



Dialogue Summarization with Retrieval-Augmented Transformers

Diogo Miguel Rodrigues Cruz Fouto Silva

Thesis to obtain the Master of Science Degree in

Computer Science and Engineering

Supervisors: Prof. Maria Luísa Torres Ribeiro Marques da Silva Coheur
Prof. Bruno Emanuel da Graça Martins

Examination Committee

Chairperson: Prof. Luís Manuel Antunes Veiga
Supervisor: Prof. Maria Luísa Torres Ribeiro Marques da Silva Coheur
Member of the Committee: Prof. Ricardo Daniel Santos Faro Marques Ribeiro

June 2024

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowledgments

Writing this thesis would have been impossible without help and support.

I would first like to thank my supervisors, Professor Luísa Coheur and Professor Bruno Martins. Their guidance and support have been essential throughout this research journey. Their expertise and thoughtful feedback have significantly shaped this dissertation, and I am thankful for their mentorship.

I am also grateful to my family for their unwavering support. Your sacrifices and steadfast support have made this achievement possible.

I also wish to thank my classmates and friends. Your encouragement, insightful discussions, and willingness to lend a hand have been invaluable. The collaboration and shared experiences have enriched my work and made this journey more rewarding.

Lastly, I would like to acknowledge everyone who has supported me along the way. Each contribution, whether academic or emotional, has played a crucial role in the completion of this dissertation. I am grateful for your support and encouragement. Thank you for being an integral part of this journey.

Abstract

Dialogue summarization is the task of condensing a dialogue into a shorter version which aims to preserve the highlights of the conversation. Recent methods based on encoder-decoder neural networks have already achieved good results for single-document summarization (e.g., news or scientific articles). However, major problems arise when these methods are applied to dialogue summarization. Long inputs and topic drifts can mislead even the best models. In this dissertation, I propose a novel, retrieval-augmented approach for dialogue summarization that fetches the most important passages from a dialogue and summarizes only those. Experimental results demonstrate superior performance for this approach compared to previous methods. This work also introduces an augmented version of an existing dataset for dialogue summarization, which includes the most relevant passages for each dialogue. This contribution can further incentivize future work in this area.

Keywords

Dialogue summarization; Retrieval augmentation; Transformers; Training; Inference.

Resumo

A sumarização de diálogos consiste em condensar um diálogo numa versão mais curta que tem como objetivo preservar os destaques da conversa. Métodos recentes baseados em redes neuronais codificador-decodificador já alcançam bons resultados em sumarização de documentos (ex., notícias ou artigos científicos). No entanto, quando se aplicam os mesmos métodos a diálogo, surgem vários problemas. Diálogos longos e desvios de tópicos enganam até mesmo os melhores modelos. Nesta dissertação, proponho uma nova abordagem com aumento por reaquisição que procura as passagens mais importantes num diálogo e resume apenas essas. Os resultados experimentais mostram resultados melhores para este método quando comparado a métodos prévios. Este trabalho também apresenta uma versão aumentada de um *dataset* para sumarização de diálogo já existente que contém, para cada diálogo, as suas passagens mais relevantes. Esta contribuição pode incentivar trabalho futuro nesta área.

Palavras Chave

Sumarização de diálogo; Retrieval augmentation; Transformers; Treino; Inferência.

Contents

1	Introduction	1
1.1	Motivation	3
1.2	Goals	3
1.3	Contributions	3
1.4	Outline	4
2	Fundamental Concepts	5
2.1	Neural Networks	7
2.1.1	Perceptron	7
2.1.2	Multi-Layer Perceptron	7
2.1.3	Supervised Learning	8
2.2	Transformer Language Models	11
2.2.1	The Original Transformer	11
2.2.2	Transfer Learning	14
2.2.3	BERT	16
2.2.4	GPT	17
2.2.5	BART	17
2.3	The Summarization Task	18
2.4	Summary	19
3	Related Work	21
3.1	Document Summarization	23
3.1.1	PEGASUS	24
3.1.2	SimCLS	25
3.2	Retrieval Augmentation for Text Generation	27
3.2.1	Retrieval-Enhanced Transformer	28
3.2.2	Atlas	30
3.3	Dialogue Summarization	32
3.3.1	DialogLM	34

3.3.2	MV-BART	35
3.4	Summary	36
4	Retrieval-augmented Dialogue Summarization	37
4.1	Component Overview	39
4.1.1	Retriever	40
4.1.2	Language Model	41
4.2	Augmenting a Dialogue Summarization Dataset	42
4.3	Retriever Fine-tuning	44
4.4	Retriever Inference	45
4.5	Summary	46
5	Evaluation	47
5.1	Experimental Setup	49
5.1.1	Dataset	49
5.1.2	Evaluation Metrics	49
5.2	Experimental Results	50
5.2.1	Augmented Dataset Analysis	50
5.2.2	New Architecture Results	52
5.3	Summary	53
6	Conclusion and Future Work	55
6.1	Conclusion	57
6.2	Future Work	57
	Bibliography	59

List of Figures

1.1	Relevant turns from a dialogue and respective summary.	4
2.1	The Transformer architecture.	12
2.2	Bidirectional Encoder Representation from Transformers (BERT) input representation. . .	16
2.3	BART architecture. The corrupted document (left) is encoded with the bidirectional encoder, and the probability of a document (right) is calculated with the autoregressive decoder.	18
3.1	SimCLS framework. At the first stage (far left), BART generates candidate summaries. At the second stage (far right), Robustly Optimized BERT Approach (RoBERTa) ranks them against a source document.	26
3.2	Retrieval-Enhanced Transformer (RETRO) architecture.	29
3.3	Cross-attention step.	30
4.1	Component overview. The retriever classifies the turns from a dialogue as relevant or irrelevant, re-ranks them, and feeds them to a language model for summarization.	39
4.2	The MPNet architecture. The light grey lines represent the bidirectional self-attention in the non-masked part. The blue, green, and black lines represent self-attention in the masked part.	42
4.3	Dataset augmentation.	44
5.1	Excerpt of a dialogue with the original names (above) and the corrected names (below). .	50

List of Tables

3.1	Summary example from BART.	24
3.2	Abstractive summarization performance.	27
3.3	Performance for meeting summarization (left) and chat summarization (right) models.	33
5.1	ROUGE scores of the generated summaries for the different DialogSum versions.	51
5.2	Average Precision and Recall scores at word level for Dialogue and Relevant Turns entries when compared to Summary in the Augmented dataset.	51
5.3	ROUGE scores of the retrieved turns from the different retrievers.	52
5.4	ROUGE scores of the generated summaries for the different DialogSum versions.	53

