– Independent Work Report Spring 2019 –

# Making Words Less Wordy

Khyati Agrawal
Adviser: Karthik Narasimhan

## Abstract

*Verbosity in written language is undesirable in many contexts, especially in technical or academic writing. Adhering to the word limits, and presenting the information in a concise manner is a challenge for many new learners, college students and experts of the language alike. Writers are often faced with trade-off between presenting the finer details of their research and keeping their writing of reasonable length. However, as many of us notice while proofreading, length of sentences can be reduced simply by reordering, substituting or deleting redundant words. Because writers feel compelled to include more information to paint a more comprehensive picture, my approach emphasizes preservation of details instead of the deletion of words. In this paper, I customized the recently developed Transformer model to the sentence-level abstractive text compression problem. To put my work in context, "summarization" or "deletion" has been the focus of most existing research to date, and the Transformer model hasn't been previously fine-tuned with my objective. My model attained a ROUGE-2 F1 score of 0.54, and a Compression Ratio of 0.78. I assessed the results from my model both quantitatively and qualitatively. Qualitative assessment indicated that, barring few outliers, the output sentences from the model were readable (although not fully grammatical), moderately compressed, and retained most of the details. Using the ROUGE-2 scores and Compression Ratios of previous summarization tasks as benchmarks, I show that a simple Transformer model generalizes well to reducing verbosity and redundancy in sentences.*

## 1. Introduction

### 1.1. Challenge Overview

Sentence compression and text summarization is a well-researched problem. Over the last two decades, many attempts have been made to compress text sequences, both extractively and ab-

stractively. Earlier attempts were geared towards “Extractive" text summarization which entails deletion of details that are less important. Note that Extractive compression does not put words in context, and Extractive models do not have to deal with the semantic aspects of Natural Language Generation. These models simply weigh the words by importance and “delete" the least important words to achieve a target Compression Ratio (CR). For these reasons, Extractive models are not suitable for the task of intelligent editing and rephrasing, and therefore my approach focuses on Abstractive text compression.

The more sophisticated approach, Abstractive text compression, involves understanding sentences semantically to discover ways of shortening sentences by substitution, reordering and deletion of redundancies. In order to identify redundant and replaceable sub-sequences, the model must:

1. Put words in context. (at least local context, though global context may be helpful as well)
2. Understand the meaning of the words in the sub-sequence
3. Generate a replacement sequence that aims to be syntactically correct.

In other words, “Abstractive" approaches aim to mimic the kind of text compression that a human editor would produce. The complexity of this task lies in the subtleties a model must consider. As we all know, language is generated by combining a large corpus of words in variety of different ways following a set of rules. This generates an infinite number of possibilities as inputs, and exceptions in these rules, stylistic connotations such as sarcasm, similes, dependencies between distant words in a sentence, and the description and interpretation of numbers only serve to make semantic understanding harder. Fortunately, this paper caters to the streamlined task of compressing academic, technical and other non-expressive forms of writing (such as news reports), and so hidden connotations (such as sarcasm) and poetic aspects (such as metaphors, personification) aren’t big considerations. However, understanding facts and numbers, and presenting them concisely is a key aspect for such a problem. Another key aspect of the task is recombining sub-sequences to generate a coherent, grammatical sentences. Together, “semantic" and “syntactic" learning for sentence compression is a interesting research challenge that has been approached in a variety of ways in the past as discussed in Sections 2 and 2.2.

### 1.2. Approach and Implementation Overview

My attempt at this problem involves modifying and fine-tuning the highly successful “Transformer" architecture described in the work “Attention is all you need" [14]. (Details described in Section 4) . Because the Transformer model beat the state of art in domains like multi-document summarization, image generation, and language translation, I adopted the model to see how well it could generalize to Abstractive sentence compression.

The dataset I used was the open source Microsoft Research Abstractive Text Summarization dataset introduced in [12]. For training, I used 9,211 pairs of sentences, the inputs were wordy sentences, and the targets were the human produced concise versions. I used 10% of these pairs to evaluate the results. For the model I experimented with the quality of the target sentences, the number of training steps, the hyper-parameters for the model, the decode parameters and different types of regularization. I present my findings in Section 6.4.

### 1.3. Results Overview

For evaluation, I used the different ROUGE scores, the character compression ratio, and the BLEU score all defined in Section 6.3. Due to difference in size of the data, the results I present in this paper are not directly comparable with those presented in “Related Works”, Section 2.2. However, a high ROUGE-2 F1 score of .54, and a compression ratio of .78 signal the success of the Transformer model in the domain of sentence-level abstractive text compression. Moreover, qualitative results from my experiment suggest that a more sophisticated Transformer model in the future could be directly used by writers to shorten their writing.

## 2. Related Work

### 2.1. Extractive sentence compression

Before the advent of Neural Networks in Natural Language Processing, one of the earliest attempts in the early 2000s at sentence compression involved rule-based approaches to determine the phrases

that do not contribute grammatically to a sentence as presented by Hongyan Jing [5]. Other approaches that followed also focused on selecting a subset of words from a sentence that best convey the meaning and retain grammar. The most notable was Clarke and Lapata's approach in 2008. They used Integer Linear Programming (ILP) to infer compressions that were contextually optimal under specified linguistic constraints [2]. Although fully rule-based, these approaches served as crucial benchmarks for the research that followed; The application of deep learning models such as Recurrent Neural Networks (RNNs) and Long Short Term Memory Networks (LSTMs) in sentence compression began soon after.

The LSTM was a breakthrough model introduced by Sepp Hochreiter and Jurgen Schmidhuber that largely solved the vanishing gradient and efficiency problems in the existing Recurrent Neurak Networks (RNNs) [4]. Though proposed in 1997, LSTMs weren't used majorly in sentence compression till 2015, when a Google Research team of Filippova, Alfonseca, Colmenares, Kaiser, and Vinyals outperformed the state of the art in deletion-based sentence compression using an LSTM approach. Without any parallel data in the form syntactic trees or parsed constituency information that earlier approaches required, they showed that dependencies can be captured by simple tokenization even between distant words [3]. This work was perhaps the most significant in signalling the shift from Linguistic and Rule-based models (requiring excess human labelling to generate syntactic and constituency trees), to LSTM approaches in sentence compression (requiring only tokenization of the data itself).

Most of the works above exclusively use news datasets, which may raise some doubts about the effectiveness of LSTMs for other out of domain technical texts. The robustness of the LSTM model in cross-domain settings was analyzed by another group of researchers from the Beijing Institute of Technology and Singapore Management University. They showed that the LSTM does not perform as well on examples outside of the news dataset, and proposed a hybrid model comprising of both ILP and LSTMs as an improvement [7]. Even though my approach does not involve ILP, or any constraint based models, the findings of the paper above motivated me to search for a more diverse dataset. I discuss ILP and predefined constraints as additions to my model in the Section 7.3.

