# Exploring techniques to improve RNN translation performance

**Patrick Kelly**
pkellyl@stanford.edu

## Abstract

Machine translation performance has advanced substantially in the past 6 years. The appearance of RNNs marked the beginning of a rapid progression of architectural advances that included Attention, Convolutional Networks, and, most recently, Transformers. Besides major architectural changes, these advances have included less publicized techniques can improve the performance of any model. In the spirit of the work of Chen, First, Bapna et al, we try to identify techniques that can be applied to improve translation quality. We study a number of possibilities across the NMT process pipeline, which starts with the original text and text preprocessing and culminates in the final translated text. We test new approaches as well as try to confirm the effect of techniques publicized in other research. We developed baseline 2 and 4 head architectures that can be used for further study and optimization. In exploring these architectures, we confirmed weight dropping as one of the more effective techniques to improve system performance. We also found found that preprocessing by separating commas and full-stops form accompanying words raises BLEU performance substantially due to a better interpretation of punctuation and a lower incidence of OoV words. We conclude that for comprehensive NMT system optimization, the entire process pipeline should be considered and a structured approach to optimization efforts should be employed drawing from a toolbox of techniques such as those we tested.

## 1 Introduction

seq2seq models have recently and dramatically evolved the state of the art in Neural Machine Translation (NMT). The encoder-decoder RNN architecture introduced in 2013 ([7] Kalchbrenner and Blunsom, 2013; [8] Sutskever et al., 2014; [9] Cho et al., 2014) set a new stage for increasing NMT SOTA performance. Attention mechanisms provided the next boost ([10] Bahdanau et al., 2015) to performance. Convolutional neural networks applied to NMT ([11] Jouppi et al., 2017) added to the scalability of NMT by facilitating the parallel processing of encoding stages allowing for the efficient implementation of character-based models. The latest step in NMT performance has been delivered by transformer models ([12] Vaswani et al., 2017), which dispense with the recurrent nature of the seq2seq model and rely entirely on self-attention and feed-forward connections.

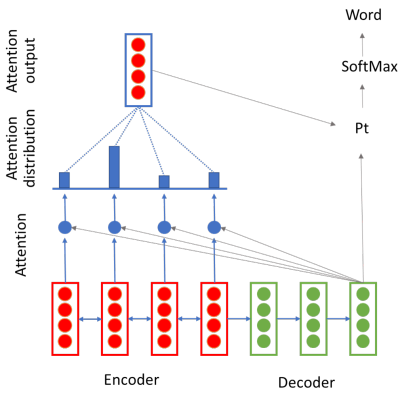
These architectural advances have been accompanied by a number of techniques that can be applied across architectures. Multi-head attention, various regularization approaches (weight drop, dropout, L2 regularization), feature addition to input text (part of speech tagging) are but a few examples. In the spirit of the work done at google AI ([1] Chen, Firal, Bapna et. al) we seek to, on the one hand, identify and test some of these techniques on a simple seq2seq model in order to document their impact and the challenges to reaching performance improvement, and on the other, test other novel approaches that may lead to further study and improvement opportunities.

We now briefly discuss the items we have tested.

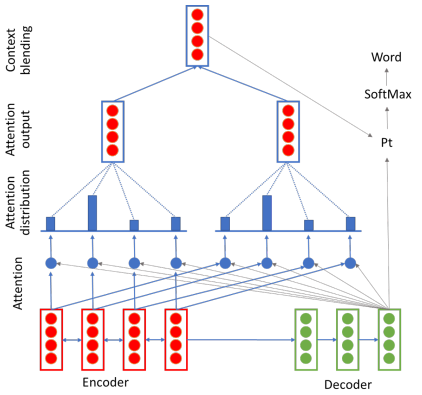
32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

### 1.1 Attending different structural characteristics

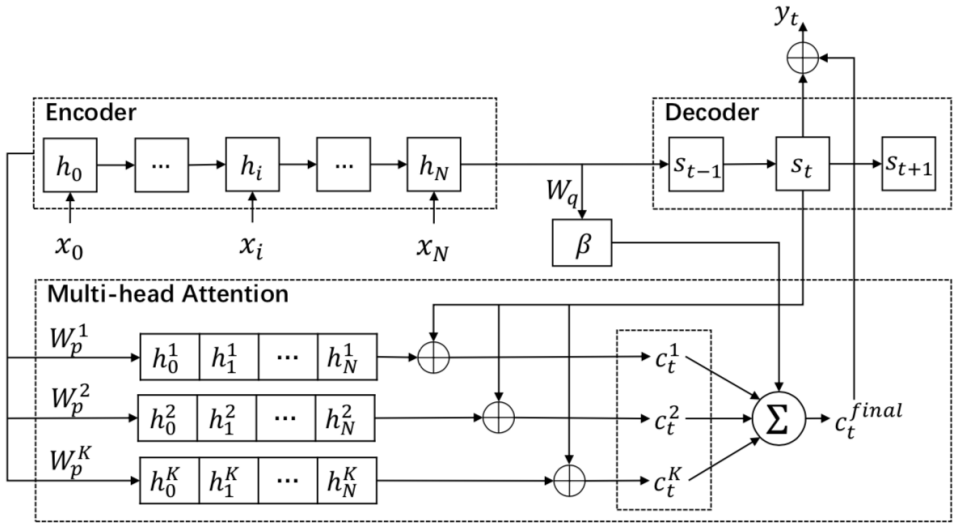
Multi head attention was key factor in the transformer architecture. It drew from the experiences in attention that were in widespread use after initial implementations in 2015 ([10] Bahdanau et al., 2015). We implemented basic single head (figure 1-a), two head (figure 1-b) and 4 head attention (figure 1-c with k=4). In multi-head experiments attention contexts were blended with concatenation and projection as well as with additive attention. The idea behind having more than one attention head is that each head can learn different structural features that contribute separately to better translation.




