NEURAL MACHINE TRANSLATION WITH CHARACTERS AND HIERARCHICAL ENCODING

Alexander Rosenberg Johansen [a], *Jonas Meinertz Hansen* [a], *Elias Khazen Obeid* [a] *Casper Kaae Sønderby* [b], *Ole Winther* [a,b]

[a] Department for Applied Mathematics and Computer Science, Technical University of Denmark (DTU), 2800 Lyngby, Denmark
[b] Bioinformatics Centre, Department of Biology, University of Copenhagen, Copenhagen, Denmark

ABSTRACT

Most existing Neural Machine Translation models use groups of characters or whole words as their unit of input and output. We propose a model with a hierarchical `char2word` encoder, that takes individual characters both as input and output. We first argue that this hierarchical representation of the character encoder reduces computational complexity, and show that it improves translation performance. Secondly, by qualitatively studying attention plots from the decoder we find that the model learns to compress common words into a single embedding whereas rare words, such as names and places, are represented character by character.

1. INTRODUCTION

Neural Machine Translation (NMT) is the application of deep neural networks to translation of text. NMT is based on an end-to-end trainable algorithm that can learn to translate just by being presented with translated language pairs. Despite being a relatively new approach, NMT has in recent years surpassed classical statistical machine translation models and now holds state-of-the-art results [Chung et al., 2016, Luong and Manning, 2016, Wu et al., 2016].

Early NMT models introduced by Cho et al. [2014a], Kalchbrenner and Blunsom [2013], Sutskever et al. [2014] are based on the *encoder-decoder* network architecture. Here the *encoder* compresses an input sequence of variable length from the source language to a fixed-length vector representing the sentiment of the sentence. The *decoder*, takes the fixed-length representation as input and produces a variable length translation in the target language. However, due to the fixed length representation the naïve encoder-decoder approach have limitations when translating longer sequences.

To overcome this shortcoming, the *attention* mechanism proposed by Bahdanau et al. [2014] assists the decoder by learning to selectively attend to parts of the input sequence, which it deems most relevant for generating the next element in the output sequence and effectively reducing the distance from encoding to decoding. This approach has made it possible for encoder-decoder models to produce high quality translations over longer sequences. However, it suffers from the significant amount of computational power and memory needed to compute the relevance of every element of the input sequence for every element of the output sequence.

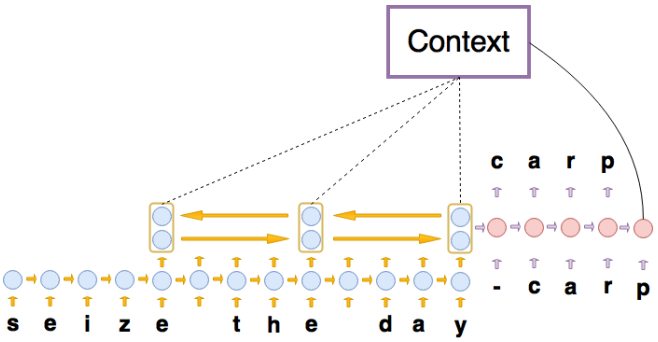

Fig. 1. Our `char2word-to-char` model using hierarchical encoding in English and decoding one character at a time in Latin. "-" marks sequence start for decoding.

For this reason, training the models on individual characters is not practical, and most current solutions instead use word segmentation [Bahdanau et al., 2014, Sutskever et al., 2014] or multiple characters [Schuster and Nakajima, 2012, Sennrich et al., 2015, Wu et al., 2016] to represent sentences. However, this high-level segmentation approach has a set of drawbacks; most Latin based languages have millions of words with the majority occurring rarely. To handle this, current models use confined dictionaries with only the k most common words with the remaining words being represented by a special `<UNK>`-token [Bahdanau et al., 2014, Sutskever et al., 2014]. As a result, names, places, and other rare words are typically translated as unknown by these models. Further, when all words are represented as separate entities, the model has to learn how every word is related to one-another, which can be challenging for rare words even if they are obvious variations of more frequent words [Luong and Manning, 2016]. Figure 2 illustrates some of the challenges of characters versus words for the encoder-decoder model.

