Backward-Forward Sequence Generative Network for Multiple Lexical Constraints

Seemab Latif[1(✉)], Sarmad Bashir[1], Mir Muntasar Ali Agha[1], and Rabia Latif[2]

[1] National University of Sciences and Technology (NUST), Islamabad, Pakistan
{seemab.latif,sbashir.msit16seecs,magha.bese15seecs}@seecs.edu.pk
[2] College of Computer and Information Sciences, Prince Sultan University, Riyadh, Saudi Arabia
rlatif@psu.edu.sa

Abstract. Advancements in Long Short Term Memory (LSTM) Networks have shown remarkable success in various Natural Language Generation (NLG) tasks. However, generating sequence from pre-specified lexical constraints is a new, challenging and less researched area in NLG. Lexical constraints take the form of words in the language model's output to create fluent and meaningful sequences. Furthermore, most of the previous approaches cater this problem by allowing the inclusion of pre-specified lexical constraints during the decoding process, which increases the decoding complexity exponentially or linearly with the number of constraints. Moreover, some of the previous approaches can only deal with single constraint. Additionally, most of the previous approaches only deal with single constraints. In this paper, we propose a novel neural probabilistic architecture based on backward-forward language model and word embedding substitution method that can cater multiple lexical constraints for generating quality sequences. Experiments shows that our proposed architecture outperforms previous methods in terms of intrinsic evaluation.

Keywords: Recurrent Neural Networks · Natural Language Generation · Language Models · Lexical constraints · Word embedding

1 Introduction

Recently, Recurrent Neural Networks (RNNs) and their variants such as Long Short Term Memory Networks (LSTMs) and Gated Recurrent Units (GRUs) based language models have shown promising results in generating high quality text sequences, especially when the input and output are of variable length. RNN based Language Models (LM) have the ability to capture the sequential nature of language, be it for words, characters or whole sentences. This allows them to outperform other language models in sequence prediction and classification tasks. To learn the distributed representation of data efficiently by RNNs, multiple methods have been proposed such as word embeddings. It mainly include

Published by Springer Nature Switzerland AG 2020
I. Maglogiannis et al. (Eds.): AIAI 2020, IFIP AICT 584, pp. 39–50, 2020.
https://doi.org/10.1007/978-3-030-49186-4_4

Continuous Bag-Of-Words (CBOW) and Skip-Gram (SG) model [10,12]. CBOW model predicts the word as vector at a current time step, given preceding and proceeding context word vectors. The SG model is opposite in approach to predict the representation of target word vector, but same in the architecture.

Existing methods to incorporate constraints in the output sentences or generating lexical constrained sentences have multiple limitations. [13] proposed variants of backward-forward generation approach which can not handle Out-of-Vocabulary (OOV) words and only generate sentences with single lexical constraint. Similarly, [8] proposed a synchronous training approach to generate lexical constrained sequences with Generative Adversarial Networks (GANs). Moreover, various lexical constrained decoding methods have been proposed for constrained sequence generation through the extension of beam search to allow the inclusion of constraints [1,6]. Such lexical constrained decoding methods do not examine what specific words need to be included at the start of generation, but try to force specific words at each time step during the generation process at a cost of high computational complexity [14].

The remainder of this paper is organized as follows. We review the related work in Sect. 2. Section 3 describes our proposed architecture and Sect. 4 explains the dataset, experimental setup, comparison models and evaluation criteria. Section 5 gives in detail result analysis, finding and discussions about future directions. Finally, Sect. 6 concludes the paper.

2 Literature Review

In general, the purpose of LM is to capture the regularities of a language as well as its morphological and distributional properties. LM aims to compute the probability of a word sequence in order to estimate the maximum likelihood of an upcoming word to be predicted in the sequence. LM learns the distributed representation of words to interpret semantic and syntactic relations between the sequence of words. In past, RNN has shown progressive success in language modeling over traditional methods based on statistical counts. The ability of RNN Language Model (RNNLM) to learn long term contextual dependency and capturing inherited sequential nature of language makes it better than other traditional methods [11]. Particularly in sentence generation task, RNNLM performed well because of its capability of learning highly complicated structures of language. RNNLM makes Maximum A Posteriori (MAP) estimation for predicting words in a sentence [17].