(a) Basic Attention




(b) 2 Head attention - attention contexts were blended in two ways: concatenation with projection and additive




(c) Multihead attention as implemented by Tao et al. [5] uses additive attention. We implemented this approach with k=4. Figure taken from the original paper

Figure 1: Three attention mechanisms were tested

$$\alpha_{t,i}^k = \frac{e^{g(h_i^k, s_t)}}{\sum_i e^{g(h_i^k, s_t)}};\ c_t^k = \sum_{i=1}^{T} \alpha_{t,i}^k \cdot h_i^k$$

(a) Defining the attention vector ct for a given head k

$$r = softmax(W_q \cdot h_N)$$

$$c_t^{final} = \sum_{k=1}^{K} r_k \cdot c_t^k$$

(b) Combining attention heads for final attention context

Figure 2: Multi head additive attention generates a weighted sum (learned form the final encoder hidden layer though Wq) of attention weights that are the result of a Softmax applied to a function of the encoder hidden weights and the decoder hidden weight for a given step.

### 1.2 Avoiding redundant attention

The intuition behind multihead attention is that each attention head will extract a different feature from the encoder and that these different features, blended additively or through a projection layer, will influence decoding and raise performance. One risk is that different attention heads learn the same feature. To avoid this, we applied similarity penalties. Two approaches were tested - cosine similarity and Frobenius Norm.

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

Figure 3: Cosine similarity was used to generate a similarity penalization for 2 head attention

$$\delta_i^k = \frac{1}{M}\sum_{t=1}^{M} \alpha_{t,i}^k \qquad \mathcal{L}_{penalization} = \|\Delta \cdot \Delta^T - I\|_F^2$$

Figure 4: Delta i,k represents the the average accumulated attention weight on i-th input word for k-th head. The matrix DELTA is the concatenation these average deltas. The penalty for additive attention is the Frobenius norm of DELTA times its transpose minus the identity matrix. The effect if to have each head concentrate on a different feature - potentially a single word - that is different from what other heads are attending.

### 1.3 Learning punctuation better

The text splitting that the base system uses does not separate commas or full stops from the preceding words. This means that words may exist in the vocabulary with and/or without added punctuation. We tested the performance of the system separating the punctuation marks from the surrounding words. This intuition behind this is that the embedding of the commas and stops should convey meaning related to the grammatical functions separate form the words they are next to, and that, conversely, the meaning of words should be separate from the accompanying punctuation.

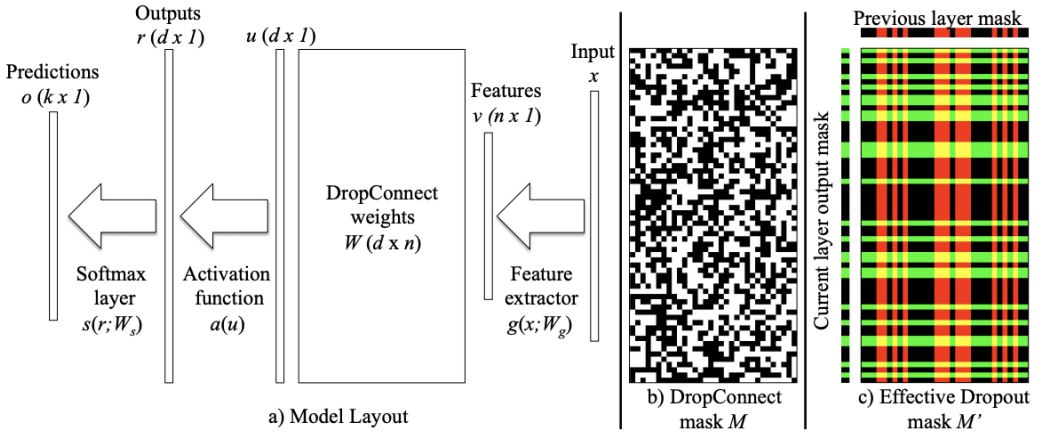
```
Base sentence: "The cat, the dog, and the hen ate dinner. "
Original Split: "The", "cat,", "the", "dog,", "and", "the", "hen", "ate", "dinner. "
New Split: "The", "cat", ",", "the", ",", "dog", "and", "the", "hen", "ate", "dinner", ". "
```

