BOHR International Journal of Computer Science
2022, Vol. 1, No. 1, pp. 31–36
https://doi.org/10.54646/bijcs.005
www.bohrpub.com

Hindi/Bengali Sentiment Analysis using Transfer Learning and Joint Dual Input Learning with Self Attention

Shahrukh Khan and Mahnoor Shahid

E-mail: shkh00001@stud.uni-saarland.de; mash00001@stud.uni-saarland.de

Abstract. Sentiment Analysis typically refers to using natural language processing, text analysis, and computational linguistics to extract effect and emotion-based information from text data. Our work explores how we can effectively use deep neural networks in transfer learning and joint dual input learning settings to effectively classify sentiments and detect hate speech in Hindi and Bengali data. We start by training Word2Vec word embeddings for Hindi HASOC dataset and Bengali hate speech [1] and then train LSTM and subsequently, employ parameter sharing based transfer learning to Bengali sentiment classifiers by reusing and fine-tuning the trained weights of Hindi classifiers with both classifiers being used as the baseline in our study. Finally, we use BiLSTM with self-attention in a joint dual input learning setting where we train a single neural network on the Hindi and Bengali datasets simultaneously using their respective embeddings.

Keywords: Deep Learning, Transfer Learning, LSTMs, Sentiment Analysis, Opinion Mining.

INTRODUCTION

There have been certain huge breakthroughs in the field of Natural Language Processing paradigm with the advent of attention mechanism and its use in transformer sequence-sequence models [2] coupled with different transfer learning techniques have quickly become state-of-the-art in multiple pervasive Natural Language Processing tasks such as classification, named entity recognition etc [3, 4]. In our work we propose, a novel self-attention-based architecture for sentiment analysis and classification on **Hindi HASOC dataset** [1] here we use sub-task A which deals with whether a given tweet has hate speech or not. Moreover, this also serves as a source domain in the subsequent task-specific transfer learning task, where, we take the learned knowledge from the Hindi sentiment analysis domain to a similar binary **Bengali sentiment analysis** task.

Given the similar nature of both Bengali and Hindi sentiment analysis tasks (i.e., binary classification), we conceptualized the problem as joint dual input learning setting on top of the work of Zhouhan Lin et al. [5] where they suggested how we can integrate self-attention with BiLSTMs and have a hidden representation containing different aspects for each sequence which results in sentence embeddings whilst performing sentiment analysis and text classification more broadly. One significant beneficial side effect of using such an approach is that the attention attributions can easily be visualized which implies we can see what portions of the sequence attention mechanism have put more impetus on via its generated summation weights, this visualization technique played a pivotal role in selecting the number of attention hops r also referred to as how many attention vectors of summation weights for each sequence in our study. Moreover, we employed this approach in a joint dual input learning setting where we have a single neural network that is trained on Hindi and Bengali data simultaneously.

Our proposed approach which is a variant of the multitask learning offers an alternative framework for training text classification based neural networks on low resource corpora. Also, since we train a single joint network on multiple tasks simultaneously, it essentially allows for better generalization and task specific weight-sharing which can be a reasonable alternative for pre-training based transfer learning approaches.

EXPERIMENTAL SETTING

Word Embeddings

Starting with the Hindi dataset, we prepared the training dataset in which employed sub-sampling technique in

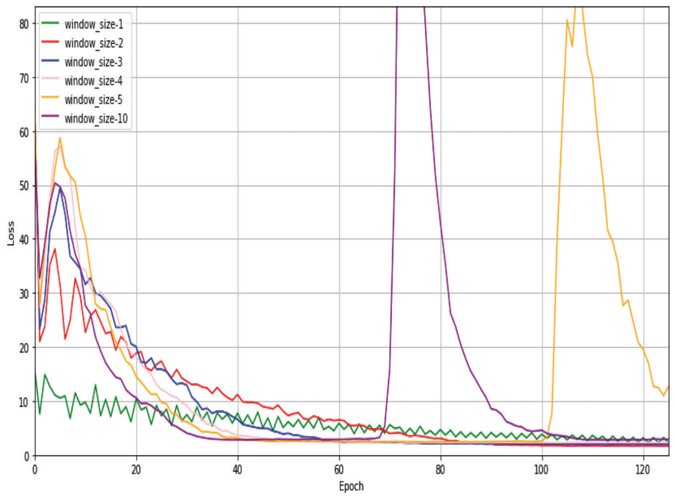

Figure 1. Train loss convergence for different values of window size with fixed embedded size = 300 Hindi Dataset.

