A Context-aware Convolutional Natural Language Generation model for Dialogue Systems

Sourab Mangrulkar
National Institute of Technology Goa
100rabmangrulkar@gmail.com

Suhani Shrivastava
National Institute of Technology Goa
suhani1396@gmail.com

Veena Thenkanidiyoor
National Institute of Technology Goa
veenat@nitgoa.ac.in

Dileep Aroor Dinesh
Indian Institute of Technology Mandi
addileep@iitmandi.ac.in

Abstract

Natural language generation (NLG) is an important component in spoken dialog systems (SDSs). A model for NLG involves sequence to sequence learning. State-of-the-art NLG models are built using recurrent neural network (RNN) based sequence to sequence models (Dušek and Jurcicek, 2016a). Convolutional sequence to sequence based models have been used in the domain of machine translation but their application as natural language generators in dialogue systems is still unexplored. In this work, we propose a novel approach to NLG using convolutional neural network (CNN) based sequence to sequence learning. CNN-based approach allows to build a hierarchical model which encapsulates dependencies between words via shorter path unlike RNNs. In contrast to recurrent models, convolutional approach allows for efficient utilization of computational resources by parallelizing computations over all elements, and eases the learning process by applying constant number of nonlinearities. We also propose to use CNN-based reranker for obtaining responses having semantic correspondence with input dialogue acts. The proposed model is capable of entrainment. Studies using a standard dataset shows the effectiveness of the proposed CNN-based approach to NLG.

1 Introduction

In task-specific spoken dialogue systems (SDS), the function of natural language generation (NLG) components is to generate natural language response from a dialogue act (DA) (Young et al., 2009). DA is a meaning representation specifying actions along with various attributes and their values. NLG plays a very important role in realizing the overall quality of the SDS. Entrainment to users way of speaking is essential for generating more natural and high quality natural language responses. Most of the approaches for incorporating entrainment are rule-based models. Recent advances have been in the direction of developing a fully trainable context aware NLG model (Dušek and Jurcicek, 2016a). However, all these approaches are based on recurrent sequence to sequence architecture.

Convolutional neural networks are largely unexplored in the domain of NLG for SDS inspite of having several advantages (Waibel et al., 1989; LeCun and Bengio, 1995). Recurrent networks depend on the computations of previous time step and thus inhibits parallelization within a sequence. Convolutional networks on the other hand, allows parallelization within a sequence resulting in efficient use of GPUs and other computational resources (Gehring et al., 2017). Multi-block (multi-layer) convolutional networks enable controlling the upper bound on the effective context size and form a hierarchical structure. In contrast to the sequential structure of RNNs, hierarchical structure provides shorter paths for modeling long-range dependencies. Recurrent networks apply variable number of nonlinearities to the inputs, whereas convolutional networks apply fixed number of nonlinearities which simplifies the learning (Gehring et al., 2017).

In this paper, we present a novel approach of using convolutional sequence to sequence model (ConvSeq2Seq) for the task of NLG. ConvSeq2Seq generator is an encoder decoder model where convolutional neural networks (CNNs) are used to build both encoder and decoder states. It uses multi-step attention mechanism. In the decoding phase, beam search is implemented and n-best natural language responses are chosen. The n-best beam search responses from ConvSeq2Seq

Proceedings of the SIGDIAL 2018 Conference, pages 191–200,
Melbourne, Australia, 12-14 July 2018.

generator may have some missing and/or irrelevant information. To address this, we propose to rank the *n*-best outputs from ConvSeq2Seq generator using convolutional reranker (CNN reranker). CNN reranker implements one dimensional convolution on beam search responses and generates binary vectors. These binary vectors are used to penalize the responses having missing and/or irrelevant information. We evaluate our model on the Alex Context natural language generation (NLG) dataset of Dušek and Jurcicek (2016a) and demonstrate that our model outperforms the RNN-based model of Dušek and Jurcicek (2016a) (TGen model) in automatic metrics. Training time of proposed model is observed to be significantly lower than TGen model. The main contributions of this work are (i) ConvSeq2Seq generator for NLG and (ii) CNN-based reranker for ranking *n*-best beam search responses for obtaining semantically appropriate responses with respect to input DA.

The rest of this paper is organized as follows. Section 2 gives a brief review of different approaches to NLG. In Section 3, proposed convolutional natural language generator (ConvSeq2Seq) is described along with CNN reranker. The experimental studies are presented in Section 4 and conclusions are given in Section 5.

2 Related Work

Natural language generation (NLG) task is divided into two phases: sentence planning and surface realization. Sentence planning generates intermediate structure such as dependency trees or templates modeling the input semantic symbols. Surface realization phase converts the intermediate structure into the final natural language response.

