Poetry to Prose Conversion in Sanskrit as a Linearisation Task: A case for Low-Resource Languages

Amrith Krishna[1], Vishnu Dutt Sharma[2], Bishal Santra[1], Aishik Chakraborty[3]*, Pavankumar Satuluri[4] and Pawan Goyal[1]
[1]Dept. of Computer Science and Engineering, IIT Kharagpur
[2]American Express India Pvt Ltd [3]School of Computer Science, McGill University
[4]Chinmaya Vishwavidyapeeth
{amrith,bishal.santra}@iitkgp.ac.in, pawang@cse.iitkgp.ac.in
chakraborty.aishik@gmail.com

Abstract

The word ordering in a Sanskrit verse is often not aligned with its corresponding prose order. Conversion of the verse to its corresponding prose helps in better comprehension of the construction. Owing to the resource constraints, we formulate this task as a word ordering (linearisation) task. In doing so, we completely ignore the word arrangement at the verse side. *kāvya guru*, the approach we propose, essentially consists of a pipeline of two pretraining steps followed by a seq2seq model. The first pretraining step learns task specific token embeddings from pretrained embeddings. In the next step, we generate multiple hypotheses for possible word arrangements of the input (Wang et al., 2018). We then use them as inputs to a neural seq2seq model for the final prediction. We empirically show that the hypotheses generated by our pretraining step result in predictions that consistently outperform predictions based on the original order in the verse. Overall, *kāvya guru* outperforms current state of the art models in linearisation for the poetry to prose conversion task in Sanskrit.

1 Introduction

Prosody plays a key role in the word arrangement in Sanskrit Poetry. The word arrangement in a verse should result in a sequence of syllables which adhere to one of the prescribed meters in Sanskrit Prosody (Scharf et al., 2015). As a result, the configurational information of the words in a verse is not aligned with its verbal cognition (Bhatta, 1990; Dennis, 2005). Obtaining the proper word ordering, called as the prose ordering, from a verse is often considered a task which requires linguistic expertise (Shukla et al., 2016; Kulkarni et al., 2015).

In this work, we use neural sequence generation models for automatic conversion of poetry to prose. Lack of sufficient poetry-prose parallel data is an impediment in framing the problem as a seq2seq task (Gu et al., 2018).[1] Hence, we formulate our task as that of a word linearisation task (He et al., 2009). In linearisation, we arrange a bag of words into a grammatical and fluent sentence (Liu et al., 2015). This eliminates the need for parallel data, as the poetry order is not anymore relevant at the input. A neural-LM based model from Schmaltz et al. (2016) and a seq2seq model form Wiseman and Rush (2016) are the current state of the art (SOTA) models in the linearisation task.

We first show that a seq2seq model with gated CNNs (Gehring et al., 2017), using a sequence level loss (Edunov et al., 2018) can outperform both the SOTA models for the Sanskrit poetry linearisation task. But using a seq2seq model brings non-determinism to the model as the final prediction of the system is dependent on the order at which the words are input to the encoder (Vinyals et al., 2016). We resolve this, by using a pretraining approach (Wang et al., 2018) to obtain an initial ordering of the words, to be fed to the final model. This approach consistently performs better than using the original poetry order as input. Further, we find that generating multiple hypotheses[2] using this component (Wang et al., 2018), to be fed to the final seq2seq component, results in improving the results by about 8 BLEU points. Additionally, we use a pretraining approach to learn task specific word embeddings by combining multiple word embeddings (Kiela et al., 2018). We call our final configuration as *kāvya guru*. ‘*kāvya guru*’ is a compound word in Sanskrit, which roughly translates to ‘an expert in prosody’.

*Work done while the author was at IIT Kharagpur

[1] Refer to the supplementary material for details on our preliminary experiments in this direction.

