BOHR International Journal of Computer Science
2022, Vol. 1, No. 1, pp. 32–37
DOI: 10.54646/bijcs.2022.05
www.bohrpub.com

METHODS

Hindi/Bengali sentiment analysis using transfer learning and joint dual input learning with self-attention

Shahrukh Khan* **and Mahnoor Shahid**

***Correspondence:**
Shahrukh Khan,
shkh00001@stud.uni-saarland.de

Received: 25 February 2022; **Accepted:** 05 April 2022; **Published:** 20 April 2022

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 data set and Bengali hate speech (1) and then train long short-term memory 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 data sets 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 has quickly become state-of-the-art in multiple pervasive natural language processing tasks such as entity recognition classification (3, 4). In our work, we propose a novel self-attention-based architecture for sentiment analysis and classification on **Hindi HASOC data set** (1). We use sub-task A which deals with whether or not a given tweet has hate speech. Moreover, this also serves as a source domain in the subsequent task-specific transfer learning task, in which 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 a joint dual input learning setting. Lin et al. (5) suggested how we can integrate self-attention with BiLSTMs and gave a hidden representation containing different aspects for each sequence, which results in sentence embeddings while 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 show 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. Further, 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 pretraining-based transfer learning approaches.

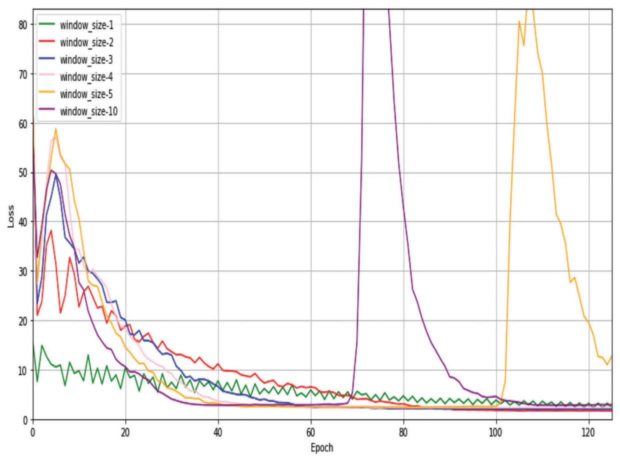


FIGURE 1 | Train loss convergence for different values of window size with fixed embedded size = 300 Hindi data set.

Experimental setting

Word embeddings

Starting with the Hindi data set, we prepared the training data set employing sub-sampling technique by first computing 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. 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 as frequent words like "the," "me," etc. do not necessarily carry much semantic meaning in a given sequence. Hence, 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. Furthermore, for choosing hyperparameters, we performed the following analysis (Figure).

It is apparent from the above visualization that Word2Vec models with smaller context windows converged faster and had better train loss at the end of the training process. However, to retain some context-based information, we selected the window size 2 as it has contextual information and the model had better train loss.