which we first computed the probability of keeping the word using the following formula:

$$P_{keep}(w_i) = \left(\sqrt{\frac{z(w_i)}{0.000001}} + 1\right) \times \frac{0.000001}{z(w_i)}$$

Where $z(w_i)$ is the relative frequency of the word in the corpus. Hence we used $P_{keep}(w_i)$ for each context word while sampling context words for a given word and randomly dropped frequent context words by comparing them against a random threshold sampled each time from a uniform distribution, since if we kept all the frequent words in our context for training data, we may not get the rich semantic relationship between the domain-specific words since frequent words like "the", "me" etc don't necessarily carry much semantic meaning in a given sequence. Hence dropping randomly dropping them made more sense as compared to keeping or dropping all of them. Also, another important design decision that we made here was to curate the train set for Word2Vec only once before training the model as opposed to creating a different one for each epoch as we were randomly sub-sampling context words because the earlier mentioned approach gives faster execution time for training the model while the model also converged well to a relatively low train loss value as well. Furthermore, for choosing hyper-parameters we performed the following analysis.

As it is apparent from the above visualization WordVec models with smaller context windows converged faster and had better train loss at the end of the training process. However, in order to retain some context-based information we selected the window size 2 as it has contextual information as well the model had better train loss.

After testing different values for hyper-parameters with different combinations, this was observed that for the better performance of the model, they should be set to

Table 1. Different combinations of hyperparameters from Hindi Dataset.

Embedded Size	Learning Rate	Window Size	Min Loss Score
300	0.05	1	0.841
		2	1.559
		3	1.942
		4	2.151
		5	2.321
		10	2.792
	0.01	1	1.298
		2	3.295
		10	2.747
	0.1	1	1.311
		2	1.557
		10	3.551

Epochs = 500, Window Size = 2, Embedded Size = 300, and Learning Rate = 0.05 in the case of our study.

Also, we have set **Cross Entropy Loss** as the criterion used for adjusting the weights during the training phase. When softmax converts logits into probabilities then, Cross-Entropy takes those output probabilities (p) and measures the distance from the truth values to estimate the loss. Cross entropy loss inherently combines **log softmax and negative log-likelihood loss** so we didn't apply log softmax on the output of our Word2Vec model.

For optimization, we have selected Adam (Adaptive Moment Estimation algorithm) [6] which is an optimization technique that, at present, is very much recommended for its computational efficiency, low memory requirement, invariant to diagonal rescale of the gradients, and extremely better results for problems that are large in terms of data/parameters or for problems with sparse gradients. Adam provides us with the combination of best properties from both AdaGrad and RMSProp and is often used as an alternative for SGD + Nesterov Momentum as proposed by adam [6].

Baseline Models

For the choice of baseline, we reproduced the work by Jenq-Haur Wang et al. [7] which primarily focuses on performing sentiment classification on short social media texts using long short-term memory neural networks using distributed representations of Word2Vec learned using Skip-gram approach. We chose to reproduce their work for our baseline as they also were using Word2Vec Skip-gram based distributed representation of words and also since our datasets were also sourced from social media. Moreover, the neural network LSTM is an upgraded variant of the RNN model, that serves as the remedy to some extent of the problems that require learning long-term temporal dependencies; due to vanishing gradients, since LSTM uses a gate mechanism and memory cell to control the memorizing process.