Conventional approaches to NLG are rule based approaches (Stent et al., 2004; Walker et al., 2002). Most recent NLG approaches include sequence to sequence RNN models (Wen et al., 2015a,b; Dušek and Jurcicek, 2016b,a). Sequence to sequence learning is to map the input sequence to a fixed sized vector using one RNN, and then to map the vector to the target sequence with another RNN. In (Wen et al., 2015a), a sequence to sequence RNN model is used with some decay factor to avoid vanishing gradient problem. The *n*-best outputs generated by the model are ranked using a CNN-based reranker. The model also uses a backward sequence to sequence RNN reranker to further improve the performance. Model proposed by Wen et al. (2015b) is a statistical language generator based on a semantically controlled long-short term memory (LSTM) structure. The LSTM generator can learn from unaligned data by jointly optimizing sentence planning and surface realization using a simple cross entropy training criterion, and language variation can be easily achieved by sampling from output candidates.

Model proposed by Dušek and Jurcicek (2016b) serves as a sequence to sequence generation model for SDS which doesn't take into account the context. The model uses single layer sequence to sequence RNN encoder decoder architecture along with attention mechanism to generate *n*-best output utterances. It then uses RNN reranker to rank the *n*-best outputs of generator to get the utterance which best describes the input DA. The model can also be used to generate deep syntax trees which can be converted to output utterance using a surface realization mechanism. This model is context unaware because it takes into account only the input DA and no preceding user utterance(s). This leads to generation of very rigid responses and also inhibits flexible interactions. Context awareness adapts/entrains to the user's way of speaking and thereby generates responses of high quality and naturalness. The semantic meaning which is required to be given in response to a query is very well modelled if context awareness is taken into account. This leads to generation of more informative response.

Model proposed by Dušek and Jurcicek (2016a) serves as a baseline sequence to sequence generation model (TGen model) for SDS which takes into account the context. The model takes into account the preceding user utterance while generating natural language output. The model implemented three modifications to the model proposed by Dušek and Jurcicek (2016b). The first modification was prepending context to the input DAs. The second modification was implementing a separate encoder for user utterances/contexts. The third modification was implementing a N-gram match reranker. This reranker is based on n-gram precision scores and promotes responses having phrase overlaps with user utterances (Dušek and Jurcicek, 2016a).

In the next section, we present the proposed CNN-based sequence to sequence generator for NLG.

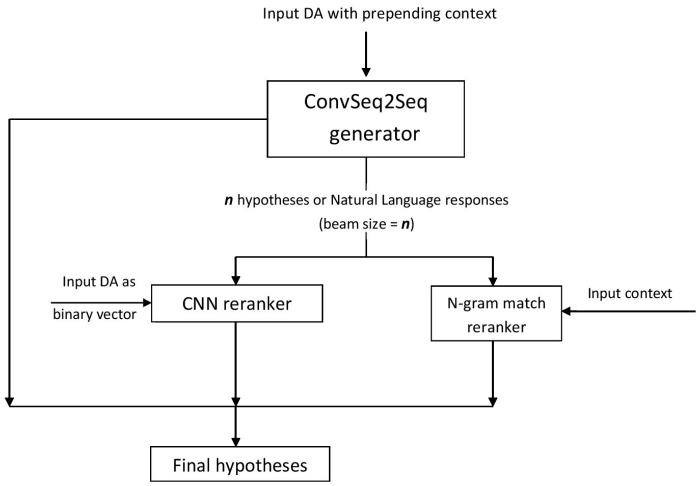

Figure 1: Pipeline of the proposed convolutional NLG model.

3 Proposed Approach

The pipeline of the proposed approach for NLG is shown in Figure 1. Input DA with prepended context is first given to convolutional sequence to sequence generator (ConvSeq2Seq) to get *n*-best natural language responses or hypotheses (*n* is beam size). These *n*-best hypotheses and binary vector representation of input DA are given as input to CNN reranker to get the misfit penalties of the hypotheses. The *n*-best hypotheses and context user utterance are given as input to the N-gram match reranker to get bigram precision scores of hypotheses. Final rank of each hypothesis *i* where $1 \leq i \leq n$ is calculated as follows:

$$rank_i = log\_probability_i$$
$$+(\omega * bigram\_precision_i)$$
$$-(W * misfit\_penalty_i)$$

Here, we get log probabilities from ConvSeq2Seq generator, bigram precision scores from N-gram match reranker and misfit penalties from CNN reranker. Here, ω and W are constants. We implement the N-gram match reranker as given by Dušek and Jurcicek (2016a). We describe the proposed convolutional sequence to sequence generator in Section 3.1 and convolutional reranker in Section 3.2.

