ST449 Final Project: Recurrent Neural Networks vs. Transformers in Neural Machine Translation

41294
Department of Statistics
The London School of Economics and Political Science

May 14, 2020

Contents

1 Introduction **1**
1.1 The Data 1

2 Methodology **1**
2.1 Preprocessing Data 2
2.2 The RNN Model 3
2.3 The Transformer 4

3 Results **6**
3.1 RNN 6
3.1.1 Dutch → German 6
3.1.2 German → Dutch 6
3.1.3 English → Brazilian-Portuguese 7
3.2 Transformer 7

4 Conclusion **8**
4.1 Further Steps 8

1 Introduction

In the field of Natural Language Processing (NLP), machine translation has emerged as a subfield of great research and promise. Generally speaking, machine translation is the ability of a computer to read a sentence in one language and translate it to another [5]. In recent years, this process has been improved through approaching the problem using neural networks, in turn creating Neural Machine Translation (NMT). While all NMT systems are made up of an encoder and decoder network as well as an attention mechanism, they differ in the precise architecture and mechanism used to build the translation model as well as the hyperparameters set to train it [2].

For this paper, I directly compared the translation accuracy between a Recurrent Neural Network (RNN) with Long Short Term Memory (LSTM) attention and a Transformer NMT with self-attention. Each model was built according to the specifications used by Cettolo and his partners in their submission for the ISWLT 2017 conference [2]. Both models were applied on three different translation pairs, Dutch $\rightarrow$ German, German $\rightarrow$ Dutch, and English $\rightarrow$ Brazilian-Portuguese, though the Transformer was only used on the latter. Each model had its testing accuracy evaluated through a BLEU score between 0 and 100. In this scoring system, the predicted target language output is compared to the true output for the testing set, and a more accurate model will score closer to 100 while poor models will produce a result closer to 0.

As only Dutch $\rightarrow$ German was originally evaluated in [2], the other language pair BLEU scores were simply compared to one another where appropriate or examined at singular level given the clear nature of their results.

1.1 The Data

The data used comes from the Web Inventory of Transcribed and Translated Talks, a publicly available inventory of translated TED talks across various different languages [3]. As the talks range across various topics, the data is not limited to one domain and as such can act as a reliable corpus for Machine Translation. Additionally, the training, validation and testing datasets have been pre-specified by the International Conference on Spoken Language Translation (IWSLT) for each language pair observed, in order to fully compare with the baseline models they generated in 2014 and 2017 for English $\rightarrow$ Brazilian-Portuguese, Dutch $\rightarrow$ German, and German $\rightarrow$ Dutch [4].

For German $\rightarrow$ Dutch and Dutch $\rightarrow$ German, the training data consisted of only 1779 lines, while the validation was 1001 and the test was 1174. It should be noted that the training data is quite small compared to the traditional size used in NMT, which can range from 100,000 to millions of lines [6]. As such, the model for both of these translations may not generate a high translation accuracy or BLEU score due to this limited training corpus. However, the English $\rightarrow$ Brazilian-Portuguese data had 177,275 lines in the training set, 1556 in the validation and 1262 in the testing set, which falls within the more typical range of training data sizes and therefore may not result in that same problem.

2 Methodology

In order to implement the RNN and Transformer models in a reasonable timeframe, they were trained using the Deep Learning Virtual Machine from Google Cloud Platform. The VM instance was made up of 2 virtual CPUs with 13 GB of memory and one NVIDIA Tesla K80 GPU. Using more GPUs would have been ideal in speeding up training and exactly replicating the results in [2] and [8], but this was not possible given the quotas on my personal Google Cloud account. Additionally, as OpenNMT is built on PyTorch, this was specified as the primary machine learning framework for the instance.

After setting up the VM instance, OpenNMT-py was downloaded onto it and the data used for training,

validation and testing was moved into the data folder in OpenNMT. Used by the original paper [2], OpenNMT is a toolkit for Neural Machine Translation that provides prebuilt model architectures that can be easily adapted to a particular problem, as well as a clear training and testing function for the unique models [6].