Figure 5: We tested the effect of splitting commas and full stops from words on NMT performance

### 1.4 Effectively regularizing a very complex model

NMT models with attention are complex and have a huge number of parameters. This fact makes them susceptible to high variance. RNNs in particular are hard to regularize since hidden to hidden

connections should retain their continuity in order not to lose meaning. The regularization methods that have the greatest impact on performance are weight drop, hidden drop and embedding drop ([2] Merity et al., 2018). The weight drop approach generates a mask for all time-steps of the RNN, and thus there is consistent, regularized propagation of messages across time-steps. Masks change for each sentence. We tested the Weight Drop model (following [6] Wan et al., 2013, in their DropConnect work) to understand its effect on single and multihead attention models performance. We used t




(a) Basic Attention

Figure 6: DropConnect ([6] Wan et al.), implemented with the WeightDrop class, is a regularization approach that zeros input weights for layers. Figure taken form the original paper.

### 1.5 Failures!

A few of the strategies we tested were, sadly, failures. A summary of these unfruitful endeavours follows. They are not further documented in this report:

- Adding noise: We attempted regularizing by adding 5, 10, or 20 percent of sentences with random word orders. This approach consistently reduced performance.

- Basic curriculum learning: Training the system with gradually sentences of gradually increasing complexity (from short with no punctuation to long with lots of punctuation). The intuition was to mimic human learning, which is progressive. This had no visible effect on the baseline.

- Balanced complexity training set: Balancing the frequency with which sentences of low and high complexity appear during training. The intuition is that short easy sentences appear more frequently in the training set and are translated with higher sentence BLEUs. Longer, more complex sentences are less frequent in the training set and thus contribute disproportionately to reducing corpus BLEU. We tried balancing the number of short/easy and hard/long sentences in the training set to counter this effect. This had no visible impact on performance.

- Adding context word: Averaging the embeddings of all the words in a sentence to be translated and either: a) Creating an 'average word' to be added at the beginning and/or the end of the sentence b) Initializing forward and/or backward encoder hidden layers with this average word. The overall intuition was that by adding this average word into the encoder at some place, encoding could be performed with some knowledge of the broader context of the sentence. This had no visible effect.

# 2 Experiments

## 2.1 Setup, Data, Baseline

We used 4 Amazon Web Services p3.2xlarge instances, which house an NVIDIA P100 GPU, to run 76 distinct experiments. Experiments were run on an incremental basis, adding or varying features to test their impact on BLEU results. Data was the train/dev/test data with parallel Spanish/English sentences used in cs224n assignments. This data set consists of 216,617 train sentences, 851 dev sentences and 8064 test sentences. Each experiment has its own baseline and incremental impact on performance of each step is recorded.

## 2.2 Multi-head attention

### 2.2.1 2 head attention

2 head attention was run with additive and projected attentions strategies.

| Experiment number | Host/ directory | Strategy | BLEU | Delta vs min |
|---|---|---|---|---|
| 2 | ex-1 | 2 attenion heads, regular training set | 19.48 | |
| 8 | ex-5 | 2 attention heads, similarity penalty (cosine similarity) | 20.59 | 1.11 |
| 9 | ex-6 | 2 attention heads, penalty for similar attenitons - double the penalty as original test (score = score - penalty * 2) | 20.89 | 1.41 |
| 62 | ex-6 | Same as 9 but with dropconnect in encoder hh and bias 0.5 - 2 attention layers, cosine penalty * 2 | 21.27 | 1.79 |
| 76 | h3-ex-6 | 74 without embed dropout | 21.49 | 2.01 |
| 74 | h3-ex-6 | 73 with 0.3 dropout in att_p1 att_p2 | 21.9 | 2.42 |
| 65 | ex-6 | 62 plus embed dropout 0.5: 2 heads, cosine penalty, hh and bias weight drop, cosine penalty * 2 | 0.56 | |
| 73 | h3-ex-6 | see 68: 65 but embed dropout 0.3 in stead of 0.5, plus score | 1.01 | |

Figure 7: Results of 2 head attention with projection layers. Cosine similarity penalties on attention heads added 1.41 to the BLEU score. Weight drop on hidden layers and attention layers contributed an additional 0.99. Interestingly adding 0.3 or 0.5 dropout on the embed layer pretty much eliminated learning.

.

| Experiment number | Host/ directory | Strategy | BLEU | Delta vs min |
|---|---|---|---|---|
| 10 | ex-7 | 2 attention heads, penalty for similar attenitons - double the penalty as original test (score = score - penalty * 2) - attentions are combined in a weighted sum, not with a projection layer (as in the paper) | 0.516 | |
| 63 | h4-ex-7 | like 11 but with weight drop hh, bias: 2 heads on encoder hidden layers, no penalty, attentions with weighted sum (as in the paper) | 21.69 | 21.174 |
| 66 | h4-ex-7 | 63 but with cos sim x 2 penalty | 22.23 | 21.71 |
| 67 | h4-ex-7 | 66 plus embed dropout 0.5 | 0.31 | |

Figure 8: Results of 2 head additive attention. Here the baseline was close to zero. Adding weight drop on the hidden layers of the encoder boosted BLEU to 21.69. Adding a similarity penalty gave performance a 0.54 bump. As in the projected attention example, adding dropout to the word embeddings eliminated learning.