Mou et al. first proposed multiple variants of Backward and Forward (B/F) language models based on GRUs for constrained sentence generation [13]. For training the B/F language models, sentences were split by choosing a word randomly. This resulted in the positional information of words getting smoothed out while generating sentences, and thus they lose the positional information of the word. This method of choosing a split word badly influences the joint probability estimation of a sentence.

Liu et al. proposed an algorithmic framework dubbed as Backward and Forward Generative Adversarial Networks (BFGAN) for constrained sentence generation [8]. BFGAN constitutes three modules; a discriminator, LSTM based backward and a forward generator with attention mechanism. The purpose of discriminator is to distinguish the real sentences from constrained sentences generated by machine and to guide the joint training of both backward and forward generators by assigning them reward signals. The backward generator takes lexical constraint as an input, which can be a word, phrase or fragment and generate the first half of the sentence backwards. The Forward generator takes the input half sentence generated by backward generator to complete the sentence with the aim of fooling the discriminator. The sentences prepared for training of backward generator relies on random splitting of sentences and the proposed framework can tackle single lexical constrained sentence generation.

Another line of work tackles the problem of constrained sentence generation by sampling the sentences from search space. Su et al. proposed a Gibbs sampling method based on Markov Chain Monte Carlo (MCMC) method for decoding constrained sentences [16]. The proposed approach consists of a discriminator and a pure language model conditioned on a bi-directional RNN. Introducing discriminator in the proposed method caters the job for calculating probability of a sentence satisfying the constraints. Gibbs method samples the set of random variables $x_1...n$ from a joint distribution, which takes the form of words to make a sentence. The shortcoming of Gibbs sampling is that it cannot change the length of sentences and hence not able to solve complicated tasks like directly generating sentences from constraints established in advance. Miao et al. extends Gibbs sampling by introducing Metropolis-Hastings for Constrained Sentence Generation (CGMH) [9]. The proposed method directly samples from the sentence space by defining local operations in the sentence space such as word replacement, insertion and deletion.

Hokamp et al. proposed Grid Beam Search (GBS) algorithm, an extension of beam search, for incorporating specified lexical constraints in the output sequences [6]. In Neural Machine Translation (NMT) task, the proposed algorithm ensures that all specified constraints must meet the hypothesis before they can be considered to be completed. To generalize image caption generative models for out-of-domain images constituting novel scenes or objects, Anderson et al. proposed a Constrained Beam Search (CBS) decoding method, which utilizes Finite-State Machine (FSM) [1]. The proposed search algorithm is capable of forcing certain image tags over resulting output sequences by recognizing valid sequences with FSM.

Table 1 summarizes techniques for generating constrained sequences. It is evident that many of the architectures are designed for specific scenarios and have high computational complexity. Due to performance gaps and inability to handle multiple constraints efficiently, a method need to be addressed. Therefore, we have proposed a neural probabilistic Backward-Forward architecture that can generate high quality sequences, with word embedding substitution method to satisfy multiple constraints.

Table 1. Comparison of different constrained sequence generation techniques.

	Multiple constraints	Computational time	Decoding complexity	Decoder	Target domain
Mou et al. [13]	x	Low	$\mathcal{O}(Nk)$	–	Research titles
Anderson et al. [1]	✓	High	$\mathcal{O}(Nk2^C)$	CBS	Image captioning
Su et al. [16]	✓	High	$\mathcal{O}(N + dNM)$	GSM	Product sentiments
Liu et al. [8]	x	–	–	Beam search	Product reviews
Hokamp et al. [6]	✓	High	$\mathcal{O}(Nk2C)$	GBS	NMT
Post et al. [14]	✓	Low	$\mathcal{O}(Nk)$	DBA	NMT
Miao et al. [9]	✓	High	$\mathcal{O}(N + dNM)$	MH	Generic
Proposed technique	✓	Low	$\mathcal{O}(Nk)$	Greedy search	Generic

