Project Report

Sentence Classification using Densely Connected Bidirectional LSTM

Md Tahmid Yasar
Khalid Saifullah

Abstract

Deep Neural Networks are known for their performance due to the flexibility of exploration in larger hypothesis space. However, deep architectures such as Recursive Neural Networks have vanishing gradient and overfitting problems, which is why they are often underappreciated in many Natural Language Processing applications. In this project, we have modeled the paper [1] which proposes a novel multi-layer RNN model called densely connected bidirectional long short-term memory (DC-Bi-LSTM). Each layer in our proposed DC-Bi-LSTM network is represented by the concatenation of its hidden state and the hidden states of all the preceding layers, followed by recursively passing each layer's representation to all subsequent layers.

We have evaluated our sentence classifier model on a total of seven different datasets including five benchmark datasets of sentence classification which will be described as we proceed. This model that we have built exceeds in terms of performance when compared with the performance of the state-of-the-art approaches.

1. Introduction

Speech and sentences are sequential data. In order to classify sentences accurately, we need to take the sequence or order of the word into account. The recurrent neural network (RNN) was developed to solve the problem of classifying such sequential data.

In this project, we have built a sentence classifier using densely connected bidirectional LSTM which can address the overfitting issues and vanishing gradient problems while performing better than many of the state-of-the-art approaches. Our first step was to preprocess the sentence datasets so that it can be given as input to the deep learning architecture. We shall remove the punctuations and convert the words into one-hot encoded vectors using pre-trained dictionary. In our model, we will be using the publicly available 300-dimensional Glove vectors [3] which were trained on 42 billion words. Deep-stacked RNNs [4] are created by stacking multiple RNN hidden layers on top of each other, The output sequence of one layer forming the input sequence for the next. Densely Connected Bidirectional Long Short Term Memory (DC-Bi-LSTM) has been proposed where the input sequence of each layer is the concatenation of the output of the previous layer and inputs of all the previous layers, which alleviates the issues of vanishing-gradient. We will evaluate the DC-Bi-LSTM on five benchmark datasets mentioned in the paper. Moreover, we will also implement the model on another dataset to verify the performance. We have also built and evaluated the performance of Bi-LSTM, which is considered as the baseline, with our model. In addition, we have built the deep stacked Bi-LSTM model (DS-Bi-LSTM) and compare the performance with the proposed model which is not mentioned in the paper originally.

2. Dataset and Data Preprocessing

Our model is evaluated on a total of seven datasets of sentence classification. The CR and MPQA datasets used to evaluate our model were not mentioned originally in the paper. They are included along with the five datasets presented in the paper in order to strengthen the justification for the model's superior performance.

i) MR: Movie Review Data [6]
ii) SST-1: Stanford Sentiment Treebank [7]
iii) SST-2: Same as ii) but used in a binary mode without neutral sentences [7]
iv) Subj: Subjectivity Dataset [8]
v) TREC: Question Type Classification Task [9]
vi) CR: Customer reviews of various products [10]
vii) MPQA: manually annotated for opinions and other private states [11]

Table 1: Summary of the datasets used in the model evaluation

Data	**No. of Classes**	**Avg. Sentence Length**	**Dataset Size**	**Vocab size**
MR	2	20	10662	18765
SST-1	5	18	11855	17836
SST-2	2	19	9613	16185
Subj	2	23	10000	21323
TREC	6	10	5952	9592
CR	2	19	3775	5340
MPQA	2	3	10606	6246

The first step in our method is the sentence pre-processing, which is converts the sentences in our datasets to a form that is suitable for a sentiment analysis system. These pre-processing tasks include punctuation removal, Latin characters removal, stop word removal, digits removal, tokenization. The purpose is to minimize the ambiguity of words to increase the accuracy and the effectiveness of our model. [2]

Tokenization is a process which separates the texts into tokens. Often, words are separated from each other by blanks (white space, semicolons, commas, quotes, and periods). These tokens may represent individual words (noun, verb, pronoun, and article, conjunction, preposition, punctuation, numbers, and alphanumeric) which are converted without any prior comprehension of their meaning or relationships. So, the list of tokens becomes an input for further processing.

Also, note that traditional methods are commonly based on the bag-of-words (BoW) model, which treats sentences as unordered collections and hence fails to capture the syntactic structures and contextual information. Recently, deep learning has developed rapidly in natural language processing, the first breakthrough is learning word embedding. In our model, we have used the Glove vectors as the word embedding [3]. For each of the datasets used, dictionaries are built with the unique vocabs of the datasets and then the glove vectors file is loaded, and the corresponding filtered vectors are saved in jason files to be used as input to the RNN layers. The labels are one hot encoded. The datasets are splitted into 80% training, 10% validation and 10% test sets after shuffling.

3. Network Architecture

In this project, we have built a Bi-LSTM, Deep Stacked Bi-LSTM in addition to the Densely Connected Bi-LSTM which is proposed in the paper we have chosen to implement. These three architectures are discussed below along with their implementation details.