### 2.2. Absractive text compression

Pioneering the use of LSTM based Deep Neural Networks in Sequence to Sequence modeling, a team of Google researchers introduced a combination of two multilayered LSTMs as an encoder-decoder architecture in 2014 [11]. That is, they used one LSTM as an encoder, and one as a decoder, and matched the state of the art on Translation tasks from English to French. Quickly following this, Li, Luong and Jurafsky from the Stanford CS department introducted the “Attention" mechanism and “Hierarchial LSTMs" which enabled better reconstruction of the original texts after performing the typical encoding/decoding procedure [6]. Work on focusing on (or “attending to" as discussed previously) selective parts of a source sentence “that are most relevant to predicting a target word" was further researched by Bahdanau, Cho and Bengio in 2016 [1]. The “Attention" mechanism was a major milestone in Machine Translation and Abstractive compression because it enhanced semantic understanding of the sentence using contextual information. For reference, a summary and a figure (Figure 8) describing its implementation is included in the appendix, Section 9.

The encoder-decoder architecture, Hierarchial LSTMs, and the “Attention" mechanism were being widely used in Machine Translation, but they weren’t used in Abstractive text compression until the “Attentional Encoder-Decoder Recurrent Neural Network" model for Abstractive summarization was proposed by a group researches from IBM and the University of Montreal in 2016 [9]. They applied the model to the task of Abstractively summarizing a full document to few sentences, and beat the state of the art for two different corpora. Similar works used the DUC corpus as a standard task. The DUC corpus competition uses a document-summary paired dataset, which is significantly different from the task of converting lengthy, convoluted sentences to more concise ones, the problem that I approach in this work.

Despite the popularity of RNN and LSTM models, effort was made to improve the slow training time of these models. The Transformer model was proposed in 2017 in the paper “Attention is All You Need” [14] (Authors: Vaswani, Shazeer, Parmar, Uszkoreit, Jones, Gomez, Kaiser, and Polosukhin). The Transformer model is a novel architecture that dispensed of recurrence and convolutions entirely. It focused solely on “Attention", and greatly improved the training time

and the BLEU scores for Translation tasks . Because of the success of this model, it was used for a variety of different tasks; The most relevant to this discussion is the use of a transformer model for generating Wikipedia articles as a summary of multiple lengthy, related documents. This research was conducted just last year, by a Google Brain team, utilizing a decoder only Transformer [8]. Although, noticeably different from my task, this was the first Transformer based attempt at abstractive compression, and hence the positive results from this research justified my approach.

The closest work to this report was presented very recently in August 2018 (Yu, Zhang, Huang and Zhu 2018) [15]. The team of researchers from Tsinghua University and Microsoft Search Technology Center in Beijing presented an “Operation Network" which built upon the sequence to sequence model with Attention. Using the same encoder architecture as described in ( 2014), they modified the decoder architecture to sequentially perform delete and generate based operations. Their experiment bettered the state of the art in Abstractive Sentence Compression. They achieved an average ROUGE-2 score of .174 and a compression ratio of 0.65 on the MSR Abstractive sentence compression dataset available online[1]. For this report, I adopt the same dataset for training my models, and I compare the results of my experiment to their results as a baseline.

Outside of the research domain, practical online editors such as “Grammarly" rely on either human editing or rule-based restructuring to suggest improvements to sentences. To gauge the utility of the platform, I inputted 10 randomly picked inputs from my dataset into the “Grammarly" online editor. I noticed that while the platform identifies these sentences as convoluted and lengthy, it offers no replacement for the sentence. Because existing online tools are insufficient, expensive and rely heavily on human editing, I exclude these from further comparison and discussion.

## 3. Background

The general structure for the model I used is illustrated in Figure 1 below. The entirety of this section provides a brief description of features and general structure of Transformer model, as presented in the paper “Attention is All You Need" (Vaswani et al. 2017) [14].

[1]The dataset can be downloaded at:
https://www.microsoft.com/en-us/research/project/intelligent-editing/

The Transformer model shown in Figure 1 consists of a stack of layers called the encoder, followed

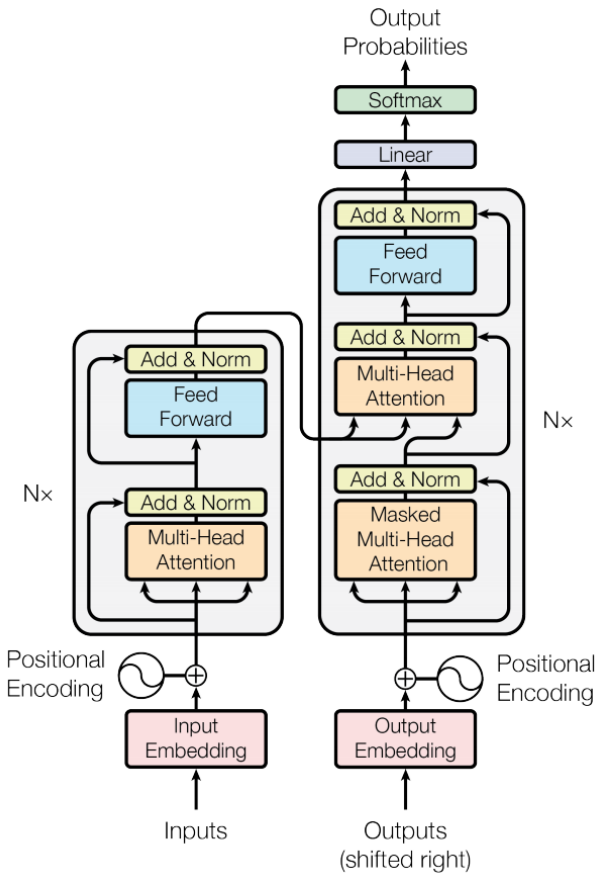


**Figure 1: The Transformer Model from Vaswani et al. 2017**

by a stack of layers called the decoder. The Encoder can have any number N of layers, where each has 2 sub-layers, the first is a multi-head self attention layer, and the second is a fully connected feed forward layer. each sub-layer has a residual connection around it, such that the output from each sub-layer becomes LayerNorm(x + Sublayer(x)) The Decoder comprises of three sub-layers, one additional sublayer to perform multi-head attention over the output of the encoder stack. With the "Masked Multi-Head Attention" in the decoder prevents positions from attending to subsequent positions. Figure 9 illustrates how dependencies between different time steps are captured in through Attention in the Transformer.

The attention function in the model in [14] maps a query ($Q$) and a set of key ($K$) - value ($V$) pairs to an output that is a weighted sum of the values, where the weight assigned to each value is

computed by a compatibility function of the query with the corresponding key. The Transformer model presented in the paper used Scaled Dot-product Attention [14], mathematically represented as:

$$Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

where $Q,K,V$ represent the query, key and value matrices respectively; $Attention(Q,K,V)$ represents the matrix of outputs. $\frac{1}{\sqrt{d_k}}$ is the scaling factor, where $d_k$ is the dimension of the keys $K$.
The researchers introduced the mechanism of "Multi-head attention", that "allows the model to jointly attend to information from different representation subspaces at different positions." The three-step algorithm for "Multihead attention" in [14] is as follows:

1. Linearly project the queries, keys, values, h times, each time with different, learned linear projections
2. Perform the attention function On each projected version of queries, keys, and values in parallel."
3. Concat and linearly re-project the outputs of attention heads.