[2] Empirically shown to be 10

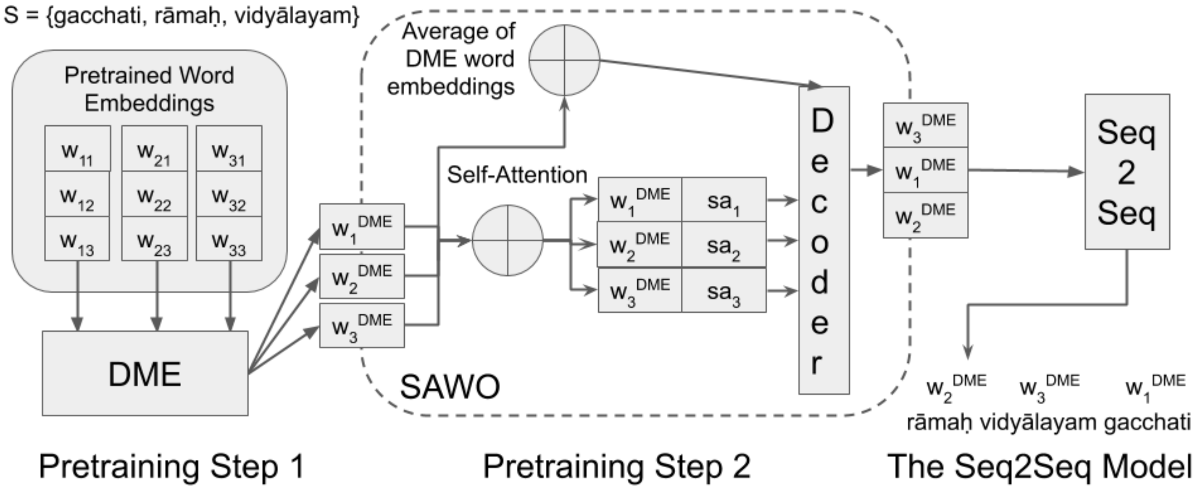

Figure 1: Configuration for *kāvya guru*, demonstrated for a 3 word sentence with a prose order '*rāmaḥ vidyālayam gacchati*'. English translation: "Rāma goes to School". We show generation of only one hypothesis from SAWO.

2 Poetry to Prose as Linearisation

Given a verse sequence $x_1, x_2......x_n$, our task is to rearrange the words in the verse to obtain its prose order. As shown in Figure 2, *kāvya guru* takes the Bag of Words (BoW) S as the input to the system. We use two pretraining steps prior to the seq2seq component in our approach. The first step, 'DME', combines multiple pretrained word embeddings, say $\{w_{11}, w_{12}, w_{13}\}$ for a token $x_1 \in S$, into a single meta-embedding, w_1^{DME}. The second component, 'SAWO', is a linearisation model in itself, which we use to generate multiple hypotheses, i.e., different permutations of the tokens, to be used as input to the final 'seq2seq' component.

Pretraining Step 1 – Dynamic Meta Embeddings (DME): Given a token $x_i \in S$, we obtain r different pre-trained word embeddings, represented as $\{w_{i1}, w_{i2}....w_{ir}\}$. Following Kiela et al. (2018), we learn a single task specific embedding, $\mathbf{w_i^{DME}}$ using weighted sum of all the r embeddings. The scalar weights for combining the embeddings are learnt using self-attention, with a training objective to minimise the negative log likelihood of the sentences, given in the prose order.

Pretraining Step 2 – Self-Attention Based Word-Ordering (SAWO): SAWO allows us to generate multiple permutations of words as hypotheses, which can be used as input to a seq2seq model. Here, we use a word ordering model itself as a pretraining step, proposed in Wang et al. (2018). From step 1, we obtain the DME embeddings, $\{w_1^{DME}, w_2^{DME}, ...w_n^{DME}\}$, one each for each token in S. For each token in S, we also learn additional embeddings, $\{sa_1, sa_2, ...sa_n\}$, using the self-attention mechanism. These additional vectors are obtained using the weighted sum of all the DME embeddings in the input BoW S, where the weights are learned using the self-attention mechanism (Wang et al., 2018; Vaswani et al., 2017). As shown in Figure 2, the DME vector $\mathbf{w_i^{DME}}$ and the vector $\mathbf{sa_i}$ are then concatenated to form a representation for the token X_i. The concatenated vectors so obtained for all the tokens in S, form the input to the decoder.