Hindi Neural Sentiment Classifier Baseline

We then implemented the architecture for the LSTM classifier which used pre-trained 300-dimensional word embeddings obtained as described in section "Word Embeddings". We used the Adam optimizer with the initial learning rate of 10^{-4} which helped the train and validation loss to converge at a relatively fast rate, the optimizer didn't optimize the weights of the embedding layer via gradient optimization since they were pre-trained already. Moreover, we chose the binary cross-entropy loss function as we are doing binary classification. In model architecture, we used 8 layers of LSTMs with each having a hidden dimension of 64 followed by a dropout layer with a dropout probability of 0.5 to counterbalance over fitting and finally fully connected output layer wrapped by a sigmoid activation function since our target is binary and sigmoid is the ideal choice for binary classification given its mathematical properties. We kept a batch size of 32 and trained the model for 30 epochs while monitoring its accuracy and loss on the validation set. The choice of hyper-parameters was made after trying different combinations and we chose the best set of hyper-parameters while monitoring the validation set accuracy.

Bengali Neural Transfer Learning Based Sentiment Classifier Baseline

Similarly to the Hindi sentiment classification pipeline, we first obtained the word embeddings for Bengali data using the Word2Vec skip-gram approach, the same set of hyper-parameters that we chose for the Hindi dataset, worked fine here well, so we didn't tune the hyper-parameters here, as the model's train loss converged to the similar value we had for the Hindi dataset. Subsequently, we then same the architecture for LSTM based classifier architecture as explained in section "Hindi Neural Sentiment Classifier Baseline". Since our goal here was to perform transfer learning and re-use and fine-tune the learned weights of the Hindi classifier. We replaced the Hindi embeddings layer with a Bengali 300 dimensional embedding layer and also didn't optimize its weights during training. Then loaded the weights from the Hindi classifier for LSTM layers and fully connected layer to apply parameter sharing-based task-specific transfer learning. Additionally, we trained the Bengali classifier for 30 epochs with a batch size of 32 and used the Adam optimizer with an initial learning rate of 10^{-4} while using the binary cross-entropy function for computing loss on the training and validation set. The choice of batch size hyper-parameter was made after trying different values and we chose the best hyper-parameter while monitoring the validation set accuracy. After training the classifier using the pre-trained weights from the Hindi classifier, we got better performance results than the Hindi baseline, this implies task-based transfer learning actually boosted the performance of the Bengali classifier and it performed better.

PROPOSED METHOD

The LSTM based classifier coupled with transfer learning in the Bengali domain does a fairly good job of providing the baselines in our study. However, one main prominent short-coming of Recurrent Neural Network based architectures is they fall short to capture the dependencies between words that are too distant from each other. LSTM's forget gate enables it to retain information about the historical words in the sequence however, it doesn't completely resolve the RNN based networks' vanishing gradients problem. We wanted to investigate whether using self-attention with LSTMs would improve our model's performance. Also, we propose the joint dual input learning setting where both Hindi and Bengali classification tasks can benefit from each other rather than the transfer learning setting where only the target task takes the advantage of pre-training.

Hindi & Bengali Self Attention Based Joint Dual Input Learning BiLSTM Classifier

Instead of training two separate neural networks for Hindi & Bengali, here we simultaneously trained a joint neural network with the same architecture on Hindi and Bengali data in parallel, and optimized its weights using the combined binary cross-entropy loss over Hindi & Bengali datasets respectively, we also added the Hindi and Bengali batches' attention loss to the joint loss in order to avoid overfitting, which we would present in detail in the subsequent sections. Here we switched between the embedding layers based on the language of the batch data. Following is the block architecture we propose.

One major benefit of using such technique is that it increases the model capability of generalization since the size of the training data set roughly doubles given if both languages have an equal number of training examples. Consequently, it reduces the risk of over-fitting.

