Machine Translation System for Vietnamese-English and Chinese-English

Chaitra Hegde
Center for Data Science
New York University
cvh255@nyu.edu

Aakash Kaku
Center for Data Science
New York University
ark576@nyu.edu

Rong Zhao
Center for Data Science
New York University
rz729@nyu.edu

Abstract

We build a neural machine translation system to translate from Vietnamese to English and from Chinese to English. We experiment with various encoder-decoder architectures. The 2-layer Bi-LSTM encoder and 2-layer LSTM decoder is seen to give the best BLEU scores for both Vietnamese-English and Chinese-English translation task.

1 Introduction

The task of machine translation started around 1950s. It has evolved from rule-based, to example-based, then to statistical, and now to neural machine translation (NMT) system.

In NMT, the basic architecture is in the form of an encoder and decoder. The encoder encodes the source sentence, whereas the decoder decodes/produces the target sentence. The recurrent neural networks, like LSTM and GRU models, have shown to perform well on this task. However, a simple architecture of encoding the source sentence into an hidden representation and passing the hidden representation to the decoder to produce the translation seems to give limited and poor performance. This is understandable as the model is expected to encode the entire sentence in a fixed size feature vector from which the decoder has to decode. Therefore, attention mechanism was introduced by (Cho et al., 2014) which gives massive improvements to the performance of NMT models.

Additionally, there have been work done by (Gehring et al., 2017) and (Gehring et al., 2016) and many others to introduce convolutional encoder and decoder instead of recurrent encoder and decoder. Recently, (Vaswani et al., 2017) introduce a transformer architecture which shows that neither convolution nor recurrent neural network is needed. The only thing needed is the attention. Hence, (Vaswani et al., 2017) is able to use fully connected layers and self attention mechanism to achieve the state-of-the-art results.

For this project, we build an end-to-end neural machine translation system to translate much more difficult translation tasks, translating Vietnamese to English and translating Chinese to English. We implement several models ranging from RNN based encoder-decoder to self-attention based models. In this work, we also perform hyper parameter search over a large space to find the optimal model configuration. From the experiments, it is seen that for both of the translation tasks, the LSTM based encoder-decoder performs the best followed by the self attention based model.

2 Data

We separate our Vietnam-English(VE) dataset into approximately $133,317$ training data, $1,268$ validation data and $1,553$ test data and Chinese-English(CE) data set into approximately $212,922$ training data, $1,261$ validation data and $1,397$ test data. The 95-th percentile sentence length in the training data for VE is 48 and for CE is 57. We use the pre-tokenized data provided to us by the instructors. We just split the Vietnamese and English sentences based on spaces between tokens and lowercase all the words. We also filter out the empty rows in both of the datasets. Regarding the Chinese sentences, we have two different ways of tokenization. One way is to split sentences into all possible vocabulary with respect to Chinese (as given by the TAs/Course instructor), and another way is to tokenize into character level. When we construct vocabulary for VE, we choose the words that showed up at least 5 times in the training dataset. Therefore, For Vietnamese to English models, Vietnamese has $13,294$ words in the vocabulary while English has $16,031$ tokens. And for CE, Chinese has $47,133$ words in the vocabulary while English has $28,812$ tokens

3 Model Architecture

3.1 RNN Encoder - Decoder

We use an RNN-encoder(RNN-enc) to encode the source sentence, Chinese or Vietnamese. At each time step, we feed the current token embedding and the previous hidden state to generate the next hidden state. We, therefore, encode a variable length sequence using an RNN. For this project, we have experimented with bi-directional LSTM (Hochreiter and Schmidhuber, 1997) and bi-directional GRU (Cho et al., 2014) to encode the source sentence. A bi-directional RNN makes sense as the hidden state will now be able to capture dependencies from both directions and hence, will be a much richer representation of the source sentence. During the decoding phase, we initialize the hidden state of the decoder RNN with the last hidden state of the encoder RNN. We experiment only with the LSTM decoder. During decoding phase, we feed the token embedding of the true previous word (if teacher forcing) or previous output (if no teacher forcing) and previous hidden representation to generate the next hidden state and apply softmax over the target vocabulary for the next word. The input embeddings of the source sentence and target tokens are passed through a dropout layer with p = 0.1. We also have dropout between 2 layers of RNN if number of layers is greater than or equal to 2.