We use an LSTM based decoder, initialised with the average of DME embeddings of all the tokens ($\{w_1^{DME}, w_2^{DME}, ...w_n^{DME}\}$) at the input. A special token is used as the input in the first time-step, and based on the predictions from the decoder, the concatenated vectors are input in the subsequent time-steps. The decoder is constrained to predict from the list of words in BoW, which are not yet predicted at a given instance. We use a beam-search based decoding strategy (Schmaltz et al., 2016) to obtain top-k hypotheses for the system.

For both the pretraining steps, the training objective is to minimise the negative log likelihood of the ground truth (prose order sentences), and both the components are trained jointly. The multiple hypotheses so generated are used as independent inputs to the seq2seq model, with the prose order as their corresponding ground truth for training. In the figure 2, we show only one hypothesis from SAWO. This helps us to obtain a k-fold increase in the amount of available training data.

The seq2seq model: We use the seq2seq model comprising of gated CNNs (Gehring et al., 2017) for the task. Our training objective is a weighted combination of the expected risk minimisation ($RISK$) and the token level negative log likelihood with label smoothing ($TokLS$) (Edunov et al., 2018). Here, we use a uniform prior distribution over the vocabulary for label smoothing. $RISK$ minimises the expected value of a given cost function, BLEU in our case, over the space of candidate sequences.

$$\mathcal{L}_{Risk} = \sum_{\mathbf{u}\in\mathcal{U}} cost(\hat{\mathbf{y}}, \mathbf{u}) \frac{p(\mathbf{u}|\mathbf{x}')}{\sum_{\mathbf{u}'\in\mathcal{U}} p(\mathbf{u}'|\mathbf{x}')} \quad (1)$$

Here $\mathcal{U}$ is the candidate set, with $|\mathcal{U}|= 16$ and the sequences in $\mathcal{U}$ are obtained using Beam Search. The size for the beam search was determined empirically.[3] $\hat{\mathbf{y}}$ is the reference target sequence, i.e., the *prose*. $\mathbf{x}'$ is the input sequence to the model, which is obtained from SAWO. In $\mathcal{L}_{Risk}$, $cost(\hat{\mathbf{y}}, \mathbf{u}) = 1 - BLEU(\hat{\mathbf{y}}, \mathbf{u})$, where $0 \leq BLEU(\hat{\mathbf{y}}, \mathbf{u}) \leq 1$. Similar to Wiseman and Rush (2016), we constrain the prediction of tokens to those available at the input during testing.

Majority Vote Policy: For an input verse, SAWO generates multiple hypotheses and seq2seq then predicts a sequence corresponding to each of these, of the same size as the input. To get a single final output, we use a 'Majority Vote' policy. For each position, starting from left, we find the token which was predicted the most number of times at that position among all the seq2seq outputs, and choose it as the token in the final output.

3 Experiments

Dataset: We obtain 17,017 parallel poetry-prose data from the epic "*Rāmāyaṇa*".[4] Given that about 90 % of the vocabulary appears less than 5 times in the corpus, we use BPE to learn a new vocabulary (Sennrich et al., 2016). We add about 95,000 prose-order sentences from Wikipedia into our training data, as the poetry order input is irrelevant for linearisation.[5]

Data Preparation: With a vocabulary of 12,000, we learn embeddings for the BPE entries using Word2vec (Mikolov et al., 2013), FastText (Bojanowski et al., 2017), and character embeddings from Hellwig and Nehrdich (2018). The embeddings were trained on 0.8 million sentences (6.5 million tokens) collected from multiple corpora including DCS (Hellwig, 2011), Wikipedia and Vedabase[6]. Finally, we combine the word embeddings using DME (Kiela et al., 2018).

From the set of 17,017 parallel poetry-prose corpus, we use 13,000 sentence pairs for training, 1,000 for validation and the remaining 3,017 sentence pairs for testing. The sentences in test data are not used in any part of training or for learning the embeddings.

