# Project - 1: Machine Translation

Anonymous ACL-IJCNLP submission

## Abstract

In this assignment, I have implemented, evaluated and analysed sequence to sequence models both qualitatively and quantitatively. I also present and describe implementations of copy mechanism, beam search and nucleus sampling. As a part of the optional tasks, I have explored and implemented the task mentioned: seq2seq model for translating French to English, built a convolutional seq2seq learning model, built a transformer for *Attention is all you need*, and designed and explored a new top-p:top-k model as an advancement to beam search.

## 1 Task 1 Building a sequence to sequence model

Using the initial code provided, I have implemented a seq2seq model for translating German text to English using a bidirectional GRU units in the encoder and single dimensional GRU units in the decoder. I also used an attention mechanism to enable the decoder to look into the source sentence instead of forcing the encoder to compress the complete information in the context vector. I trained the model for 10 epochs. At the end of 10 epochs, the training loss was **1.475** and validation loss was **3.246**. The loss on the test set was **3.176** . I have also plotted the train and validation loss curves in Figure 1. The BLEU score for the experiment is **29.24**.

### 1.1 Examples of Translation

**Example 1:** *ein boston terrier läuft über saftig-grünes gras vor einem weißen zaun .*
**Expected Translation:** *a boston terrier is running on lush green grass in front of a white fence .*
**Model's Output:** *a small dog runs through grass grass in front of a white fence*
**Attention Visualization:**

**Analysis:** While the model is able to identify the that *boston terrier* is a dog, it looses the


Figure 1: Training and Validation Loss

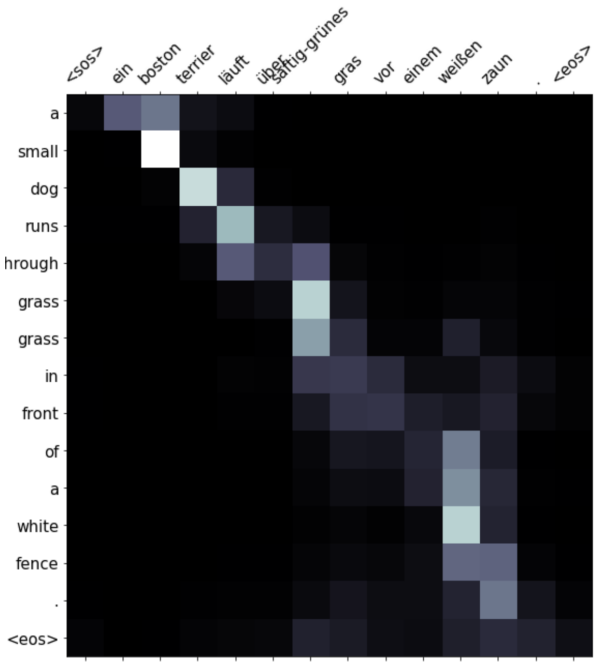


Figure 2: Attention Matrix

information that it is a specific breed of dog in the translation. Moreover, from the attention matrix we can see that it identifies *boston* as *small*, which is incorrect. Also, the model suffers with a habit of repeating the characters as *grass* is repeated multiple times. This behaviour is seen in many instances.

**Example 2:** *eine mutter und ihr kleiner sohn genießen einen schönen tag im freien*
**Expected Translation:** *a mother and her young song enjoying a beautiful day outside .*
**Model's Output:** *a mother and her son enjoying enjoying a beautiful day .*
**Attention Visualization:**

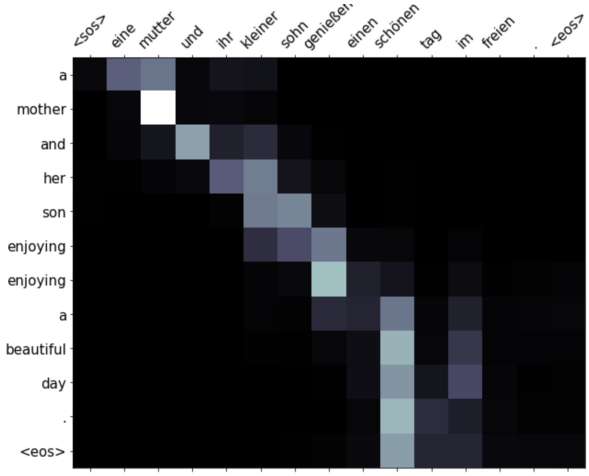


Figure 3: Attention Matrix

**Analysis:** Notice that the expected translation is incorrect, while the model gives the correct transalation for the word *son* (expected output is *song*). Also, it is not able to figure out the they are outside. Enjoying is also repeated in the translation. To debug this, I analyzed the training data to see why the model is predicting repeated words. I found that the training data has several instances of repeating words and therefore the model tends to learn this behaviour.