3.1 ConvSeq2Seq Generator

The proposed sequence to sequence generator is based on convolutional sequence to sequence approach proposed by Gehring et al. (2017)[1]. It is a CNN-based encoder decoder architecture. Figure 2 shows the working of proposed ConvSeq2Seq generator on an input instance from training dataset. In this architecture, CNNs are used to compute the encoder states and decoder states. This architecture is based on succession of convolutional blocks/layers. Input sequence is represented as a combination of word and position embeddings. These embeddings are operated upon by first convolutional block and gated linear units (GLUs) to get the outputs for the first block. This can be seen in Figure 2 where only one convolutional block is shown for representation purpose. The output from first block is input to the second convolutional block and this succession follows till the last convolutional block.

Stacking of several convolutional layers/blocks allows to increase and control the effective context size. For example, stacking 10 layers of convolutional blocks, each having a kernel width of *k*=4, results in effective context size of 31 elements. Each output is dependent on 31 inputs. Stacking of several convolutional layers/blocks results in a hierarchical structure. In hierarchical structure, nearby elements interact at lower blocks and distant elements at higher blocks. It provides a shorter path for modeling long-range dependencies and eases discovery of compositional structure in sequences compared to sequential structure of RNNs. For example, to model dependencies between *n* words, only $\mathcal{O}\ (\frac{n}{k})$ convolutional operations would be required by CNN in contrast to $\mathcal{O}(n)$ operations in RNN. RNNs over-process the first word and under-process the last word,

[1]We use the implementation in the pytorch framework (Gehring et al., 2017)

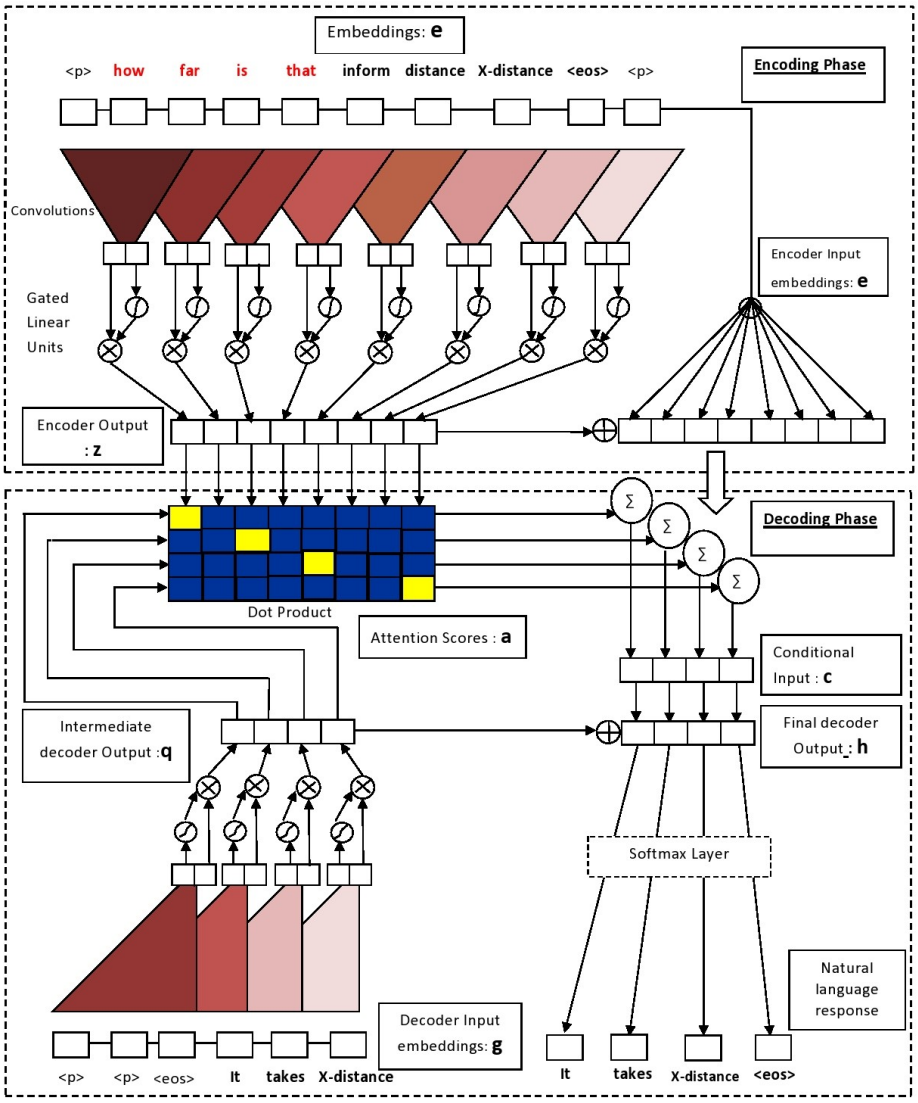

Figure 2: Working of ConvSeq2Seq generator on an input instance from training dataset. Here, for representation purpose, encoder and decoder consists of only one convolutional layer with kernel width k=3. The encoder input sequence **x** =(how, far, is, that, inform, distance, X-distance) comprises of user context "how far is that" prepended to input DA "inform distance X-distance".

whereas a constant number of kernels and nonlinearities are applied to the inputs of CNN which eases the learning process (Gehring et al., 2017).

