A sequential approach to handle machine translation of low resource languages

K Vimal Kumar[1], Yamuna Prasad[2]

[1] Jaypee Institute of Information Technology, Noida, India
[2] Indian Institute of Technology, Jammu, India
vimalkumar.k@gmail.com, yamuna.prasad@iitjammu.ac.in

Abstract. Machine translation is the process of generating target language text from source language text using suitable features of both languages. Sequence to Sequence model consists of an encoder and a decoder, where, an encoder converts the input vector into an intermediate vector form and these intermediate vectors are fed to a decoder to generate an output vector. In case of machine translation, the sequence to sequence model uses input language text in the vector form and generates the output language text in the form of vector. The generated output vector is further mapped to the target language for generating the target language text. The generated output should retain the syntactic and semantics of the target language. These two features contribute to the machine translation system's accuracy. To perform translation from source to target, a Long-Short Term Memory (LSTM) neural network is being used in the sequence to sequence model. LSTM network captures the syntactic and semantics features which can be used for mapping source text with target text. Since, the languages under consideration are having low resources, the use of language specific feature will be helpful to improve the accuracy of the overall neural machine translation system. The Indian languages used in this proposed system are morphologically rich and free-word ordered. The proposed system makes use of these features to enhance its accuracy. The model is found to have an average sentence-level BLEU score as 0.7588 and corpus-level BLEU score is found to be 0.2134.

Keywords: Sequence to sequence model, Long-short term memory (LSTM) network, Low-resource languages, Machine translation, Language specific features.

1 Introduction

Machine translation is a process of converting the source text into target text by considering the features of the languages being used. Machine translation is an application of natural language processing, but the complexity in this system is more as there are two languages involved and their mapping is more tedious due to the difference in language specific features. There are various machine translation systems and each of them uses its own features. The accuracy of such systems has dependency on the size of corpus being used and more on the quality of corpus used.

The issue with Indian languages is its low resource availability. To handle this issue, the existing system makes use of a pivot language in between the source and target language. But, the introduction of one more language (as pivot language) in the translation system will degrade the accuracy of the system due to generation of more noise in the system. These noises are produced due to difference in mapping from source to pivot and mapping from pivot to target. In case of Hindi language, affixes contribute to the tense, aspect and modality (TAM) of its root word. These affixes also contribute to the mapping between the languages, so that, the TAM information is retained in the target text. The same case happens with the Tamil language as well. Due to which there will be one to many mappings between Tamil language and Hindi language. Thus, there is a need for a system that can handle these language specific features without compromising on its accuracy. The existing machine translation systems accuracy can be improved further by considering both these features. The syntactic feature contributes to accurate grammatically correct translation and these features can be extracted using a trigram feature extraction model. Whereas, the semantic feature will be more helpful to perform a context sensitive translation. The semantic features are captured and embedded into a vector by using a continuous bag-of-words model (CBOW). These vectors basically have encoding of syntactic and semantic features of the languages and are further used to train a machine translation system. The machine translation system learns the feature mapping of the vectors which is a complex task. Deep learning networks are being used to handle this complex task, one such network is recurrent neural network (RNN). The recurrent neural network has a binding with the length of input and output vector. In case of a machine translation system, there is no fixed length of input and output vector. The input vector of the source language will have a different size as compared with the output vector of the target language. So, in such a case, sequence to sequence approach will be helpful, which in turn uses a recurrent neural network.

In a sequence to sequence network, the $(n+1)^{th}$ sequence is predicted based on the existing n-sequences. But, in case of a machine translation where there are two languages being used, a sequence to sequence network should predict the target text based on the input source text. To perform this there is a need of two recurrent neural network (RNN) called as encoder and a decoder. The encoder and decoder use long-short term memory neural network (LSTM), which is a type of RNN. The vector of input source text is fed to the encoder which in turn maps it to an intermediate fixed length vector and these vectors are further fed to a decoder which converts it to target text. A trigram continuous bag-of-words (CBOW) model is used to embed the syntactic and semantic features of words in source language onto a vector. This vector is used as input to the encoder. So that, the embedded features contribute more during the translation.

The section-2 of this paper gives an overview of the recent works that has been carried out in this area of research. The proposed sequence to sequence learning for Hindi-Tamil machine translation has been described in section-3. Section-4 details more about the result outcome of this proposed system. Section-5 concludes about the proposed work and section-6 discusses the various improvements that can be carried out in order to improve the performance of the system.