$\mathrm{h}_t = ENC(x_t, h_{t-1}), t = 1, ..., T$

$g_t = DEC(y_{t-1}, g_{t-1}, h_T)$

3.2 RNN Encoder - Decoder with Attention

We implement the attention mechanism proposed by (Bahdanau et al., 2014). Following from the above architecture of RNN encoder decoder, during the decoder phase, at each time step, the hidden representation generated by the decoder attends to all hidden representation generated by the encoder, which gives us a score between the current decoder hidden representation and one of the encoder hidden representation. We transform those scores using softmax into weights, and generate a weighted sum of all the encoder hidden representations. This weighted sum vector is concatenated with the current decoder hidden representation and passed through a linear layer. We project this concatenation to the same dimension as the hidden representation of the decoder, and apply point-wise non-linear function, tanh, to get a context vector. This context vector is passed through another linear layer (so that it is lifted to target vocabulary size) and logsoftmax layer to get log of softmax probabilities over the entire target vocabulary. Additionally, this context vector is concatenated with the next time step target input embedding which, in turn, is fed to the decoder.

$\mathrm{g}_t = DEC(y_{t-1}, g_{t-1}, c_{t-1})$

$s_i = f_{score}(g_t, h_i), \forall i \in 1, ..., T$

$a_i = \frac{e^{s_i}}{\sum_{j=1}^{T} e^{s_j}}$

$c_t = tanh(W[\sum_{i=1}^{T} a_i h_i, gt])$

3.3 CNN Encoder and LSTM Decoder

We replace the RNN Encoder with a CNN encoder. The architecture for the CNN encoder is the same as (Gehring et al., 2016). Here, we have two convolutional network that encodes the sentence. One convolutional network generates the keys, whereas the second convolutional network generates the values which would be used for the attention mechanism and extracting a context vector. The decoding network is still the same LSTM decoder with attention mechanism (Bahdanau et al., 2014). The query vector is the hidden state of the decoder. The general architecture of each convolutional network is multiple layers of convolutions which are connected using residual connections. As given in the paper by (Gehring et al., 2016), we use 6 layers of convolutions for the convolutional network that produces keys, whereas 3 layers of convolutions for the network that produces values. The kernel size for the convolution is chosen as 3. The context vector is the weighted average of the values where weights are determined by the dot product between keys and the query vector. All the weights are normalized to sum to 1 using softmax.

3.4 Self-Attention Encoder RNN Decoder

We use the self-attention based encoder as described in the paper by (Vaswani et al., 2017). The self-attention based encoder doesn't have recursive components like RNN. In the self-attention based encoder, the source sentence is passed through the fully connected layer to get the keys, values and queries for each token in the source sentence. The final representation of the token w_i is the weighted sum of the value vectors weighted by the attention score which are proportional to the similarity (dot product) between the query vector and the key vectors. Similar to (Vaswani et al., 2017), we also use multi-headed attention (8 headed attention) to obtain a rich feature representation of each token and the source sentence. We also use the positional embeddings to encode the token's order information. We add the positional embedding to the token embedding to obtain the final embedding for a token. The decoder for this network is still the LSTM decoder with the same attention mechanism (Bahdanau et al., 2014) as described in the previous section.

4 Experiments and Results

4.1 Training Methodology

Each model is trained for around 30 epochs and early stopping is used for saving the best model. We observe that most of the models doesn't improve after 20 epochs. Teacher forcing is used during training for 95% of the time, which results in better and faster convergence, and we use the non-teacher forcing for 5% of the time to induce some stability in the models. For the training, the first 10 epochs we use SGD with momentum and nestrov and then switch to Adam for the remaining epochs. For Adam optimizer, initial learning rate is set to $1e-3$ and reduced by a magnitude when the validation loss plateaus or increases (minimum LR = $1e-5$). For SGD, the momentum is set to 0.99, initial LR is set to 0.25 and reduced by a magnitude when the validation loss plateaues or increases (minimum LR = $1e-3$).