Acronyms

BERT	Bidirectional Encoder Representation from Transformers
C4	Colossal Clean Crawled Corpus
CNN/DM	CNN/DailyMail
GPT	Generative Pre-trained Transformer
GSG	Gap Sentence Generation
GSR	Gap Sentence Ratio
LLM	Large Language Model
MLE	Maximum Likelihood Estimation
MLM	Masked Language Modeling
MLP	Multi-Layer Perceptron
NLP	Natural Language Processing
NQ	Natural Questions
NN	Neural Network
NSP	Next Sentence Prediction
RAG	Retrieval-Augmented Generation
RETRO	Retrieval-Enhanced Transformer
RoBERTa	Robustly Optimized BERT Approach
RMS	Root-Mean-Square
SGD	Stochastic Gradient Descent
SOTA	State of the Art
XSum	Extreme Summarization
MTEB	Massive Text Embeddings Benchmark

MNR	Multiple Negatives Ranking
nDCG	normalized Discounted Cumulative Gain
PLM	Permuted Language Model
kNN	k -nearest neighbors

1

Introduction

Contents

1.1 Motivation	3
1.2 Goals	3
1.3 Contributions	3
1.4 Outline	4

1.1 Motivation

Dialogue summarization aims at distilling the most important information from a dialogue into a shorter passage. This can help people capture the highlights of a conversation without having to review all the inherent context. Most previous work on summarization has focused on well-organized, single-speaker documents (e.g., news or scientific articles), and methods based on encoder-decoder neural networks have achieved remarkable results. However, despite achieving impressive results on abstractive summarization, these methods cannot easily be applied onto dialogue, since the latter adds challenges that are unique to it, like long inputs and inherent topic drifts. This offers a good opportunity for a different, better approach that is especially suitable for dialogue summarization.

1.2 Goals

In this dissertation, I set out to explore pre-trained neural language models for dialogue summarization by focusing on retrieval-augmented Transformers (Borgeaud et al., 2021; Martins et al., 2022) as the aforementioned better approach. With these models, we can identify and retrieve the essential passages from a dialogue to then pass them onto a language model for summarization. Thus, the research question becomes: *can I improve summary quality by retrieving the relevant parts of a dialogue and summarizing only those?* The proposed approach was evaluated on a dialogue-summarization dataset and the validity of the research question was confirmed.

1.3 Contributions

The chief contributions of this thesis are:

- The development of a new Retrieval-Augmented Generation (RAG) approach for dialogue summarization that alleviates dialogue summarization-specific problems (long inputs and inherent topic drifts), i.e, will result in better ROUGE scores. Experiments with this approach show that with a massive input reduction (less text for the language model to summarize) summary quality is maintained and even surpassed with some retrieval techniques. The source code is publicly available¹.
- The augmentation² of an existing dialogue summarization dataset by adding the pivotal passages from each dialogue in the original dataset using GPT-4³. Summaries with only these sentences were generated and compared to those generated with the entire dialogues. Baseline results with this dataset are also presented.

¹<https://github.com/diogofouto/retrieval-augmented-dialogue-summarization>

²<https://huggingface.co/datasets/diogofouto/dialogsum-augmented>

³<https://openai.com/research/gpt-4>

Dialogue

Robina: Good afternoon, sir. Can I help you?

Meredeth: I'd like a beef-burger, a French fries and a milk shake, please.

Robina: What flavor would you prefer, sir?

Meredeth: I'm not quite sure. What do you have?

Robina: We have strawberry, chocolate, vanilla and banana.

Meredeth: Very well, I'll try the banana flavor.

Robina: Anything else, sir?

Meredeth: No, thanks. That will be all.

Summary

Meredeth orders a beef-burger, french fries, and a banana milkshake at Robina's shop.

Relevant Turns

Meredeth: I'd like a beef-burger, a French fries and a milk shake, please.

Robina: We have strawberry, chocolate, vanilla and banana.

Meredeth: Very well, I'll try the banana flavor.

Figure 1.1: Relevant turns from a dialogue and respective summary.

1.4 Outline

This dissertation is organized as follows: Chapter 2 gives the reader an introduction to the fundamental concepts necessary to understand this work. It starts from the very basics of Neural Network (NN) and advances toward the Transformer architecture and its many implementations. Chapter 3 presents previous work on the three most relevant areas for this dissertation: Document Summarization, Retrieval Augmentation, and Dialogue Summarization. Chapter 4 describes the proposed approach for dialogue summarization with RAG. Chapter 5 covers the evaluation performed for this approach in depth. Chapter 6 gathers the main findings and suggests paths for future work.

2

Fundamental Concepts

Contents

2.1 Neural Networks	7
2.2 Transformer Language Models	11
2.3 The Summarization Task	18
2.4 Summary	19

In this chapter we review the theory required to understand this thesis. We will start with neural networks and how they learn. Then, we will discuss the Transformer architecture and a learning technique called transfer learning. Afterward, we will peruse models like Bidirectional Encoder Representation from Transformers (BERT) (Devlin et al., 2018) and Generative Pre-trained Transformer (GPT) (Radford et al., 2018), which are the foundation for much of the work in Chapter 3. Lastly, we will formally define the summarization task.

2.1 Neural Networks

Neural networks are a type of machine learning algorithms based on the human brain. In a neural network, a neuron (or node) is a computational unit with input links and one output link. It sums up the input values, applies an activation function to this sum, and outputs the outcome onto the next neurons in the network. The simplest neural network is called the Perceptron (McCulloch and Pitts, 1943).

2.1.1 Perceptron

The Perceptron is a neural network with just one node. This node is a binary classifier. It receives an input x , weighs it through the dot product with a vector of weights w , and adds a bias term b :

$$f(x) = x \cdot w + b. \quad (2.1)$$

The strength of this model lies on its simplicity. It is useful for decision problems whose solutions are linearly separable. For more complex problems (i.e., not linearly separable), we need more neurons.

2.1.2 Multi-Layer Perceptron

A Multi-Layer Perceptron (MLP) is composed of three or more layers of neurons. The first layer is the input; the layers in the middle (the hidden layers) and the last layer (the output layer) consist of neurons with a non-linear activation function (unlike the Perceptron). This makes the model useful for decision problems whose solutions are not linearly separable.