## 2 Task 2

Beam search (Sutskever et al., 2014) finds the output sequence having a maximum likelihood. Instead of considering only 1 outcome at the end of each decoding step, it considers top-k tokens most probable tokens. At timestep $t$ it considers next $k$ most probable words, where k is the beam width. In the implementation, I stop the beam search whenever I get an token. Mathematically, beam search can be described as,

$$P(y|x) = P(y_1|x) * P(y_2|y_1, x) * P(y_3|y_2, y_1, x) \dots P(y_T|y_{T-1}, \dots y_1, x)$$
$$= \prod_{t=1}^{T} P(y_t|y_{t-1}, \dots y_1, x) = \sum_{t=1}^{T} \log P(y_t|y_{t-1}, \dots y_1, x)$$

The higher BLEU scores in Figure 4 shows that beam search is more effective than the greedy search. It can be also analyzed from Figure 5 that the output's likelihood increases as we increase the beam width and then it stabilizes. Beam search is better than greedy search because it considers multiple options in the tree to optimize on the overall likelihood.

As the beam search maximizes likelihood, it has a monotonic nature, but that does not translate to high BLEU score for higher beam widths.

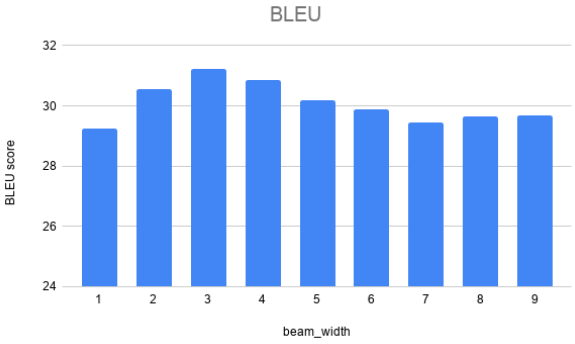


Figure 4: BLEU scores for Beam search for various widths

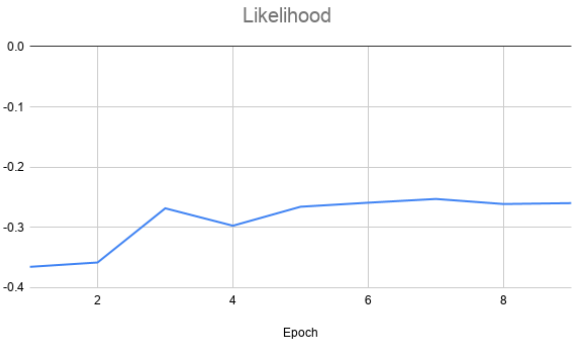


Figure 5: Likelihood scores for Beam search for various widths

### 2.1 Examples

In this section, I present various examples to demonstrate why beam search outperforms the greedy search.
**Example 1:** *fünf leute in winterjacken und mit*

*helmen stehen im schnee mit schneemobilen im hintergrund .*
**Expected Translation:** *five people wearing winter jackets and helmets stand in the snow , with snowmobiles in the background .*
**Greedy search: likelihood: -0.3585:** *five people in in in in in in and and helmets are standing in the snow with the background .*
**Beam search (beam width = 2):** *five people in life jackets and helmets are standing in the snow with the background .*
**Beam search (beam width = 5):** *five people are wearing life jackets and hard hats are standing in the snow in the background background .*
**Beam search (beam width = 9): likelihood: -0.2304:** *five people are in life jackets and hard hats are standing in the snow in the the background .*
**Analysis:** Notice that the repetition of words in greedy search's output, but beam search do not suffer from this problem. Since, beam search has the luxury to analyze more branches of the tree, it is able to come up with a solution with a higher likelihood ( -0.2304 vs -0.3585). **Note**: Although in my analysis in later sections, I have observed that beam search is also susceptible to repetition, but the problem is larger in greedy search.