Bi-LSTM

We use RNN architectures like LSTM and Bi-LSTM when the learning problem is sequential, for instance, we have a video and we want to know what is it all about or we want to generate the text from an image of text. Bi-LSTM is popular because as it can learn how and when to forget by using gates in their architecture.

A Bidirectional LSTM, or Bi-LSTM, is a sequence processing model that consists of two LSTMs where one takes the input in a forward direction, and the other in a backward direction. Bi-LSTMs can essentially increase the amount of information available to the network, thereby improving the context available to the algorithm (e.g. knowing which words immediately follow and precede a word in a sentence). The Figure 1 below shows the Bi-LSTM architecture used in our model.

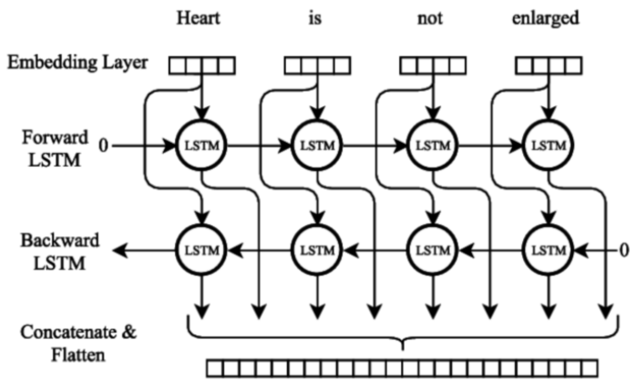


Figure 1: Architecture of Bi-LSTM [12]

Let's take a closer look at how this Bi-LSTM is implemented from start. Given an arbitrary-length input sentence S = $\{w_1, w_2, \dots, w_s\}$, Long Short-Term Memory (LSTM) [5] computes the hidden states h = $\{h_1, h_2, \dots, h_s\}$ by iterating the following equations:

$$\overrightarrow{h_t} = \mathrm{lstm}(\overrightarrow{h_{t-1}}, \mathrm{e}(w_t))$$

The detailed computation is as follows:

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \mathrm{sigm} \\ \mathrm{sigm} \\ \mathrm{sigm} \\ \mathrm{sigm} \end{pmatrix} T_{m+d,4d}([\mathrm{e}(w_t); \overrightarrow{h_{t-1}}]),$$

$c_t = f \odot c_{t-1} + i \odot g$

$h_t = o \odot \tanh(c_t)$,

Where e(w_t) ∊ R^m is the word embedding of w_t, h_{t-1} ∊ R^d is the hidden state of LSTM at time step t-1, [e(w_t); $\overrightarrow{h_{t-1}}$] ∊ R^{m+d} is the concatenation of two vectors. $T_{m+d,4d}$: R^{m+d} → R^{4d} is an affine transform (Wx + b for some weight and bias). The sigm and tanh are sigmoid and hyperbolic tangent activation functions, *i, f, o, g* ∊ R^d are the input gate, forget gate, output gate and new candidate memory state respectively. Particularly, c_t ∊ R^d is the additional memory cell used for capturing long distance dependencies, ⊙ denotes elementwise multiplication.

The calculations of Bi-LSTMs can be formulated as follows:

$\overrightarrow{h_t}$ = lstm ($\overrightarrow{h_{t-1}}$, e(w_t))

$\overleftarrow{h_t}$ = lstm ($\overleftarrow{h_{t-1}}$, e(w_t))

Then the concatenation of forward and backward hidden states is taken as the representation of each word. For word w_t, the representation is denoted as h_t = [$\overrightarrow{h_t}$; $\overleftarrow{h_t}$]

DS-Bi-LSTM

Deep Stacked Bi-LSTM [4] uses multiple Bi-LSTMs with different parameters in a stacking fashion as shown in Figure 2. The hidden state of l-layer Bi-LSTM can be represented as h_t^l, which is the concatenation of forward hidden state $\overrightarrow{h_t^l}$ and backward hidden state $\overleftarrow{h_t^l}$. The calculation of h_t^l is as follows:

h_t^l = [$\overrightarrow{h_t^l}$; $\overleftarrow{h_t^l}$], specially $h_t^o = e(w_t)$,

$\overrightarrow{h_t^l}$ = lstm($\overrightarrow{h_{t-1}^l}$, h_t^{l-1}),

$\overleftarrow{h_t^l}$ = lstm($\overleftarrow{h_{t+1}^l}$, h_t^{l-1}).