3 General Model

To begin with, we state the problem of constrained sequence generation as follows: given the constraint(s) c as input, the proposed B/F LM needs to generate a fluent sequence $s = w_1, \cdots, w_v, \cdots, w_m$ maximizing the conditional probability $p(s|c)$. For this purpose, we need to select a split word in a sequence s to train the proposed B/F LM. As a sequence provides us an expression, the Parts-Of-Speech (POS) *verb* plays a vital role in placing the *subject* of a sequence into motion and offers more clarification about sequence. In this section, we first discuss the general seq2seq model for generation of sequences. After that, we discuss our proposed architecture to deal with constrained sequence generation.

Conventionally, RNNLMs for text generation are trained to maximize the likelihood of a word w_t or character c_t at time step t while given the context of previous observations in the sequence. This type of learning technique for generating sequences is known as *teacher forcing* [4]. In such learning technique, input to the recurrent neural probabilistic language model is of fixed size. The training purpose is to predict only next token until a special stop sign is generated or specific constraint is satisfied in a sequence given the context of previous observations.

In traditional seq2seq models we cannot satisfy lexical constraints, where disintegrating joint probability of a sentence $y = y_1, y_2 \cdots y_m$ for given input sentence $x = x_1, x_2 \cdots x_n$ is given by

$$p(y|x) = \prod_{i=1}^{m} p(y_i|y_1 \cdots y_{i-1}, x) \tag{1}$$

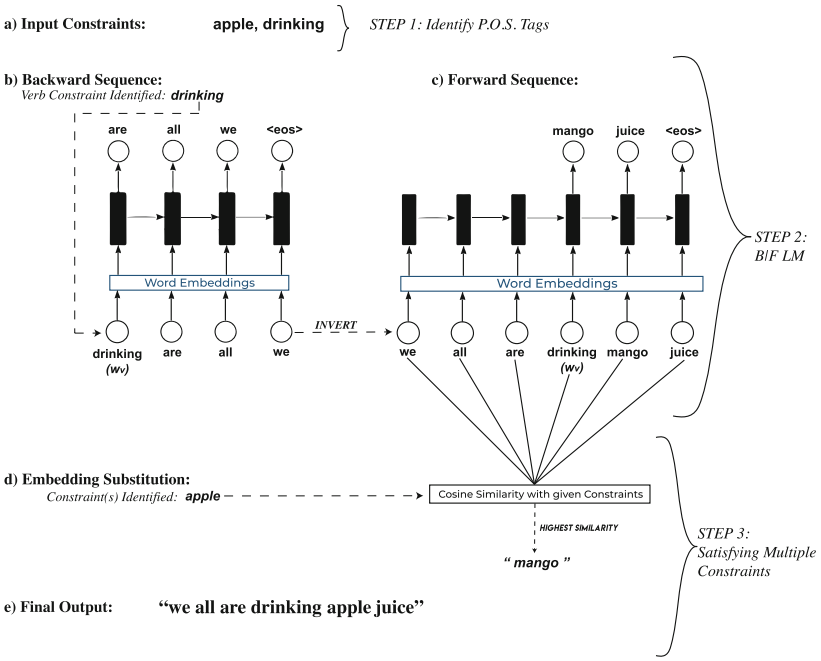

Fig. 1. An illustration of proposed system architecture

Thus, the output sentence y is predicted from y_1 to y_m in sequence either by a greedy or beam decoder. Such decomposition is because of natural language's sequential nature.