Two branches of methods to circumvent these drawbacks have been proposed. The first branch is based on extending

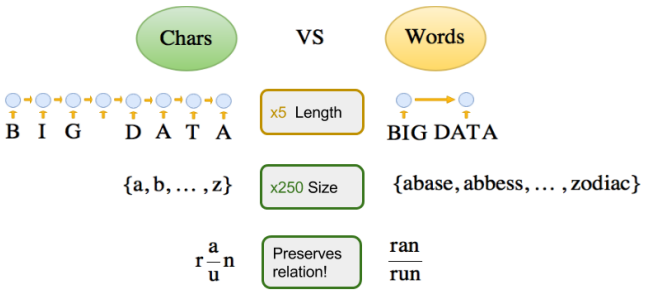

Fig. 2. Challenges in *encoder-decoder* models: Characters versus Words.

the current word-based encoder-decoder model to incorporate modules, such as dictionary look-up for out-of-dictionary words [Jean et al., 2014, Luong et al., 2014]. The second branch, which we will investigate in this work, is moving towards smaller units of computation.

In order to represent sentence context over longer sequences, memory based recurrent neural networks (RNNs), such as the long-short term memory (LSTM) [Gers et al., 2000, Hochreiter and Schmidhuber, 1997] cell, are a key ingredient to a state-of-the-art encoder-decoder model [Bahdanau et al., 2014, Sutskever et al., 2014]. Memory based RNNs allows parametrization of cells giving the neural network the ability to exhibit a dynamic temporal behaviour when encoding and decoding a sentence. The gated recurrent unit (GRU) is our preferred choice. The GRU is a LSTM variant that simplifies some of the steps from the original LSTM function for easier implementation and faster computation. However, these modifications might cause the RNN difficulty to solve tasks that require to learn stacks and push/pop ops or their equivalents, e.g., for context sensitive grammars. [Gers and Schmidhuber, 2001]

In this paper we demonstrate models that use a new *char2word* mechanism (illustrated in figure 1) during encoding, which reduces long character-level input sequences to word-level representations. This approach has the advantages of keeping a small alphabet of possible input values and preserving relations between words that are spelled similarly, while significantly reducing the number of elements that the attention mechanism needs to consider for each output element it generates. Using this method the decoder's memory and computational requirements are reduced, making it feasible to train the models on long sequences of characters as input and output. Thus avoiding the drawbacks of word based models described above. And lastly, we give a qualitative analysis of attention plots produced by a character-level encoder-decoder model with and without the hierarchical encoding mechanism, char2word. This shed light on how a character-level model uses attention, which might explain some of the success behind the BPE and hybrid models.

2. RELATED WORK

Other approaches to circumventing the increase in sequence lengths while reducing the dictionary size have been proposed: First, byte-pair Encoding (BPE) [Sennrich et al., 2015], currently holding state-of-the-art in the WMT'14 English-to-French and English-to-German [Wu et al., 2016], where the dictionary is a combination of the most common characters. In practice this means that most frequent words are encoded using fewer bits than less frequent words. Secondly, a hybrid model [Luong and Manning, 2016] where a word encoder-decoder consults a character encoder-decoder when confronted with out-of-dictionary words. Thirdly, pre-trained word-based encoder-decoder models with character input used for creating word embeddings [Ling et al., 2015] have been shown to achieve similar results to word-based approaches. As a last mention, character decoder with BPE encoder has shown to be end-to-end trained successfully [Chung et al., 2016].

Wu et al. [2016] provides a good summary and large-scale demonstration of many of the techniques that are known to work well for NMT and RNNs in general. The RNN encoder-decoder with attention approach is used not only within machine translation, but can be regarded as a general architecture to extract information from a sequence and answer some type of question related to it [Kumar et al., 2015].

3. MATERIALS AND METHODS

First we give a brief description of the Neural Machine Translation model, afterwards we will go into detail of explaining our proposed architecture for character and word segmentation.

3.1. Neural Machine Translation

From a probabilistic perspective, translation can be defined by maximizing the conditional probability of $\arg\max_y p(y|x)$, where $y_1, y_2, \ldots, y_{T_y}$ is the target sequence and $x_1, x_2, \ldots, x_{T_x}$ is the source sequence. The conditional probability $p(y|x)$ is modelled by an encoder-decoder model where the encoder and the decoder are modelled by separate Recurrent Neural Networks (RNNs) and the whole model is trained end-to-end on a large parallel corpus. The model uses memory based RNN variants, as they enable modelling of longer dependencies in the sequences [Cho et al., 2014b, Hochreiter and Schmidhuber, 1997].