Though OpenNMT-py was built on PyTorch, and as such uses that framework to build and train the models on the VM, I wrote up a TensorFlow version of both the RNN and Transformer in order to illustrate how each was built. This can be found in the Jupyter Notebook file "TensorFlow Implementation" found in the project repository. It also includes a OpenNMT-tf version of the training function used on both models.

2.1 Preprocessing Data

Prior to any training of the model or even generation of a vocabulary base, the text must first be tokenized. This can be done independently, according to one's own function or using the built-in tokenizer file with OpenNMT. For this project, I used both methods and determined that they generated essentially the same final tokenization of the text documents save for one negligible difference (with respect to the exact token used for "). My independent tokenizer file is included as a supplemental Jupyter Notebook file, "Preprocessing Data", in the project repository.

Once the data has been tokenized, the vocabularies for each word in the source and target languages in the translation must be computed. This can be done using the preprocess command in OpenNMT, where the tokenized training and validation files are specified for both the source and target [6].

```
(base) Carolines-MacBook-Air:OpenNMT-py carolinemcdonald$ onmt_preprocess -train_src data/final_data/german_train.txt -train_tgt data/final_data/
dutch_train.txt -valid_src data/final_data/german_val.txt -valid_tgt data/final_data/dutch_val.txt -save_data data/final_data/processed_data
[2020-04-16 21:44:28,439 INFO] Extracting features...
[2020-04-16 21:44:28,440 INFO]  * number of source features: 0.
[2020-04-16 21:44:28,440 INFO]  * number of target features: 0.
[2020-04-16 21:44:28,440 INFO] Building `Fields` object...
[2020-04-16 21:44:28,441 INFO] Building & saving training data...
[2020-04-16 21:44:28,461 INFO] Building shard 0.
[2020-04-16 21:44:28,525 INFO]  * saving 0th train data shard to data/final_data/processed_data.train.0.pt.
[2020-04-16 21:44:28,779 INFO]  * tgt vocab size: 4249.
[2020-04-16 21:44:28,789 INFO]  * src vocab size: 4841.
[2020-04-16 21:44:28,823 INFO] Building & saving validation data...
[2020-04-16 21:44:29,378 INFO] Building shard 0.
[2020-04-16 21:44:29,401 INFO]  * saving 0th valid data shard to data/final_data/processed_data.valid.0.pt.
(base) Carolines-MacBook-Air:OpenNMT-py carolinemcdonald$
```

Figure 1: Preprocessing Dutch → German Tokenized Data for Later Training. Takes the tokenized data, creates the word and features vocabularies for Dutch as source and German as target languages, and gives an index value to each token. In order to preprocess for German → Dutch translation the order of the source and target languages is reversed.

```
(base) Carolines-MacBook-Air:OpenNMT-py carolinemcdonald$ onmt_preprocess -train_src data/English_PB/e_tr
ain.tok -train_tgt data/English_PB/pb_train.tok -valid_src data/English_PB/e_val.tok -valid_tgt data/Engl
ish_PB/pb_val.tok -save_data data/English_PB/e-pb-processed
[2020-04-27 19:08:44,688 INFO] Extracting features...
[2020-04-27 19:08:44,690 INFO]  * number of source features: 0.
[2020-04-27 19:08:44,691 INFO]  * number of target features: 0.
[2020-04-27 19:08:44,691 INFO] Building `Fields` object...
[2020-04-27 19:08:44,691 INFO] Building & saving training data...
[2020-04-27 19:08:45,216 INFO] Building shard 0.
[2020-04-27 19:08:56,870 INFO]  * saving 0th train data shard to data/English_PB/e-pb-processed.train.0.p
t.
[2020-04-27 19:09:06,560 INFO]  * tgt vocab size: 50004.
[2020-04-27 19:09:06,697 INFO]  * src vocab size: 50002.
[2020-04-27 19:09:07,229 INFO] Building & saving validation data...
[2020-04-27 19:09:07,871 INFO] Building shard 0.
[2020-04-27 19:09:08,013 INFO]  * saving 0th valid data shard to data/English_PB/e-pb-processed.valid.0.p
t.
(base) Carolines-MacBook-Air:OpenNMT-py carolinemcdonald$
```

Figure 2: Preprocessing English → Brazilian-Portuguese Tokenized Data for Later Training. Takes the tokenized data, creates the word and features vocabularies for English and source and Brazilian-Portuguese as target languages, and gives an index value to each token.