3.1 Proposed Architecture

Our proposed approach consists of a neural probabilistic architecture that is an ensemble of two LSTM based B/F LM for generating lexical constrained sequences, which captures the statistical properties of text sequences effectively. In order to generate the coherent sequences from given multiple constraints as input, we first generate the sequence from verb constraint w_v through B/F LM, and then we satisfy the other given constraints by word embedding substitution method during the inference process. The predicted verb v splits the sequence into two sub-sequences as:

$$\text{Backward Sequence} = w_{v-1}, w_{v-2}, \cdots, w_1$$
$$\text{Forward Sequence} = w_{v+1}, w_{v+2}, \cdots, w_m$$

If m denotes the length of words in a sequence s i.e. $s = w_1, \cdots, w_v, \cdots, w_m$, then the joint conditional probability of remaining m words, given lexical constraint w_v and training parameters θ can be calculated as:

$$\begin{aligned} p(s) &= p(s|w_v;\theta) \\ &= p_\theta^{bw}(s_{<v}|w_v)\,.\,p_\theta^{fw}(s_{>v}|s_1 : w_v) \end{aligned} \tag{2}$$

Where $\mathbf{p_\theta^{bw}}$ and $\mathbf{p_\theta^{fw}}$ depict the probabilities of generated sub-sequences by backward and forward language models. The sub-sequences are generated asynchronously i.e. we first generate the half sequence $s_{<v}$ in reverse order given verb constraint w_v, then generate the other half sequence $s_{>v}$ conditioned on backward sequence $s_1 : w_v$. Therefore, following the spirit of ensemble models that are trained separately, joint probability factors in Eq. 2 becomes

$$p_\theta^{bw}(s_{<v}|w_v) = \prod_{j=1}^{w_{v-1}} p_\theta^{bw}(w_{v-j}|w_v, \cdots, w_{v-j+1}) \tag{3}$$

Where $1 \leq j \leq v-1$. Backward LM decodes the output in reverse order from w_{v-1}, w_{v-2} to w_1, which is reversed again to input forward language model for decoding the complete sequence. Consequently,

$$p_\theta^{fw}(s_{>v}|s_{1:v}) = \prod_{j=1}^{m-w_v} p_\theta^{fw}(w_{v+j}|w_1, \cdots, w_{v+j-1}) \tag{4}$$

Here $1 \leq j \leq m-v$. As the output order of sub-sequence generated by backward LM is reversed again to decode the entire sequence from forward language model, therefore $\mathbf{s_{1:v}}$ is equal to $\mathbf{w_1}, \cdots, \mathbf{w_v}$.

For learning the sequences, we used LSTM networks in proposed architecture. The LSTM networks have the capability of capturing sequential data effectively where the network transforms a sequence of given input word vectors $x = x_1, \cdots, x_n$ into the sequence of hidden states $h = h_1, \cdots, h_t$ by maintaining a history of inputs at each hidden state. The LSTM cell depends on gating mechanism for information processing.

LSTM network's hidden state h at time step t is dependent on the previous state $\mathbf{h_{t-1}}$ and current input $\mathbf{x_t}$ word vectors. Particularly, in our scenario for generating variable length text sequences, the probability of an output word w_{out} from both language models calculated as:

$$p_\theta^{bw}(w_{v-t}|h_t) = Softmax(w_{out}^{bw}\, h_t + b_{out}) \tag{5}$$

$$p_\theta^{fw}(w_{v+t}|h_t) = Softmax(w_{out}^{fw}\, h_t + b_{out}) \tag{6}$$

Where w_{out}^{bw} and w_{out}^{fw} are shared across all time steps in their respective LSTM models, which projects the hidden state vector h_t into a fixed same size vector as target vocabulary in order to generate a sequence of outputs $y_t = w_{v-t}, \cdots, w_1$ for backward language model and $y_t = w_{v+t}, \cdots, w_m$ for forward language model.

The softmax function is in the final layer of LSTM network, applied to each word vector for calculating the probability distribution over vocabulary of distinct word vectors.