The ConvSeq2Seq model uses position embeddings in addition to word embeddings in order to get a sense of which part of the input sequence it is currently processing (Gehring et al., 2017). Let **e** = $(w_1 + p_1,\ldots,w_s + p_s)$ be the input sequence representation, where **w** = $(w_1,\ldots,w_s)$ and **p** = $(p_1,\ldots,p_s)$ are the the word embeddings and positional embeddings of the input sequence **x** = $(x_1,\ldots,x_s)$ (having s elements) to the encoder network respectively. Intermediate states are computed based on a fixed number of input elements. In encoding phase, input is padded with $\frac{k-1}{2}$ elements on the left and right side with zero vectors. For each block l, the output $\mathbf{z}^l = (z_1^l,\ldots,z_s^l)$ is computed as follows:

$$z_i^l = \nu(\mathbf{W}_z^l[z_{i-k/2}^{l-1},\ldots,z_{i+k/2}^{l-1}] + b_z^l) + z_i^{l-1}$$

Here, $[z_{i-k/2}^{l-1},\ldots,z_{i+k/2}^{l-1}]$ is the input $\mathbf{A} \in R^{kd}$ from the previous block, $\mathbf{W}_z^l \in R^{2d\times kd}$, $b_z^l \in R^{2d}$ are parameters of convolution kernel and d is embedding dimension. Let $\mathbf{B} \in R^{2d}$ be the output of convolution kernel. $\nu()$ is gated linear unit(GLU) which is the nonlinearity function applied to the output **B** of convolution kernel. Let $\mathbf{z}^u$ be the encoder output from the last block u.

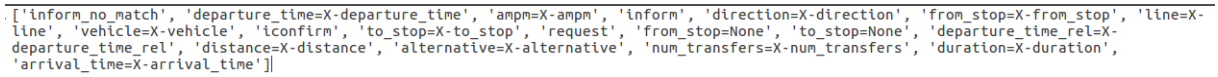

Figure 3: 19 classes of CNN reranker.

Let $\mathbf{g} = (g_1,\ldots,g_t)$ be the representation of the sequence that is being fed to the decoder network. Computation of $\mathbf{g}$ is similar to that of encoder network. Input to decoder is padded with k-1 elements on both left and right side with zero vectors to prevent decoder from having access to future information. As a result, last k-1 intermediate decoder outputs are removed. In decoding phase, for each block l, the output $\mathbf{h}^l = (h_1^l,\ldots,h_t^l)$ is computed as follows:

$$q_i^l = \nu(\mathbf{W}_q^l[h_{i-k+1}^{l-1},\ldots,h_i^{l-1}] + b_q^l) \quad (1)$$

$$d_i^l = \mathbf{W}_d^l q_i^l + b_d^l + g_i \quad (2)$$

$$a_{ij}^l = \frac{exp(d_i^l.z_j^u)}{\sum_{m=1}^{s} exp(d_i^l.z_m^u)} \quad (3)$$

$$c_i^l = \sum_{j=1}^{s} a_{ij}^l(z_j^u + e_j) \quad (4)$$

$$h_i^l = c_i^l + q_i^l + h_i^{l-1} \quad (5)$$

Here, $q^l = (q_1^l,\ldots,q_t^l)$ is the intermediate decoder output and its computation is similar to that of encoder network. For computing attention, current intermediate decoder state q_i^l is combined with the embedding of the previous target element g_i as shown in Equation (2). Equation (3) computes attention of i-th decoder state and j-th encoder output element for the l-th decoder block. Equation (4) computes the conditional input which is weighted sum of combination of encoder outputs and input embeddings. Equation (5) computes the current decoder output which is combination of conditional input, intermediate decoder output and previous layer decoder output. Let h_i^L be the decoder output of i-th element and the final decoding block L. Distribution over T possible next target elements y_{i+1} is computed as follows:

$$p(y_{i+1}|y_1,\ldots,y_i,\mathbf{x}) = \zeta(\mathbf{W}_o h_i^L + b_o) \in R^T$$

Here, ζ is softmax function, $\mathbf{W}_o$ and b_o are the weights and bias of fully connected linear layer.

