Variable Length Joint Source-Channel Coding of Text Using Deep Neural Networks

Milind Rao*, Nariman Farsad*, and Andrea Goldsmith
Electrical Engineering, Stanford University
{milind, nfarsad, andreag}@stanford.edu

***Abstract*—We consider joint source and channel coding of natural language over a noisy channel using deep learning. While the typical approach based on separate source and channel code design minimizes bit error rates, the proposed deep learning approach preserves semantic information of sentences. In particular, unlike previous work which used a fixed-length encoding per sentence, a variable-length neural network encoder is presented. The performance of this new architecture is compared to the one with fixed-length encoding per sentence. We show that the variable-length encoder has a lower word error rate compared with the fixed-length encoder as well as separate source and channel coding schemes across several different communication channels.**

I. Introduction

In communication systems, source and channel codes are typically designed separately. This is motivated by the separation theorem [1], which states that no loss in optimality is incurred if the source and channel codes are designed separately, for several classes of channel, with infinite block length codes [2]. However, the separation theorem assumes no constraint on the complexity of the source and channel code design. Therefore, in practice, when large block lengths may not be possible due to complexity and/or delay constraints, jointly designing the source and channel codes may be beneficial. Some examples demonstrating this benefit include: wireless channels [3], video transmission over noisy channels [4], and image transmission over noisy channels [5], [6].

In this work, we consider design of joint source-channel coding using deep neural networks for structured data such as text data with *small but variable* code lengths. One of the first works that considered joint source-channel coding using neural networks is [7], where simple neural network architectures were used as encoder and decoder for Gauss-Markov sources over the additive white Gaussian noise channel. There are also a number of works that use neural networks for compression without a noisy channel (i.e., only source coding). In particular, in [8], [9] image compression algorithms are developed using RNNs, which outperformed other image compression techniques. Sentence and document encoding is proposed in [10] using neural autoencoders.

Our motivation in using deep learning for the joint source-channel coding design is that in many applications, instead of recovering the exact transmitted data, we are interested in recovering the relevant information of interest from the data. In particular, for text data, instead of recovering the exact sentence at the receiver, we are interested in recovering the semantic information such as facts or meanings of the sentence. Any sentence that conveys the information in the originally transmitted sentence would be considered as an error free output by the decoder, even if it differed from the exact sentence. For example, the sentences "the car stopped" and "the automobile came to a halt" are considered equivalent. For image and video data, the information of interest could be the main object of interest in an image rather than the entire image, and any image that preserves this information is considered as an error free output by the decoder.

In this work, we build upon our previous work in [11], where we proposed a joint source-channel coding scheme for text data using deep neural networks. In particular, the architecture used in [11] encodes sentences of various length into a fixed-length bit sequence. Here, we propose a new architecture where the length of the encoding is proportional to the sentence length. We evaluate the performance of this architecture for the binary erasure channel (BEC), binary symmetric channel (BSC), and the additive white Gaussian noise (AWGN) channel. Note that in [11] we only considered the BEC channel. For our performance metric, instead of using edit distance as a measure for word error rate (WER) which does not capture semantic equivalence, we propose a new metric. Specifically, the relative distance in meaning between replaced words in the transmitted and received sentences is used to calculate the error associated with the replacement when evaluating edit distance. We show that this new architecture results in better WER performance compared to the fixed-length encoding in [11] and to separate source and channel code design for all channel models considered.

II. System Model

Our system model is defined as follows. Let $\mathcal{V}$ be the entire vocabulary, which contains the set of all the words in the language. Let $\mathbf{s} = [w_1, w_2, \cdots, w_m]$ be the sentence to be transmitted where $w_i \in \mathcal{V}$ is the i^{th} word in the sentence. At the transmitter, source and channel coding is used to convert the sentence into a sequence of bits prior to transmission. Let $\mathbf{b} = \varphi(\mathbf{s})$ be a binary vector, where the length of the vector depends on the length of the sentence, and φ be the function representing the combined effect of the source and channel encoder. The encoded bit sequence is then transmitted over a channel C. Let $\mathbf{o} = \mathsf{C}(\mathbf{b})$ be the vector representing the channel output (i.e., the observations at the receiver corresponding to each of the transmitted bits). In this work, we assume the channel is the BEC, BSC, or the AWGN channel. Therefore, $\mathbf{o}$ is not necessarily a binary vector, and it could be a vector of real or

*The authors contributed equally.