2 Related Work

In paper by Ilya Sutskever et. al. [1], sequence to sequence learning with neural networks was developed, which consists of an encoder and decoder. Encoder converts the input sentence into a fixed-length vector. These vectors are further converted into target sentences by the decoder module. Both the encoder and decoder are jointly trained such that the probability of accuracy in translation is maximized. The issue that occurs during training a neural network is overfitting. A simple way to prevent overfitting in neural network was introduced by Nitish Srivastava et. al. [2]. In this paper, the author introduced dropout regularization. During the training of neural network, the weights of neuron are randomly made zero based on the dropout percentage which prevents the neurons from adapting too much according to the training data. Author has also found that it improves the problem of overfitting in a significant manner as compared with the other regularization methods.

Accuracy of neural machine translation depends more on the encoder which captures the semantic information that is being conveyed in the source text. These encoders can be unidirectional or bidirectional recurrent neural network (RNN). In the paper authored by B. Zhang et. al., [3], the author has proposed a context aware recurrent neural network for the encoders in a neural machine translation. This context aware encoders works in two level hierarchy, where history information is extracted in the bottom level and are aligned with the future context in the upper level.

There are various deep neural networks designed by researchers such as deep belief networks, recurrent neural network, convolutional neural networks and deep stack networks. In a paper [4] by Zhang et. al., it has been mentioned that there are various types of deep neural networks being used in language modeling, word alignment, translation rule selection and reordering which are used as major phases of a machine translation system. Apart from having different deep neural network for each of these tasks, it is also better to have a pure neural machine translation system which facilitates the computation of semantic distance between texts. The issues that will be existing in machine translation system with deep neural network are - computational complexity, error analysis and reasoning.

Since sense-based machine translation is under progress these days, there is multi-sense based neural machine translation being introduced by Z. Yang et. al. [5]. In their proposed system, they generate the word embedding based on different senses of the word, called as sense-specific embedding. The multi-sense embedding module was developed using a recurrent neural network (RNN). Zhen Yang et. al. also proposes that this sense-based embedding is task independent and it can be applied on any natural language processing task.

Ananthakrishnan et. al. [6] has proposed an English to Hindi statistical machine translation with main focus on the key challenge of mapping the Hindi language (a morphologically rich language) with the English language. Reordering of English source sentences in accordance with Hindi language and/ or making use of suffixes of Hindi words are the two strategies that can facilitate reasonable performance. Since

Indian languages and English differ in word order, morphologically rich Indian languages and unavailability of huge parallel corpora for two languages make the above two strategies more challenging to derive the desired results with reasonable performance.

Goyal [7] proposed a machine translation system mainly from Hindi- Punjabi in 2009 at Punjabi University Patiala. It is based on direct word-to-word translation and reported 95% accuracy. In 2010, Vishal Goyal and Prof. G.S. Lehal proposed a machine translator from Hindi to Punjabi using direct translation and later on improving the language learning modules for enhancement of quality of the system. The accuracy of the translation [8] is approximately found to be 95%.

3 Proposed system

The sequence to sequence machine translation model is basically a combination of an encoder which encodes the input sentence into a fixed length intermediate vector and a decoder which decodes the target sentence from the fixed length vector generated in the encoding process. The encoder and decoder are trained parallelly using the vectors of languages under consideration. To train the sequence to sequence model, the vectors are generated using continuous bag-of-words (CBOW) model. Continuous bag-of-words model is a deep learning model which predicts the word at a context by considering the neighboring words in the training corpus. The neighboring words before that word and after the word are considered during training of the network as in case of a trigram model. These vectors which are generated for the languages under consideration, i.e., Hindi and Tamil, are further used for training the sequence to sequence model. The sequence to sequence model makes use of a Long-Short term memory (LSTM) network at the back end. LSTM network is a special variant of recurrent neural network which is capable of learning long term dependencies between sentences. This network keeps a memory of information over a long period of time so that the dependencies between them are captured properly. In machine translation, the meaning of a sentence depends on its preceding sentences in the text corpus. Thus, a LSTM network will be helpful for this sequence to sequence model.