We propose an extension of the work of [5] where they proposed the method of **"A Structured Self-attentive Sentence Embedding"** on the Hindi dataset. The key idea of that work was to propose document level embeddings by connecting the self-attention mechanism right after a Bi-directional LSTM, which leverages information of both past and future in the sequence as opposed to unidirectional LSTM which only relies on past information in the sequence. The self-attention mechanism results in a matrix of attention vectors which are then used to produce sentence embeddings, each of them equivalent to the length of the sequence and the number of vectors depend on the value of r which is the output dimension of the self-attention mechanism, where each vector is representing how attention mechanism is putting more relative weight on different tokens in the sequence. Following are the key takeaways on how self-attentive document embeddings are produced:

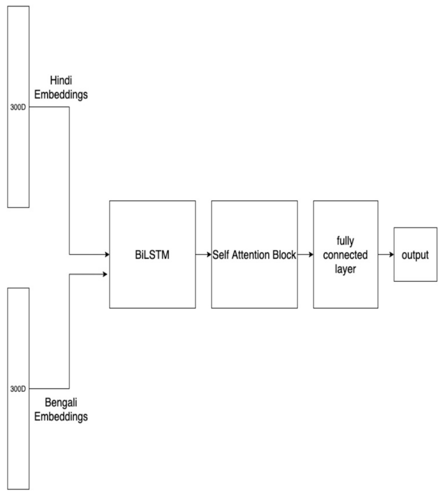

Figure 2. Hindi Bengali Joint Dual Input Learning Architecture.

We start with a input text T of $(n,\ d)$ dimension, where n are the number of tokens, each token is represented by its embedding e in the sequence and d is the embedding dimension.

$$T_{hindi} = [e_1, e_2, e_3, \dots, e_n]$$
$$T_{bengali} = [e_1, e_2, e_3, \dots, e_n]$$

Based on the source language of the input text the corresponding embedding lookup is performed.

Token embeddings are then fed into the BiLSTM, which individually processes each token from left to right and left to the right direction, each BiLSTM cell/layer producing two vectors of hidden states equivalent to the length of the sequence.

$$\begin{aligned}&[[\overrightarrow{h_1}, \overrightarrow{h_2}, \dots, \overrightarrow{h_n}], [\overleftarrow{h_1}, \overleftarrow{h_2}, \dots, \overleftarrow{h_n}]]\\ &= BiLSTM([e_1, e_2, \dots, e_n]; \theta)\end{aligned}$$

Here **H** is the concatenated form of bi-directional hidden states. If there are l LSTM layers/cells then the dimension of **H** is going to be $(n, 2l)$.

$$\mathbf{H} = [[\overrightarrow{h_1}, \overrightarrow{h_2}, \dots, \overrightarrow{h_n}], [\overleftarrow{h_1}, \overleftarrow{h_2}, \dots, \overleftarrow{h_n}]]$$

For self-attention, Zhouhan Lin et al. [5] proposed having two weight matrices, namely W_{s_1} with dimension $(d_a, 2l)$ and W_{s_2} with dimension (r, d_a), here d_a is the hidden dimension of self-attention mechanism and as described earlier r is the number of attention vectors for given text input and then we apply following set of operations to produce the attention matrix for input text T.

$$H_a = tanh(W_{s_1} H^T)$$

Here H_a has dimensions (d_a, n)

$$A = softmax(W_{s_2} H_a)$$

Finally, we compute sentence/document level embeddings

$$M = AH$$

A has dimensions (r, n) and M has dimensions $(r, 2l)$ also, earlier the softmax applied along the second dimension of A normalizes attention weights so they sum up 1 for each attention vector of length n.

Zhouhan Lin et al. [5] also proposed penalization term in place of regularization to counterbalance redundancy in embedding matrix M when attention mechanism results in the same summation weights for all r hops, additionally, We initially started by setting this penalization term to 0.0 however, as self-attention generally works well for finding long term dependencies the neural network started to overfit after few epochs of training on train data. We started with the same hyper-parameters setting of self-attention block as described by Zhouhan Lin et al. [5] while setting $r = 30$ however, we started with no penalization to start with and found the best values for them while monitoring the validation set accuracy which is a hidden dimension of 300 for self-attention, with 8 layers of BiLSTM with a hidden dimension of 32 and also, the output of self-attention mechanism (sentence embeddings M) goes into a fully connected layer with its hidden dimension set to 2000, finally we feed the fully connected layer's results to output layer wrapped with sigmoid activation. The choice of the loss function, learning rate, and optimizer remain unchanged from the baseline, the number of epochs are 20 here. After training the model with hyperparameters suggested in the above text, we observed the model started to overfit on train data after a few epochs and almost achieved 99% train accuracy and loss less than 0.5 average epoch train loss, in order to add the remedy for this we visually inspected the few of the examples from the test set in attention matrix with confidence $>$0.90 and observed for longer sequences the attention mechanism worked as expected, however, as the sequence length decreased the attention mechanism started producing roughly equal summation weights on all r hops which intuitively makes sense in short sequences all tokens would carry more semantic information, however, this result in redundancy in attention matrix A and in embedding matrix M. Below we present some of the examples from the Hindi test set, also since showing all the vectors would make it redundant so we only present 5 vectors for a given sequence even though we had r set to 30 which implies we had 30 vectors for each sequence.