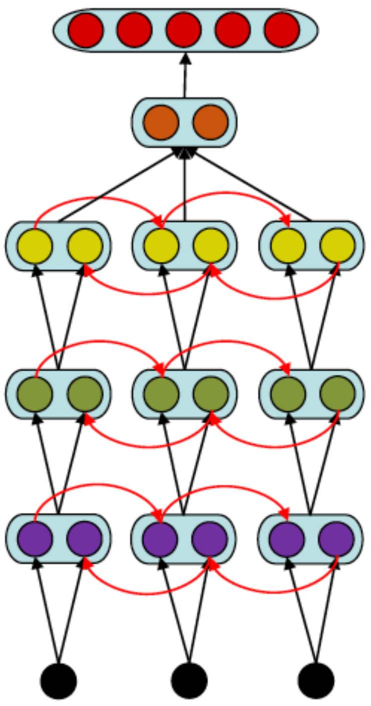


Figure 2: Illustration of Deep Stacked Bi-LSTM

DC-Bi-LSTM

The architecture of DC-Bi-LSTM is given in Figure 3, where we use Bi-LSTM to encode the input sequence and regard the sequence of hidden states as reading memory for each layer. Essentially, we obtain first-layer reading memory based on original input sequence, and second layer reading memory based on the position-aligned concatenation of original input sequence and first layer reading memory, and so on. Finally, based on the concatenation of original input sequence and all previous reading memory, we get the n-th layer reading memory, which is then taken as the final feature representation for classification.

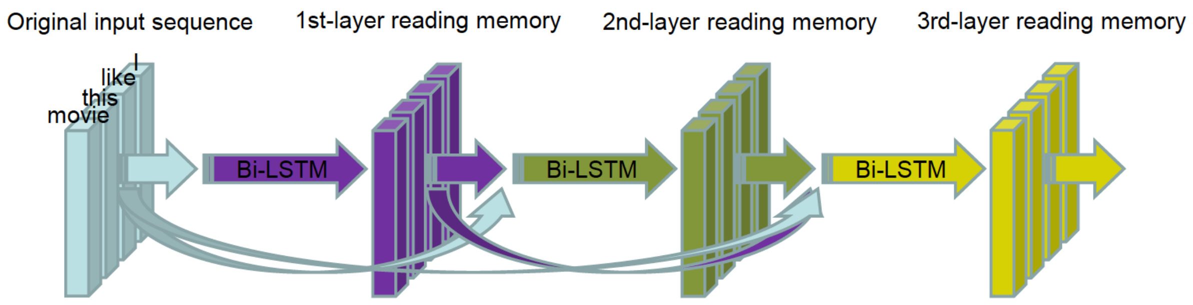


Figure 3: Architecture of Densely Connected Bi-LSTM

The Densely Connected Bi-LSTM (DC-Bi-LSTM), proposed approach in the paper which we are implementing, consists of four modules: network inputs, dense Bi-LSTM, average pooling and soft-max layer as seen from Figure 4.