From [14], the method can be represented mathematically as,

$$MultiHead(Q,K,V) = Concat(head_1,...,head_h)W^o$$

where $head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$; and $W_i$ are the projection matrices with the appropriate dimensions. Figure 10 in the Appendix is a flow chart of "Multihead attention", and Figure 11 highlights the advantage of this method in capturing dependencies between different parts of the sequence.

## 4. Approach

From the history of works on text compression as described in the Sections 2 and 2.2, it is evident that models developed for broader tasks generalize well to text compression problems. It is also clear that modifying existing models to fit a particular problem has been a recurrent approach in

research with much success. I observed that the domain of abstractive text compression, and more specifically, sentence-level abstractive compression has remained relatively unexplored. I also noticed that even though the Transformer had been used with great results for other tasks like Image Generation, and document summarization, it was yet to be applied to sentence-level abstractive compression. Furthermore, analysis presented in [14] shows that, by using Self-Attention, the Transformer model reduces the maximum path length between two words in the input sentence from $O(n)$ (in RNNs) to $O(1)$, consequently improving Backpropagation learning by enabling long-range relationships between words to propagate faster. The number of sequential operations also greatly reduced from $O(n)$ in RNNs to $O(1)$ in Transformers, which allowed for parallel encoding of input words, followed by parallel decoding, thereby greatly reducing the training time. The reasons above motivated me to design my experiment around fine-tuning Transformer model for the sentence-level Abstractive text compression. To best fit the model for my task, I experimented with :

1. Data having different quality of compressed sentences
2. Hyper-parameters for the encoder and parameters for the decoder
3. The number of training steps (or equivalently training time)
4. Different kinds of regularization (for example: layer dropout vs attention dropout)

The dataset I used contained human compressed sentences of different quality. Each compression was judged on a scale on information retention and grammar. For initial runs, I used all, the "good", "average" and "bad" compressions for training and evaluation. In later runs, I narrowed my dataset to only contain the 2 or 3 best targets and noticed significant improvements. The hyperparameters I tuned for the encoder were

1. Hidden layers: The number of hidden layers in the encode (i.e. the variable N in Fig 1)
2. Hidden Size: The number of trainable parameters for each layer is a function of the hidden size. Or, More exactly, the number of trainable parameters in the feed-forward sub-layer.
3. The learning rate: The starting learning rate, typically annealed or modified using the Adam Optimizer.
4. Attention dropout: The dropout ratio applied to the sums of embeddings and the positional

encodings (This mechanism is applied in the beginning; the sums are the inputs to encoder stack (Fig 1))

For decoder, a tried a few different values for the "alpha" and "beam-size" parameters. Towards the end, I experimented with using dropout for the output for each sub-layer in addition to Attention dropout. For evaluation, I chose to focus my analysis on the ROUGE-2 F1 score and Compression Ratio because these metrics are a standard for research in text compression tasks. I also report ROUGE-2 Recall, ROUGE-1 Recall, ROUGE-L scores, and BLEU scores for the purpose of comparison with previous work.

## 5. Implementation

### 5.1. Data preprocessing

I wrote a preprocessing script to convert the dataset into a simple format of pair-wise input and target sequences. I filtered the data to get rid of the sentence scores, combined ratings, judge ids etc., delimiters such as "|" using standard python file reader and writer libraries.

### 5.2. Framework and Libraries

I utilized the Tensor2Tensor(T2T) library of deep learning models for my implementing my experiment [13]. T2T allows easy access to a variety of models, and hyperparameter sets for these models. It also provides an easy way of processing and generating data. Because the model is bulit on TensorFlow framework, it allows the use of the TensorBoard framework for visualizing training runs, and learned model graphs.

First I registered a problem in the T2T library [13]. The problem setup allowed me to define a class for my problem; I picked "Text2Text"; Considering the fairly limited vocabulary of news articles, technical documents and business letters, which is what my dataset comprised of, I chose a vocabulary size of around 8K. I defined a 90/10 split for partitioning the training and evaluation data. Within the problem, I also defined the initial values to use for hyperparameters, and the ranges to search in while performing hyperparameter tuning.

I used the T2T datagen library to render the input-target pairs in a format usable by T2T models [13]. This library handles tokenization, stemming and all other preprocessing required to handle the associated problem as detailed above. The advantage of this approach is the need to process data only once, and use the same data and problem for all training runs. This also ensures consistency of data throughout the experiment.

Likewise, I used the T2T trainer (varying either the hyperparameters, or the number of train steps, or the data itself each time) to train and evaluate my model [13].

Finally, I used the T2T decoder to obtain the actual predictions. On these predictions, I used my own post-processing scripts and functions to analyze results as described in later sections [13].

### 5.3. Platform

I ran all my training tasks on the Google Cloud compute engine, on a gpu. I used the Datalab environment to run the code in a jupyter notebook interface.

### 5.4. Visualization

I used the TensorBoard framework, built upon TensorFlow to visualize the training runs, to diagnose problems and to identify potential improvements in the training process.

### 5.5. Hyper-parameter tuning

I used the T2T "autotune" properties, to perform hyperparameter search in the grid of values that I originally described for my problem. This selects the best set of hyperparameters based on a validation set to improve results. For later steps, I manually modified my hyperparameters to include a different type of regularization called the $layer\_prepostprocess\_dropout$, and observed the changes in results.

# 6. Evaluation

## 6.1. Experiment design

The experiment was designed to discover the best hyperparameters, the optimal training time, and the most appropriate dataset to apply the Transformer model to Abstractive Sentence Compression. The ranges I used for hyperparameter grid search were:

1. Number of hidden layers: Integers between [2, 4]
2. Hidden Size : In the set [128, 256, 512]
3. Attention dropout: Float values between [0.4, 0.7]
4. Learning rate: Float values between [0.05, 0.25]

I varied the number of training steps in the set [20K, 30K, 50K, 120K]
The values I tried for pre-postprocess dropout were [0, 0.1, 0.54, 0.6]. These values were tried manually and not using auto-tuning procedure in T2T. For the "beam size" parameter in the decoder I tried commonly used values of 4, 6 and 10. For the "alpha" parameter in the decoder I tried the values 0.1, 0.3 and 0.6.

## 6.2. Data

The Microsoft Research Abstractive Text compression dataset first used by Toutanova [12], has 26,000 pairs, 6,000 unique sentences and each sentence has four or five compressions scored on a scale of grammar and detail preservation. The inputs are either single sentences (3,769 in number) of a short paragraph of two sentences (2,231 in number). The dataset contains inputs from news journals, business letters and technical documents making it fairly diverse and suitable for the task. The dataset is pre-split into a training, validation and test set, however I for training my final model, I utilize only the inputs and their best compressions. To simplify data generation with the T2T libraries, I disregard the train, validation and test divisions. The subset of the data I use contains 8,290 pairs training and 921 pairs for evaluation.

### 6.3. Metrics

The two primary metrics that I use for evaluation are the ROUGE-2 F1 score, and the compression ratio. I use these and some secondary metrics described below:

**6.3.1. ROUGE-2 score:** ROUGE-2 refers to the number of overlapping bigrams between the output sentences and the targets.