4.2 Inference Methodology

Once the model is trained, validation performance is used for model comparison and selection. We use beam search with beam size of 1 (greedy), 3, 5 and 10 to perform the inference. Generally it is seen that using the beam of size 3 or 5 increases BLEU by 1-1.5 points on average. The other thing that we take into account while evaluating is the way we tokenize the validation and test set. In the first method, the original reference sentence is kept as it is and in the second method, all the out-of-vocabulary(OOV) words in the reference sentence are substituted with UNK token before computing the BLEU. Since there are more n-grams predicted right by the model in the second case, the BLEU is, on average, 2% higher than the first method where we keep the sentence intact.

4.3 Results

4.3.1 Best Vi-En Model

The best Vi-En model is found to be the bi-LSTM encoder with 2 layers and hidden dimension of 512 and LSTM decoder with 2 layers and hidden dimension of 1024,and inference with beam size of 10. This model is tested on the test dataset to get the final generalization performance. We get 24.03 BLEU score (without replacing OOV tokens with UNK in the reference sentence) and of 25.1 (with replacing OOV tokens with UNK in the reference sentence). In Figure 1b, we are visualizing the attention weights for one example sentence. As it can be seen in the heatmap, the number, "150", in the source sentence is activated when the model predicts "150" for the target sentence. Similarly, the punctuation "," in the source gets activated when it predicts comma in the target sentence.

4.3.2 Best Zh-En Model

The best Zh-En model is found to be the bi-LSTM encoder with 2 layers and hidden dimension of 512 and LSTM decoder with 2 layers and hidden dimension of 1024, and inference with beam size of 10. This model is tested on the test data et to get the final generalization performance. We get 15.90 BLEU score (without replacing OOV tokens with UNK in the reference sentence) and 16.14 (with replacing OOV tokens with UNK in the reference sentence). In Figure 1a, we are visualizing the attention weights for one example sentence.

5 Analysis

5.1 Trend in the performance of recurrent encoder based models

As it can be seen from the performance of the recurrent encoder based models in tables 1, 2, 3 and 4, adding attention gives massive improvement in the BLEU scores (on average, improvement is of 10 BLEU). Additionally, higher number of layers (2) in the encoder and decoder, with high number of hidden dimension (512) seems to perform better than the models with less complexity. It suggests that the task of translating Chinese to English or Vietnamese to English is a complex task and hence requires complex models to learn richer representations.

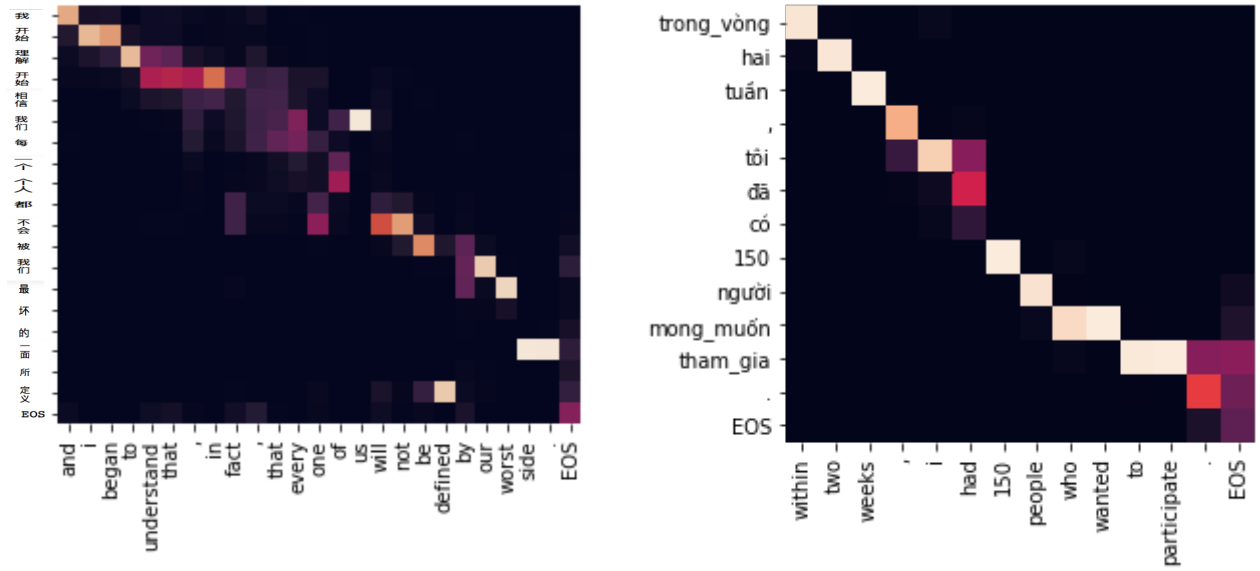
5.2 Poor Performance of the Non-Recurrent models