This work was funded by the TI Stanford Graduate Fellowship, NSF under CPS Synergy grant 1330081, and NSF Center for Science of Information grant NSF-CCF-0939370.

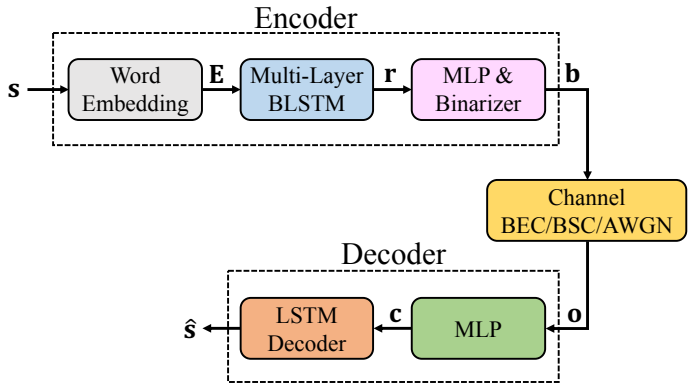


Fig. 1: The encoder-decoder architecture.

natural numbers depending on the channel considered. Let the combined effect of the source and channel decoder function be given by $\nu(\mathbf{o})$. Then $\hat{\mathbf{s}} = [\hat{w}_1, \hat{w}_2, \cdots, \hat{w}_{m'}] = \nu(\mathbf{o})$, where $\hat{\mathbf{s}}$ is the recovered sentence. Note that it is possible that $m \neq m'$. The traditional approach to designing the source and channel coding schemes is to minimize the word error rate while also minimizing the number of transmission bits. However, jointly optimizing the source coding and the channel coding schemes is a difficult problem and therefore, in practice, they are treated separately.

The problem considered in this work is jointly designing the source and channel codes by using deep neural networks in the design of φ and ν. Instead of designing these modules to minimize the word error rate, the goal is to design them such that the meaning between the transmitted sentence $\mathbf{s}$ and the recovered sentence $\hat{\mathbf{s}}$ is preserved. Therefore, the transmitted and recovered sentences may have different words and different lengths. We now describe each module in the system.

III. Neural Network Architecture

The network architecture we consider is similar to the sequence-to-sequence learning framework [12], which has resulted in state-of-the-art performance in different tasks such as machine translation [13], [14]. The complete neural network architecture is shown in Fig. 1. It has three components: the encoder, the channel, and the decoder. The encoder takes as an input a sentence $\mathbf{s}$, and outputs a bit vector $\mathbf{b}$. The channel takes an input bit vector $\mathbf{b}$ and produces an output vector $\mathbf{o}$. The effects of this module is random. The channel output $\mathbf{o}$ is the input to the decoder, and the output of the decoder is the estimated sentence $\hat{\mathbf{s}}$. We now describe each of these modules in detail.

A. The Encoder

The first step in the encoder uses an embedding vector to represent each word in the vocabulary. In this work, we initialize our embedding vectors using Glove [15]. The embedding is represented by:

$$\mathbf{E} = \varphi_e(\mathbf{s}), \tag{1}$$

where $\mathbf{E} = [\mathbf{e}_1, \mathbf{e}_2, \cdots, \mathbf{e}_m, \mathbf{e}_{\text{eos}}]$ is the $m+1$ embeddings of words in the sentence. In the second step, the embedded words are the inputs to a stacked bidirectional long short term memory (BLSTM) network [16]. The LSTM cell used in this work is similar to that used in [17]. The BLSTM stack is represented by