We represent a hidden layer's activation function, with weight matrix \mathbf{W}_i and bias term b_i , like so:

$$h_i(x) = x \cdot \mathbf{W}_i + b_i. \quad (2.2)$$

Let us now consider an example of a MLP with two hidden layers. The input is x , and the weight matrix and bias term for layer i are \mathbf{W}_i and b_i , respectively. The MLP is as follows:

$$\begin{aligned}
h_1(\mathbf{x}) &= \mathbf{x} \cdot \mathbf{W}_1 + \mathbf{b}_1, \\
h_2(\mathbf{x}) &= h_1(\mathbf{x}) \cdot \mathbf{W}_2 + \mathbf{b}_2, \\
f(\mathbf{x}) &= h_2(\mathbf{x}) \cdot \mathbf{W}_3 + \mathbf{b}_3.
\end{aligned}
\tag{2.3}$$

2.1.3 Supervised Learning

To approximate a mathematical function, these networks need the right weights. The process of finding those is called *learning*. We will cover a learning technique called *supervised learning*.

In supervised learning, we feed the network a training set of n examples, $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_n)$, and corresponding labels, $\mathbf{Y} = (\mathbf{y}_1, \dots, \mathbf{y}_n)$. Then, a learning algorithm strives to return the most accurate predictions $\hat{\mathbf{y}} = f(\mathbf{x})$ (see Equation 1).

To measure this accuracy, we need a loss function, $L(\hat{\mathbf{y}}, \mathbf{y})$, which measures the error between $\hat{\mathbf{y}}$ and \mathbf{y} . In the previous MLP example, the learning algorithm would set values for the matrices \mathbf{W}_n and the \mathbf{b}_n bias terms that minimize that loss. We can formally define a loss function L , with θ as the parameter set and $\mathcal{L}(\theta)$ the average loss over the training set, as:

$$\mathcal{L}(\theta) = \frac{1}{n} \sum_{i=1}^n L(f(\mathbf{x}_i; \theta), \mathbf{y}_i).
\tag{2.4}$$

The MLP's training goal is to find the parameter values that minimize the loss function:

$$\hat{\theta} = \arg \min_{\theta} \mathcal{L}(\theta).
\tag{2.5}$$

A curious side-effect of brute loss-minimization is a problem called *overfitting*: the production of a model which sticks too closely to its training data. A usual way to minimize this is by restricting the solution. With $T(\theta)$ as a regularization term and λ a regularization factor, $\hat{\theta}$ becomes:

$$\hat{\theta} = \arg \min_{\theta} (\mathcal{L}(\theta) + \lambda T(\theta)).
\tag{2.6}$$

What we now have is an optimization problem: minimizing a function by minutely fetching values from the training set and updating the parameters accordingly. A common way to solve this is with gradient calculus: applying the chain-rule to solve consecutive transformations.

An algorithm called *backpropagation* gives us this gradient. Once we get it, we use an optimization algorithm to improve the weight and bias values. A well-known example of such an algorithm is *gradient descent*. It minimizes a function $g(\theta)$ by updating θ in the direction of the negative gradient of g relative to θ , which is $\nabla g(\theta)$. To speed up this update, we use a learning rate α .

The simplest form of gradient descent is batch gradient descent. This computes the gradient of the cost function relative to θ for the entire training set in a single update:

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \alpha \nabla g(\boldsymbol{\theta}_t). \quad (2.7)$$

This approach guarantees global minimum convergence and scales well with the number of features. However, it requires calculations over the entire training set for each gradient-computation step, which can be very slow.

Stochastic Gradient Descent (SGD) alleviates this. Like the name implies, it chooses an instance from the training set randomly at each step and computes the gradient for that instance. This rules out the possibility of repeated similar update computations. With x_i as the training instance and y_i as the corresponding label, this means:

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \alpha \nabla g(\boldsymbol{\theta}_t, \mathbf{x}_i, \mathbf{y}_i). \quad (2.8)$$

Because of these individual update steps, the stochastic approach is much faster than batch. Nonetheless, for exactly the same reason, its result is always fluctuating and does not necessarily converge to the global minimum like its predecessor. To mitigate this, there is the Momentum (Qian, 1999) technique.

Momentum changes the update rule by adding a portion of the previous update step to the current one. This reduces result fluctuation and accelerates convergence towards the global minimum. With γ as the aforementioned portion and s_t as the update step, we get:

$$\begin{aligned} s_t &= \alpha \nabla g(\boldsymbol{\theta}_{t-1}) + \gamma s_{t-1}, \\ \boldsymbol{\theta}_t &= \boldsymbol{\theta}_{t-1} - s_t. \end{aligned} \quad (2.9)$$

For better performance in problems with sparse gradients (e.g., Natural Language Processing (NLP) problems), there is Adam (Kingma and Ba, 2014). This method abandons the one-size-fits-all approach to learning rates and introduces individual, adaptive parameter learning rates based on estimates for the first and second moments of the gradients:

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1 - \beta_1) \cdot \nabla g(\boldsymbol{\theta}_{t-1}), \\ s_t &= \beta_2 s_{t-1} + (1 - \beta_2) \cdot (\nabla g(\boldsymbol{\theta}_{t-1}))^2, \end{aligned} \quad (2.10)$$

where m_t (the estimate of the first moment) is the average of past gradients, s_t (the estimate of the second moment) is the average of past squared gradients, and β_1 and β_2 are the momentum portions (equivalent to γ in Eq. 2.9).

Since m_t and s_t start as null vectors, they are inherently biased toward $\mathbf{0}$. To correct this, we need:

$$\begin{aligned} \hat{m}_t &= \frac{m_t}{(1 - \beta_1^t)}, \\ \hat{s}_t &= \frac{s_t}{(1 - \beta_2^t)}. \end{aligned} \quad (2.11)$$

Finally, with ϵ as a small constant to avoid division by zero, here is the Adam update rule:

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \hat{m}_t \left(\frac{\alpha}{\sqrt{\hat{s}_t + \epsilon}} \right). \quad (2.12)$$

Despite bringing in a smoother descent and better results, Adam requires keeping two additional values for each parameter in memory – the momentum and the average of past squared gradients – which results in tripled memory usage when compared to vanilla SGD. To alleviate this, (Shazeer and Stern, 2018) suggest AdaFactor. This optimization technique reduces memory usage for the squared gradients from $O(mn)$ to $O(m+n)$ while keeping similar performance, where m and n are the row and column lengths of the squared gradients matrix, respectively. It does this through five innovations.