3.2 CNN Reranker

The n-best beam search responses from ConvSeq2Seq model may have missing information and/or irrelevant information. CNN reranker reranks the n-best beam search responses and heavily penalizes those responses which are not semantically in correspondence with the input DA. Responses having missing information and/or irrelevant information are heavily penalized. Convolutional networks are excellent feature extractors and have achieved state-of-the-art results in many text classification and sentence-level classification tasks such as sentiment analysis, question classification, etc (Kim, 2014; Kalchbrenner et al., 2014). This classifier takes as input a natural language response and outputs a binary vector. Each element of binary vector is a binary decision on the presence of DA type or slot-value combinations. For the dataset which we have used (Dušek and Jurcicek, 2016a), there are 19 such classes of DA types and slot-value combinations. These 19 classes are shown in Figure 3.

Input DAs are converted to similar binary vector. Hamming distance between the classifier output and binary vector representation of input DA is considered as reranking penalty. The weighted reranking penalties of all the n-best responses are subtracted from their log-probabilities similar to Dušek and Jurcicek (2016a).

The architecture and working of the CNN reranker on an input instance from training dataset is shown in Figure 4. It is based on the CNN architecture proposed for sentence classification by Kim (2014). Input is a natural language response $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ where x_i's are word embeddings each having m dimensions, resulting in a input matrix of $n \times m$ dimensions. Each filter has the width equal to the size of word embeddings, i.e., m and its height specifies the number of words it will operate on. This one dimensional convolution is followed by applying activation function and 1-max pooling. The resulting feature vector has the dimension equal to the total number of filters. This penultimate layer is operated upon by a logistic layer to output the binary vector. Given penultimate layer feature vector $\mathbf{t}$, the output binary vector $\mathbf{y}$ is computed as:

$$\mathbf{y} = \sigma(\mathbf{t}.\mathbf{W}_f + \mathbf{b})$$

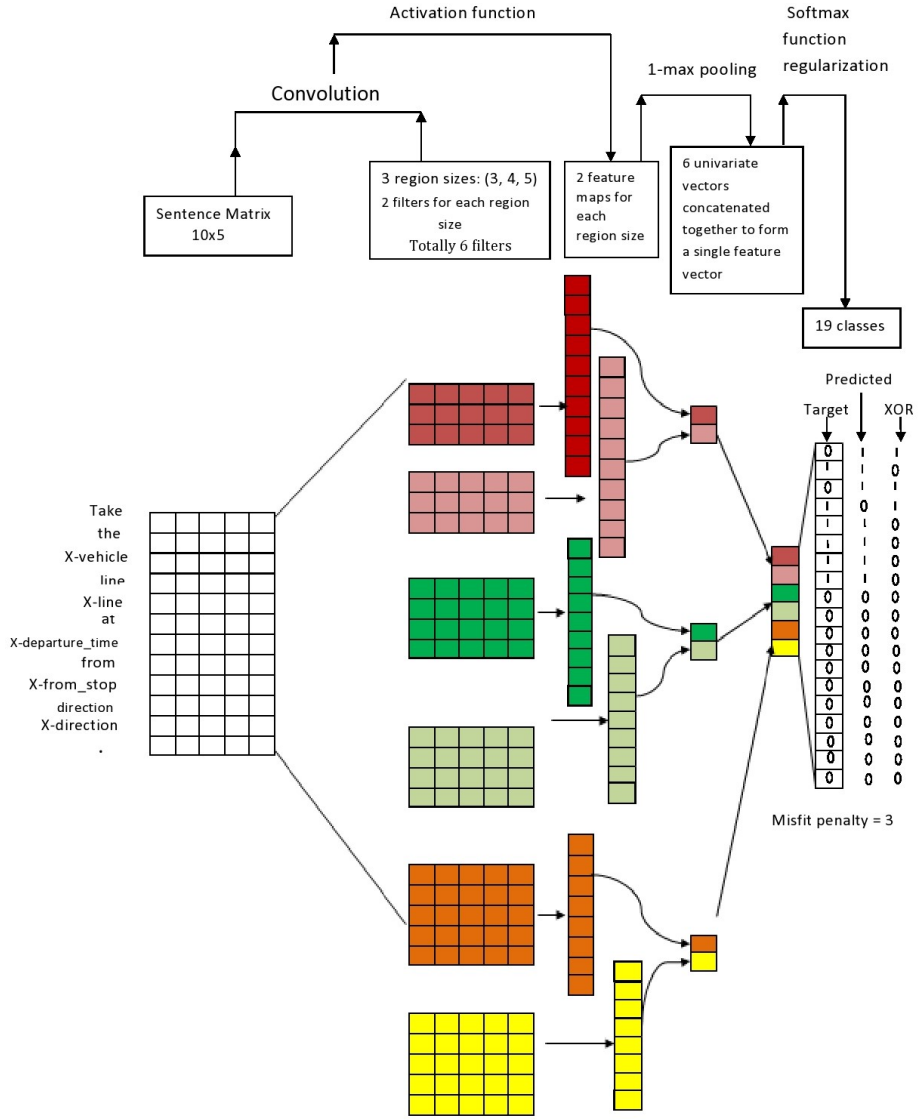

Figure 4: Architecture and working of the CNN reranker on an input instance from training dataset.

Here, σ is sigmoid activation function, $\mathbf{W}_f$ is the weight matrix and $\mathbf{b}$ is the bias vector.

The model proposed by Wen et al. (2015a) implements a CNN reranker that uses one-dimensional filters where convolutional operations are carried out on segments of words. It uses padding vectors. Proposed CNN reranker uses two-dimensional filters which operate on complete words rather than segments of words. This is more intuitive and meaningful. Also, no padding is required. The feature vector from proposed CNN reranker is v-dimensional whereas CNN reranker by Wen et al. (2015a) outputs longer feature vectors having dimension equal to $v * m$, where v = total number of filters and m = embedding size. Thus, proposed CNN reranker requires lesser number of computations.

4 Experimental Studies

The studies in this work are performed on Alex Context natural language generation (NLG) dataset (Dušek and Jurcicek, 2016a). This dataset is intended for fully trainable NLG systems in task-oriented spoken dialogue systems (SDS). It is in the domain of public transport information and has four dialogue act (DA) types namely request, inform, iconfirm and inform no match. It contains 1859 data instances each having 3 target responses. Each data instance consists of a preceding context (user utterance), source meaning representation and target natural language responses/sentences. Data is delexicalized and split into training, validation and test sets as done by Dušek and Jurcicek (2016a). For training and validation, the three paraphrases are used as separate

instances. For evaluation they are used as three target references.

Input to our ConvSeq2Seq generator is a DA prepended with user utterance. This allows entrainment of the model to the user utterances. A single dictionary is used for context utterances and DA tokens. Our model is trained by minimizing cross-entropy error using Nesterov Accelerated Gradient (NAG) optimizer (Nesterov, 1983). The hyper-parameters are chosen by cross-validation method. Based on our experiments on validation set, we use maximum sentences per batch 20, learning rate 0.07, minimum learning rate 0.00001, maximum number of epochs 2000, learning rate shrink factor 0.5, clip-norm 0.5, encoder embedding dimension 100, decoder embedding dimension 100, decoder output embedding dimension 100 and dropout 0.3. Encoder part includes 10 layers/blocks, each having 100 units and kernel width of 7. Decoder part includes 10 layers, each having 100 units and kernel width of 7. For generating outputs on test set, we choose batch size 128 and beam size 20.

For our CNN reranker, all the possible combinations of DA tokens and its values are considered as classes. We have 19 such classes. Each input is a natural language sentence and each output is a set of class labels. Training is done by minimizing cross-entropy loss using Adam optimizer (Kingma and Ba, 2015). Cross-entropy error is measured on validation set after every 100 steps. Misclassification penalty for CNN reranker is set to 100. Based on our experiments, we choose embedding dimension 128, filter sizes (3,5,7,9), number of filters 64, dropout keep probability 0.5, batch size 100, number of epochs 100 and L2 regularization, λ=0.05.

The performance of the proposed ConvSeq2Seq model for NLG is compared with that of TGen model (Dušek and Jurcicek, 2016a). For comparison, we have considered NIST (Doddington, 2002), BLEU (Papineni et al., 2002), METEOR (Denkowski and Lavie, 2014), ROUGE_L (Lin, 2004) and CIDEr metrics (Vedantam et al., 2015). For this study, we have considered script "mteval-v13a-sig.pl" (version 13a) that implements these metrics. This script was used for E2E NLG challenge (Novikova et al., 2017). We focus on the evaluations using this version. Our model has also been evaluated using the metric script "mteval-v11b.pl" (version 11b) to compare our results with those stated in (Dušek and Jurcicek, 2016a). The 13a version takes into account the closest reference length with respect to candidate length for calculation of brevity penalty. This is in accordance with IBM BLEU. On the contrary, 11b version takes shortest reference length for measuring brevity penalty. This is the reason behind higher BLEU scores in the 11b version when compared to 13a version. Both the models have been evaluated on five different metrics, with NIST and BLEU scores being of atmost importance.