2.2 The RNN Model

In using the Recurrent Neural Network with LSTM for machine translation, an encoder-decoder system is used where each is a fully formed neural network of four layers. In it, the encoder is fed an input sequence of words in the source language as a vector **x** and converts them into a continuous intermediate vector **z**. This is in turn fed to the decoder, which creates an output sequence **y** in the target language one word at a time [8]. Each target word is predicted by combining the present hidden state in the decoder's recurrent network with the result of LSTM attention on the given hidden states in the encoder's recurrent network. This predicted word is then fed back into the decoder in order to predict the next target word [2]. The LSTM, or Long Short Term Memory, is a feature of the RNN that allows it to remember previous cell states in the network and thereby use knowledge of previous words to help determine the context and best prediction for the current word [7].

Encoder-Decoder Type	Embedding Size	Hidden Units	Encoder Depth	Decoder Depth	Batch Size	Dropout	Optimizer	Initial Learning Rate
LSTM	512	1024	4	4	128 segments	.3	ADAM	.001

Table 1: Hyperparameters for RNN

```
[2020-04-24 21:37:38,000 INFO]  * src vocab size = 4841
[2020-04-24 21:37:38,001 INFO]  * tgt vocab size = 4249
[2020-04-24 21:37:38,001 INFO] Building model...
[2020-04-24 21:37:39,268 INFO] NMTModel(
  (encoder): RNNEncoder(
    (embeddings): Embeddings(
      (make_embedding): Sequential(
        (emb_luts): Elementwise(
          (0): Embedding(4841, 512, padding_idx=1)
        )
      )
    )
    (rnn): LSTM(512, 1024, num_layers=4, dropout=0.3)
  )
  (decoder): InputFeedRNNDecoder(
    (embeddings): Embeddings(
      (make_embedding): Sequential(
        (emb_luts): Elementwise(
          (0): Embedding(4249, 512, padding_idx=1)
        )
      )
    )
    (dropout): Dropout(p=0.3, inplace=False)
    (rnn): StackedLSTM(
      (dropout): Dropout(p=0.3, inplace=False)
      (layers): ModuleList(
        (0): LSTMCell(1536, 1024)
        (1): LSTMCell(1024, 1024)
        (2): LSTMCell(1024, 1024)
        (3): LSTMCell(1024, 1024)
      )
    )
    (attn): GlobalAttention(
      (linear_in): Linear(in_features=1024, out_features=1024, bias=False)
      (linear_out): Linear(in_features=2048, out_features=1024, bias=False)
    )
  )
  (generator): Sequential(
    (0): Linear(in_features=1024, out_features=4249, bias=True)
    (1): Cast()
    (2): LogSoftmax()
  )
```

Figure 3: RNN Structure

2.3 The Transformer

The Transformer is also built on encoder and decoder neural networks, much like the RNN as described earlier, however it now utilizes the self-attention mechanism rather than the LSTM. As such, the need for a recurrent network is removed and positional-encoding is built into the embedding, thereby allowing the Transformer to know the relative order of each sequence [2]. Each encoder and decoder is also made up of 6 layers rather than 4, with each layer of the encoder also being made up of two sublayers for multi-head self-attention and a fully connected feed-forward network respectively. The layers of the decoder are also made up of sublayers, with both self-attention and the feed-forward as in the encoder, but between them is another multi-head attention layer over the encoder output [8].

Just as LSTM allows the RNN to remember and contextualize each word given the words that came before, attention and self-attention allow the Transformer to consider other words in an input sequence or sentence when processing a single word. In this way, the model can better encode a particular word by understanding the context it comes from [1]. By using 'multi-headed attention', the Transformer improves its ability to examine many different words in a sentence, rather than simply focus on one or two words in particular. Mathematically speaking, it also creates several different query, key and value matrices that are multiplied in parallel by given weight matrices before calculating their attention and ultimately concatenating their results into the final output from the attention layer, to be fed into the feed-forward network [1]. For our purposes, there were 8 'heads' and thus computations performed in parallel, a specification originally used in the first Google paper utilizing self-attention for NMT [9].

