Imperial College London

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

On the Summarization and Evaluation of Long Documents

Author:
Alexander Gaskell

Internal Supervisors:
Dr. Pedro Baiz
Prof. Lucia Specia

External Supervisors:
Hugo Barbaroux
Dr. Eric Topham

Submitted in partial fulfillment of the requirements for the MSc degree in Artificial Intelligence of Imperial College London

Date: September 3, 2020

Abstract

Text summarization has been a key language generation task within Natural Language Processing (NLP) for over 60 years. The field has advanced tremendously during the past two years, benefiting from the proliferation of neural networks across NLP, and in particular, the Transformer. The recent advances have focused primarily on short documents; long documents are the focus of this thesis. We investigate two promising areas: 1) identifying a superior automatic evaluation metric, and 2) creating and experimenting with novel Transformer-based long document summarization models. There are several key findings in this thesis. Regarding the evaluation metrics, we test eight metrics on five experiments spanning three datasets. We develop a novel evaluation metric, `BARTScore`, and find this metric correlates twice as well with human judgement as the prevailing `ROUGE` metrics, and often was the strongest performer of every metric we considered. We establish a set of four evaluation metrics, `BARTScore`, `BERTScore`, `Mover-1` and `Mover-2`, each of which consistently outperform the `ROUGE` metrics. Regarding the novel architectures, we experiment with the `Longformer Encoder Decoder` (`LED`) a Transformer model designed for long documents. We demonstrate state of the art performance on one dataset, beating the incumbent by over one `ROUGE`. We also show that approximate self-attention layers perform comparably to dense self-attention layers for summarization while being much more memory-efficient. These efficiencies allow us to fine-tune summarization models using 2.5x longer sequences on standard hardware. The accompanying code can be found at `https://github.com/alexgaskell10/nlp_summarization`.

Acknowledgements

I am grateful for the valuable help and insight I have received from a number of people affiliated with this project. I would first like to thank my supervisor, Dr. Pedro Baiz, for his guidance and insightful suggestions throughout. These have been invaluable for steering the direction of the project and helping me to always see the bigger picture. I would also like to thank The Data Analysis Bureau, T-DAB, and in particular Dr. Eric Topham and Hugo Barbaroux for your contributions to this project. You have each provided several important ideas which have helped shaped the project to what it currently is. Equally importantly, our frequent catch-ups helped maintain my sanity throughout the long Covid-19 dominated months without any access to the Imperial campus and regular human contact in general.

Table of Contents

1 Introduction

This thesis seeks to contribute to the existing literature on automatic text summarization. This field has recently undergone rapid and substantial progress, driven by the success of sequence to sequence (*seq2seq*) modelling in NLP. However, the research emphasis has focused on developing architectures suitable for short, single documents and has neglected long and multi-document summarization. The ultimate goal for this field is to develop a framework which can produce high quality summaries independent of the source document lengths, whether it is a single or multi-document task and agnostic over domain and whether the lexicon is technical or colloquial.

Given the impressive recent progress seen in short document summarization, the next frontier is to replicate the results using long documents. The focus of this thesis is therefore on advancing the literature in long document summarization. We identify the evaluation of summaries and architectures for long document summarization as two particularly important areas and these are the central themes of this thesis.

Regarding the evaluation metrics, we rigorously test eight evaluation metrics for their correlation with human judgement when scoring summaries, their ability to detect semantic equivalence and their sensitivity to artificial corruption. It will be seen that a set of four novel, model-based evaluation metrics considerably outperform the prevailing evaluation metrics. It will be argued that these are well-placed to become the new primary evaluation metrics for text summarization research. Considering the architectures angle, we experiment using the `LED`, a Transformer Encoder Decoder (TED) well-suited to long document summarization. By facilitating the summarization of longer documents, we will demonstrate that the `LED` performs near or at state of the art levels on long document summarization tasks. The key findings of this thesis are as follows:

- We achieve state of the art text summarization performance on the arXiv dataset, beating the incumbent, `PEGASUS`, by over one `ROUGE`[1]. This is in spite of our modest computational resources
- We develop a novel evaluation metric, `BARTScore`. This correlates approximately 2x better with human judgement than `ROUGE`[2]. Often performed best of all metrics we tested
- We establish a superior set of model-based evaluation metrics, consisting of `BARTScore`, `BERTScore`, `Mover-1` and `Mover-2`. All shown to outperform `ROUGE` on five tasks spanning three datasets
- Demonstrate that sparse self-attention performs comparably to dense self-attention for summarization. This allows 2.5x longer sequences to fit onto standard hardware[3]

This thesis is structured as follows: section 2 provides the essential background for this project, including the architectural building blocks in section 2.3 and an overview of relevant

[1] `PEGASUS` was state of the art at the time of writing. Since then, Zaheer et al. [2020] have released `Big Bird`, an adaptation for `PEGASUS` for long document tasks. This has set a new state of the art on the arXiv dataset and our results do not beat these.
[2] See section 6.1.2.
[3] See section 6.2.2.

approaches in section 2.5. We outline the datasets used in this study in section 3 followed by the design details of our evaluation metrics and model architectures in section 4. We motivate and outline our experimental methodology in section 5, followed by corresponding results in section 6. We provide some additional analysis in section 7 before offering some concluding thoughts in section 8. The accompanying code can be found at `https://github.com/alexgaskell10/nlp_summarization`.

1.1 Ethical, Legal and Environmental Considerations

GDPR We have included the Ethics Checklist as provided by the Imperial Department of Computing in tables F.1 and F.2. "Section 4: Protection of Personal Data" is most pertinent to this thesis, specifically the use of the CNN/DailyMail dataset (See et al. [2017]) as this contains articles published on individuals including celebrities and politicians. According to Art.(4)(1) and Art.(4)(2) of the General Data Protection Regulation (GDPR)[4], by storing and training our models on this data we are "processing" the "personal data" of "data subjects". The CNN/DailyMail dataset contains articles reporting on the personal data of data subject including celebrities and politicians. Given that these contains information on data subjects' political beliefs and sexual orientation, this data qualifies as special category data under Art.(9)(1) of the GDPR, and is thereby prohibited unless one of the exemptions listed in Art.(9)(2) apply. As we are conducting scientific research and we assess that this "processing is necessary for archiving purposes in the public interest, scientific or historical research purposes or statistical purposes", the exemption from Art.(9)(2.j) applies. We therefore believe we are in compliance with the GDPR when processing the special category data within the CNN/DailyMail dataset in this thesis.

Plagiarism Looking beyond the Ethics Checklist, there is a risk that users of our code could unknowingly commit plagiarism. Summarizing a document without providing the source is plagiarism[5], therefore any content generated using a summarization model must cite the author of the original article appropriately. For example, using our model to write a blog post including summaries of the latest research in a field would be plagiarism if there is no citation. This fact is little-known and could easily catch users of our code unaware.

Environmental footprint The leading models in text summarization and NLP have become larger and increasingly data-hungry since the development of the Transformer. The NLP community is becoming more cognisant of the environmental impacts of training these models, which frequently require days of energy-intensive GPUs for a single training cycle (Strubell et al. [2019]). Using the Weights & Biases compute tracker, we estimate that we have used approximately 300 days of GPU compute for our experiments in this thesis. Using the MLCO2 machine learning emissions calculator[6], this equates to over 1.5 tonnes of CO_2 emissions. This is similar to the emissions cost of a flight to New York from London[7]. We did not conduct any pre-training of models in this thesis conscious of the cost of doing so (both economically and environmentally). Despite this, we have still been responsible for considerable CO_2 emissions and this is becoming an increasingly problematic issue for the NLP community.

[4]`https://gdpr-info.eu/art-4-gdpr/`

[5]See `https://academicintegrity.uoguelph.ca/plagiarism/paraphrasing-and-summarizing`.

[6]`https://mlco2.github.io/impact/#compute`

[7]`https://www.carbonfootprint.com/calculator.aspx`

2 Background

2.1 Overview of Text Summarization

Summaries play a key role in communicating written information. Defined as a document reduced only to its essential content, a summary helps readers to understand material more easily and quickly. In a digital world with ever-increasing volumes of writing available online, summaries are playing a central role in synthesising information into a digestible format for readers. For example, between the onset of Covid-19 in January and early May 2020 there were an estimated 23,000 research papers published on the virus with the number doubling every week[8]. For medical researchers to distil useful information from this volume of publications a comprehensive toolkit aiding with information processing and condensing is a necessity; an automatic summarizer could become a central component of such a toolkit.

Summarization is interesting from a research perspective for several reasons: first, it mas myriad academic and commercial applications. Consider research literature surveys, summaries of the outcomes of legal verdicts or legislative bills in the legal sector, daily market summaries in the finance sector or general message reading for enterprise workers (which is estimated at costing 2.6 hours per day[9]); a tool able to perform human-quality summarization but with digital speed would be valuable. Second, it provides a challenging extension to other *seq2seq* natural language tasks. Viewed alongside machine translation for example, text summarization inherits the difficult task of producing a coherent and informatically accurate output, with the additional challenges of distilling the salient points. The asymmetric lengths of inputs and outputs in this task also poses engineering problems owing to model capacity issues and these will be a focus of this study. Finally, evaluating text summarization is an interesting sub-problem due to the intra-observer variability of summaries, meaning there potentially several valid summaries for each source document. Taken together, these reasons justify the substantial research attention that has recently been paid to automatic summarization and explain why it was considered one of the top AI research priorities for 2020[10].

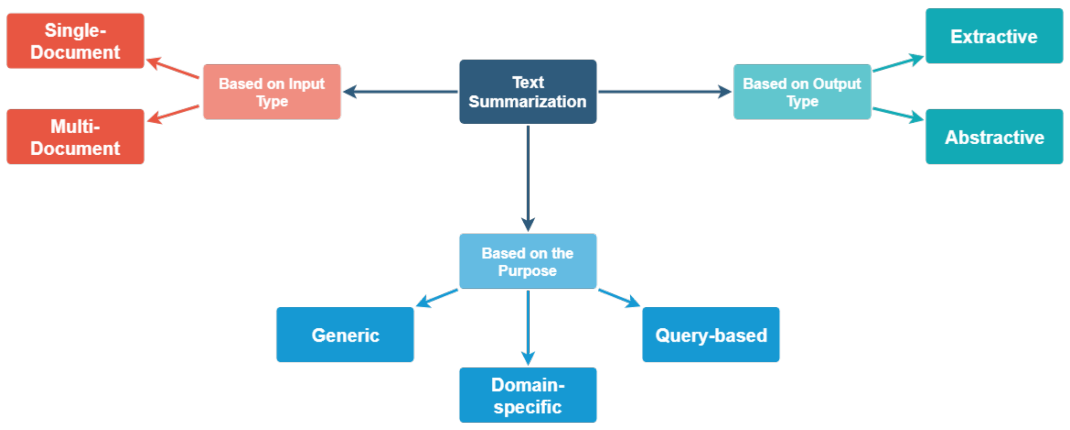


Figure 2.1: Illustration of the sub-fields within text summarization. Source: Chauhan [2018].

[8]Source: ScienceMag.org `https://tinyurl.com/ybmmdjkl`. Visited on 28/05/2020.
[9]Source: Harvard Business Review `https://tinyurl.com/y89fxw6r`. Visited on 28/05/2020.
[10]`https://blog.deeplearning.ai/blog/the-batch-happy-new-year-hopes-for-ai-in-2020-yann-lecun-kai-fu-lee-anima-anandkumar-richard-socher`

As shown in Figure 2.1, the field of text summarization can be split based on input document type, output type and purpose. Regarding output type, text summarization dissects into extractive and abstractive methods. Extractive forms the output summary by selecting and arranging sentences from the source document, while abstractive generates new content for the output summary. As the more desirable approach with greater potential, research interested has switched from initially focusing on extractive to abstractive methods in recent years, and this study will follow this lead. The following section provides some historical context to text summarization, leading into the technical building blocks and successful implementations which comprise the remainder of this section.

2.2 A Brief History of Text Summarization

Research into automatic text summarization dates back to over 60 years ago (Saggion and Poibeau [2013]). Automatic summarization first caught the scientific community's attention in the 1950s, focusing on technical domains (Luhn [1958]). The AI community began showing interest in the 1980s (Lehnert and Ringle [1982]) believing that this task was an challenging test of AI systems' natural language capabilities. Several summarization-focused conferences were established in the 2000s, such as the Document Understanding Conferences and Text Analysis Conferences (Over et al. [2007], Ji and Grishman [2011]), contributing to further research interest.

During this period, the leading contemporary approaches divided into statistical and knowledge-based (Saggion and Poibeau [2013]). Statistical approaches looked at extractive summarization as a binary classification problem for each sentence in the source document; i.e. should a sentence appear in the output summary or not. Features were extracted from each sentence and fed into a classifier on this basis. One such approach used rules to highlight salient excerpts within the text, such as the title or sentences containing specific cue-words such as "in conclusion" or "to summarize". Features were then extracted by comparing each sentence in the document to these flagged passages and these were fed into a classifier (Edmundson [1969]).

In contrast, knowledge-based approaches attempted to incorporate semantic knowledge and sophisticated lexical resources within the summarization process. One prominent approach was the Fast Reading and Understanding Memory Program (FRUMP, Lehnert and Ringle [1982]), used for automatically summarizing news articles. FRUMP used rich data structures called *scripts* which provided a detailed description of a generic event occurring over time. The summaries were produced using a top-down approach by mapping news articles to these structures (Copeland [2015]). However, these approaches required the costly manual encoding of world knowledge into the model, preventing their widespread adoption.

Until recently, extractive summarization took precedence over abstractive because of its greater simplicity. This changed in 2014 with the advent of *seq2seq* neural methods which rendered abstractive summarization viable (see section 2.3.1). Since this time, both extractive and abstractive methods have undergone rapid and consistent progress, driven initially by the use of Recurrent Neural Networks (RNNs). The pace of progress accelerated after the development of the Transformer (Vaswani et al. [2017]) and this architecture has come to dominate state-of-the-art approaches. These architectures will be explained in section 2.3, followed by some of the leading relevant methods in section 2.5.

2.3 Building Blocks

2.3.1 Sequence to Sequence

Deep Neural Networks (DNNs) have risen to prominence on the back of impressive performance across a wide range of domains (e.g. Krizhevsky et al. [2012], Devlin et al. [2018], He et al. [2015]). However, DNNs are normally restricted to problem formulations where the input and output dimensionality can be specified a-priori. This proved an early impediment to their widespread adoption for natural language tasks as many of these are sequential by nature; letters make up words, which make up sentences, which make up documents. Such tasks include machine translation, question answering, speech recognition and text summarization.

The development of *seq2seq* models (Sutskever et al. [2014]) was a landmark moment. Early approaches used an RNN, (Rumelhart et al. [1986]) to encode the intput sequence into a single vectorial representation, and then used a separate RNN to extract the target sequence from this vector. The canonical *seq2seq* problem is formalized below, following the lectures from Specia et al. [2020]:

$$\text{Source sequence:} \quad \mathbf{X} = \{\mathbf{x_1}, \ldots, \mathbf{x_S}\} \tag{2.1}$$
$$\text{Target sequence:} \quad \mathbf{Y} = \{\mathbf{y_1}, \ldots, \mathbf{y_T}\} \tag{2.2}$$
$$\text{Encoder:} \quad E() \tag{2.3}$$
$$\text{Decoder:} \quad D() \tag{2.4}$$
$$\text{Training corpus:} \quad C = \{(\mathbf{X^i}, \mathbf{Y^i})\}_{i=1}^{N} \tag{2.5}$$

Optimization procedure The standard training procedure for *seq2seq* models uses maximum likelihood. This means that we are trying to find a set of model parameters θ which maximise the likelihood function $\mathcal{L}(\mathbf{X}, \mathbf{Y})$:

$$\theta^* = \text{argmax}_\theta\, \mathcal{L}(\mathbf{X}, \mathbf{Y}) = \text{argmax}_\theta \prod_{\mathbf{i=1}}^{\mathbf{N}} \mathrm{P}(\mathbf{X^i}, \mathbf{Y^i}) \tag{2.6}$$
$$= \text{argmax}_\theta \sum_{\mathbf{i=1}}^{\mathbf{N}} \log \mathrm{P}(\mathbf{X^i}, \mathbf{Y^i}) \tag{2.7}$$

This makes the standard assumption that the training data are independent and identically distributed (IID), followed by taking `logs`. Equation 2.7 is the standard supervised learning objective function, saying that the optimal set of parameters are those which make the model best fit the training data. Adapting this for *seq2seq* problems:

$$\theta^* = \text{argmax}_\theta \sum_{\mathbf{i=1}}^{\mathbf{N}} \sum_{\mathbf{t=1}}^{\mathbf{T}} \log \mathrm{P}(\mathbf{y_t^i} | \mathbf{y_{<t}^i}, \mathbf{X^i}) \tag{2.8}$$

This equation adapts Equation 2.7 to conditional language generation problems, whereby the objective is to generate a word conditioning on a set of words. This equation can be read as using the entire input sequence $\mathbf{X^i}$ and all previous ground-truth tokens $\{\mathbf{y_0^i}, \mathbf{y_1^i}, \ldots, \mathbf{y_{t-1}^i}\}$

to maximise the probability of predicting token $\mathbf{y_t}$. $P(\mathbf{y_t^i}|\mathbf{y_{<t}^i}, \mathbf{X^i})$ is a distribution over the vocabulary, and is the conditional distribution we are trying to model. Assuming the input sequence X is encoded into a vectorial representation $\mathbf{v}$ then Equation 2.7 becomes:

$$\theta^* = \text{argmax}_\theta \sum_{\mathbf{i=1}}^{\mathbf{N}} \sum_{\mathbf{t=1}}^{\mathbf{T}} \log P(\mathbf{y_t^i}|\mathbf{y_{<t}^i}, \mathbf{v^i}) \tag{2.9}$$

The model is trained end-to-end using source-target sequence pairs until convergence is reached.

From Equation 2.9 we can see an issue with this framework: the model is trained by predicting a token conditioned on the previous tokens in the sequence, known as *teacher-forcing* (Williams and Zipser [1989]). At inference-time however, the ground-truth is not available so the previously sampled tokens are fed back in instead. This makes the model auto-regressive (Graves [2013]) at inference time but not training time. As the model conditions on its previously generated token, the model samples from the joint distribution over the sequence during inference. This creates *exposure bias* (Keneshloo et al. [2018]) and can lead to poor model performance as the conditioning context at inference time diverges from that seen at training time (Lamb et al. [2016]).

2.3.2 Recurrent Neural Networks

The first generation of *seq2seq* models used RNN-based architectures. This section provides a general overview of the RNN, some powerful extensions and their applicability to *seq2seq* modelling[11].

Overview of RNNs Feedforward DNNs are powerful function approximators and can learn useful representations in high dimensions. However, one drawback is that they have no form of memory so inputs are processed independently at each time step. This is problematic for sequential tasks such as text summarization, as the network would not be able to relate what it learned in the introduction to the methodology section, for example. The RNN is a variant of the DNN with loops so the output from a neuron can be fed into a neuron in the same or previous layers, permitting memory through information persistence.

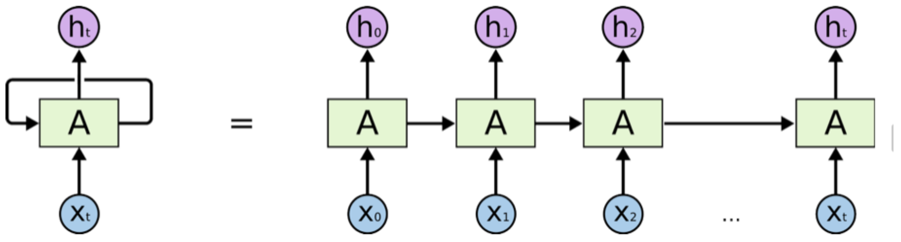


Figure 2.2: A vanilla RNN. Source: Olah [2015].

Figure 2.2 shows a vanilla RNN. **A** is the network, which takes some input X and outputs a *hidden state*. As the input is sequential, this occurs for each *time step, t*. The loop

[11]This explanation follows Olah [2015].

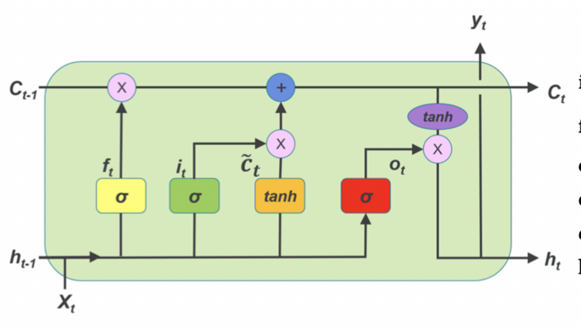
means that the hidden state from the previous time step, $\mathbf{h_{t-1}}$, is also used as an input for the following step. Conceptually, an RNN can be envisaged as copies of a single feedforward network with each network able to pass a message to the next. This loop enables memory within the network as inputs early in the sequence (e.g. $\mathbf{x_0}$) can persist through the hidden state and influence the outputs later in the sequence (shown as $\mathbf{h_t}$). The recurrence formula is:

$$\mathbf{h_t} = \tanh(\mathbf{W_h}\mathbf{h_{t-1}} + \mathbf{W_x}\mathbf{x_t} + \mathbf{b}) \tag{2.10}$$

As shown, the hidden state is computed by projecting the previous and current hidden states through a `tanh` activation function. The projection matrices $\mathbf{W_h}$ and $\mathbf{W_x}$ contain the parameters learned by the network. The network is then trained using Back-Propagation Through Time, BPTT (Werbos [1988]), a generalisation of Back-Propagation to cover sequential networks.

LSTM The vanilla RNN described above is useful for situations where sequences are short. However, this simple architecture struggles to learn long distance dependencies (Hochreiter et al. [2001]). One reason is that RNNs suffer from the vanishing and exploding gradients problem (Bengio et al. [1994]) resulting from repeated matrix multiplication of the hidden state. This hinders BPTT for words early in the sequence.

The LSTM (Hochreiter and Schmidhuber [1997]) addresses this issue. The core idea behind the LSTM is the use of *gating* to regulate the information flow, giving the cell control over what information is retained or forgotten. This requires a more sophisticated unit, shown in Figure 2.3.



$$\begin{aligned}
\mathbf{i}_t &= \sigma\left(\mathbf{W}_{xi}\mathbf{x}_t + \mathbf{W}_{hi}\mathbf{h}_{t-1} + \mathbf{b}_i\right)\\
\mathbf{f}_t &= \sigma\left(\mathbf{W}_{xf}\mathbf{x}_t + \mathbf{W}_{hf}\mathbf{h}_{t-1} + \mathbf{b}_f\right)\\
\mathbf{o}_t &= \sigma\left(\mathbf{W}_{xo}\mathbf{x}_t + \mathbf{W}_{ho}\mathbf{h}_{t-1} + \mathbf{b}_o\right)\\
\tilde{\mathbf{c}}_\mathrm{t} &= \tanh\left(\mathbf{W}_{xc}\mathbf{x}_t + \mathbf{W}_{hc}\mathbf{h}_{t-1} + \mathbf{b}_c\right)\\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_\mathrm{t}\\
\mathbf{h}_t &= \tanh\left(\mathbf{o}_t \odot \mathbf{c}_t\right)
\end{aligned}$$

Figure 2.3: The LSTM with its equation system. Source: Ismail et al. [2018].

The LSTM outputs a cell state, $\mathbf{c_t}$, in addition to the hidden state. The cell state has few interactions and makes information retention simple. Gating provides the means for adding or removing information from the cell state. These are the *input*, *forget* and *output* gates, $\mathbf{i_t}$, $\mathbf{f_t}$ and $\mathbf{o_t}$ respectively. The forget gate first determines how much information is removed from the cell state, $\mathbf{c_{t-1}}$. The input gate then combines information from $\mathbf{x_t}$ and $\mathbf{h_{t-1}}$ into "new candidate information" $\tilde{\mathbf{c}}_\mathbf{t}$ and determines how much should be written to the cell state. This addresses vanishing/exploding gradients as the repeating module now contains additive and multiplicative operations, rather than exclusively multiplicative operations as for the vanilla RNN. The output gate determines how much of the new cell state $\mathbf{c_t}$ should be outputted as the new hidden state $\mathbf{h_t}$. Comparing Figure 2.3 to Equation 2.10 we see that the

LSTM has quadruple the number of parameters as the vanilla RNN (two projection matrices per gate), and therefore is able to model more complex, longer-distance dependencies.

Other Variants Myriad variants of the RNN exist. This section explores *bidirectional RNNs* and *deep RNNs* as two powerful examples. One drawback of the RNN above is that context can only be included from left to right. Considering summarization for example, it might be useful to refer back to the introduction after reviewing the experimental results. Bidirectional RNNs, BRNN (Schuster and Paliwal [1997]), address this by using two RNN units and feeding the sequence forwards through one unit and backwards through the other, as illustrated in Figure 2.4. This modification tends to improve performance for tasks which benefit from using context from both sides.

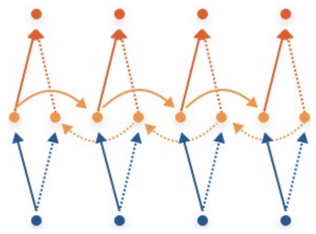


Figure 2.4: A bidirectional RNN. Source: missinglink.ai[12].

Deep RNNs (Tutschku [1995]) are another modification. These are formed by stacking multiple RNN units vertically so that the hidden state from one layer is the input for the subsequent layer, analogous to stacking layers in a multi-layer perceptron. Each layer adds another RNN unit, increasing the learnable parameters and power of the network. These modifications can be combined to create a deep BRNN as shown in Figure 2.5. When used in conjunction with powerful LSTM units, these variants can make large recurrent models able to learn complex functions.

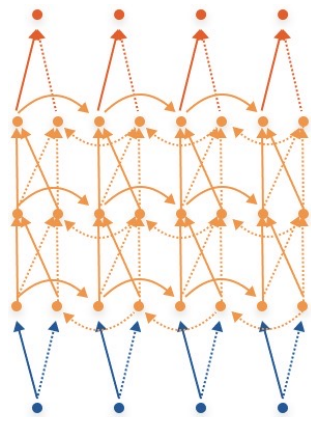


Figure 2.5: A deep, bidirectional RNN. Source: missinglink.ai[13].

RNNs for seq2seq The *seq2seq* paradigm began with Sutskever et al. [2014]. Their encoder, a four-layer uni-directional LSTM, reads the input sequentially to form a single vecto-

[12]`https://tinyurl.com/ycg6rlsz`. Visited on 23/05/2020.
[13]`https://tinyurl.com/ycg6rlsz`. Visited on 23/05/2020.

rial representation encoding. The decoder (also a four-layer uni-directional LSTM) extracts the output sequence by conditioning on the hidden state and the previous ground truth tokens.

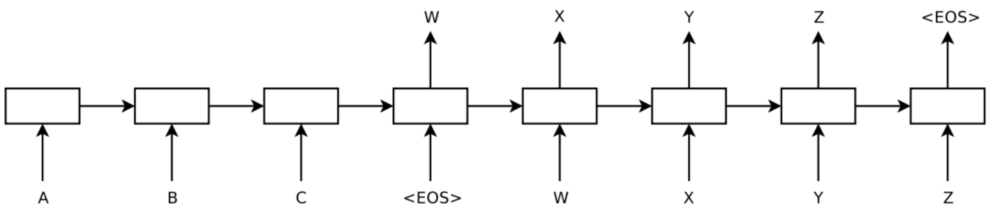


Figure 2.6: An example of using RNNs for *seq2seq* modelling. Source: Sutskever et al. [2014].

Figure 2.6 illustrates the encoder decoder structure: the input sequence, consisting of **{A, B, C}**, is encoded sequentially. The first decoder step is conditioned on the encoded vector (and a token "<**EOS** >" designating the beginning of the decoding sequence), and each subsequent decoding step conditions on the decoder hidden state and the previously generated token. The output $P(\mathbf{y_t}|\mathbf{y_{<t}}, \mathbf{v})$ as shown in Equation 2.9 is a `softmax` over the possible output tokens (the vocabulary for language tasks) and the token with the highest probability is selected. This continues until the decoder produces the "<**EOS** >" token, indicating the end of the sequence. This structure can therefore handle inputs and outputs of arbitrary lengths, making it suitable for sequential tasks.

2.3.3 Attention

In section 2.3.2 we described several flavors of RNNs and how these models could be used for *seq2seq* tasks. In this section we outline attention and how it aids RNNs for *seq2seq* modelling.

Issues with RNNs for seq2seq Despite the enhancements highlighted in section 2.3.2, the RNN still faced several challenges when applied to *seq2seq* tasks. First, the input sequence must be encoded to a single vector of fixed dimension irrespective of the input size. This creates a bottleneck in the encoder output. Second, this structure struggles learning long distance dependencies as gradients must flow through every hidden state, creating vanishing or exploding gradients again. This mean that the encoder is often under-trained for words early in the source sequence.

The Attention Mechanism Attention offers a solution to both of these issues. The idea behind attention, shown in Figure 2.7, is to allow the decoder to focus on some encoded words more than others during the decoding process. The attention mechanism computes a dynamic context vector $\mathbf{c_t}$ using all encoder hidden states, $H = \{\mathbf{h_0}, \mathbf{h_1}, ..., \mathbf{h_s}\}$ and the current decoder hidden state, $\mathbf{d_t}$, at each decoder time step.

The first stage of the attention mechanism computes a similarity score s_i between the decoder hidden state and each encoder hidden state. The two main variants for computing s_i are *MLP attention* (Bahdanau et al. [2014]) and *Dot attention* (Luong et al. [2015]), described as follows:

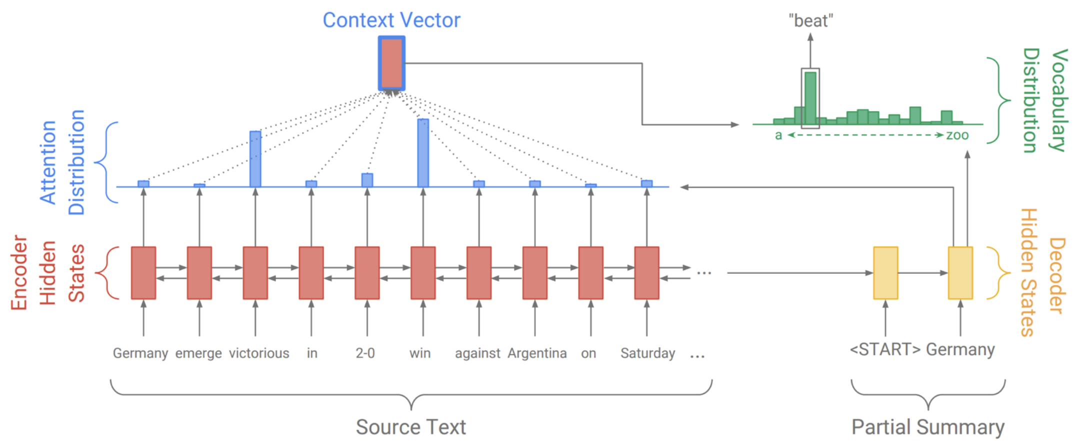


Figure 2.7: An illustration of an RNN with attention. Here the model is at the second decoding time step. Source: See et al. [2017]

$$\text{Dot attention:} \quad s_i = \mathbf{h_i^T d_t} \tag{2.11}$$

$$\text{MLP attention:} \quad s_i = \mathbf{a}^T \tanh(\mathbf{W_d d_t} + \mathbf{W_h h_i}) \tag{2.12}$$

The main difference is that MLP attention learns a vector of weights a^T while dot attention only takes dot products. As dot attention is more popular, we will focus on this hereon in. The s_i's are normalised using a `softmax`, creating a set of weights:

$$\{\alpha_0, \alpha_1, ..., \alpha_s\} = \text{softmax}\{s_0, s_1, ..., s_s\} \tag{2.13}$$

The context vector c_t is then computed as an average of the encoder states weighted by the `softmax` scores:

$$\mathbf{c_t} = \sum_{i=1}^{S} \alpha_i \mathbf{h_i} \tag{2.14}$$

The context vector is concatenated with the decoder hidden state $\mathbf{d_t}$ to produce the attentional hidden state, $\mathbf{\tilde{h}_t}$ (Luong et al. [2015]). This vector is projected through a `softmax` layer to produce the output distribution over the vocabulary:

$$\text{p}(\mathbf{y_t}|\mathbf{y_{<t}}, \mathbf{v}) = \text{softmax}(\mathbf{W_s \tilde{h}_t}) \tag{2.15}$$

Using attention, the decoder sees a different encoded representation at each time step as $\mathbf{c_t}$ is dynamic. This tackles the bottleneck issue, increasing the capacity of the network for longer sequences. In addition, gradient flow through the network is improved by providing a more direct route for gradients to flow from the decoder to the encoder via the context vector. This can be seen in Figure 2.7: gradients can flow from the output vocabulary distribution to each of the encoder hidden states via the attention computation. This alleviates exploding/vanishing gradients as this path requires fewer multiplicative operations, also helping the network to learn longer distance dependencies. This demonstrates clear advantages for using attention with RNNs for *seq2seq* tasks.