Also, we performed the same analysis as we performed for Bengali data. Following we would also show few similar examples as we showed for Hindi sequences.

In order to counterbalance this redundancy, we started increasing the value of the penalization coefficient of the attention mechanism in order to reduce the redundancy among the attention matrix and found penalization coefficient of 0.6 produced the best validation set accuracy, similarly, the other form of diagnosis we performed was to

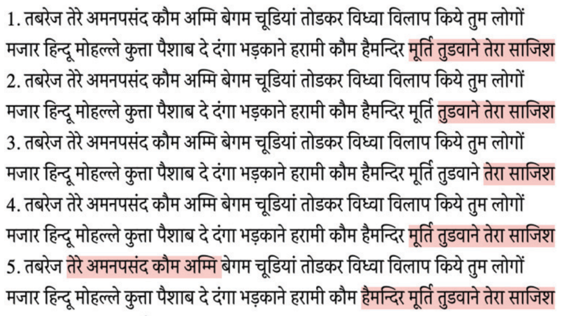
1. तबरेज तेरे अमनपसंद कौम अम्मि बेगम चूडियां तोडकर विध्वा विलाप किये तुम लोगों मजार हिन्दू मोहल्ले कुत्ता पैशाब दे दंगा भड़काने हरामी कौम हैमन्दिर मूर्ति तुडवाने तेरा साजिश
2. तबरेज तेरे अमनपसंद कौम अम्मि बेगम चूडियां तोडकर विध्वा विलाप किये तुम लोगों मजार हिन्दू मोहल्ले कुत्ता पैशाब दे दंगा भड़काने हरामी कौम हैमन्दिर मूर्ति तुडवाने तेरा साजिश
3. तबरेज तेरे अमनपसंद कौम अम्मि बेगम चूडियां तोडकर विध्वा विलाप किये तुम लोगों मजार हिन्दू मोहल्ले कुत्ता पैशाब दे दंगा भड़काने हरामी कौम हैमन्दिर मूर्ति तुडवाने तेरा साजिश
4. तबरेज तेरे अमनपसंद कौम अम्मि बेगम चूडियां तोडकर विध्वा विलाप किये तुम लोगों मजार हिन्दू मोहल्ले कुत्ता पैशाब दे दंगा भड़काने हरामी कौम हैमन्दिर मूर्ति तुडवाने तेरा साजिश
5. तबरेज तेरे अमनपसंद कौम अम्मि बेगम चूडियां तोडकर विध्वा विलाप किये तुम लोगों मजार हिन्दू मोहल्ले कुत्ता पैशाब दे दंगा भड़काने हरामी कौम हैमन्दिर मूर्ति तुडवाने तेरा साजिश

Figure 3. Attention vectors for a relatively longer Hindi sequence.