$$\mathbf{r} = \varphi_{\text{BLSTM}}(\mathbf{E}), \tag{2}$$

where $\mathbf{r}$ is the output state of the BLSTM stack. The output of the BLSTM is then fed into a multilayer perceptron (MLP), where the final layer of the perceptron has a tanh activation function. The MLP is used to systematically reduce or increase the dimension of $\mathbf{r}$ to ℓ_{max}, where ℓ_{max} is the maximum number of bits that is used to encode a sentence. This is given by

$$\mathbf{v} = \varphi_{\text{MLP}}(\mathbf{r}), \tag{3}$$

where $\mathbf{v} \in [-1, 1]^{\ell_{\text{max}}}$. The final encoding step is to binarize $\mathbf{v}$ from the interval $[-1, 1]$ to binary values $\{-1, 1\}$. Define a stochastic binarization function as

$$\varphi_\beta^{\text{sto}}(x) = x + Z_x, \tag{4}$$

where $Z_x \in \{1 - x, -x - 1\}$ is a random variable distributed according to

$$P(Z_x) = \begin{cases} \frac{1+x}{2} & Z_x = 1 - x \\ \frac{1-x}{2} & Z_x = -x - 1 \end{cases}. \tag{5}$$

Then final binarization step during training is

$$\mathbf{b} = \varphi_\beta^{\text{sto}}(\mathbf{v}) \tag{6}$$

for the forward pass. During the back-propagation step of the training, the gradients pass through the $\varphi_\beta^{\text{sto}}$ function unchanged. This is because the derivative with respect to the expectation $\mathbb{E}[\varphi_\beta^{\text{sto}}(\mathbf{v})] = \mathbf{v}$ is used to calculate the gradient [18]. Once the network is trained, for deployment or testing, instead of the stochastic function $\varphi_\beta^{\text{sto}}(\mathbf{v})$, the deterministic function $\varphi_\beta^{\text{det}}(\mathbf{v}) = 2u(\mathbf{v}) - 1$ is used, where $u(x)$ is the unit step function.

Since in this work we consider variable-length encoding, where the length of the encoding is dependent on the length of the sentence, only the first $\ell = L(m)$ encoded bits are transmitted. Here L is a function that maps the sentence length m into the length of transmitted bits ℓ. To model this effect in our neural network architecture, we set the last $\ell_{\text{max}} - \ell$ bits in $\mathbf{b}$ to 0. Note that bits are represented using -1 and 1 and 0 is equivalent to dropping a bit.

B. The Channel

The next module in the neural network models the channel. The function that represents the relationship between the channel input and the channel output must be differentiable function to allow for end-to-end training of the encoder and the decoder using stochastic gradient descent. In this work we consider three different channels: the BEC, the BSC, and the AWGN channel.

The BEC can be represented by a dropout layer [19],

$$\mathbf{o} = \mathsf{C}_{\text{BEC}}(\mathbf{b}, p_d), \tag{7}$$

where $\mathbf{o}$ is the vector of observations at the receiver, and p_d is the probability that a bit is dropped. The elements of $\mathbf{o}$ are in $\{-1, 0, 1\}$, where 0 indicates erasure (i.e., a dropped bit). Every bit in $\mathbf{b}$ may be dropped independent of other bits with probability p_d.

The BSC can be represented by,

$$\mathbf{o} = \mathsf{C}_{\text{BSC}}(\mathbf{b}, p_e) = \mathbf{n}_{p_e} \odot \mathbf{b}, \tag{8}$$

where $\mathbf{n}_{p_e}$ is a noise vector with every element equal to -1 with probability p_e and 1 otherwise, and $\odot$ is element-wise multiplication. Finally, the AWGN channel is represented by

$$\mathbf{o} = \mathsf{C}_{\text{AWGN}}(\mathbf{b}, \sigma^2) = \mathbf{b} + \mathbf{n}_{\sigma^2}, \tag{9}$$

where $\mathbf{n}_{\sigma^2}$ is an additive noise term with elements that are independent and identically distributed (i.i.d.) Gaussian random variables with variance σ^2.