The ROUGE-2 Recall score is calculated as:

$$\text{ROUGE-2}_{\text{Recall}} = \frac{\text{Number of overlapping bigrams in the model outputs and targets}}{\text{Number of bigrams in the targets}}$$

The ROUGE-2 Recall score by itself isn't sufficient because a very long output sentence may be capturing all the bi-grams in the target sequence, but this output is contrary to our task of compression. Therefore, it important to include a measure of compression.

The ROUGE-2 precision score enables us to do that roughly. It is defined as:

$$\text{ROUGE-2}_{\text{Precision}} = \frac{\text{Number of overlapping bigrams in the model outputs and targets}}{\text{Number of bigrams in the outputs}}$$

If the output length is very high then the $\text{ROUGE-2}_{\text{Precision}}$ will be low indicating that the model's output is much longer than the target sentence. Moreover, the $\text{ROUGE-2}_{\text{Precision}}$ can be 1 only if the bigrams in the output exactly match the bigrams in the target; not less, not more. Therefore, $\text{ROUGE-2}_{\text{Precision}}$ score acts as a penalty for long output sequences, and serves as a rough measure of compression.

The ROUGE-2 F1 score, which my primary method for evaluation, combines these two metrics by taking their harmonic mean.

$$\text{ROUGE-2}_{\text{F1}} = \left(\frac{\text{ROUGE-2}_{\text{Recall}}{}^{-1} + \text{ROUGE-2}_{\text{Precision}}{}^{-1}}{2}\right)^{-1}$$

A very high ROUGE-2 F1 score could be deceptive because the input sentence may be changing only slightly. If the input is reproduced almost as it is, the Recall will be high, and in turn the

harmonic mean could be high despite zero compression. This is not desirable for compression tasks. On the other hand a very low ROUGE-2 F1 score indicates that there are very few overlapping bigrams between the outputs and the targets, suggesting that the outputs are far from the desired targets. Therefore, I approached the task with the aim of obtaining a mid-range ROUGE-2 F1 score in [0.4, 0.8] approximately. To see why mid-range ROUGE-2 F1 scores could be empirically better, refer to the Figures 12 and 12 in the Appendix 9. The plots indicate that for the initial training steps the ROUGE-2 F1 score is very high because the model is leaving the input unmodified. As training proceeds, the ROUGE-2 F1 score sharply decreases first, and then gradually rises to mid-range values as the model learns to compress better.

**6.3.2. Compression Ratio:** For a more direct measure of compression, I calculated the compression ratio as:

$$\text{Compression Ratio (CR)} = \frac{\text{Number of characters in the output}}{\text{Number of characters in the input}}$$

For the final model, I additionally report the BLEU, ROUGE-L F1, ROUGE-L Recall, ROUGE-1 Recall and the ROUGE-2 Recall scores for comparison with related works. The ROUGE-L (both F1 and Recall) are defined in the same way as the ROUGE-2 scores, except that instead of the number of bigram overlaps, the length of the longest common sequence between the output and target is used. Similarly for calculating ROUGE-1 scores, the number of overlapping unigrams is used.

**6.3.3. BLEU score:** The bilingual evaluation understudy (BLEU) score is decribed in detail in [10]. Mathematically the BLEU score is calculated as:

$$\text{BLEU} = \min\left(1, exp(1 - \frac{\text{target-length}}{\text{output-length}})\right)\left(\Pi_{i=1}^{4} precision_i\right)^{\frac{1}{4}}$$

$$precision_i = \frac{\sum_{\text{snt}\in\text{Cand-Corpus}} \sum_{i\in\text{snt}} \min(m^i_{cand}, m^i_{ref})}{w^i_t = \sum_{\text{snt'}\in\text{Cand-Corpus}} \sum_{i'\in\text{snt'}} m^{i'}_{cand}}$$

Here, snt stands for "sentence"; Cand-Corpus stands for "the data, i.e. input-target sequences"; $m^i_{cand}$ is the count of i-gram in output matching the target translation; $m^i_{ref}$ is the count of i-gram in

the target translation; $w_t^i$ is the total number of i-grams in output translation. The above mathematical definition was obtained from [2].

### 6.4. Quantitative results

Table 1 represents the improvement in metrics when the quality of training data was improved by filtering out the “average" and “bad" target compressions. Table 2 summarizes the changes made after the tuning process. Table 3 represents the improvement in metrics after tuning selected hyperparameters.

| **Data** | ROUGE-2 F1 | ROUGE-L |
|---|---|---|
| Full | 0.413 | 0.61 |
| “Best” Targets | 0.508 | 0.61 |

**Table 1: Improvements by improving quality of compressions**

| **Hyperparameter** | **Old Model** | **New Model** |
|---|---|---|
| Hidden Layers | 6 | 3 |
| Hidden Size | 512 | 256 |
| Initial Learning Rate | 0.05 | 0.1925 |
| Attention Dropout | 0.1 | 0.6 |

**Table 2: Changes made to hyperparameters**

| **Model** | ROUGE-2 F1 | ROUGE-L |
|---|---|---|
| Base Transformer | 0.508 | 0.610 |
| Hyperparameter Tuned | 0.530 | 0.613 |

**Table 3: Improvements by tuning hyperparameters**

Tables 2 and 3 , provide evidence that a simpler model with fewer layers, smaller size, more dropout and higher learning rate performs much better on the held-out examples. This shows that my model was grossly fitting when I was using the default Transformer model (For context, decreasing complexity, increasing regularization reduces overfitting to the Training examples). The training plots in Figures 2 and 3 also corroborate this hypothesis. Notice that the gap between the training and evaluation loss is much much smaller for the simpler, more regularized model in Figure 3.

[2]Source: https://cloud.google.com/translate/automl/docs/evaluate

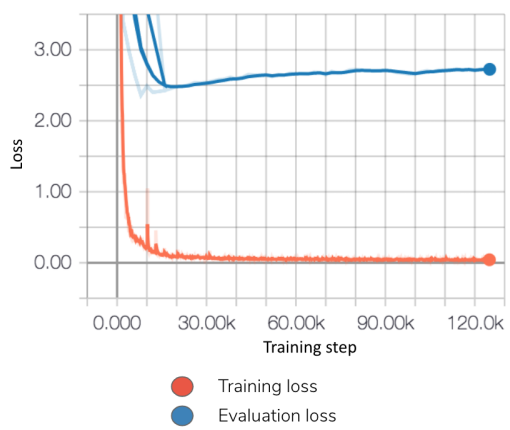


**Figure 2: Transformer Default**

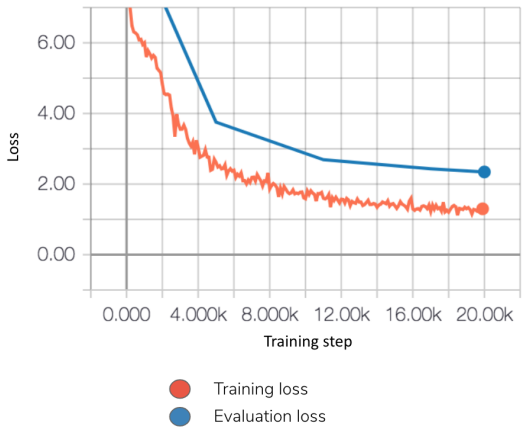


**Figure 3: Transformer Tuned**

The plot visualizing the grid search for the best hyperparameters is attached in Figure 4 for reference.