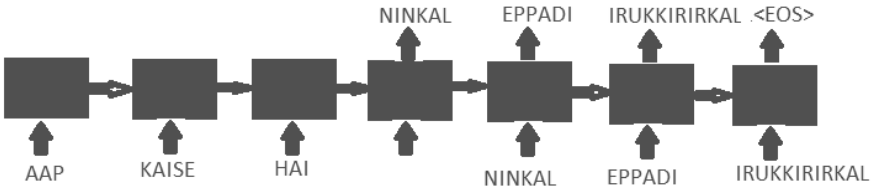

Fig. 1. Architecture of sequence to sequence model-based Hindi to Tamil machine translation

3.1 Word Embedding

Word embedding is basically used to represent the words in a continuous vector space and those words which are semantically similar are placed at the neighboring points in the vector space. Since the proposed system works based on syntactic and semantic features of the languages, there is a need for a method to learn this word embedding from the textual data. So that, the vectors that are generated using the embedding process would have the semantic relations between the words. One such approach is continuous bag-of-words (CBOW) model developed by Mikolov et. al. [9,10], which predicts the word at n-position if its preceding words are known to the model. In the CBOW model, a neural network is being used to learn based on the training data provided to it. The semantic information is represented in the form of vectors. During the training and extraction phase, the CBOW model considers a trigram model as it is found to be more accurate as compared with any n-gram. Thus, the CBOW model considers the two words and predicts the third one during training phase. Based on the training, it generates the vectors, that can be fed to the sequence to sequence based machine translation.

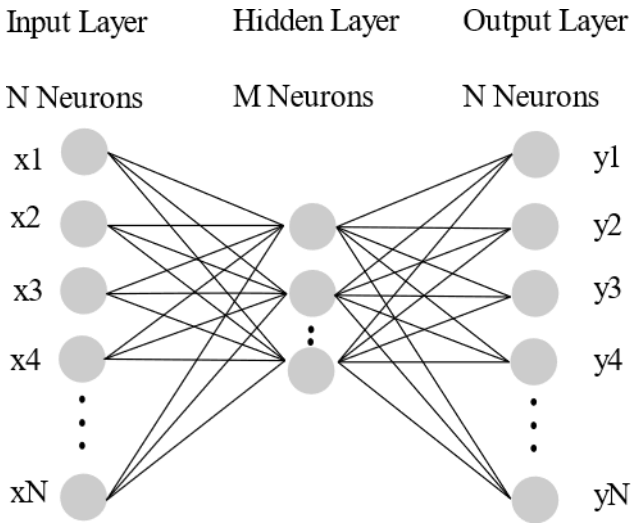

Fig. 2. CBOW model for word embedding

As shown in figure-2, the input layer of continuous bag-of-words model will be fed with the word's one hot vector based on its various context. The output layer will generate word vectors which is embedded with its contextual information. The distance between the vectors are calculated using cross entropy and it makes use of gradient descent to update the word vectors accordingly.

3.2 Long-Short Term Memory Network (LSTM)

A recurrent neural network (RNN) keeps a memory of the previous computations made in the network. RNN considers that the current state has dependency with the

previous state of the network, which is like a sequence mapping kind. In natural language processing, the sequential dependency has a vital role in various task. Thus, RNN is more appropriate for various NLP related tasks. But, in case of machine translation, there is need for contextual information and the contextual information in n^{th} sentence has some dependency with m^{th} sentence that occurred before the n^{th} sentence. So, the memory which is being used in RNN must store the information for longer duration such that the current state has dependency with m^{th} state before it. One such system was developed by Hochreiter et. al. [11] and is called as long short term memory network (LSTM).

3.3 Sequence to sequence model

In a sequence to sequence model, there are two LSTM networks called as encoder and decoder, as shown in figure 3. The encoder encodes the input vector into a single intermediate vector. The decoder reads this intermediate vector and generates the target vector which is further mapped to its corresponding words in the target language.

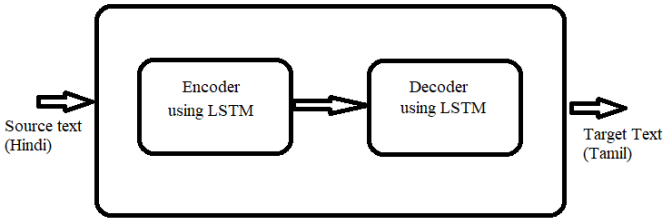

Fig. 3. Sequence to sequence model

3.4 Encoder and Decoder

The encoder encodes the input vector to an intermediate vector using a long short term memory (LSTM) network. Since the LSTM network needs to keep a memory of the vectors, there is LSTMCell being used in it to store these values. To protect and control these values there are four different gates being used in it – input gate, out gate, cell gate and forget gate. Based on these gate values, the LSTM network discards a value during training or keeps it during training. The input gate and cell gate are the one which decides what value is to be stored in the LSTMCell. Forget gate is used to predict whether the value must be retained for further use or not. The out gate is the used to decide upon the value that needs to be updated on the LSTMCell. Using these gate values, the LSTM network computes the output of the hidden state h_t at time t based on the expression,