We have used N-gram match reranker with the weight ω set to 1 based on experiments done on validation set. When using 11b version for evaluating automatic metrics, weight ω is set to 5.

4.1 Studies of the models using 13a version of the metrics

The comparison of the performance of the proposed model with that of TGen model using the 13a version of the metric implementation is given in Table 1. It is seen from Table 1 that there is a slight improvement in the scores of our ConvSeq2Seq generator after using CNN reranker. However, scores improve significantly when N-gram match reranker is used in addition to CNN reranker. An improvement of 3.32 BLEU points is seen. The best scores are obtained when ω is set to 1 for N-gram match reranker.

ConvSeq2Seq model in combination with CNN reranker and N-gram match reranker outperforms TGen model with N-gram match reranker in all the metrics, with a difference of 0.65 in terms of NIST score which is 8% more than the TGen NIST score on this setup. ConvSeq2Seq model with CNN reranker outperforms TGen model with RNN reranker in all the metrics, with a difference of 1.8 in terms of NIST score which is 27% more than the TGen NIST score on this setup. In the domain of NLG, NIST score is found to have highest correlation with human based judgments when compared to other metrics (Belz and Reiter, 2006). In Table 1, the bold numbers indicate the best scores.

4.2 Studies of the models using 11b version of the metrics

The comparison of the performance of the proposed model with that of TGen model using the 11b version of the metric implementation is given in Table 2. A slight improvement in the scores of our ConvSeq2Seq generator after using CNN reranker is seen in Table 2 except for BLEU score.

Evaluation metric version 13a	NIST	BLEU	METEOR	ROUGE_L	CIDEr
Baseline Model: TGen(RNN+RNN)					
Prepending context (using RNN reranker)	6.660	62.13	0.4434	0.7269	3.6956
+ N-gram match reranker	7.956	65.49	0.4655	0.7547	3.8515
ConvSeq2Seq(CNN + CNN)					
Prepending context	8.450	62.08	0.4670	0.7557	3.7281
+ CNN reranker	8.474	62.23	0.4692	0.7561	3.7412
+ CNN reranker + N-gram match reranker	**8.608**	**65.55**	**0.4762**	**0.7654**	**3.8725**

Table 1: Different automatic metric scores of TGen model (RNN+RNN) and proposed model (CNN+CNN) on test data using evaluation metric version 13a.

Evaluation metric version 11b	NIST	BLEU	METEOR	ROUGE_L	CIDEr
Baseline Model: TGen(RNN+RNN)					
Prepending context (using RNN reranker)	6.456	63.87	-	-	-
+ N-gram match reranker	7.772	69.26	-	-	-
ConvSeq2Seq(CNN + CNN)					
Prepending context	8.450	63.02	0.4670	0.7557	3.7281
+ CNN reranker	8.474	62.93	0.4692	0.7561	3.7412
+ CNN reranker + N-gram match reranker	7.920	69.60	0.4534	0.7238	3.6955

Table 2: Different automatic metric scores of TGen model (RNN+RNN) and proposed model (CNN+CNN) on test data using evaluation metric version 11b.

We see an improvement of 6.7 BLEU points when using N-gram match reranker with ω set to 5. A decrease in scores of other metrics is seen. These inconsistencies are due to the way brevity penalty is calculated for computing BLEU scores in 11b version of metric implementation.

BLEU and NIST scores of the TGen model given in Table 2 match with that represented in (Dušek and Jurcicek, 2016a). The scores of our model shows slight improvement over TGen model.

The studies done to compare the proposed model with the TGen model, show the effectiveness of considering the CNN-based approach to NLG. Studies also show that CNN reranker outperforms the RNN reranker. Further, CNN-based model is expected to take less time to train when compared to RNN-based model. We compare the time taken by the models in the next section.

4.3 Studies on the models based on training time

In this section, we compare the proposed model with that of TGen based on time taken for training. All the experiments were performed on 8GB Nvidia GeForce GTX 1080 GPU. The time taken for training ConvSeq2Seq generator is approximately 4 minutes. The time taken for training CNN reranker is approximately 2 minutes. The time taken for training TGen model is approximately 128 minutes which is 21 times more than our ConvSeq2Seq generator in combination with CNN reranker. This shows the effectiveness of using convolutional neural network in building a model for NLG than using recurrent neural network based approach used in TGen.

5 Conclusion and Future Work

In this paper a novel approach to natural language generation (NLG) using convolutional sequence to sequence learning is proposed. The convolutional model for NLG is found to encapsulate dependencies between words in a better way than recurrent neural network (RNN) based sequence to sequence learning. It is also seen that the convolutional approach makes efficient use of computational resources. The proposed model in combination with CNN reranker and N-gram match reranker is capable of entraining to users' way of speaking. Studies conducted on a standard dataset shows the effectiveness of proposed approach which outperforms the conventional RNN-based approach.