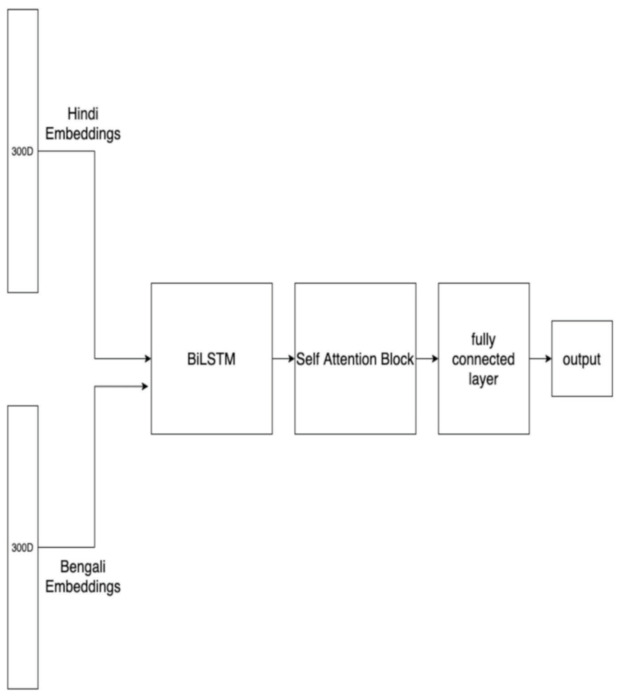
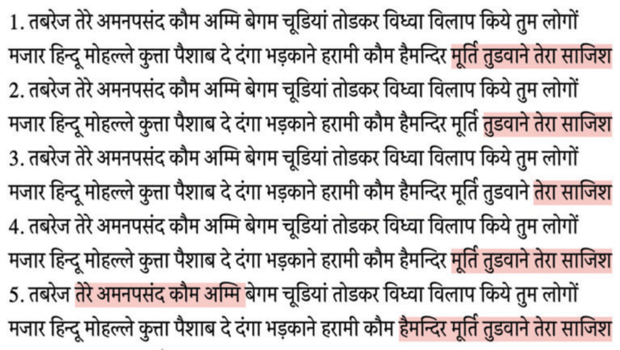
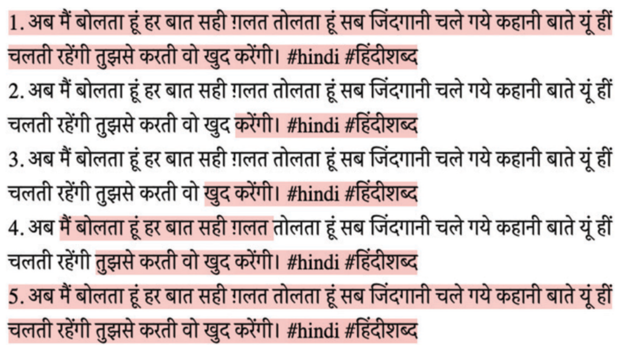


FIGURE 2 | Hindi and Bengali joint dual input learning architecture.



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

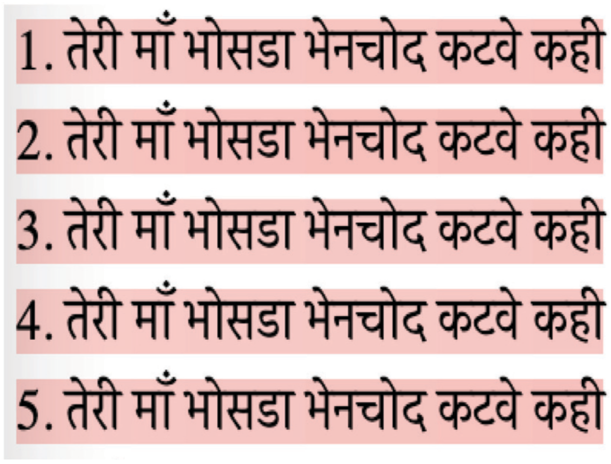
FIGURE 3 | Attention vectors for a relatively longer Hindi sequence.



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

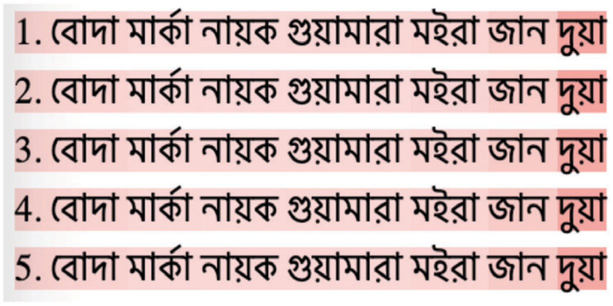
FIGURE 4 | Attention vectors for a relatively medium Hindi sequence.

After testing different values for hyperparameters with different combinations, it was observed that for the better performance of the model, they should be set to



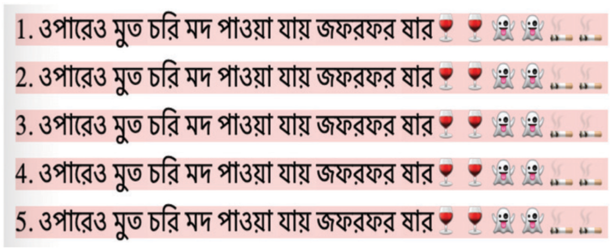
1. तेरी माँ भोसडा भेनचोद कटवे कही
2. तेरी माँ भोसडा भेनचोद कटवे कही
3. तेरी माँ भोसडा भेनचोद कटवे कही
4. तेरी माँ भोसडा भेनचोद कटवे कही
5. तेरी माँ भोसडा भेनचोद कटवे कही

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.