The first is called factored second moment estimation. AdaFactor replaces the average of past squared gradients with a low-rank approximation that is optimal with respect to the generalized Kullback-Leibler divergence (Lee and Seung, 1999) by tracking moving averages of the row and column sums like so:

$$\begin{aligned} \mathbf{R}_t &= \beta_2 \mathbf{R}_{t-1} + (1 - \beta_2) \cdot (\nabla g(\boldsymbol{\Theta}_{t-1}))^2 \mathbf{1}_m, \\ \mathbf{C}_t &= \beta_2 \mathbf{C}_{t-1} + (1 - \beta_2) \mathbf{1}_n^\top \cdot (\nabla g(\boldsymbol{\Theta}_{t-1}))^2, \\ \hat{\mathbf{S}}_t &= \frac{(\mathbf{R}_t \mathbf{C}_t / \mathbf{1}_n^\top \mathbf{R}_t)}{(1 - \beta_2^t)}, \end{aligned} \quad (2.13)$$

where $\boldsymbol{\Theta}$ is the parameter set $\boldsymbol{\theta}$ viewed as a single matrix, \mathbf{R}_t and \mathbf{C}_t are the averages of the row sums and column sums of the squared gradients at time t , respectively, $\hat{\mathbf{S}}_t$ is the factored second moment estimate at time t , and $\mathbf{1}_l$ is a normalization vector containing l ones.

The second is abandoning the other additional value in memory by simply turning momentum off:

$$\beta_1 = 0. \quad (2.14)$$

The third is update clipping, which addresses the possible “instability” of the second moment; second moment estimates based on gradients far in the past can have high error, which can lead to larger-than-desired updates. AdaFactor clips an update \mathbf{U}_t when its Root-Mean-Square (RMS) exceeds a threshold value d :

$$\begin{aligned} \hat{\mathbf{U}}_t &= \frac{\mathbf{U}_t}{\max(1, \text{RMS}(\mathbf{U}_t)/d)}, \\ \mathbf{U}_t &= \frac{\nabla g(\boldsymbol{\Theta}_{t-1})}{\sqrt{\hat{\mathbf{S}}_t}}, \\ \text{RMS}(\mathbf{U}_t) &= \sqrt{\text{mean}_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \left(\frac{(\nabla g(\boldsymbol{\theta}_{t-1}))^2}{\hat{s}_t} \right)}. \end{aligned} \quad (2.15)$$

The fourth, which is another solution for the aforementioned instability, is an increase of the decay parameter with time:

$$\hat{\beta}_{2t} = 1 - \frac{1}{tc}, \quad (2.16)$$

where $t \geq 1$ and $c > 0$.

Finally, AdaFactor's optimization algorithm is defined in terms of relative step sizes instead of absolute step sizes. This way, bigger-valued parameters get bigger-sized updates:

$$\alpha_t = \max(\epsilon, \text{RMS}(\Theta_{t-1})) \rho_t, \quad (2.17)$$

where α_t is the absolute step size, ρ_t the relative step size, and ϵ a small constant to allow parameters initialized as 0 to escape 0.

The update rule then becomes:

$$\Theta_t = \Theta_{t-1} - \alpha_t \hat{U}_t. \quad (2.18)$$

2.2 Transformer Language Models

Transformer models are a type of neural network architecture introduced by [Vaswani et al. \(2017\)](#). They have become the foundation technology used to tackle difficult NLP tasks like language translation, text summarization, question answering, and more. We will explore them first by reading about the original Transformer and then about newer models that build on that foundation.

2.2.1 The Original Transformer

The original Transformer follows the typical encoder-decoder structure of sequence-to-sequence (seq2seq) models (see Figure 2.1). The encoder transforms an input sequence of symbol representations $\mathbf{x} = (x_1, \dots, x_n)$ into a sequence of continuous representations $\mathbf{z} = (z_1, \dots, z_n)$. Then, the decoder uses \mathbf{z} to generate an output sequence $\mathbf{y} = (y_1, \dots, y_m)$. This structure can feature encoder and decoder stacks of arbitrary size.

The encoder comprises two sub-layers. The first is a multi-head attention mechanism, the second is a simple position-wise fully-connected feed-forward network. Going around each of them is a residual connection, which is then added to the product of the sub-layer and normalized: $\text{LayerNorm}(\mathbf{x} + \text{SubLayer}(\mathbf{x}))$.

The decoder features one sub-layer, which performs a masked multi-head attention over the output of the encoder. ("Masked" means that prediction for a certain position can depend only on the outputs of the positions that preceded it.) It also features the residual connection and normalization of the encoder.