The convolutional and the self-attention based models are not seen to have performance as good as the recurrent network with attention. There are two possible reasons for it: the hyper-parameter space is quite large for these models and hence, given the time constraint, we were not able to perform exhaustive search over the hyper-parameter space. Additionally, some of the non-recurrent model implementations have some engineering hacks which we might have missed and hence, this could result in poor model performance.

5.3 Error Analysis

In general, out-of-vocabulary words in source sentence is problematic. Because the word is not learned by the model, decoder fails to predict it, and decoder misses out some information about source sentence. Two examples are shown in Fig.2

The best Zh-EN translation model is giving 17.19 BLEU score on validation set. After analyzing the re-



(a) Zh-EN

(b) Vi-EN

Figure 1: Attention weight visualization

```
Encoder input: 可口 可口可乐 可乐 的 主席 最近 UKN 在 多个 发展 发展中国家 中国 国家 对 数百 UKN 百台 弹弓 进行 大规 大规模
规模 测试
True Sentence: the chairman of coca-cola has just agreed to do a major test of hundreds of units of this in the
developing world .
Pred Sentence: the president of coke recently has a massive test for hundreds of UKN countries .
```

(a)

```
Encoder input: 所有 拿破仑 的士 士兵 使用 UKN 器用 用餐 拿破仑 自己 则 使用 金器
True Sentence: all of napoleon &apos;s troops were fed with silver utensils , napoleon himself with gold utensi
ls .
Pred Sentence: all of the UKN soldiers used UKN and UKN himself with gold gold .
```

(b)

Figure 2: Problem with UNK

Encoder Rnn Type	**Encoder Decoder RNN Layers**	**Encoder Hidden Size**	**Deocder Hidden Size**	**Val BLEU Without UNK**	**Val BLEU With UNK**
Bi-LSTM	1	256	512	7.65(with beam-5)	8.7(with beam-5)
Bi-LSTM	1	512	1024	8.79 (with beam-3)	9.136 (with beam-3)
Bi-LSTM	2	256	512	8.14 (with beam-3)	9.19(with beam-10)
Bi-LSTM	**2**	**512**	**1024**	**11.03 (with beam-5)**	**11.55 (with beam-5)**
Bi-GRU	1	256	512	5.5 (with beam-3)	5.3 (with beam-3)
Bi-GRU	1	512	1024	6.61 (with beam-3)	7.06 (with beam-3)
Bi-GRU	2	256	512	8.11 (with beam-3)	8.57 (with beam-3)
Bi-GRU	2	512	1024	9.28 (with beam-10)	8.84 (with beam-10)