Encoder-Decoder Type	Embedding Size	Hidden Units	Encoder Depth	Decoder Depth	Batch Size	Dropout	Optimizer	Initial Learning Rate
Self-Attention	512	512	6	6	4096 tok	.2	ADAM	2

Table 2: Hyperparameters for Transformer

```
[2020-04-30 02:56:08,399 INFO] NMTModel(
  (encoder): TransformerEncoder(
    (embeddings): Embeddings(
      (make_embedding): Sequential(
        (emb_luts): Elementwise(
          (0): Embedding(50002, 512, padding_idx=1)
        )
        (pe): PositionalEncoding(
          (dropout): Dropout(p=0.2, inplace=False)
        )
      )
    )
    (transformer): ModuleList(
      (0): TransformerEncoderLayer(
        (self_attn): MultiHeadedAttention(
          (linear_keys): Linear(in_features=512, out_features=512, bias=True)
          (linear_values): Linear(in_features=512, out_features=512, bias=True)
          (linear_query): Linear(in_features=512, out_features=512, bias=True)
          (softmax): Softmax(dim=-1)
          (dropout): Dropout(p=0.1, inplace=False)
          (final_linear): Linear(in_features=512, out_features=512, bias=True)
        )
        (feed_forward): PositionwiseFeedForward(
          (w_1): Linear(in_features=512, out_features=2048, bias=True)
          (w_2): Linear(in_features=2048, out_features=512, bias=True)
          (layer_norm): LayerNorm((512,), eps=1e-06, elementwise_affine=True)
          (dropout_1): Dropout(p=0.2, inplace=False)
          (relu): ReLU()
          (dropout_2): Dropout(p=0.2, inplace=False)
        )
        (layer_norm): LayerNorm((512,), eps=1e-06, elementwise_affine=True)
        (dropout): Dropout(p=0.2, inplace=False)
      )
      (1): TransformerEncoderLayer(
        (self_attn): MultiHeadedAttention(
          (linear_keys): Linear(in_features=512, out_features=512, bias=True)
          (linear_values): Linear(in_features=512, out_features=512, bias=True)
          (linear_query): Linear(in_features=512, out_features=512, bias=True)
          (softmax): Softmax(dim=-1)
          (dropout): Dropout(p=0.1, inplace=False)
          (final_linear): Linear(in_features=512, out_features=512, bias=True)
        )
        (feed_forward): PositionwiseFeedForward(
          (w_1): Linear(in_features=512, out_features=2048, bias=True)
          (w_2): Linear(in_features=2048, out_features=512, bias=True)
          (layer_norm): LayerNorm((512,), eps=1e-06, elementwise_affine=True)
```

Figure 4: First Part of Structure for Transformer

3 Results

3.1 RNN

```
carolinekmcdonald@ckm-instance-1-vm:~/OpenNMT-py$ CUDA_VISIBLE_DEVICES=0 python train.py -data data/final_data/nl-d
e-processed -save_model nl-to-ger-model --encoder_type rnn --decoder_type rnn --rnn_type LSTM --word_vec_size 512 -
-rnn_size 1024 --batch_size 128 --enc_layers 4 --dec_layers 4 --optim adam --learning_rate .001 --dropout .3 --worl
d_size 1 --gpu_ranks 0 --layers 4
```