$$h_t = o_t \tanh c_t$$

Where, o_t - out gate and c_t - value at LSTMCell

As per the expression above, the output of hidden state depends on the current value stored in the LSTMCell and the out gate. The current value of the LSTMCell c_t is calculated using the previous cell value and the gate value based on the equation [12],

$$c_t = f_t * c_{t-1} + i_t * g_t$$

Where, f_t – forget gate and i_t- input

During the calculation of the current value of LSTMCell, the values of forget gate, input gate, cell gate and previous cell value are multiplied element-by-element. The out gate value is calculated based on the weights assigned between input and output along with the weights that are assigned between hidden and output layers. Apart from these two parameters, the out gate also uses the value of hidden state in the previous layer (x_t) and hidden layer value at time t-1, h_{t-1}. Sigmoid function is being used to keep the values in the range of 0 to 1. It is mathematically expressed as,

$$o_t = sigmoid\,(w_i^o * x_t + b_i^o + w_h^o * h_{t-1} + b_h^o)$$

Where,
w_i^o- Weight vector between input layer and out gate
w_h^o- Weight vector between hidden layer and out gate
b_h^o- Bias between hidden layer and out gate
b_i^o- Bias between input layer and out gate

The values of LSTMCell and the out gate value are the primary requirement of the LSTM network to predict the vector at the hidden state h_t. But these two values, LSTMCell value and out gate value, are dependent on other parameters such as, input gate, forget gate and cell gate, which are further calculated using the below mentioned equations,

$$g_t = \tanh\,(w_i^g * x_t + b_i^g + w_h^g * h_{t-1} + b_h^g)$$
$$f_t = sigmoid\,(w_i^f * x_t + b_i^f + w_h^f * h_{t-1} + b_h^f)$$
$$i_t = sigmoid\,(w_i^i * x_t + b_i^i + w_h^i * h_{t-1} + b_h^i)$$

The decoder also uses a LSTM network for its mapping between the intermediate vector and the output vector. The mathematical representation is same as in encoder. But, here the input to the decoder will be the output from the encoder module. Based on the intermediate vector, it generates the output vector in the target language. The gate values and LSTMCell are calculated using the intermediate vector, which is the output of the encoder.

3.5 Attention mechanism

In case of the sequence to sequence network described above, the contextual information of the word at n^{th} position is passed on to the word at $(n+1)^{th}$ position. The decoder gets input as an encoded vector from the last sequence of an encoder. It is considered that the contextual information of all the preceding sequences is stored

in the last sequence of encoder which will help the decoder to generate the sequence of words in target languages. It actually stores the contextual information, but, in case of Indian languages such as Hindi and Tamil, the contextual information of target language has some dependency with any of the sequences in the source language text. Thus, the vector generated from last word's contextual information won't be sufficient for an accurate translation. Also, there is a need of special alignment phase during this process such that the target text is grammatically correct with respect to the target language grammar. Both these issues have been taken care by the introduction of attention mechanism in the sequence to sequence model that too in between the encoder and decoder model. In the attention mechanism [13], there is a method to identify which encoded sequence is important for the decoder during its processing. In the attention mechanism, multiplication of the encoded output with the weights is performed to generate a weighted vector. The weights are identified using the training of a feed forward network which takes the encoded output as input to it and the decoder output as its output.

3.6 Dataset

The dataset for Indian languages has been provided by the Technology development for Indian languages programme (TDIL), an initiative by the Department of IT, Ministry of Communication and IT, Government of India. The dataset is basically a parallel corpus between these two languages – Hindi and Tamil. The parallel corpus has 25000 sentence pairs and these sentence pairs are part-of-speech tagged based on the tag-set predefined for the languages. The corpus used is specifically for the health domain. This sequence to sequence model has been developed using randomly generated set of training and testing sentences from the provided corpus.