Evaluation Metrics: Linearisation tasks are generally reported using BLEU (Papineni et al., 2002) score (Hasler et al., 2017; Belz et al., 2011). Additionally, we report Kendall's Tau (τ) and perfect match scores for the models. Perfect match is the fraction of sentences where the prediction matches exactly with the ground truth. Kendall's Tau (τ) is calculated based on the number of inversions needed to transform a predicted sequence to the ordering in the reference sequence. τ is used as a metric in sentence ordering tasks (Lapata, 2006), and is defined as $\frac{1}{m}\sum_{i=1}^{m} 1 - 2 \times inversions\ count/\binom{n}{2}$ (Logeswaran et al., 2018; Lapata, 2003). In all these three metrics, a higher score always corresponds to a better performance of the system.

3.1 Baselines

LSTM Based Linearisation Model (LinLSTM): LinLSTM is an LSTM based neural language model (LM) proposed by Schmaltz et al. (2016). Sequences in sentence/prose order are fed to the system for learning the LM. Beam search, constrained to predict only from the bag of words given as input, is used for decoding. The authors obtained SOTA results in their experiments on the Penn Treebank, even outperforming different syntax based linearisation models (Zhang and Clark, 2015; Zhang, 2013). The best result for the model was obtained using a beam size of 512, and we use the same setting for our experiments.

[3] We experimented with beam sizes from 1 to 32, in powers of 2. Since the increase in beam size from 16 to 32 did not result in significant improvements in system performance, we set the beam size as 16.

[4] Filtered from 18,250 verses. The remaining were ignored due to corrupted word constructions.

[5] For heuristics used for identifying prose order sentences, refer supplementary material

[6] https://www.vedabase.com/en/sb

System	Augmentation	τ	BLEU	PM(%)
LinLSTM	Ramayaṇa dataset	61.47	35.51	8.22
	+ Wikipedia Prose	58.86	31.39	7.14
BSO	Ramayaṇa dataset	58.62	29.16	7.61
	+ Wikipedia Prose	65.38	41.22	12.97
	+ DME	68.45	44.29	19.69
	+ SAWO	72.89	52.37	24.56
kāvya guru	Ramayaṇa dataset	59.27	31.55	8.62
	+ Wikipedia Prose	66.82	42.91	13.52
	+ DME	70.8	48.33	20.21
	+ SAWO	74.32	54.49	25.72
	+ Self-Attention	**75.58**	**55.26**	**26.08**

(a) Results for all the three competing models. The '+' sign indicates that the augmentation is added to the configuration in the row above it.

k	τ	BLEU	PM
1	71.14	48.26	20.15
5	74.15	53.74	25.02
10	75.58	55.26	26.08

(b) Results for *kāvya guru* when trained (and at test-time) using different values of k at the SAWO pretraining step.

Encoding	τ	BLEU	PM
IAST	73.64	53.46	23.73
SLP1	73.79	53.91	24.16
Syllable	75.58	55.26	26.08

(c) Results for *kāvya guru*, when using different sequence encoding schemes.

Table 1: Experimental results for different configurations and different settings, performed on the test data. Table b and Table c use the configuration in the last row of Table a, which is the best performing configuration of *kāvya guru*.

Seq2Seq with Beam Search Optimisation (BSO): The seq2seq model uses a max-margin approach with a search based loss, designed to penalise the errors made during beam search (Wiseman and Rush, 2016). Here scores for different possible sequences are predicted and then they are ranked using beam search. The loss penalises the function when the gold sequence falls off the beam during training. For our experiments, we use a beam size of 15 for testing and 14 for training, the setting with best reported scores in Wiseman and Rush (2016).

3.2 Results

Table 1a provides the results for all the three systems under different settings. *kāvya guru* reports the best results with a BLEU score of 55.26, outperforming the baselines. We apply both the pretraining components and the 'Majority Vote' policy (§2) to both the seq2seq models, i.e. 'BSO' and the proposed model '*kāvya guru*'.