It is also observed that a lower beam width results in lesser quality of translations. But as we increase the beam width the number of instruction required to converge to a solution increases. As a result CPU and memory requirements also spike up.

## 3 Task 3: Nucleus Sampling

It is a stochastic decoding method (Holtzman et al., 2019) where we determine the set of tokens in the decoding step via the shape of probability distribution. First we prune the number of candidates based on a probability threshold. In the next step, we sample the token from the re-scaled probability distributions. Mathematically it can be expressed as:

$$\sum_{x \in V^{(p)}} P(x|x_{1:i-1}) \geq p.$$

$$P'(x|x_{1:i-1}) = \begin{cases} P(x|x_{1:i-1})/p' & \text{if } x \in V^{(p)} \\ 0 & \text{otherwise.} \end{cases}$$

It is called nucleus sampling because there are a handful of words which take up majority of probability mass.
**Experimentation setup:** I iterated over probability threshold values from 0.1 to 1. and temperature values from 0.1 to 0.9. BLEU and likelihood are plotted in Figure 6 and Figure 7 respectively.

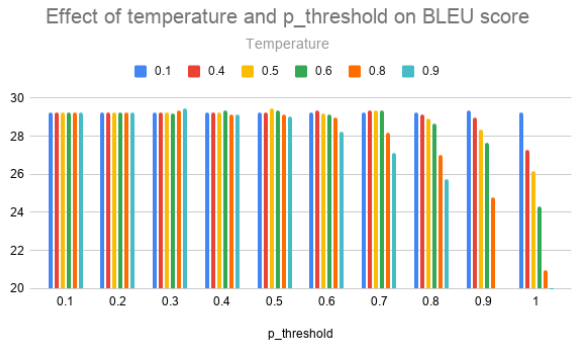


Figure 6

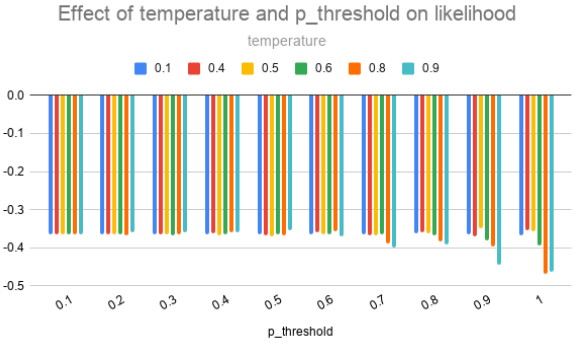


Figure 7

### 3.1 Effect of probability threshold

If the probability threshold is very less then the output of nucleus sampling will be very similar to greedy search. It can be inferred via the BLEU score of greedy search and nucleus sampling with threshold = 0.1. As the probability threshold increases BLEU score goes down. This is because the algorithm starts to consider a lot of unrelated words as well as synonyms as candidates which might not be relevant to the input sentence. For instance, the output of nucleus sampling for Example in section 2 with p=0.9 is *five people wearing in winter attire and life kneeling in the snow - personnel in the background*

### 3.2 Effect of temperature

If the temperature is less, nucleus sampling behaves like argmax(greedy search) which is evident from similar BLEU and likelihood scores for temperature=0.1 and it behaves more like multinomial function on higher temperature values. Temperature's effect can be characterized by this equation: We can see that a lower temperature $t$ makes it more

$$p(x = V_l|x_{1:i-1}) = \frac{\exp(u_l/t)}{\sum_{l'} \exp(u'_l/t)}.$$

likely to get top choices and leads to a narrower distribution, while a higher temperature results in a flatter, smoother distribution.