3.2 Word Embedding Substitution

In order to satisfy the given lexical constraints c other than verb constraint w_v, we have used a lexical substitution method based on word embedding substitution. SG model embeds both target words and their context in the same dimensional space. In this space, the vector representations of words are drawn closer together when they co-occur more frequently in a learning corpus. Thus, Cosine distance between them can be viewed as *target-to-target* distributional similarity measure. Our method relies on a natural assumption that a good lexical constraint substitution for a target word w instance in a generated sequence $s = w_1, \cdots, w_v, \cdots, w_m$ needs to be consistent with the given sequence and lexically similar to the target word w instance. During inference, we find the cosine similarity [2] of given input constraint c with every word w in a sequence s generated by the proposed B/F LM. After that, we replace the constraint c with the closest matching (least cosine distance) word w in a sequence s. Step 3 of Fig. 1 illustrates the concept. For this purpose, we have created word embedding vectorization from FastText.

4 Experiments

In this section, we introduced our experimental designs, containing the preparation of dataset for training and testing, experimental configuration, comparison architectures and evaluation criteria.

4.1 Dataset

There are many benchmark datasets for evaluating pure LM consisting of seq2seq networks for text classification and generative models, but specifically there is no such benchmark corpus for evaluation of constrained sequence generation based on statistical language models. As far, we have used Stanford Natural Language Inference (SNLI) [3] dataset for evaluation and training of proposed architecture. As we target the domain of generating sequences from lexical constraints, we extracted unlabeled sequences within range of minimum 3 and maximum 25 tokens, resulting in 451k sequences for training of proposed architecture. The proposed architecture ensemble backward-forward LM, therefore, to prepare training sequences for backward LM, following steps have been carried out:

- Annotate the tokens with their lexical categories using POS Tagging.
- Split the sentences on verb category instead of random splitting.
- Sentences with more than one verb are broken up into multiple sequences.
- After splitting the sequence on verb category, invert the half sequences.

For the forward language model, the dataset contains complete sequences for training the network. Here, it should be noted that backward language model requires only half sequences till verb token for training the network.

4.2 Word Vectorization

We follow the work of Bojanowski *et al.* [2] to create dense representations of words in dataset. A word vector is represented by augmenting the character n-grams appearing in the word, where the scoring function **s** takes into consideration the internal structure information of words, which is ignored by conventional skip-gram models [10]. The proposed model represents each word **w** as a bag of character n-gram, where adding special boundary symbols <and> at the beginning and end of words for distinguishing prefixes and suffixes from other character sequences. In addition to character n-grams of word **w**, the word **w** is also included in its set of n-grams for learning representation of each word. For example, taking the word '**apple**' and let n = 3, it will be represented by the character n-grams as <**app**, **ppl**, **ple**> and the special sequence <*apple*>.

Let a dictionary of n-grams with size **G**. Given a word w where $L_w \subset 1, ...G$ is the set of n-grams appearing in word w. Vector $\mathbf{z_g}$ represents the each n-gram g, therefore a word **w** is represented by the sum of vectors of its n-gram g. In this regard, scoring function of word w with surrounding set of word indices c is calculated by:

$$s(w, c) = \sum_{g \in L_w} z_g^T v_c \tag{7}$$

This extension of skip-gram model for creating word embedding allow the sharing of word vector representations across all words, thus enabling the reliable representational learning of rare or Out-Of-Vocabulary (OOV) words.

We have used extension of FastText's SG model to learn such data representations for both backward and forward language model given their respective data sets. In order to train the FastText model, the word embedding dimension set to 300. Min_count value set to 2, which represents that all the word frequencies lower than 2 were ignored while learning the word representations. Window size set to 5, defining the maximum distance between a current and predicted word within a sequence. Workers parameter set to 16, explaining the worker threads for faster training of FastText SG model. Epochs value set to 30 iteration, over the whole data set.