2.3.4 Transformers

The next stage in the evolution of *seq2seq* modelling came with the development of the Transformer (Vaswani et al. [2017]). RNNs are sequential models by nature as they process the input one token at a time, limiting their capacity to be parallelized. The Transformer is designed to be easily parallelized and suitable for pre-training on large datasets by discarding recurrence and primarily using attention. Transformer-based architectures have come to dominate the state-of-the-art approaches in *seq2seq* tasks including text summarization. This section explains the TED following Rush [2018].

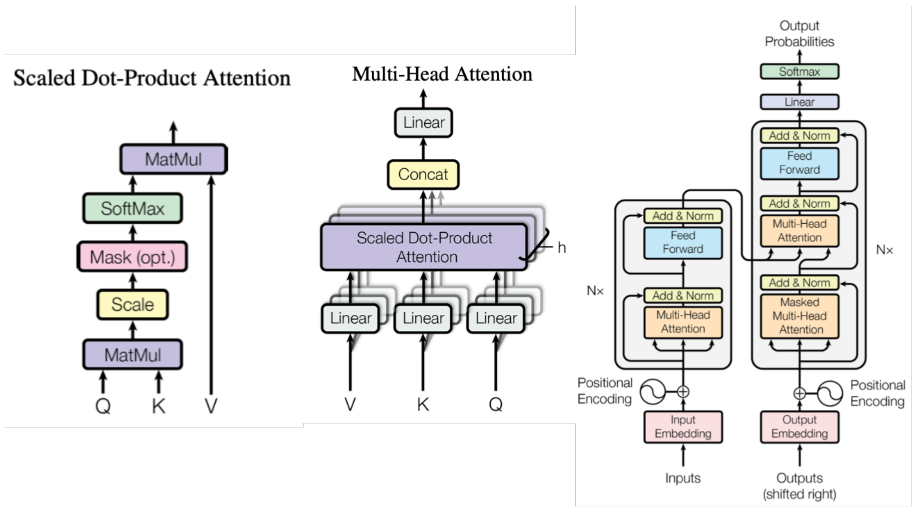


Figure 2.8: Schematic of a TED showing (left to right): self-attention, multi-head attention and the overall architecture. Source: Vaswani et al. [2017]

Self-attention The key component within the Transformer is self-attention. This allows the network to use context by relating each token to every other token in the sequence. Formally, self-attention uses a *query* $\mathbf{Q}$ and a set of *key-value pairs*, $\{\mathbf{K}, \mathbf{V}\}$. The output is a weighted sum of the values, where the weights are computed using a *compatibility-function* of the query with the keys. The *compatibility-function* used in Vaswani et al. [2017] was *Scaled Dot-Product Attention*. This is illustrated in Figure 2.8 and computed as follows[14]:

$$\text{Attention}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^{\mathbf{T}}}{\sqrt{\mathrm{d_k}}}\right)\mathbf{V} \tag{2.16}$$

Multi-Head Attention The above computation comprises a single head; multi-head attention computes *h* versions of this in parallel and concatenates and projects the result. This allows the network to learn different representations where each input attends differing amounts to other tokens in the sequence. This computation is show below, with $\mathbf{W_i^Q}, \mathbf{W_i^K}, \mathbf{W_i^V}$ and $\mathbf{W^O}$ being the learnable weight matrices.

$$\text{MultiHeadAttn}(\mathbf{Q},\mathbf{V},\mathbf{K}) = \text{Concat}(\text{head}_0, \text{head}_1, \ldots, \text{head}_\mathrm{h})\mathbf{W^O} \tag{2.17}$$

[14]This formula divides by $\sqrt{d_k}$, the dimensionality of the keys/query, as otherwise the variance of the dot product would increase linearly with the dimensionality, pushing the `softmax` into regions with negligible gradients.

$$\text{with head}_\text{i} = \text{Attn}(\mathbf{QW_i^Q}, \mathbf{KW_i^K}, \mathbf{VW_i^V}) \tag{2.18}$$

Transformer Architecture The complete TED architecture is illustrated in Figure 2.8. The left half is the encoder and the right half is the decoder. The encoder consists of N (six in the paper) stacked blocks, each block containing a multi-head attention layer (eight heads) and a feedforward layer. The feedforward layer comprises of two projections with a ReLU activation function nested between. Each layer uses residual connections and are followed by a normalisation layer. Positional encodings are added to the input embeddings to capture the order of words in the sequence as attention is order-agnostic.

The decoder stack is equivalent to the encoder stack except with an additional multi-head cross attention layer inserted between the first multi-head attention layer and the MLP layer. This allows the decoder to attend to the encoder output vector. The decoder stack output is projected through a `softmax` layer to generate the distribution over the vocabulary, $p(\mathbf{y_t}|\mathbf{y_{<t}}, \mathbf{v})$. As for *seq2seq* RNNs, the decoder is auto-regressive as it consumes its previously generated tokens as inputs for generating subsequent tokens. To maintain the auto-regressive property, future tokens must be masked from the decoder. This is done by setting the values of illegal positions within the decoder to $-\infty$, preventing the decoder from drawing on information from later positions in the sequence.

Complexity A primary motivation for the Transformer was to create a non-sequential architecture suitable for *seq2seq*, enabling greater parallelization during training. Moving from a recurrent network to an attention-based network has implications for computation complexity, with the cost of a single layer of each being $O(nd^2)$ and $O(n^2d)$ respectively, with sequence length n and embedding dimensionality d (Vaswani et al. [2017]). For the majority of *seq2seq* tasks, $d > n$, hence Transformers are a more efficient option. However, we are interested in long sequences, where $n > d$[15]. This proves a bottleneck as the complexity scales quadratically, meaning the memory requirement quickly becomes too large to train models on current GPUs. Section 2.5 presents some potential workarounds to this problem.

2.4 Evaluation

Performing evaluation for *seq2seq* tasks is challenging given there is no ground truth. For text summarization, there may exist two equally good summaries for a single document which focus on different content and are lexically diverse. This section outlines the `ROUGE` package, discusses its flaws and proposes a set of alternative metrics based on contextualised word embeddings.

2.4.1 ROUGE

The Recall-Oriented Understudy for Gisting Evaluation (Lin [2004]), `ROUGE`, is the most widespread summarization evaluation metric. Despite its shortcomings (to be discussed), it is the yardstick decreeing which approaches are considered state-of-the-art. `ROUGE` performs evaluation by comparing the candidate summary to a set of human-produced reference summaries, specifically by computing the co-occurrences of n-grams between the candidate and each reference. `ROUGE` is a package containing several metrics; for this study we report

[15] By convention, d is normally not larger than 1024. In later chapters we will test the `LED-4096` which uses inputs with $n = 4,096$.

ROUGE-1, ROUGE-2 (both instances of ROUGE-N) and ROUGE-L as these are most prevalent in the literature.

ROUGE-N computes the recall of n-grams between an output text and a set of reference texts as follows:

$$\texttt{ROUGE-N} = \frac{\sum\limits_{S \in \{References\}} \sum\limits_{gram_n \in S} \mathrm{Count}_{\mathrm{match}}(gram_n)}{\sum\limits_{S \in \{References\}} \sum\limits_{gram_n \in S} \mathrm{Count}(gram_n)} \tag{2.19}$$

Here, n is the length of the n-gram and $\mathrm{Count}_{\mathrm{match}}(gram_n)$ is the number of n-grams which co-occur in the candidate text and the set of references. The denominator counts the number of n-grams available on the reference side, hence this metric measures recall. This is closely related to the BLEU (Papineni et al. [2002]) metric for evaluating machine translation quality, the main difference being that BLEU is a precision metric so counts n-grams on the candidate side in the denominator[16].

We also consider ROUGE-L, measuring the longest common subsequence[17] between two texts. This measure has two advantages in that it does not require words to be consecutive provided they occur sequentially, meaning it is more robust to meaning-invariant lexical permutations. Also, it automatically includes the longest sequence length so does not require explicitly stating the n-gram length as for ROUGE-N. The metric is an F-measure computed as follows:

$$R_{lcs} = \frac{LCS(X,Y)}{m} \tag{2.20}$$

$$P_{lcs} = \frac{LCS(X,Y)}{n} \tag{2.21}$$

$$F_{lcs} = \frac{(1+\beta^2)R_{lcs}P_{lcs}}{R_{lcs} + \beta^2 P_{lcs}} \tag{2.22}$$

Here m and n are the lengths of texts X and Y respectively, P_{lcs} is the precision measure, R_{lcs} is the recall measure and β specifies the weighting (usually β is set very large meaning that only R_{lcs} is considered).

Shortcomings of ROUGE ROUGE has contributed substantially towards the field of text summarization by enabling benchmarking and comparison of models. However, it has a number of shortcomings. As explained earlier, ROUGE is designed to compare a candidate against a set of reference summaries; in practise, rarely do multiple summaries exist for a single text. ROUGE correlates well with human judgement when averaged over a set of references (Louis and Nenkova [2013]) but performs poorly with a single reference, so much so that it struggles to distinguish between good and bad summaries for the same source document (Böhm et al. [2019]). As ROUGE is computed using n-gram statistics, it only performs a surface-level comparison and penalises lexical and compositional diversity even if the output

[16]As precision-based metrics favour shorter translations, BLEU adds a *brevity penalty* to penalise short candidates.

[17]Formally, a sequence $Z = [z_0, z_1, ..., z_n]$ is a subsequence of $Y = [y_0, y_1, ..., y_n]$ if there is a strictly increasing sequence indices $I = [i_0, i_1, ..., i_j]$ such that for all $i_k \in I$, $y_{i_k} = z_k$ (Cormen et al. [2001]).

is semantically analogous to the reference. With this in mind, this study interprets ROUGE as a *necessary but not sufficient* condition; high scores do not necessarily mean the model is producing strong summaries but low scores are a red flag.

2.4.2 Human Evaluation

Human evaluation is viewed as the gold standard in text summarization. Small-scale experiments using human evaluators are often used to supplement the reporting of ROUGE scores in the literature (Zhang et al. [2019a], Böhm et al. [2019], Yoon et al. [2020]), as experimental evidence comparing a model to its peers using human assessors is the most conclusive evidence as to the performance of a model. However, these experiments are costly and impractical, especially on longer and technical datasets such as arXiv and PubMed as used in this study (see section 3). Furthermore, because of budget limitations, experiments are normally small-scale and compare to only a subset of the rival architectures. This makes it difficult to perform benchmarking of different models using only human evaluations.

2.4.3 Model-Based Metrics

One drawback of ROUGE is it cannot account for lexical diversity between texts as it is based on n-gram overlaps. Some recent approaches have investigated using word embeddings as a means to analyse the semantic similarity of two texts (Zhao et al. [2019]). This section highlights several examples.

BERTScore BERTScore uses BERT (Devlin et al. [2018]) to perform the automatic evaluation of two texts by comparing the weighted cosine similarities of their embedded representations. Figure 2.9 outlines the computation of the recall metric, $\mathrm{R}_{\mathrm{BERT}}$. The first stage is to feed the candidate $\hat{x} = \{\hat{x}_1, ..., \hat{x}_k\}$ and reference summary $x = \{x_1, ..., x_m\}$ through BERT[18] to obtain the embedded sequences $\hat{\mathbf{x}} = \{\hat{\mathbf{x}}_1, ..., \hat{\mathbf{x}}_k\}$ and $\mathbf{x} = \{\mathbf{x}_1, ..., \mathbf{x}_m\}$ respectively.

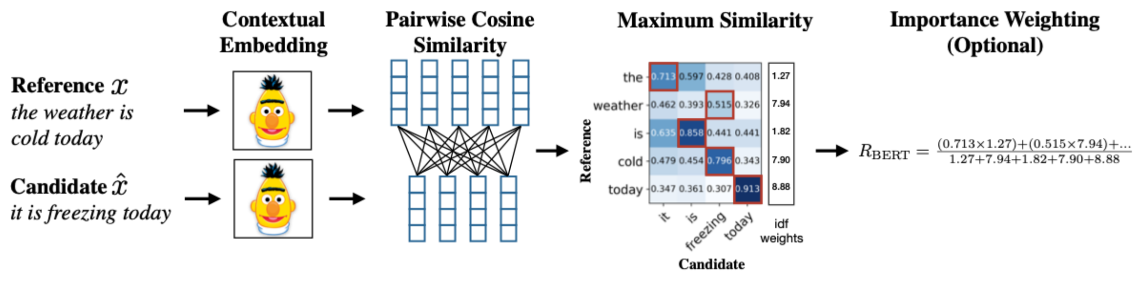

Contextual Embedding
Pairwise Cosine Similarity
Maximum Similarity
Importance Weighting (Optional)
Reference x
the weather is cold today
Candidate $\hat{x}$
it is freezing today
Reference
the 0.713 0.597 0.428 0.408
weather 0.462 0.393 0.515 0.326
is 0.635 0.858 0.441 0.441
cold 0.479 0.454 0.796 0.343
today 0.347 0.361 0.307 0.913
it
is
freezing
today
Candidate
1.27
7.94
1.82
7.90
8.88
idf weights
$R_{\mathrm{BERT}} = \frac{(0.713\times1.27)+(0.515\times7.94)+...}{1.27+7.94+1.82+7.90+8.88}$

Figure 2.9: The computation of the BERTScore recall metric, $\mathrm{R}_{\mathrm{BERT}}$. Source: Zhang et al. [2019b]

The authors use cosine similarity of the embeddings as a distance measure of the two texts' embeddings. This is computed for each candidate-reference token pair as follows:

$$\text{Cos-Sim} = \frac{\mathbf{x}_\mathbf{i}^\top \hat{\mathbf{x}}_\mathbf{j}}{||\mathbf{x}_\mathbf{i}||\;||\hat{\mathbf{x}}_\mathbf{j}||} \tag{2.23}$$

[18]The authors used twelve variants of BERT from the huggingface transformers library (Wolf et al. [2019]).

From here, greedy matching is used to compare each token to the most similar token in the other sequence. A recall, precision and F1 measure are computed as follows[19]:

$$R_{BERT} = \frac{1}{|x|} \sum_{x_i \in x} \max_{\hat{x}_j \in \hat{x}} \mathbf{x_i}^\top \hat{\mathbf{x}}_\mathbf{j} \tag{2.24}$$

$$P_{BERT} = \frac{1}{|\hat{x}|} \sum_{x_i \in x} \max_{\hat{x}_j \in \hat{x}} \mathbf{x_i}^\top \hat{\mathbf{x}}_\mathbf{j} \tag{2.25}$$

$$F_{BERT} = 2 \frac{P_{BERT} \cdot R_{BERT}}{P_{BERT} + R_{BERT}} \tag{2.26}$$

There is an additional (optional) phase to the computation using inverse document frequency (idf) as a form of importance weighting[20]. The authors include this as previous studies indicate that rare words are more useful for comparing similarities between texts than common words (Banerjee and Lavie [2005]). Assuming a set of M texts $\{x^{(i)}\}_{i=1}^{M}$, the idf for each token w is computed as follows:

$$\text{idf}(w) = -\log \frac{1}{M} \sum_{i=1}^{M} \mathcal{I}[w \in x^{(i)}] \tag{2.27}$$

Here $\mathcal{I}[\cdot]$ is an indicator function representing the presence of w in text $x^{(i)}$. Combining Equation 2.24 and Equation 2.27, the importance weighted recall measure is computed as:

$$R_{BERT} = \frac{\sum_{x_i \in x} \text{idf}(x_i) \max_{\hat{x}_j \in \hat{x}} \mathbf{x_i}^\top \hat{\mathbf{x}}_\mathbf{j}}{\sum_{x_i \in x} \text{idf}(x_i)} \tag{2.28}$$

The authors find that `BERTScore` correlates better with human evaluation than other metrics on a machine translation evaluation task[21]. Furthermore, Li et al. [2019] demonstrate the potential to use `BERTScore` to evaluate abstractive summarization by using the (unweighted) version of F_{BERT} as a reward function to fine-tune a pre-trained summarization model. They evaluate on the CNN/DailyMail See et al. [2017] dataset and show improved performance using `BERTScore` to fine-tune their results compared to using `ROUGE`, with improved fluency and fewer repetitions. These results show the promise of `BERTScore` as an evaluation metric for text summarization.

2.4.4 BLEURT

This paper has already well-documented the impact of Transformers throughout NLP. `BERTScore` is one example of using these models for language evaluation. `BLEURT` (Sellam et al. [2020]) extends this a step further by pre-training `BERT` to act as an effective evaluator of natural language which is less susceptible to domain drift. The authors assert that a robust and expressive evaluation metric can be trained by leveraging unsupervised pre-training combined

[19]Using pre-normalized vectors reduces the computation in Equation 2.23 to just the numerator.

[20]These are computed from the test corpus

[21]The WMT18 metric evaluation dataset (Ma et al. [2018]), containing translations of 159 different models spanning 14 languages.

with fine-tuning on human evaluations.

Assuming a training set of $\{(\mathbf{x_i}, \tilde{\mathbf{x}}_\mathbf{i}, y_i)\}_{i=1}^N$ with $\mathbf{x_i}$, $\tilde{\mathbf{x}}_\mathbf{i}$ and y_i representing the reference sequence, candidate sequence and human evaluation score respectively ($y_i \in \mathcal{R}$), their objective is to train a model to learn $\mathcal{F} : (\mathbf{x}, \tilde{\mathbf{x}}) \longrightarrow y$. The output ratings are given by:

$$y_i = \mathcal{F}(\mathbf{x}, \tilde{\mathbf{x}}) = \mathbf{W} v_{CLS} + \mathbf{b} \tag{2.29}$$

Here the *CLS* token from the output of `BERT` is projected through a linear layer with weights $\mathbf{W}$ and biases $\mathbf{b}$. Both the linear layer and `BERT` are trained during fine-tuning to minimize a regression loss function, while only `BERT` is trained during pre-training.

The main innovation from `BLEURT` comes from their novel pre-training scheme using a synthetic dataset tailored to evaluation. They create a dataset of 6.5 million samples, each containing a sentence extracted from Wikipedia alongside an artificially perturbed version of that sentence. Their intentions are that via the synthetic perturbations, the model will be familiar with many of the issues it might encounter between the hypothesis and target summaries when deployed as an evaluation metric. To produce the dataset they use three techniques: 1) `BERT` Mask-filling (mask 15% of tokens and then in-fill these tokens using `BERT`, thus introducing lexical diversity without ruining the fluency and grammar of the original sentence); 2) back-translation to generate semantic-preserving variants on sentences; 3) word-dropping.

Each sample is then augmented with a set of k signals indicating the similarity between the two sentences. Each of the k signals correspond to individual tasks and include automatic metrics such as `ROUGE` or `BLEU` or a textual entailment signal indicating if a sentence is entailed by the other[22]. Their experiments show `BLEURT` has state-of-the-art performance on the WMT Metrics Shared Task (2017-2019, to-English pairs, Ma et al. [2018]). `BLEURT` is an obvious candidate for us to consider in our search for a better evaluation metric and it will be interesting to test its generalization to summarization.

2.4.5 MoverScore

`MoverScore` (Zhao et al. [2019]) was developed contemporaneously to `BERTScore` and can be seen as a generalized version of `BERTScore` combined with Word Mover's Distance (WMD, Kusner et al. [2015]). WMD computes the similarity of two documents as the minimum distance between the embedded representations of two documents. This is an instance of the widely-studies Earth Mover's Distance (EMD) transportation problem for which numerous efficient solvers exist. Their experiments show that `MoverScore` performs competitively or state-of-the-art on numerous evaluation tasks including on the 2008 and 2009 TAC (Ji and Grishman [2011]) summarization datasets.

Here we formally introduce `MoverScore`. Assume we have a sentence $\mathbf{x} = (x_1, ..., x_m)$, with $\mathbf{x}^n$ representing the sequence of n-grams of $\mathbf{x}$ (e.g. the sequence of bigrams of $\mathbf{x}$ is $\mathbf{x}^2$). Assume that we also have a vector of weights $\mathbf{f}_{\mathbf{x}^n} \in \mathcal{R}_+^{|\mathbf{x}^n|}$, where $\mathbf{f}_{\mathbf{x}^n}^\top \mathbf{1} = 1$. Defining

[22]Measured using `BERT| fine-tuned on a Natural Language Inference (NLI) dataset, MultiNLI Williams et al. [2018])`

MoverScore for different n-grams allows for different variants of MoverScore analogously to the n-gram variants of ROUGE. In this thesis we use the 1 and 2-gram flavours, Mover-1 and Mover-2.

Solving for the WMD between two n-gram collections $(\mathbf{x}^n, \mathbf{y}^n)$ with associated weights $(\mathbf{f}_{\mathbf{x}^n}, \mathbf{f}_{\mathbf{y}^n})$ is akin to solving the following optimization problem:

$$\text{WMD}(\mathbf{x}^n, \mathbf{y}^n) := \min_{\mathbf{F} \in \mathcal{R}^{|\mathbf{x}^n| \times |\mathbf{y}^n|}} \langle \mathbf{C}, \mathbf{F} \rangle, \tag{2.30}$$

$$\text{s.t.} \quad \mathbf{F1} = \mathbf{f}_{\mathbf{x}^n}, \quad \mathbf{F}^\top \mathbf{1} = \mathbf{f}_{\mathbf{y}^n} \tag{2.31}$$

Here $\mathbf{C}$ is the transportation cost matrix, with the distance between the i-th n-gram of $\mathbf{x}$ and the j-th n-gram of $\mathbf{y}$ being $\mathbf{C}_{ij} = d(x_i^n, y_j^n)$. $\mathbf{F_ij}$ represents the transportation flow between the between the i-th n-gram of $\mathbf{x}$ and the j-th n-gram of $\mathbf{y}$, collected in the transportation flow matrix $\mathbf{F}$.

The distance function is the Euclidean distance between the two representations in the embedded space:

$$d(x_i^n, y_j^n) = ||E(x_i^n) - E(y_j^n)||_2 \tag{2.32}$$

Here E is the embedding function. BERT (Devlin et al. [2018]) is used to compute the contextual embeddings here[23]. The n-gram embedding is computed as the weighted sum of the individual token embeddings. By computing the WMD, MoverScore reflects both where the candidate and reference document overlap but also where they deviate.

Comparison with BERTScore BERTScore can be viewed as a hard-aligned case of WMD. BERTScore computes maximum pairwise cosine similarities for each token in the two sequences. In the context of WMD, this means that each token travels to the most semantically similar token in the other sequence, hence there is a hard one-to-one mapping for each token the sentence pairs. MoverScore uses soft alignments and the mapping across the sentence pairs is determined by solving the constrained optimization problem in Equation 2.31.

2.5 Leading Approaches to Text Summarization

The leading approaches to text summarization could be grouped into Transformers, RNNs and Reinforcement Learning (RL). The first generation of abstractive summarization models used RNNs but have since been overtaken by Transformers (Liu [2019], Zhang et al. [2019a], Radford et al. [2019], Raffel et al. [2019]). However, as discussed in section 2.3.4, the self-attention layer is a bottleneck for longer sequences as its memory requirement scales quadratically with input length, hence requiring modifications for use in long document summarization. RNNs scale linearly with sequence length so are more easily adapted. This study investigates some promising approaches to long document summarization and these architectures are outlined below.

[23]Any embedding function can be used by the authors' experiments show that BERT performs best.

Model	Variant	Pre-training Obj.	Self-Attention Type
BART	TED	Shuffle & mask spans.	n^2
PEGASUS	TED	`Gap-Sentence Prediction`	n^2
ProphetNet	TED	`Future n-gram prediction`	N-stream
Longformer	TE	N.A.*	Sliding window
Reformer	TE	N.A.*	`LSH & local`

Table 2.1: Summary table of the Transformer architectures used in this thesis. * pre-training was not conducted for these models.

2.5.1 TED Architectures

This section introduces the primary TED architectures used in this thesis. These are summarized in Table 2.1.

BART BART (Lewis et al. [2019]) is a leading model for generative tasks including text summarization. It is a TED[24] , which the authors explain is a combination of GPT (Radford et al. [2019]) and BERT (Devlin et al. [2018]), as shown in Figure 2.10. BERT is pre-trained using the Cloze task (Taylor [1953]), meaning that tokens are masked at random from the sequence and the model is trained to predict these tokens. When predicting a target token, the model can draw on context from before and after the target, hence BERT is bidirectional. In contrast, GPT is pre-trained to predict the next word in a sequence so can only use context from the left. This makes GPT more effective for generative tasks as context is only available from the left when performing these tasks, but less effective for non-generative downstream tasks as using context from both sides allows BERT to learn more meaningful representations during pre-training.

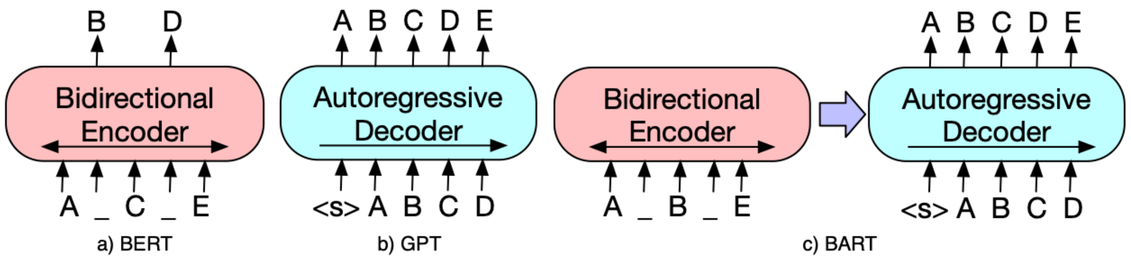


Figure 2.10: A schematic comparison of BERT, GPT and BART. BART has a bidirectional encoder allowing context from both sides, following BERT, and an auto-regressive decoder allowing context only from the left, following GPT. Source: Lewis et al. [2019].

BART (short for Bidirectional and Auto-Regressive Transformers) combines the two architectures by using a bidirectional encoder and an auto-regressive decoder, as seen in in Figure 2.10. This architecture broadens the scope of possible noising transformations that can be applied to the input sequence during pre-training, including modifications to the sequence length. The authors experiment with numerous pre-training objectives and find a combination of randomly shuffling the order of the input sequence and masking subsequences of tokens of random length with a single mask token to be the most successful. By

[24]The main architectural difference to the original TED is that the activation functions are changed from ReLUs to GELUs (Hendrycks and Gimpel [2016])

altering the lengths of sequences during pre-training, the model learns to reason over longer-range dependencies and overall output length. This makes BART's pre-training regime better suited for text summarization, resulting in new state-of-the-art performance.

PEGASUS BART demonstrated the effectiveness of the TED architecture for summarization. Subsequently, Zhang et al. [2019a] developed PEGASUS as an alternative TED model for summarization. The main difference between PEGASUS and BART is the choice of pre-training objective: Zhang et al. [2019a] introduce Gap Sentence Generation, GSG, a novel pre-training objective whereby entire sentences are masked and the objective is to generate these conditioned on the remainder of the document. The authors find pre-training using GSG to be most effective when salient sentences are masked and these are identified by computing the ROUGE score between each sentence and the remainder of the document.

ProphetNet Yan et al. [2020] introduced ProphetNet as another TED architecture for text generation tasks. The authors identify a tendency for language models (LMs) to learn biases through training whereby the model over-fits on local token correlations but under-fits on global token correlations. This stems from two factors: 1) teacher-forcing means models are trained to only predict one token ahead and therefore do not learn to plan ahead; 2) signals from local token correlations (i.e. bi-gram correlations) are normally much stronger than wider window dependencies.

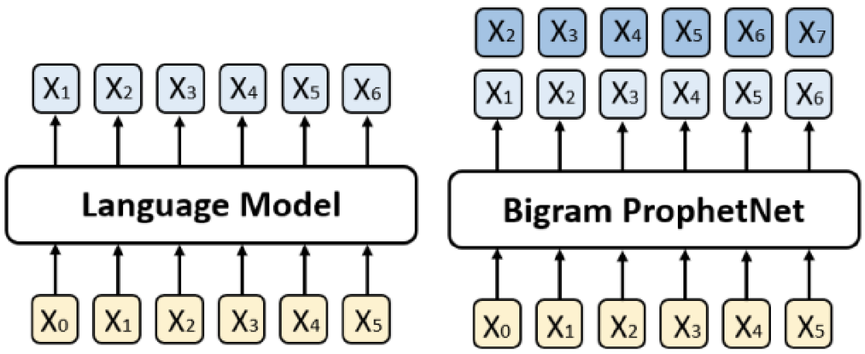


Figure 2.11: Schematic comparison of *ProphetNet's* future bigram prediction compared to the usual 1-step prediction in LMs. Here the next two tokens are predicted simultaneously at each training step.

ProphetNet is trained using a novel pre-training methodology, future n-gram prediction, to counteract these weaknesses. As illustrated in Figure 2.11, during training the model predicts the next n tokens simultaneously per step. This alters the *seq2seq* objective from predicting $p(y_t|y^i_{<t}, x)$ into predicting $p(y_{t:t+n-1}|y^i_{<t}, x)$. Figure 2.11 illustrates the bi-gram case where the model is trained to predict $p(y_t, y_{t+1}|y^i_{<t}, x)$. Predicting n tokens ahead is not possible using the usual Transformer Decoder; the authors modify the decoder to allow the model to predict more than one token ahead (N-stream self-attention). At inference time, the model only predicts the following token as for conventional LMs. This approach proved successful and, at the time of writing, ProphetNet has state-of-the-art performance on the CNN/DailyMail summarization task.

Transformer Decoder Another renowned study approaches the problem of generating Wikipedia articles as one of multi-document summarization of a set of reference documents[25] (Liu et al. [2018]). Being a multi-document summarization task, the input sequences were longer than other contemporaneous approaches. To account for this, the author used a two-stage approach: 1) use extractive summarization to select a subset of the input and 2) train an abstractive summarization model to generate the output articles conditioning on the extracted summary as input. They use a number of methods including eschewing the encoder and using local self-attention in place of dense self-attention (see section 2.5.2 for more details). We would have liked to have included this model as a baseline but the authors have not released pre-trained model weights and we do not have the hardware available to train from scratch.

2.5.2 Memory-Efficient Transformers

As discussed in section 2.3.4, the attention mechanism is a bottleneck for longer sequences as its memory requirement scales quadratically with input length. This means that the self-attention mechanism in TEDs requires modification for their efficient use in long document summarization. A crude (but common) solution is to truncate the input document and we will include this as a baseline model. More sophisticated solutions replace the self-attention layer with a less computationally and memory intensive alternative and we examine the `Longformer` and `Reformer` as two solutions next. These are summarized in Table 2.1.

Longformer Beltagy et al. [2020] design the `Longformer` as a Transformer Encoder (TE) suitable for long document tasks. Their solution is to replace the $O(n^2)$ attention mechanism with a sparse attention mechanism which scales linearly with input length, n. This sparsity is achieved through using "attention patterns" which specify how positions attend to other positions in the sequence. Consequently, the `Longformer` has a maximum input size of 4,096 tokens compared with 1,024 for `BART`; this results in the `Longformer` being able to read 76% of PubMed documents and 39% of arXiv documents without truncation, compared to 13% and 3% respectively for `BART`.

Figure 2.12 illustrates this schematically: a dense matrix represents "normal" attention as each position attends every other position. Sliding window attention uses a w-sized window[26], where each token attends to $\frac{w}{2}$ tokens each side of its position. Drawing from dilated convolutional neural networks, CNNs, (van den Oord et al. [2016]), each token attends to every other token on each side when using dilated sliding window attention. Analogously to CNNs, these layers can be stacked, making a receptive field of size $l * d * w$, with l layers and d dilation.

The final attention pattern is global + sliding window. This allows global attention to be specified for some (small number) of tokens in the sequence. For example, in classification tasks, global attention can be specified for the whole sequence `[CLS]` token (when using `BERT`). This means that `[CLS]` can attend to every token and every token can attend to it, adding high representational power to select positions within the sequence. The attention matrix for global + sliding window attention is shown in Figure 2.12. Their model shows

[25] These consist of articles cited in the Wikipedia document and supplemented by crawled Google web search results when the article had few citations.
[26] $w << n$ for efficient implementation.