The encoder part (input RNN) computes a set of hidden representations, $h_1, h_2, \ldots, h_{T_x}$ based on the input

$$h_t = h(x_t, h_{t-1}) \tag{1}$$

where h is a RNN with memory cells, $h_t \in \mathbb{R}^{m_h}$ is a hidden state representation at time step t, with m_h hidden units.

The decoder part (output RNN) then computes a context vector, c_t, based on the hidden representations from the encoder:

$$c_t = q(h_1, h_2, \ldots, h_{T_x}), \tag{2}$$

where q is a function that takes a set of hidden representations and returns a context vector $c_t \in \mathbb{R}^{m_c}$ where m_c number of context units. For a decoder without attention, the value of c_t is the same for all time steps.

Finally the decoder combines the previous predictions, $y_{<t}$, and the context vector, c_t, to predict the next unit (word, BPE, or character), such that it maximises the log conditional probability

$$\log p(y \mid x) = \sum_{t=1}^{T_y} \log p(y_t \mid y_{<t}, c), \tag{3}$$

$$p(y_t \mid y_{<t}, c_t) = g(y_{t-1}, s_t, c_t), \tag{4}$$

where g is a non-linear, potentially multi-layered function that outputs the probability of y_t, ande s_t is the hidden state of the decoder RNN, such that

$$s_t = f(s_{t-1}, y_{t-1}, c_t). \tag{5}$$

We minimise the cross entropy loss averaged over all time steps with n sized mini-batches and add L^2 regularisation, such that

$$J = -\sum_{i=1}^{n} \log p(y_i \mid x_i) + \lambda \left(\sum_{n'=1}^{N'} \theta_{n'}^2 \right),$$

where λ is a tune-able hyper-parameter, N' is the number of non-bias weights and θ is the weights in the neural network.

3.1.1. Attention

As motivated in section 1, the attention mechanism can compute a new context vector c_t for every time step by combining the hidden representations from the encoder as well as the previous hidden state, s_{t-1}, of the decoder

$$c_t = \sum_{j=1}^{T_x} a_{tj} h_j \tag{6}$$

where the weight parameter a_{tj} of each annotation h_j is computed as

$$a_{tj} = \frac{\exp(e_{tj})}{\sum_k^{T_x} \exp(e_{tk})} \tag{7}$$

and we have that

$$e_{tj} = a(s_{t-1}, h_j) \tag{8}$$

where a_{tj} and e_{tj} reflect the importance of h_j, w.r.t. the previous decoder state s_{t-1}. The attention function, a, is a non-linear, possibly multi-layered neural network.

The encoder-decoder and attention model is trained jointly to minimise the loss function.

4. OUR MODEL

We propose two models: The *char-to-char* NMT model and the *char2word-to-char* NMT model. Both models build on the encoder-decoder model with attention as defined in section 3.1 and section 3.1.1. Below we will give specific model definitions.

4.1. The `char` encoder

Our character-level encoder (referred to as the `char` encoder) is built upon a bi-directional RNN [Schuster and Paliwal, 1997]. The encoder function, h_t, in equation (1) becomes

$$h_t = \begin{bmatrix} h_f(Ex_t, \overrightarrow{h_{t-1}}) \\ h_b(Ex_t, \overleftarrow{h_{t+1}}), \end{bmatrix} \tag{9}$$

where $x_t \in \{0, 1\}^{m_x}$ is a one-hot encoded vector and m_x is the amount of input classes, $E \in \mathbb{R}^{m_e \times m_x}$ is an embedding matrix with m_e being the size of the embedding, h_f and h_b are RNN functions and h is initialised as $h_0 = 0$.

The `char` encoder is illustrated with the yellow arrows and blue circles in figure 3.

4.2. The `char2word` encoder

The character-to-word-level encoder (referred to as the `char-2word` encoder) samples states from the forward pass of the `char` encoder defined in the above section. The states it samples are based on the locations of spaces in the original text, resulting in a sequence of outputs from the `char` encoder that essentially represents the words from the text and acts as their embeddings. We sample the indices from $\overrightarrow{h}$, such that

$$h_t^{spaces} = \overrightarrow{h}_{\varphi t}, \tag{10}$$

where $\overrightarrow{h}_t$ is defined from above equation (9) and φ is an ordered list of indices defining spaces in the input sequence x. Given h^{spaces} equation (9) is used with h_t^{spaces} replacing Ex_t.

The `char2word` encoder is illustrated with the yellow arrows and blue circles in figure 1.