### 2.2.2 4 head attention

| Experiment number | Host/ directory | Strategy | BLEU | Delta vs min |
|---|---|---|---|---|
| 23 | ex-7 | four attention heads, additive context, no penalty like nr 11, no ave word, batch 64 | 9.59 | |
| 71 | h5-ex-10 | 70 with penalty 0.01 (vs 0.001) | 17.49 | 7.9 |
| 38 | ex-10 (host 2) | WeightDrop 0.5, batch 64, 4 heads addititve | 18.77 | 9.18 |
| 69 | ex-10 (host 2) | frob norm penalty 4 heads | 19.83 | 10.24 |
| 70 | ex-10 (host 2) | 69 but weight drop 0.6 (vs 0.5) | 19.85 | 10.26 |
| 75 | ex-10 (host 2) | 70 but with dropout 0.3 on each attention head | 21.18 | 11.59 |

Figure 9: Results of 4 head additive attention. Here the baseline was close to not zero but very low - 9.59. Adding weight drop and penalties raised performance by 10.26 points. In this more complex system dropout on attention heads was beneficial adding 1.33 to the BLEU score. The 4 head attention setup was prone to gradient explosion. Weights for loss and penalty scores had to be adjusted to avoid this. 0.999/0.001 and 0.995/0.005 were used.

### 2.2.3 Multi-head attention - discussion

We first note that the highest performance did not exceed a well tuned single head system - for instance, the reference system used in cs224n produced a BLUE score of about 22.60. Here the best is 22.23, reached with 2 head attention with projection layers. In this context we nevertheless can draw valuable conclusions:

- Adding attention heads makes models more complex. Comprehensive analyses of architectures for hyperparameter tuning can involve hundreds of tests for a single architecture. Due to time and processing power (budget) limitations We only had the opportunity to test each architecture with 5 - 10 different variants. In this paper we tried to reach a general understanding of architectures and did not have time or resources to fine tune any particular model

- Models were very susceptible to gradient explosion - when adding a similarity penalty for 4 heads in particular - and vanishing gradients - when multihead architectures were not regularized with WeightDrop or dropouts.

- WeightDrop, consistent with the papers cited in this report, proved to be very effective in regularising as stabilizing complex architectures. When applied to encoder RNNs and attention heads the effects are material.

- Embedding dropout caused vanishing gradients on 2 head models with additive and projected attention models. It, nevertheless provided an improvement in 4 head attention.

## 2.3 Punctuation pre-processing

### 2.3.1 Experiment

We developed a training and test set that separated commas and stops from their surrounding words. Training this way, BLEU scores increased dramatically, from a base of 22.60, to 32.00. To calculate the final score we glued the mentioned punctuation marks to their accompanying words to make comparisons with original results valid. The average increase in sentence BLEU was of 2.07.

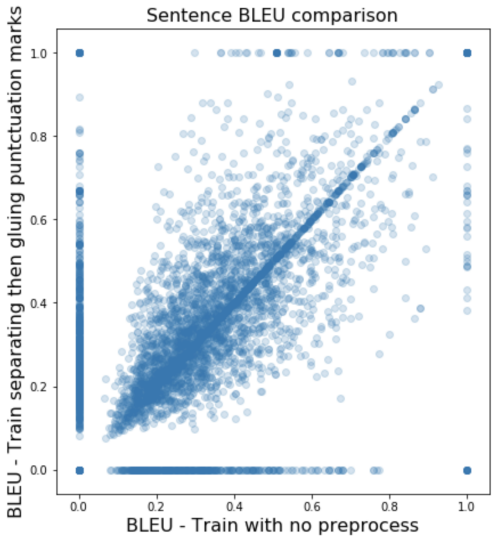



Figure 10: Sentence BLEU comparison for all sentences in test set. The horizontal axis represents normal preprocessing, the vertical axis represents preprocessing separating punctuation (for training) and gluing back of punctuation to accompanying words prior to sentence BLEU measurement. Average sentence BLEU increased by 2.07.

Some examples of the effect of separating commas and full-stops follow:

1) Correctly interpreting commas. Here the standard system inserts '–' where a comma is required. Our preprocessed system inserts the comma correctly.

- Source -> Creo que, como ella, a veces jugamos solos, y exploramos los lmites de nuestros mundos interior y exterior.
- Reference -> I think, like her, we sometimes play alone, and we explore the boundaries of our inner and our outer worlds.
- Standard, Sentence BLEU 20.09 -> I think, as we play – sometimes we play themselves, and we explore the limits of our inner worlds and outside.
- Preprocessed, Sentence BLEU 45.48 -> I think, like, we sometimes play alone, and we explore the limits of our inner worlds and outside.