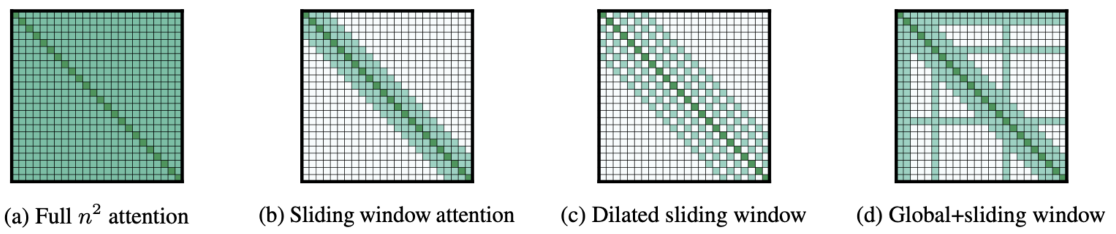

(a) Full n^2 attention
(b) Sliding window attention
(c) Dilated sliding window
(d) Global+sliding window

Figure 2.12: A schematic comparison of the different attention patterns in the Longformer. n^2 attention corresponds to the "normal" dense self-attention. The remaining figures show different examples of the sparse self-attention introduced in Beltagy et al. [2020]. Source: Beltagy et al. [2020]

consistent improvement over a baseline obtained using `RoBERTa` (Liu et al. [2019]) on a number of long document tasks. As their model is only a TE, it cannot be applied as-is to text summarization. A drawback of this approach is that the tokens selected for global attention are task specific and must be manually selected.

Reformer The `Longformer` (Beltagy et al. [2020]) outlined above is one solution for tackling the quadratic bottleneck of self-attention within Transformers. An alternative architecture tackling the same issue is the `Reformer` (Kitaev et al. [2020]). The authors question whether a "lighter" version of the Transformer (Vaswani et al. [2017]) with fewer parameters and less intensive computation can perform on par with the original. Identifying memory as the primary bottleneck, they introduce three innovations to reduce the memory requirements of the `Reformer`: 1) an approximate self-attention computation, `LSH` self-attention, 2) reversible layers and 3) chunking the feed-forward layers.

The first of these is `locality-sensitive-hashing, (LSH)` self-attention. Similar to sliding-window attention for the `Longformer`, this is an approximation to $O(n^2)$ attention, reducing the complexity to $O(n \log n)$. Equation 2.16 displays the self-attention computation. The main bottleneck is the $\mathbf{QK}^\top$ term. As we are concerned with $\text{softmax}(\mathbf{QK}^\top)$ we only need to compute the largest items as these will dominate the `softmax` with the remaining values near zero. Therefore attention can be approximated by identifying the most similar keys in $\mathbf{K}$ to each query vector $\mathbf{q_i} \in \mathbf{Q}$ and computing the dot product between these.

Kitaev et al. [2020] also notice that using a shared-QK Transformer, where $\mathbf{Q} = \mathbf{K}$ does not reduce performance. This permits bucketing of query vectors with high cosine similarities resulting from the transitivity of the cosine similarity operator: if q_i and q_j, and q_j and q_k have high cosine similarities, then q_i and q_k also have high cosine similarity. Taking the `softmax` of the vectors in each `LSH` bucket, we can approximate the full `softmax` computation with lower cost.

The challenge now is to efficiently bucket the most similar query vectors. This is a high-dimensional nearest-neighbour problem but using nearest neighbours is too expensive. The authors approximate this solution using the Angular `LSH` algorithm, a variant on the `LSH` algorithm (Andoni et al. [2015]). This is illustrated schematically in Figure 2.13: the query vectors are projected onto the unit circle which is divided into a number of regions. The projected vectors are then subjected to random rotation. This step is repeated several times, each time recording the region the two points end up in. The logic is that two similar queries

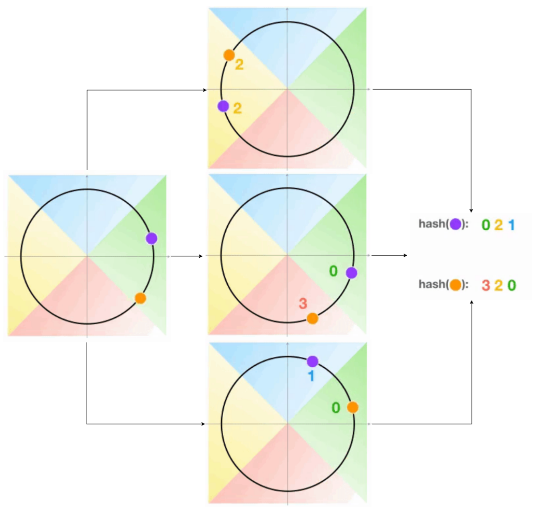


Figure 2.13: Angular LSH with two dissimilar query vectors. Angular LSH is an approximate method for solving for nearest-neighbours in high-dimensional space. As two of the three permutations result in the points being in different regions, these two points would not be placed into the same hash bucket. Source: Dirafzoon [2020].

will end up in the same region lots of times, while dissimilar points will end up in the same region rarely.

Figures 2.13 and 2.14 illustrate this schematically: in Figure 2.13, the two points are not similarly located and therefore only end up in the same region once after the random rotations. In Figure 2.14 the projected points are similar and therefore are in the same region all three times. Consequently, the queries corresponding to the two projected points in Figure 2.13 would not share an `LSH` bucket, while the points in Figure 2.14 would.

In their final model, the authors alternate between `LSH` self-attention and `local` self-attention. `Local` layers chunk the input and computes dense self-attention on each chunk separately[27]. The output is concatenated to form the attention layer output. These layers reduce memory consumption from $O(n^2)$ to $O(n*c^2)$, where c is chunk size (von Platen [2020]).

As mentioned, the `Reformer` also has two additional innovations. These will not feature in our research so we only touch on them. The first is reversible layers, inspired by RevNets from Computer Vision (Gomez et al. [2017]). The observation here is that each layers' activations can be computed using the previous layers' activations, and therefore we can trade speed off for memory consumption by re-computing each layers' activations prior to performing backpropagation. This makes memory consumption independent of the number of layers in the encoder / decoder.

The final innovation is chunking the feedforward module computations following the self-attention layers. This module consists of two projections with a non-linear activation function between them, and the projection matrices have dimension d_{hid} x $d_{intermediate}$ and $d_{intermediate}$ x d_{hid}. Transformers tend to have $d_{intermediate} > d_{hid}$, which can cause a large

[27]As this does not allow tokens on the boundary of the chunk to attend to their neighbours across chunks, this design is relaxed to allow the boundary tokens to attend to their immediate neighbours in the preceding/subsequent chunks.

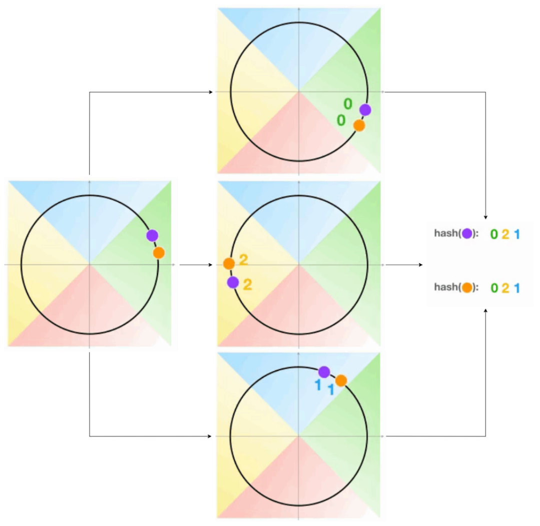


Figure 2.14: Angular LSH with two similar query vectors. Angular LSH is an approximate method for solving for nearest-neighbours in high-dimensional space. These queries would be placed into the same LSH bucket as they are in the same region after each of the rotations. Source: Dirafzoon [2020].

memory bottleneck here[28]. The authors observe that this computation can be performed for each hidden unit independently, and therefore the computation can be chunked and concatenated after being projected back down to size d_{hid}. This avoids having to have to compute the (large) intermediate tensor in one go and therefore reducing memory consumption. The authors find that the benefits only manifest when inputs start becoming very long (i.e. over 4K tokens, beyond the range we experiment with) so we do not implement this method.

2.5.3 Using Recurrent Neural Networks

RNNs were overtaken by Transformers as state-of-the-art for text summarization because of the latter's greater proficiency with producing fluent and readable summaries. However, the Transformer cannot be used for long documents without substantial surgery. Given that RNNs' memory complexities scale linearly with sequence length, they should provide a competitive baseline without requiring substantial surgery. This section outlines the RNN-based text-summarization architecture used as baselines for this study.

Pointer Generator Networks In an early pioneering work, See et al. [2017] devised the Pointer Generator Network (PGN) as a hybrid of Vinyals et al. [2015] Pointer Networks[29] and the traditional *seq2seq* RNN with attention (e.g. Nallapati et al. [2016]). The RNN *seq2seq* component of this model follows that outlined in section 2.3.2, using a single layer bidirectional LSTM as the encoder and a single-layer unidirectional LSTM as the decoder. This generates P_{vocab}, the usual distribution over the vocabulary. The main development with the PGN is that the model can choose to generate words from the vocabulary or copy tokens from the input via pointing. This modification can be seen by comparing Figure 2.15, an illustration of the PGN, to the *seq2seq* RNN in Figure 2.7. The novel section is the p_{gen} node, a scalar denoting the probability of generating a word. Formally, the final vocabulary

[28]The large version of BART has $d_{intermediate} = 4,096$ and $d_{hid} = 1,024$ (Lewis et al. [2019]).
[29]This uses attention to select (or "point" to) one of the input tokens as the next output token.

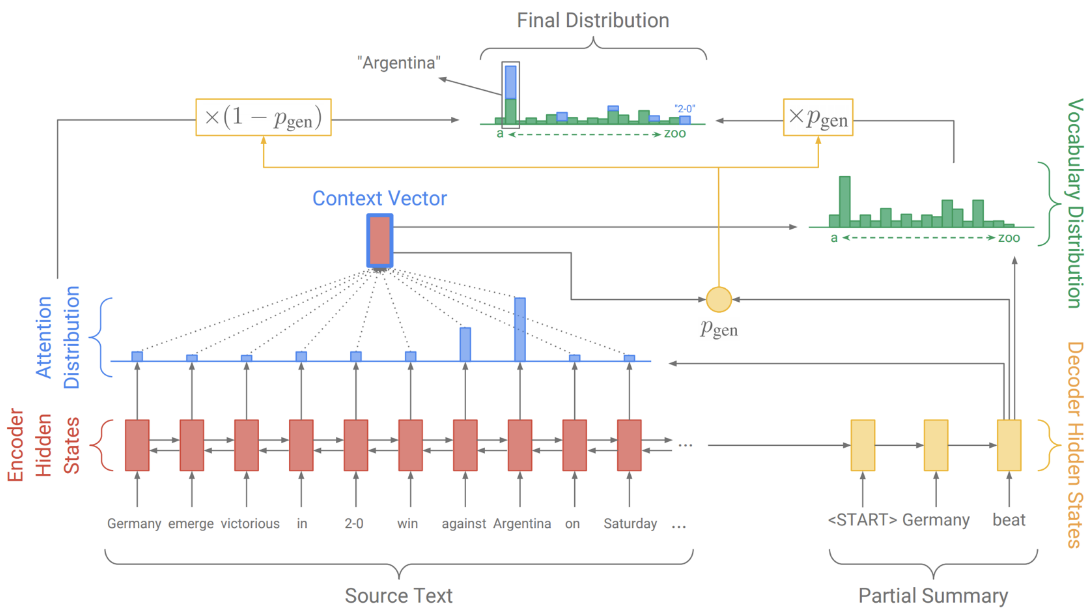


Figure 2.15: A schematic of the Pointer Generator network. Source: See et al. [2017]

distribution P_{final} is computed with:

$$P_{final} = p_{gen}P_{vocab}(w) + (1-p_{gen})\sum_{i:w_i=w}\alpha_i \tag{2.33}$$

Here α_i are the attention weights computed in Equation 2.13 on page 10. This architecture allows the model to copy from the input, a feature unavailable in contemporaneous RNNs for *seq2seq*. The network learns to adjust p_{gen} so it can copy from the source text when the most appropriate token at the next decoding step exists in the summary or when the model encounters out-of-vocabulary (OOV) words. When published, this model improved substantially on the state-of-the-art for the CNN/DailyMail dataset by over 2 ROUGE and is often used as a baseline for text summarization experiments.

2.5.4 Beyond Maximum Likelihood

The text summarization community has seen an interesting recent shift towards extending the training optimisation procedure to alternatives to Maximum Likelihood. These approaches share two common pillars of understanding: 1) Maximum Likelihood training cannot account for lexical and compositional diversity and therefore is too restrictive as the sole training method; 2) training using Maximum Likelihood but evaluating on ROUGE creates a metric discrepancy and calls the training methodology further into question. This section highlights reinforcement learning as a significant niche within this area.

Reinforcement Learning One approach addressing the above problems involves incorporating reinforcement learning (RL) into the training pipeline (Böhm et al. [2019], Li et al. [2019], Pasunuru and Bansal [2018]). In comparison to the Maximum Likelihood paradigm which pushes the model to reproduce the reference summary exactly, RL optimises the model to maximise expected rewards, hence the model is more directly incentivised to produce higher-quality summaries. In these approaches, the model is usually pre-trained using supervised learning (usually a language modelling task) and RL is used as a fine-tuning step because of the poor sample efficiency resulting from sole use of RL.

Early approaches (e.g. Pasunuru and Bansal [2018]) used ROUGE directly as the reward function, addressing the metric discrepancy. However, as discussed in section 2.4.1, ROUGE has its own shortcomings, so Li et al. [2019] extended this idea to use BERTScore (Zhang et al. [2019b]; see section 2.4.3) as the reward function[30]. In a similar vein, Böhm et al. [2019] learn a reward function with which they fine-tune their summarizer. To train their reward function they use a dataset consisting of 500 randomly sampled CNN/DailyMail articles. For each article they produce five summaries (the reference summary plus four from different summarization models) and use human evaluators to score the quality of each of them. They then learn a reward function to best approximate the human scores and find that using BERT with an multi-layer-perceptron worked best. This model was then used as the reward function for training their summarizer.

Unsurprisingly, given the modified training objective, these approaches do not compete with leading Transformer summarizers on ROUGE scores. These studies resort to human evaluation as a means of benchmarking, and their results show improvements in fluency, lexical diversity and read-ability when compared to Maximum Likelihood methods. However, the major drawback of human evaluation is the inability to compare against models which were not included within the experiment, and therefore it is difficult to assess the success of this class of models overall.

2.5.5 Unsupervised Learning

Unsupervised learning has been a key driver for progress in NLP and text summarization in recent years. The Transformer paradigm relies on unsupervised pre-training on large unlabeled corpora. The key innovation behind the leading text summarizers was to modify the pre-training objective to be better suited to generation and summarization (e.g. BART, section 2.10). The unsupervised learning approach to text summarization seeks to push this a step further, using summarization as the sole pre-training task on large unlabelled corpus, presupposing that pre-training using an explicit summarization task will further improve model performance. This section is less concerned with model architectures; instead it highlights lead bias use and "mean review summarization" as two novel tasks which open the door for pre-training models on large, unlabelled corpora using an explicit summarization pre-training objective.

Lead Bias It is journalistic convention that news articles follow a structure known as the "Inverted Pyramid" (Scanlan [2008]), where the most fundamental information is placed at the beginning of the article followed by background and supporting details. It is well-understood within the summarization community that the first three sentences of a news article are actually a strong baseline for a summary and one that summarization models struggle to beat (See et al. [2017]), a phenomenon known as *lead bias*. Lead bias is generally a challenge for the summarization community as the output of summarizers trained on

[30]They use a *seq2seq* RNN LM as the backbone and leave analysis of the impact on Transformer-based summarizers to future work.

Dataset	**# docs (K)**	**Avg. doc # words**	**Avg. sum # words**	**% over 1,024 toks**	**% over 4,096 toks**	**Citations**	**Year**
CNN	92	656	43	8%*	0%	998	2017
DailyMail	219	693	52	8%*	0%	998	2017
PubMed	133	3,016	203	87%	24%	81	2019
arXiv	215	4,938	220	97%	61%	81	2019

Table 3.1: Descriptive statistics of the datasets used in this study. The fifth and sixth columns indicate the share of each dataset that would not fit in `BART` and `LED-4096` models given their capacities of 1,024 and 4,096 tokens respectively. The number of citations is the Google Scholar citations from the original papers (Cohan et al. [2018], See et al. [2017]) as of 24/08/2020. * The CNN/DailyMail dataset is 8% overall.

these datasets are inevitably skewed to favour the article lead, meaning the remainder of the article is neglected and the model fails to learn to distil content as desired.

Zhu et al. [2019] recognise lead bias as an opportunity: by training a model to reproduce the first three lines using the remainder of the article, this renders the large quantity of news articles available online suitable for pre-training models. They amass a corpus consisting of 21.4M news articles, two orders of magnitude larger than the CNN/DailyMail dataset. Their results show that their model (a TED) outperforms other unsupervised learning approaches when evaluated on the CNN/DailyMail datasets and is competitive with many strong supervised learning baselines even without any fine-tuning.

Mean Review Summarization Chu and Liu [2018] examine the problem of unsupervised multi-document summarization using Amazon and Yelp reviews. Their approach can be used in situations where multiple reviews exist for a single entity; by encoding these reviews and averaging in the latent space, they suppose that you reach the "canonical review". Their model then decodes this review and attempts to minimize a reconstruction loss between this review and all of the source reviews (their model is an autoencoder). This is another novel approach permitting large-scale unsupervised learning.

3 Datasets

This section introduces the datasets which will be used throughout the subsequent chapters. These can be divided into summarization and evaluation datasets following the work-stream split of this thesis. These datasets are outlined below.

3.1 Summarization Datasets

The summarization datasets are CNN/DailyMail, arXiv and Pubmed, and these will be the focus of our architectural analysis in section 5.2. Summary statistics for these datasets can be found in Table 3.1 and the distributions of document lengths for their respective tests sets are plotted in Figure 3.1. Sample summaries for each of the datasets can be found in Appendix A.

CNN/DailyMail Contains articles from the DailyMail and CNN newspapers paired with summaries (in the form of story highlights) written by the same author, which act as the ground truth. This is the most commonly used and reported-on dataset within the text summarization community. It was first introduced in Hermann et al. [2015], although use the non-anonymized version from See et al. [2017] following recent convention in the literature. This dataset is extracted from the raw text files from the original CNN/DailyMail articles[31]. Capitalization is preserved and newline characters are removed. The CNN source articles begin with a "(CNN)" token and we preserve this. We use same train/test/validation split as See et al. [2017], amounting to 91.4%, 4.0% and 4.6% of the dataset respectively. This dataset is freely available for commercial use, modification, distribution and private use under the MIT license[32].

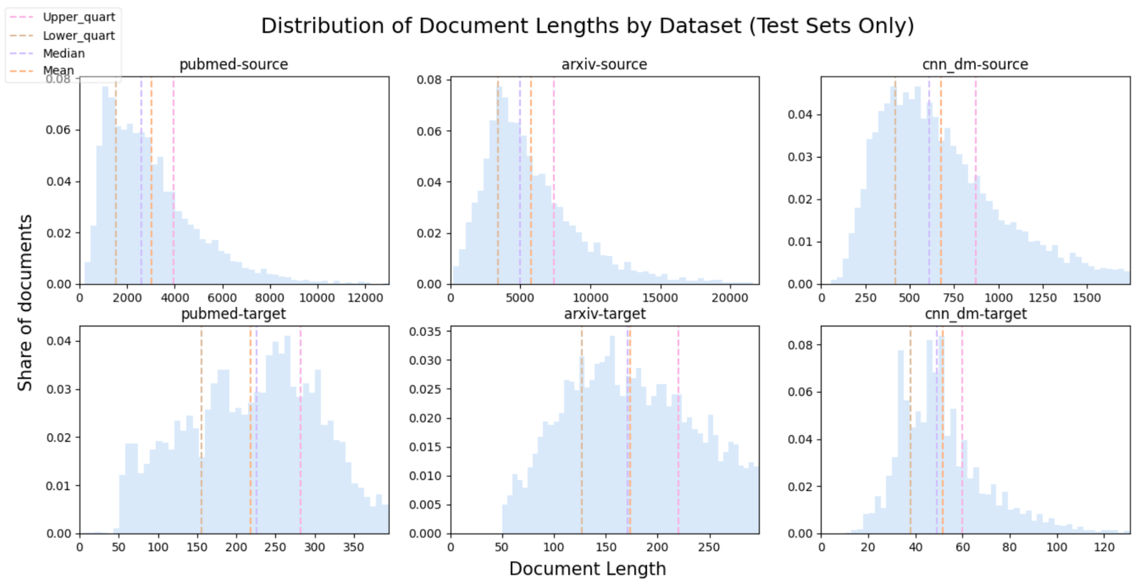


Figure 3.1: Histograms showing the distributions of document lengths for the test set of each dataset. The top figures show for the source document and the bottom figures show for the target documents. The mean, median, upper and lower quartiles are annotated onto each figure.

arXiv & PubMed Introduced by Cohan et al. [2018] as two long document summarization datasets. The task is to reproduce the article abstract, which operates as the ground-truth summary. arXiv.org and PubMed.com are online repositories containing scientific research papers, primarily from maths, computer science and engineering for the former and biomedical and life sciences for the latter. These datasets can be downloaded and used without a license as they are open access repositories.

We use the extracted dataset from Cohan et al. [2018][33]. This dataset is generated by extracting raw text files from the articles' LaTeX files using Pandoc[34]. Sections following the conclusion are discarded; the conclusion section is identified by the most common phraseology for the section header (i.e. concluding remarks, summary or conclusion). The document is lower-cased, figures and tables are removed, maths formulae are replaced with "@xmath0" and citations are replaced with "@xcite" tokens. For the summaries, sentence

[31] Download instructions: `https://github.com/abisee/cnn-dailymail`
[32] `https://github.com/abisee/cnn-dailymail/blob/master/LICENSE.md`
[33] Link: `https://github.com/armancohan/long-summarization`
[34] Link: `https://pandoc.org`

Quora Question Pairs		
Question 1	Question 2	Equivalent?
Can we ever store energy produced in lightning?	Is it possible to store the energy of lightning?	Yes
What Game of Thrones villain would be the most likely to give you mercy?	What Game of Thrones villain would you most like to be at the mercy of?	Yes
Why do some people think Obama will try to take their guns away?	Has there been a gun control initiative to take away guns people already own?	No
What are the best YouTube channels to learn medicine?	What are some of the best YouTube channels for learning Git?	No

Table 3.2: Two positive and two negative samples from the QQP dataset. The objective is to ascertain whether the two questions are semantically equivalent and these labels are provided in the final column.

boundaries are marked using “< **S** >” and “< /**S** >” tokens. 3% and 5% of arXiv and PubMed respectively are used for each of the test and validation sets, with the remainder used for training (using the same train/test/validation split as in Cohan et al. [2018]). Cohan et al. [2018] preserve the discourse structure of the documents by keeping section headings but our models do not utilize this discourse structure so we discard this information. In addition, some source documents and/or summaries are excessively short; we remove documents with fewer than 200 tokens or with summaries shorter than 5 tokens following Cohan et al. [2018]. These account for 0.7% of arXiv and 3.3% of PubMed documents.

3.2 Evaluation Datasets

The evaluation datasets consist of the Quora Question Pairs and annotated CNN/DailyMail datasets, and these are the focus of the evaluation-metric analysis in section 5.1.

Quora Question Pairs The Quora Question Pairs dataset (QQP, Sharma et al. [2019b]) tests a system’s Natural Language Understanding (NLU)[35]. The dataset is composed of 404K pairs of questions published by uses to the question answering forum, `www.quora.com`. They span a wide range of topics including common sense, politics and health and medicine. Many of the questions published to the website are duplicates and the QQP dataset was designed to train machine learning systems to determine the duplicate questions. Each of the samples is therefore labelled by human experts as duplicate or distinct. The task is a binary classification objective of predicting the correct label. 63% of the samples are distinct with the remaining 37% being duplicates. Several examples are illustrated in Table 3.2.

Annotated CNN/DailyMail Introduced by Chaganty et al. [2018] to assess the degree to which human evaluator judgement and summarization evaluation metrics’ scores correlate. This dataset contains 500 random samples from the CNN/DailyMail dataset. They produce four summaries for each sample using text summarization models[36] and tasked human evaluators with assigning each of the summaries a score[37]. Additionally, each sample has the

[35]Download: `https://www.kaggle.com/c/quora-question-pairs`

[36]Specifically using the *seq2seq* and `pointer` models from See et al. [2017] and the `ml` and `ml+rl` models from Paulus et al. [2017].

[37]The score was based on their fluency, lack of redundancy and overall quality

standard reference summary, meaning the dataset comprises of 500 samples, each with four scored hypothesis summaries and a reference summary.

3.3 Discussion

As highlighted above, we are using the CNN/DailyMail, arXiv and PubMed datasets as the primary datasets in this thesis. Another option for long document summarization is the BIGPATENT dataset (Sharma et al. [2019a]). This consists of 1.3 million U.S. patent documents. The target is a human-written abstractive summary. This dataset is advantageous in that salient information is distributed throughout the document, whereas in academic research papers or news articles important information is normally highlighted early-on in the document. We decided not to use this dataset for two reasons: 1) arXiv and PubMed had more community recognition at the time of writing. For example, the arXiv/PubMed paper had over seven times more citations than the BIGPATENT paper[38]. 2) The computational cost of running one epoch of BIGPATENT is over six times that of arXiv given their relative sizes. This would pose problems for fine-tuning the models.

As mentioned in the introduction, multi-document summarization is a prominent unsolved research goal in this field. We would have liked to address this question in this thesis. The limiting factor was the dataset: the MultiNews dataset (Fabbri et al. [2019]) is the solitary multi-document summarization dataset at the time of writing. This dataset is relatively small (44K samples) and primarily uses short documents. These limitations deterred us from pursuing this path. This is one area where datasets are limiting the progress of Text Summarization. Two other areas similarly constrained by the absence of an adequate dataset are whole-book summarization and daily financial news generation.

4 Design

4.1 Evaluation Analysis

In this section we outline the set of eight evaluation metrics used in subsequent analysis. A summary comparison of each of these can be seen in Table 4.1.

4.1.1 Description of Metrics

In section 2.4 we explained the need for an improved set of evaluation metrics and theoretically motivate four model-based candidates. This section outlines the specific metrics used in our analysis, alongside any required configuration decisions. We also introduce our novel metric, `BARTScore`, based on `BERTScore` but using a TED in place of a TE. A side-by-side comparison of each of the metrics can be found in Table 4.1.

ROUGE-1, ROUGE-2 & ROUGE-L Computes the lexical overlap between the candidate and reference summaries[39]. `ROUGE-1` / `ROUGE-2` computes the co-occurrences of 1 / 2-grams, while `ROUGE-L` measures the longest common subsequence between two texts [40].

[38]Measured on 24/08/2020; Cohan et al. [2018] and Sharma et al. [2019a] respectively.

[39]Github repository: `https://github.com/google-research/google-research/tree/master/rouge`.

[40]Two flavors of `ROUGE-L` exist with the two differing on how newlines are treated. The literature generally reports the variant that treats newlines as sentence boundaries and the longest common subsequence is com-

Metric	Citations	Year	Contextual Embeddings?	Language Model	n-gram comparison
BERTScore	94	2019	Yes	RoBERTa-lg-MNLI	1-gram
BARTScore	N.A.	N.A.	Yes	BART-lg-MNLI	1-gram
Mover-1	26	2019	Yes	BERT-base-MNLI	1-gram
Mover-2	26	2019	Yes	BERT-base-MNLI	2-gram
BLEURT	3	2020	Yes	BERT-lg	Entire seq.
ROUGE-1	5,124	2004	No	N.A.	1-gram
ROUGE-2	5,124	2004	No	N.A.	2-gram
ROUGE-L	5,124	2004	No	N.A.	Entire seq.

Table 4.1: Side-by-side comparison of each of the evaluation metrics. The fourth and fifth columns indicate if the metric uses contextual embeddings, and the Language Model used to compute these if so. The number of citations is taken from the Google Scholar citations of the original papers introducing the metrics, recorded on 24/08/2020. These papers are: Zhang et al. [2019b], Zhao et al. [2019], Sellam et al. [2020], Lin [2004].

BERTScore Evaluates two texts based the cosine similarities between their embedded representations. Semantically similar texts will have lots of tokens which are co-located in the embedded space and therefore the pairwise cosine similarities between the two texts will be high (see section 2.4.3). BERTScore[41] (Zhang et al. [2019b]) is best-understood as a variant of ROUGE incorporating bidirectional context via encoding using a TE LM such as BERT (Devlin et al. [2018]). BERTScore can use any TE as the encoder and we provide a comparison of them in section 6.1.1. We use RoBERTa$_{\text{large}}$ which has 24 layers, 16 attention heads and 1,024 embedding size (Liu et al. [2019]).

BARTScore Analogous to BERTScore except a TED, in this instance BART (Lewis et al. [2019]), is used as the LM instead of a TE. Aside from this, BERTScore and BARTScore are identical and BARTScore was implemented following the structure of the BERTScore API. This might appear a counter-intuitive design choice as architecturally the TED differs from the TE in that the latter uses bidirectional context whereas the former uses unidirectional context (within the decoder). When encoding a sequence, bidirectional context seems preferable as the model has a wider perceptive field and is therefore better able to output meaningful embeddings. This implies the TE should be chosen over the TED. However, in the BART paper, the authors compare BART with RoBERTa (Liu et al. [2019]) on SQuAD (Rajpurkar et al. [2016]) and GLUE (Wang et al. [2018]) tasks and conclude that uni-directional context in the decoder does not harm performance.

The potential benefits of using BART stem from it being pre-trained on tasks more suited to text generation. BARTScore uses the large pre-trained version of BART[42]. BERT is trained using Masked Language Modelling, where 15% of tokens are masked and the objective is to reproduce the original document[43]. The emphasis when pre-training BERT is on predicting single tokens independently. In comparison, BART was pre-trained using text infilling (similar to Masked Language Modelling except spans of tokens are masked rather

puted between each pair of source and target sentences (sometimes called ROUGE-LSum). We follow convention and report this version.

[41]Github repository: `https://github.com/Tiiiger/bert_score`.

[42]12 layers in the encoder and decoder, 16 heads, and 1024 hidden units (Lewis et al. [2019])

[43]BERT was also trained using next sentence prediction but this was removed for RoBERTa.

than single tokens) and sentence permutation (the input document is split according to full stops and sentences are shuffled). The use of text-infilling places a greater emphasis on modeling sequence lengths as BART must predict an unknown number of tokens per masked span. We conjecture that this pre-training emphasis on modeling spans will help BART to form more meaningful representations of longer sequences and therefore would act as a better encoder for an evaluation metric. We use `bart-large` as the LM here. Model specification can be found in Table D.1.

Mover-1 & Mover-2 These are the 1 and 2-gram variants of `MoverScore`[44] (see section 2.4.5). Similar to `BERTScore`, `MoverScore` computes the distance between the embedded candidate and reference summaries. However, `MoverScore` does this by finding the minimum distance one document must be moved in the embedded space to be transformed into the other document (i.e. solving the constrained Word Mover's Distance optimization problem). To compute word embeddings MOVERSCORE uses `bert-MNLI` (Devlin et al. [2018]), BERT fine-tuned on the Multi-NLI dataset (Williams et al. [2018]). This model has 12 layers, 12 attention heads and 768 embedding size.

BLEURT BLEURT[45] is a version of BERT (Devlin et al. [2018]) pre-trained explicitly to act as an evaluation metric for natural language generation tasks (see section 2.4.4). It can also be fine-tuned on human ratings to further increase its sensitivity, although we choose not to do this to preserve comparability with the other metric and over concerns regarding the generalizability. BLEURT has pre-trained checkpoints for base and large models. We use the large model with 24 layers, 16 attention heads and 1,024 embedding size (experiments comparing the models can be found in section 6.1.1).

4.2 Architecture Analysis

This section motivates the primary models used and gives details of their designs and implementation processes. A schematic comparison of BART, LED and RED is shown in Figure 4.1.

4.2.1 BART

With the architectures workflow of this project, our goal was to determine whether approximate self-attention layers could deliver strong results in Transformer-based summarization models. This necessitated choosing a reference Transformer model, one that we could adapt and would provide a performance reference. We used BART as this model.

Motivations There were two compelling reasons to choose BART for this role: 1) at the time of writing, BART, PEGASUS and `ProphetNet` were all state-of-the-art summarization models[46]. 2) BART has the strongest momentum from the summarization community at this time. It is available through the huggingface transformers library (Wolf et al. [2019]) and has a large and active user-base meaning it is available with a rich set of useful features. This library also has implementations and pre-trained weights for the Longformer (Beltagy et al. [2020]) and the Reformer (Kitaev et al. [2020]), which we combine with BART to make our "long-form" model as will be outlined in the next section. Having all the code and model weights

[44]Github repository: `https://github.com/AIPHES/emnlp19-moverscore`
[45]Github repository: `https://github.com/google-research/bleurt`
[46]Using ROUGE metrics and the CNN/DailyMail dataset, these three models score near-identically.