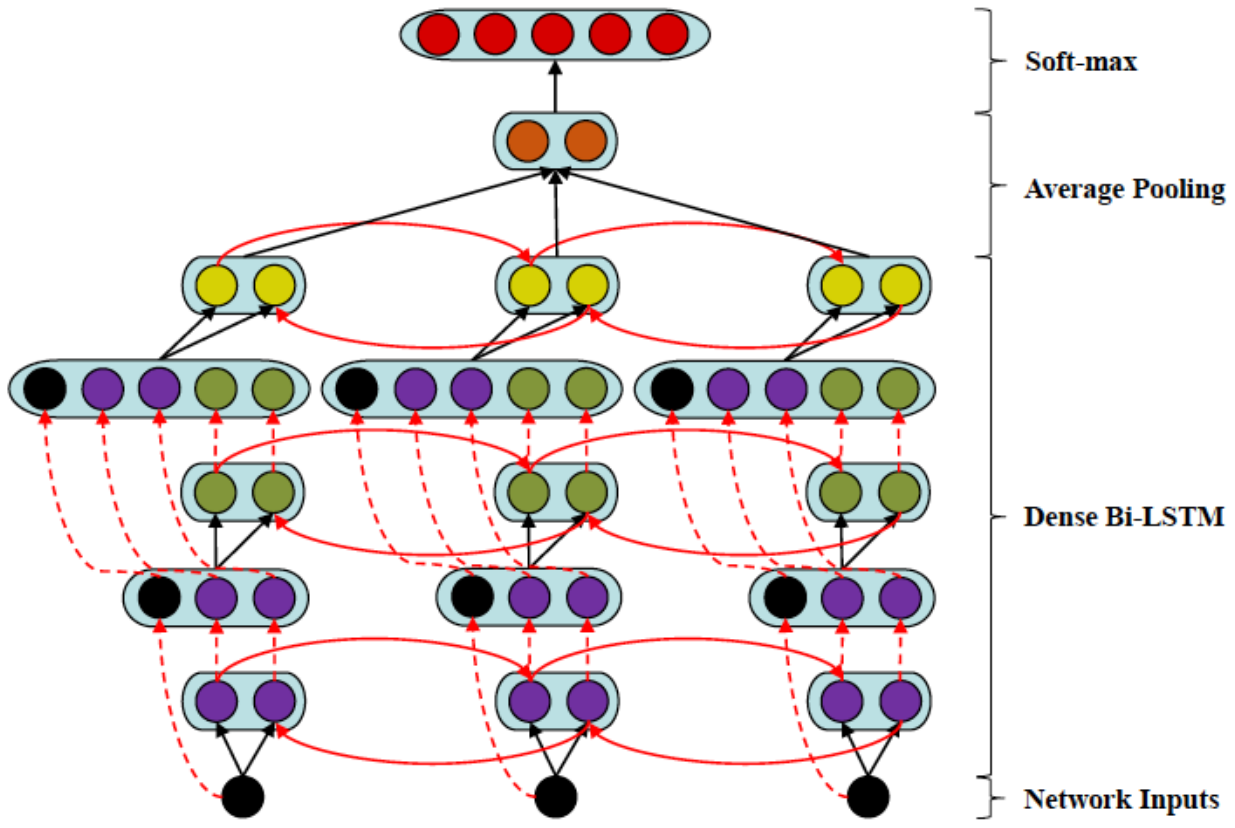


Figure 4: Illustration of Densely Connected Bi-LSTM

The four modules of Densely Connected Bi-LSTM (DC-Bi-LSTM) are explained below in brief.

(1) Network Inputs

The input of our model can be a variable-length sentence, which can be represented as S = $\{w_1, w_2, \dots, w_s\}$. Like other deep learning models, each word is represented as a dense vector extracted from a word embedding matrix. Finally, a sequence of word vectors $\{e(w_1), e(w_2), \dots, e(w_s)\}$ is sent to the dense Bi-LSTM module as inputs. In our model, we have used the publicly available 300-dimensional Glove vectors [3], as our word embedding, which were trained on 42 billion words.

(2) Dense Bi-LSTM

This module consists of multiple Bi-LSTM layers. For the first Bi-LSTM layer, the input is a vector sequence $\{e(w_1), e(w_2), \dots, e(w_s)\}$, and the output is $h^l = \{h_1^l, h_2^l, \dots, h_s^l\}$, where $h_t^l = [\overrightarrow{h_t^l}\ ; \overleftarrow{h_t^l}]$ as described in the Bi-LSTM portion above. For the second Bi-LSTM layer, the input is not the sequence $h^l = \{h_1^l, h_2^l, \dots, h_s^l\}$, (the way stacked RNNs use), but the concatenation of all previous outputs, formulated as $\{[e(w_1);\ h_1^1;\ h_1^2], [e(w_2);\ h_2^1;\ h_2^2], \dots, [e(w_s);\ h_s^1;\ h_s^2]\}$, and the output is $h^2 = \{h_1^2, h_2^2, \dots, h_s^2\}$. For the third layer, whose input is $\{[e(w_1);\ h_1^1], [e(w_2);\ h_2^1], \dots, [e(w_s);\ h_s^1]\}$, works like the second layer. The rest layers process similarly. The above process is formulated as follows:

$h_t^l = [\overrightarrow{h_t^l}\ ; \overleftarrow{h_t^l}]$, specially $h_t^o = e(w_t)$,

$$\overrightarrow{h_t^l} = \text{lstm}(\overrightarrow{h_{t-1}^l}, M_t^{l-1}),$$
$$\overleftarrow{h_t^l} = \text{lstm}(\overleftarrow{h_{t+1}^l}, M_t^{l-1}),$$
$$M_t^{l-1} = [h_t^o;\ h_t^l;\ \dots;\ h_s^{l-1}]$$

(3) Average Pooling
For a L layer Dense Bi-LSTM, the output is $h^L = \{h_1^L, h_2^L, \dots, h_s^L\}$. Average pooling module reads in h^L and calculate the average value of these vectors and the computation can be formulated as h^* = average ($h_1^L, h_2^L, \dots, h_s^L$).

(4) Soft-max
This module is a simple soft-max classifier, which takes h^* as features and generates predicted probability distribution over all sentence labels.

4. Training Strategy and Evaluation

The preprocessed data were fed into LSTM-based models for sentence classification. The words are converted into sequences of numbers following the embedding described in Section 2. The labels are converted to one-hot vectors before feeding into the classifier.

The sentence is sequential data and the order of the words in a sentence carries significant importance for classification. LSTM-based recurrent neural networks have been used successfully for sequence classification. Unlike most of the numerical sequences, the order of the words in a sentence influences both directions. The first model we implemented is Bi-directional-LSTM.

With added layers, the classifier achieves better differentiability like any other deep learning architecture. The immediate drawback of an additional layer is the vanishing gradient problem. In the backpropagation algorithm, the gradient gets multiplied from one layer to the next. The learning process becomes slower with each added layer. We implemented the DS-Bi-LSTM architecture, with a fewer layers and parameters to achieve reasonable performance within comparable execution time.