A result of this "downsampling" by using spaces, the `char2word` encoder only has about a fifth of the hidden states the `char` encoder has. As we described in the introduction, the computationally expensive part of the encoder-decoder with attention is the attention part. By significantly reducing the encoder we could train the `char2word` encoder in half the time compared to the `char` encoder.

4.3. The `char` decoder

Our character-level decoder (referred to as the `char` decoder) works with characters as the smallest unit of computation and decodes one character at a time. The decoder uses a RNN

and the attention mechanism [Bahdanau et al., 2014] when decoding each character.

The new state in our decoder RNN, s_t, as defined in equation (5) is computed as follows

$$s_t = f(s_{t-1}, y_{t-1}, c_t), \tag{11}$$

$$s_t = f(\begin{bmatrix} E' p_{t-1} \\ c_t \end{bmatrix}, s_{i-1}) \tag{12}$$

$$p_{t-1} = \arg\max(y_{t-1}), \tag{13}$$

where $p_{t-1} \in 0, 1^{k_p}$ is a one-hot encoded vector with k_p being the amount of input classes, $E' \in \mathbb{R}^{m'_e \times m_p}$ is an embedding matrix with m'_e being the size of the embedding, f is a RNN function and s is initialised as $s_0 = h_{T_x}$.

4.3.1. *Attention mechanism*

The attention model a (defined in equation (14)) is used to compute the context c_t for time step t, which is utilised by the decoder to perform variable length attention. The attention function, a, was parametrized as

$$a(s_{t-1}, h_j) = v_a^T \tanh(W_a s_{t-1} + U_a h_j + b_a), \tag{14}$$

where $W_a \in \mathbb{R}^{m_s \times m_s}$, $U_a \in \mathbb{R}^{m_s \times m_h}$, $v_a \in \mathbb{R}^{m_h}$, $b_a \in \mathbb{R}^{m_h}$, m_s is the amount of hidden units in the decoder and m_h is the amount of hidden units in the encoder. As $U_a h_j$ does not depend on t, we can pre-compute it in advance for optimisation purposes.

4.3.2. *Output function*

The output of the decoder $g(y_{t-1}, s_t, c_t)$ uses a linear combination of the current hidden state in the decoder, s_t, followed by a softmax function.

$$g(y_{t-1}, s_t, c_t) = \frac{\exp(W_y s_t + b_y)}{\sum_{k=1}^{K} \exp(W_y s_i + b_y)}, \tag{15}$$

where $W_y \in \mathbb{R}^{K \times m_s}$, $b_y \in \mathbb{R}^K$ and K is the amount of output classes.

We use the same decoder with attention for both the `char-to-char` and `char2word-to-char` model, which is illustrated in figure 1 and figure 3. The main difference is that our figure 1 model has significantly lower amount of units to attend over.

5. EXPERIMENTS

All models were evaluated using the BLEU score[1] [Papineni et al., 2002].

[1]We used the `multi-bleu.perl` script from Moses (`https://github.com/moses-smt/mosesdecoder`).

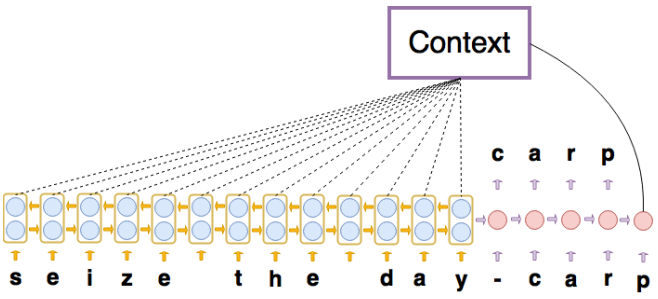

Fig. 3. Our `char-to-char` model encoding and decoding a sentence from English to Latin on character level. "-" marks sequence start for decoding.

5.1. Data and Preprocessing

We trained our models on two different datasets of language pairs from the WMT'15: `En-De` (4.5M) and `De-En` (4.5M). For validation we used the `newstest2013` and for testing we used `newstest2014` and `newstest2015`.

The data preprocessing applies is identical to Chung et al. [2016] on `En-De` and `De-En` with the source sentence length set to 250 characters instead of 50 BPE units. In short, that means; We normalise punctuations and tokenise using Moses scripts[2]. We exclude all samples where the source sentence exceed 250 characters and the target sentence exceed 500 characters. The source and target language has separate dictionaries, each containing the 300 most common characters. Characters not in the dictionary is replaced with an unknown token.