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.

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 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 did not 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 due to 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 Adam (6).

Baseline models

For baseline, we reproduced the work by Wang et al. (7), which primarily focuses on performing sentiment classification on short social media texts by using long short-term memory (LSTM) neural networks using distributed representations of Word2Vec skip-gram approach. We chose these authors concepts because they used Word2Vec Skip-gram based distributed representation of words and also our data sets were sourced from social media. Moreover, the neural network LSTM is an upgraded variant of the recurrent neural network (RNN) model that serves as the remedy, to some extent, of the problems that require learning long-term temporal dependencies; due to vanishing gradients, LSTM uses a gate mechanism and memory cell to control the memory process.

Hindi neural sentiment classifier baseline

We then implemented the architecture for the LSTM classifier using pretrained 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 did not optimize the weight of the embedding layer via gradient optimization since they were pretrained. Moreover, we chose the binary cross entropy loss function as we are dealing with binary classification. In model architecture, we used eight 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 overfitting and then fully connected the 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 for its accuracy and loss on the validation set. The choice of hyperparameters was made after trying different combinations and we chose the best set of hyperparameters 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 hyperparameters that we chose for the Hindi data set, worked fine here as well, so we did not tune the hyperparameters. Also, the model's train loss converged to the similar value we had for the Hindi data set. Subsequently, we then create the same architecture for LSTM-based classifier architecture as explained in section "Hindi Neural Sentiment Classifier Baseline." Our goal was to perform transfer learning and reuse and fine-tune the learned weights of the Hindi classifier. We replaced the Hindi embeddings layer with a Bengali 300-dimensional embedding layer and not optimized its weights during training. Next, we loaded the weights from the Hindi classifier for LSTM layers and fully connected the layers applying sharing-based task-specific transfer learning technique. 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 hyperparameter was made after trying different values and we chose the best hyperparameter while monitoring the accuracy of validation set. After training the classifier using the pretrained 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 provides better results.

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 RNN based architectures is they lack the ability capture the dependencies between words that are too distant from each other. The forget gate of LSTM enables to retain information about the historical words in the sequence; however, it does not 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 pretraining.

Hindi and Bengali self-attention-based joint dual input learning BiLSTM classifier

Instead of training two separate neural networks for Hindi and 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 and Bengali data sets, respectively. We also added the Hindi and Bengali batches' attention loss to the joint loss in order to avoid overfitting, which we present in detail in the subsequent sections. Here we switched between the embedding layers based on the language of the batch data. A block architecture we proposed is explained herein.

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 that both languages have an equal number of training examples. Consequently, it reduces the risk of overfitting.

We propose an extension of the work of (5) where they proposed the method of **a structured self-attentive sentence embedding** on the Hindi data set. The key idea was to propose document-level embeddings by connecting the self-attention mechanism right after a BiLSTM, 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 represents how attention mechanism is putting more relative weight on different tokens in the sequence. He key takeaways on how self-attentive document embeddings are produced are discussed as follows:

We start with an 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.

$$^{T}{}_{hindi} = [e_1, e_2, e_3, \ldots, e_n]$$

$$^{T}{}_{bengali} = [e1, e2, e3, \ldots, e_n]$$

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 the from left to right and from the left to right direction, with each BiLSTM cell/layer producing two vectors of hidden states equivalent to the length of the sequence.

$$[[\overrightarrow{h_1}, \overrightarrow{h_2}, \ldots, \overrightarrow{h_n}], [\overleftarrow{h_1}, \overleftarrow{h_2}, \ldots, \overleftarrow{h_n}]]$$

$$= BiLSTM([e_1, e_2, \ldots, en]; \theta)$$

$\boldsymbol{H}$ is the concatenate form of bidirectional hidden states. If there are l LSTM layers/cells, then the dimension of $\boldsymbol{H}$ is going to be $(n, 2l)$.

TABLE 1 | Different combinations of hyperparameters from Hindi data set.