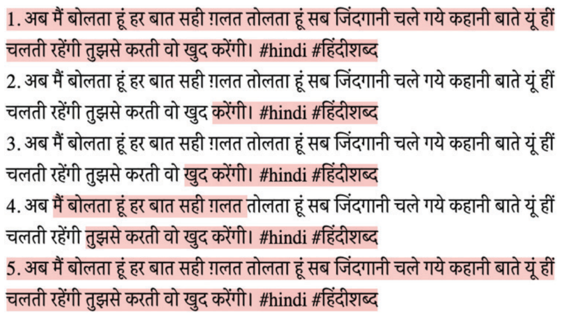
1. अब मैं बोलता हूं हर बात सही ग़लत तोलता हूं सब जिंदगानी चले गये कहानी बाते यूं हीं चलती रहेंगी तुझसे करती वो खुद करेंगी। #hindi #हिंदीशब्द
2. अब मैं बोलता हूं हर बात सही ग़लत तोलता हूं सब जिंदगानी चले गये कहानी बाते यूं हीं चलती रहेंगी तुझसे करती वो खुद करेंगी। #hindi #हिंदीशब्द
3. अब मैं बोलता हूं हर बात सही ग़लत तोलता हूं सब जिंदगानी चले गये कहानी बाते यूं हीं चलती रहेंगी तुझसे करती वो खुद करेंगी। #hindi #हिंदीशब्द
4. अब मैं बोलता हूं हर बात सही ग़लत तोलता हूं सब जिंदगानी चले गये कहानी बाते यूं हीं चलती रहेंगी तुझसे करती वो खुद करेंगी। #hindi #हिंदीशब्द
5. अब मैं बोलता हूं हर बात सही ग़लत तोलता हूं सब जिंदगानी चले गये कहानी बाते यूं हीं चलती रहेंगी तुझसे करती वो खुद करेंगी। #hindi #हिंदीशब्द

Figure 4. Attention vectors for a relatively medium Hindi sequence.

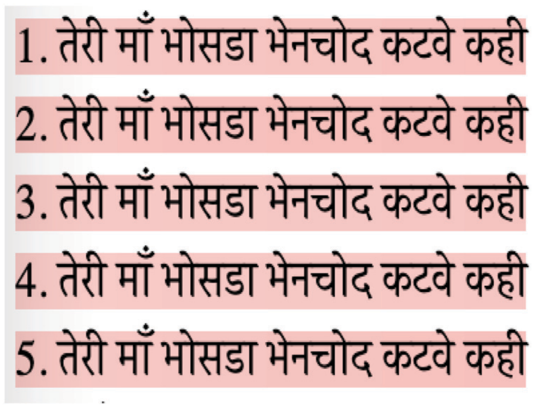
1. तेरी माँ भोसडा भेनचोद कटवे कही
2. तेरी माँ भोसडा भेनचोद कटवे कही
3. तेरी माँ भोसडा भेनचोद कटवे कही
4. तेरी माँ भोसडा भेनचोद कटवे कही
5. तेरी माँ भोसडा भेनचोद कटवे कही

Figure 5. Attention vectors for a short Hindi sequence.

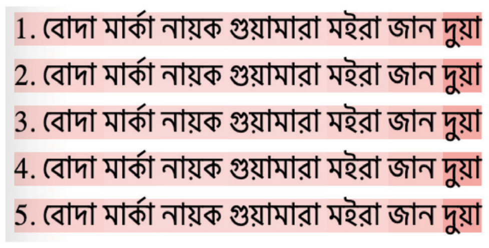
1. বোদা মার্কা নায়ক গুয়ামারা মইরা জান দুয়া
2. বোদা মার্কা নায়ক গুয়ামারা মইরা জান দুয়া
3. বোদা মার্কা নায়ক গুয়ামারা মইরা জান দুয়া
4. বোদা মার্কা নায়ক গুয়ামারা মইরা জান দুয়া
5. বোদা মার্কা নায়ক গুয়ামারা মইরা জান দুয়া

Figure 6. Attention vectors for a short Bengali sequence.

actually reduce the number of attention hops, i.e., varying the hyper-parameter r and observed network with r = 20 had better performance on validation, alongside setting the hidden size of attention mechanism to 150 set as compared