2) In this example the preprocessing correctly interprets the full-stop at the end of the sentence:

- Source -> Muchas gracias.
- Reference -> Thank you very much.
- Standard, Sentence BLEU 50.81 -> Thank you very much. Thank you.
- Preprocessed, Sentence BLEU 100.00 -> Thank you very much.

3) In a case with several commas, the commas and surrounding words are better translated:

- Source -> Queremos alentar un mundo de creadores, inventores y colaboradores porque este mundo en el que vivimos, este mundo interactivo, es nuestro.
- Reference -> We want to encourage a world of creators, of inventors, of contributors, because this world that we live in, this interactive world, is ours.
- Standard, Sentence BLEU 24.68 -> We want to encourage a globalized world, inventors and collaborators because this world we live, this interactive world is ours.
- Preprocessed, Sentence BLEU 75.15 -> We want to encourage a world of creators, inventors and collaborators, because this world that we live in, this interactive world, is ours.

### 2.3.2 Discussion

Some observed improvements were rather trivial. For instance the reduction in the appearance of <unk>s for the simple reason that a token with a comma next to it now not interpreted as word distinct from the stand alone word. Others, as seen above, were more interesting. The interpretation of punctuation marks was more correct, and surrounding words were better translated. The advantage of investing in preprocessing is that it can produce benefits very cheaply - cleaning data is much easier than learning with a complex neural net.

## 3 Conclusions and further study

### 3.1 Conclusions

Our experiments found avenues that should be considered to improve NMT seq2seq performance in different stages of the process pipeline. Clearly, there are low cost/high impact items such as proper preprocessing, that should always be considered. Regularization, cost function penalties, and architecture decisions are more costly but necessary to fine tune the process. Overall a comprehensive view of the NMT process should be considered and the effort invested in optimization should be allocated considering the cost per unit improvement of each strategy.

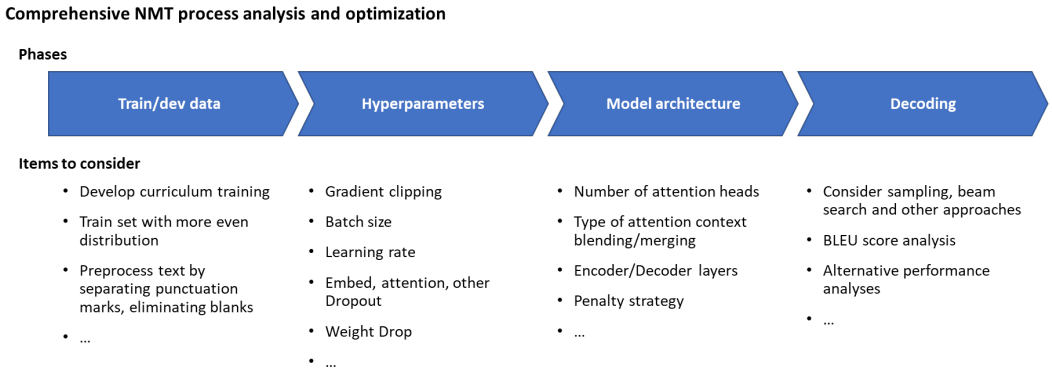



Figure 11: The investment in optimization should be allocated considering the NMT process broadly. Wisely deciding where to allocate time and processing budget is key to reaching best performance withing given constraints.

Specific conclusions

- WeightDrop in the encoder can have a high impact on BLEU, specially in more complex systems with several attention heads. We saw improvements from 1 to 8 points. Whereas this was our experience, it is not a formal benchmark as out experiments were not standardized.
- Embedding dropout in some cases brought about an improvement in the order of 0.5 BLEU points and in others it made the system unstable
- Dropout on attention projections had a positive impact in the order of 0.5 BLEU points
- Preprocessing, in particular treating punctuation marks separately form words, gives words embeddings better tied to their meaning, punctuation marks embeddings more representative of their structural function in the sentence, and reduces OoV words.

### 3.2 Further study

Several avenues for further study appear interesting:

- Developing a SOTA preprocessing standard that can be applied to text prior to translation to achieve the best translation results.

- Developing a framework and automated test bench to better manage the overall analysis and optimization process for NMT systems

- It is our impression that many papers take a hit and miss approach to hyperparameter tuning. A more robust approach, with standard steps and guidelines, would be enormously useful for accelerating future research.

## References


[1] Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Mike Schuster, Zhifeng Chen, Yonghui Wu, Llion Jones, Niki Parmar & Macduff Hughes, (2018) The Best of Both Worlds: Combining Recent Advances in Neural Machine Translation. Google AI

[2] Merity, S., Keskar N.S. & Socher, R. (2018) An Analysis of Neural Language Modeling at Multiple Scales.

[3] Merity, S., Keskar N.S. & Socher, R. (2017) Regularizing and Optimizing LSTM Language Models.

[4] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens & Zbigniew Wojna (2015) Rethinking the Inception Architecture for Computer Vision.