### 3.3 Comparison with Beam search and greed search

There are two major differences between the output of beam search, greedy search and nucleus sampling. Firstly, both greedy and beam search suffer from a problem of repetition. Secondly, nucleus search confuses between various words which might be similar in the word embedding space. This can be demonstrated via few translations from the test set. **Example 1:** *asiatische frau trägt einen sonnenhut beim fahrradfahren .*
**Expected Translation:** *asian woman wearing a sunhat while riding a bike .*
**Two of nucleus search translations, p=0.9** *asian woman wearing a hat on beach . ; asian woman carries a pier , pedal .*
**Greedy Search** *asian woman wearing a a a train .*
**Beam search (beam width = 9):** *asian woman is wearing a top of a train*

The greedy search is stuck in repetition and beam search as emitted a sentence which has no relevance with bike. Whereas, nucleus sampling emits words such as pedal which are related to bike and hat which is similar to sunhat. If we increase the threshold of nucleus search to 0.99, we get sentences such as *asian woman wearing a floor is running across the bench*. This demonstrates that the output of nucleus sampling if subjected to high parameter ranges, emits random sentences which are totally unrelated to the input sentence.

Hence, it is necessary to keep the parameters of nucleus search within a reasonable range to get relevant outputs.

## 4 Task 3: Copy Mechanism

I have also implemented copy mechanism in the given code stub of seq2seq model. I merged the dictionary of source and target languages to enable predictions in the source language. I implemented the copy mechanism as proposed in the paper (Gu et al., 2016). I trained the model for 10 epochs and achieved a training loss of **1.632** and validation loss of **3.300**. The loss on the test set was **3.198**. BLEU score on the test dataset was observed to be 30.01. I have plotted the training and validation loss in Figure 8

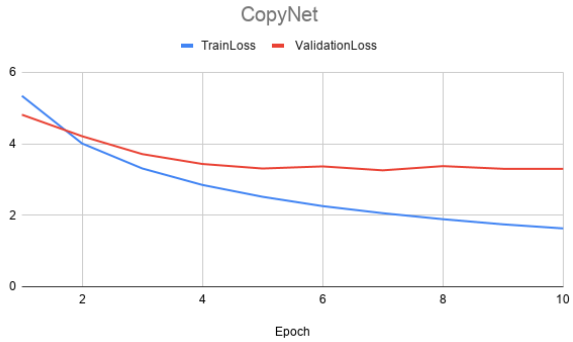


Figure 8

The copy mechanism corrects the problems of identifying phrases in the input sentence. For instance,
**Example 1:** *ein boston terrier läuft über saftig-grünes gras vor einem weißen zaun .*
**Expected Translation:** *a boston terrier is running on lush green grass in front of a white fence .*
**CopyNet:** *a boston terrier runs across grass grass in front of a white fence .*
**Seq2seq:** *a small dog runs through grass grass in front of a white fence*

As we can see that the copy mechanism is effectively able to transfer **boston terrier** from the input sentence. The RNN seq2seq model is able to identify that it is a dog but misses out on the exact breed of dog. Both these model suffer from problem of repetition.

Copy mechanism also introduces new errors in the test model. I found that the sentences in the translation were sometimes incorrectly ending. Here is one instance:
**Example 2:** *eine frau spielt ein lied auf ihrer geige .*
**Expected Translation:** *a female playing a song*