The improvement presented in the paper with DC-Bi-LSTM achieves peak performance in fewer epochs than DS-Bi-LSTM. Also, in this architecture, the number of layers can be increased significantly without compromising computation time. The hyperparameters used for these three architectures are given in the following Table 2.

Table 2: Hyperparameters of the models after tuning

Hyperparameters	Used value
Learning rate	0.01
Learning rate decay	0.05

Optimizer	Adam
Clip value	5
Dropout keep probability	0.5

The network dimensions for the three models implemented are given in the following Table 3.

Table 3: Dimensions of network architecture

Dimensions	Bi-LSTM	DS-Bi-LSTM	DC-Bi-LSTM
Number of layers	1	5	15
Number of units, up to penultimate layer	0	39	13
Number of units in last layer	300	100	100
Number of parameters	1582306	392834	1445554

5. Result Comparison and Problems Encountered

Results of the three implemented models on five benchmarks and two additional datasets are listed in Table 4. For comparative analysis, the performance achieved by the reference paper is also shown in Table 4. Along with the highest accuracy, the epoch at which the best accuracy is achieved is tabulated.

Table 4: Comparative performance analysis of maximum accuracy and corresponding epochs.

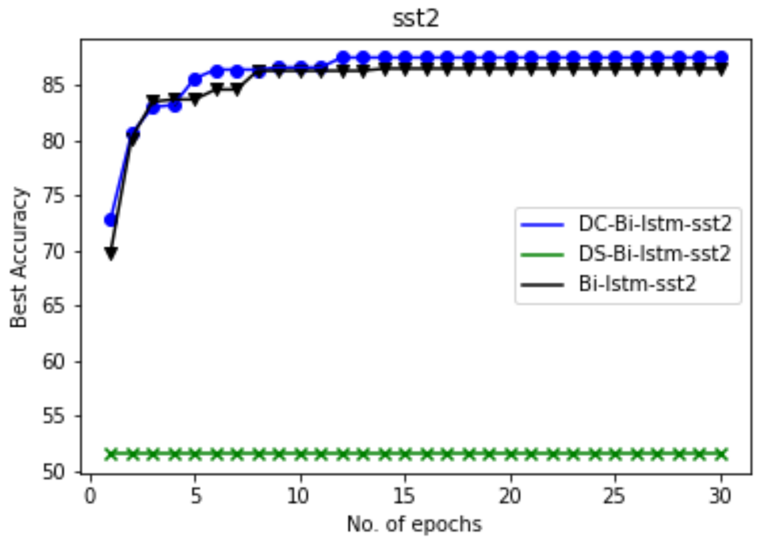
Model	**MR**	**SST-1**	**SST-2**	**Subj**	**TREC**	**CR**	**MPQA**
Bi-LSTM	81.2 (8th epoch)	46.4 (8th epoch)	86.5 (14th epoch)	94.2 (8th epoch)	94.8 (20th epoch)	81.6 (10th epoch)	87.3 (17th epoch)
DS- Bi-LSTM	82.4 (25th epoch)	46.0 (33rd epoch)	86.5 (30th epoch)	94.0 (18th epoch)	91.3 (30th epoch)	81.0 (25th epoch)	87.2 (21st epoch)
DC-Bi-LSTM (ours)	81.9 (20th	47.2 (23rd	87.5 (12th	95.1 (8th	94.7 (15th	80.7 (23rd	86.7 (6th

	epoch)	epoch)	epoch)	epoch)	epoch)	epoch)	epoch)
DC-Bi-LSTM (paper)	82.8	51.9	89.7	94.5	95.6		

The evaluation results on some datasets are slightly lower than those proposed in the paper may be caused by data processing, different parameter settings (like batch size, learning rate, learning rate decay, grad clip, char emb and etc.), or without applying cross-validation.

DC-Bi-LSTM performs the best on MR, SST-1, SST-2, and Subj data. The maximum accuracy of TREC for DC-Bi-LSTM and Bi-LSTM are very close. Bi-LSTM achieved the highest accuracy on the two datasets (CR and MPQA) that were not used in the paper.

DS-Bi-LSTM performed relatively poorly compared to Bi-LSTM and DC-Bi-LSTM. It is noteworthy that the number of parameters for DS-Bi-LSTM was kept significantly lower than the other two models. When we set the number of parameters of DS-Bi-LSTM compatible to other models, the performance was unacceptable.



