyi Yang. 2020. Mixtext: Linguistically-informed interpolation of hidden space for semi-supervised text classification.

Sankalan Pal Chowdhury. 2020. Masked adversarial auto-encoder and an analysis of lstm gradients.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *EMNLP*.

G. Cybenko. 1989. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, 2(4):303–314.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018a. Bert: Pre-training of deep bidirectional transformers for language understanding.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018b. Bert: Pre-training of deep bidirectional transformers for language understanding.

Felix A. Gers, Juergen Schmidhuber, and Fred Cummins. 2000. Learning to forget: continual prediction with lstm. In *Neural Computation*.

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

Caglar Gulcehre, Sarath Chandar, and Yoshua Bengio. 2017. Memory augmented neural networks with wormhole connections.

Suchin Gururangan, Tam Dang, Dallas Card, and Noah A. Smith. 2019. Variational pretraining for semi-supervised text classification.

Michael Hahn. 2019. Theoretical limitations of self-attention in neural sequence models.

Mikael Henaff, Arthur Szlam, and Yann LeCun. 2016. Recurrent orthogonal networks and long-memory tasks.

S Hochreiter. 1991. Untersuchungen zu dynamischen neuronalen netzen. *Diploma thesis, T.U. Munich*.

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

Alon Jacovi, Oren Sar Shalom, and Yoav Goldberg. 2018. Understanding convolutional neural networks for text classification. *arXiv preprint arXiv:1809.08037*.

Sarthak Jain and Byron C. Wallace. 2019. Attention is not explanation.

Rie Johnson and Tong Zhang. 2016. Supervised and semi-supervised text categorization using lstm for region embeddings. In *ICML*.

Urvashi Khandelwal, He He, Peng Qi, and Daniel Jurafsky. 2018. Sharp nearby, fuzzy far away: How neural language models use context. In *ACL*.

Valentin Khrulkov, Alexander Novikov, and Ivan Oseledets. 2017. Expressive power of recurrent neural networks.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *AAAI*.

Quoc V. Le, Navdeep Jaitly, and Geoffrey E. Hinton. 2015. A simple way to initialize recurrent networks of rectified linear units.

Shuai Li, Wanqing Li, Chris Cook, Ce Zhu, and Yanbo Gao. 2018. Independently recurrent neural network (indrnn): Building a longer and deeper rnn.

Qi Liu, Matt J. Kusner, and Phil Blunsom. 2020. A survey on contextual embeddings.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Avinash Madasu and Vijjini Anvesh Rao. 2019. Sequential learning of convolutional features for effective text classification.

Pratyush Maini, Keshav Kolluru, Danish Pruthi, and Mausam. 2020. Why and when should you pool? analyzing pooling in recurrent architectures.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2017. Regularizing and optimizing lstm language models. *arXiv preprint arXiv:1708.02182*.

William Merrill, Gail Weiss, Yoav Goldberg, Roy Schwartz, Noah A. Smith, and Eran Yahav. 2020. A formal hierarchy of rnn architectures.

Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 188–197, Hong Kong, China. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D 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.

Danish Pruthi, Mansi Gupta, Bhuwan Dhingra, Graham Neubig, and Zachary C Lipton. 2020. Learning to deceive with attention-based explanations. *The 58th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Anton Maximilian Schäfer and Hans Georg Zimmermann. 2006. Recurrent neural networks are universal approximators. In *Proceedings of the 16th International Conference on Artificial Neural Networks - Volume Part I*, ICANN'06, page 632–640, Berlin, Heidelberg. Springer-Verlag.

Yikang Shen, Shawn Tan, Alessandro Sordoni, and Aaron Courville. 2018. Ordered neurons: Integrating tree structures into recurrent neural networks.

Corentin Tallec and Yann Ollivier. 2018. Can recurrent neural networks warp time?

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

Lyan Verwimp, Hugo Van hamme, Vincent Renkens, and Patrick Wambacq. 2018. State gradients for rnn memory analysis. In *BlackboxNLP@EMNLP*.

Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. 2013. Regularization of neural networks using dropconnect. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1058–1066, Atlanta, Georgia, USA. PMLR.

Gail Weiss, Yoav Goldberg, and Eran Yahav. 2018. On the practical computational power of finite precision rnns for language recognition.

Jos van der Westhuizen and Joan Lasenby. 2018. The unreasonable effectiveness of the forget gate.

Sarah Wiegreffe and Yuval Pinter. 2019. Attention is not not explanation. *Proceedings of the 2019 conference on Empirical Methods in Natural Language Processing, EMNLP*.

Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V. Le. 2019. Unsupervised data augmentation for consistency training.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016a. 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.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016b. 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, San Diego, California. Association for Computational Linguistics.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pages 649–657, Cambridge, MA, USA. MIT Press.

Ye Zhang and Byron Wallace. 2015. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820*.

THESIS COMMITTEE

ADVISOR : Dr. Mausam
Professor
Department of Computer Science and Engineering
IIT Delhi

MEMBERS : Dr. Parag Singla
Associate Professor
Department of Computer Science and Engineering
IIT Delhi

Dr. Prathosh A.P.
Assistant Professor
Department of Electrical Engineering
IIT Delhi