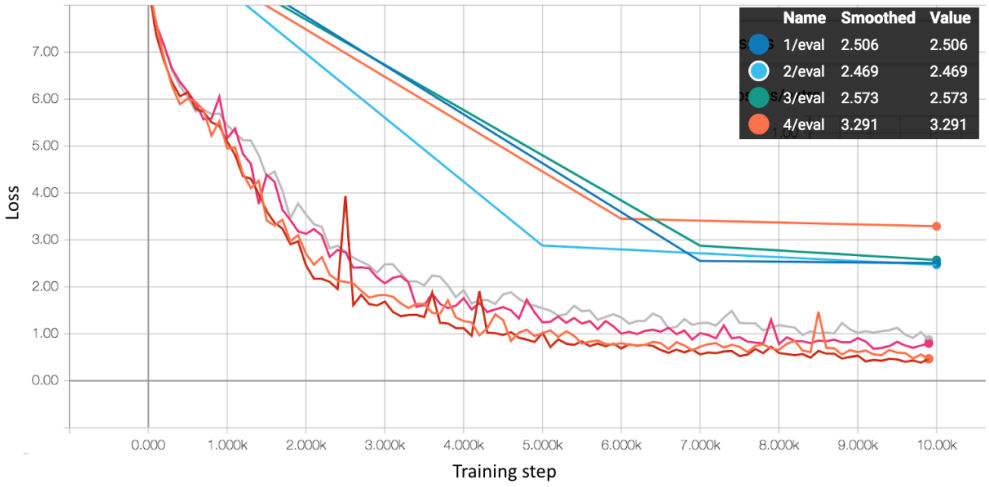


**Figure 4: The Training and Validation loss curves for 4 hyperparameter sets**

Layer pre-postprocess dropout refers to dropping a ratio of outputs from a sublayer and making them zero before they serve as inputs to the next sublayer. Manually experimenting with different values of layer dropout also marginally improved the ROUGE-2 F1 from 0.530 to 0.547.

The optimum number of Training steps remained inconclusive from my experiment. Reducing Training steps from 125K to 20K improved the evaluation metrics slightly, but between 20K and 30K metrics slightly improved going from 20K to 30k. This may be because past a certain time, the learning rate becomes extremely small, and varying the number Training steps causes little or

no change in the learned parameters. I drew the loose conclusion that training for 125K steps is unnecessary and perhaps detrimental given the amount of training data that I used. Likewise, the decoder parameters "beam size" and "alpha" didn't affect the results significantly, but qualitative observation suggested the values 4 and 0.6 respectively worked well. I decided to train my final model for 30K training steps and use the set of hyperparameters described in the "New Model" column of table 2, along with an additional layer pre-postprocess regularization of 0.6.

### 6.5. Qualitative results

The predictions from the default Transformer model showed some invalid recombination of tokens to generate invalid and out of vocabulary words. One such example is represented below. Notice that "counselzed" is not a word, and the sentence has out of input words like "members".

| **Input** | **Prediction** | **Target** |
|---|---|---|
| Meantime, she has graduated from Coppin State College, works as a drug and alcohol counselor, left subsidized housing and plans to start work on a master's degree this year. | Members has graduated from Coppin State College, works and alcohol **counselzed** and plans to start on a master this year. | She has graduated, works as a counselor, left subsidized housing and plans to start work on a master's degree this year. |

**Table 4: Invalid recombination of token in the default Transformer**

The prediction for my customized model is included in Table 5. The new prediction not only reduces the invalid token recombination problem, but also makes grammatical sense, and retains most of the detail. (Notice in example 2: The default Transformer was splitting the word "Action" to "act" and "ion" and dropping "act", while incorrectly retaining "ion". The new model drops the word "Action" as a whole suggesting some improvement).

I used the input sentence length, the ROUGE-2 F1 score, and CR of individual pairs to see if there was a pattern in the sentences that were poorly compressed. The plot above (Figure 5) is 3D visualization of 477 predictions of unique inputs. From the plot, we can see that the model doesn't perform well on very long input sequences (i.e. sequence length of greater than 300 characters). For these outlier sentences the CR is around 1 indicating that, with respect to length of the input, little

| Input | Old Prediction | New Prediction |
|---|---|---|
| Meantime, she has graduated from Coppin State College, works as a drug and alcohol counselor, left subsidized housing and plans to start work on a master's degree this year. | Members has graduated from Coppin State College, works and alcohol **counselzed** and plans to start on a master this year. | Meantime, she has graduated from Coppin State College, works as drug and alcohol counsel plans to work on a master's degree |
| We'll also award more small grants through our Species Action Fund to help a wide range of animals, including whooping cranes, giant river otters, and Tibetan antelopes. | We'll also award more small grants through our Species **ion** Fund, including **whoing** cranes, giant river and Tibetan **pes** | We'll award more small grants through our Species Fund to help a wide range of animals, including **woping** cranes, giant river ters, and Tibetan antelopes. |

**Table 5: Qualitative improvement in predictions using customized model**

or compression occurs. For some very short sentences, the model may be compressing too much as indicated by the few points that have a ROUGE-2 of almost 0 (Notice the 5 points in the bottom left where the input length is less than 100). For sentences having very few words, dropping each

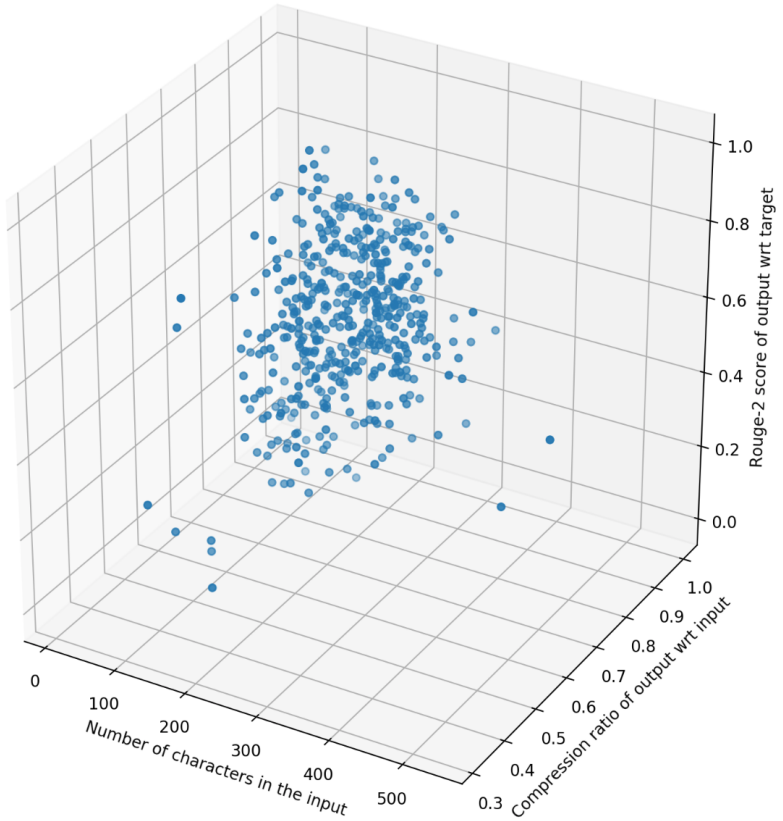


**Figure 5: Variation of the prediction ROUGE-2 F1 score, and CR with the input sequence length**

word can lead to significant compression, but also leads to loss in important details. It is satisfying, however, that most of the input sentences were of length 100 to 300, and for these regular sentences the ROUGE-2 hovered around 0.4-0.6 with a CR of 0.7-0.9 (Observe the higher density, and dark blue color of dots in these ranges).