4.3 Experimental Configuration

We performed different experiments on test set to get the most optimal hyperparameters and evaluate change in performance of the model. Table 2 shows the different experimental configurations and change in performance w.r.t perplexity metric. In the proposed architecture, we get the best results by employing 2-layers of LSTM in both backward and forward language model. Both the LSTM networks were trained with Adam algorithm [7] for stochastic optimization of networks. During training, the parameters were adjusted using Adam optimizer for minimizing the training loss function, also known as misclassification rate. For calculating optimization score, we used categorical cross entropy loss function between the actual y and predicted $\hat{y}$ word probability distribution [5]. In target of accurately capturing the regularities by the neural networks and

Table 2. Hyper-parameter tuning and model performance

LSTM layers	Hidden units	LR	Drop-out	PPL score
1	256	0.01	0.2	35.48
1	512	0.001	0.3	33.15
2	256	0.01	0.2	27.48
2	**512**	**0.001**	**0.3**	**24.20**

preventing overfitting, we appended drop-out layer after every LSTM layer in both the networks. The idea of drop-out layer is to randomly drop units with their connections while training, thus preventing units from co-adapting too much. Dropping units significantly leads to major improvements than other regularization methods [15]. The epochs value was set to 50 and mini batch size was set to 128 in both the networks.

Both the Backward and Forward models are trained on NVIDIA GTX 1080 Ti GPU. The LSTM based networks are developed in keras. Training took about 17 h approx. per model with this implementation and optimal hyper-parameter configuration.

4.4 Comparison and Evaluation Metrics

We compared our proposed methodology with state-of-the art sampling method CGMH [9] for satisfying multiple constraints in a sequence. We also evaluated our methodology of verb based split generation with different variants [13], which can only handle single lexical constraint. We have used intrinsic evaluation metric that allows to determine the quality of a LM without being associated or embedded to a particular application. The most conventional intrinsic evaluation metric is perplexity (PPL). PPL of a language model given a test set $\mathbf{w} = \mathbf{w_1}, \mathbf{w_2}, ...\mathbf{w_m}$ is the inverse probability of $\mathbf{w}$ where the probability is normalized by the number of words

$$PPL(w) = \sqrt[m]{\prod_{i=1}^{m} \frac{1}{P(w_i, \cdots, w_{i-1})}} \tag{8}$$

5 Results and Discussions

For intrinsic evaluation of our proposed methodology, we first make comparisons with variants such as separate B/F and asynchronous B/F language models proposed by [13]. As mentioned earlier, in our proposed methodology the given word is verb constraint w_v through which we decode complete sequence whereas in variants of B/F, the complete sequence is decoded by random split word. We calculated PPL with both verb and random constraint as input to decode the

complete sequences. Table 3 represents the comparison in terms of PPL, where the higher probability of a sequence results in the lower of perplexity, which is better. Separate B/F variant yields worse sequences with huge perplexity score because both the B/F LM were enforced to output separately with the input constraint and concatenated after decoding of sequences. This is due to the fact that forward LM does not have the context of half sequence decoded by backward LM. Our proposed approach is more similar to asynchronous B/F LM, but technically very different as we are satisfying multiple constraints while asynchronous approach can deal with only single constraint. The results clearly shows that decoding a sequence on specific verb constraint can make use of the positional information of words in a sequence, that is smoothed out when we generate a sequence with random constraint.

Table 3. Intrinsic evaluation

Model	Perplexity	
Input constraint	Verb	Random
Separate B/F	74.56	80.43
Asynchronous B/F	26.63	28.32
Proposed B/F approach	**24.20**	**27.84**

Table 4. CGMH vs proposed

Constraints	Perplexity (PPL)	
	CGMH	Proposed B/F
1	19.34	**18.04**
2	19.71	**18.92**
3	21.36	**20.13**
4	**20.87**	21.63

Table 4 shows the comparison of our proposed approach for catering multiple constraints with CGMH [9]. Our proposed approach shows lower perplexity than CGMH sampling method for sentence generation through keywords/constraints 1 to 3, while with 4 constraints as input CGMH shows slightly better result than our approach of generating sequence with verb constraint and during inference replacing the words in sequence with closest embedding similarity. The decoding complexity of CGMH increases linearly with the number of constraints, while there is no such factor in our approach for catering multiple constraints. There is always a trade-off between fluency of sequence and decoding complexity. In practice, the downside of CGMH sampling methods is that we are not sure of which sampling step size is best for proposal distribution.