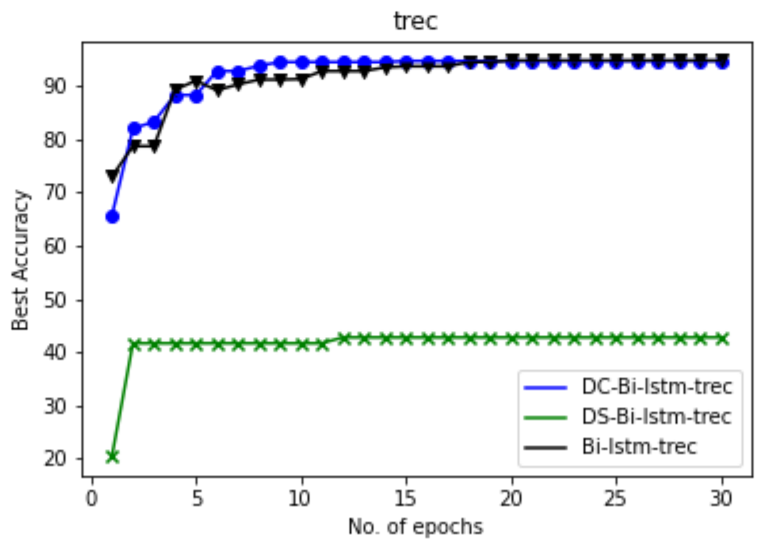
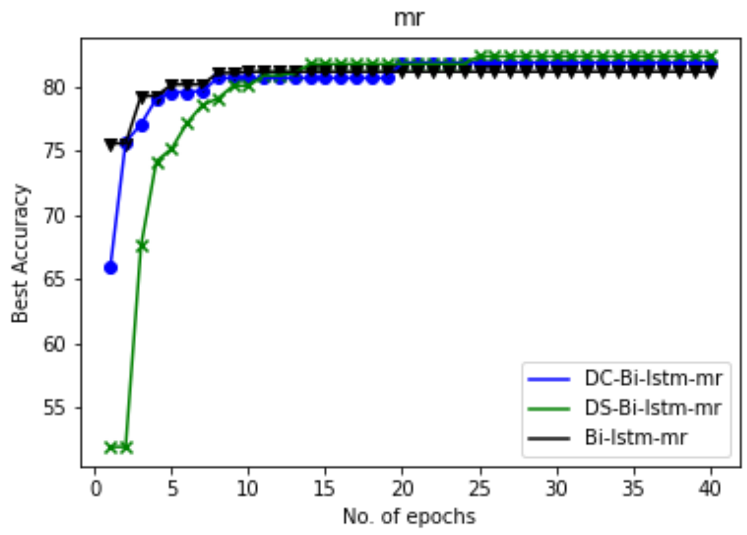
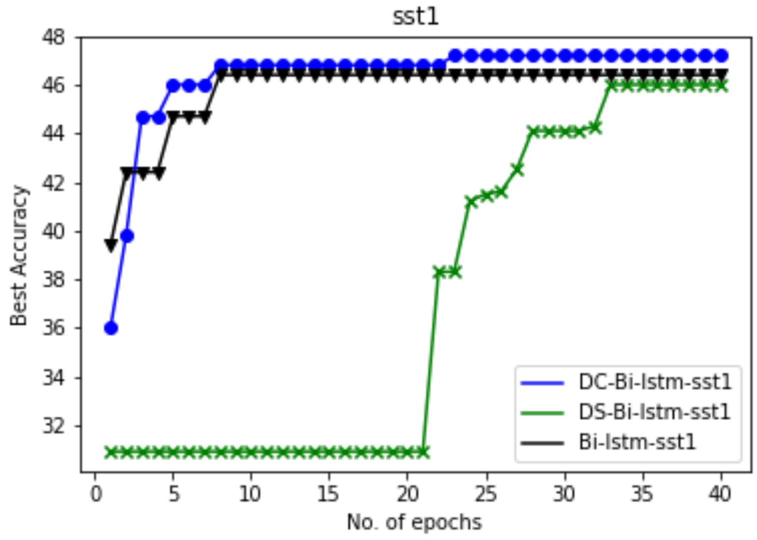
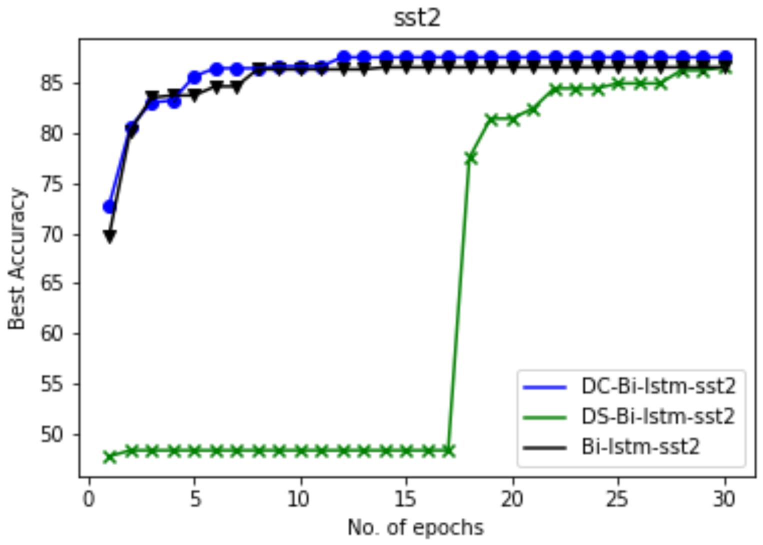