From the visualization, it is clear that length of input sentence was a fairly important consideration. A more sophisticated model should be adaptable to input sequence length. One rudimentary way to approach this problem could be defining a threshold for the compressed sentence length, and passing all sentences already below the threshold unchanged through the model.

### 6.6. Comparisons

**6.6.1. Comparison with Related Works:** Table 6 provides a comparison of the results from my model to the "OperationalNetwork" model in [15], the work from where I obtained the dataset. I provide the metrics below as benchmark and not a direct comparison, because I selected a subset of the dataset for training and evaluating my model. The results are comparable only as a measure of performance on a held-out set. I make no further claims because of significant variation in size of evaluation sets

| Model | CR | ROUGE-1 | ROUGE-2 | ROUGE-L | BLEU |
|---|---|---|---|---|---|
| OperationNet | 0.655 | 0.362 | 0.174 | 0.337 | .2630 |
| TunedTransformer | 0.777 | 0.757 | 0.568 | 0.755 | 0.440 |

**Table 6: Comparison with (Yu et 2018) [15]**

A salient takeaway from the comparison in Table 6 is that, based on prior benchmarks, the regularized Transformer shows promise of being a better model for Abstractive Text Compression. Another key metric to note is the high BLEU score of 0.44. According to research standards in Machine Translation, a BLEU score of above 0.4 indicates high quality translations [3]

[3]Source: https://cloud.google.com/translate/automl/docs/evaluate

**6.6.2. Comparison with Target compressions:** From Figure 6 , we can see that the human produced compressions achieve a compression ratio of around 0.7, while the results from the model's outputs have an average CR of 0.78

However, it is important to note that the dispersion of CRs for Output sentences is much higher. The standard deviation from the average CR appears to be very low for the Target compressions, and very high for the model's outputs. This suggests that even though my model achieves an average CR comparable to that of the Targets, it fails to maintain consistency of the CR. The dispersion is especially high for inputs sequences of length less than 150 as is seen from Figure 6. (For mid-sized inputs the deviation from mean is reduced; Interestingly, the shape of the plot of blue dots resembles a right arrowhead ">" indicating more consistent CR for mid-sized and long input sequences)

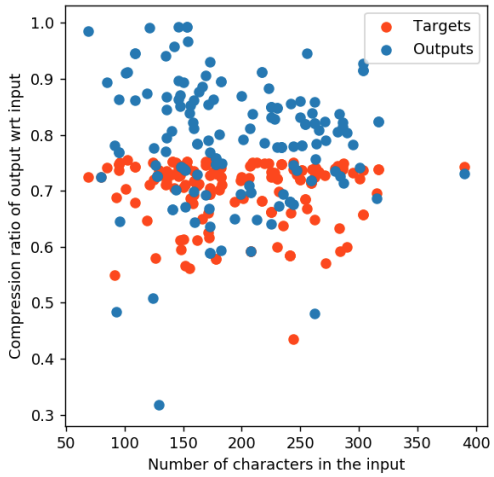


**Figure 6: Compression ratios of Targets and Outputs for 200 randomly picked sequences**

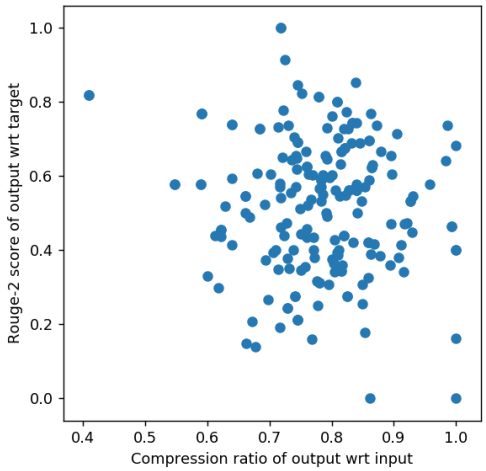


**Figure 7: Variation of CR with ROUGE-2 F1 score for 200 randomly picked sequences**

### 6.7. Trade off between compression quality and compression ratio

From the comparison in Table 6, we observe that the improved ROUGE and BLEU scores come at the cost of a worse Compression Ratio. This makes intuitive sense, because past a certain point, compression can be performed only by losing out on details, or by reducing the semantic quality of the output. This raises another interesting question of whether it is possible pass "the scale of the desired compression" (ranging from highly to lightly compressed) as a parameter to the compression

model. In fact the end user experience goal for a comprehensive model could be to enable users to input a word limit or page limit as parameter.

Figure 7 loosely demonstrates the tradeoff between compression quality and compression ratio. We can see that for most sequences having compression ratios below 0.8, the ROUGE-2 scores are under 0.5 indicating compromised compression quality. Determining the correct balance between the compression quality and compression ratio is another interesting question that depends on the input to and the intended users of the model.

# 7. Summary

## 7.1. Conclusions

In this paper, I showed that the Transformer model generalizes well to the task of sentence-level Abstractive Text compression. My experiment showed that simplifying the default Transformer by reducing the number or size of layers, increasing regularization by adding or increasing dropout, ensuring the quality of target compressions, and using appropriate parameters for decoding produces very good results on the MSR dataset [12].

## 7.2. Limitations

By quantitatively and qualitatively analyzing my results, I discussed the following limitations:

1. Invalid recombination of tokens to form nonexistent words
   (Although greatly reduced after customizing, the issue still persisted for some inputs)
2. Large deviation from the average compression ratio raised concerns about consistency
3. Lack of robustness for outlier sequences (i.e. inputs that too long or too short)
4. Inability to make direct comparisons to closely related works due to my smaller dataset

## 7.3. Future Work

In this paper, I discuss several considerations for Future Work in this domain. A hybrid model with the Transformer encoder and Integer Linear programming constraints within the Decoder

could help solve the invalid recombination of tokens. A hybrid model could compress very short sentences using predefined constraints; This is similar to the "thresholding" concept I discussed in Section 6.5. The results and limitations of my work highlight two things; First that the Transformer model is suitable for Abstractive Sentence Compression, and second that there is a lot of room of improvement in the model by tweaking the model and adding simple mechanisms to the model. In Section 6.7, I bring forward two interesting ideas that could be researched in the future; The first question explores if its possible to develop a model that accepts the desired compression level as a parameter from the user; this would allow writers to specify "how much" they want their text to be compressed. The second question explores if there is an appropriate way to allocate importance between the metrics measuring compression quality and those measuring the amount of compression to ensure a good balance between compression, detail preservation and syntactic correctness.