From Table 1a, it is evident that infusing prose-only training data from Wikipedia, and applying both the pretraining steps leads to significant[7] and consistent improvements for both the seq2seq models. LinLSTM shows a decrease in its performance when the dataset is augmented with sentences from Wikipedia. We obtain the best results for *kāvya guru* when self-attention was added to the seq2seq component of the model (Edunov et al., 2018; Paulus et al., 2018) (final row in Table 1a). Table 1c shows that the text-encoding/transliteration scheme in which a sequence is represented affects the results. *kāvya guru* performs the best when it uses syllable level encoding of input, as compared to character level transliteration schemes such as IAST[8] or SLP1[9].

Effect of increase in training set size due to SAWO: Using SAWO, we can generate multiple word order hypotheses as the input to the seq2seq model. Results from Table 1b show that generating multiple hypotheses leads to improvements in the system performance.[7] It might be puzzling that *kāvya guru* contains two components, i.e. SAWO and seq2seq, where both of them perform essentially the same task of word ordering. This might create an impression of redundancy in *kāvya guru*. But, a configuration that uses only the DME and SAWO (without the seq2seq), results in a BLEU score of 33.8 as against 48.26 for *kāvya guru* (Table 1b, $k = 1$). Now, this brings the validity of SAWO component into question. To check this, instead of generating hypotheses using SAWO, we used 100 random permutations[10] for a given sentence as input to the seq2seq component. The

[7] For all the reported results, we use approximate randomisation approach for significance tests. All the reported values have a p-value < 0.02

[8] https://en.wikipedia.org/wiki/International_Alphabet_of_Sanskrit_Transliteration

[9] https://en.wikipedia.org/wiki/SLP1

[10] Empirically decided from 1 to 100 random permutations with a step size of 10

first 3 rows of BSO and *kāvya guru* in Table 1a show the results for non-SAWO configurations. These configurations do not outperform SAWO based configurations, in spite of using as many as 10 times the candidates than those used in SAWO based configuration. For SAWO (non-SAWO), we find that the system performances tend to saturate with number of hypotheses greater than 10 (100).

Effect of using word order in the verse at inference: During inference, the test-set sentences are passed as input in the verse order to each of the *kāvya guru* configurations in Table 1a. *kāvya guru*+DME configuration achieves the best result for this. But here also, the system performance drops to $\tau = 68.92$ and $BLEU = 45.63$, from 70.8 and 48.33, respectively. To discount the effect of majority vote policy used in SAWO, we consider predictions based on individual SAWO hypotheses. However, even the lowest τ score (70.61), obtained while using the 10th ranked hypothesis from SAWO, outperforms the predictions based on the verse order.[7]

4 Conclusion

In this work, we attempt to address the poetry to prose conversion problem by formalising it as an LM based word linearisation task. We find that *kāvya guru* outperforms the state of the art models in word linearisation for the task. Though tremendous progress has been made in digitising texts in Sanskrit, they still remain inaccessible largely due to lack of specific tools that can address linguistic peculiarities exhibited by the language (Krishna et al., 2017). From a pedagogical perspective, it will be beneficial for learners of the language to look into the prose of the verses for an easier comprehension of the concepts discussed in the verse.

Acknowledgements

We are grateful to Dr Amba Kulkarni and Dr Peter M Scharf for their valuable guidance and support throughout the work. We extend our gratitude to the anonymous reviewers for their valuable comments and suggestions to improve the quality of the paper.

References

Anja Belz, Mike White, Dominic Espinosa, Eric Kow, Deirdre Hogan, and Amanda Stent. 2011. The first surface realisation shared task: Overview and evaluation results. In *Proceedings of the Generation Challenges Session at the 13th European Workshop on Natural Language Generation*, pages 217–226, Nancy, France. Association for Computational Linguistics.

Vinayak P. Bhatta. 1990. Theory of verbal cognition (bdabodha). *Bulletin of the Deccan College Research Institute*, 49:59–74.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Simon Dennis. 2005. A memory-based theory of verbal cognition. *Cognitive Science*, 29(2):145–193.