C. The Decoder

At the receiver, the decoder first expands or shrinks the dimension of the observation vector $\mathbf{o}$ using an MLP:

$$\mathbf{c} = \nu_{\text{MLP}}(\mathbf{o}). \tag{10}$$

This module changes the dimension of the observation vector to the dimension of the initial state of the next module, a stacked LSTM decoder. The stacked LSTM decoder uses $\mathbf{c}$ as its initial state to decode the sentence word by word. This is given by

$$\hat{\mathbf{s}} = \nu_{\text{LSTM}}(\mathbf{c}, <\text{sos}>), \tag{11}$$

where $\hat{\mathbf{s}}$ is the decoded sentence. The first input to the LSTM stack is the embedding vector for a special start of the sentence symbol <sos>. Note that after the first word $\hat{w}_1$ is estimated, its embedding vector will be used as the input for the next time step. During deployment and testing we always use the estimated words and the beam search algorithm to find the most likely sequences of words [20], [13].

IV. Results

In this section we compare the deep learning approach with traditional separate source and channel code design. The source code used to generate the results is available on github[1].

A. The Dataset

The dataset we work with here is the News Crawl 2015 dataset [21]. This was obtained by crawling through the article text of various online publications in English. We first log the most popular 20000 words that appear in the dataset and make this our vocabulary. We then select sentences from lengths 4-30 with less than 20% unknown words. 90% of these sentences are placed in a training dataset and the rest in a test dataset. The training dataset has 18.5 million sentences and the test dataset has more than 2 million.

B. Deep Learning Approach

We use the pre-trained Glove embeddings [15] of dimension 200 to initialize the word embeddings of the words in our vocabulary. We also add a few special words such as unknown words, padded words, start, and end symbols with randomized initialization. In a multithreaded process, we read the input files and batch sentences of size 512 according to their lengths and this is fed to the encoder BLSTM. The encoder BLSTM has two layers of dimension 256 with peephole connections.

The resultant end states of the BLSTM layers are concatenated. In the work of [11], a dense layer is used for channel coding to bring this to a vector of dimension ℓ, the number of transmission bits. In this work, we further process the training dataset to log the frequencies of sentences of various lengths. For instance, the smallest batch (of length 4-7) are alloted 250 bits. There is a linear increment of 50 bits for subsequent batches. This results in an average bit allotment of 400 for sentences in the dataset. A dense layer produces a map to vectors of these dimensions. The decoder consists of two LSTM layers of dimension 512 with peephole connections. A learning rate of 0.001 with the Adam optimizer was employed for 6 epochs on the training dataset. A beam decoder with beam size 10 is used in the testing stage.

C. Separate Source and Channel Coding

Baselines which are separate source and channel coding schemes are used for comparison. Separate source and channel coding is optimal in the asymptote of arbitrarily large block lengths and delays. We try 3 approaches to source coding:

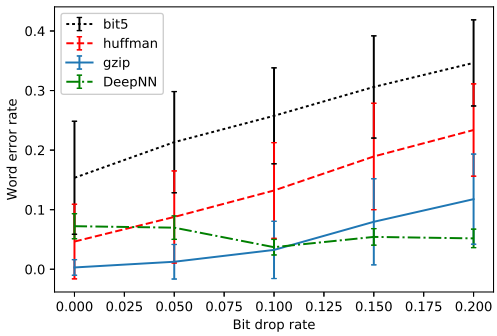
1) Universal compressors: The popular gzip tool which uses the Lempel-Ziv universal compression [22] scheme and Huffman coding is the first method. Such a universal compressor reaches the entropic limit of compression for any kind of data in the asymptote. Empirically, performance is good only for large collections of sentences. We use collection of 30 sentences for our numerical evaluations here. Note that the other baselines and the variable length deep encoder operate on single sentences (i.e., collection of sentences are not required to achieve a good compression performance).
2) Huffman coding: In the method, we encode single sentences at a time by using a Huffman encoding scheme on the characters using character frequencies obtained from the training dataset.
3) Fixed length character encoding: This is the simplest baseline where each character (a-z and some special symbols) have a fixed 5-bit encoding. The earlier baselines cannot be decoded from noisy versions of their encodings whereas we can partially retrieve the sentence from a noisy version of this baseline.