[5] Chongyang Tao, Shen Gao, Mingyue Shang, Wei Wu, Dongyan Zhao & Rui Yan Get The Point of My Utterance! Learning Towards Effective Responses with Multi-Head Attention Mechanism. *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence* (IJCAI-18)

[6] Wan L., Zeieler M., Zhang S., LeCun Y., & Fergus, R., (2013) Regularization of Neural Networks using DropConnect. *Proceedings of the 30th International Conference on Machine Learning*, Atlanta, Georgia, USA, 2013

[7] Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In Conference on Empirical Methods in Natural Language Process- ing.

[8] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural net- works. In Advances in Neural Information Process- ing Systems. pages 3104–3112.

[9] Kyunghyun Cho, Bart van Merrienboer, C aglar Gu lc ehre,FethiBougares,HolgerSchwenk,and Yoshua Bengio. 2014. Learning phrase repre- sentations using RNN encoder-decoder for statistical machine translation. In Conference on Em- pirical Methods in Natural Language Processing. http://arxiv.org/abs/1406.1078.

[10] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In Inter- national Conference on Learning Representations. http://arxiv.org/abs/1409.0473.

[11] Norman P. Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, and et al. 2017. In-datacenter performance analysis of a tensor processing unit. CoRR abs/1704.04760. http://arxiv.org/abs/1704.04760.

[12] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Atten- tion is all you need. CoRR abs/1706.03762. http://arxiv.org/abs/1706.03762.

| Experiment number | Host/ directory | Strategy | input | heads | Weight drop | Separate punct | Encoder layers | Average word | Scrambling | Curriculum | BLEU |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | ex-0 | 2 attention heads, easy to hard training set | batch_pk | 2 | no | no | 1 | yes | no | yes | 5.01 |
| 2 | ex-1 | 2 attenion heads, regular training set | train.es | 2 | no | no | 1 | yes | no | no | 19.48 |
| 3 | ex-2 | 1 attention head, regular traininng set | train.es | 1 | no | no | 1 | yes | no | no | 22.46 |
| 4 | ex-3 | 1 attention head, easy to hard training | batch_pk | 1 | no | no | 1 | yes | no | yes | 21.70 |
| 5 | ass_4 | ass_4 basic - 1 attention | train.es | 1 | no | no | 1 | no | no | no | 19.48 |
| 6 | ex-4 | 2 attention heads, 1 in hidden layers and 1 in embeddings | train.es | 2 | no | no | 1 | no | no | no | 21.43 |
| 7 | ex-4 | 2 attention heads, 1 in hidden and 1 in embeddings | train.es | 2 | no | no | 1 | no | no | no | 21.23 |
| 8 | ex-5 | 2 attention heads, similarity penalty (cosine similarity) | train.es | 2 | no | no | 1 | no | no | no | 20.59 |
| 9 | ex-6 | 2 attention heads, penalty for similar attenitons - double the penalty as original test (score = score - penalty * 2) | train.es | 2 | no | no | 1 | no | no | no | 20.89 |
| 10 | ex-7 | 2 attention heads, penalty for similar attenitons - double the penalty as original test (score = score - penalty * 2) - attentions are combined in a weighted sum, not with a projection layer (as in the paper) | train.es | 2 | no | no | 1 | no | no | no | 0.52 |
| 11 | ex-7 | 2 heads on encoder hidden layers, no penalty, attentions with weighted sum (as in the paper) | | 2 | no | no | 1 | no | no | no | 22.36 |
| 12 | ex-8 | Same as ex-2 but making . , ! ? Be separate words and taking newlines out | train.es | 1 | no | yes | 1 | no | no | no | 33.02 |