## 8. Ethics

This paper represents my own work in accordance with University regulations.

Khyati Agrawal

## References


[1] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2014, cite arxiv:1409.0473Comment: Accepted at ICLR 2015 as oral presentation. Available: http://arxiv.org/abs/1409.0473

[2] J. Clarke and M. Lapata, "Global inference for sentence compression an integer linear programming approach," *J. Artif. Int. Res.*, vol. 31, no. 1, pp. 399–429, Mar. 2008. Available: http://dl.acm.org/citation.cfm?id=1622655.1622667

[3] K. Filippova *et al.*, "Sentence compression by deletion with lstms," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP'15)*, 2015.

[4] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997. Available: http://dx.doi.org/10.1162/neco.1997.9.8.1735

[5] H. Jing, "Sentence reduction for automatic text summarization," in *Proceedings of the Sixth Conference on Applied Natural Language Processing*, ser. ANLC '00. Stroudsburg, PA, USA: Association for Computational Linguistics, 2000, pp. 310–315. Available: https://doi.org/10.3115/974147.974190

[6] J. Li, M. Luong, and D. Jurafsky, "A hierarchical neural autoencoder for paragraphs and documents," *CoRR*, vol. abs/1506.01057, 2015. Available: http://arxiv.org/abs/1506.01057

[7] W. Liangguo *et al.*, "Can syntax help? improving an lstm-based sentence compression model for new domains," 01 2017, pp. 1385–1393.

[8] P. J. Liu *et al.*, "Generating wikipedia by summarizing long sequences," *arXiv:1801.10198 [cs]*, 2018. Available: http://arxiv.org/abs/1801.10198

[9] R. Nallapati, B. Xiang, and B. Zhou, "Sequence-to-sequence rnns for text summarization," *CoRR*, vol. abs/1602.06023, 2016. Available: http://arxiv.org/abs/1602.06023

[10] K. Papineni *et al.*, "Bleu: A method for automatic evaluation of machine translation," in *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ser. ACL '02. Stroudsburg, PA, USA: Association for Computational Linguistics, 2002, pp. 311–318. Available: https://doi.org/10.3115/1073083.1073135

[11] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *CoRR*, vol. abs/1409.3215, 2014. Available: http://arxiv.org/abs/1409.3215

[12] K. Toutanova *et al.*, "A dataset and evaluation metrics for abstractive compression of sentences and short paragraphs," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 340–350. Available: https://www.aclweb.org/anthology/D16-1033

[13] A. Vaswani *et al.*, "Tensor2tensor for neural machine translation," *CoRR*, vol. abs/1803.07416, 2018. Available: http://arxiv.org/abs/1803.07416

[14] A. Vaswani *et al.*, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017. Available: http://arxiv.org/abs/1706.03762

[15] N. Yu *et al.*, "An operation network for abstractive sentence compression," in *Proceedings of the 27th International Conference on Computational Linguistics*. Santa Fe, New Mexico, USA: Association for Computational Linguistics, Aug. 2018, pp. 1065–1076. Available: https://www.aclweb.org/anthology/C18-1091

## 9. Appendix

Figure 8 obtained from [4] illustrates how the the attention weights are learned in the model. The context vector is simply a vector of weights that represents the importance of each input for translating a particular input. For example: To translate "I" to "Je", the model learns to attend to "I" the most, and learns to give high importance to "am" because the translation for "I" in French is either "Je" or "J' " based on the succeeding verb.

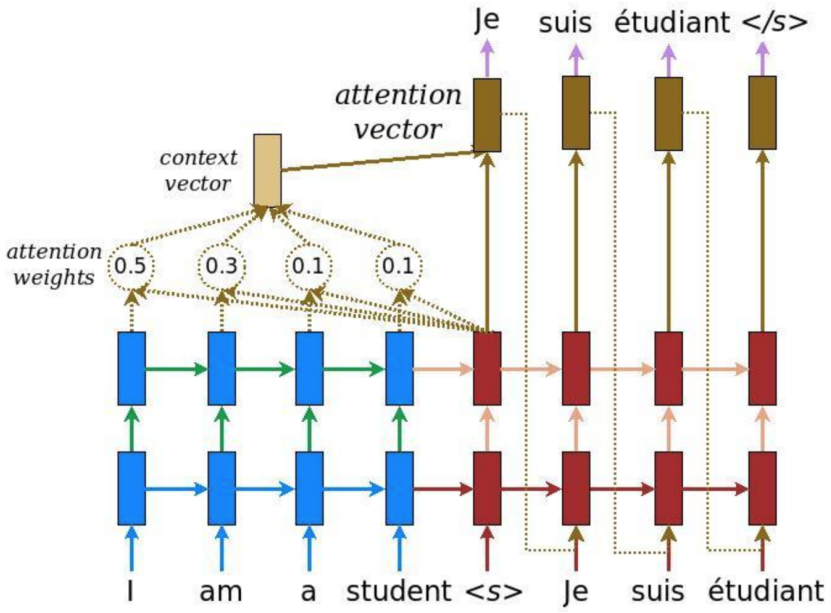


**Figure 8: Illustration of the "Attention" mechanism for Machine Translation tasks**

[4] Source: https://medium.com/syncedreview/a-brief-overview-of-attention-mechanism-13c578ba9129

Figure 9 obtained from [5] shows the three kinds of attention used in the Transformer model. It depicts that the each input attends directly to every other input in the "Encoder Self-Attention"; In the "MaskedDecoder Self-Attention", an input only attends to the inputs that came before it; In the "Encode-Decoder Attention", the decoder attends to each of the encoded inputs.

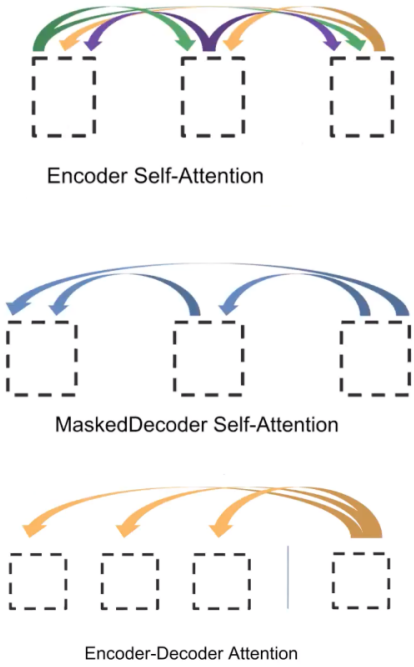


**Figure 9: Dependencies captured in the different types of Attention in the Transformer [14]**

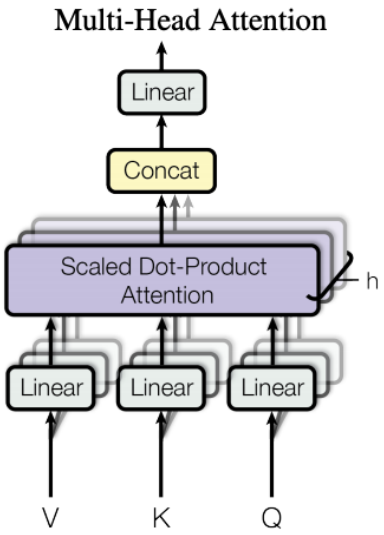


**Figure 10: A flow chart of "Multi-head Attention" from (Vaswani et 2018) [14]**

Figure 11 obtained from [6] showcases the benefits from using multiple attention heads. Using "Multi-head Attention", "it" can be encoded to independently attend to the "the animal", i.e. the noun it replaces, and to "tired", i.e. the state of that animal. Therefore, this mechanism captures distant and finer dependencies better without significant increase in computation costs [14].