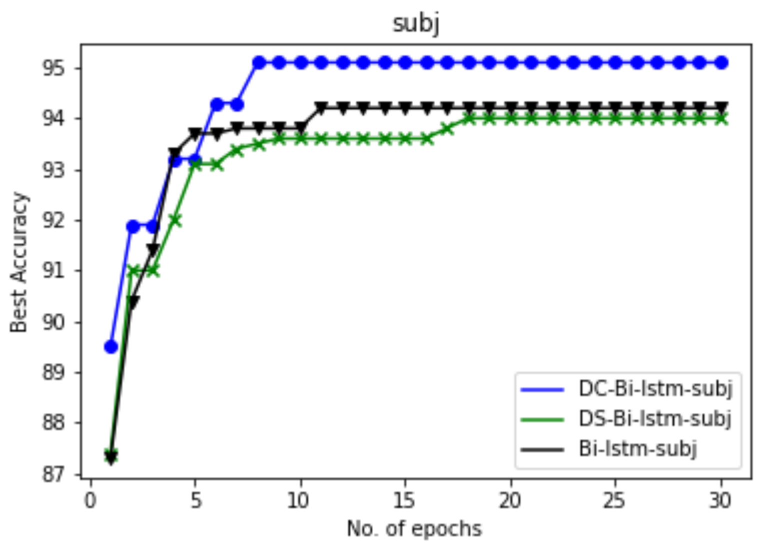
Figure 5: Best accuracy vs number of epochs for the models Bi-LSTM, DS-Bi-LSTM (Number of units, up to penultimate layer = 85 and Number of units in the last layer =200) DC-Bi-LSTM on the datasets, (a) SST-2 and (b) TREC

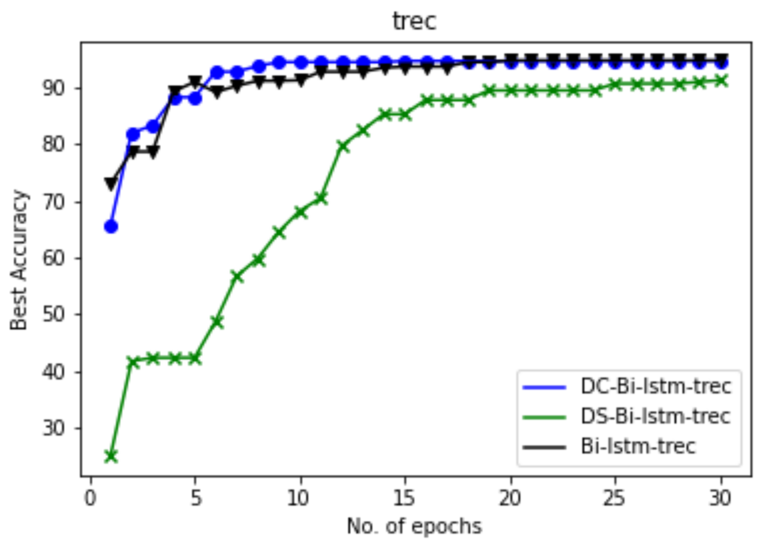
Figure 5 illustrates that DS-Bi-LSTM is not scalable as Bi-LSTM and DC-Bi-LSTM.