Figure 5: Training the RNN on a Single GPU using OpenNMT

3.1.1 Dutch → German

```
carolinekmcdonald@ckm-instance-1-vm:~/OpenNMT-py$ python translate.py -model nl-to-ger-model_step_100000.pt -src data/final_data/dutch_
test.txt -tgt data/final_data/german_test.txt -replace_unk -verbose -output nl-to-ger-pred-tok.txt
[2020-04-27 20:16:50,962 INFO] Translating shard 0.

SENT 1: ['Dit', 'is', 'James', 'Risen', '.']
PRED 1: Das ist echt cool . ''
PRED SCORE: -0.0000
GOLD 1: Das ist <unk> <unk> .
GOLD SCORE: -154.0459

SENT 2: ['Je', 'kent', 'hem', 'wellicht', 'als', 'de', 'journalist', 'die', 'de', 'Pulitzer', 'Price', 'won', 'voor', 'The', 'New', 'Yo
rk', 'Times', '.']
PRED 2: Man könnte sagen , dass sie ein Rezept ist für die Zukunft guter Lebensmittel , ob wir von Barsch oder Fleischrindern sprechen
.
PRED SCORE: -1.5848
GOLD 2: Sie kennen ihn vielleicht , weil er als <unk> den <unk> <unk> .
GOLD SCORE: -324.4589

SENT 3: ['Jaren', 'voordat', 'we', 'Edward', 'Snowdens', 'naam', 'leerden', 'kennen', ',', 'schreef', 'Risen', 'een', 'boek', 'waarin',
 'hij', 'glorieus', 'blootlegde', 'dat', 'de', 'NSA', 'illegaal', 'telefoons', 'van', 'Amerikanen', 'aftapte', '.']
PRED 3: als er also so CO2 : Wenn ich ein Placebo in Form einer weißen Pille anbieten , in der Form einer Aspirintablette , ist das ein
fach eine runde ,
PRED SCORE: -1.9131
GOLD 3: <unk> bevor je jemand von Edward <unk> gehört hatte , schrieb <unk> ein Buch , in dem er <unk> <unk> , dass die <unk> <unk> <un
k> von Amerikanern <unk> habe .
GOLD SCORE: -1040.8088

SENT 4: ['Maar', 'een', 'ander', 'hoofdstuk', 'in', 'dat', 'boek', 'had', 'nog', 'veel', 'meer', 'impact', '.']
```

Figure 6: RNN Predicting German Output from Dutch Test Set. Each Dutch sentence is broken into its tokens and translated into German. Thus, the prediction is performed and scored line by line.

3.1.2 German → Dutch

```
PRED 777: En zo stelt hielden ze 250 Bismark , en verkenden het van robotische voertuigen .
PRED SCORE: -2.0242
GOLD 777: Je hebt het druk , er moet eten op tafel komen .
GOLD SCORE: -254.2391

SENT 778: ['Aber', '15', 'Minuten', 'sind', 'drin', '!']
PRED 778: Maar het Maar deze geraakt .
PRED SCORE: -0.3949
GOLD 778: Je moet ze het <unk> in <unk> , maar die 15 minuten kun je missen .
GOLD SCORE: -460.1078

SENT 779: ['Meine', 'Kinder', 'sind', 'meine', 'Wohlfühloase', ',', 'meine', 'Welt', '.', 'Es', 'müssen', 'nicht', 'die',
 'Kinder', 'sein', '.', 'Es', 'gilt', ',', 'das', 'Summen', 'zu', 'fühlen', ',', 'einen', 'Platz', 'für', 'seinen', 'Seel
enfrieden', 'zu', 'haben', '.']
PRED 779: En mijn reactie hierop was : schilderen , tekenen buitenaardse wezens , buitenaardse werelden , robots , ruimte
schepen , al dat soort dingen .
PRED SCORE: -2.1248
GOLD 779: Mijn kinderen zijn mijn <unk> , ze zijn mijn wereld , maar het hoeven niet je kinderen te zijn die je <unk> voe
den , de plek waar het leven die positieve <unk> heeft .
GOLD SCORE: -806.0035

SENT 780: ['Es', 'geht', 'nicht', 'ums', 'Spielen', 'mit', 'den', 'eigenen', 'Kindern', '.', 'Es', 'geht', 'um', 'Freude'
, ',']
PRED 780: Het gaat over onze mening over mensen als we ze benoemen met deze woorden .
PRED SCORE: -0.1533
GOLD 780: Het gaat niet over spelen met je kinderen , het gaat over geluk .
GOLD SCORE: -232.0648
```

Figure 7: RNN Predicting Dutch Output from German Test Set. Each German sentence is broken into its tokens and translated into Dutch. Thus, the prediction is performed and scored line by line.