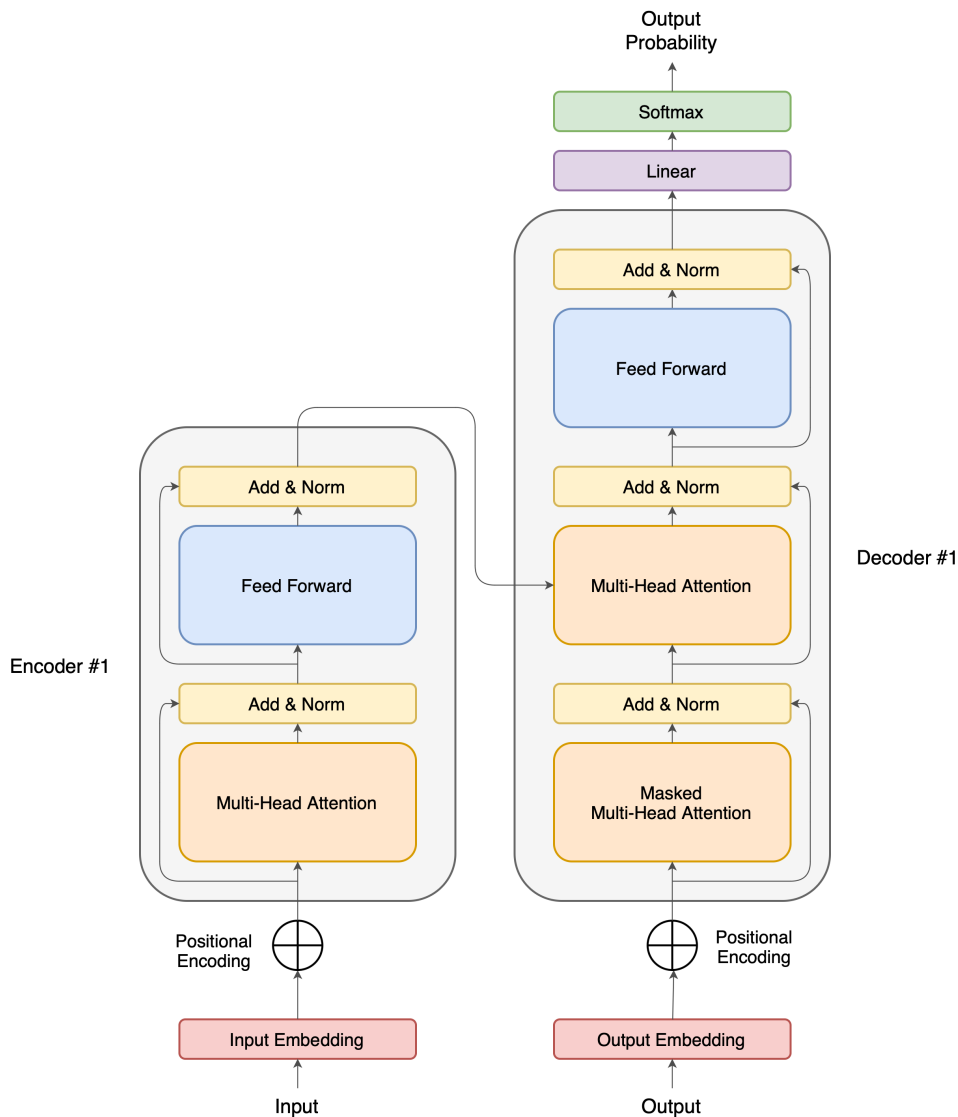


Figure 2.1: The Transformer architecture.

The attention mechanism is what makes the Transformer shine. Very briefly, attention is the mapping of a query and a set of key-value pairs to an output. The Transformer employs self-attention to provide context, which simply means that relevant input tokens influence the token we are currently processing.

The authors of the Transformer employ a form of attention called scaled dot-product attention. To compute it, they first create query, key, and value vectors for each word embedding by multiplying it by weight matrices that the model learned during training. Then, they multiply the query vector with the transposed key vector and scale the result by the square root of the key vector's size. Finally, they multiply this outcome by the value vector. In mathematical parlance, this means:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right) \mathbf{V}, \quad (2.19)$$

where \mathbf{Q} , \mathbf{K} , \mathbf{V} are the query, key, and value vectors, respectively, and d_k is the query and key vector size.

The aforementioned multi-head attention mechanism expands this attention concept to allow the model to jointly attend to information from different representation subspaces by employing h parallel attention layers, or *heads*. For this, we calculate attention in each *head*, concatenate the outcomes, and project them onto the original dimensions:

$$\begin{aligned} \text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) &= \text{Concat}(\mathit{head}_1, \dots, \mathit{head}_h) \mathbf{W}^O, \\ \mathit{head}_i &= \text{Attention}(\mathbf{W}_i^Q, \mathbf{W}_i^K, \mathbf{W}_i^V), \end{aligned} \quad (2.20)$$

where $\mathbf{W}_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $\mathbf{W}_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $\mathbf{W}_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$, $\mathbf{W}^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$, and d_v is the value vector size.

Besides the attention sub-layer, there is a fully-connected feed-forward network sub-layer. This network comprises two linear transformations and a ReLU activation:

$$\text{FeedForward}(\mathbf{x}) = \max(\mathbf{0}, \mathbf{x} \cdot \mathbf{W}_1 + \mathbf{b}_1) \cdot \mathbf{W}_2 + \mathbf{b}_2. \quad (2.21)$$

Before any of these stages happen, though, the input embedding module transforms the input tokens into embeddings of size $d_{\text{model}} = 512$. This word-embedding stage abstracts every input token into a lower-dimension, continuous space that preserves the token's contextual similarity to other tokens. However, the absence of recurrence or convolution in the model means that the latter has no clue about the position of the tokens in the sequence. To amend this, the authors add "positional encodings" to the embeddings.

Positional encodings contain information about the relative or absolute position of the input tokens in the sequence. These encodings have the same dimension, d_{model} , as the embeddings, which means that the two can be summed. To bring them about, the authors resorted to simple sine and cosine functions. With pos as the position and i as the dimension, this begets:

$$\begin{aligned} \text{PE}(pos, 2i) &= \sin\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right), \\ \text{PE}(pos, 2i + 1) &= \cos\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right). \end{aligned} \quad (2.22)$$

For a smoother introduction to the Transformer, I refer the interested reader to a popular tutorial¹. Now, let us see how Transformers learn.

¹<http://jalamar.github.io/illustrated-transformer/>

2.2.2 Transfer Learning

The Transformer architecture shines brighter when combined with *transfer learning*. This technique extends isolated, supervised learning by leveraging data from additional NLP tasks, which gives the model better generalization abilities. It consists of pre-training a model on a data-heavy task before fine-tuning it on a downstream task. The flurry of works within this framework has produced a plethora of model architectures, pre-training objectives, fine-tuning methods, datasets, evaluation metrics, and has now become State of the Art (SOTA) (Devlin et al., 2018; Liu et al., 2019; Radford et al., 2018).

To cover all the aforementioned concepts and give a consistent perspective on transfer learning, I will resort to the colossal Raffel et al. (2020) study. Its basic premise was to perceive every NLP task as a “text-to-text” problem — text in, text out — which allows for directly applying the same model onto every task.

Model architecture. Seq2seq has become the dominant framework and the “workhorse” of NLP because of its powerful encoder-decoder architecture (Sutskever et al., 2014). Nowadays, this framework is usually implemented with variants of the Transformer. Raffel et al. (2020) study two major variants. The first is the classic encoder-decoder. The second is just the decoder stack, which, if you recall from before, samples a token from the model’s predicted distribution and feeds it back into the model to generate the next token. The authors compare these variants experimentally with both a basic language modeling objective and a denoising objective as unsupervised pre-training strategies. They conclude that the classic encoder-decoder Transformer architecture outperforms the other in every NLP task.

Pre-training. Pre-training is the mechanism through which the model learns general knowledge. Therefore, the choices for pre-training framework, corpus, and objective are crucial. Traditionally, the pre-training framework is Maximum Likelihood Estimation (MLE) with a cross-entropy loss:

$$L(f(\mathbf{x}; \boldsymbol{\theta}), \mathbf{y}) = - \sum_{i=1}^n \mathbf{y} \log f(\mathbf{x}; \boldsymbol{\theta}), \quad (2.23)$$

where \mathbf{x} is a training instance, \mathbf{y} is the true label, $\boldsymbol{\theta}$ is the set of parameters, f is the training function, and n is the number of labels.