Table 1: Performance for the RNN Encoder Decoder (w/o attention) on Vi-EN translation

sults of the model, it's evident that the model is performing well on short sentences and it's failing to get good translation for longer ones. Especially, when the model is translating a longer sentence, initially it does a good job, but, if one out-of-vocabulary word shows up, the rest of the translation becomes poor. The highly probable reason for this kind of behavior might be due to high teacher-forcing (95%). Also, since we have one native Mandarin speaker in our project group, we were able to analyze a few examples in close. There are multiple examples where the reference sentence is wrong but the given translation is right. Because of occurrence of number of such examples, it's possible that the actual BLEU for our model is slightly higher than the reported one.

```
-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
Encoder input: 每 一年 我们 都 要求 求学 学生 创立 一件 公司 或 制作 一种 产品 或者 提供 一项 服务 能 在 十年 年内 对 十亿 人 的 生
活 活产 产生 积极 而 实际 的 影响
True Sentence: and every year we ask them to start a company or a product or a service that can affect positively the
lives of a billion people within a decade .
Pred Sentence: every year , we &apos;ve asked students to start a company or create a product or provide a service fo
r a billion people in the next 10 years .
```

(a) Though its a big input, translation is good enough

```
但 这 记忆 记忆力 是 有 选择 选择性 的
True Sentence: he has a selective one , though .
Pred Sentence: but this memory is selective .
```

(b) The source sentence speaks of ”memory” and the ground truth doesn’t but our model got it correct

Figure 3: Good translation by Zh-EN model

```
-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
Source Sentence: tôi lúc_nào cũng tự_hỏi là tại_sao họ lại có điện còn chúng_tôi thì không .
True Sentence: i always wondered why they had lights but we didn &apos;t .
Pred Sentence: i always wondered why they had electricity , and we didn &apos;t .
-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
```

(a) Good Translation

```
-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
Source Sentence: hàng năm , có vô_số người bắc triều_tiên bị bắt ở trung_quốc và bị trả về nước , nơi mà họ bị tra_tấn ,
bị giam_cầm hoặc bị UKN công_khai .
True Sentence: every year , countless north koreans are caught in china and repatriated to north korea , where they can
be tortured , imprisoned or publicly executed .
Pred Sentence: for years , there were plenty of south korea that were caught in china and was paid in the water , where
they were tortured , had UKN or UKN .
-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
```

(b) Bad Translation

Figure 4: Sample translations for Vietnamese to English

Encoder Rnn Type	**Encoder Decoder RNN Layers**	**Encoder Hidden Size**	**Deocder Hidden Size**	**Val BLEU Without UNK**	**Val BLEU With UNK**
Bi-LSTM	1	256	512	24.01(with beam-10)	25.07 (with beam-10)
Bi-LSTM	1	512	1024	24.72(with beam-10)	26.1(with beam-10)
Bi-LSTM	2	256	512	24.9 (with beam-5)	27.86 (with beam-10)
Bi-LSTM	**2**	**512**	**1024**	**26.19 (with beam-10)**	**27.62 (with beam-10)**
Bi-GRU	1	256	512	22.89 (with beam-5)	24.57 (with beam-10)
Bi-GRU	1	512	1024	24.01 (with beam-10)	25.7 (with beam-10)
Bi-GRU	2	256	512	24.78 (with beam-10)	26.12 (with beam-5)

Table 2: Performance for the RNN Encoder Decoder (w/ attention) Vi-EN translation

Encoder Rnn Type	**Encoder Decoder RNN Layers**	**Encoder Hidden Size**	**Deocder Hidden Size**	**Val BLEU Without UNK**	**Val BLEU With UNK**
Bi-LSTM	1	256	512	5.66(with beam-3)	6.01(with beam-3)
Bi-LSTM	1	512	1024	6.792 (with beam-3)	7.136 (with beam-3)
Bi-LSTM	2	256	512	4.6(with beam=5)	4.9(with beam-3)
Bi-LSTM	**2**	**512**	**1024**	**7.45 (with beam-5)**	**7.59 (with beam-5)**
Bi-GRU	1	256	512	4.12 (with beam-3)	4.3 (with beam-3)
Bi-GRU	1	512	1024	4.93 (with beam-3)	5.28 (with beam-3)
Bi-GRU	2	256	512	4.56 (with beam-3)	5.03 (with beam-3)
Bi-GRU	2	512	1024	5.11 (with beam-3)	5.51 (with beam-3)