3.1.3 English → Brazilian-Portuguese

```
SENT 1261: ['A', 'mouth', 'which', 'tells', 'you', 'where', 'to', 'get', 'out', 'is', 'an', 'exit', '.']
PRED 1261: Uma boca que lhe diz onde chegar é uma saída .
PRED SCORE: -0.3051
GOLD 1261: Uma boca que diz a você onde está a saída significa saída .
GOLD SCORE: -106.6664

SENT 1262: ['This', 'is', 'a', 'slide', 'to', 'remind', 'me', 'that', 'I', 'should', 'stop', 'talking', '
and', 'get', 'off', 'of', 'the', 'stage', '.', 'Thank', 'you', '.']
PRED 1262: Este é um slide para me lembrar que eu devo parar de falar e sair do palco . Obrigado .
PRED SCORE: -0.6600
GOLD 1262: Esse é um slide para eu me lembrar que preciso parar de falar e sair do palco . Obrigada .
GOLD SCORE: -52.3117
PRED AVG SCORE: -0.1404, PRED PPL: 1.1507
GOLD AVG SCORE: -4.3059, GOLD PPL: 74.1339
```

Figure 8: RNN Predicting Brazilian-Portuguese Output from English Test Set. Each English sentence is broken into its tokens and translated into Brazilian-Portuguese. Thus, the prediction is performed and scored line by line.

3.2 Transformer

```
carolinemcdonald@ckm-1-vm:~/OpenNMT-py$ CUDA_VISIBLE_DEVICES=0 python train.py -data data/English_PB/e-pb-
processed -layers 6 -rnn_size 512 -word_vec_size 512 -transformer_ff 2048 -heads 8 -encoder_type transform
er -decoder_type transformer -position_encoding -train_steps 200000  -max_generator_batches 2 -dropout 0.2
 -batch_size 4096  -batch_type tokens -normalization tokens  -accum_count 2 -optim adam -adam_beta2 0.998
-decay_method noam -warmup_steps 8000 -learning_rate 2 -max_grad_norm 0 -param_init 0  -param_init_glorot
-label_smoothing 0.1 -valid_steps 10000 -save_checkpoint_steps 10000 -world_size 1 -gpu_ranks 0
```

Figure 9: Training the Transformer on a Single GPU using OpenNMT

While 200,000 training steps were specified for the Transformer, only 100,000 were run. This was done in order to maintain a fair comparison between the RNN, which was only trained for 100,000 steps. Using the single GPU, the Transformer took over 24 hours to train, double the training time of the base Transformer in [8] using 8 NVIDIA P100 GPUs.

```
SENT 1261: ['A', 'mouth', 'which', 'tells', 'you', 'where', 'to', 'get', 'out', 'is', 'an', 'exit', '.']
PRED 1261: Uma boca que te diz onde correr é uma saída .
PRED SCORE: -4.1792
GOLD 1261: Uma boca que diz a você onde está a saída significa saída .
GOLD SCORE: -54.2113

SENT 1262: ['This', 'is', 'a', 'slide', 'to', 'remind', 'me', 'that', 'I', 'should', 'stop', 'talking', 'a
nd', 'get', 'off', 'of', 'the', 'stage', '.', 'Thank', 'you', '.']
PRED 1262: Este é um slide para me lembrar que eu deveria parar de falar e sair do palco . Obrigado .
PRED SCORE: -1.5070
GOLD 1262: Esse é um slide para eu me lembrar que preciso parar de falar e sair do palco . Obrigada .
GOLD SCORE: -33.4385
PRED AVG SCORE: -0.3479, PRED PPL: 1.4161
GOLD AVG SCORE: -3.0365, GOLD PPL: 20.8328
```

Figure 10: Transformer Predicting Brazilian-Portuguese Output from English Test Set. As with the RNN, each English sentence is broken into its tokens and translated into Brazilian-Portuguese. Thus, the prediction is performed and scored line by line.