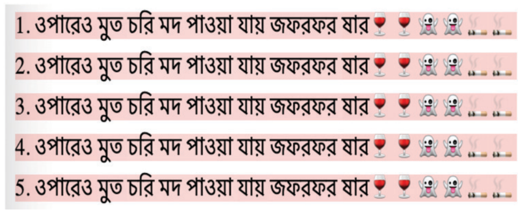
1. ওপারেও মুত চরি মদ পাওয়া যায় জফরফর ষার🍷 🍷 👻 👻 🚬 🚬
2. ওপারেও মুত চরি মদ পাওয়া যায় জফরফর ষার🍷 🍷 👻 👻 🚬 🚬
3. ওপারেও মুত চরি মদ পাওয়া যায় জফরফর ষার🍷 🍷 👻 👻 🚬 🚬
4. ওপারেও মুত চরি মদ পাওয়া যায় জফরফর ষার🍷 🍷 👻 👻 🚬 🚬
5. ওপারেও মুত চরি মদ পাওয়া যায় জফরফর ষার🍷 🍷 👻 👻 🚬 🚬

Figure 7. Attention vectors for a short Bengali sequence.

Table 2. Results of evaluating the binary neural Hindi and Bengali sentiment classifiers on their respective test sets.

Model	Accuracy	Precision	Recall	F-1 Score
LSTM-Hindi	0.74	0.74	0.74	0.74
LSTM-Bengali + Pret	0.77	0.77	0.77	0.77
SA + JDIL (Hindi)	**0.76**	**0.76**	**0.76**	**0.76**
SA + JDIL (Bengali)	**0.78**	**0.78**	**0.78**	**0.78**

to r = 30 and hidden size = 200 as suggested in the original work. Also, in order to avoid any over-fitting during the BiLSTM block, we used dropout in BiLSTM layers with a value of p = 0.5.

RESULTS

LSTM Bengali + Pret in Table 2 refers to the model which shares task-specific pre-trained weights from *LSTM Hindi* classifier. *SA + JDIL* is our method which uses self-attention with joint dual input learning to train a joint neural network. Results in Table 2 empirically show that joint-learning can benefit both the pre-training task performance as well the downstream task due to the joint training procedure. Moreover, since the pre-training model *LSTM Hindi* has to perform training starting with randomly initialized weights, it is not possible in that setting for the pre-training network to benefit from downstream task, however, our proposed approach make this possible which results in meaningful performance gain for the pre-training task on the performance metrics.

CONCLUSION

In our study we investigate whether self-attention can enhance significantly the performance over uni-directional LSTM in the binary classification task setting, moreover, we also investigated when the tasks are the same in our case binary classification in Hindi and Bengali language, whether how do transfer learning and joint dual input learning setting perform. Firstly we found when the lengths of sequences are not that long LSTMs can perform almost as well as using self-attention since there are no very distant dependencies in sequences in most of the cases. Secondly, we observed that transfer learning in case similar or same tasks can be a beneficial way of increasing the performance of the target task which in our case was Bengali binary classification. However, by introducing the joint learning setting where we trained a single network for both tasks the Hindi classification task that was the source task in the transfer learning setting, also got benefited in the joint learning setting as its performance improved. Moreover, such architecture provides an implicit mechanism to avoid overfitting as it roughly doubled the dataset size when we trained a single network.

REFERENCES

[1] Thomas Mandla, Sandip Modha, Gautam Kishore Shahi, Amit Kumar Jaiswal, Durgesh Nandini, Daksh Patel, Prasenjit Majumder, and Johannes Schäfer. Overview of the hasoc track at fire 2020: Hate speech and offensive content identification in indo-european languages, 2021. 1.

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017. 1.

[3] Katikapalli Subramanyam Kalyan, Ajit Rajasekharan, and Sivanesan Sangeetha. Ammus: A survey of transformer-based pretrained models in natural language processing, 2021. 1

[4] Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey, 2020. 1.

[5] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding, 2017. 1, 3, 4.

[6] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. 2.

[7] Jenq-Haur Wang, Ting-Wei Liu, Xiong Luo, and Long Wang. An LSTM approach to short text sentiment classification with word embeddings. In *Proceedings of the 30th Conference on Computational Linguistics and Speech Processing (ROCLING 2018)*, pages 214–223, Hsinchu, Taiwan, October 2018. The Association for Computational Linguistics and Chinese Language Processing (ACLCLP). 2.