After carrying out source coding, we use Reed-Solomon codes [23] for channel coding. Given x bits of redundancy added, they can correct up to x erasures (that arise in the binary erasure channel) or $\lfloor x/2 \rfloor$ errors. Errors occur in the binary symmetric channel or the maximum likelihood bit-stream at the decoder in the AWGN channel. In the case of gzip or Huffman codes, we will not be able to decode any of the messages unless the channel coding perfectly corrects for all errors in the channel. We optimize the number of parity bits added based on channel statistics to minimize the overall word rate that arises from the following trade-off: add too few parity bits and risk failing to decode the entire sentence, or batch with high probability or add too many parity bits that will necessitate dropping words to fit in the bit budget.

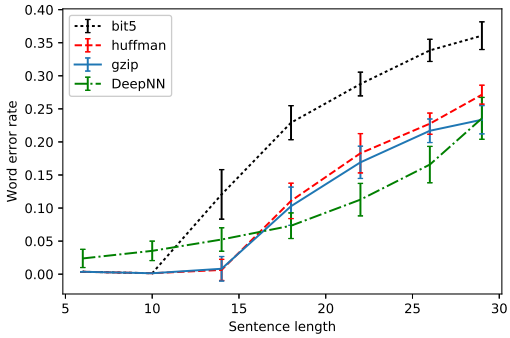
D. Performance Metric

A human judge is the best performance metric to establish the fidelity of reconstruction but is not scalable when we are looking at millions of sentences. In prior works on speech recognition

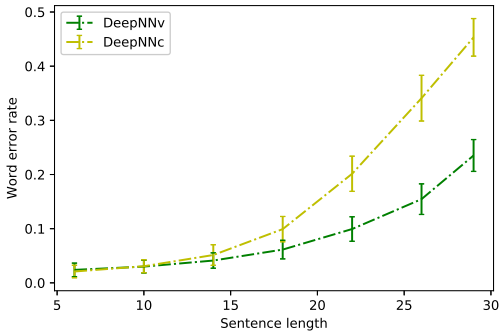
[1] https://github.com/nfarsad/NN_JSCC



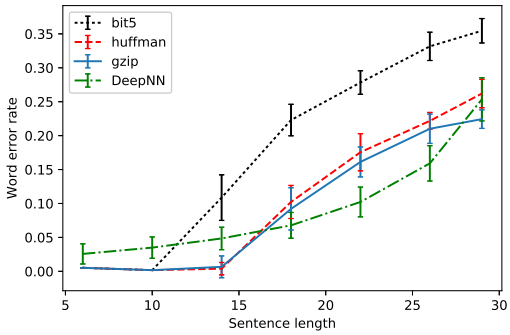
(a) Word error as erasure or bit-drop rate increases for 400 bit encoding.



(b) Word error as the sentence length changes for a binary symmetric channel with error rate 0.5.



(c) Word error with sentence length for deep neural fixed and variable encoding. Erasure channel with drop rate 0.9 and 400 bit average length encoding.



(d) Word error as the sentence length changes for an AWGN channel with standard deviation 0.6.

Fig. 2: Performance plots.