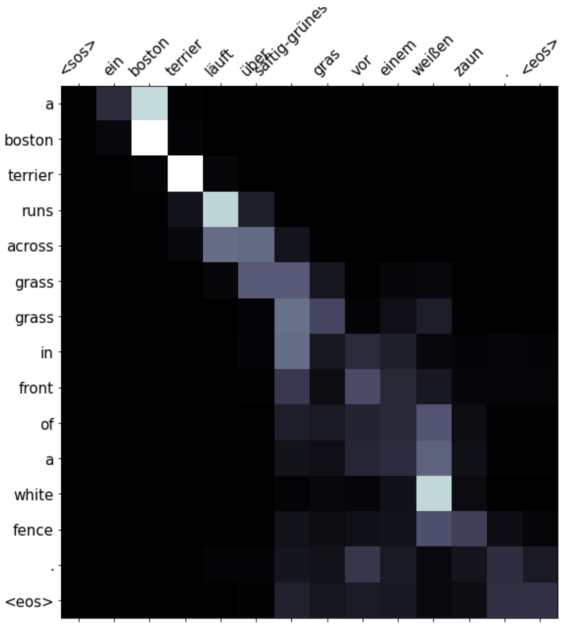


Figure 9

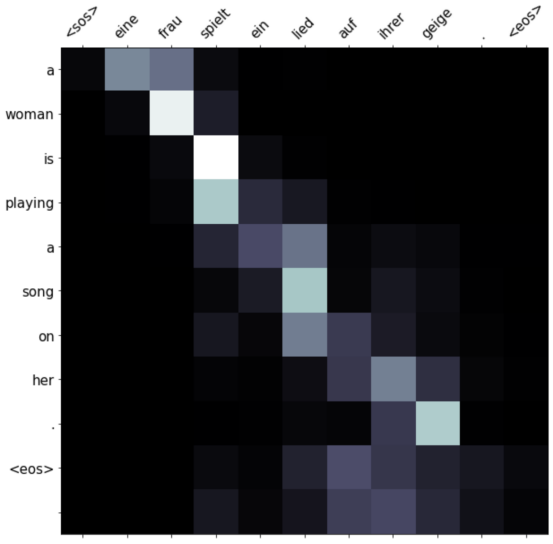


Figure 10

*on her violin .*
**CopyNet:** *a woman is playing a song on her .*
**Seq2seq:** *a woman is playing song song on her violin*

In the attention matrices of both the examples we see that the attention at the end is not that concentrated. In Figure 2 where attention matrix of seq2seq matrix has been plotted, we observe that at the end of the sentence the attention weights distributes, but this problem is amplified in Copynet.

### 4.1 Impact on Fluency:

Upon manually analysing all 1000 data points in the test dataset, I would conclude that the overall fluency has increased in some cases and decreased in other. Like in Example 1, as we have more information about the dog and the structure of the sentence is correct, I would say that the overall fluency has improved a bit. But there are cases like Example 2 when the sentence is grammatically incorrect, therefore the fluency has taken a hit.

One peculiar thing that I have observed is the overall probabilities of the target token increases as compared the the seq2seq model, when the word is in English as well as in German. For instance:

**Example 3:** *eine person mit kapuze steht vor einem heruntergekommenen gebäude .*
**Expected Translation:** *a person in a hood is standing in front of a run down building .*
**CopyNet:** *a person wearing a hood standing standing in front of a building .*
**Seq2seq:** *a person in a standing in front of a building building .*

## 5 Optional tasks

### 5.1 Convolutional seq2seq and transformer seq2seq

Using (Seq) as a reference, I build a convolutional seq2seq model and transformer seq2seq model. Metrics of the model have been mentioned in the Figure 11. Their training and validation loss curves also have been plotted in Figure 12 and Figure 13.

| Model | Transformer | Convolutional | CopyNet | RNN |
|---|---|---|---|---|
| Train Loss | 0.961 | 1.819 | 1.632 | 1.475 |
| Validation loss | 1.636 | 1.754 | 3.3 | 3.246 |
| Test loss | 1.659 | 1.822 | 3.198 | 3.176 |
| BLEU score | 34.83 | 30.81 | 30.01 | 29.24 |

Figure 11: Metrics of various algorithms

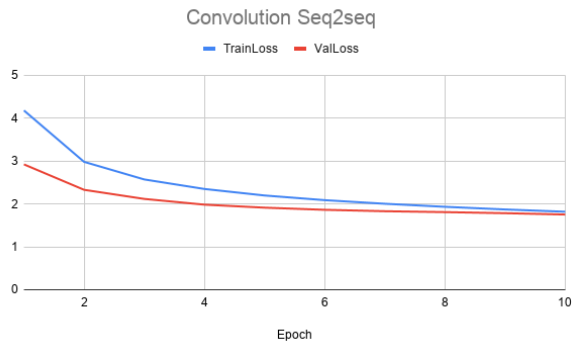


Figure 12: Loss curve for convolutional seq2seq

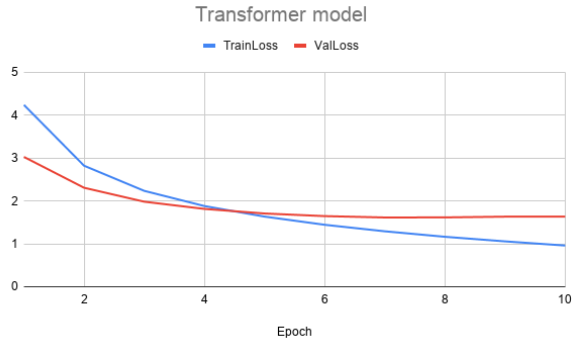


Figure 13: Loss curve for Transformer seq2seq

As compared to the copynet and rnn seq2seq model, the translations of these two were far superior. The problem that I had mentioned regarding the attention of RNN model diminishing in the end, that is not observed in these two models.