3.7 Training

The proposed model needs to learn features of languages fed to the system. In order to make the model learn the features, there is need for huge amount of corpus. But, the languages being considered in this proposed system has low resources. To handle this low resource availability issue, there is significant need for a mechanism that can assist in translation process. By exploiting the language specific features, it has been identified that both Hindi and Tamil language are free-word ordered languages. Free-word ordered language is one which doesn't have a rigid grammatical structure such as English. In case of Hindi and Tamil, the words in a sentence can be reordered to form a valid variant of the given sentence. By reordering the words, the meaning of the sentence is not affected in free word-ordered languages.

The parallel corpus has been increased by the use of various combinations of sentences in the limited corpus available. For every sentence in the source language with N words, a combination of N! sentences will be formed. Since Hindi (source language) is partially free word-ordered, there is need to check the validity of these N! variants of the given sentence. To check the validity of variants, the Hindi shallow

parser [14] is being used. The validated sentences are merged with the parallel corpus if it is found to be syntactically correct. Similarly, the Tamil language texts are also used to generate the variants of each text. Since Tamil is fully free-word ordered language, there is no need to perform the validation of the text.

The sequence to sequence model is trained using various combinations of the training set. This model is trained using the input vector of the source language (Hindi) and the output vector of the target language (Tamil). The input vector is fed to the encoder as input, which in turn generates an intermediate vector. This intermediate vector is used by the decoder to generate an output vector, which is compared with the actual output vector based on the training corpus. Until, the error is as low as possible, it keeps on training the system. But there was a problem of overfitting in the network that may led to poor accuracy in the translation. If there is overfitting in a network, it may lead to more fluctuations for every training data provided to it. In order to reduce the problem of overfitting, the dropout, a regularization mechanism has been introduced in the model. Dropout mechanism drops certain percentage of neuron in a random fashion to train the network based on the rest of the active neurons. The model has been tried with various dropout percentage to train the system for improving the overall accuracy. It has been found that the optimal percentage value of dropout should be in the range of 20% to 60%. The system's performance degrades if the dropout percentage is kept at 20% for both encoder and decoder modules. This degradation in the performance is due to underfitting in the network thus producing an erroneous result. In order to improve the training accuracy, the dropout for encoder is kept at 20% and for the decoder it is kept at 60%. It means the encoder, randomly drops the 20% of neurons during training at each epoch and the decoder randomly drops 60% of the neurons.

The sequence to sequence model is tuned using the following parameters during training,

Number of epochs = 22
Learning rate = 0.01
Hidden layer size = 2
Dropout = 0.2 (in encoder) and 0.6 (in decoder)

It has been noted that the model's performance degrades after 22 epochs due to the problem of overfitting. In case, the number of epochs is kept lower, then the model is not trained sufficiently that leads to underfitting. After 22 epochs, the model is trained properly with the maximum log likelihood error to be around 0.0134. The learning rate is found to be ideally at 0.01 for this model. The model's accuracy fluctuates more when the learning rate is kept at 0.1 and this is due to more error caused by the change in weights. If it is kept lesser, i.e., 0.001, the model takes more time to train the model. If there is increase in hidden layer size, there is no improvement on the performance of the model. Increasing number of hidden layers doesn't work well for this model. It has been found that with the hidden layer size as 2, the model outperforms than the one with more than two hidden layers.

4 Results

The neural machine translation system was developed using sequence to sequence model and the output of this system was evaluated using the Bilingual evaluation understudy (BLEU) score [15]. The BLEU score for the various training and testing corpus is as shown in table-1. The BLEU score has been calculated using a different set of testing and training pairs. It is found that the BLEU score increases with respect to the increase in the percentage of training corpus. It is found that the accuracy of the model is more when the training corpus is kept ideally at 80% of the corpus.

Table 1. Result analysis of neural machine translation

S. No.	Training/Testing Corpus Size (in %)	BLEU Score
1	60/40	0.7037
2	70/30	0.7234
3	80/20	0.7588
4	90/10	0.6628

The result table-2 shows the sentence-level BLEU score on 5-different runs with different random set of training and testing corpus but by keeping the dataset in 80-20 ratio, i.e., 80% of data as training set and the rest as test set.