5.2. Training details

The model hyperparameters are listed in tables 1 and 2. For the RNN functions in the encoder and the decoder we use gated recurrent units (GRU) [Cho et al., 2014b]. For training we use back-propagation with stochastic-gradient descent using the Adam optimiser [Kingma and Ba, 2014] with a learning rate of $\alpha = 0.001$. For L2 regularization we set $\lambda = 1 \times 10^{-6}$. In order to stabilise training and avoid exploding gradients, the norms of the gradients are clipped with a threshold of 1 before updating the parameters. All models are implemented using TensorFlow [Abadi et al., 2016] and the code and details of the setup are available on GitHub[3].

5.2.1. *Batch details*

When training with batches, all sequences must be padded to match the longest sequence in the batch, and the recurrent layers must do the full set of computations for all samples and all

[2]From Moses (`https://github.com/moses-smt/mosesdecoder`) using `normalize-punctuation.perl` and `tokenizer.perl`

[3]`https://github.com/Styrke/master-code`

layer	no. units
input alphabet size (X)	300
embedding sizes	256
char RNN (forward)	400
char RNN (backward)	400
attention	300
char decoder	400
target alphabet size (T)	300

Table 1. Hyperparameter values used for training the `char-to-char` model. Where Σ_{src} and Σ_{trg} represent the number of classes in the source and target languages, respectively.

layer	no. units
input alphabet size (X)	300
embedding sizes	256
char RNN (forward)	400
spaces RNN (forward)	400
spaces RNN (backward)	400
attention	300
char decoder	400
target alphabet size (T)	300

Table 2. Hyperparameter values used for training the `char2word-to-char` model. Where Σ_{src} and Σ_{trg} represent the number of classes in the source and target languages, respectively.

timesteps, which can result in a lot of wasted resources [Hannun et al., 2014] (see figure 4). Training translation models is further complicated by the fact that source and target sentences, while correlated, may have different lengths, and it is necessary to consider both when constructing batches in order to utilize computation power and RAM optimally.

To circumvent this issue, we start each epoch by shuffling all samples in the dataset and sorting them with a stable sorting algorithm according to both the source and target sentence lengths. This ensures that any two samples in the dataset that have almost the same source and target sentence lengths are located close to each other in the sorted list while the exact order of samples varies between epochs. To pack a batch we simply started adding samples from the sorted sample list to the batch, until we reached the maximal total allowed character threshold (which we set to 50,000) for the full batch with padding after which we would start on a new batch. Finally all the batches are fed in random order to the model for training until all samples have been trained on, and a new epoch begins. Figure 5 illustrates what such dynamic batches might look like.

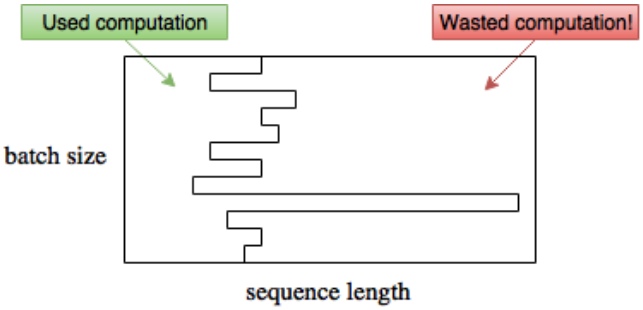

Fig. 4. A regular batch with random samples.

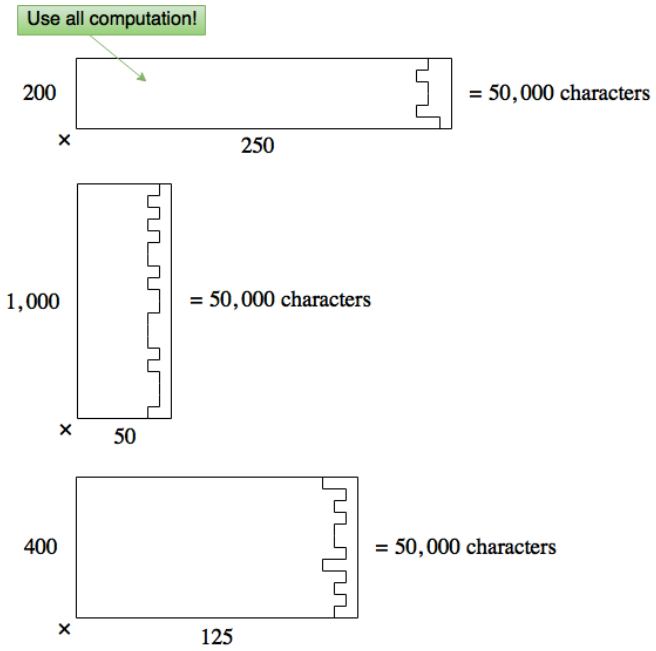

Fig. 5. Our dynamic batches of variable batch size and sequence length.