The *teacher-forcing* (Williams and Zipser, 1989) strategy is also commonplace: when training, the inputs are the previous ground-truth tokens; when testing, the inputs are the previous predicted tokens.

For the pre-training corpus, we need a dataset big-enough to avoid repeating examples, since these can be detrimental (Raffel et al., 2020). Because no labeled dataset satisfies this, we have to make do with unlabeled datasets like the Colossal Clean Crawled Corpus (C4) (Raffel et al., 2020).

This leaves us with the choice of pre-training objective. [Raffel et al. \(2020\)](#) study three disparate objectives, alike only from a very general point of view: they all consume a tokenized sequence of text from an unlabeled dataset, corrupt it in some way, and build a target output that the model has to predict with MLE. The first is a simple language modeling objective. It halves the input and uses the first half as the encoder input and the other as the decoder target sequence. The second is Masked Language Modeling (MLM). It corrupts a random sample of input tokens and replaces them with [MASK] tokens. In an encoder-only model, the pre-training goal is to predict the masked tokens. In an encoder-decoder model, the uncorrupted input sequence becomes the decoder target sequence. The third objective is deshuffling, which simply ingests an input sequence, shuffles it, and tries to deshuffle it back to original form. The authors test each objective against several NLP task-specific datasets and find that MLM performs best. Besides pre-training on a single task, the authors also experimented with *multi-task learning*, which pre-trains the model on multiple tasks at once. They did not see improvements.

Fine-tuning strategies. [Peters et al. \(2019\)](#) argues that fine-tuning every parameter leads to suboptimal results. [Raffel et al. \(2020\)](#) studies two alternative fine-tuning strategies. The first involves “adapter modules” ([Houlsby et al., 2019](#)), which are feed-forward layers applied to the output of the feed-forward sublayers of the Transformer. When fine-tuning, only the adapter module and layer normalization parameters are tweaked. This keeps most of the original model fixed and lowers fine-tuning time. The second is gradual unfreezing ([Howard and Ruder, 2018](#)). Here, the number of fine-tuned parameters grows with time. In the first update cycle, only parameters of the last layer of both encoder and decoder are fine-tuned, the rest are frozen. In the next cycle, the second-to-last thaw and are also fine-tuned. And so on. Although promising, none of these strategies matched the performance of all-parameter fine-tuning.

Decoding. During inference, a pre-trained model generates the output text autoregressively. However, it is unfeasible to enumerate all possible output tokens. To reduce this search, a technique called *beam search* is commonplace. Beam search is an optimization of breadth-first search. At each level of the search tree, it generates all successor states, sorts them according to some heuristic, and stores the w best ones. At the next level, it expands only those stored states. w is known as beam width. Higher beam width means more states to expand and greater memory usage. Because beam width is not infinity, the algorithm is neither complete nor optimal. If $w = 1$, then beam search is equivalent to *greedy search*: selecting the state with the least cost from the current node as the next state.

Scaling. The “bitter lesson” of machine learning ([Sutton, 2019](#)) argues that models which benefit from

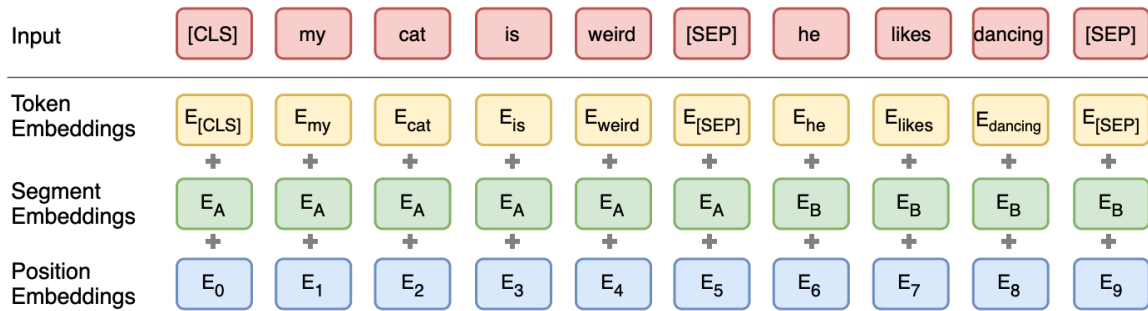


Figure 2.2: BERT input representation.

additional computations ultimately triumph over cleverly designed models. Recent research suggests the same is true for transfer learning (Radford et al., 2019).

What follows is an overview of three foundational transfer learning-based models. BERT introduces a bidirectional encoder. GPT presents an autoregressive decoder. BART (Lewis et al., 2019) combines the two into a model that is applicable to a wide range of NLP tasks.

2.2.3 BERT

BERT (Devlin et al., 2018) is a language representation model designed to pre-train deep bidirectional representations from unlabeled text. It adopts and expands the encoder module of the Transformer. Its framework features an input/output representation step and the two major transfer learning stages: pre-training and fine-tuning.

For the input/output representation step, the authors define *sentence* as an arbitrary span of contiguous text and *sequence* as an input token sequence. The input representation comprises three distinct embeddings for each token: a token embedding based on WordPiece (Wu et al., 2016); a segment embedding to tell which sentence the token belongs to within a sequence; and a position embedding to store the global position of the token within the sequence. BERT also introduces two special classification tokens to distinguish sentences inside a sequence: [CLS] starts a sentence, and [SEP] ends one. The input representation for a token (see Fig. 2.2 for an example) is a simple summation of the aforementioned embeddings.

In pre-training, the model trains on unlabeled data over two unsupervised tasks. The first is the aforementioned MLM. BERT chooses 15% of the input tokens for possible replacement. Of these, it masks 80%, replaces 10% with a random vocabulary token, and leaves the last 10% unchanged. The objective is to predict the masked tokens.

The second unsupervised task is called Next Sentence Prediction (NSP), which is essential for NLP tasks where understanding the relationship between two sentences is key. Here the authors generate an NSP task from a given monolingual corpus, which the model has to solve.

When fine-tuning, BERT’s pre-trained parameters are tweaked using labeled data from the intended downstream tasks. For each of these tasks, we feed BERT the proper inputs and outputs and fine-tune all its parameters. The self-attention mechanism of the original Transformer makes this a straightforward process.