### 5.2 Train the model on another machine translation dataset for a language pair of your interest

I have trained model for French to English (Fre). The same model architecture performs decently well on this dataset as well. The dataset has 135842 pairs of translations, out of which 75000 were used for training, 40000 for validation and rest as test set. The model was trained for 10 epochs and the final training loss was 0.5917. Few examples have been listed down below:

**Example 1:** *nous sommes confrontes a de nombreuses difficultes*
**Expected Translation:** *we are faced with many difficulties*
**Seq2seq:** *we are faced with many difficulties*

**Example 2:** *il a accepte de faire le travail .*
**Expected Translation:** *he s agreed to do the job .*
**Seq2seq:** *ahe is eager to do the job . .*

### 5.3 Choose your own adventure

In the section, I have proposed a different sampling technique based on nucleus sampling and beam search. This aims to select the candidate words for beam search efficiently at each time step. The process of decoding is similar to beam search with difference lying in selecting a new candidate word at each iteration. Instead of selecting fixed number of candidates, I select top tokens with cumulative probability of p_threshold. This process will help us eliminate less probable candidates in beam search, hence cutting down our inference time. Code for the method is listed below:

```
            sorted_probs, sorted_indices = torch.sort(predictedNormProb, descending=True)
            cumulative_probs = torch.cumsum(sorted_probs, dim=1)
            sorted_indices_to_remove = cumulative_probs < p_threshold

            if len(sorted_indices[sorted_indices_to_remove]) == 0:
                pred_tokens = [sorted_indices[0][0]]
                scores = [sorted_probs[0][0]]
            else:
                pred_tokens = sorted_indices[0][:len(sorted_probs[sorted_indices_to_remove])]
                scores = log_probs[0][:len(sorted_probs[sorted_indices_to_remove])]

            logSoftmax = nn.LogSoftmax()(log_probs)
            for n in range(len(pred_tokens)):
                newNode = BeamSearchNode(hiddenstate=hidden,
                                         previousNode=node,
                                         logProb=node.logProb + scores[n].item(),
                                         length=node.length + 1,
                                         totalSentence=node.totalSentence + [pred_tokens[n].item()],
                                         encoder_outputs=encoder_outputs,
                                         logsoftmax=logSoftmax[0][pred_tokens[n]]
                                        )

                trg_indexes_queue.append(newNode)

        # keep only top k
        trg_indexes_queue = sorted(trg_indexes_queue, key=lambda tup:tup.eval())

        trg_indexes_queue = trg_indexes_queue[-beam_width:]
```

Figure 14: Top p top k sampling method

BLEU score of **29.27** was achieved on the dataset. This did not alter the fluency of the translated sentence, and the computation time has significantly gone down. This can be seen in the below examples of the generated sentence using top_p:top_k model next to the samples from beam search and nucleus sampling.

**Example 1:** *ein mann mit freiem oberkörper und shorts steht auf ein paar steinen und angelt*
**Expected Translation:** *a shirtless man in shorts is fishing while standing on some rocks .*
**top_p:top_k :** *a shirtless man wearing shorts and on green pants , standing in fishing .*
**nucleus sampling:** *a shirtless man with shorts is standing on at the poles .*
**beam search:** *a shirtless man wearing shorts is standing on a bunch of rocks .*

## 6 Conclusion

I have presented implementations of seq2seq model, copy mechanism, beam search and nucleus

sampling. I have analysed and identified the trade-offs between sentence quality, time, and performance. Explored models for different language translations, Convolutional ses2seq model, transformer seq2seq and proposed a new top_p:top_k as an advancement. Future work could include improving the efficiency the of the topp-topk model. This will allow the model to produce better quality translated sentence with significantly less computation time.

## References

French - English Dataset. https://download.pytorch.org/tutorial/data.zip.

PyTorch Seq2Seq. https://github.com/bentrevett/pytorch-seq2seq/.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. *CoRR*, abs/1603.06393.

Ari Holtzman, Jan Buys, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. *CoRR*, abs/1904.09751.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215.