In future, we propose to perform human based evaluations to support the present performance of the model.

References

A. Belz and E. Reiter. 2006. Comparing Automatic and Human Evaluation of NLG Systems. In *11th Conference of the European Chapter of the Association for Computational Linguistics*. http://www.aclweb.org/anthology/E06-1040.

M. Denkowski and A. Lavie. 2014. Meteor Universal: Language Specific Translation Evaluation for Any Target Language. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*. Association for Computational Linguistics, pages 376–380. https://doi.org/10.3115/v1/W14-3348.

G. Doddington. 2002. Automatic Evaluation of Machine Translation Quality Using N-gram Co-occurrence Statistics. In *Proceedings of the Second International Conference on Human Language Technology Research*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, HLT '02, pages 138–145. http://dl.acm.org/citation.cfm?id=1289189.1289273.

O. Dušek and F. Jurcicek. 2016a. A Context-aware Natural Language Generator for Dialogue Systems. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. Association for Computational Linguistics, pages 185–190. https://doi.org/10.18653/v1/W16-3622.

O. Dušek and F. Jurcicek. 2016b. Sequence-to-Sequence Generation for Spoken Dialogue via Deep Syntax Trees and Strings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, pages 45–51. https://doi.org/10.18653/v1/P16-2008.

J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin. 2017. Convolutional Sequence to Sequence Learning. In *ICML*. PMLR, volume 70 of *Proceedings of Machine Learning Research*, pages 1243–1252.

N. Kalchbrenner, E. Grefenstette, and P. Blunsom. 2014. A Convolutional Neural Network for Modelling Sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 655–665. https://doi.org/10.3115/v1/P14-1062.

Y. Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, pages 1746–1751. https://doi.org/10.3115/v1/D14-1181.

D. P. Kingma and J. L. Ba. 2015. Adam: A Method for Stochastic Optimization. In *Proceedings of International Conference on Learning Representations*. pages 1–13.

Y. LeCun and Y. Bengio. 1995. *Convolutional networks for images, speech, and time series*. MIT Press, Cambridge, MA, USA, 1st edition.

C.-Y. Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*. http://www.aclweb.org/anthology/W04-1013.

Y. Nesterov. 1983. A method for unconstrained convex minimization problem with the rate of convergence $o(1/k^2)$. *Doklady AN USSR* 269:543–547. https://ci.nii.ac.jp/naid/20001173129/en/.

J. Novikova, O. Dušek, and V. Rieser. 2017. The E2E Dataset: New Challenges for End-to-End Generation. In *Proceedings of the 18th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. Saarbrücken, Germany. ArXiv:1706.09254. https://arxiv.org/abs/1706.09254.

K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. 2002. Bleu: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. pages 311–318. http://www.aclweb.org/anthology/P02-1040.

A. Stent, R. Prasad, and M. Walker. 2004. Trainable Sentence Planning for Complex Information Presentation in Spoken Dialog Systems. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL '04. https://doi.org/10.3115/1218955.1218966.

R. Vedantam, C. L. Zitnick, and D. Parikh. 2015. CIDEr: Consensus-based image description evaluation. In *CVPR*. IEEE Computer Society, pages 4566–4575.

A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang. 1989. Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 37(3):328–339. https://doi.org/10.1109/29.21701.

M. A. Walker, O. Rambow, and M. Rogati. 2002. Training a sentence planner for spoken dialogue using boosting. *Computer Speech & Language* 16(3-4):409–433. https://doi.org/10.1016/S0885-2308(02)00027-X.

T.-H. Wen, M. Gasic, D. Kim, N. Mrksic, P.-H. Su, D. Vandyke, and S. Young. 2015a. Stochastic Language Generation in Dialogue using Recurrent Neural Networks with Convolutional Sentence Reranking. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. Association for Computational Linguistics, pages 275–284. https://doi.org/10.18653/v1/W15-4639.

T.-H. Wen, M. Gasic, N. Mrkšić, P.-H. Su, D. Vandyke, and S. Young. 2015b. Semantically Conditioned LSTM-based Natural Language Generation for Spoken Dialogue Systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1711–1721. https://doi.org/10.18653/v1/D15-1199.

S. Young, M. Gašić, S. Keizer, F. Mairesse, J. Schatzmann, B. Thomson, and K. Yu. 2009. The Hidden Information State Model: a practical framework for POMDP-based spoken dialogue management. *Computer Speech and Language* 24(2):150. https://doi.org/10.1016/j.csl.2009.04.001.