Soon after BERT, [Liu et al. \(2019\)](#) presented Robustly Optimized BERT Approach (RoBERTa). The authors replicated BERT pre-training and found that it was deeply undertrained. They modified the training procedure and matched the (then) SOTA. This modification included adding more training data, removing the NSP task, replacing WordPiece with Byte-Pair Encoding ([Sennrich et al., 2015](#)), and adding dynamic masking.

2.2.4 GPT

GPT is a multi-layer Transformer decoder model designed to learn a universal representation applicable to a wide range of NLP tasks with minimum fine-tuning.

Given an unsupervised corpus of tokens $\mathbf{u} = (u_1, \dots, u_n)$ and a standard language modeling objective, pre-training consists of maximizing the following likelihood:

$$L_1(\mathbf{u}) = \sum_{i=1}^n \log P(u_i | u_{i-k}, \dots, u_{i-1}; \theta), \quad (2.24)$$

where k is the context window length, P is the conditional probability, and θ is the set of parameters (trained with SGD).

Given a labeled dataset \mathcal{C} with instances $\mathbf{x}_i = (x_1, \dots, x_m)$ and a label y_i , fine-tuning consists of maximizing the following likelihood:

$$L_2(\mathcal{C}) = \sum_{(\mathbf{x}_i, y_i)} \log (\text{softmax}(h(\mathbf{x}_i), \mathbf{W}_{y_i})), \quad (2.25)$$

where h is the final Transformer decoder activation function, and \mathbf{W}_{y_i} are the parameters for predicting y_i of the linear output layer.

2.2.5 BART

BART is a denoising autoencoder built with the standard Transformer architecture. It combines the bidirectional encoder of BERT with the autoregressive decoder of GPT (see Fig. 2.3). Because of this, BART merges the best of both worlds and is applicable to a wide range of tasks.

BART is pre-trained by (1) corrupting input text with an arbitrary noise function, and by (2) optimizing a reconstruction loss, which is used to reconstruct the original text. In (1), the authors try several corrupting transformations and settle for a combination of text infilling and sentence permutation. Text infilling,

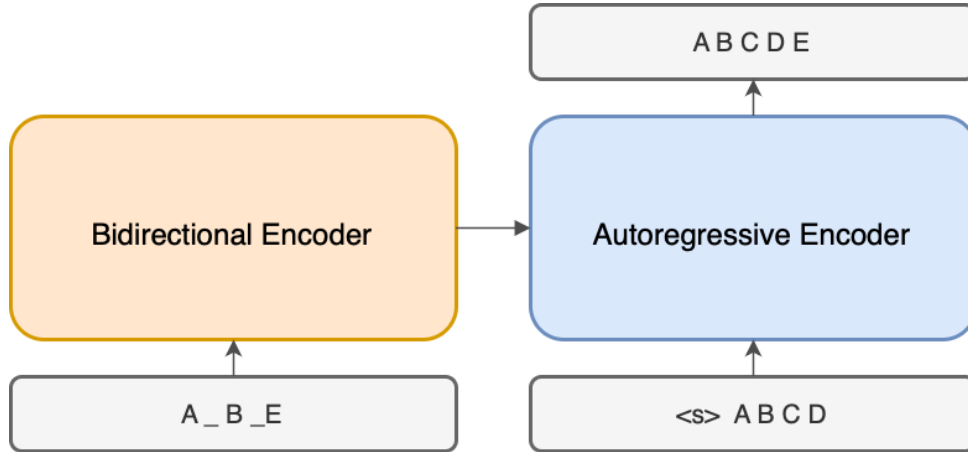


Figure 2.3: BART architecture. The corrupted document (left) is encoded with the bidirectional encoder, and the probability of a document (right) is calculated with the autoregressive decoder.

which was inspired by SpanBERT (Joshi et al., 2020), samples a number of test spans with length drawn from a Poisson distribution and replaces them with single [MASK] tokens. Sentence permutation means shuffling sentences of a document randomly. Then, in (2), the model has to learn to reorder the sentences and predict the number of tokens missing from a span.

Following this pre-training stage, BART can be fine-tuned for downstream tasks. For sequence classification, for example, we feed the same input into both encoder and decoder, and use the representation of the final output for the label. This process is related to the [CLS] token of BERT. For text generation tasks, like summarization, the model takes information from the input and manipulates it similarly to the denoising pre-training objective. The encoder takes as input the original text, and BART's GPT-inspired decoder generates output autoregressively.

2.3 The Summarization Task

Given a source document D and a reference summary S^* , the goal of a summarization model g is to produce a summary $S = g(D)$ that maximizes $m = M(S, S^*)$, where M is an evaluation metric.

Traditionally, this is formulated as a sequence-to-sequence problem (Sutskever et al., 2014) with MLM as the training framework. MLE maximizes the likelihood of S^* like so:

$$S = \arg \max_{S_i} h(S_i, D), \quad (2.26)$$

where h is a scoring function.

2.4 Summary

This chapter presented the theory required to understand the remaining of this dissertation. We started by introducing what a NN is, its variants, and how they learn. Then we moved onto the Transformer architecture and real-world applications of it in models like BERT and BART. With these fundamental concepts reviewed, we can now proceed onto the Related Work of this thesis.

3

Related Work

Contents

3.1 Document Summarization	23
3.2 Retrieval Augmentation for Text Generation	27
3.3 Dialogue Summarization	32
3.4 Summary	36

Here I discuss previous work on summarization and detail studies that are relevant for this thesis. This chapter comprises three sections: an overview of document summarization, an analysis of retrieval-augmented models, and a survey on dialogue summarization.

3.1 Document Summarization

Summarization is the task of condensing a text into a shorter version that captures the highlights of the former. In contrast to extractive summarization, which merely assembles excerpts from the input, abstractive summarization may use novel words to build the summary.

Rush et al. (2015) developed the first neural network-based seq2seq model for abstractive summarization. The authors combined a convolutional neural network to encode the source document with a context-sensitive attention-based feed-forward neural network to generate the summary. For datasets, they used Gigaword (Graff and Cieri, 2003), which contains a little less than ten million news articles sourced from various news sources, and DUC (Over et al., 2007), which contains 500 New York Times and Associated Press Wire news articles paired with four human-written reference summaries each. They achieved (then) SOTA on both. Nallapati et al. (2016) further improved performance by applying an off-the-shelf attention-based recurrent neural network and achieved new SOTA on the same datasets. Despite this, no neural network-based model was a match against the Transformer. Its introduction was *the* turning point in the field.