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

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

For self-attention, Lin et al. (5) proposed having two weight matrices: Ws_1 with dimension $(d_a, 2l)$ and Ws_2. with dimension (r, d_a), where d_a is the hidden dimension of self-attention mechanism and r is the number of attention vectors for given text input. We then apply following set of operations to produce the attention matrix for input text T.

$$H_a = tanh(Ws_1 H^T)$$

Here H_a has dimensions (d_a, n).

$$A = softmax(Ws_2 H_a)$$

Finally, we compute sentence/document-level embeddings

$$M = AH$$

A has dimensions (r, n) and M has dimensions $(r, 2l)$: 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.

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. We first 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 hyperparameters setting of self-attention block as described by Lin et al. (5) while setting $r = 30$; however, we started with no penalization to start with and found the best

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**

values for them while monitoring the validation set accuracy, which is a hidden dimension of 300 for self-attention, with eight layers of BiLSTM with a hidden dimension of 32 and 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 remains unchanged from the baseline, and the number of epochs is 20. After training the model with hyperparameters, 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 the attention mechanism worked as expected for longer sequences; 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 that all tokens would carry more semantic information. This results in redundancy in attention matrix A and in embedding matrix M. Below we present some of the examples from the Hindi test set. Since showing all the vectors would make it redundant, so we present only five vectors for a given sequence even though we had set r to 30; thus, we had 30 vectors for each sequence.

We performed the same analysis for Bengali data. Following are examples for Bengali sequences.

To counterbalance the redundancy among the attention matrix, we started increasing the value of the penalization coefficient of the attention mechanism and found the value 0.6 produced the best validation set accuracy. Next, we aimed to reduce the number of attention hops, i.e., varying the hyperparameter r, and we observed network with $r = 20$ had better performance on validation, alongside setting the hidden size of attention mechanism to 150, as compared to $r = 30$ and hidden size = 200 as suggested in the original work. Also, to avoid any overfitting 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 pretrained 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 pretraining task performance and the downstream task due to the joint training procedure. Moreover, since the pretraining model *LSTM Hindi* has to perform training starting with randomly initialized weights, it is not possible in that setting for the pretraining network to benefit from downstream task; however, our proposed approach makes this possibles which results in meaningful performance gain for the pretraining task on the performance metrics.

Conclusion

In our study, we investigate whether self-attention can enhance significantly the performance over unidirectional LSTM in the binary classification task setting. We also investigated how to perform transfer learning and joint dual input learning setting when the tasks are the same in binary classification in Hindi and Bengali languages. First, we found that if the length of sequence is not long enough, LSTM can be performed using self-attention since there are no very distant dependencies in sequences in most of the cases. Second, we observed that transfer learning in 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, which was the source in the transfer learning setting, benefited with improved performance. Therefore, such architecture provides an implicit mechanism to avoid overfitting as it roughly doubled the data set size when we trained a single network.

References

1. Mandla T, Modha S, Shahi GK, Jaiswal AK, Nandini D, Patel D, et al. Overview of the hasoc track at fire 2020: hate speech and offensive content identification in Indo-European languages. *arXiv* [Preprint]. (2021). Available online at: https://doi.org/10.48550/arXiv.2108.05927
2. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention is all you need. *arXiv* [Preprint]. (2017). Available online at: https://doi.org/10.48550/arXiv.1706.03762
3. Kalyan KS, Rajasekharan A, Sangeetha S. Ammus: a survey of transformer-based pretrained models in natural language processing. *arXiv* [Preprint]. (2021). Available online at: https://doi.org/10.48550/arXiv.2108.05542
4. Tay Y, Dehghani M, Bahri D, Metzler D. Efficient transformers: a survey. *arXiv* [Preprint]. (2020). Available online at: https://doi.org/10.48550/arXiv.2009.06732
5. Lin Z, Feng M, Nogueira dos Santos C, Yu M, Xiang B, Zhou B, et al. A structured self-attentive sentence embedding. *arXiv* [Preprint]. (2017). Available online at: https://doi.org/10.48550/arXiv.1703.03130
6. Kingma DP, Ba J. Adam: a method for stochastic optimization. *arXiv* [Preprint]. (2017). Available online at: https://doi.org/10.48550/arXiv.1412.6980
7. Wang J-H, Liu T, Luo X, Wang L. An LSTM approach to short text sentiment classification with word embeddings. *Proceedings of the 30th conference on computationa linguistics and speech processing (ROCLING 2018)*. Hsinchu: The Association for Computationa Linguistics and Chinese Language Processing (2018). p. 214–223.