Table 3: Performance for the RNN Encoder Decoder (w/o attention) on Zh-EN translation

6 Conclusion

Attention works! Attention is helping the model to weight the source sentence’s hidden representa-

Encoder Rnn Type	Encoder Decoder RNN Layers	Encoder Hidden Size	Deocder Hidden Size	Val BLEU Without UNK	Val BLEU With UNK
Bi-LSTM	1	256	512	14.2 (with beam-5)	15.7 (with beam-10)
Bi-LSTM	1	512	1024	15.7 (with beam-10)	16.4 (with beam-10)
Bi-LSTM	2	256	512	14.6 (with beam-5)	15.02 (with beam-10)
Bi-LSTM	**2**	**512**	**1024**	**16.82 (with beam-10)**	**17.19 (with beam-10)**
Bi-LSTM at **char level**	2	512	1024	11.77 (with beam-3)	12.7 (with beam-5)
Bi-GRU	1	256	512	12.9 (with beam-5)	14.02 (with beam-10)
Bi-GRU	1	512	1024	14.9 (with beam-10)	15.37 (with beam-10)
Bi-GRU	2	256	512	15.31 (with beam-10)	14.55 (with beam-5)
Bi-GRU	2	512	1024	15.48 (with beam-5)	16.21 (with beam-5)

Table 4: Performance for the RNN Encoder Decoder (w/ attention) on Zh-EN translation

Encoder Type and Language	Encoder Hidden Size	Deocder Hidden Size	Val BLEU Without UNK	Val BLEU With UNK
CNN ; Vi-EN	512	1024	5.7 (with beam-1)	7.5 (with beam-1)
Self attention ; Vi-EN	256	512	12.01(with beam-3)	12.76 (with beam-3)
Self attention ; Vi-EN	512	1024	15.1 (with beam-1)	16.14 (with beam-1)
Self attention ; Zh-EN	512	1024	7.32 (with beam-1)	7.32 (with beam-1)

Table 5: Non RNN encoders on Vi-EN and Zh-EN

tion dynamically and hence leading to the better performance as compared to the non-attention models. Our experiments showed that recurrent encoders performed better than the non-recurrent encoders.

7 Future Work

Due to lack of time, we were not able to perform in-depth hyper-parameter tuning for self attention encoder and convolution encoder model. Hence, we would like to further improve the performance of those models by experimenting on a relatively larger hyper-parameter space. Also, for all the models, the decoder was fixed to LSTM. We would like to experiment non-recurrent based decoders like one in the Transformer. Further, we would also like to build bilingual translation system as instructed in optional part. Additionally, for Chinese to English translation task, the performance is poor. Hence, we plan to use better tokenization schemes to segment the source and target sentence and improve the performance on this task.

8 Miscellaneous

8.1 Contribution

All the members of the team have contributed equally.

Chaitra Hedge: Built the baseline (RNN Enc-Dec w/o attention) model, hyper parameter tunig for the baseline model and the utility functions for loading and cleaning the data.

Rong Zhao: Built the attention based recurrent models, performed hyper-parameter tuning for attention based recurrent models and cleaned and maintained the code base.

Aakash Kaku: Built the non-recurrent encoder models (Conv and self attention), performed hyper-parameter tuning for attention based recurrent models and non-recurrent models and implemented beam search.

Everybody contributed equally for writing the project report.

8.2 Github Repository

All the code pertaining to this project is available in the github repository at `https://github.com/HegdeChaitra/machine_translation`

References

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

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Jonas Gehring, Michael Auli, David Grangier, and Yann N. Dauphin. 2016. A convolutional encoder model for neural machine translation. *CoRR*, abs/1611.02344.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. *CoRR*, abs/1705.03122.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

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