| 13 | ex-9 | base = ex2: 1 attention head, regular training set, encoder hnidden layers (both directions) initialized with average word embeddings | | 1 | no | no | 1 | yes | no | no | 21.25 |
| 14 | ex-9 | base = ex2: 1 attention head, regular training set, encoder hnidden layers (only forward direction) initialized with average word embeddings | | 1 | no | no | 1 | yes | no | no | 21.31 |
| 15 | ex-9 | base = ex2: 1 attention head, regular training set, encoder hnidden layers (only backward direction) initialized with average word embeddings | | 1 | no | no | 1 | yes | no | no | 20.89 |
| 16 | ex-2 | same as ex2 but using a train set where the number of longer sentences is higher - 128 batch size | | 1 | no | no | 1 | no | no | no | 20.69 |
| 17 | ex-2 | same as ex2 but using a train set where the number of longer sentences is higher - 32 batch size | | 1 | no | no | 1 | no | no | no | 21.07 |
| 18 | ex2 | same as above but no extra word representing ave embedding of the sentence | | 1 | no | no | 1 | no | no | no | 21.44 |
| 19 | ex2 | same as abive but normal train set 32 batch size no average embed word | | 1 | no | no | 1 | no | no | no | 21.87 |
| 20 | ex2 | same as above but with the average embedd word | | 1 | no | no | 1 | yes | no | no | 22.46 |
| 21 | EX2 | Same as ex-2 base but using resampled trains set (more long sentences) and dev set (before dev set was same as base) - **with average embedd word** | | 1 | no | no | 1 | no | no | no | 21.09 |
| 22 | EX2 | Same as ex-2 base but using resampled trains set (more long sentences) and dev set (before dev set was same as base) - **without average embedd word** | | 1 | no | no | 1 | no | no | no | 21.56 |
| 23 | ex-7 | four attention heads, additive context, no penalty like nr 11, no ave word, batch 64 | | 4 | no | no | 1 | no | no | no | 9.59 |
| 24 | ex2 | Same as ex-2 12 but making '.' and ',' Be separate words and taking newlines out (12 included ! ?) | | 1 | no | yes | 1 | no | no | no | 32.00 |
| 25 | h3-ex-2 | 2 layer LSTM in the encoder - batch 64, separating ， . | | 1 | no | yes | 2 | no | no | no | 32.00 |
| 26 | root (server .85) | 512 embed size | | 1 | no | no | 1 | no | no | no | 18.53 |
| 26 | h3-ex-2 | Same as 25 but not separating ',' and '.' | | 1 | no | no | 1 | no | no | no | |
| 27 | ex-2 | Ex-2 2 layer LSTM encoder | | 1 | no | no | 2 | no | no | no | |
| 28 | ex2 | EX-2 512 embed size (vs 256), b32, no average word | | 1 | no | no | 2 | no | no | no | 22.04 |
| 29 | h3-ex2 | 2 layer LSTM in the encoder - batch 64 | | 1 | no | no | 2 | no | no | no | 22.78 |
| 30 | ass 4 | As reported in gradescope | | 1 | no | no | 1 | no | no | no | 22.61 |
| 31 | ex-2 | Base with dropout in hidden LSTM layers (one per batch) This did not use WeightDrop class | | 1 | no | no | 1 | no | no | no | 22.58 |
| 32 | ex-2 | Dropout on weights (same for each batch) and embeddings (0.5) This did not use WeightDrop class | | 1 | yes | no | 1 | no | no | no | 21.88 |
| 33 | h4-ex-7 | 4 HEADS, DROPOUT 0.3 IN LSTM (This did not have more than 1 layer so droupout did nothing I thinl), no penalty no average word | | 4 | no | no | 1 | no | no | no | 9.59 |
| 34 | h5-ex-7 | 4 HEADS, DROPOUT 0.5 IN LSTM (This did not have more than 1 layer so droupout did nothing I thinl), no penalty no average word | | 4 | no | no | 1 | no | no | no | 9.12 |