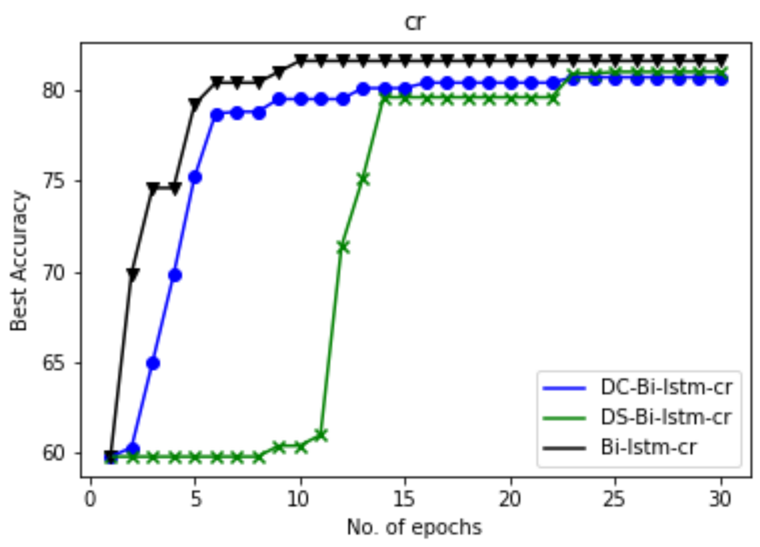

mr
Best Accuracy
No. of epochs
DC-Bi-lstm-mr
DS-Bi-lstm-mr
Bi-lstm-mr


sst1
Best Accuracy
No. of epochs
DC-Bi-lstm-sst1
DS-Bi-lstm-sst1
Bi-lstm-sst1


sst2
Best Accuracy
No. of epochs
DC-Bi-lstm-sst2
DS-Bi-lstm-sst2
Bi-lstm-sst2


subj
Best Accuracy
No. of epochs
DC-Bi-lstm-subj
DS-Bi-lstm-subj
Bi-lstm-subj


trec
Best Accuracy
No. of epochs
DC-Bi-lstm-trec
DS-Bi-lstm-trec
Bi-lstm-trec


cr
Best Accuracy
No. of epochs
DC-Bi-lstm-cr
DS-Bi-lstm-cr
Bi-lstm-cr

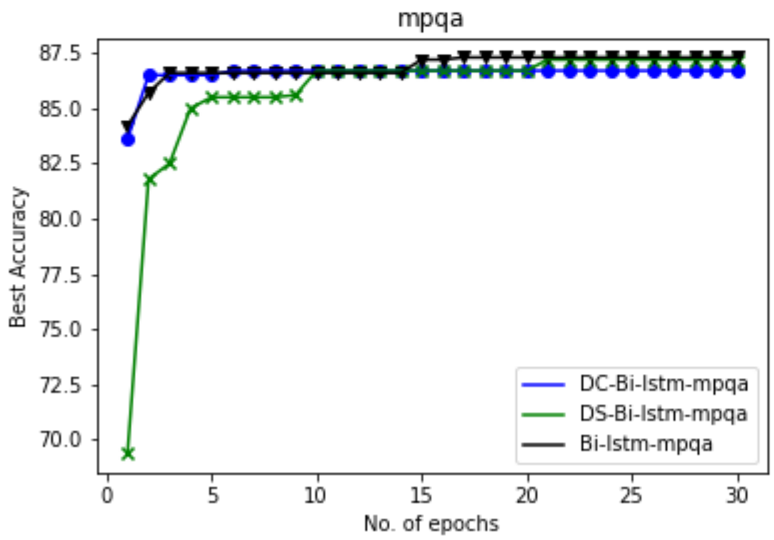


Figure 6: Best accuracy vs number of epochs for the models Bi-LSTM, DS-Bi-LSTM DC-Bi-LSTM on the datasets, (a) MR, (b) SST-1, (c) SST-2, (d) Subj, (e) TREC, (f) CR and (g) MPQA

Figure 6(a-g) illustrates the performance of three models at each epoch after tuning. It is evident that, Bi-LSTM achieves the highest accuracy faster than DS-Bi-LSTM and that validates the vanishing gradient problem of complex models with multiple layers. DC-Bi-LSTM aims to alleviate the problem and Figure 6(a-g) illustrates that.

6. Discussion and Conclusion

In this project, we have explored sentence classification using state LSTM based recurrent neural networks following the work of [1]. To utilize order information in both forward and reverse path, Bi-directional LSTM based models are used for classification. The model is enhanced with an increasing number of layers to the architecture. But the drawback of gradient vanishing rendered the DS-Bi-LSTM impractical. To overcome the issue, the input of each layer is fed as an additional input in the next layer. The DC-Bi-LSTM performs better and faster than DS-Bi-LSTM. The enhancement opens the door to build sophisticated models where traditional classifiers fail. It is noteworthy that the two datasets that were not evaluated in [1] showed the best performance with Bi-LSTM. It indicates that the standard models perform more robustly whereas complex models are more problem specific.

Responsibilities:

Team Member Responsibilities:

1) Md Tahmid Yasar
    - Data preparation and preprocessing
    - Build RNN model (Bi-LSTM, DS-Bi-LSTM, DC-Bi-LSTM)
    - Report and presentation

2) Khalid Saifullah
    - Build RNN model (Bi-LSTM, DS-Bi-LSTM, DC-Bi-LSTM)
    - Performance evaluation and comparison
    - Report and presentation

References

[1] https://arxiv.org/pdf/1802.00889.pdf
[2] https://github.com/IsaacChanghau/Dense_BiLSTM/blob/master/dataset/prepro.py
[3] https://nlp.stanford.edu/projects/glove/
[4] https://arxiv.org/pdf/1303.5778.pdf
[5] https://www.bioinf.jku.at/publications/older/2604.pdf
[6] https://www.cs.cornell.edu/people/pabo/movie-review-data/
[7] https://nlp.stanford.edu/sentiment/
[8] https://www.cs.cornell.edu/people/pabo/movie-review-data/
[9] https://cogcomp.seas.upenn.edu/Data/QA/QC/
[10] https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html
[11] https://mpqa.cs.pitt.edu/
[12] https://arxiv.org/abs/1609.08409