5.3. Results

5.3.1. Quantitative

The quantitative results of our models are illustrated in table 3. Notice that the `char2word-to-char` model outperforms the `char-to-char` model on all datasets (average 1.28 BLEU performance increase). This could be an indication that either having hierarchical, word-like, representations on the encoder or simply the fact that the encoder was significantly smaller, helps in NMT when using a character decoder with attention.

Model	Language	*validation set* newstest2013	*test sets* newstest2014	newstest2015
`char-to-char`	De–En	18.89	17.97	18.04
`char2word-to-char`	De–En	**20.15**	**19.03**	**19.90**
`char-to-char`	En–De	15.32	14.15	16.11
`char2word-to-char`	En–De	**16.78**	**15.04**	**17.43**

Table 3. Results: WMT'15, *newstest2013* was used as validation set, *newstest2014* and *newstest2015* were used as test sets. The results with bold indicates the best results on that dataset.

5.3.2. Qualitative

Plotting the weights of a_{tj} (defined at equation (14)) is popular in NMT research, as these gives an indication of where the model found relevant information while decoding. We have provided plots of both our `char-to-char`- and `char2word-to-char` models in figures 6 and 7. The more intense the blue colour, the higher the values of a_{tj} at that point. Notice that each column corresponds to the decoding of a single unit, resulting in each column summing to 1.

The char-to-char attention plot, attending over every character, interestingly indicates that words that would normally be considered out-of-dictionary (see *Lisette Verhaig* in figure 6) are translated character by character-by-character, whereas common words are attended at the end/start of each word [4] to use as a single embedding. This observation might explain why using hierarchical encoding improves performance. BPE based models and the hybrid word-char model by Luong and Manning [2016] effectively works in the same manner, when translating common words BPE- and hybrid word-char models will work on a word level, whereas with rare words the BPE will work with sub-parts of the word (maybe even characters) and the hybrid approach will use character representations.

The `char2word-to-char` attention plot has words, or character-made embeddings of words, to perform attention over. The attention plot seems very similar to the BPE-to-Char plot proposed by Chung et al. [2016]. This might indicate that it is possible to imitate lexeme (word) based models using smaller dictionaries and preserving relationship between words.

6. CONCLUSION

We have proposed a pure character based encoder-decoder model with attention using a hierarchical encoding. We find that the hierarchical encoding, using our newly proposed char2word encoding mechanism, improves the the BLEU score by an average of 1.28 compared to models using a standard character encoder.

[4] As we use a bi-directional RNN, full information will be available at both the end and start of a word

Qualitatively, we find that the attention of a character model without hierarchical encoding learns to make hierarchical representations even without being explicitly told to do so, by switching between word and character embeddings for common and rare words. This observation is in line with current research on Byte-Pair-Encoding- and hybrid word-character models, as these models uses word like embeddings for common words and sub-words or characters for rare words.

Furthermore, qualitatively we find that our hierarchical encoding finds lexemes in the source sentence when decoding similarly to current models with much larger dictionaries using Byte-Pair-Encoding.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Gregory S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian J. Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Józefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Gordon Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul A. Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda B. Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *CoRR*, abs/1603.04467, 2016. URL `https://arxiv.org/abs/1603.04467`. Software available from tensorflow.org.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014. URL `https://arxiv.org/abs/1409.0473`.

KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *CoRR*, abs/1409.1259, 2014a. URL `https://arxiv.org/abs/1409.1259`.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014b. URL `https://arxiv.org/abs/1406.1078`.

Junyoung Chung, Kyunghyun Cho, and Yoshua Bengio. A character-level decoder without explicit segmentation for neural machine translation. *CoRR*, abs/1603.06147, 2016. URL `https://arxiv.org/abs/1603.06147`.

Felix A Gers and E Schmidhuber. Lstm recurrent networks learn simple context-free and context-sensitive languages. *IEEE Transactions on Neural Networks*, 12(6): 1333–1340, 2001.

Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. *Neural computation*, 12(10):2451–2471, 2000.