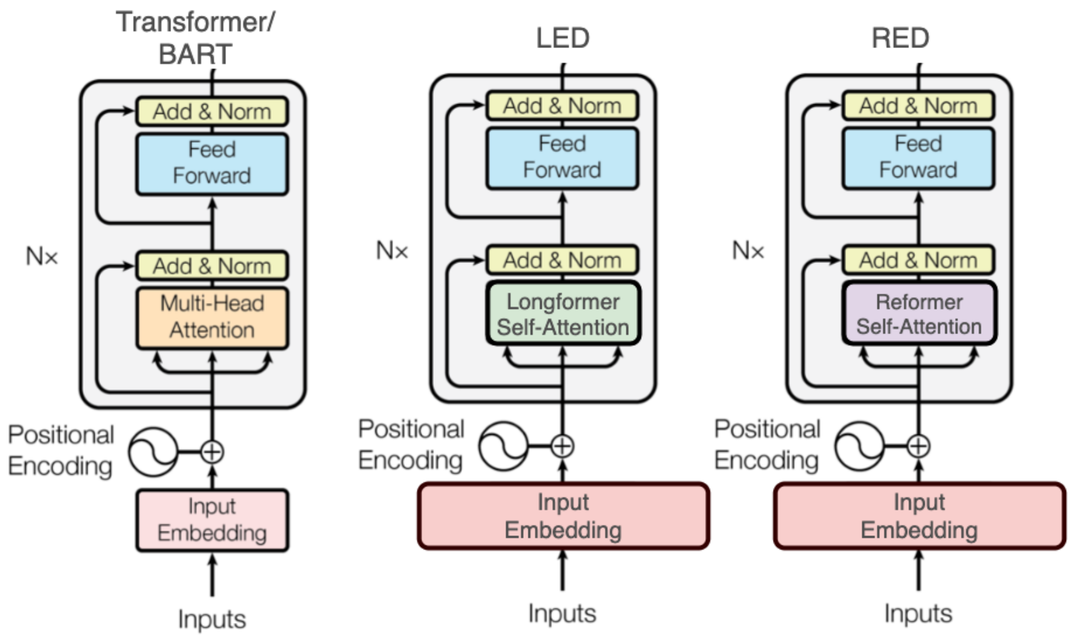


Figure 4.1: Schematic comparison of the original Transformer/BART, LED and RED architectures. The LED and RED have different self-attention layers and widened input embedding matrices in their encoders compared to the original Transformer and BART.

available in one API presented huge implementation efficiency gains and made this library the obvious choice to proceed with.

Use for long documents As outlined in section 2.5.2, n^2 self-attention memory consumption scales quadratically with input length. This makes it expensive to use BART for long sequences. BART is therefore designed with 1,024-width positional embeddings, meaning the maximum sequence length it can process is 1,022 tokens[47]. Although it is straightforward to adapt BART to use longer sequences by copying and concatenated the positional embedding matrix, we choose not to do this as the memory cost becomes impractical and our focus is on using approximate self-attention layers. Thus we simply truncate the input document by taking the first 1,022 tokens to permit us to use BART on longer sequences.

4.2.2 LED

The Longformer (Beltagy et al. [2020]) offers a solution to the Transformer's quadratic memory complexity whereby the self-attention layers (the bottleneck) are replaced with sliding-window attention. The effect of this is to replace a dense matrix multiplication with a sparse matrix multiplication, thereby creating linear complexity with respect to input length. The Longformer was developed as a TE so cannot be used as-is for *seq2seq* tasks. Beltagy et al. [2020] begin with a RoBERTa model and replace its self-attention layers with their sliding window attention in order to create the Longformer. This idea of substituting dense self-attention layers for sliding-window self-attention is applicable to other Transformer architectures, namely the TED. The LED does exactly this, beginning with BART (Lewis et al. [2019]) and substituting the self-attention layers for sliding-window self-attention.

[47]Two less than the maximum positional embeddings because of the beginning and end of sequence special tokens.

Implementation details The LED is built using the huggingface transformers library (Wolf et al. [2019]) as this libaray contains implementations of both Longformer and BART. Converting BART to the LED is a two-stage process. First, BART's encoder must be "widened" to allow longer inputs to be processed by the model[48]. This occurs through enlarging the positional embedding matrix by copying the weights and concatenating the matrices. As highlighted above, BART has a maximum input length of 1,022 tokens. If we wanted to process sequences of up to 2,044 tokens, we would copy and concatenate the positional embedding matrix, making it twice as wide. This method has proved effective at allowing Transformers to process longer inputs (Zhang et al. [2019a]).

The second step is to replace BART's self-attention layers with sliding-window self-attention. Figure 2.8 shows the canonical TED architecture. Attention is used in three places in the TED: encoder self-attention, decoder self-attention and cross attention. When performing summarization, the input length (n) is usually (often much) greater than the output length (m). The primary bottleneck is therefore the encoder self-attention layer as $n^2 > n*m > m^2$. The simple implementation of the LED replaces the self-attention layers only in BART's encoder. The weights in the query, key and value matrices are then replaced with the weights from the respective weights matrices from BART for each layer. These steps are demonstrated schematically in Figure 4.1. We leave implementing sliding-window attention in the decoder (a straightforward extension) and the cross-attention layers (less straightforward) to future work.

Concurrent to the development phase of our implementation of the LED, a repository was published on GitHub containing a parallel implementation[49]. This was shortly followed by Beltagy et al. [2020] augmenting the Longformer codebase to add the capability to run the Longformer as a TED by adapting BART as described above. The Beltagy et al. [2020] implementation had numerous useful additional features such as gradient checkpointing (Chen et al. [2016]) and convenient model saving/loading; satisfied that our implementations matched, we proceeded with the experimentation phase of this project using a combination of the huggingface and Beltagy et al. [2020] libraries. Our contribution with the LED is to be the first to systematically benchmark and experiment with this model.

4.2.3 RED

Using the Longformer's sliding window self-attention in place of regular self-attention is just one method of reducing the complexity bottleneck of TEDs. Any "approximate" self-attention layer could perform this job, raising the question of which one is most effective for TED summarization models. At the time of writing we could not find any evidence of a side-by-side comparison of approximate self-attention layers in the summarization domain. In this section we introduce the Reformer Encoder Decoder (RED), analogous to the LED except we replace BART's self-attention layers with Reformer self-attention (Lamb et al. [2016])[50].

[48]The decoder does not need to be "widened" for BART as conventionally the model is intialized with a "wide" decoder. In our version of BART (huggingface's bart-large-cnn, Wolf et al. [2019]) the decoder "width" is 1,024 tokens which is wide enough for the majority of summarization use-cases.

[49]Link: https://github.com/patil-suraj/longbart

[50]See section 2.5.2 for details of Reformer self-attention

Implementation details Creating the `RED` follows the same two-step process as creating the `LED`: 1) positional embedding matrix of the encoder must be widened to allow longer inputs, and 2) `Reformer` self-attention is substituted in for the `Encoder`'s self-attention layers. We alternate between `local` and `lsh` layers as in the `Reformer` (see section 2.5.2 for details)[51]. As before, we use `BART` as the base model. These steps are shown schematically in Figure 4.1. We implement the `RED` using the huggingface `BART-large` model and replacing the self-attention layers with the huggingface implementation of the `Reformer`'s `lsh` and `local` self-attention.

Limitations The `Reformer` self-attention weight matrices are not compatible with `BART`'s self-attention weight matrices as they have different hidden dimension sizes. This means that we could not begin with using `BART`'s pre-trained self-attention weights and therefore we had to begin pre-training the `Reformer` without pre-trained weights in the attention layers. Our experimental methodology in light of this is outlined in section 5.2.6.

5 Experimental Methodology

5.1 Evaluation Analysis

This section introduces our experimental design to compare between the relative performances of the evaluation metrics outlined in section 4.1.1. These experiments assess how well the metrics correlate with human judgement, how well they detect the semantic similarity between pairs of questions and how sensitive they are to various forms of corruption when these are applied to summaries. It will be seen that the model-based metrics outperform the `ROUGE` metrics on almost all tasks.

5.1.1 Preprocessing

ROUGE As `ROUGE` performs a surface-level comparison of n-gram overlaps between two texts, the output score is sensitive to the pre-processing pipeline (particularly stop-word removal, stemming, capitalization). Implementing `ROUGE` for this study was surprisingly problematic because the official `ROUGE` package is implemented in `perl` and is not conducive to a `python` workflow. Many unofficial GitHub repositories therefore exist providing `python` wrappers or `python` re-implementations of `ROUGE`. However, these often handle the pre-processing pipeline slightly differently (either by using different pre-processing steps or subtle differences between `python` and `perl` standard functions), and therefore their outputs rarely agree.

Following Zhang et al. [2019a], we use the Google Research Rouge implementation. This is a native `python` re-implementation of `ROUGE` and in our tests the outputs matched the official `perl` script very closely. As in Zhang et al. [2019a], our pre-processing pipeline involved tokenizing and stemming but not stop-word removal. The tokenizer is from the same codebase and is based on the Lin [2004] tokenizer. This involves lower-casing, replacing all non-alphanumeric characters with spaces (and therefore removing punctuation) and

[51]We experiment with using alternating layers, solely `local` and solely `lsh` layers and find alternating performs best.

Metric	Tokenizer	Type	Vocab # words
BERTScore	RoBERTa	BPE	50,000
BARTScore	BART	BPE	50,000
MoverScore	BERT	WordPiece	30,000
BLEURT	BERT	WordPiece	30,000

Table 5.1: The pre-trained tokenizer used by each of the evaluation metrics. See section 5.1.2 for experiment details.

tokenizing based on spaces. The stemmer uses the `PorterStemmer` from the Natural Language ToolKit (NLTK, Bird et al. [2009])[52], an implementation of the Porter suffix-stripping algorithm from Porter et al. [1980].

Model-based metrics An advantage of the model-based metrics is that they use Transformers and each of the models has a paired tokenizer which handles the pre-processing pipeline. This simplifies the workflow in comparison to ROUGE as the pre-processing reduces to using the usual pre-trained tokenizers.

Table 5.1 outlines the tokenizers used by each of the evaluation metrics. BERTScore and BARTScore use the RoBERTa and BART tokenizers respectively, and these are both heavily based on the GPT-2 tokenizer (Radford et al. [2019]). This uses Byte Pair Encoding (BPE, Gage [1994]), a method of constructing sub-word vectors adapted from the data compression algorithm. BPE is a middle-ground between character-level and word-level language modelling and helps circumvent the issue of unknown words while maintaining the empirical decoding advantages of word-level tokenization. BPE constructs a vocabulary initially by splitting words into all combinations of sequences of characters, computing frequency counts for each character-sequence and filling the vocabulary until the pre-defined limit is reached. The tokenizer then assigns words the longest character sequence present in that word. This is repeated recursively until all characters of all words have been encoded. MoverScore and BLEURT use the BERT tokenizer (Devlin et al. [2018]) which uses the WordPiece model (Wu et al. [2016]). The WordPiece model is an alternative to BPE for producing sub-word vectors. WordPieces are added to the vocabulary on the basis of increasing the likelihood of the training data, as opposed to frequency counts as for BPE.

It is important to note that we do not train any part of these metrics. We feel this is critical as we would like a universally applicable metric "off the shelf", in the same vein as ROUGE. However, this presents a potential limitation when it comes to uncommon words, as would be expected in the arXiv and PubMed datasets. Through the use of sub-word embeddings as outlined above, the metrics will be able to tokenize and compute embeddings for rare words; however, there is no guarantee that these representations will be meaningful. How meaningful these are will depend on the corpus the metrics' LMs were trained upon and the overlap in lexicon between the candidate summary and the pre-training corpus. BERT was pre-trained using 16Gb of text from the BookCorpus (Zhu et al. [2015]) and Wikipedia. BART and RoBERTa were both trained on the same 160Gb corpus which consists of the BERT corpus plus CC-News, OpenWebText and Stories (Gebhard and Hamborg [2020], Gokaslan and Cohen and Trinh and Le [2018]). It is likely that the LMs are exposed to some of the technical lexicon present in the arXiv and PubMed datasets through Wikipedia

[52]GitHub repository: `https://github.com/nltk/nltk`

although it is unclear how often. This is a potential reason why the metrics might not perform well on the arXiv and PubMed datasets.

5.1.2 Human-Metric Evaluation Correlations

As explained in section 2.4.2, human evaluation is the gold standard in text summarization. We can therefore compare the relative performances of the evaluation metrics by the degree to which they correlate with human judgement. For this, inspired by Böhm et al. [2019], we use the annotated CNN/DailyMail dataset introduced in section 3.2. With this dataset, we have 500 samples of CNN/DailyMail articles with four summarization-model produced summaries and the gold standard summary. For each sample we can therefore compute the correlation between the automatic metric scores for each of our evaluation metrics and the human evaluator scores to obtain per-sample correlations. We then take the mean over all samples to obtain performance by metric.

5.1.3 Quora Question Pairs

We would like our evaluation metrics to accurately grasp the semantic similarity between two texts and this is a NLU question. One dataset testing NLU is the Quora Question Pairs dataset (QQP, Sharma et al. [2019b]). As explained in section 3, QQP contains pairs of (often similarly worded) questions and the tasks is to identify if these questions are duplicates or not. This is a good test of the evaluation metrics' NLU as performing well in this task requires the metric to capture the semantic similarities between the pair of questions, rather than simply measuring the lexical similarities. Evidently, a strong summarization evaluation metric should outperform a weak one on this task.

5.1.4 Adversarial Analysis

This section describes a series of adversarial tasks we performed to further probe the effectiveness of the metrics. These tasks consist of corrupting a set of summaries and assessing how well the evaluation metrics can distinguish the un-corrupted from the corrupted summaries. As corrupting the summaries will (most likely) degrade their quality, the uncorrupted summaries should be awarded higher evaluation scores than the corrupted summaries. By comparing these on a per-summary basis, we can assess the robustness of our evaluation metrics to different forms of noise.

Data All of our evaluation metrics operate by comparing a hypothesis to a reference. Ideally we would have access to two reference summaries per sample when performing this analysis as the higher the quality of summary, the higher the probability that noise will make it worse. As discussed in section 7.1.3, Zhang et al. [2019a] found that `PEGASUS`'s summaries for the test set on the CNN/DailyMail dataset were not significantly worse than the human written reference summaries. This implies that these can be used as a proxy for the second set of reference summaries. Therefore, for this analysis we used the CNN/DailyMail test set with `PEGASUS`'s summaries acting as the hypothesis summaries. These were then corrupted to form a corrupted and uncorrupted hypothesis set.

We also repeat this analysis on the PubMed dataset. While there has not been a study demonstrating statistical parity between `PEGASUS`'s summaries and the gold standard for PubMed, qualitatively the summaries are of high quality and we believe that, by and large, applying the corruption techniques outlined below will make the summaries worse. In this thesis we are especially interested in long document summarization and the mean PubMed summary length is longer than the mean CNN/DailyMail summary length. Therefore it is of particular interest that we are able to reproduce any findings on the PubMed dataset.

Corruption Methods Our chosen methods of corruption were `BERT mask-filling`, `word-dropping` and `word permutation`. `BERT mask-filling` and `word-dropping` were inspired by the synthetic dataset generation process for pre-training `BLEURT` (Sellam et al. [2020]) and are outlined in section 4.1. For each of these methods, the input summary was tokenized and chunked into sequences of length ten and the corruption was performed once to each of these sequences. This method ensured that the corruption spans sentences and can be applied to full stops, therefore gauging the sensitivity of the metric to coherence and grammaticality.

`BERT mask-filling` is a denoising auto-encoding task whereby some of the input tokens are masked and a pre-trained `BERT` model is used as the decoder to infill these tokens. This introduces lexical variety but preserves the fluency and coherence of the document[53]. `Word-dropping` corrupts the summary by omitting tokens. Sellam et al. [2020] find this to be useful as it mimics some of the common "pathological" issues encountered with automatic summarizers (e.g. truncating summaries or incoherent generations). Finally, `word-permutation` switches the ordering of two adjacent tokens throughout the summary, testing the metrics' sensitivities to syntax. This metric should be biased in favour of the model-based metrics as contextual embeddings should be more sensitive to lexical ordering than the `ROUGE` metrics. Indeed, `ROUGE-1` cannot distinguish at all between the corrupted and uncorrupted summaries here as it is syntax-insensitive.

5.1.5 Benchmarking of Existing Studies

The focus of this section has been on establishing a superior metric for automatically evaluating summaries. We conjecture that `ROUGE` is an insufficient evaluation metric which is insensitive to lexical and syntactical subtleties, is unable to grasp semantic equivalence and correlates poorly with human judgement. We test these hypotheses in the subsequent chapters and results can be found in section 6.1. Using a poor evaluation metric can be biased against better models as it is insensitive to subtle improvements and this acts as a drag on the rate of progress in text summarization. It is our goal to establish a superior set of metrics. These can then be used to determine which is the current leading model for text summarization.

We report results for `BART` (Lewis et al. [2019]), `PEGASUS` (Zhang et al. [2019a]), `ProphetNet` (Yan et al. [2020]) and PGN (See et al. [2017]). The first three are the current state-of-the-art approaches to summarization while the PGN was state-of-the-art in 2018 but is now outdated so is useful as a weaker benchmark. The first three score very similarly using `ROUGE` and are difficult to discriminate from human-written summaries (see section 5.1.4); we conjecture that `ROUGE` is not sensitive enough to effectively evaluate these models' summaries.

[53]Here we used `distilroberta-base` from the huggingface transformers library Wolf et al. [2019] as it showed adequate performance and its tokenizer maintained capitalization. The choice of model here is not important as the objective is to corrupt the document and therefore a particularly strong model was not necessary.

We report using only the CNN/DailyMail dataset. This is the most-commonly reported dataset for summarization and is the only dataset for which all of the above models contain fine-tuned weights. We did not have the hardware available to train these architectures on a new dataset and therefore leave reporting on an additional dataset to future work.

5.2 Architecture Analysis

This section outlines our experimental methodology for the architectures component of this thesis. Here we give details on the pre-processing steps, the hyper-parameter search and experimental design for comparing the `LED` with `BART` and assessing the performance of the `LED`. We also introduce Random Starts analysis, a method of removing the bias towards shorter models as salient content is often distributed primarily in the beginning of the document.

5.2.1 Preprocessing

The preprocessing steps used to generate the datasets are outlined in section 3. As explained in section 5.1.1, `BART` uses the pre-trained `BARTTokenizer`, which we also use the `BARTTokenizer` for the `LED`. We do not perform any additional preprocessing to maximize comparability against existing studies.

5.2.2 Hyper-parameter Search

Zhang et al. [2019a] selected the fine-tuning parameters for `PEGASUS` using a comprehensive grid-search for each dataset. Given that a single epoch with batch size of one (the largest batch size that will fit on a 12 Gb GPU) takes between 30-45 hours and lacking the available hardware to repeat this approach, we used a more focused approach to hyper-parameter search. Where possible[54], we started with the `BART` hyper-parameters used by Lewis et al. [2019]. We identified a set of salient hyper-parameters and tested each of these individually. We conducted a thorough hyper-parameter search using the PubMed dataset and used these results to inform limited hyper-parameter search on the arXiv and CNN/DailyMail datasets using only the most significant hyper-parameters (e.g. learning rate). Hyper-parameters were selected using the validation sets of the respective datasets. We used Weights and Biases to track our experiments[55], a suite of developer visualization and tracking tools which assist with monitoring machine learning experiments.

5.2.3 LED vs BART

Recent studies comparing self-attention to approximate self-attention layers in Transformers (e.g. Beltagy et al. [2020], Kitaev et al. [2020]) have concluded that the two perform similarly. However, this hypothesis has not been tested for text summarization. Before proceeding with subsequent analysis, we must first establish that `BART` and the `LED` perform similarly and this is the focus of this section.

[54]We used the huggingface transformers library (Wolf et al. [2019]), whereas `BART` was trained using the fairseq library (Ott et al. [2019]). Some of the functionality was not available on huggingface during experimentation.
[55]`https://www.wandb.com/`

We begin with the 1,024-width BART model fine-tuned on the CNN/DailyMail dataset, `bart-large-cnn`. We modify this to create the 1,024-width LED using the procedure outlined in section 4.2.2. We use an attention window of 512 tokens in the LED as this is most comparable to BART's n^2 self-attention[56]. The full model configurations are detailed in Table D.2. We fine-tune both models for two epochs on the PubMed dataset and for one epoch on the arXiv and CNN/DailyMail datasets corresponding to the relative sizes of the datasets. Using a 12 Gb Nvidia GeForce GTX TITAN X GPU (provided by Imperial College London Department of Computing), these runs take 45, 40 and 65 hours respectively. Our strategy for selecting hyper-parameters is detailed in section 5.2.2.

5.2.4 LED performance

In section 5.2.3 we outlined our methodology for comparing between the LED and BART. We were also interested in the performance of different configurations of the LED. There are two hypotheses to test here: 1) using a "longer" model is beneficial for summarizing longer documents; 2) reducing the attention window size will moderately reduce performance, but not catastrophically so. To this end, we run two sets of experiments beginning with the LED-1024 model with 512 window size. The first set varies along the model size dimension; the second set varies along the window size dimension. These results are displayed in section 6.2.2.

The attention-window experiments were run using the Imperial College London 12 Gb Nvidia GeForce GTX TITAN X GPUs. The input-length experiments experiments exceed 12 Gb of GPU memory so are run using the 24Gb Nvidia RTX6000 GPUs provided by the Imperial College London High Performance Cluster (HPC). Experiments using the HPC have a 24 hour time limit so these we only ran these runs for this time[57].

LED profiling A key theoretical advantage for introducing the LED was to reduce the complexity in BART's self-attention layers from quadratic to linear. This will dramatically reduce the memory consumption for longer sequences. This section describes how we profile the memory consumption of the LED with respect to the input length and the attention window, the two primary memory-related configuration decisions. The goal of this analysis is to confirm that the memory complexity is indeed linear. To profile memory usage we logged the maximum memory consumption used during fine-tuning for different combinations of attention windows and input sizes. We used pytorch's *torch.cuda.max_memory_allocated()* to implement this. These results and discussion can be viewed in section 6.2.3.

5.2.5 Random Starts Analysis

This thesis has focused on the PubMed and arXiv datasets as they are the most studied long document summarization datasets. We use these datasets to test our hypothesis that using a "longer" Transformer input length will lead to better performance on long documents. The rationale being that the longer the model, the less the input document need be truncated and therefore the summary should capture the salient information throughout the document. However, this logic implicitly assumes that the salient information is distributed throughout the document. The arXiv and PubMed datasets may violate this assumption as

[56]Recall that the attention window is two-sided.

[57]As the RTX6000 machines are faster than the GeForce GTX TITAN X this normally equated to nearly one epoch.

they comprise of highly structured academic documents. The canonical skeleton of a scientific research paper consists of an introduction, body and conclusion. The introduction is ostensibly a summary of the remainder of the paper and therefore likely captures the most salient information.

The above reasoning casts doubt over whether increasing the input length of the `LED` will improve its performance. Conversely, it could actually worsen the model performance for two reasons: 1) if we assume that the salient information is included within the first 1,022 tokens then, if we use the `LED-4096` rather than the `LED-1024`, this effectively corresponds to using noisier data; 2) summarizing a longer document is more challenging than summarizing a shorter document as the information distillation requirements are greater.

Random Starts On the surface, it would seem that using a longer model is advantageous for longer document summarization. In light of the above reasoning however, there may in fact be a positive or negative relationship between model length and performance measured on arXiv and PubMed. To this end, we introduce a novel *Random Starts* (RS) task. The remainder of this section explains our methodology for this analysis.

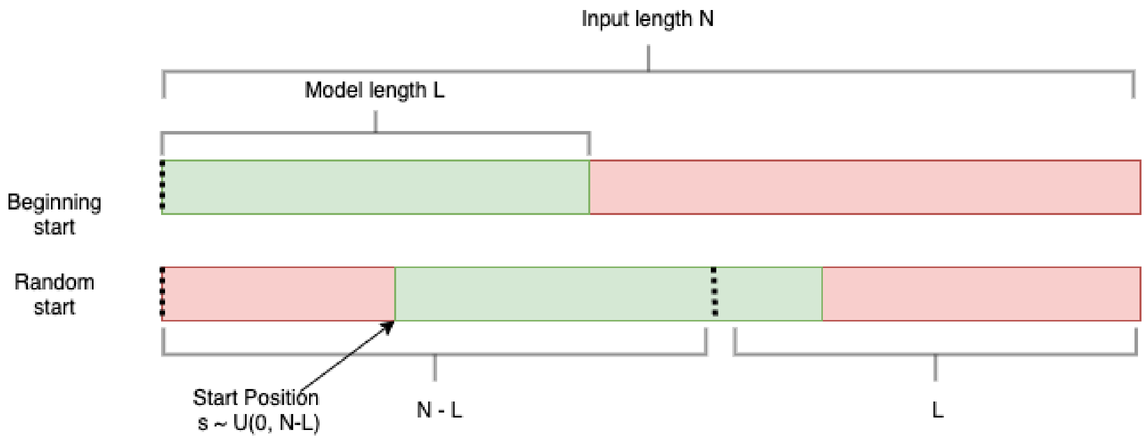


Figure 5.1: Schematic comparison of Beginning Starts versus Random Starts. These refer to different methods of truncating source documents which are longer then the model length when performing summarization. The green region is the input sequence for the model while the red region is truncated and discarded. Beginning Starts is the "normal" case where the input document is truncated at the end. This is the default case when using LMs for *seq2seq* With Random Starts, truncation occurs before and after the input sequence and the starting index is drawn from a random uniform distribution.

When using arXiv and PubMed, the problem is that the salient information is distributed at primarily the beginning of the document. RS tackles this by truncating the input document both at the beginning and the end rather than solely at the end as the conventional task would. This is illustrated schematically in Figure 5.1: the normal "Beginning Start" case takes the first L tokens, where L is model length. With Random Starts, any sequence of L adjacent tokens can be used as the model input, with the starting index S drawn from a random uniform distribution $S \sim U(0, N-L)$ and document length N[58]. This is a crude

[58]Note that this is only for the source document. If the target summary is truncated, this still occurs only at the end.

but effective method of ensuring that longer models have a higher probability of seeing the salient information than shorter models. Clearly we expect longer models to perform better on this task.

Algorithm 1: Random Starts Truncation

```
Result: Set of truncated documents, T = $\{t_1, \dots, t_n\}$
for source document $d_i \in D$ do
    Tokens to truncate $m_i = length(d_i) - model\ length\ L$
    Draw starting position $s_i,\ s_i \sim U(0, m_i)$
    $t_i = d_i[s_i : s_i + L]$
end
```

To implement this we modified the huggingface `BARTTokenizer` (Wolf et al. [2019]) to create a custom tokenizer. The tokenization process used by our custom tokenizer is described in algorithm 1. Applying this to the PubMed dataset, we obtain a distribution over the starting positions within the document (s_i in algorithm 1). This distribution is illustrated in Figure 5.2. The spikes at zero indicate the share of documents which are shorter than the model length and therefore begin at the first token (i.e. these are not truncated). As the model length increases it becomes longer than more of the documents and therefore the spike at zero becomes increasingly dominant. When using the `LED-4096`, over 65% of the documents are not truncated as they are shorter than the model, so the Random Starts and Beginning Starts cases converge as model length rises.

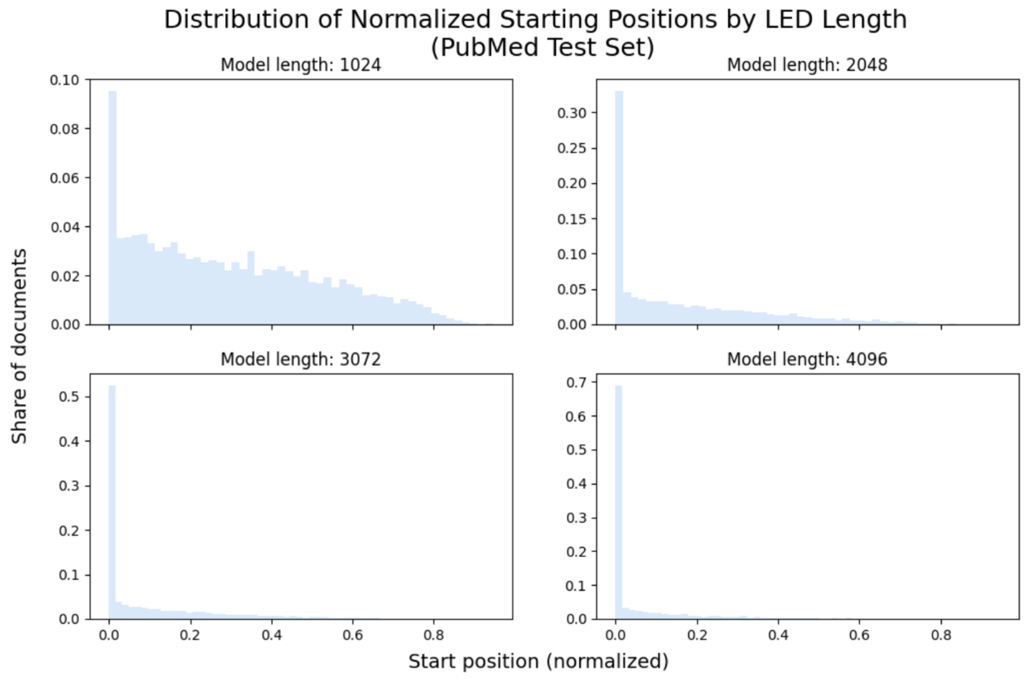


Figure 5.2: Histograms plotting the normalized starting position by model length. The normalized starting position is the starting index of the input sequence divided by document length, s_i / length(d_i). This therefore shows the percentage of the document which is truncated at the beginning. If a document is shorter then the model, it is not truncated and therefore the (normalized) starting position is zero, hence the spikes at zero. These plots show four different length versions of the `LED| and use the PubMed test set.`

5.2.6 RED vs LED

As mentioned in section 4.2.3, an open research question in text summarization is which approximate self-attention layer has optimal performance. This section outlines our methodology for comparing between the LED and the RED as two candidate approximate self-attention layers. We explained in section 4.2.3 that BART's trained self-attention weights are incompatible with Reformer self-attention, thus we cannot begin with pre-trained weights in the RED. This limits the scope of experiments we can perform as pre-training has been shown to increase model performance when using TED summarization models (e.g. Zhang et al. [2019a], Lewis et al. [2019]).

We do not have the hardware available to pre-train a set of compatible Reformer self-attention weights. To make a meaningful comparison between the LED and RED we therefore randomise the weights of the self-attention layers in the LED. Note that it is only the self-attention layers in the RED and LED which are randomised; the remainder of the model (e.g. feedforward layers, the decoder, the positional embedding matrices) still use BART's pre-trained weights. We then run fine-tuning using both models for three epochs on the PubMed dataset. Our final model configurations are in Table D.2 and results in section 6.2.5.

6 Results

6.1 Evaluation Experiments

A key objective of this thesis is to establish a set of new evaluation metrics which are superior to the ROUGE metrics. We outlined the selection of metrics used in section 4.1 and our experimental methodologies in section 5.1. Here we present the results from these experiments with corresponding analysis. It will be seen that BERTScore, BARTScore, Mover-1 and Mover-2 consistently outperform the ROUGE metrics on these tasks and therefore would be strong candidates to become the new prevailing metrics in text summarization. It will also be seen that BLEURT performs erratically and hence would not be a strong choice.

6.1.1 Metric Configurations

In section 4.1.1 we outlined each of the evaluation metrics used for this thesis. As discussed, several of the metrics use Transformer LMs to compute the embeddings which are then used to score the summary. For BERTScore, BARTScore and BLEURT there are several different LMs able to perform this role[59]; here we perform preliminary experimentation to decide which configuration performs best. We then used the best-performing configuration for all subsequent analysis in this thesis.

BERTScore & BARTScore The key configuration decision within BERTScore / BARTScore is the choice of LM to compute the embedded representation. The authors experiment with twelve BERT TE model flavours and find the 24-layer RoBERTa$_{\text{large}}$ (Liu et al. [2019]) correlated highest with human judgement (on English tasks). Recall in section 5.1.2 we explained our methodology for computing the correlation between human evaluator judgements and

[59]While it would be possible to change the LM within Mover-1/2 (Zhao et al. [2019]), this is technically more challenging than for the above metrics. Moreover, the key innovation with MoverScore is the solving of the WMD problem to compute the similarity between two texts. We therefore did not experiment with different LMs for MoverScore.