Figure 12: Table of tests part 1.

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 34 | h5-ex-7 | 4 HEADS, DROPOUT 0.5 IN LSTM (This did not have more than 1 layer so droupout did nothing I thinl), no penalty no average word | | 4 | no | no | 1 | no | no | no | 9.12 |
| 35 | h3-ex-2 | 3 layert LSTM encoder, no ave, no penalty | | 1 | no | no | 3 | no | no | no | 22.52 |
| 36 | h3-ex-2 | 3 layert LSTM encoder, no ave, no penalty, dropout 0.3 in lstm | | 1 | no | no | 3 | no | no | no | 22.58 |
| 37 | h5-ex-7 (?) | 4 heads dropout 0.5 in lstm, 3 layer lstm, no penalty, b 128 | | 4 | no | no | 1 | no | no | no | 2.50 |
| 38 | ex-10 (host 2) | WeightDrop 0.5, batch 64, 4 heads addititve | | 4 | yes | no | 1 | no | no | no | 18.77 |
| 39 | h3-ex-10 | WeightDrop 0.5, batch 64, 4 heads addititve, embed drop 0.5 | | 4 | yes | no | 1 | no | no | no | 20.67 |
| 40 | h4-ex-10 | WeightDrop 1, batch 64, 4 heads addititve, embed drop 0.5 | | 4 | yes | no | 1 | no | no | no | 20.95 |
| 41 | h5-ex-2 | weightdrop 0.5 embed drop 0.5, batch 64 | | 1 | yes | no | 1 | no | no | no | 21.55 |
| 42 | h5-ex-2 | | | | | no | 1 | no | no | no | |
| 43 | h5-ex-2 | Weightdrop 0.5 embed drop 0.5, batch 64, 2 layer lstm | | 1 | yes | no | 1 | no | no | no | 21.98 |
| 44 | h4-ex-10 | Same as 40 but with dropout in embed = 1!!! | | 1 | yes | no | 1 | no | no | no | 0.46 |
| 45 | h3-ex-2 | ex-2 basic with new version of average word (start of sentence) | | 1 | no | no | 1 | yes | no | no | 22.51 |
| 46 | h3-ex-2 | 45 but no average word | | 1 | no | no | 1 | no | no | no | 22.51 |
| 47 | h3-ex-2 | 46 but 50% of scrambled sentences | | 1 | no | no | 1 | no | yes | no | 13.95 |
| 48 | h3-ex-2 | 47 but 20% scrambled sentences | | 1 | no | no | 1 | no | yes | no | 18.90 |
| 49 | ex-2 | 48 with 10% scrambled sentences | | 1 | no | no | 1 | no | yes | no | 18.45 |
| 50 | h3_ex_2 | 49 with 5% scrambled sentences | | 1 | no | no | 1 | no | yes | no | 20.87 |
| 51 | h3-ex-2 | like 45 - ex-2 basic with new version of average word (end of sentence) b32 | | 1 | no | no | 1 | yes | no | no | 22.61 |
| 52 | h4-ex-1 | #2 with weight drop: 2 attenion heads, regular training set | | 2 | yes | no | 1 | no | no | no | 15.33 |
| 53 | h5-ex-10 | like 39 but with weight drop on hidden and and input to hidden - WeightDrop 0.5, batch 64, 4 heads addititve, embed drop 0.5 (cant get biuas to wirk!!) | | 4 | yes | no | 1 | no | no | no | 10.28 |
| 54 | h4-ex-2 | Repeat 45/51 no average word | | 1 | no | no | 1 | no | no | no | |
| 55 | h2-ex-11 | basic with 0.5 weight drop on encoder and decoder, batch 128 | | 1 | yes | no | 1 | no | no | no | 8.30 |
| 56 | h5-ex-10 | Weight drop 0.5 on hh and bias, 4 heads additive | | 4 | yes | no | 1 | no | no | no | 17.62 |
| 57 | h3-ex-2 | like 51, 45, but no average word | | 1 | no | no | 1 | no | no | no | 22.59 |
| 58 | h3-ex-2 | Repeat old avberage word, after sentence | | 1 | no | no | 1 | yes | no | no | 22.56 |
| 59 | h3-ex-2 | Old average word start and end | | 1 | no | no | 1 | yes | no | no | 22.47 |
| 60 | h5-ex-10 | n 56 but no weight drop - 4 heads additive contexts | | 4 | no | no | 1 | no | no | no | 13.32 |
| 61 | h3-ex-2 | Same as 57, but with dropconnect: basic 1 head model | | 1 | yes | no | 1 | no | no | no | 21.98 |
| 62 | ex-6 | Same as 9 but with dropconnect in encoder hh and bias 0.5 - 2 attention layers, cosine penalty * 2 | | 2 | yes | no | 1 | no | no | no | 21.27 |
| 63 | h4-ex-7 | like 11 but with weight drop hh, bias: 2 heads on encoder hidden layers, no penalty, attentions with weighted sum (as in the paper) | | 2 | yes | no | 1 | no | no | no | 21.69 |
| 64 | h5-ex-10 | 60 but with 0.5 weight drop and embed dropout 0.3 - 4 heads additive | | 4 | yes | no | 1 | no | no | no | 18.97 |
| 65 | ex-6 | 62 plus embed dropout 0.5: 2 heads, cosine penalty, hh and bias weight drop, cosine penalty * 2 | | 2 | yes | no | 1 | no | no | no | 0.56 |
| 66 | h4-ex-7 | 63 but with cos sim x 2 penalty | | | yes | no | 1 | no | no | no | 22.23 |
| 67 | h4-ex-7 | 66 plus embed dropout 0.5 | | | yes | no | 1 | no | no | no | 0.31 |
| 68 | ex-6 | 65 but embed dropout 0.3 in stead of 0.5 | | | yes | no | 1 | no | no | no | |
| 69 | ex-10 (host 2) | frob norm penalty 4 heads | | | yes | no | 1 | no | no | no | 19.83 |
| 70 | ex-10 (host 2) | 69 but weight drop 0.6 (vs 0.5) | | | yes | no | 1 | no | no | no | 19.85 |
| 71 | h5-ex-10 | 70 with penalty 0.01 (vs 0.001) | | 4 | yes | no | 1 | no | no | no | 17.49 |
| 72 | h4-ex-7 | 63 with 0.95/0.05 penalty | | 4 | yes | no | 1 | no | no | no | 22.45 |
| 73 | h3-ex-6 | see 68: 65 but embed dropout 0.3 in stead of 0.5, plus score 0.95,0.05 | | 2 | yes | no | 1 | no | no | no | 1.01 |
| 74 | h3-ex-6 | 73 with 0.3 dropout in att_p1 att_p2 | | 2 | yes | no | 1 | no | no | no | 21.90 |
| 75 | ex-10 (host 2) | 70 but with dropout 0.3 on each attention head | | 4 | yes | no | 1 | no | no | no | 21.18 |
| 76 | h3-ex-6 | 74 without embed dropout | | 2 | yes | no | 1 | no | no | no | 21.49 |
| 77 | ex-10 (host 2) | 75 but 0.5 weight drop on attentions | | 4 | yes | no | 1 | no | | no | 21.06 |
| 78 | h3-ex-6 | 76 with 0.5 weight drop on attention heads | | 2 | yes | no | 1 | no | | no | 21.74 |
| 79 | ex-2 | basic with weoght drio 0.5 | | | | | | | | | 18.41 |

Figure 13: Table of tests part 2.