Table 2. Analysis of neural machine translation at different runs

Number of runs	BLEU score
1	0.7478
2	0.7176
3	0.6784
4	0.7118
5	0.7588

5 Conclusion

The word embedding is performed using a continuous bag-of-words model and it is found to capture the semantics in the words. This in turn helped in improving the accuracy of the sequence to sequence model. Since Hindi and Tamil language are morphologically rich, there is need for semantic features which is used in the sequence to sequence model. Since both these languages has low resources, the free word order feature of both these languages are used to improve the accuracy of sequence to sequence model. The results are found to be far better than any state-of-art method for these two languages. The average sentence level BLEU score is found to be 0.7588 and can be improved further using a properly aligned parallel corpus. The overall corpus level BLEU score is found to be 0.2134.

6 Future Work

To improve overall accuracy of the system, the training corpus size is increased. But, by increasing the training corpus to 90% of given corpus it is found that the model degrades the performance of system and thus, it declines the accuracy of translation. This is due to the slight deviation in mapping of the source text with the target text due to equally probable chances for multiple target sentences. But it can be improved by training the model with increase in the size of parallel corpus to reduce the ambiguity in the translation. The problem with decrease in BLEU score can be handled by improving the word embedding approach which can capture the multi sense information in a more appropriate manner.

References

1. Sutskever, I., Vinyals, O. & Le, Q. V. Sequence to Sequence Learning with Neural Networks. in *Proceedings of Neural Information Processing Systems (NIPS 2014)* 1–9 (2014). doi:10.1007/s10107-014-0839-0
2. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.* **15,** 1929–1958 (2014).
3. Zhang, B., Xiong, D., Su, J. & Duan, H. A Context-Aware Recurrent Encoder for Neural Machine Translation. *IEEE/ACM Trans. Audio, Speech Lang. Process.* **25,** 2424–2432 (2017).
4. Jiajun Zhang, C. Z. Deep Neural Networks in Machine Translation : An Overview. *IEEE Intell. Syst.* **30,** 16–25 (2015).
5. Yang, Z., Chen, W., Wang, F. & Xu, B. Multi-sense based neural machine translation. in *2017 International Joint Conference on Neural Networks (IJCNN)* 3491–3497 (2017). doi:10.1109/IJCNN.2017.7966295
6. Ramanathan, A., Bhattacharyya, P., Hegde, J., Shah, R. & Sasikumar, M. Simple Syntactic and Morphological Processing Can Help English-Hindi Statistical Machine Translation. in *Proceedings of the 3rd International Joint Conference on Natural Language Processing (IJCNLP 2008)* 513–520 (2008).

7. Goyal, V. & Lehal, G. S. Web Based Hindi to Punjabi Machine Translation System. *J. Emerg. Technol. Web Intell.* **2,** 148–151 (2010).
8. Goyal, V. & Lehal, G. S. Hindi to Punjabi machine translation system. in *Information Systems for Indian Languages* 236–241 (Springer Berlin Heidelberg, 2011). doi:10.1007/978-3-642-19403-0_40
9. Mikolov, T., Chen, K., Corrado, G. & Dean, J. Efficient Estimation of Word Representations in Vector Space. in *International conference on Learning Representations* 1–12 (2013). doi:10.1162/153244303322533223
10. Mikolov, T., Chen, K., Corrado, G. & Dean, J. Distributed-Representations-of-Words-and-Phrases-and-Their-Compositionality. *Comput. Res. Repos.* **abs\1310.4,** 1–9 (2013).
11. Hochreiter, S. & Urgen Schmidhuber, J. Long Short-Term Memory. *J. Neural Comput.* **9,** 1735–1780 (1997).
12. Feng, C., Li, T. & Chana, D. Multi-level Anomaly Detection in Industrial Control Systems via Package Signatures and LSTM Networks. in *Proceedings of 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN '17)* 261–272 (2017). doi:10.1109/DSN.2017.34
13. Luong, M.-T., Pham, H. & Manning, C. D. Effective Approaches to Attention-based Neural Machine Translation. in *Proceedings of Empirical Methods in Natural Language Processing (EMNLP '15)* (2015). doi:10.18653/v1/D15-1166
14. Ambati, B. R., Husain, S., Jain, S., Sharma, D. M. & Sangal, R. Two methods to incorporate local morphosyntactic features in Hindi dependency parsing. in *Proceedings of NAACL/HLT workshop on Statistical Parsing of Morphologically-Rich Languages (SPMRL 2010)* 22–30 (2010).
15. Papineni, K., Roukos, S., Ward, T. & Zhu, W. BLEU: a method for automatic evaluation of machine translation. in *Proceedings of the 40Th Annual Meeting on Association for Computational Linguistics* 311–318 (2002). doi:10.3115/1073083.1073135