Figure 10 obtained from [14] is a simple flow chart to illustrate the mechanism of "Multi-head Attention". The figure makes it easy to observe the different Attention heads learning in parallel.

[5] Source: https://www.youtube.com/watch?v=cS2UZKHq4i4&t=1905s
[6] Source: http://jalammar.github.io/illustrated-transformer/

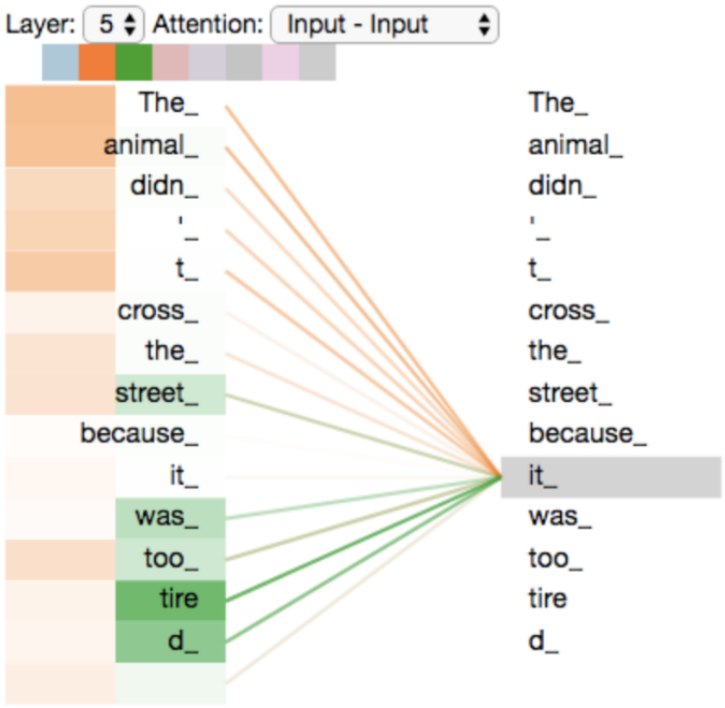


**Figure 11: The benefit of having multiple "Attention" heads**

Figure 12 is the variation of the ROUGE-2 F1score over 125K training steps for the default Transformer model. For initial steps the model hasn't learned much so the score is high due to high Recall as described in Section 6.3. As the model learns, the ROUGE-2 F1 sharply decreases and then slowly increases to a 0.5 and after about 20K iterations it plateaus.

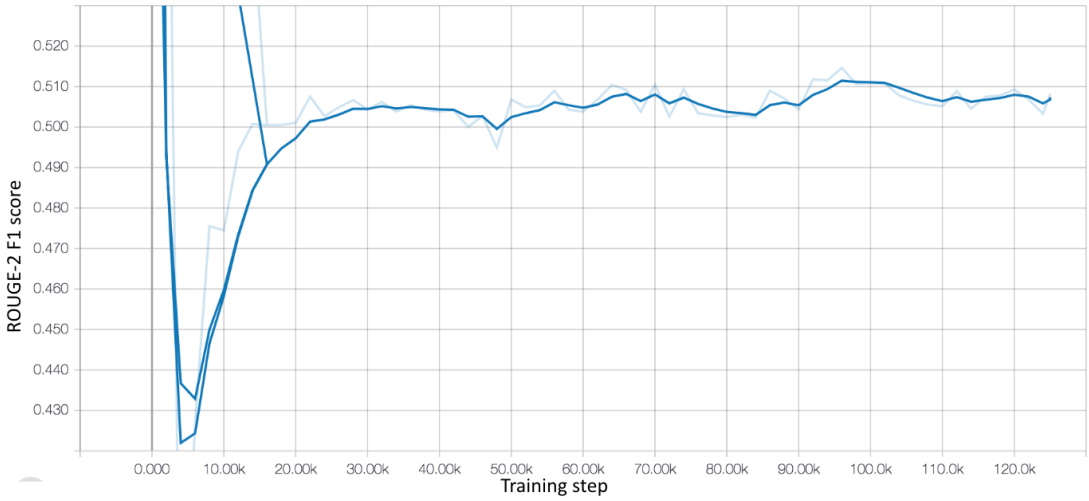


**Figure 12: Variation of ROUGE-2 score with the training time (Default Transformer)**

Because the score plateaus after 20K training steps in Figure 12, the Tuned Transformer is trained for only 20K steps as shown in Figure 13. Due to fewer evaluation checkpoints, this plot is less representative of the variation, but we can still observe the initial dip and then steady rise in the ROUGE-2 F1 score.

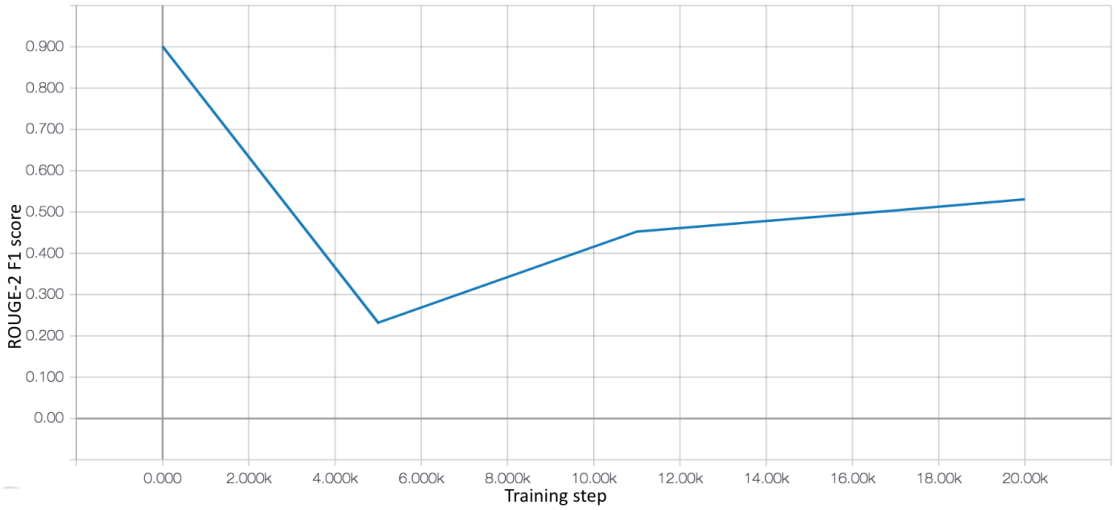


**Figure 13: Variation of ROUGE-2 score with the training time (Tuned Transformer)**