Language Pair	Model Type	BLEU Score
German → Dutch	RNN with LSTM	.56
Dutch → German	RNN with LSTM	.47
English → Brazilian-Portuguese	RNN with LSTM	28.56
English → Brazilian-Portuguese	Transformer	31.64

Table 3: BLEU Scores for All Models and Language Pairs. Neither the German → Dutch nor the Dutch → German translations generated high BLEU scores, suggesting that the single direction translation models were quite inaccurate. However, rather than conclude that the structure of the RNN was flawed, it is possible that the data used for the translations was simply lacking in size. As such, the very same RNN structure was used with respect to the English → Portuguese-Brazilian translation, and in turn resulted in a BLEU score of 28.56. This is a significant improvement on the baseline score of 9.33 created by the WIT prior to the 2014 conference. But the Transformer with self-attention generated an even better BLEU score of 31.64, marking the best performance of any of the models tested.

4 Conclusion

Using the hyperparameters as specified by the original paper, the Transformer model generated a noticeable improvement upon the RNN framework. When translating from English → Brazilian-Portuguese, the Transformer gave a BLEU score of 31.64, 3.08 BLEU higher than the RNN with LSTM. This is an even more significant difference than in [2], where the Transformer only gave a BLEU score of 18.37 and the RNN gave 18.05 when translating from Dutch → German. This larger difference is likely due to the fact that they were trained on different language pairs, and as such there are inherent differences in translation as well as in the size of training, validation and testing data.

However, when translating between Dutch and German for this paper, the results generated by the RNN were much worse than the original paper. While [2] gave a BLEU score of 18.05, the recreated model only had .47. As the model I used was exactly the same as that in [2], the only possible source of this discrepancy is in the data it was trained and evaluated on. Though we both seemed to gather our data from the same source, WIT, they must have supplemented their training dataset for the problem as the data provided for the original challenge only has 1779 lines for the Dutch → German pair. On such a small set of data any model trained, no matter the hyperparameters or architecture, will perform poorly as seen in Figure 3 with both German → Dutch and Dutch → German [6].

These dataset issues aside, ultimately the Transformer did outperform the RNN when applied to a dataset made up of a suitable training set size. As such, the results were in line with the findings of [2], and the basis of [8].

4.1 Further Steps

As the Transformer produced the highest BLEU score at only 100,000 steps, immediate improvement could occur simply by increasing the number of steps to 300,000 as Google did in its paper creating the Transformer for NMT [9]. However, this improvement would only occur if the training accuracy had not converged prior to to 100,000 steps–though it seems likely that it has not.

In the larger context of NMT, models with Transformers or RNNs with LSTM can and should be used outside of single directional translation. Instead, they could be used in a bi-directional translation problem or even a 'zero-shot' translation. While the former is self-explanatory, the latter would mean that a model

could translate between any two language pairs without prior training in that target language [2]. This naturally presents a greater challenge from a machine translation perspective, but also a greater reward in terms of universal use in society.

References

[1] Jay Alammar. *The Illustrated Transformer*. 2018. http://jalammar.github.io/illustrated-transformer/

[2] Mauro Cettolo, Marcello Federico, and Surafel M. Lakew. *A Comparison of Transformer and Recurrent Neural Networks on Multilingual Neural Machine Translation*. Fondazione Bruno Kessler. Trento, Italy. 2017.

[3] Mauro Cettolo, Marcello Federico, and Christian Girardi. *WIT3: Web Inventory of Transcribed and Translated Talks*. Fondazione Bruno Kessler. Trento, Italy. 2012.

[4] M. Cettolo et. al. *Overview of the IWSLT 2017 Evaluation Campaign*. IWSLT. 2017.

[5] Ian Goodfellow and Yoshua Bengio and Aaron Courville. *Deep Learning*. MIT Press. 2016. http://www.deeplearningbook.org

[6] Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. *OpenNMT: Open-Source Toolkit for Neural Machine Translation*. Proceedings of ACL 2017, System Demonstrations, 2017. Association for Computational Linguistics. https://www.aclweb.org/anthology/P17-4012.

[7] Christopher Olah. *Understanding LSTM Networks*. 2015. http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[8] Alexander Rush, Vincent Nguyen, and Guillaume Klein. *The Annotated Transformer*. HarvardNLP. 2018. http://nlp.seas.harvard.edu/2018/04/03/attention.html.

[9] Vaswani et. al. *Attention Is All You Need*. Computing Research Repository. 2017. https://dblp.org/rec/journals/corr/VaswaniSPUJGKP17.bib