Model Name	Spearman ρ	Pearson r	Kendall τ
BERTScore			
roberta-large-mnli	**0.298**	**0.337**	**0.263**
roberta-large	0.285	0.299	0.254
albert-xxlarge-v2	0.268	0.291	0.232
albert-xlarge-v1	0.250	0.290	0.215
bert-large-uncased	0.227	0.253	0.198
roberta-base	0.221	0.243	0.197
distilroberta-base	0.174	0.188	0.154
BARTScore			
bart-large-mnli	**0.308**	**0.339**	**0.273**
bart-large	0.183	0.200	0.162
bart-large-cnn	0.111	0.130	0.010
bart-large-xsum	0.127	0.163	0.109

Table 6.1: Performance for different configurations of BERTScore and BARTScore on the annotated CNN/DailyMail task. The task assesses how well the scores given by each evaluation metric correlate with scores given by human evaluators. The figures reported are the correlations, shown for Spearman ρ, Pearson r and Kendall τ (Definitions of each of the correlation metrics can be found in appendix D). The model names correspond to those from the huggingface transformers library (Wolf et al. [2019]) and can be viewed at: `https://huggingface.co/models`. See section 5.1.2 for additional experiment details.

the evaluation metric score for summaries of CNN/DailyMail articles. Using this procedure we analysed the impact of different LM choices on BERTScore's ability to evaluate summaries.

The results of this analysis can be seen in Table 6.1. As shown, the `bart-large-mnli`[60] variant of BARTScore correlates best with human judgement. This is narrowly followed by the `roberta-large-mnli` version of BERTScore. These configurations were used for all subsequent analyses and latter references in this paper will assume these metrics are configured as such.

The strong performance of the metrics using models fine-tuned on the MultiNLI dataset is intuitive. The MultiNLI (Williams et al. [2018]) dataset is a Natural Language Inference (NLI) task which requires the model to classify whether one sentence entails another. This task is well-suited to fine-tuning the model within BERTScore as NLI requires strong NLU for good performance. Equally, a model with strong NLU should serve as a competent evaluation metric by grasping the semantic similarities between two texts. The model used for MoverScore is BERT fine-tuned on the MultiNLI dataset. Given that ourselves and Zhao et al. [2019] both separately find the MultiNLI dataset to be the most effective fine-tuning task for evaluation metrics this conclusion would seem robust.

In contrast, fine-tuning using a summarization task would not seem sensible here. Summarization teaches the model to condense information down to only the essential points and also to generate an output sequence. This means that the decoder will be trained to generate text but this is not useful when we want the model to produce a meaningful encoding of a sequence such that its semantics can be easily compared to another encoded sequence.

[60]The large version of BART fine-tuned on the MultiNLI dataset (Williams et al. [2018])

Model Name	Spearman ρ	Pearson r	Kendall τ
bleurt-large-512	**0.246**	**0.274**	**0.216**
bleurt-base-512	0.223	0.241	0.200

Table 6.2: Performance for different configurations of `BLEURT`. The task assesses how well the scores given by each evaluation metric correlate with scores given by human evaluators. The figures reported are the correlations, shown for Spearman ρ, Pearson r and Kendall τ (Definitions of each of the correlation metrics can be found in appendix D). See section 5.1.2 for experiment details.

Metric	Spearman ρ	Pearson r	Kendall τ
BARTScore	**0.308**	**0.339**	**0.273**
BERTScore	0.298	0.337	0.263
BLEURT	0.246	0.274	0.216
Mover-2	0.242	0.282	0.209
Mover-1	0.236	0.274	0.203
ROUGE-1	0.171	0.216	0.149
ROUGE-L	0.162	0.191	0.145
ROUGE-2	0.127	0.133	0.115

Table 6.3: Performance on the annotated CNN/DailyMail correlation task by evaluation metric. The task assesses how well the scores given by each evaluation metric correlate with scores given by human evaluators. The figures reported are the correlations, shown for Spearman ρ, Pearson r and Kendall τ (Definitions of each of the correlation metrics can be found in appendix D). None of the differences between the metrics are statistically significant as the standard deviations of all metrics were large (approximately 0.5), due to the relatively small dataset size and the volatility of computing correlations using only four data points. Definitions of each of the correlation metrics can be found in appendix D.

Additionally, summarization has many opportunities for over-fitting, such as prioritising the first few sentences of the input document due to lead bias in the case of news articles or sentences beginning with phrases such as "we conclude that...". This over-fitting means the model is unlikely to be transferable to act as an evaluation metric. Therefore, while it might sound sensible to fine-tune a summarization evaluation metric with a summarization task, the two tasks are orthogonal and therefore unlikely to yield good performance. This is reinforced empirically in Table 6.1 with the variants of `BARTScore` fine-tuned on summarization tasks (`bart-large-cnn` and `bart-large-xsum`) performing worst.

BLEURT As for `BERTScore` described above, we ran experiments to compare the relative performances of configurations. As shown in Table 6.2, the large version correlated better with human judgement and therefore we proceeded with using `bleurt-large-512` in subsequent analyses.

6.1.2 CNN/DailyMail Correlation Experiments

The results for the correlation experiments are displayed in Table 6.3. As expected, the `ROUGE` metrics correlate poorly with human evaluator scores and perform worse than all of the model-based metrics. Of the model-based metrics, `BARTScore` and `BERTScore` perform best with `BLEURT` and `MoverScore` clustered around the mid-point. This provides strong evidence in favour of our hypothesis that the model-based metrics will better reflect the true

Metric	Correlation			Binary Classification	
	ρ	r	τ	Acc.	F1
BLEURT	**0.492**	**0.490**	**0.416**	**0.725**	**0.617**
Mover-2	0.461	0.455	0.377	0.690	0.542
Mover-1	0.460	0.451	0.375	0.687	0.532
BERTScore	0.450	0.450	0.367	0.688	0.546
BARTScore	0.448	0.443	0.366	0.680	0.525
ROUGE-L	0.376	0.365	0.308	0.651	0.471
ROUGE-1	0.370	0.365	0.303	0.651	0.459
ROUGE-2	0.323	0.270	0.270	0.630	0.338

Table 6.4: Performance on the QQP binary classification task. The task is to determine if two questions are semantically equivalent or not. The first three columns indicate the correlation between the metric output score and the ground truth label. The coefficients represent Spearman ρ, Pearson r, Kendall τ and definitions of each of these can be found in appendix D. The final two columns correspond to the performance of a decision boundary using only the metric output as the solitary feature, i.e. how well would a classifier perform on this task if its only feature was the score given by the evaluation metric. We report the accuracy and F1 score here.

semantic similarity than the ROUGE metrics.

6.1.3 Quora Question Pairs Analysis

Table 6.4 displays the results of the QQP task. The first three columns are correlation coefficients between the metric output scores and the labels (reported for Spearman ρ, Pearson r, Kendall τ, see Appendix D for definitions). As an alternative view, we could ask how well a classifier would perform if it only uses the metric output as its solitary feature (i.e. train a decision boundary or a single feature logistic regression classifier using only the metric score). To do this, we split the data into 50% training and learned a decision boundary to predict the class of the remaining 50% test set. These results corresponds to the final two columns of Table 6.4. As the dataset is imbalanced (one third duplicates), F1 score is reported alongside accuracy as this is the harmonic mean of precision and recall and so implicitly accounts for differences in per-class accuracy.

The results in Table 6.4 corroborate our earlier finding that the model-based metrics outperform the ROUGE metrics. This supports our hypothesis that contextualized embeddings are beneficial for automatic evaluation metrics as semantic similarity is easier to determine in the embedded space than by performing surface-level lexical comparisons. Of the model-based metrics, BLEURT clearly performs the best with the remaining metrics clustered around the mid-point. BLEURT's strong performance on this task but average performance on the CNN/DailyMail correlation task in section 6.1.2 are not consistent and this is the first evidence of BLEURT's erratic performance across the tasks. This will be seen more in the next sections.

6.1.4 Adversarial Analysis

Table 6.5 contains the results for the three adversarial tasks described in section 5.1.4. Here we see that BERTScore and BARTScore are the best-performing metrics across both datasets. Additionally, by and large all of the model-based metrics outperform all of the ROUGE metrics

Metric	**Word-Dropping (%)**		**BERT Mask-Filling (%)**		**Word Permutation (%)**	
	CNN/DM	PubMed	CNN/DM	PubMed	CNN/DM	PubMed
BARTScore	**94.6**	**95.5**	**98.0**	**98.2**	**97.6**	**97.4**
BERTScore	92.9	**94.8**	95.4	95.7	**97.8**	**97.9**
Mover-1	86.2	88.6	84.7	88.3	91.9	93.1
Mover-2	83.2	84.6	82.1	85.5	87.7	89.5
BLEURT	70.4	49.8	82.2	86.0	92.8	92.4
ROUGE-1	78.2	78.6	73.5	89.6	00.0	00.0
ROUGE-2	74.4	87.8	65.7	88.8	78.5	90.2
ROUGE-L	77.0	77.3	71.4	85.4	53.8	57.0

Table 6.5: Mean accuracy by metric on the corruption tasks. These tasks apply various forms of corruption to a set of summaries. Each metric is then used to score the corrupted and uncorrupted versions of these summaries; the objective is to give the uncorrupted version a higher score. The results reported therefore shows the accuracy by metric on this task. Scores within 1% of the maximum score have been bolded. All standard deviations were small (less than 0.3%). We run experiments using the CNN/DailyMail and PubMed datasets here.

on each of the tasks and both datasets. These results corroborate the findings from section 6.1.2. These tasks were designed to test the sensitivity of the metrics to syntax, lexical discrepancies and other common pathologies present in automatically-produced summaries. These results indicate the model-based metrics are favourable along these dimensions.

We expressed concern in section 5.1.1 that the model-based metrics might perform poorly on the PubMed dataset because their LMs were not exposed to the biomedical lexicon during pre-training. This would prevent the models from forming meaningful embedded representations of these tokens and therefore the models would perform poorly. This does not appear to be the case here as the model-based metrics' performance stays roughly constant across the two datasets.

Of the ROUGE metrics ROUGE-2 performs best, particularly on the PubMed dataset. This is unsurprising given that PubMed consists of medical research papers containing specific medical terms. These often span several tokens so large phrases of the summaries will be extracted directly from the source document. When the corruption is applied to a single token from one of these medical terms it will disrupt the n-gram sequence and will be detected by ROUGE-2 (but not necessarily the other ROUGE metrics as they are less sensitive to word order).

The other notable result here is BLEURT's poor performance, particularly on the word-dropping and BERT mask-filling tasks. This is surprising given that these corruption methods were directly inspired by the methods used to generate the synthetic dataset BLEURT was trained on. Given the proximity of these tasks to BLEURT's pre-training, BLEURT was expected to excel on these tasks. This is another example of BLEURT's inconsistent performance across our evaluation experiments.

Model	BA	BE	M-1	M-2	BLEURT	R-1	R-2	R-L
BART	0.608	0.318	0.225	0.286	-0.287	**0.442**	0.211	**0.410**
	0.07	0.13	0.13	0.12	0.28	0.12	0.13	0.12
PEGASUS*	0.613	0.310	0.217	0.282	**-0.215**	0.433	**0.213**	0.403
	0.08	0.15	0.15	0.14	0.29	0.14	0.14	0.14
ProphetNet	**0.624**	**0.320**	**0.227**	**0.289**	-0.263	**0.442**	**0.212**	**0.411**
	0.07	0.13	0.14	0.13	0.28	0.13	0.14	0.13
PGN	0.554	0.186	0.117	0.187	-0.340	0.362	0.156	0.333
	0.09	0.16	0.14	0.13	0.28	0.13	0.12	0.13

Table 6.6: Reproducing the results from Lewis et al. [2019], Zhang et al. [2019a] and See et al. [2017] on the CNN/DailyMail summarization task. The mean score is reported with the standard deviation underneath. **BA** / **BE** are BARTScore / BERTScore, **M-n** is Mover-n and **R-n** is ROUGE-n. Scores within 0.01 of the highest score are bold. * Re-eval, results marginally lower than in Zhang et al. [2019a].

6.1.5 Benchmarking of Leading Architectures

Table 6.6 contains the results of the benchmarking analysis[61]. We provide boxplots of a selection of the metrics in Figure 6.1 for additional visibility of the distributions' statistics. Full distributions for each of the metrics' scores for ProphetNet's summaries are shown in Figure E.2.

Aside from BLEURT, the model-based metrics unanimously agree that ProphetNet is the leading approach on the CNN/DailyMail dataset, with split opinions over the relative performances of PEGASUS and BART. Our goal with this section is to establish a superior set of evaluation metrics and provide a barometer of the performances of the leading architectures in the field using these metrics for the CNN/DailyMail dataset. The information contained within Table 6.6 will serve as a useful reference for future summarization models to be benchmarked against using a wider range of evaluation metrics than the status quo.

Also notable here is that BLEURT does not rank ProphetNet as the best model contrasting with every other metric. This another occurrence of BLEURT's erratic predictions, reinforcing the instances seen in throughout this chapter. This will be discussed in further detail in section 7.1.1.

6.2 Architecture Experiments

This section contains the results of the architectures experiments outlined in section 5.2. We begin by establishing that the LED and BART perform comparably as anticipated. We then proceed to analyse in detail the performance of the LED. It will be seen that there is a large increase in performance of the LED provided using a longer model increases the chance that the model sees the salient content in the input document.

[61]There are some minor differences between the reported ROUGE scores here and the scores originally reported by the authors. These stem from using different ROUGE packages to generate the scores as discussed in section 5.1.1.

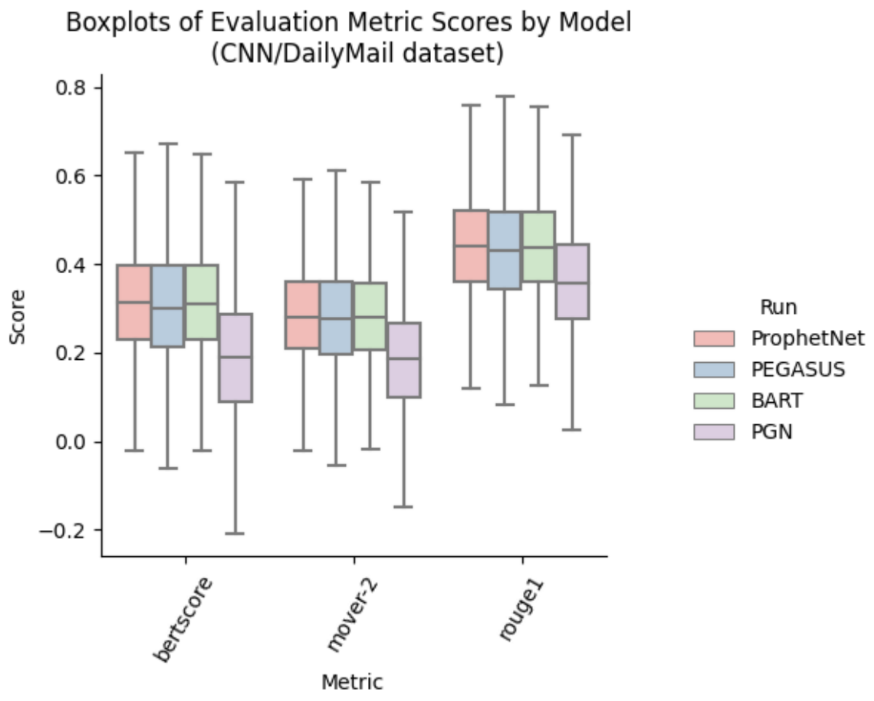


Figure 6.1: Boxplots of the distributions of scores illustrated in Table 6.6. Displayed here are the distributions of evaluation scores for each of the four architectures of interest. Scores are only shown for `BERTScore`, `Mover-2` and `ROUGE-1` for visibility.

6.2.1 LED vs BART

This section contains our results for the experiments comparing the `LED` with `BART`. The primary results is that the two models perform similarly, hence we conclude that using approximate self-attention is a viable strategy in text summarization. We elaborate upon these results in the remainder of this section.

As outlined in section 4.2.2, our hypothesis is that TED models can perform similarly well when using approximate self-attention layers compared to normal self-attention. We must first test this hypothesis before proceeding with using the `LED` on long document summarization tasks. Our experimental methodology was highlighted in subsubsection 5.2.3; this section contains the results of these experiments.

The results illustrated in Table 6.7 affirm our hypothesis. Here we show a side-by-side comparison for the `LED-1024` and `BART` for the CNN/DailyMail, PubMed and arXiv datasets. For each of the datasets and over each metric the mean scores are very similar, with `BART` performing better on PubMed and arXiv but the `LED` outperforms using CNN/DailyMail.

These results are unsurprising: n^2 self-attention and sliding window self-attention are similar when using a 512 window size with 1024 input size[62]. As shown in Figure 2.12, sliding window self-attention does not convolve (i.e. the first token only attends to the first 512 tokens; likewise the final token only attends to the final 512 tokens). The 513^{th} token attends to, and is attended to, by all tokens in the sequence. This makes the `BART` and the

[62]Recall that the attention window is double-sided.

Metric	BA	BE	M-1	M-2	BLEURT	R-1	R-2	R-L
CNN/DM								
BART	0.597	0.302	0.203	0.268	-0.249	0.425	0.204	0.394
	0.08	0.13	0.14	0.12	0.27	0.12	0.13	0.12
LED	0.599	0.303	0.203	0.269	-0.237	0.423	0.201	0.394
	0.08	0.13	0.14	0.12	0.27	0.12	0.13	0.13
PubMed								
BART	0.606	0.275	0.181	0.237	-0.047	0.437	0.188	0.389
	0.01	0.11	0.11	0.10	0.17	0.10	0.13	0.11
LED	0.603	0.270	0.172	0.229	-0.053	0.431	0.184	0.383
	0.01	0.11	0.11	0.10	0.17	0.10	0.12	0.11
arXiv								
BART	0.597	0.268	0.161	0.217	-0.091	0.444	0.166	0.389
	0.04	0.07	0.08	0.07	0.15	0.08	0.07	0.07
LED	0.597	0.265	0.156	0.213	-0.097	0.439	0.165	0.388
	0.04	0.08	0.09	0.08	0.16	0.08	0.08	0.08

Table 6.7: Performance for `BART` and `LED-1024` on the summarization task using the PubMed and CNN/DailyMail datasets. The mean metrics scores for the respective tests sets are reported with corresponding standard deviations underneath.

`LED` similar when using 1024 inputs; it is only when we increase the input length further that the models diverge. This will be explored later in this section.

Here we should highlight that we were unable to reproduce the results from Lewis et al. [2019] when fine-tuning `BART` on the CNN/DailyMail dataset ourselves, despite extensive and exhaustive experimentation. This accounts for the scores for `BART` on the CNN/DailyMail dataset being higher in Table 6.6 than in Table 6.7. This is because the authors originally fine-tune `BART` using the fairseq library (Ott et al. [2019]). Several of the features the authors originally use had not been released on the huggingface library (Wolf et al. [2019]) at the time of writing (e.g. layer-drop, attention dropout and label-smoothing). In addition, due to hardware constraints we used a much smaller batch size than the authors. However, even after including these features it appears there are differences between the underlying code-bases which prevent exact replication of the results using huggingface[63]. This likely accounts for our best-performing `BART` configuration on the CNN/DailyMail dataset being different to the author's best performing configuration, as can be seen in tables Table D.1 and Table D.2.

6.2.2 LED performance

In this section we display the results of the set of experiments probing the performance of the `LED`. There are two main results here: 1) using a longer version of the `LED` improves performance on the arXiv dataset but not using PubMed; 2) reducing the attention window size by a factor of 8 only reduces the performance of the `LED` by 3.5% on average. These results will be elaborated upon in the remainder of this section.

Section 5.2.4 outlined our methodology for assessing the performance of the `LED`. Here we examine the impact of changing the input length and the attention window size on the

[63]This is discussed in the following GitHub issues thread: `https://github.com/huggingface/transformers/issues/5654`

model performance. We provide the reader with samples of the `LED-1024` output summaries for the CNN/DailyMail test set in tables A.1, A.2, A.3, A.4. The corresponding evaluation scores for each metric are also provided. These contain four sets of samples: those scored highly by all metrics, those scored poorly by all, those scored well by the model-based metrics and poorly by `ROUGE` and vice versa.

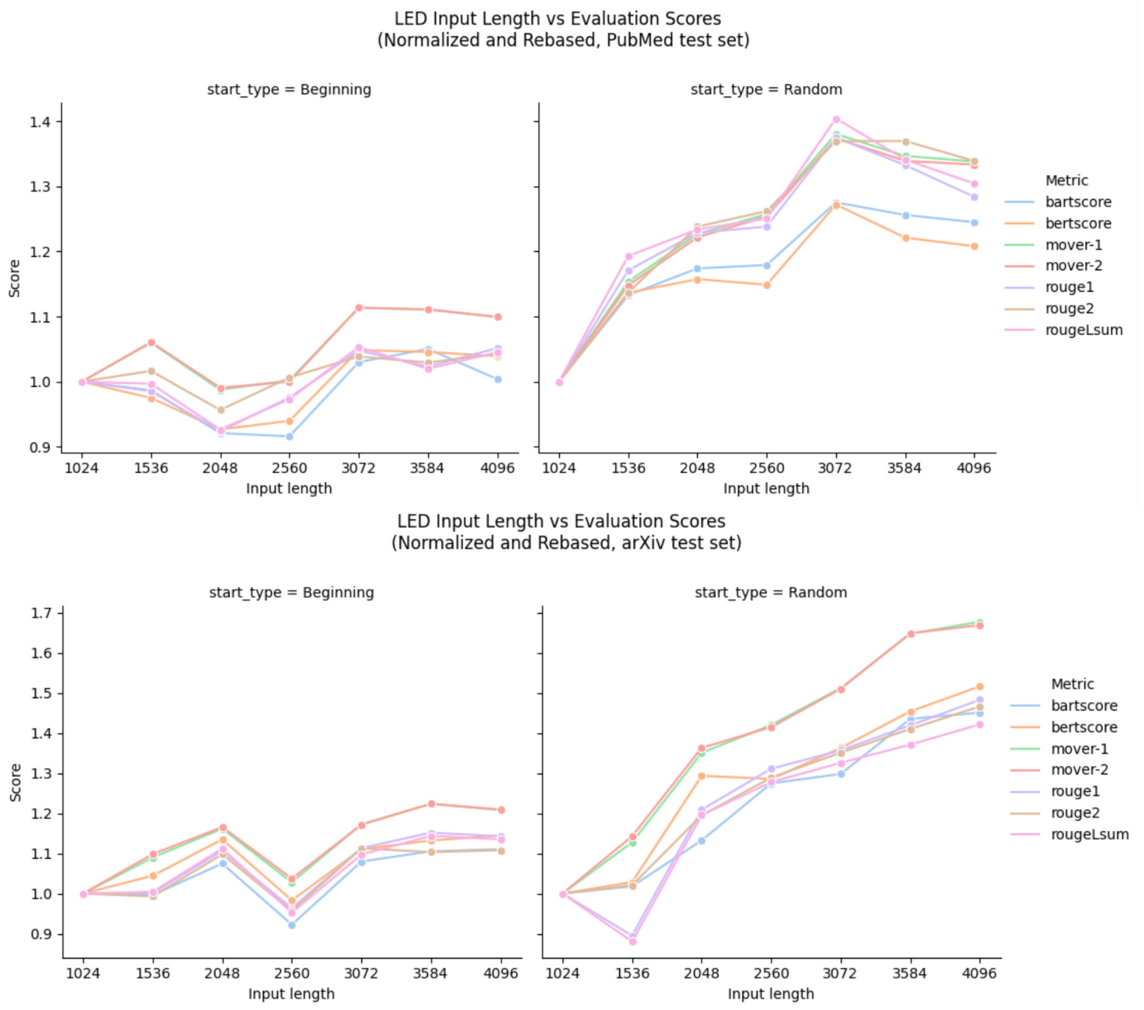


Figure 6.2: Analysis of the performance of the `LED` by model length on a summarization task. These results have been normalized by standard deviation and indexed for better comparability of the trends across the different metrics. The top figures show on the PubMed dataset, arXiv shown below. The left-sided plots show the Beginning Starts case and the Random Starts in the right-sided plots. The raw version of these results can be seen in tables E.3 and E.4. We exclude `BLEURT` to improve visibility.

Performance by input length Results are available in tabular format in Table E.3 and illustrated graphically in Figure 6.2. Figure 6.2 shows a side-by-side comparison of input-length against (normalized) metric scores when using Beginning Starts vs Random Starts (see section 5.2.5). on the arXiv and PubMed datasets. Currently we are analyzing the normal case (i.e. Beginning Starts) so consider only the left panels of Figure 6.2. Here we can see that there is at best a modest improvement in model performance when using longer model configurations on the PubMed dataset (top-left panel). In section 5.2.5 we hypothesized that longer models may not be helpful on the PubMed dataset as the salient information is clustered at the beginning of the document so using a longer input length surmounts to using noisier data. The results in Figure 6.2 support this hypothesis for the PubMed dataset.

In contrast, there is a stronger positive trend using the arXiv dataset. This suggests that the salient content is distributed more evenly throughout the documents on this dataset. It also shows that longer versions of the LED can outperform shorter versions of the model in some domains. This suggests that the benefits of using a longer context window can outweigh the increased information distillation challenges from using a longer model. It should also be highlighted that the LED performed particularly strongly on the arXiv dataset, surpassing the state of the art approach (PEGASUS) on this dataset. PEGASUS has ROUGE scores of 0.442/0.169/0.388 whereas our strongest model here, LED-3584, has ROUGE scores of 0.451/0.174/0.400. Recall that we only fine-tune each of the models for a 24 hour period to fit with the HPC scheduling and preserve comparability across the architectures. Given that we could likely achieve higher performance given longer training times, the performance of the LED on this dataset is particularly impressive.

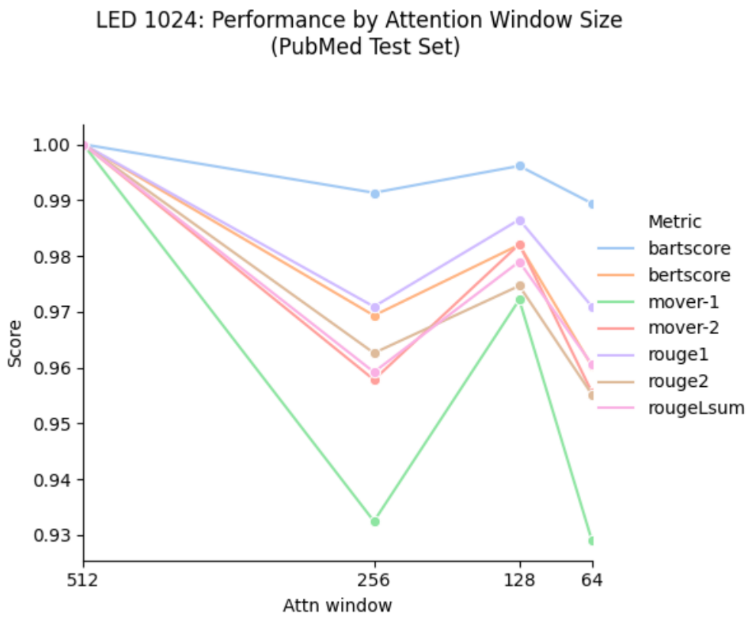


Figure 6.3: Performance of the LED-1024 on the PubMed summarization task by attention window length. Here the results have been normalized by dividing by standard deviation and rebased to increase the visibility of the trends across the different metrics. A raw version of this plot can be seen in Figure E.1. These scores are obtained using the test set after fine-tuning the models each for 24 hours on the PubMed dataset. These results are displayed in tabular format in Table E.1.

Performance by attention window These results can be seen in Figure 6.3 or in table format in Table E.1. As illustrated, reducing the attention window size has a marginal negative impact on the LED's performance. Cutting the LED-1024's attention window from 512 to 64 resulted in between a 1% and 7% fall in performance (metric-dependant). These results indicate that the model can perform better if the input length is increased while holding the attention window constant. This also support our hypothesis that the complexity of

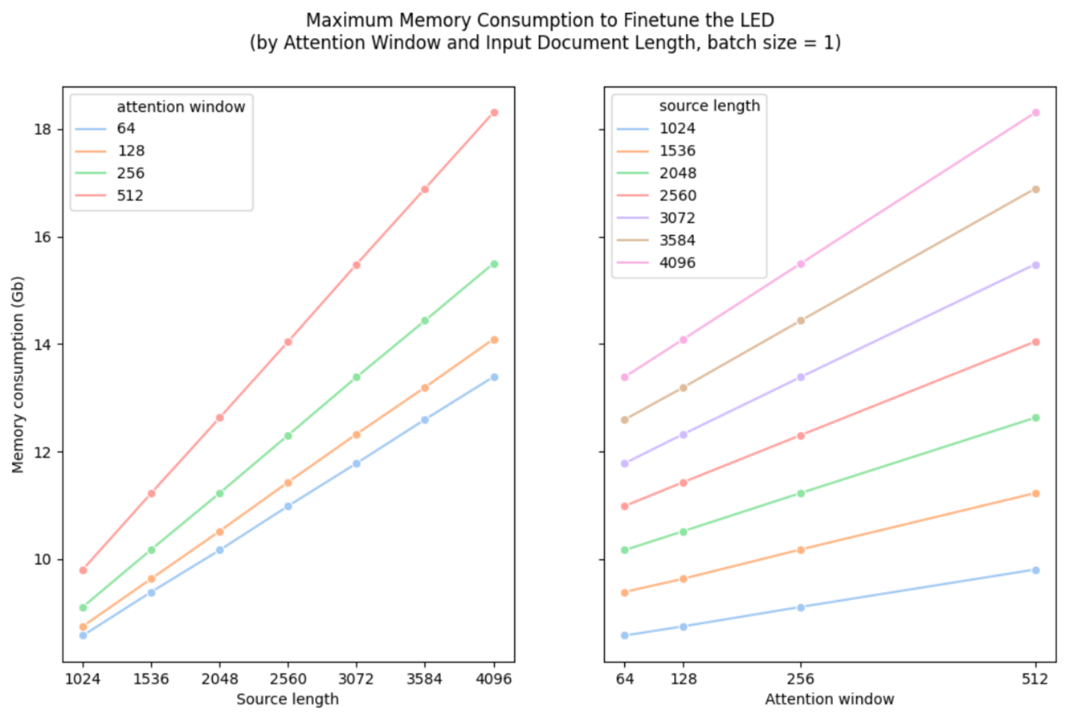


Figure 6.4: Results of profiling the memory consumption of the LED according to input length and attention window size while performing the PubMed summarization task. These results are displayed in tabular format in Table E.2.

Transformers' self-attention layers can be reduced without dramatically harming their performance.

This is significant as it greatly reduces the memory consumption of these models. This reduces the cost of long document summarization and allows considerably longer sequences to be used on standard hardware without truncation. For example, with an attention window of 512 and a batch size of 1, the maximum sequence length that can fit on a 12 Gb GPU is 1,024. With an attention window size of 64, sequence lengths of 2,560 tokens can fit onto the same hardware.

6.2.3 LED Profiling

In section 5.2.4 we discuss our methodology for profiling the memory consumption of the LED. Our goal here is to confirm that it does indeed have linear memory complexity. The results of the profiling analysis are illustrated in Figure 6.4. These results affirms that there is a linear relationship between the memory consumption and both the input length and the attention window size as anticipated.

Note that using sliding window attention should theoretically have an equal improvement on the asymptotic execution speed of the model as the self-attention layers also have quadratic time cost. In recognition of this, Devlin et al. [2018] pre-train BERT using sequence lengths of only 128 tokens for the first 90% of steps and the 512-length sequences for the remaining 10% of pre-training steps to speed up training. However, as noted in Beltagy et al. [2020], pytorch uses a high optimized GPU kernel for dense matrix multiplication.

This cannot be used for sliding window attention and using a native `pytorch` implementation dramatically reduces speed. With this in mind, Beltagy et al. [2020] design a custom GPU kernel to perform the sliding window attention computation, but this only matches the speed of the original self-attention computation and does not result in a speed-up. Given that memory is usually the bottleneck for Transformers rather than speed, this is not a limiting factor. With this in mind, we chose not to profile execution speed as we would not expect an improvement.

6.2.4 Random Starts Analysis

This section presents the results of the Random Starts analysis. The primary result here is that longer forms of the `LED` perform dramatically better than shorter forms of the `LED` when using Random Starts analysis. This is reflected by a 35% and 45% improvement[64] using the PubMed and arXiv datasets respectively. These results will be elaborated upon in the remainder of this section.

The results of the Random Starts analysis are shown graphically in Figure 6.2 and in tabular format in Table E.4. Considering Figure 6.2, the left and right panels correspond to the performance of the `LED` by input length when using Beginning and Random Starts respectively. As mentioned above, the top charts display results for the PubMed dataset and the bottom charts show for arXiv. This figure shows that when using Random Starts, model performance clearly improves when using longer inputs on both datasets. In contrast, when using Beginning Starts there is a clear but modest improvement in performance when using the arXiv dataset and no clear trend when using the PubMed dataset.