Punctuation error	TX: watch ” this week ” later . RX: watch ” this week ,” later .
Rephrasing	TX: this writer ’ s lasting impression of him regards his performance in the last new year ’ s derby . RX: this writer ’ s lasting impression of him is his performance in the last new year ’ s derby .
Rephrasing	TX: that would probably mean a drop in income for some doctors , particularly high - priced specialists , as well as for many hospitals . RX: that would probably mean a drop in income for some doctors , particularly high - grade specialists , as well as for many hospitals .
Incorrect but prescient	TX: world no . 2 <unk> <unk> and world no . 4 maria sharapova are also battling injury concerns . RX: world no . 2 <unk> <unk> and world no . <unk> maria sharapova are also battling for doping .
Subtle error	TX: if you win the game , you ’ re not going to be second - guessed as much . RX: if you win the game , you ’ re not going to be second - guessed by much .
Proper names error	TX: 19 , 2014 , in lancaster , n . y RX: 19 , 2014 , in northwest pasadena n y .
Long sentence correctly decoded	TX: the cosmic event will be visible for much of the globe , although best viewing will be in the pacific region , including parts of the western united states . RX: the cosmic event will be visible for much of the globe , although best viewing will be in the pacific region , including parts of the western united states .

TABLE I: Sample sentences which were transmitted and received using the deep learning approach.

and machine translation [11], [24], [25], the edit distance or Levenshtein distance is used to measure the dissimilarity of two sentences. The metric is obtained by using a recursive algorithm that finds the smallest sequence of insert, substitute, and delete operations on words to map one sentence to another. The length of this sequence normalized by the sentence length is the word error rate.

This automated metric, however, penalizes all substitutions equally even if a word has been replaced by a synonym. In order to factor this effect in, we propose a modified edit distance that uses a dissimilarity score between words as the substitution cost. The similarity score considered between words here is the Wu-Palmer score for relatedness. This metric still does not capture other semantic elements and will penalize synonymous phrases [26].

E. Results

In Fig. 2a, we observe the impact of the bit drop rate of the erasure channel on the word error rate. We find that gzip outperforms the other baselines as it is a more efficient compression scheme across multiple batches. In the case where

the bit drop rate is high, the deep neural network outperforms the baselines, implying the neural encoding is more resilient to channel errors.

In Fig. 2b we observe the performance of the source-channel coding schemes with a binary symmmetric channel. Here we see the word error rate as it scales with sentence lengths. As the sentence length is longer for the same bit allocation rate, the performance of the deep learning method is comparable to the baseline. The performance for the longest sentence length is poorer in the figure as the number of training instances of high sentence lengths are fewer. Finally, we note that word errors for the baselines imply that a word was not transmitted or could not be decoded whereas for the deep learning method, a word error may still occur in a sentence that has preserved the semantic information. We produce a similar plot to demonstrate the efficacy of the deep neural network joint source-channel encoder on the third channel model - the additive white Gaussian noise channel in Fig. 2d.

In Fig. 2c, we compare the fixed length encoding network of [11] to the variable length encoding network of this paper. As can be seen, the increased bit allotment for longer sentences results in fewer errors without much loss in performance for shorter sentences. The variable encoding shares the property with the fixed encoding that similar sentences have encodings that do not differ by too much.

A few representative errors are shown in Table I. Proper names have similar word vectors and are thus determined to have similar meaning. This also happens with numbers. A future direction would be to improve performance with numbers and infrequent words.

V. Conclusion

We considered the problem of joint source-channel coding of text data across a wireless channel using deep learning networks from natural language processing. Our proposed model works for a variety of channel models including erasure, binary symmetric and additive white Gaussian noise channels. The model outperforms separate source and channel coding baselines in regimes where we have fewer bits to encode a sentence by replicating the strategy of producing encodings of different lengths for sentences of different lengths. We also introduce a modified edit distance metric that does not penalize the substitution of words by synonyms as much.

In the numerical evaluations, we use a single layer for the MLP at the encoder and decoder. Since the MLP can have considerable effect in correcting errors that are introduced by the channel, a deeper multilayer MLP may be able to perform better. Other future directions would include designing networks for the joint source and channel transmission of images, audio, or video.

References

[1] Claude E Shannon and Warren Weaver, *The mathematical theory of communication*, University of Illinois press, 1998.