5.1 Discussion

To validate our proposed architecture of generating sequence, we performed a series of experiments. Results of intrinsic evaluation confirms that our proposed approach for sequence generation given constraint(s) outperforms previous methods. Splitting and generating a sequence on verb constraint makes use of positional information, which is smoothed out in breaking down a sequence with random word. We observe that decoding a sequence given random word as input in proposed B/F LM even performs better when the backward LM is trained over half sequences till verb. Moreover, in future we would like to explore about the

constraint-to-target context similarity, indicating their syntagmatic compatibility for improving the word embedding substitution method. Introducing attention mechanism as context vectors for constraints would be an interesting side in the proposed architecture.

6 Conclusion

In this paper, we have proposed a novel method, dubbed Neural Probabilistic Backward-Forward language model and word embedding substitution method to address the issue of lexical constrained sequence generation. Our proposed system can generate constrained sequences given multiple lexical constraints as input. To the best of our knowledge, this is the first time that multiple constraints have been handled through LSTM based backward-forward LM and word embedding substitution of the sequences. The proposed method contains a backward language model based on LSTM network, which learns the half representation of a sentence until the verb splitting word and forward language model constitute LSTM Network, learning the complete representation of a sequence. Moreover, word embedding substitution method satisfy other constraints by substituting the target word in the sequence with given constraints based on similar context in an embedding space.

References

1. Anderson, P., Fernando, B., Johnson, M., Gould, S.: Guided open vocabulary image captioning with constrained beam search. arXiv preprint arXiv:1612.00576 (2016)
2. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. Trans. Assoc. Comput. Linguist. **5**, 135–146 (2017)
3. Bowman, S.R., Angeli, G., Potts, C., Manning, C.D.: A large annotated corpus for learning natural language inference. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 632–642. Association for Computational Linguistics (2015). https://doi.org/10.18653/v1/D15-1075
4. Dai, J., Zhang, P., Mazumdar, J., Harley, R.G., Venayagamoorthy, G.: A comparison of MLP, RNN and ESN in determining harmonic contributions from nonlinear loads. In: 2008 34th Annual Conference of IEEE Industrial Electronics, pp. 3025–3032. IEEE (2008)
5. De Boer, P.T., Kroese, D.P., Mannor, S., Rubinstein, R.Y.: A tutorial on the cross-entropy method. Ann. Oper. Res. **134**(1), 19–67 (2005). https://doi.org/10.1007/s10479-005-5724-z
6. Hokamp, C., Liu, Q.: Lexically constrained decoding for sequence generation using grid beam search. arXiv preprint arXiv:1704.07138 (2017)
7. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. CoRR (2014)
8. Liu, D., Fu, J., Qu, Q., Lv, J.: BFGAN: backward and forward generative adversarial networks for lexically constrained sentence generation. IEEE/ACM Trans. Audio Speech Lang. Process. **27**(12), 2350–2361 (2019). https://doi.org/10.1109/TASLP.2019.2943018
9. Miao, N., Zhou, H., Mou, L., Yan, R., Li, L.: CGMH: constrained sentence generation by metropolis-hastings sampling. CoRR abs/1811.10996 (2018)

10. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
11. Mikolov, T., Karafiát, M., Burget, L., Černockỳ, J., Khudanpur, S.: Recurrent neural network based language model. In: Eleventh Annual Conference of the International Speech Communication Association (2010)
12. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems, pp. 3111–3119 (2013)
13. Mou, L., Yan, R., Li, G., Zhang, L., Jin, Z.: Backward and forward language modeling for constrained sentence generation. arXiv preprint arXiv:1512.06612 (2015)
14. Post, M., Vilar, D.: Fast lexically constrained decoding with dynamic beam allocation for neural machine translation. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pp. 1314–1324. Association for Computational Linguistics (2018). https://doi.org/10.18653/v1/N18-1119
15. 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**(1), 1929–1958 (2014)
16. Su, J., Xu, J., Qiu, X., Huang, X.: Incorporating discriminator in sentence generation: a Gibbs sampling method. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
17. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems 27, pp. 3104–3112. Curran Associates, Inc. (2014)