This plot provides strong evidence in favor of both of the hypotheses outlined in section 5.2.5. We conjectured that using a longer model would improve performance if the salient information is distributed throughout the document. We also hypothesised that this may not manifest for the PubMed dataset as the salient information is skewed towards the introduction section, thus biasing in favor of shorter models. Given that the `LED` only improves modestly with increasing input length when using Beginning Starts but improves markedly when using Random Starts, this suggests the above hypotheses are correct for the PubMed dataset. Using arXiv, we see an upward trend when using Beginning Starts and a strong and significant upward trend when using Random Starts. There seems clearer benefits to using a longer model on the arXiv dataset in any scenario, whereas this is only the case when using Random Starts for PubMed. This is likely due in part to the arXiv documents being on average 60% longer than their PubMed counterparts.

Note that Figure 6.2 is re-based as this better reflects the trend of input length versus output scores; it does not represent the differences in absolute performance. In absolute terms, the Beginning Starts case always outperforms the Random Starts case on both datasets, although the gap closes the longer the input length. This results can be seen using Table E.3 and Table E.4.

[64]The improvement is in standardized metric scores. The scores for each metric are standardized by dividing by the standard deviation to make results comparable across the evaluation metrics.

Model	BA	BE	M-1	M-2	BLEURT	R-1	R-2	R-L
RED	0.582	0.237	0.140	0.201	-0.112	0.408	0.164	0.366
	0.06	0.11	0.112	0.10	0.18	0.11	0.12	0.11
LED	0.569	0.213	0.110	0.173	-0.135	0.380	0.126	0.333
	0.06	0.10	0.10	0.09	0.18	0.10	0.10	0.10
LED (pre-trained)	0.603	0.270	0.172	0.229	-0.053	0.431	0.184	0.383
	0.01	0.11	0.11	0.10	0.17	0.10	0.12	0.11

Table 6.8: Comparison of performance of the LED with the RED on the PubMed summarization task. The performance of the pre-trained LED is also shown as a benchmark. Shown above is mean performance by metric on the PubMed test set with corresponding standard deviations underneath.

6.2.5 RED vs LED

Our methodology for comparing the LED and RED was outlined in section 5.2.6 and the results are shown in Table 6.8. The key results is that the RED outperforms the LED on all metrics. Recall that we could not use transfer learning with the RED as the hidden dimension using the pre-trained models was incompatible with BART. The results therefore compare the LED and RED both without pre-trained weights in the self-attention layers in their encoders. The results from the version of the LED using pre-trained weights is also shown as a benchmark.

The results show that the RED performs much better then the LED when neither has pre-trained weights. This holds for all eight of the evaluation metrics and is especially pronounced for some (e.g. over three ROUGE improvement using ROUGE-2). The benefits of transfer learning are clearly evident here as the pre-trained version of the LED significantly both models. Clearly we cannot draw any firm conclusions about how well a fully pre-trained RED model would perform by comparing its performance without pre-trained self-attention layers; however, this does indicate that different self-attention layers do result in different performance levels. In section 8.2 we propose several future promising research angles in text summarization. The analysis here indicates that a systematic comparison of different self-attention layers is an important and obvious topic to dictate the future research emphasis in the field.

7 Analysis

7.1 Evaluation Analysis

This section offers additional analysis to support the conclusions reached in section 6.1. Here we offer our opinion over the performance hierarchy of the metrics analysed in this thesis. We profile the metrics to show their run-times and memory usage before discussing some of the limitations of our analysis and some possible solutions.

7.1.1 Choosing Between the Metrics

Section 6.1 explained the results of the evaluation metric experiments. Here we saw that the model-based metrics were almost always superior to the ROUGE metrics, making a strong case to replace the ROUGE metrics with a selection of these model-based metrics. This section will address the question of which of the metrics are strongest.

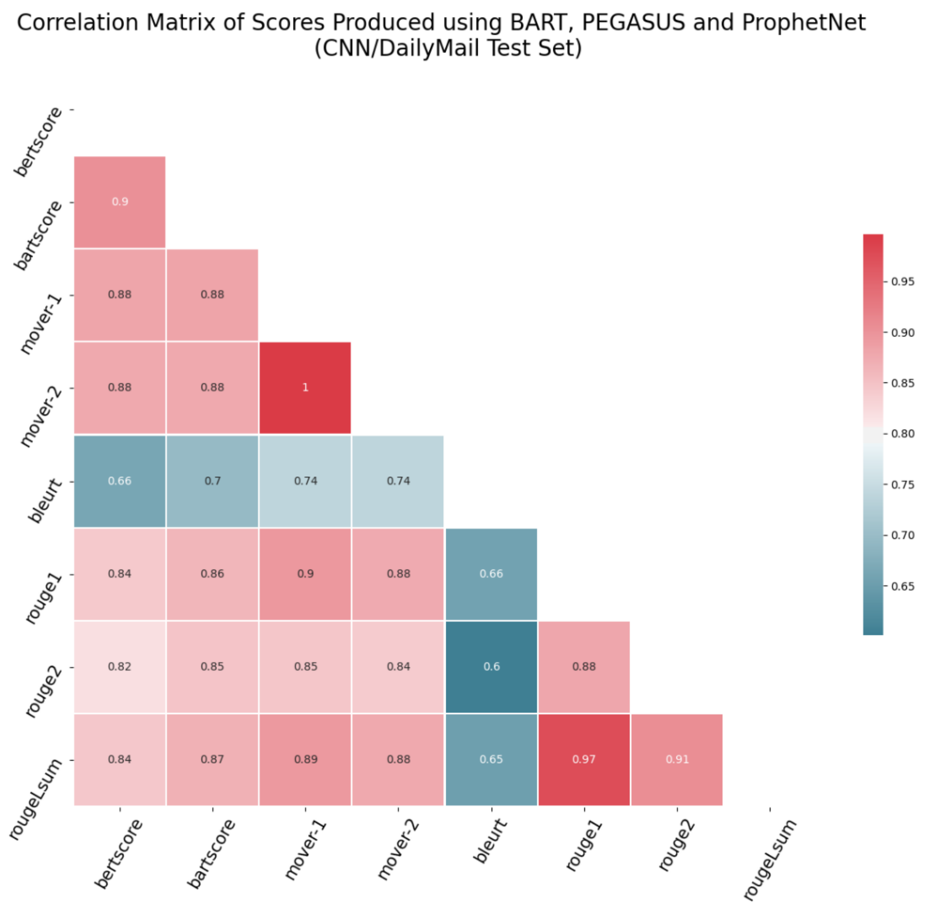


Figure 7.1: Correlation matrix of evaluation metrics' scores using the summaries produced by ProphetNet, BART and PEGASUS on the CNN/DailyMail test set.

It was a recurring theme in section 6.1 for BLEURT to behave erratically. It had the strongest performance on the QQP task from section 6.1.3, moderately performance on the CNN/DailyMail human correlation experiments in section 6.1.2 and poorly on the adversarial tests in section 6.1.4, surprising given the adversarial tests closely resembled BLEURT's pre-training task. Shown in Figure 7.1 is the correlation matrix of the evaluation metrics scores on the using the summaries produced by ProphetNet, BART and PEGASUS on the CNN/DailyMail test set. Here we see that in general the metrics' scores are highly correlated. Notably, BLEURT is the least correlated with every other metric. This reinforces the result seen in section 6.1.5, where BLEURT was the only metric to not award ProphetNet's metrics the highest mean score. While low correlation between all of the other metrics is not necessarily problematic in isolation, coupled with BLEURT's inconsistent performance on our tests it does becoming concerning. We therefore would recommend against using BLEURT in future research as it has not demonstrated reliably strong performance. This is a surprise as we expected BLEURT to perform well given its sophisticated and rigorous pre-training procedure.

Metric	Batch Size	Run-time (s / 1K)	Max Memory (Gb)
BARTScore	64	82	2.68
BERTScore	64	47	1.86
BLEURT	64	122	11.55
Mover-1	64	201	2.31
Mover-2	64	220	2.31
ROUGE-1	N.A.	12	N.A.
ROUGE-2	N.A.	11	N.A.
ROUGE-L	N.A.	12	N.A.

Table 7.1: Profiling maximum GPU memory use and run-time for each metric when evaluating summaries. The summaries are produced by the `LED-1024` on the PubMed dataset with a maximum length of 200 tokens. Run-time displays the time in seconds to score 1,000 summaries. Maximum memory use is found using `pytorch`'s *torch.cuda.max_memory_allocated()*.

Returning to Figure 7.1, we see that `Mover-1` and `Mover-2` are almost perfectly correlated with a value of 0.996. This result is not shared across all of the tasks seen in section 6.1 as the metrics do not perform identically on all of these tasks, although it is usually highly correlated. It therefore seems excessive to use both `Mover-1` and `Mover-2`; given `Mover-2` is slightly more correlated with human judgement on the CNN/DailyMail correlation experiments, we recommend this metric. Similarly, `BERTScore` and `BARTScore`'s scores are highly correlated in Figure 7.1. Choosing between them, we would recommend `BARTScore` as it performs better on the CNN/DailyMail correlation experiments and the adversarial experiments. We would recommend reporting over all of the metrics bar `BLEURT` for subsequent summarization research. However, if a more economical approach is desired, we would recommend using only `BARTScore` and `Mover-2`.

7.1.2 Metrics profiling

In Table 7.1 we profile the run-times and maximum GPU memory usage of each of the metrics. This is of interest because the model-based metrics use Transformer LMs which require GPU use (to avoid being excessively slow). This is a limitation of the model-based metrics when compared to `ROUGE` as `ROUGE` runs quickly on CPU. Of all of the metrics `BLEURT` has the largest memory requirement. This can be reduced by reducing the batch size but `BLEURT` is fundamentally memory-hungry when using the large model configuration. This memory requirement is another strong reason against using `BLEURT` as computing `BLEURT` scores on validation sets during training will require a substantial chunk of GPU memory be set aside. In contrast, the memory requirements of `MoverScore`, `BERTScore` and `BARTScore` are relatively modest. Of each of the metrics, `Mover-2` is the slowest. Given there are 6.4K summaries in the PubMed evaluation set, it takes `Mover-2` 24 minutes to process one pass, a notable time cost (although it can easily be reduced via parallelization).

7.1.3 Limitations

The two difficulties automatic evaluation metrics face are domain drift and quality drift (Sellam et al. [2020]). Domain drift refers to when the metric is applied to a different problem setting or dataset. We have taken steps to address domain drift in this thesis by using three different datasets (QQP, CNN/DailyMail and PubMed) and a range of tasks. As with all machine learning, however, results are dataset-dependant hence we cannot conclude that these results are repeatable in an entirely different setting, for example the summarization of entire books. Future work is to extend our experimental regime to other datasets to see how robust our conclusions are to new domains.

Quality drift refers to results becoming outdated because the underlying summarization systems improve. Quality drift is problematic in this instance as the models used to generate the summaries have been surpassed by the latest generation of Transformer-based summarization models. We mentioned in section 5.1.4 that Zhang et al. [2019a] conduct human evaluation experimentation and find that their model is not significantly worse than the reference summary by human evaluators. The premise of the current evaluation procedure is that the reference summary is the gold standard; once the model summaries become competitive with the reference summary it no longer makes sense to evaluate a model summary by comparing it to the reference summary.

In section 2.5.4 we highlighted a reinforcement-learning based approach robust to this issue. Using the Chaganty et al. [2018] dataset we used for our human correlation evaluation experiments, Böhm et al. [2019] train a reinforcement learning model to learn to score summaries using the "ground-truth" score provided by human evaluators. This remove the ceiling of the quality of human produced summaries as model summaries can be scored higher than the human summary. This approach may become more widespread future text summarization research, although the lack of suitably scored datasets here is currently an issue.

7.2 Architecture Analysis

In this section we provide some additional analysis to support the results seen in section 6.2. We observe the qualitative performance of the models by outlining examples for each dataset. We show that using approximate self-attention layers in the `LED` do result in linear space complexity as expected by profiling the `LED`'s memory use. And finally we provide an in-depth analysis into the performance of the `LED` by model length, giving additional insight into the conclusions reached in sections 6.2.2 and 6.2.4.

7.2.1 Qualitative Analysis

Section A contains samples of each of the datasets with their corresponding target (gold standard) summary and generated model summaries. The examples illustrate that each of the models produce highly coherent and semantically relevant summaries across all three datasets. The models adjust well to the diverse discourse styles across the three datasets. Previous studies (Zhang et al. [2019a], Lewis et al. [2019]) have performed blind human experiments and concluded that their summaries are not significantly worse than the target summary on the CNN/DailyMail dataset. We judge the `LED`'s summaries to be of a similarly high standard.

However, there is a clear quality gap between the human-written abstracts and the model summaries on the PubMed and arXiv. It is still an open research question of how to verify the factual authenticity of the generated summaries with erroneous claims being a widespread problem in the field. These issues are common for all of the models we experimented with on arXiv and PubMed. We conjecture this is in part due to the specific factual nature and in part due less commonality between the lexicon used in these articles with the pre-training corpus for LMs due to the often niche focus of the articles.

Focusing just on the LED, the model struggles to grasp seemingly simple cosmetic features of the dataset. For example, every PubMed article begins with a "$< S >$" token. However, some of our trained models struggled to reproduce this consistently and would often start articles erroneously, such as with "$<< S >$" or "$<$ 15...". Similarly, in the PubMed and arXiv target summaries in section A all full stops are succeeded by "$< /S > < S >$" tokens. None of the models we experimented with consistently captured these artifacts. We conjecture that Transformers struggle with the sequential element of these particularities as the major computational components of the Transformer (i.e self-attention or the feedforward projections) are permutation equivariant. We leave further investigation of this hypothesis for future work.

7.2.2 Random Starts Discussion

In section 6.2.4 we presented the results of the Random Starts analysis. Here we saw that using a longer model had a big performance improvement if we use Random Starts. However, there is a more modest performance improvement from using a longer model when using Beginning Starts.

One might ask why we are interested in the performance of a model when it is fed an input from a random position in a document. We would argue that in fact this analysis gives better insight as to how well a Transformer-based summarization model would perform if applied to "real-world" long document summarization tasks. In these scenarios it is unlikely that the salient information would be clustered at the beginning of the document and therefore truncating to 1,024 tokens would result in substantial information loss. Using a longer model would retain more information, but as the input length grows the challenge of distilling this information into a fixed length summary increases. Because these forces operate in opposite directions, it was an open research question as to how well long-version Transformer summarizers would work. The analysis presented above suggests that they would work well in practise.

7.2.3 Deep Dive Into Model Performance by Document Length

The Random Starts analysis outlined in sections 5.2.5 and 6.2.4 offers a good opportunity to gain additional insight into the characteristics of the LED. In this section we provide an in-depth analysis into the LED's performance with respect to document length and starting position within the document when using Random Starts, which we can compare with the Beginning Starts case. It will be shown that the LED improves equally on longer and shorter documents as we increase the model length when using the arXiv dataset. However, increasing the model length on the PubMed dataset trades off better performance on longer documents for worse performance on shorter documents.

Starting position vs model performance Recall in section 5.2.5 we outlined our methodology for truncating documents for our Random Starts analysis. We illustrated the resulting distribution over the normalized starting position by model length in Figure 5.2. The high-level results from the Random Starts analysis were explained in section 6.2.4. We saw that there is a strong positive effect of using a longer LED model on performance when using Random Starts, but a weaker impact when using Beginning Starts. Here we dive deeper into these results by analyzing the relationship between the normalized starting position and model performance to understand the key drivers behind these results.

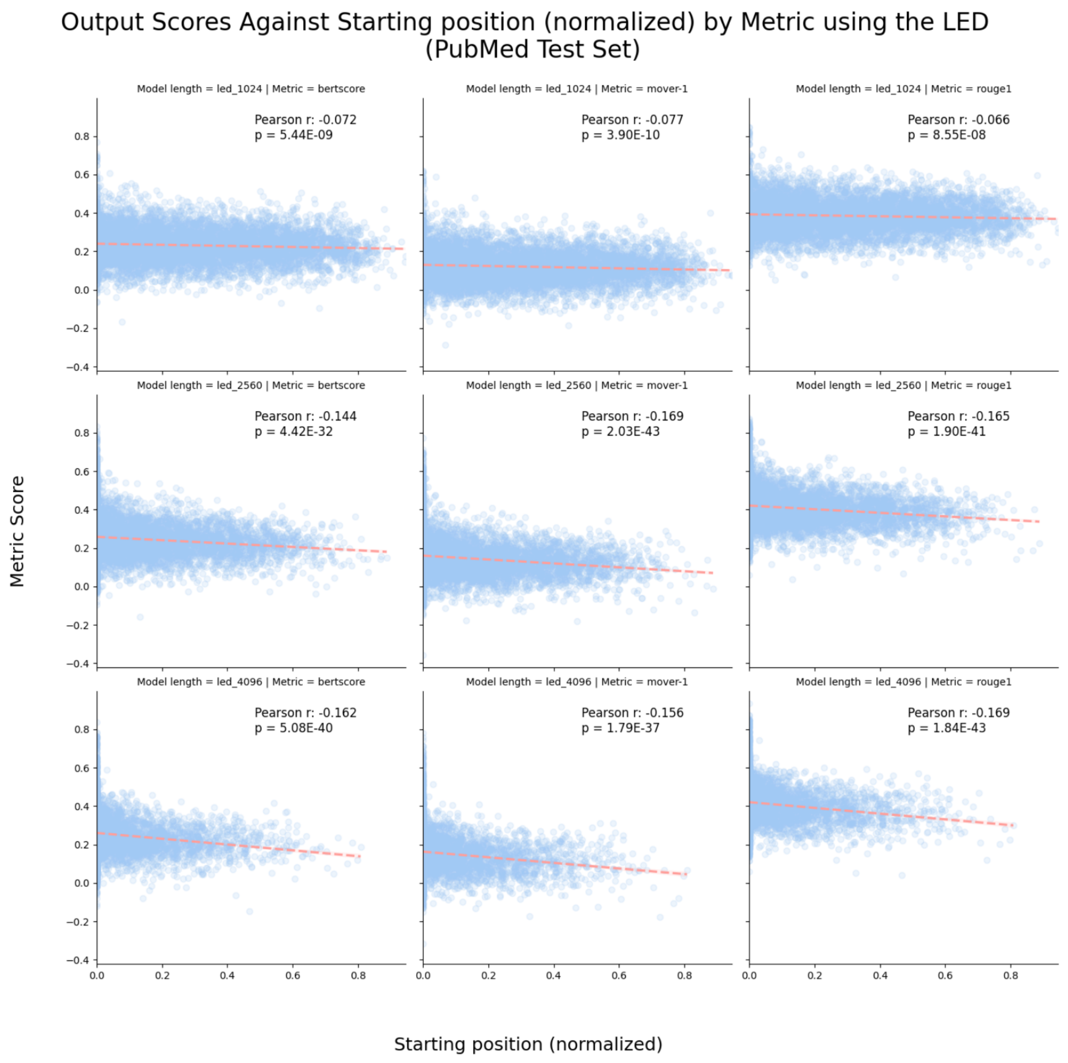
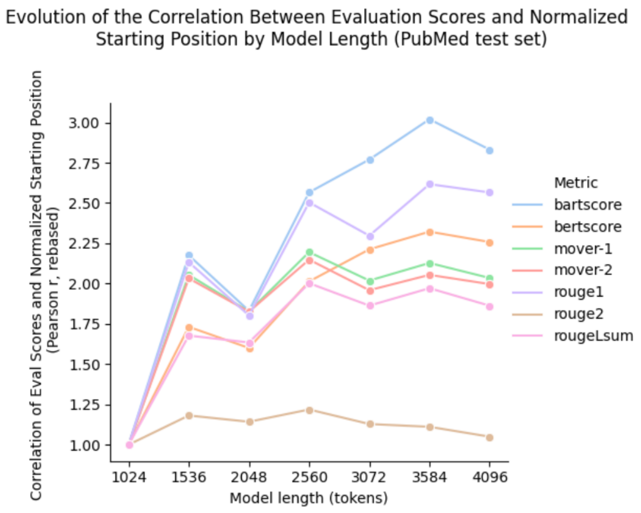


Figure 7.2: Scatter-plots of the normalized starting position against the evaluation metric score on the PubMed dataset. The columns display `bertscore`, `mover-1` and `ROUGE-1`; the rows display the 1024, 2560 and 4096 length `LED` models. The Pearson r correlation with associated p-stat is annotated for each subplot.

Figure 7.2 displays scatter plots of the (normalized) starting position in the source document against the metric score for the model's output summary conditioned upon evaluation metric and `LED` model length. Recall from section 5.2.5 that the normalized starting position indicates the share of the document truncated before the input sequence[65]. The Pearson r correlation is annotated for each subplot; a negative correlation here implies that the more tokens that are truncated from the beginning of the document, the worse the model performs. Given that these are negative for all subplot, this is further evidence in favour of our hypothesis that the PubMed dataset is biased toward shorter models as the salient content

[65]I.e. if a document is 2K tokens long and the normalized starting position is 0.25, the input to the model is the 500th - 1,500th tokens (assuming 1K model length).

congregates at the beginning of documents.



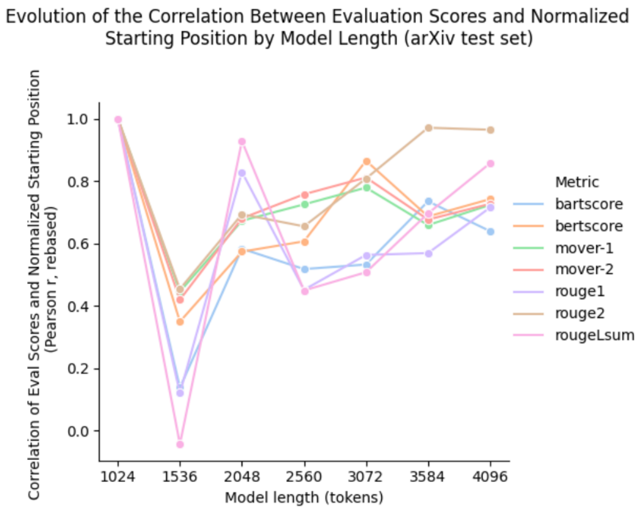


Figure 7.3: Analysis of the LED's performance on a summarization task when using Random Starts. The plots show the correlation between output score and normalized starting position for different length LED models. The top and bottom figures show for PubMed and arXiv respectively. These plots have been re-based to improve visibility of the trends.

Having established that models perform better if content is truncated at the end rather than the beginning of the document, we might be interested in how this factor depends on the model length, i.e. how does this Pearson r value change if we vary the model length. Intuitively, one would believe that increasing model length would make the model better at handling longer documents relative to short documents as less content is truncated overall. However, as the model length rises, the rarer it becomes for the model to encounter documents during training which have a significant truncation at the beginning[66]. Hence when

[66]This is evident from the plots of the distribution over starting positions shown in Figure 5.2: approximately 65% of the source documents do not get truncated for the LED-4096. Of those that do, few samples have long enough source documents for a significant portion of the introduction to be truncated. For 1K tokens to be truncated, the document must be at least 5K tokens long and the distribution shown in Figure 3.1 indicate that few PubMed documents are this long.

one such document arises at inference time, the produced summary will likely be poor relative to the non-truncated samples. This implies the negative correlation would strengthen with longer models. Given that the two hypotheses outlined above oppose one-another, it is unclear a-priori how this correlation statistic and the model length will interact.

To answer this question, observe the upper plot in Figure 7.3. This aggregates the Pearson r correlations (between starting position and output score) seen in Figure 7.2, rebases them and plots against LED model length. The top panel shows for the PubMed dataset with arXiv underneath. Here can clearly see the correlation gets stronger as model length increases. This means that longer models perform relatively worse on documents that are truncated at the beginning than shorter models do. This suggest the latter of the two hypotheses above is likely to be correct; i.e. that longer models see few truncated documents during training and therefore perform worse on these during inference.

We repeated this analysis using the arXiv dataset and this is shown in the bottom panel of Figure 7.3[67]. Here we see no clear trend between the model length and the strength of this correlation. This is to be expected with arXiv as the mean document length is much longer than in PubMed, and hence the longest models still are exposed to lots of beginning-truncated documents during training. As displayed in Table 3.1, 61% of arXiv documents are longer than 4,096 tokens compared to 24% of PubMed samples. Because of this, we do not find convincing support in favor of either of the two hypotheses outlined above.

Document length vs model performance Parallel to the above analysis, we can observe the relationship between the input document length and model output scores. As above, we can inspect how this evolves as we increase model length. This gives us a deeper insight understanding of the LED's characteristics with respect to different document lengths across different datasets.

Figure 7.4 plots the relationship between the LED model length and the correlation metric. The top plots are for the PubMed dataset with arXiv underneath. Here the correlation metric is between model performance and document length rather than normalized starting position as above[69]. We plot a side-by-side comparison for the Beginning Starts vs Random Starts case. Consider first the upper plots showing the PubMed tests. For the Beginning Starts case there is a weakening of the correlation between document length and model output score. This implies that longer models are relatively better at summarizing long document compared to short documents than shorter models are. Note that this only speaks of the correlation between document length and output score and does not reflect absolute model performance. We saw in Figure 6.2 that there is at best a modest relationship between the LED model length and absolute metric score on the PubMed dataset. Considering the arXiv equivalent in the panel below, the correlations are close to zero and there is no clear trend with respect to model length. This implies that all model lengths find longer and shorter documents equally difficult to summarize.

[67]The analogous plot to Figure 7.2 for the arXiv dataset is shown in Figure E.4

[68]These plots show the raw figures and have not been re-based (as for Figure 7.3, for example). This is because some metric scores cross the zero boundary therefore re-basing is problematic.

[69]We described Figure 7.3 as the aggregation of the scatter-plots displayed in Figure 7.2. The corresponding scatter-plots for Figure 7.4 can be found in Figure E.3.

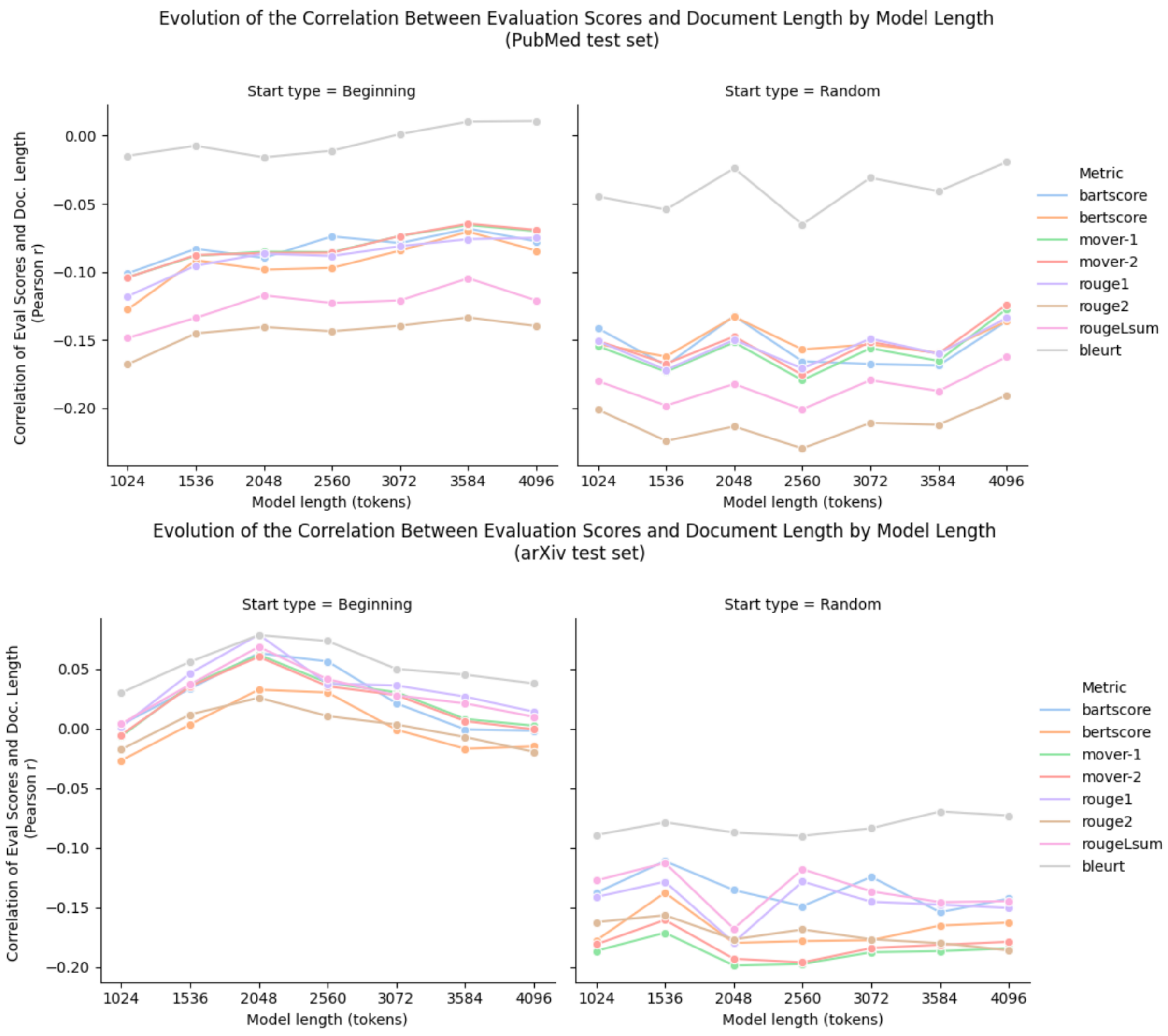


Figure 7.4: Plot showing the correlation between output score and document length against model length. The top plots show for the PubMed dataset, the bottom plots for arXiv. The left-sided panels illustrate when using Beginning Starts and the right-sided panels show for Random Starts[68].

Consider now the Random Starts experiments in the right-hand panels of Figure 7.4. We see immediately that all of the correlations are much lower when using Random Starts than beginning starts. This means that the models struggle relatively more on longer documents when using Random Starts than Beginning Starts. This is expected as longer documents are more likely to have salient content truncated than shorter documents when using Random Starts. We also notice that, on both datasets, this correlation does not exhibit any obvious trend as we increase model length. This tells us that the improvement seen in longer models' performance compared to shorter models when using random start is driven by improving performance across all document lengths, rather than being dominated by shorter or longer documents.

In this section we have provided an in-depth analysis into the performance of the LED with respect to document lengths over several dimensions. We have seen that using a longer model improves performance on the arXiv dataset across all lengths of documents. This is likely because salient content is distributed more evenly throughout arXiv documents. However, there is not a corresponding performance improvement on the PubMed dataset, likely driven by the salient content being present primarily in the beginning of the document. Given that longer LEDs is relatively worse on shorter documents than shorter LEDs, this implies that there is a performance cost to using a long LED model if only a short LED is required to preform effectively on this dataset.

8 Concluding Remarks

8.1 Contributions

This section highlights the key contributions of this thesis. Listed below are the key results and these are elaborated upon for the remainder of this section.

Key findings

- State of the art text summarization performance on arXiv, beating the incumbent, PEGASUS, by over one ROUGE[70]. This is in spite of our modest computational resources
- Develop a novel evaluation metric, BARTScore. Correlates approximately 2x better with human judgement than ROUGE[71]. Often performed best of all metrics we tested
- Establish a superior set of model-based evaluation metrics, consisting of BARTScore, BERTScore, Mover-1 and Mover-2. All shown to outperform ROUGE on five tasks spanning three datasets
- Demonstrate that sparse self-attention performs comparably to dense self-attention for summarization. This allows 2.5x longer sequences to fit onto standard hardware[72].

Evaluation metrics We have identified a set of model-based evaluation metrics which outperform the prevailing ROUGE metrics for text summarization. This set consists of our novel metric, BARTScore, as well as BERTScore, Mover-1 and Mover-2, all of which use Transformer LMs to compute contextual embeddings for the input sequence. These metrics have been rigorously tested using three datasets and on a summarization task, a NLU and three synthetic tasks. Our novel evaluation metric, BARTScore, Correlates approximately 2x better with human judgement than ROUGE[73]. We also find that BLEURT often performs erratically and poorly on these tasks and therefore recommend against using this metric for summarization. This thesis argues that these metrics should become the prevailing metrics in summarization. This would benefit the community by providing a more accurate measure of model performance and reducing the reliance on human-evaluation trials, in turn reducing the time and financial cost of running evaluation and increasing comparability between studies.

[70] PEGASUS was state of the art at the time of writing. Since then, Zaheer et al. [2020] have released Big Bird, an adaptation for PEGASUS for long document tasks. This has set a new state of the art on the arXiv dataset and our results do not beat these.
[71] See section 6.1.2.
[72] See section 6.2.2.
[73] See footnote 71.