[2] Sridhar Vembu, Sergio Verdu, and Yossef Steinberg, "The source-channel separation theorem revisited," *IEEE Transactions on Information Theory*, vol. 41, no. 1, pp. 44–54, 1995.

[3] A. Goldsmith, "Joint source/channel coding for wireless channels," in *IEEE Vehicular Technology Conference. Countdown to the Wireless Twenty-First Century*, Jul 1995, vol. 2, pp. 614–618 vol.2.

[4] Fan Zhai, Yiftach Eisenberg, and Aggelos K Katsaggelos, "Joint source-channel coding for video communications," *Handbook of Image and Video Processing*, 2005.

[5] Geoffrey Davis and John Danskin, "Joint source and channel coding for image transmission over lossy packet networks," in *Conf. Wavelet Applications to Digital Image Processing*, 1996, pp. 376–387.

[6] Ozgun Y Bursalioglu, Giuseppe Caire, and Dariush Divsalar, "Joint source-channel coding for deep-space image transmission using rateless codes," *IEEE Transactions on Communications*, vol. 61, no. 8, pp. 3448–3461, 2013.

[7] Li Rongwei, Wu Lenan, and Guo Dongliang, "Joint source/channel coding modulation based on bp neural networks," in *Proceedings of the International Conference on Neural Networks and Signal Processing*. IEEE, 2003, vol. 1, pp. 156–159.

[8] George Toderici et al., "Variable rate image compression with recurrent neural networks," in *International Conference on Learning Representations*, 2016.

[9] George Toderici, Damien Vincent, Nick Johnston, Sung Jin Hwang, David Minnen, Joel Shor, and Michele Covell, "Full resolution image compression with recurrent neural networks," *arXiv preprint arXiv:1608.05148*, 2016.

[10] Jiwei Li, Minh-Thang Luong, and Dan Jurafsky, "A hierarchical neural autoencoder for paragraphs and documents," *arXiv preprint arXiv:1506.01057*, 2015.

[11] N. Farsad, M. Rao, and A. Goldsmith, "Deep learning for joint source-channel coding of text," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, accepted.

[12] Ilya Sutskever, Oriol Vinyals, and Quoc V Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[13] Yonghui Wu et al., "Google's neural machine translation system: Bridging the gap between human and machine translation," *CoRR*, vol. abs/1609.08144, 2016.

[14] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[15] Jeffrey Pennington, Richard Socher, and Christopher Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[16] Alex Graves and Jürgen Schmidhuber, "Framewise phoneme classification with bidirectional lstm and other neural network architectures," *Neural Networks*, vol. 18, no. 5, pp. 602–610, 2005.

[17] Haşim Sak, Andrew Senior, and Françoise Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[18] Tapani Raiko, Mathias Berglund, Guillaume Alain, and Laurent Dinh, "Techniques for learning binary stochastic feedforward neural networks," *stat*, vol. 1050, pp. 11, 2014.

[19] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting.," *Journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[20] Alex Graves, "Sequence transduction with recurrent neural networks," *arXiv preprint arXiv:1211.3711*, 2012.

[21] "News crawl dataset in second conference on machine translation wmt17," .

[22] Jacob Ziv and Abraham Lempel, "A universal algorithm for sequential data compression," *IEEE Transactions on information theory*, vol. 23, no. 3, pp. 337–343, 1977.

[23] Irving S Reed and Gustave Solomon, "Polynomial codes over certain finite fields," *Journal of the society for industrial and applied mathematics*, vol. 8, no. 2, pp. 300–304, 1960.

[24] Chris Quirk, Chris Brockett, and William Dolan, "Monolingual machine translation for paraphrase generation," in *Proceedings of the 2004 conference on empirical methods in natural language processing*, 2004.

[25] Sander Wubben, Antal Van Den Bosch, and Emiel Krahmer, "Sentence simplification by monolingual machine translation," in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*. Association for Computational Linguistics, 2012, pp. 1015–1024.

[26] Zhibiao Wu and Martha Palmer, "Verbs semantics and lexical selection," in *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 1994, pp. 133–138.