Sergey Edunov, Myle Ott, Michael Auli, David Grangier, and Marc'Aurelio Ranzato. 2018. Classical structured prediction losses for sequence to sequence learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 355–364. Association for Computational Linguistics.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pages 1243–1252. JMLR.org.

Jiatao Gu, Hany Hassan, Jacob Devlin, and Victor O.K. Li. 2018. Universal neural machine translation for extremely low resource languages. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 344–354, New Orleans, Louisiana. Association for Computational Linguistics.

Eva Hasler, Felix Stahlberg, Marcus Tomalin, Adria de Gispert, and Bill Byrne. 2017. A comparison of neural models for word ordering. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 208–212.

Wei He, Haifeng Wang, Yuqing Guo, and Ting Liu. 2009. Dependency based chinese sentence realization. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 809–816, Suntec, Singapore. Association for Computational Linguistics.

Oliver Hellwig. 2011. DCS - The Digital Corpus of Sanskrit.

Oliver Hellwig and Sebastian Nehrdich. 2018. Sanskrit word segmentation using character-level recurrent and convolutional neural networks. In *Proceedings of the 2018 Conference on Empirical Methods* *in Natural Language Processing*, pages 2754–2763. Association for Computational Linguistics.

Douwe Kiela, Changhan Wang, and Kyunghyun Cho. 2018. Dynamic meta-embeddings for improved sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1466–1477. Association for Computational Linguistics.

Amrith Krishna, Pavan Kumar Satuluri, and Pawan Goyal. 2017. A dataset for sanskrit word segmentation. In *Proceedings of the Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature*, pages 105–114, Vancouver, Canada. Association for Computational Linguistics.

Amba Kulkarni, Preethi Shukla, Pavankumar Satuluri, and Devanand Shukl. 2015. *How Free is free Word Order in Sanskrit*. The Sanskrit Library, USA.

Mirella Lapata. 2003. Probabilistic text structuring: Experiments with sentence ordering. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 545–552, Sapporo, Japan. Association for Computational Linguistics.

Mirella Lapata. 2006. Automatic evaluation of information ordering: Kendall's tau. *Computational Linguistics*, 32(4):471–484.

Yijia Liu, Yue Zhang, Wanxiang Che, and Bing Qin. 2015. Transition-based syntactic linearization. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 113–122, Denver, Colorado. Association for Computational Linguistics.

Lajanugen Logeswaran, Honglak Lee, and Dragomir Radev. 2018. Sentence ordering and coherence modeling using recurrent neural networks. In *AAAI Conference on Artificial Intelligence*, pages 5285–5292.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *International Conference on Learning Representations*.

Peter Scharf, Anuja Ajotikar, Sampada Savardekar, and Pawan Goyal. 2015. Distinctive features of poetic syntax preliminary results. *Sanskrit syntax*, pages 305–324.

Allen Schmaltz, Alexander M. Rush, and Stuart Shieber. 2016. Word ordering without syntax. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2319–2324, Austin, Texas. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Preeti Shukla, Amba Kulkarni, and Devanand Shukla. 2016. Revival of ancient sanskrit teaching methods using computational platforms. In *Bridging the gap between Sanskrit Computational Linguistics tools and management of Sanskrit Digital Libraries Workshop*. ICON 2016, IIT BHU.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. 2016. Order matters: Sequence to sequence for sets. In *International Conference on Learning Representations (ICLR)*.

Wenhui Wang, Baobao Chang, and Mairgup Mansur. 2018. Improved dependency parsing using implicit word connections learned from unlabeled data. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2857–2863. Association for Computational Linguistics.

Sam Wiseman and Alexander M. Rush. 2016. Sequence-to-sequence learning as beam-search optimization. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1296–1306, Austin, Texas. Association for Computational Linguistics.

Yue Zhang. 2013. Partial-tree linearization: Generalized word ordering for text synthesis. In *IJCAI*, pages 2232–2238.

Yue Zhang and Stephen Clark. 2015. Discriminative syntax-based word ordering for text generation. *Computational Linguistics*, 41(3):503–538.