LED We motivate the need for approximate self-attention layers Transformers to be successfully applied to long document summarization. This because of the $O(n^2)$ memory requirement of vanilla self-attention creates a memory bottleneck for long sequences. We introduce the `Longformer Encoder Decoder`, `LED`, as one such solution, combining the sliding-window self-attention from the `Longformer` with `BART`. We profile the model to show the memory complexity with respect to input length and attention window size is linear. We use the `LED` to summarize documents up to 4,096 tokens in length, a 4x increase from `BART`'s maximum input length of 1,024 tokens. We also demonstrate that sparse self-attention performs comparably to dense self-attention for summarization. This allows 2.5x longer sequences to fit onto standard hardware[74].

LED Experimentation We show the `LED` performs similarly to `BART` on three datasets. We then show that using a longer context window is beneficial to model performance one some tasks. This culminates with state of the art performance using the `LED-3584` on the arXiv dataset, beating the incumbent `PEGASUS`, (Zhang et al. [2019a]) by over one `ROUGE`[75]. Using a longer context window does not lead to a corresponding improvement on the PubMed dataset. This is because the salient information on PubMed dataset is displayed in the beginnings of the documents. We show this using our Random Starts analysis, demonstrating that if the document is truncated before and after the input sequence (as opposed to just after), then there is a strong advantage to using a longer model. This trend is clear on the arXiv and PubMed datasets. We therefore conclude that long document summarization can be improved by using longer inputs for TED summarizers.

RED We introduce a novel model, the `Reformer Encoder Decoder`, `RED`, as an alternative TED architecture for long document summarization. This model replaces `BART`'s self-attention with `Reformer` self-attention, reducing the asymptotic complexity from $O(n^2)$ to $O(n \log n)$. The pre-trained weights from `BART` self-attention layers are not compatible with `Reformer` self-attention and hence we cannot being with pre-trained layers as we have for the `LED`. We therefore compare this model against a version of the `LED` with randomised self-attention weights and show that the `RED` performs over 3 `ROUGE-2` better. This motivates the need to explore different combinations of base TED models and approximate self-attention layers in future research, as will be explained in section 8.2.

8.2 Future Work

Performance benchmarking of approximate self-attention layers In this thesis we compared the performance of sliding-window self-attention to dense self-attention using `BART` as the base model. We then contrasted sliding-window self-attention to `Reformer` self-attention and found they performed differently, albeit we only obtained a partial view because we did not use pre-trained weights when performing this comparison. In addition to the two mentioned above, there are several more approximate self-attention layers which have recently been released (e.g. Wang et al. [2020], Qiu et al. [2019], Child et al. [2019]). There is also the option to replace the base model of `BART` with `PEGASUS`, the state of the art model on long document summarization at the time of writing[76]. This give a wide range of possible

[74]See footnote 72
[75]See footnote 70
[76]It not be possible to use `ProphetNet` as the base model as it requires n-stream self-attention therefore requiring the approximate self-attention layers be drastically modified.

model configurations yet to be systematically explored and benchmarked.

Pre-training using approximate self-attention Section 6.2 showed that the LED performed similarly to BART on all datasets, and especially well on the arXiv dataset, beating the state of the art approach PEGASUS. If we had the hardware available, we would have pre-trained the LED following BART or PEGASUS's pre-training procedure as we expect there to be a significant performance improvement. We leave performing the entire pre-training pipeline using a "long" Transformer model such as the LED or RED to future work.

Additional datasets The conclusions of this thesis could be tested on the BIGPATENT dataset (Sharma et al. [2019a]), further testing the robustness to domain shift. Our methods could also be applied to the MultiNews dataset (Fabbri et al. [2019]) to explore their ability to perform multi-document summarization. As argued, however, this dataset has weaknesses and the summarization community lacks an adequate multi-document summarization dataset. Multi-document summarization could be viewed as the next frontier for the summarization community; creating an adequate dataset in this area would be valuable.

Use the evaluation metrics as the training objective function In section 2.5.4 we discuss some of the limitations of the maximum likelihood optimization procedure. In future work we would like to experiment with a modified fine-tuning objective function better suited to a summarization objective. This has been applied successfully using Reinforcement Learning (e.g. Böhm et al. [2019]) but we believe there is potential to use this method with supervised learning. This could be attempting to directly optimize the BARTScore score between the generated and target summaries. Alternatively, a LM could be used to compute embeddings for the target sequence, and the loss could be computed in the embedded space. The drawback of this approach is it is sample inefficient so has a larger compute requirement and therefore we leave it to future work.

8.3 Conclusion

In this thesis we have explored the summarization and automatic evaluation of long documents. We have explored a wider set of evaluation metrics to the prevailing ROUGE metrics and have established superior performance from a number of these. We have also experimented with potential solutions for using Transformers on long documents using approximate self-attention layers such as sliding window attention. We have shown that the LED performs comparably to its dense self-attention counterpart, BART, and performs considerably better on the arXiv dataset where a longer context window is beneficial.

Appendices

A CNN/DailyMail Summaries

Highest-scored summaries	
Targets	**Generations**
More details of the so-called 'everyday Americans' have been revealed. Gardener Julie Stauch was state campaign manager for Wendy Davis. Sean Bagniewski, who appears with his wife, has campaigned for Clinton. But he held an event for likely rival O'Malley days before video's release.	Julie Stauch, Sean Bagniewski and Vidhya Reddy all appear in Clinton's video. They are part of a diverse group of 'everyday Americans' chosen to show aspirational citizens as Clinton looks to speak to a wide demographic of voters. But far from being 'ordinary Americans', they are directly involved with the Democrat party - just like Jared Milrad and Nate Johnson.
Ellanora Arthur Baidoo has been trying to divorce her husband for several years. Husband doesn't have permanent address or permanent employment. Baidoo is granted permission to send divorce papers via Facebook.	Ellanora Arthur Baidoo has been trying to divorce her husband for several years, her lawyer says. A New York judge says Facebook is an acceptable way for her to serve her husband with a divorce summons. The couple never lived together, according to court documents.
Michael Gridley, 26, was jailed after running the scam at store in Basildon. Was sacked from position after supermarket received anonymous reports. But he is now employed as a manager at Lidl supermarket in Romford. Sentenced to 12 months at Southend Crown Court for leading role in scam.	Michael Gridley, 26, was jailed for a year at Southend Crown Court. He stole £15,000 worth of goods from Asda in Basildon, Essex. Stock including alcohol, cigarettes and DVDs were taken from the store. Gridley was sacked

Scores							
0.615	0.879	0.891	0.894	0.946	0.950	0.949	0.950
0.748	0.823	0.839	0.895	0.936	0.935	0.933	0.935
0.596	0.838	0.853	0.864	0.895	0.896	0.723	0.896

Table A.1: Three examples of highly scored summaries. These were rated in the top 1% by `BARTScore`, `Mover-1`, `ROUGE-1` and `ROUGE-L`. Summaries from the CNN/DailyMail test set and produced by the `LED-1024`. The evaluation scores for each of the summaries is shown below; these correspond to `BLEURT, Mover-1, Mover-2, BERTScore, BARTScore, ROUGE-1, ROUGE-2` and `ROUGE-L` (in order).

Lowest-scored summaries	
Targets	**Generations**
Kenya's security has been bogged down by concerns over civil rights. Kenyan Muslims have been targeted in raids and robbed, says Human Rights Watch.	Al-Shabaab killed 147 people at a college campus in Garissa, Kenya, on Thursday. The number of people killed is plaguing Kenyans with self-doubt, CNN's David McKenzie says. Kenya's politicians and public have struggled with these ideas...
Indiana town's Memories Pizza is shut down after online threat. Its owners say they'd refuse to cater a same-sex couple's wedding.	Memories Pizza in Indiana is at the center of the debate over the state's Religious Freedom Restoration Act. "If a gay couple was to come and they wanted us to bring pizzas to their wedding, we'd have to say no," owner says. Critics say the law
Mohonk Mountain House is a 'castle' retreat 90 minutes from New York. The hotel sits blissfully on the banks of Lake Mohonk in the Hudson Valley. The hotel was originally built as a drinking inn 145 years ago before Quaker twins Albert and Alfred Smiley made it a dry retreat - the bar is now open.	Mohonk Mountain House is a faux-gothic Victorian castle in the heart of the Hudson Valley. The lake, gardens and trails are a vast adventure playground for all ages. The 360-degree views are inspirational and the kids' club is the best we have

Scores							
-0.903	-0.142	-0.051	0.011	0.397	0.051	0.000	0.051
-0.699	-0.083	0.003	0.016	0.436	0.136	0.023	0.091
-0.901	-0.214	-0.102	0.036	0.358	0.070	0.000	0.070

Table A.2: Three examples of poorly scored summaries. These were rated in the bottom 1% by BARTScore, Mover-1, ROUGE-1 and ROUGE-L. Summaries from the CNN/DailyMail test set and produced by the LED-1024. The evaluation scores for each of the summaries is shown below; these correspond to BLEURT, Mover-1, Mover-2, BERTScore, BARTScore, ROUGE-1, ROUGE-2 and ROUGE-L (in order).

Highly scored by model-based metrics, poorly scored by ROUGE	
Targets	**Generations**
Liverpool scouts have been impressed by Geoffrey Kondogbia this season. The midfielder was one of the most coveted youngsters in Europe. France international joined Monaco from Sevilla in 2013 for £17million. Liverpool remain in the frame for James Milner and Danny Ings.	Liverpool are watching Monaco midfielder Geoffrey Kondogbia. France international has impressed in Europe and Ligue 1 this season. Real Madrid, Manchester United, Juventus and PSG were all keen. Brendan Rodgers' side are also interested in Danny Ings and James Milner.
Jeremy Trentelman, 36, of Ogden, built fort for young son and daughter. He received letter one day later saying it violated ordinance against waste. Father plans on keeping castle up for 14 days before he receives fine.	Jeremy Trentelman, 36, of Ogden, Utah, last week built a giant box fort for his son Max, 3, and daughter Story, 2, that included trap doors and a small slide. The father, who works as a florist arranging intricate displays...
Sir Bradley Wiggins left Team Sky after Paris-Roubaix on April 12. Tour de Yorkshire begins in Bridlington and finishes in Leeds from May 1-3. Wiggins' eponymous team is completed by Steven Burke, Mark Christian, Andy Tennant, Owain Doull and Jon Dibben.	Bradley Wiggins will ride for his eponymous team in the Tour de Yorkshire. The 2012 Tour de France winner was not selected in Team Sky's 2014 squad. The Tour begins in Bridlington and finishes in Leeds on May 3. It is a legacy of the Grand Depart

Scores							
-0.022	0.285	0.346	0.174	0.634	0.356	0.169	0.329
0.057	0.275	0.337	0.435	0.644	0.343	0.206	0.323
0.006	0.372	0.446	0.398	0.649	0.327	0.104	0.327

Table A.3: Three examples of summaries scored well by the model-based metrics but scored poorly by the `ROUGE` metrics. These were rated in the top third of all summaries by `BARTScore` and `Mover-1` but in the bottom third by `ROUGE-1` and `ROUGE-L`. Summaries from the CNN/DailyMail test set and produced by the `LED-1024`. The evaluation scores for each of the summaries is shown below; these correspond to `BLEURT`, `Mover-1`, `Mover-2`, `BERTScore`, `BARTScore`, `ROUGE-1`, `ROUGE-2` and `ROUGE-L` (in order).

Poorly scored by model-based metrics, highly scored by ROUGE **Targets**	**Generations**
Father-of-three Craig Sytsma was mauled to death in Michigan last year. Dog owners Sebastiano Quagliata and wife Valbona Lucaj agreed to plea deal Friday to around 15 years in jail, though judge could add six months. Sytsma was jogging in July 2014 when the two cane corsos attacked him. He was bitten almost ten times and was'screaming and begging' for help.	Sebastiano Quagliata and wife, Valbona Lucaj, pleaded no contest to owning a dangerous dog causing death in the fatal mauling last summer of Craig Sytsma of Livonia, Michigan. 'I'm so sorry,' Lucaj told Sy...
YouTube user Serpentor filmed his feline friend in action. Footage shows the tabby producing bizarre noises as she is petted.	YouTube user Serpentor filmed his feline friend in action. Footage shows tabby producing a range of unusual gurgling noises as she is petted. Her owners are heard laughing in the background as they watch her. To date the clip of her singing has been...
Little Catalina from America was filmed emptying out the contents of the kitchen cupboards at home. But when her father tells her to clean up her mess, she vehemently refuses with a heated - and rather cute - argument ensuing.	Catalina from America was filmed emptying out the contents of the kitchen cupboards at home. But when her father tells her to clean up her mess, she vehemently refuses with a heated - and rather cute - argument ensuing. 'I already cleaned the kitchen, no it...

Scores							
-0.516	0.143	0.190	0.210	0.541	0.484	0.247	0.462
-0.358	0.147	0.188	0.282	0.550	0.533	0.273	0.511
-0.510	0.120	0.186	0.199	0.555	0.500	0.184	0.480

Table A.4: Three examples of summaries scored poorly by the model-based metrics but scored well by the `ROUGE` metrics. These were rated in the bottom third of all summaries by `BARTScore` and `Mover-1` but in the top third by `ROUGE-1` and `ROUGE-L`. Summaries from the CNN/DailyMail test set and produced by the `LED-1024`. The evaluation scores for each of the summaries is shown below; these correspond to `BLEURT`, `Mover-1`, `Mover-2`, `BERTScore`, `BARTScore`, `ROUGE-1`, `ROUGE-2` and `ROUGE-L` (in order).

B Dataset Samples with Summaries

CNN/DailyMail	
Source:	(CNN)Sky watchers in western North America are in for a treat: a nearly five-minute total lunar eclipse this morning. Here's how it's unfolding: . It started at 3:16 a.m. Pacific Daylight Time, when the moon began moving into Earth's shadow. For the next hour and 45 minutes, that shadow will move across the moon and engulf it at 4:58 a.m. Pacific Time. The total eclipse will only last four minutes and 43 seconds, and NASA says that makes it the shortest one of the century. Watch it live on NASA TV . While people west of the Mississippi River will have the best view, at least a partial eclipse will be visible across the nation. But sunrise will interrupt the show on the East Coast. Parts of South America, India, China and Russia also will be able to see the eclipse, but it won't be visible in Greenland, Iceland, Europe, Africa or the Middle East. A lunar eclipse happens when the sun, Earth and moon form a straight line in space, with the Earth smack in the middle. The sun shines on the Earth and creates a shadow. As the moon moves deeper into that shadow, it appears to turn dark and may even appear to be a reddish color. Why red? Because Earth's atmosphere is filtering out most of the blue light. Some people have nicknamed the effect the "blood moon." NASA says lunar eclipses typically happen at least twice a year, but this eclipse is the third in a series of four in a row, known as a "tetrad.". The first was on April 15, 2014. The second was in September 2014, the next is Saturday and there will be one more, on September 28. If you want to learn more about the eclipse, NASA astronomer Mitzi Adams will take questions on Twitter ASA_Marshall. Did you see the total lunar eclipse? Share your photos with CNN iReport.
Target:	The total eclipse will only last 4 minutes and 43 seconds . People west of the Mississippi River will have the best view . Parts of South America, India, China and Russia also will see the eclipse .
LED-1024:	A nearly five-minute total lunar eclipse will be visible in western North America. It will only last four minutes and 43 seconds, and NASA says it's the shortest one of the century. Parts of South America, India, China and Russia will also see the eclipse.
BART:	The total lunar eclipse started at 3:16 a.m. Pacific Daylight Time. People west of the Mississippi River will have the best view. Parts of South America, India, China and Russia also will be able to see the eclipse. It won't be visible in Greenland, Iceland, Europe, Africa or Middle East.
PEGASUS:	A lunar eclipse happens when the sun, Earth and moon form a straight line in space . A lunar eclipse is the third in a series of four in a row, known as a "tetrad"
ProphetNet:	it started at 3 : 16 a . m . pacific daylight time , when the moon began moving into earth 's shadow . for the next hour and 45 minutes , that shadow will move across the moon and engulf it . the total eclipse will only last four minutes and 43 seconds .

Table B.1: A sample CNN/DailyMail article with model summaries.

arXiv	
Source:	the brain is a computational device for information processing and its flexible and adaptive behaviors emerge from a system of interacting neurons depicting very complex networks @xcite . many biological evidences suggest that the neocortex implements a common set of algorithms to perform “ intelligent ” behaviors like learning and prediction . in particular , two important related aspects seem to represent the crucial core for learning in biological neural networks : the hierarchical information processing and the abstraction process @xcite . the hierarchical architecture emerges from anatomical considerations and is fundamental for associative learning (e.g. multisensory integration) . the abstraction instead leads the inference of concepts from senses and perceptions (fig . [fig:1]d) . + specifically , information from sensory receptors (eyes , skin , ears , etc .) travels into the human cortical circuits following subsequent abstraction processes . for instance , elementary sound features (e.g. frequency , intensity , etc .) are first processed in the primary stages of human auditory system (choclea) . subsequently sound information gets all the stages of the auditory pathway up to the cortex where higher level features are extracted (fig . [fig:1]e - f) . in this way information passes from raw data to objects , following an abstraction process in a hierarchical layout . thus , biological neural networks perform generalization and association of sensory information . for instance , we can associate sounds , images or other sensory objects that present together as it happens in many natural and experimental settings like during pavlovian conditioning . biological networks process these inputs following a hierarchical order . in a first stations inputs from distinct senses are separately processed accomplishing data abstraction . this process is repeated in each subsequent higher hierarchical layer . doing so , in some hierarchical layer , inputs from several senses converge showing associations among sensory inputs . + recent findings indicate that neurons can perform invariant recognitions of their input activity patterns producing specific modulations of their synaptic releases @xcite . although the comphrension of such neuronal mechanisms is still elusive , these hints can drive the development of algorithms closer to biology than spiking networks or other brain - inspired models appear to be . + in this work , we propose a learning framework based on these ...
Target:	$< S >$ the human brain processes information showing learning and prediction abilities but the underlying neuronal mechanisms still remain unknown. recently, many studies prove that neuronal networks are able of both generalizations and associations of sensory inputs. $< /S >< S >$ + in this paper, following a set of neurophysiological evidences, we propose a learning framework with a strong biological plausibility that mimics prominent functions of cortical circuitries. $< /S >< S >$ we developed the inductive conceptual network (icn), that is a hierarchical bio - inspired network, able to learn invariant patterns by variable - order markov models implemented in its nodes. $< /S >< S >$ the outputs of the top - most node of icn hierarchy, representing the highest input generalization, allow for automatic classification of inputs. $< /S >< S >$ we found that the icn clusterized mnist images with an error of 5.73% and usps images with an error of 12.

Table B.2: A sample arXiv article with model summaries (part 1). The source has been truncated from 4,522 tokens to 400 for readability.

arXiv (continued	
LED-4096:	$< S >$ we propose a general neural learning framework, called inductive conceptual network (icn), where biological neurons are individually identified by nodes and invariant recognition is performed by variable - order markov models (vmm). the icn represents a general biological plausible model of the learning mechanisms in neuronal networks, and we tested the accuracy of this network on the mnist and usps datasets. $< /S >$ ¡ s¿ we find that the neural network exhibits a complex hierarchy of node nodes, which exhibits invariant pattern recognition by a hidden markov model algorithm implemented into a set of software libraries specialized in image processing (hierarchical temporal memory (htm))).
BART:	$< S >$ the inductive conceptual network (icn) represents a general biological plausible model of the learning mechanisms in neuronal networks. $< /S > < S >$ the icn is based on a set of considerations primarily from the memory - prediction framework, which is a generalization of jeff hawkins _ on intelligence_. we propose a learning framework based on these biological considerations and we tested the accuracy of this network on the mnist and usps datasets. in this framework, we find that neural neurons can perform invariant recognitions of their input activity patterns producing specific modulations of their synaptic releases, which can drive the development of algorithms closer to biology than other brain - inspired models appear to

Table B.3: A sample arXiv article with model summaries (part 2).

	PubMed
Source:	gossypiboma (retained surgical sponge) describes a mass of cotton or sponge that is left behind in the body cavity during a surgical operation . these foreign bodies can often mimic tumors or abscesses when detected clinically or with radiation . in fact , surgical sponges without opaque markers are the leading cause for the difficulty of a correct diagnosis . here , we present a case in which a foreign body , gauze without radiopaque markers , was left behind during a procedure that involved the removal of an intrapelvic tumor . after a computed tomography (ct) scan had revealed information leading to an inaccurate diagnosis , an accurate diagnosis of gossypiboma was successfully made using endoscopic ultrasound - fine - needle aspiration (eus - fna) . a 60-year - old female patient was referred with an unexpected detection of an intrapelvic tumor during a routine ct scan . the patient 's medical history included a hysterectomy and a left side ovariectomy at the age of 27 . the ct scan revealed a 45 mm tumor in the patient 's pelvis [figure 1a and b] , which was diagnosed as a gastrointestinal stromal tumor (gist) , a chronic expanding hematoma or leiomyoma . a forward - view eus (tgf - uc260j , olympus optical co. , tokyo , japan) identified a low echoic lesion when scanned from the sigmoid colon . fna was performed with a 22-gauge needle (ezshot2 , olympus optical co. , tokyo , japan) by using the forward - view eus [figure 2] . examination of the fna specimen revealed a foreign body (string of gauze) displaying macrophage cells and necrotic tissue [figure 3 ; hematoxylin and eosin , original magnification 40] . during the surgery , histological examination of the tissue adjacent to the foreign body revealed granuloma formations with fragmented silken threads [figure 4 ; hematoxylin and eosin , original magnification , 100 and 200] . a : sagittal section (black arrow) ; b : horizontal section (black arrow) . endoscopic ultrasound (eus)-fine - ...
Target:	$< S >$ gossypiboma is a term used to describe a foreign mass within the body that is composed of a cotton matrix. $< /S >$ $< S >$ gossypiboma can cause infection or abscess formation from an early stage, or it can remain clinically silent for many years. $< /S >$ $< S >$ gossypiboma can be diagnosed using several methods including abdominal x - ray films, us of the abdomen, contrast - enhanced computed tomography of the abdomen, and during surgery. $< /S >$ $< S >$ however, the variable appearance of gossypiboma can lead to diagnostic misinterpretations. here, we report a successful diagnosis of gossypiboma using a novel method, that is, endoscopic ultrasound - fine - needle aspiration. $< /S >$
LED-4096:	$< S >$ gossypiboma (retained surgical sponge) describes a mass of cotton or sponge that is left behind in the body cavity during a surgical operation. here, we present a case in which a foreign body, gauze without radiopaque markers, was left behind during a procedure that involved the removal of an intrapelvic tumor. after a computed tomography (ct) scan had revealed information leading to an inaccurate diagnosis, an accurate diagnosis of gossiopiboma was successfully made using endoscopic ultrasound - fine - needle aspiration (eus - fna). $< /S >$
BART:	$< S >$ a gossypiboma, also called textiloma, literally means retained surgical sponge and is derived from the latin word ” gosspium ”, which means ” cotton, ” and ” boma ” in kiswahili, ” ” place of concealment ”. ” it is difficult to diagnose ” textilomas ” in the neck ” because of its rarity, various symptoms, and non - specific radiologic findings. ” ” ” $< /S >$ ” furthermore, it can be misinterpreted as malignancy ” and finally diagnosed from a histopathological examination of the surgical specimen after unwarranted radical surgery. ” to our knowledge ”, ” text

Table B.4: A sample PubMed article with model summaries. The source has been truncated from 717 to 350 tokens for readability.

C Correlation Metric Definitions

The correlation between two variables, x and y, specifies the strength and direction of these variable's bi-variate relationship, given in the range $[-1, +1]$. Following the WMT18 guidelines (Ma et al. [2018]) we report three measures of correlation: Pearson's r, Spearman's ρ and Kendall's τ. Here we define these metrics.

Pearson's r Measures the linear relationship between two variables (Magiya [2019]). This assumes that the data is normally distributed, has no outliers, the two variables' relationships are linear, the data is complete and homoskedastic. The formula is shown in Equation C.3.

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{C.1}$$
$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad \bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$$

Kendall's τ A non-parametric test comparing the degree of similarity to which two variables rank data. This is useful if the data fails any of the assumptions for using Pearson's r and requires data to be ordinal. This is computed as follows:

$$\tau = \frac{c-d}{c+d} \tag{C.2}$$

Here c is the number of concordant pairs and d is the number of discordant pairs. A concordant pair means that $(y_2 - y_1)$ has the same sign as $(x_2 - x_1)$, otherwise it is discordant.

Spearman's ρ Similar to Kendall's τ, Spearman's ρ is a measure of the association between two sets of ranked data (GmbH [2011]). It is computed by taking the ranks of the variables ($S_i = rank(y)$ and $R_i = rank(x)$) and applying the Pearson formula to the ranked data. The formula is as follows:

$$r = \frac{\sum_{i=1}^{n}(R_i - \bar{R})(S_i - \bar{S})}{\sqrt{\sum_{i=1}^{n}(R_i - \bar{R})^2}\sqrt{\sum_{i=1}^{n}(S_i - \bar{S})^2}} \tag{C.3}$$
$$\bar{R} = \frac{1}{n}\sum_{i=1}^{n} R_i = \bar{S} = \frac{1}{n}\sum_{i=1}^{n} S_i = \frac{n(n-1)}{2}$$

D Model Configurations

Parameter	PEGASUS-LG	ProphetNet-LG	BART-LG
Attention type	n^2	n^2	n^2
Max. encoder input length			
- CNN/DM	1024	512	2048
- PubMed	1024	N.A.	N.A.
- arXiv	1024	N.A.	N.A.
Max. decoder input length			
- CNN/DM	128	110	142
- PubMed	256	N.A.	N.A.
- arXiv	256	N.A.	N.A.
Beam size	8	5	5
Length penalty	0.8	1.2	2.0
Num. of heads	16	12	16
Num. layers	16	12	12
Hidden layer dim.	1024	1024	1024
Batch size	256	512	256
Activation function	ReLU	ReLU	GeLU
Optimizer	Adafactor	Adam	Adam
Learning rate	5e-5	1e-4	3e-5
Label smoothing	0.1	0.1	0.1
Dropout	0.1	0.1	0.1

Table D.1: Side-by-side comparison of the configurations of `BART`, `PEGASUS` and `ProphetNet`. Sources: Lewis et al. [2019], Zhang et al. [2019a], Yan et al. [2020].

Parameter	**BART-LG**	**LED**	**RED**
Attention type	n^2	Window-512	Alternating `LSH` & `local`
Max. encoder input length			
- CNN/DM	1024	1024*	N.A.**
- PubMed	1024	4096	4096
- arXiv	1024	4096	N.A.**
Max. decoder input length			
- CNN/DM	142	142	142
- PubMed	200	200	200
- arXiv	200	200	200
Beam size	4	4	4
Length penalty	2.0	2.0	2.0
Num. of heads	16	16	8
Num. layers	12	12	12
Hidden layer dim.	1024	1024	1024
Batch size	1	1	1
Activation function	GeLU	GeLU	GeLU
Optimizer	AdamW	AdamW	AdamW
Learning rate	3e-5	1e-5	1e-5
Label smoothing	0.1	0.0	0.0
Dropout	0.1	0.1	0.1
Chunk size (local)***	N.A.	N.A.	64
Chunk size (LSH)***	N.A.	N.A.	64

Table D.2: Model configurations of our best-performing variants of each model after performing hyperparameter search. * We did not use a longer version of the `LED` for the CNN/DailyMail dataset as the articles are short. ** We only experimented with the `RED` on the PubMed dataset. *** `RED`-specific arguments.

E Supporting Results

Attn Window	BA	BE	M-1	M-2	BLEURT	R-1	R-2	R-L
512	0.599	0.264	0.167	0.224	-0.055	0.428	0.176	0.383
	0.06	0.11	0.11	0.10	0.17	0.00	0.00	0.00
256	0.594	0.256	0.156	0.215	-0.078	0.416	0.170	0.367
	0.06	0.11	0.11	0.10	0.18	0.00	0.00	0.00
128	0.597	0.260	0.163	0.220	-0.066	0.423	0.172	0.375
	0.06	0.11	0.11	0.10	0.17	0.00	0.00	0.00
64	0.593	0.254	0.155	0.214	-0.080	0.416	0.168	0.368
	0.06	0.11	0.11	0.10	0.17	0.00	0.00	0.00

Table E.1: LED-1024 performance on a summarization task by attention window size. Scores reported are the mean metric scores with the standard deviation underneath. These scores are obtained using the PubMed test set after fine-tuning the models for one epoch on the PubMed dataset. These results are displayed graphically in Figure 6.3.

Model Length	Attention window			
	512	256	128	64
1024	9.81	9.11	8.75	8.58
1536	11.23	10.18	9.64	9.39
2048	12.63	11.23	10.52	10.16
2560	14.04	12.31	11.43	10.99
3072	15.48	13.39	12.33	11.78
3548	16.89	14.44	13.20	12.60
4096	18.30	15.50	14.09	13.39

Table E.2: Maximum memory consumption (Gb) of fine-tuning the LED by attention window size and model length. These results are shown graphically in Figure 6.4.

Model Length	**BA**	**BE**	**M-1**	**M-2**	**BLEURT**	**R-1**	**R-2**	**R-L**
PubMed								
1024	0.596	0.262	0.159	0.217	-0.068	0.423	0.174	0.375
	0.06	0.11	0.11	0.10	0.17	0.10	0.11	0.10
1536	0.595	0.259	0.165	0.224	-0.071	0.422	0.176	0.375
	0.06	0.11	0.11	0.10	0.18	0.10	0.12	0.11
2048	0.592	0.254	0.157	0.217	-0.080	0.416	0.169	0.367
	0.06	0.11	0.11	0.10	0.18	0.10	0.12	0.11
2560	0.591	0.256	0.159	0.217	-0.070	0.421	0.174	0.372
	0.06	0.11	0.11	0.10	0.18	0.10	0.12	0.11
3072	0.598	0.268	0.171	0.229	-0.054	0.428	0.178	0.380
	0.06	0.10	0.10	0.10	0.17	0.10	0.11	0.10
3584	0.599	0.267	0.171	0.229	-0.061	0.426	0.177	0.377
	0.06	0.11	0.11	0.10	0.17	0.10	0.12	0.10
4096	0.596	0.267	0.170	0.227	-0.060	0.429	0.179	0.380
	0.06	0.11	0.11	0.10	0.17	0.10	0.12	0.10
arXiv								
1024	0.597	0.265	0.156	0.213	-0.097	0.439	0.165	0.388
	0.04	0.08	0.09	0.08	0.16	0.08	0.08	0.08
1536	0.597	0.269	0.164	0.221	-0.094	0.439	0.165	0.389
	0.04	0.08	0.08	0.08	0.16	0.08	0.08	0.08
2048	0.601	0.276	0.170	0.226	-0.080	0.448	0.173	0.397
	0.04	0.08	0.09	0.08	0.16	0.08	0.08	0.08
2560	0.594	0.264	0.158	0.216	-0.098	0.436	0.162	0.384
	0.04	0.07	0.08	0.07	0.16	0.08	0.08	0.08
3072	0.601	0.274	0.171	0.227	-0.084	0.448	0.174	0.396
	0.04	0.08	0.09	0.08	0.16	0.08	0.08	0.08
3584	0.602	0.276	0.175	0.231	-0.074	0.451	0.174	0.400
	0.04	0.08	0.08	0.08	0.16	0.08	0.08	0.08
4096	0.602	0.277	0.174	0.230	-0.073	0.451	0.174	0.399
	0.04	0.08	0.09	0.08	0.16	0.08	0.08	0.08

Table E.3: `LED` performance on a summarization task by model length with a 512 attention window (using Beginning Starts, PubMed and arXiv test sets). Scores reported are the mean metric score with the standard deviation underneath. These results are displayed graphically in Figure 6.2 (left-sided panels).