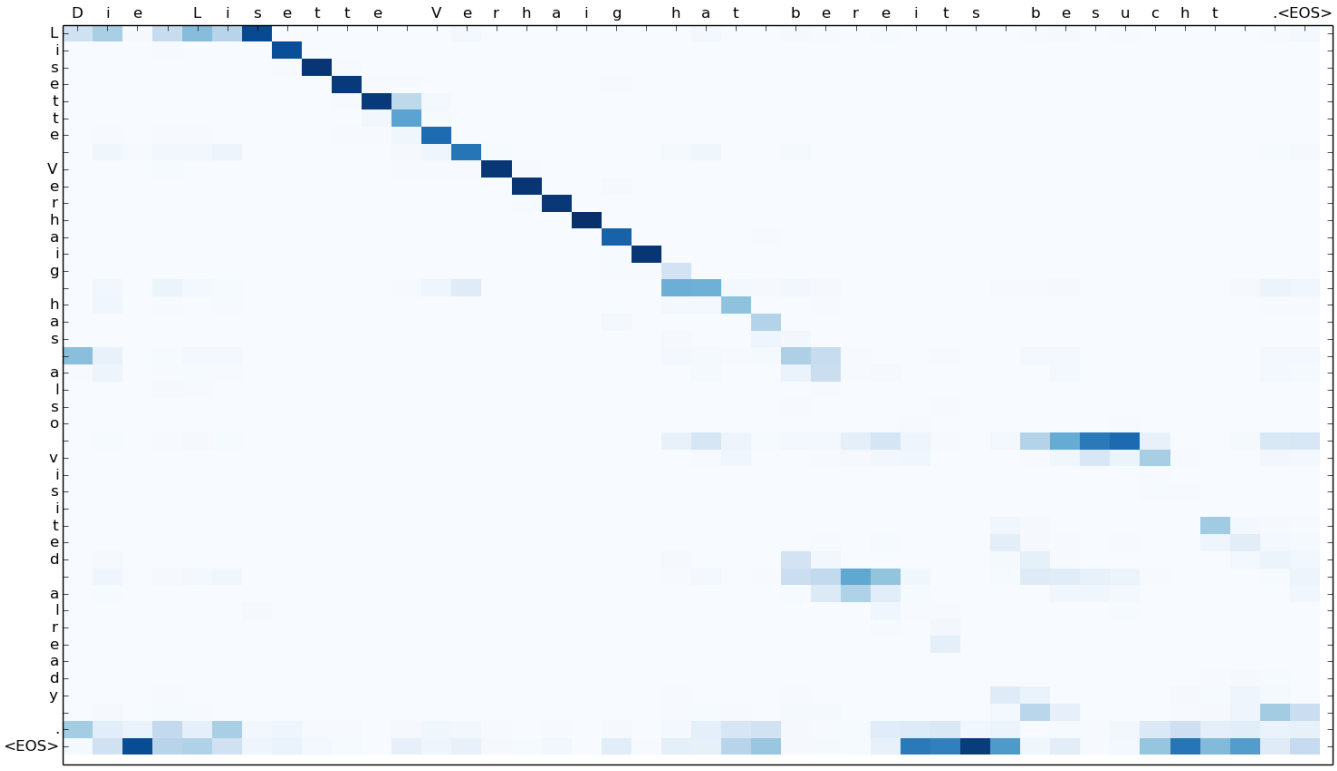

Fig. 6. Attention plot of our `char-to-char` model encoding and decoding a sentence from English to German.

Awni Y. Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, and Andrew Y. Ng. Deep speech: Scaling up end-to-end speech recognition. *CoRR*, abs/1412.5567, 2014. URL `https://arxiv.org/abs/1412.5567`.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL `http://dx.doi.org/10.1162/neco.1997.9.8.1735`.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. On using very large target vocabulary for neural machine translation. *CoRR*, abs/1412.2007, 2014. URL `https://arxiv.org/abs/1412.2007`.

Nal Kalchbrenner and Phil Blunsom. Recurrent continuous translation models. In *EMNLP*, volume 3, page 413, Seattle, October 2013. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL `https://arxiv.org/abs/1412.6980`.

Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. Ask me anything: Dynamic memory networks for natural language processing. *CoRR*, abs/1506.07285, 2015. URL `https://arxiv.org/abs/1506.07285`.

Wang Ling, Isabel Trancoso, Chris Dyer, and Alan W. Black. Character-based neural machine translation. *CoRR*, abs/1511.04586, 2015. URL `https://arxiv.org/abs/1511.04586`.