Model Length	**BA**	**BE**	**M-1**	**M-2**	**BLEURT**	**R-1**	**R-2**	**R-L**
PubMed								
1024	0.577	0.231	0.121	0.184	-0.105	0.385	0.128	0.336
	0.05	0.09	0.09	0.08	0.16	0.09	0.08	0.08
1536	0.584	0.245	0.136	0.197	-0.087	0.402	0.143	0.355
	0.05	0.09	0.09	0.08	0.16	0.09	0.09	0.09
2048	0.586	0.247	0.144	0.204	-0.085	0.407	0.153	0.359
	0.06	0.10	0.10	0.09	0.17	0.10	0.11	0.10
2560	0.587	0.246	0.147	0.207	-0.083	0.408	0.156	0.361
	0.06	0.11	0.10	0.10	0.17	0.10	0.11	0.10
3072	0.592	0.259	0.159	0.218	0.421	0.167	0.375	-0.061
	0.05	0.10	0.10	0.09	0.09	0.11	0.09	0.16
3584	0.591	0.254	0.156	0.215	-0.076	0.417	0.167	0.369
	0.06	0.11	0.11	0.10	0.17	0.10	0.12	0.10
4096	0.592	0.259	0.159	0.218	-0.061	0.421	0.167	0.375
	0.05	0.10	0.10	0.09	0.16	0.09	0.11	0.09
arXiv								
1024	0.574	0.226	0.093	0.155	-0.168	0.395	0.126	0.351
	0.04	0.07	0.08	0.08	0.17	0.08	0.06	0.07
1536	0.575	0.228	0.104	0.167	-0.173	0.386	0.127	0.341
	0.05	0.08	0.09	0.08	0.18	0.09	0.07	0.08
2048	0.593	0.260	0.150	0.208	-0.103	0.431	0.155	0.380
	0.04	0.08	0.09	0.08	0.16	0.08	0.08	0.08
2560	0.580	0.248	0.124	0.185	-0.128	0.413	0.140	0.366
	0.04	0.07	0.08	0.08	0.16	0.08	0.07	0.08
3072	0.586	0.248	0.130	0.189	-0.126	0.422	0.147	0.373
	0.04	0.07	0.08	0.08	0.16	0.08	0.07	0.08
3584	0.587	0.254	0.138	0.197	-0.120	0.426	0.151	0.377
	0.04	0.07	0.08	0.08	0.16	0.08	0.07	0.08
4096	0.594	0.265	0.152	0.210	-0.100	0.436	0.160	0.384
	0.04	0.08	0.09	0.08	0.16	0.08	0.08	0.08

Table E.4: LED performance on a summarization task by model length. This is for the arXiv and PubMed test sets using Random Starts (see section 5.2.5 for details). Scores reported are the mean metric score with the standard deviation underneath. These results are displayed graphically in Figure 6.2 (right-sided panels).

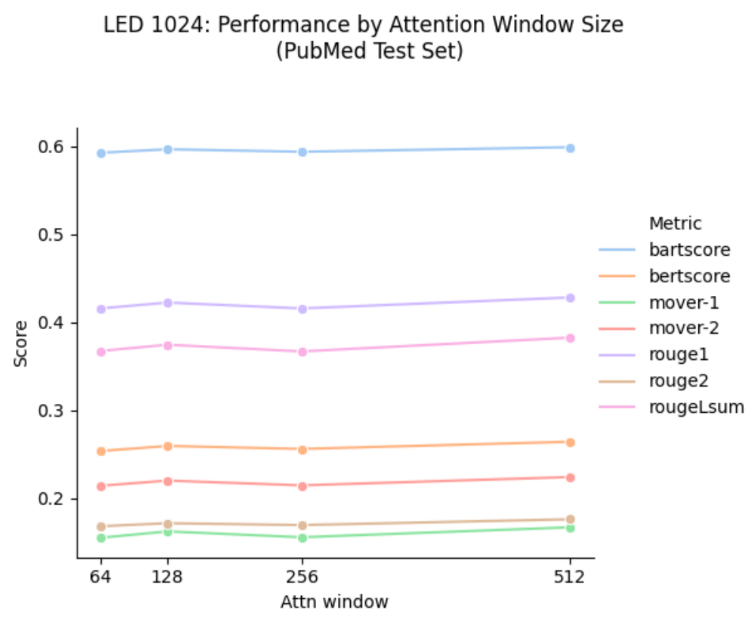


Figure E.1: Performance of the `LED 1024` by attention window length. This plot shows the mean metric scores using the PubMed test set. A re-based version of this plot is shown in Figure 6.3 to improve visibility over the trends.

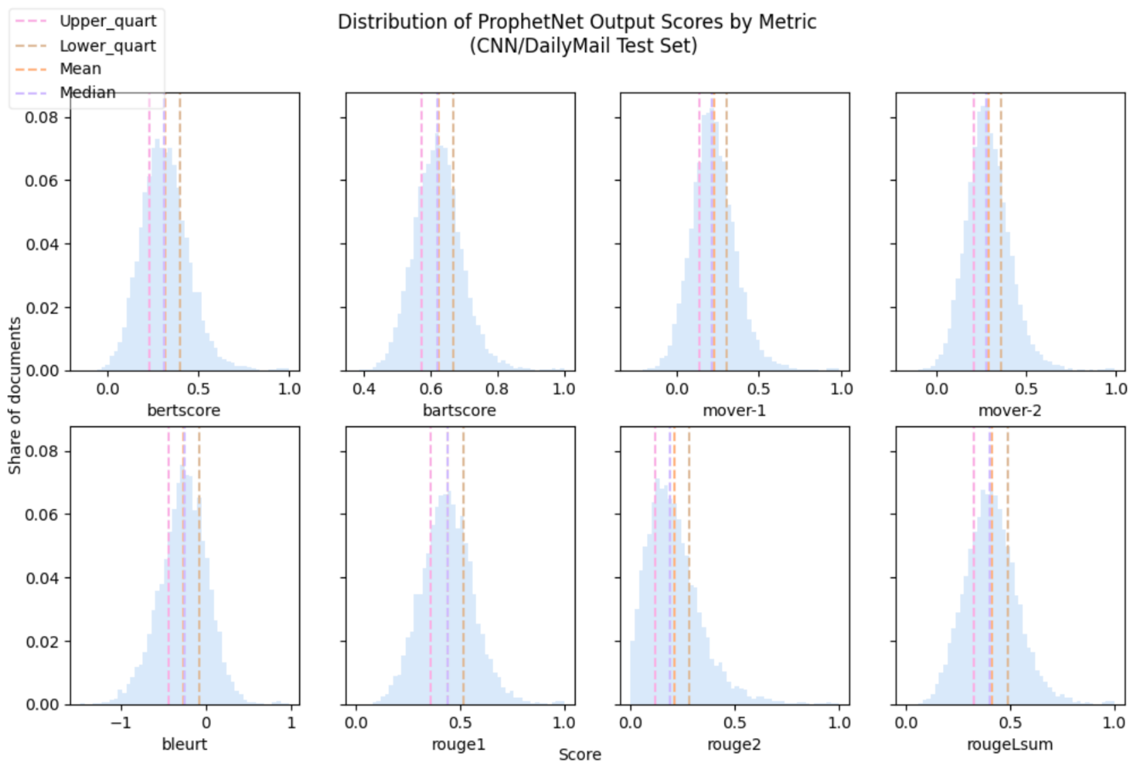


Figure E.2: Plots of the distribution of scores for ProphetNet's summaries using the CNN/DailyMail test set by evaluation metric. The mean, median, upper and lower quartiles are annotated onto the figures.

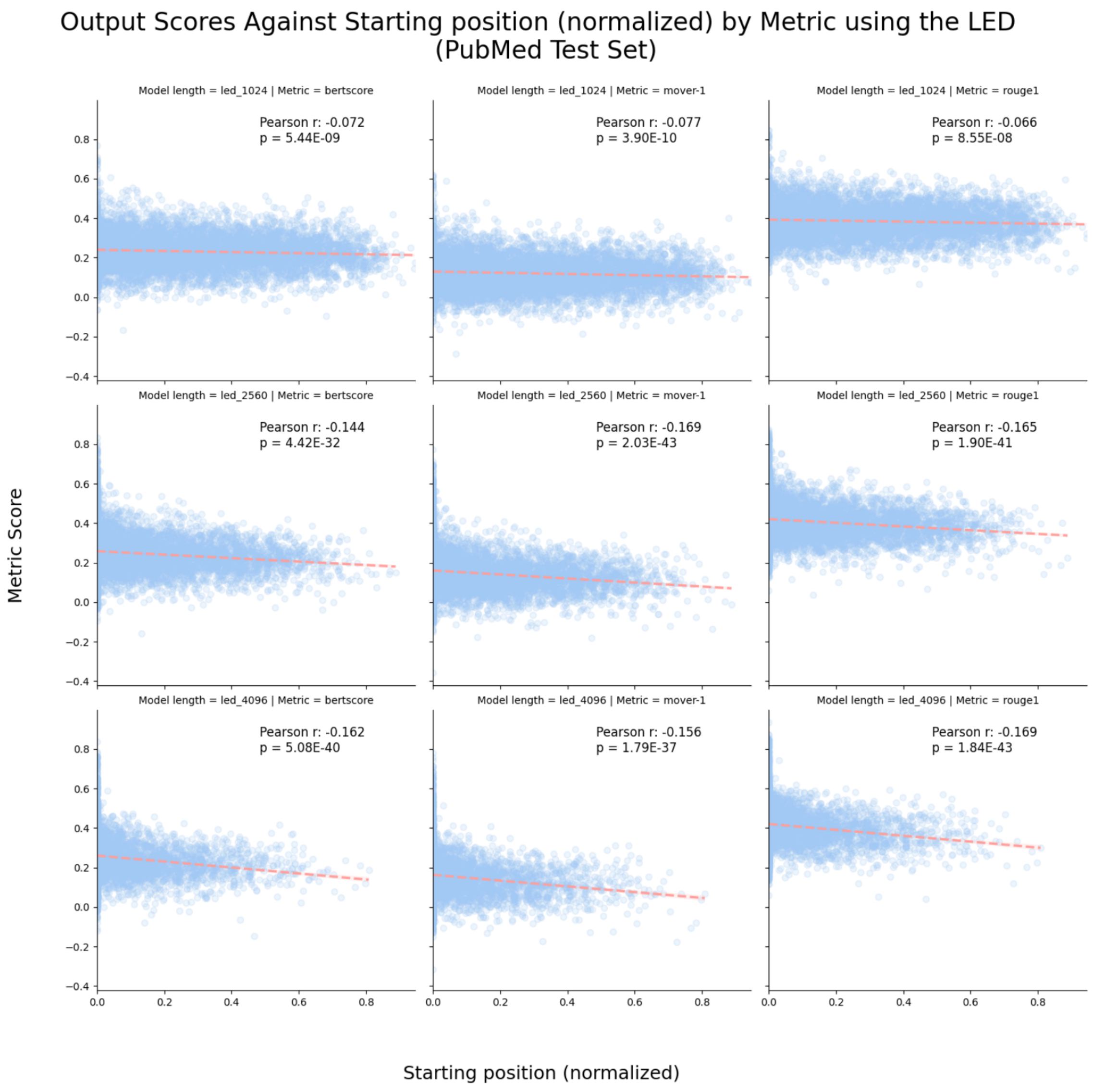


Figure E.3: Scatter-plots of the normalized starting position against the evaluation metric score on the PubMed test set. The columns display `bertscore`, `mover-1` and `ROUGE-1`; the rows display the 1024, 2560 and 4096 length `LED` models. The Pearson r correlation with associated p-stat is annotated for each subplot.

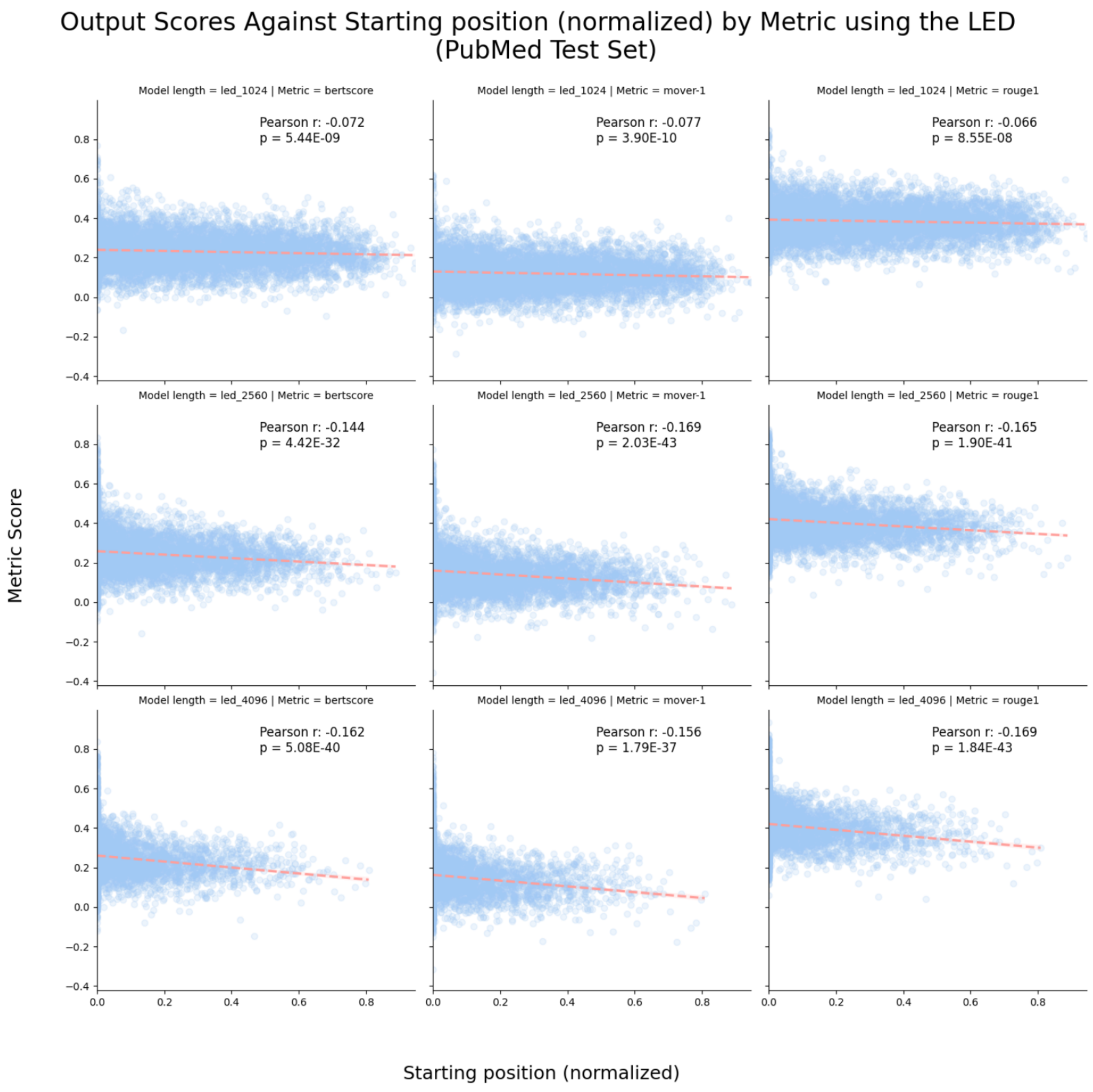


Figure E.4: Scatter-plots of the normalized starting position against the evaluation metric score on the arXiv test set. The columns display `bertscore`, `mover-1` and `ROUGE-1`; the rows display the 1024, 2560 and 4096 length `LED` models. The Pearson r correlation with associated p-stat is annotated for each subplot.

F Ethics Checklist

	Yes	No
Section 1: HUMAN EMBRYOS/FOETUSES		
Does your project involve Human Embryonic Stem Cells?		x
Does your project involve the use of human embryos?		x
Does your project involve the use of human foetal tissues / cells?		x
Section 2: HUMANS		
Does your project involve human participants?		x
Section 3: HUMAN CELLS / TISSUES		
Does your project involve human cells or tissues? (Other than from ”Human Embryos/Foetuses” i.e. Section 1)?		x
Section 4: PROTECTION OF PERSONAL DATA		
Does your project involve personal data collection and/or processing?	x	
Does it involve the collection and/or processing of sensitive personal data (e.g. health, sexual lifestyle, ethnicity, political opinion, religious or philosophical conviction)?	x	
Does it involve processing of genetic information?		x
Does it involve tracking or observation of participants? It should be noted that this issue is not limited to surveillance or localization data. It also applies to Wan data such as IP address, MACs, cookies etc.		x
Does your project involve further processing of previously collected personal data (secondary use)? For example Does your project involve merging existing data sets?		x
Section 5: ANIMALS		
Does your project involve animals?		x
Section 6: DEVELOPING COUNTRIES		
Does your project involve developing countries?		x
If your project involves low and/or lower-middle income countries, are any benefit-sharing actions planned?		x
Could the situation in the country put the individuals taking part in the project at risk?		x

Table F.1: Ethics Checklist, part 1. Adapted from the Imperial College Department of Computing website `https://www.doc.ic.ac.uk/lab/msc-projects/ethics-checklist.xlsx`.

	Yes	No
Section 7: ENVIRONMENTAL PROTECTION AND SAFETY		
Does your project involve the use of elements that may cause harm to the environment, animals or plants?		x
Does your project deal with endangered fauna and/or flora /protected areas?		x
Does your project involve the use of elements that may cause harm to humans, including project staff?		x
Does your project involve other harmful materials or equipment, e.g. high-powered laser systems?		x
Section 8: DUAL USE		
Does your project have the potential for military applications?		x
Does your project have an exclusive civilian application focus?	x	
Will your project use or produce goods or information that will require export licenses in accordance with legislation on dual use items?		x
Does your project affect current standards in military ethics ? e.g., global ban on weapons of mass destruction, issues of proportionality, discrimination of combatants and accountability in drone and autonomous robotics developments, incendiary or laser weapons?		x
Section 9: MISUSE		
Does your project have the potential for malevolent/criminal/terrorist abuse?		x
Does your project involve information on/or the use of biological-, chemical-, nuclear/radiological-security sensitive materials and explosives, and means of their delivery?		x
Does your project involve the development of technologies or the creation of information that could have severe negative impacts on human rights standards (e.g. privacy, stigmatization, discrimination), if misapplied?		x
Does your project have the potential for terrorist or criminal abuse e.g. infrastructural vulnerability studies, cybersecurity related project?		x
SECTION 10: LEGAL ISSUES		
Will your project use or produce software for which there are copyright licensing implications?		x
Will your project use or produce goods or information for which there are data protection, or other legal implications?		x
SECTION 11: OTHER ETHICS ISSUES		
Are there any other ethics issues that should be taken into consideration?		x

Table F.2: Ethics Checklist, part 2. Adapted from the Imperial College Department of Computing website `https://www.doc.ic.ac.uk/lab/msc-projects/ethics-checklist.xlsx`.

References

Alexandr Andoni, Piotr Indyk, Thijs Laarhoven, Ilya P. Razenshteyn, and Ludwig Schmidt. Practical and optimal LSH for angular distance. *CoRR*, abs/1509.02897, 2015. URL http://arxiv.org/abs/1509.02897. pages 21

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014. pages 9

Satanjeev Banerjee and Alon Lavie. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W05-0909. pages 15

Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer, 2020. pages 20, 21, 31, 32, 33, 38, 52, 53

Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *Trans. Neur. Netw.*, 5(2):157–166, March 1994. ISSN 1045-9227. doi: 10.1109/72.279181. URL https://doi.org/10.1109/72.279181. pages 7

Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. O'Reilly, Beijing, 2009. ISBN 978-0-596-51649-9. doi: http://my.safaribooksonline.com/9780596516499. URL http://www.nltk.org/book. pages 35

Florian Böhm, Yang Gao, Christian M Meyer, Ori Shapira, Ido Dagan, and Iryna Gurevych. Better rewards yield better summaries: Learning to summarise without references. *arXiv preprint arXiv:1909.01214*, 2019. pages 13, 14, 24, 25, 36, 57, 65

Arun Chaganty, Stephen Mussmann, and Percy Liang. The price of debiasing automatic metrics in natural language evalaution. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 643–653, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1060. URL https://www.aclweb.org/anthology/P18-1060. pages 28, 57

Kushal Chauhan. Unsupervised text summarization using sentence embeddings. 2018. URL https://medium.com/jatana/unsupervised-text-summarization-using-sentence-embeddings-adb15ce83db1. pages 3

Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost. *CoRR*, abs/1604.06174, 2016. URL http://arxiv.org/abs/1604.06174. pages 33

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *CoRR*, abs/1904.10509, 2019. URL http://arxiv.org/abs/1904.10509. pages 64

Eric Chu and Peter J. Liu. Unsupervised neural multi-document abstractive summarization. *CoRR*, abs/1810.05739, 2018. URL http://arxiv.org/abs/1810.05739. pages 26

Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. A discourse-aware attention model for abstractive summarization of long documents. *CoRR*, abs/1804.05685, 2018. URL `http://arxiv.org/abs/1804.05685`. pages 26, 27, 28, 29

J. Copeland. *Artificial Intelligence: A Philosophical Introduction*. Wiley, 2015. ISBN 9781119189848. URL `https://books.google.co.uk/books?id=T05ICgAAQBAJ`. pages 4

Thomas Cormen, Charles Leiserson, Ronald Rivest, and Clifford Stein. *Introduction To Algorithms*, volume 42. 01 2001. doi: 10.2307/2583667. pages 13

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018. pages 5, 14, 17, 18, 30, 31, 35, 52

Alireza Dirafzoon. Illustrating the reformer. 2020. URL `https://towardsdatascience.com/illustrating-the-reformer-393575ac6ba0`. pages 22, 23

H. P. Edmundson. New methods in automatic extracting. *J. ACM*, 16(2):264–285, April 1969. ISSN 0004-5411. doi: 10.1145/321510.321519. URL `https://doi.org/10.1145/321510.321519`. pages 4

Alexander R. Fabbri, Irene Li, Tianwei She, Suyi Li, and Dragomir R. Radev. Multi-news: a large-scale multi-document summarization dataset and abstractive hierarchical model. *CoRR*, abs/1906.01749, 2019. URL `http://arxiv.org/abs/1906.01749`. pages 29, 65

Philip Gage. A new algorithm for data compression. *C Users J.*, 12(2):23–38, February 1994. ISSN 0898-9788. pages 35

Lukas Gebhard and Felix Hamborg. The polusa dataset: 0.9m political news articles balanced by time and outlet popularity. In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries in 2020*, JCDL '20, page 467–468, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450375856. doi: 10.1145/3383583.3398567. URL `https://doi.org/10.1145/3383583.3398567`. pages 35

Verlag GmbH. Encyclopedia of Mathematics. Website, 2011. URL: `http://encyclopediaofmath.org/index.php?title=Spearman_rho_metric&oldid=50006`. Accessed on 2020-09-16. pages 75

Aaron Gokaslan and Vanya Cohen. Openwebtext corpus. pages 35

Aidan N. Gomez, Mengye Ren, Raquel Urtasun, and Roger B. Grosse. The reversible residual network: Backpropagation without storing activations. *CoRR*, abs/1707.04585, 2017. URL `http://arxiv.org/abs/1707.04585`. pages 22

Alex Graves. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850, 2013. URL `http://arxiv.org/abs/1308.0850`. pages 6

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL `http://arxiv.org/abs/1512.03385`. pages 5

Dan Hendrycks and Kevin Gimpel. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *CoRR*, abs/1606.08415, 2016. URL `http://arxiv.org/abs/1606.08415`. pages 18

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. *CoRR*, abs/1506.03340, 2015. URL `http://arxiv.org/abs/1506.03340`. pages 27

Hochreiter and Jürgen Schmidhuber, Sepp. Long short-term memory. *Neural computation*, 9 (8):1735–1780, 1997. pages 7

S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In S. C. Kremer and J. F. Kolen, editors, *A Field Guide to Dynamical Recurrent Neural Networks*. IEEE Press, 2001. pages 7

Aya Abdelsalam Ismail, Timothy Wood, and Héctor Corrada Bravo. Improving long-horizon forecasts with expectation-biased LSTM networks. *CoRR*, abs/1804.06776, 2018. URL `http://arxiv.org/abs/1804.06776`. pages 7

Heng Ji and Ralph Grishman. Knowledge base population: Successful approaches and challenges. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, page 1148–1158, USA, 2011. Association for Computational Linguistics. ISBN 9781932432879. pages 4, 16

Yaser Keneshloo, Tian Shi, Naren Ramakrishnan, and Chandan K. Reddy. Deep reinforcement learning for sequence to sequence models. *CoRR*, abs/1805.09461, 2018. URL `http://arxiv.org/abs/1805.09461`. pages 6

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer, 2020. pages 21, 31, 38

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. URL `http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf`. pages 5

Matt J. Kusner, Yu Sun, Nicholas I. Kolkin, and Kilian Q. Weinberger. From word embeddings to document distances. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 957–966. JMLR.org, 2015. pages 16

Alex M Lamb, Anirudh Goyal ALIAS PARTH GOYAL, Ying Zhang, Saizheng Zhang, Aaron C Courville, and Yoshua Bengio. Professor forcing: A new algorithm for training recurrent networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4601–4609. Curran Associates, Inc., 2016. URL `http://papers.nips.cc/paper/6099-professor-forcing-a-new-algorithm-for-training-recurrent-networks.pdf`. pages 6, 33

W.G. Lehnert and M.H. Ringle. *Strategies for Natural Language Processing*. Taylor & Francis, 1982. ISBN 9781317769255. URL `https://books.google.co.uk/books?id=feNHAwAAQBAJ`. pages 4

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension, 2019. pages 18, 23, 30, 32, 37, 38, 42, 47, 49, 57, 76

Siyao Li, Deren Lei, Pengda Qin, and William Yang Wang. Deep reinforcement learning with distributional semantic rewards for abstractive summarization. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6038–6044, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1623. URL `https://www.aclweb.org/anthology/D19-1623`. pages 15, 24, 25

Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/W04-1013`. pages 12, 30, 34

Peter J. Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. Generating wikipedia by summarizing long sequences, 2018. pages 20

Yang Liu. Fine-tune bert for extractive summarization, 2019. pages 17

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019. pages 21, 30, 42

Annie Louis and Ani Nenkova. Automatically assessing machine summary content without a gold standard. *Computational Linguistics*, 39(2):267–300, 2013. doi: 10.1162/COLI_a_00123. URL `https://doi.org/10.1162/COLI_a_00123`. pages 13

H. P. Luhn. The automatic creation of literature abstracts. *IBM Journal of Research and Development*, 2(2):159–165, 1958. URL `http://www.research.ibm.com/journal/rd/022/luhn.pdf`. pages 4

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. *CoRR*, abs/1508.04025, 2015. URL `http://arxiv.org/abs/1508.04025`. pages 9, 10

Qingsong Ma, Ondřej Bojar, and Yvette Graham. Results of the WMT18 metrics shared task: Both characters and embeddings achieve good performance. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 671–688, Belgium, Brussels, October 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-6450. URL `https://www.aclweb.org/anthology/W18-6450`. pages 15, 16, 75

Joseph Magiya. Unsupervised text summarization using sentence embeddings. 2019. URL `https://towardsdatascience.com/pearson-coefficient-of-correlation-explained-369991d93404`. pages 75

Ramesh Nallapati, Bing Xiang, and Bowen Zhou. Sequence-to-sequence rnns for text summarization. *CoRR*, abs/1602.06023, 2016. URL `http://arxiv.org/abs/1602.06023`. pages 23

Christopher Olah. Understanding lstm networks, 2015. URL `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`. pages 6

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. fairseq: A fast, extensible toolkit for sequence modeling. *CoRR*, abs/1904.01038, 2019. URL http://arxiv.org/abs/1904.01038. pages 38, 49

Paul Over, Hoa Dang, and Donna Harman. Duc in context. *Inf. Process. Manage.*, 43(6): 1506–1520, November 2007. ISSN 0306-4573. doi: 10.1016/j.ipm.2007.01.019. URL https://doi.org/10.1016/j.ipm.2007.01.019. pages 4

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL https://www.aclweb.org/anthology/P02-1040. pages 13

Ramakanth Pasunuru and Mohit Bansal. Multi-reward reinforced summarization with saliency and entailment. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 646–653, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-2102. URL https://www.aclweb.org/anthology/N18-2102. pages 24, 25

Romain Paulus, Caiming Xiong, and Richard Socher. A deep reinforced model for abstractive summarization. *CoRR*, abs/1705.04304, 2017. URL http://arxiv.org/abs/1705.04304. pages 28

Martin F Porter et al. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980. pages 35

Jiezhong Qiu, Hao Ma, Omer Levy, Scott Wen tau Yih, Sinong Wang, and Jie Tang. Blockwise self-attention for long document understanding, 2019. pages 64

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019. pages 17, 18, 35

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2019. pages 17

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text, 2016. pages 30

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning Representations by Back-propagating Errors. *Nature*, 323(6088):533–536, 1986. doi: 10.1038/323533a0. URL http://www.nature.com/articles/323533a0. pages 5

Alexander Rush. The annotated transformer. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 52–60, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-2509. URL https://www.aclweb.org/anthology/W18-2509. pages 11

Horacio Saggion and Thierry Poibeau. *Automatic Text Summarization: Past, Present and Future*, pages 3–21. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. ISBN 978-3-642-28569-1. doi: 10.1007/978-3-642-28569-1_1. URL https://doi.org/10.1007/978-3-642-28569-1_1. pages 4

Chip Scanlan. The inverted pyramid structure. 2008. URL `https://owl.purdue.edu/owl/subject_specific_writing/journalism_and_journalistic_writing/the_inverted_pyramid.html`. pages 25

M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997. pages 8

Abigail See, Peter J. Liu, and Christopher D. Manning. Get to the point: Summarization with pointer-generator networks, 2017. pages 2, 10, 15, 23, 24, 25, 26, 27, 28, 37, 47

Thibault Sellam, Dipanjan Das, and Ankur P. Parikh. Bleurt: Learning robust metrics for text generation, 2020. pages 15, 30, 37, 56

Eva Sharma, Chen Li, and Lu Wang. BIGPATENT: A large-scale dataset for abstractive and coherent summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2204–2213, Florence, Italy, July 2019a. Association for Computational Linguistics. doi: 10.18653/v1/P19-1212. URL `https://www.aclweb.org/anthology/P19-1212`. pages 29, 65

Lakshay Sharma, Laura Graesser, Nikita Nangia, and Utku Evci. Natural language understanding with the quora question pairs dataset. *CoRR*, abs/1907.01041, 2019b. URL `http://arxiv.org/abs/1907.01041`. pages 28, 36

Lucia Specia, Julia Ive, and Ozan Caglayan. Lecture notes in natural language processing (lecture 8). February 2020. pages 5

Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in NLP. *CoRR*, abs/1906.02243, 2019. URL `http://arxiv.org/abs/1906.02243`. pages 2

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215, 2014. URL `http://arxiv.org/abs/1409.3215`. pages 5, 8, 9

Wilson L. Taylor. ”cloze procedure”: a new tool for measuring readability. *Journalism Mass Communication Quarterly*, 30:415–433, 1953. pages 18

Trieu H. Trinh and Quoc V. Le. A simple method for commonsense reasoning. *CoRR*, abs/1806.02847, 2018. URL `http://arxiv.org/abs/1806.02847`. pages 35

K. Tutschku. Recurrent multilayer perceptrons for identification and control: The road to applications, 1995. pages 8

Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *CoRR*, abs/1609.03499, 2016. URL `http://arxiv.org/abs/1609.03499`. pages 20

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL `http://arxiv.org/abs/1706.03762`. pages 4, 11, 12, 21

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks, 2015. pages 23

Patrick von Platen. The reformer - pushing the limits of language modeling. 2020. URL `https://huggingface.co/blog/reformer`. pages 22

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding, 2018. pages 30

Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity, 2020. pages 64

Paul J. Werbos. Generalization of backpropagation with application to a recurrent gas market model, January 1988. URL `https://doi.org/10.1016/0893-6080(88)90007-x`. pages 7

Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics, 2018. URL `http://aclweb.org/anthology/N18-1101`. pages 16, 31, 43

R. J. Williams and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1(2):270–280, 1989. pages 6

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771, 2019. pages 14, 31, 33, 37, 38, 41, 43, 49

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016. URL `http://arxiv.org/abs/1609.08144`. pages 35

Yu Yan, Weizhen Qi, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. Prophetnet: Predicting future n-gram for sequence-to-sequence pre-training, 2020. pages 19, 37, 76

Wonjin Yoon, Yoon Sun Yeo, Minbyul Jeong, Bong-Jun Yi, and Jaewoo Kang. Learning by semantic similarity makes abstractive summarization better. *ArXiv*, abs/2002.07767, 2020. pages 14

Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big bird: Transformers for longer sequences, 2020. pages 1, 63

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization, 2019a. pages 14, 17, 19, 33, 34, 36, 37, 38, 42, 47, 57, 64, 76

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with BERT. *CoRR*, abs/1904.09675, 2019b. URL `http://arxiv.org/abs/1904.09675`. pages 14, 25, 30

Wei Zhao, Maxime Peyrard, Fei Liu, Yang Gao, Christian M. Meyer, and Steffen Eger. Moverscore: Text generation evaluating with contextualized embeddings and earth mover distance, 2019. pages 14, 16, 30, 42, 43

Chenguang Zhu, Ziyi Yang, Robert Gmyr, Michael Zeng, and Xuedong Huang. Make lead bias in your favor: A simple and effective method for news summarization, 2019. pages 26

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015. pages 35