Minh-Thang Luong and Christopher D. Manning. Achieving open vocabulary neural machine translation with hybrid word-character models. *CoRR*, abs/1604.00788, 2016. URL `https://arxiv.org/abs/1604.00788`.

Thang Luong, Ilya Sutskever, Quoc V. Le, Oriol Vinyals, and Wojciech Zaremba. Addressing the rare word problem in neural machine translation. *CoRR*, abs/1410.8206, 2014. URL `https://arxiv.org/abs/1410.8206`.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL `http://dx.doi.org/10.3115/1073083.1073135`.

M. Schuster and K.K. Paliwal. Bidirectional recurrent neural networks. *Trans. Sig. Proc.*, 45(11):2673–2681, November 1997. ISSN 1053-587X. doi: 10.1109/78.650093. URL `http://dx.doi.org/10.1109/78.650093`.

Mike Schuster and Kaisuke Nakajima. Japanese and korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152. IEEE, 2012.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *CoRR*, abs/1508.07909, 2015. URL `https://arxiv.org/abs/1508.07909`.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215, 2014. URL `https://arxiv.org/abs/1409.3215`.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016. URL `https://arxiv.org/abs/1609.08144`.

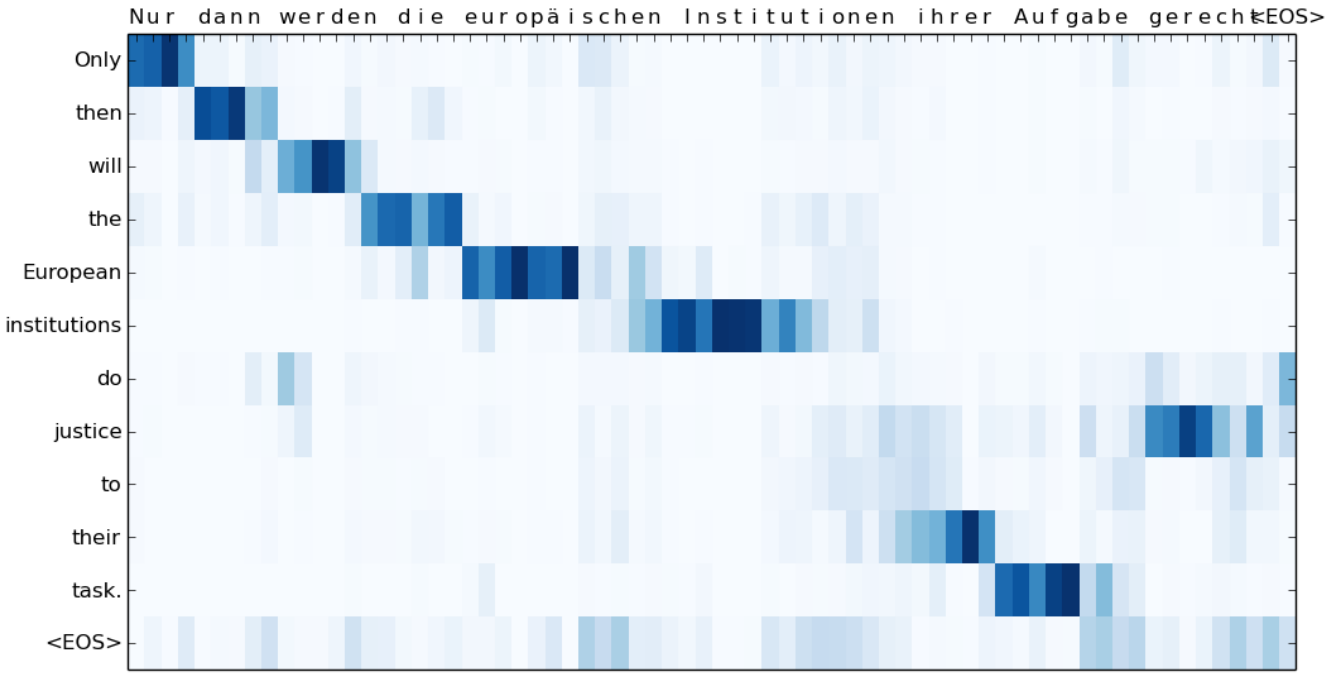

Fig. 7. Attention plot of our `char2word-to-char` model encoding and decoding a sentence from English to German.