A Transformer model that serves as sandbox for much of the work in this section is BART (see Chapter 2 for a review). As experimental setup, the authors pre-trained a model with 12-layer encoder and decoder stacks, with hidden size of 1024 per layer. The authors followed RoBERTa (Liu et al., 2019) for batch size (8,000) and number of training steps (500,000), and used Byte-Pair Encoding for input document tokenization. For the pre-training objective, they masked 30% of the input tokens and permuted all sentences. For better data fitting, they disabled dropout for the last 10% of the training steps.

To provide a comparison with (then) SOTA, which was RoBERTa, the authors evaluated the experimental setup with ROUGE metrics on two summarization datasets. These are CNN/DailyMail (CNN/DM) and Extreme Summarization (XSum). CNN/DM (Hermann et al., 2015) was introduced for question-answering purposes and adapted by Nallapati et al. (2016) for text summarization. At the time of writing, it remains the benchmark for abstractive summarization. It consists of human-written abstractive summaries posed as questions from news stories found in the CNN and Daily Mail websites, and corresponding story passages from which a model is expected to formulate the answers. It features over 280,000 training pairs, 13,000 validation pairs, and 11,000 test pairs, which roughly corresponds to a 90-5-5 split. On average, source documents span 766 words over 29.74 sentences, and summaries 53 words over 3.72 sentences. XSum (Narayan et al., 2018) contains BBC articles covering a huge variety

Source Document (abbreviated)	BART Summary
<p>This is the first time anyone has been recorded to run a full marathon of 42.195 kilometers (approximately 26 miles) under this pursued landmark time. It was not, however, an officially sanctioned world record, as it was not an "open race" of the IAAF. His time was 1 hour 59 minutes 40.2 seconds. Kipchoge ran in Vienna, Austria. It was an event specifically designed to help Kipchoge break the two hour barrier.</p>	<p>Kenyan runner Eliud Kipchoge has run a marathon in less than two hours.</p>

Table 3.1: Summary example from BART.

of domains like business, science, news, entertainment, etc. It was created with the goal of answering the "What is the article about?" question in a short sentence. It comprises over 200,000 training pairs, 11,000 validation pairs, and 11,000 test pairs, which also corresponds to a 90-5-5 split.

BART triumphed over RoBERTa on both datasets by approximately 3.5 points on every ROUGE metric. Table 3.2 contains the exact results. Qualitatively, BART's summaries are fluid, grammatically correct, and overall factually right. Table 3.1 presents an example of a summary.

3.1.1 PEGASUS

Zhang et al. (2020) explores the previously uncharted territory of pre-training objectives tailored for abstractive summarization and hypothesizes that downstream task-oriented objectives result in better performance. The authors present PEGASUS, another seq2seq model with the standard Transformer encoder-decoder architecture, which features a pre-training objective called Gap Sentence Generation (GSG). GSG builds upon BART by masking whole sentences instead of smaller continuous text spans, and by concatenating them into a pseudo summary. (Gap Sentence Ratio (GSR) is the ratio of the number of gap sentences with the number of total document sentences.) When training, the encoder input is the original document with [MASK] tokens where the gap sentences are supposed to be, and the decoder input are the gap sentences. When testing, the encoder input is the same, and the desired decoder output is a single sequence of masked sentences. This technique benefits from masking and assumes the form of the abstractive summarization task.

They study three strategies for selecting m gap sentences. The first is called Random, which, like the name implies, selects m sentences randomly. The second is Lead, which selects the first m sentences. The last one is Principal, and it selects the top-scoring m sentences according to ROUGE-1:

$$s_i = \text{ROUGE-1}(x_i, D \setminus \{x_i\}), \quad (3.1)$$

where x_i is a sentence, D the sentence matrix corresponding to document D , and s_i the ROUGE-1 score.

As pre-training corpora, the authors consider C4 (see Chapter 2) and a new text corpus collecting news-like articles, including XSum and CNN/DM, named HugeNews. As fine-tuning datasets, they use, among others, XSum, CNN/DM, and Reddit TIFU (Kim et al., 2018). Reddit TIFU contains 120K posts of informal stories from the online discussion website Reddit that follow the rule of writing a summary at the end of every story. Since stories and online discussions resemble dialogue, one immediately sees the relevance of this dataset for this report.

Similar to Raffel et al. (2020), they evaluated several pre-training choices with a smaller-size model. (This type of evaluation is called an ablation study.) For the pre-training corpora, they witnessed better performance with HugeNews on XSum and CNN/DM and with C4 on Reddit TIFU. For the pre-training objective, Principal gap-sentence generation with 15% GSR on CNN/DM and with 30% on XSum and Reddit TIFU proved the best strategy. For tokenization methods, they compared Byte-Pair Encoding with SentencePiece Unigram (Kudo and Richardson, 2018), and found that Unigram 96k was overall better. Furthermore, on all models they use: sinusoidal positional encoding (Vaswani et al., 2017); AdaFactor, with a square-root learning rate decay and a dropout rate of 0.1, for pre-training and fine-tuning optimization; and beam search with beam width of 4 and length penalty of $\alpha = 0.6$.

The authors then scale up the best configuration to build a 568M-parameter PEGASUS. This model beats BART on the XSum by approximately two points on every ROUGE metric and on CNN/DM by half a point. Performance is shown in Table 3.2. Qualitatively, PEGASUS’s summaries are fluid and coherent like BART’s.

3.1.2 SimCLS

Despite showing remarkable results, the previous models share an obvious problem of seq2seq model training. As mentioned before, they train under the MLE framework, which introduces a gap between training objective and evaluation metric. The objective function concentrates on token-level predictions, while metrics like ROUGE measure sentence similarity between gold-reference summaries and model-generated summaries. To address this and further improve performance, Liu and Liu (2021) present SimCLS (Fig. 3.1).

SimCLS is a two-stage system. A seq2seq model generates candidate summaries with traditional MLE, and a second model ranks them with *contrastive learning*. If you recall the summarization task in Chapter 2, in mathematical parlance this means that the first stage learns to maximize the likelihood of a reference summary S^* and generates several summaries S_1, \dots, S_w with beam search, where w is the beam width. The second stage ranks these candidates against a source document D with a scoring function h . The contrastive learning part consists of minimizing the following loss function:

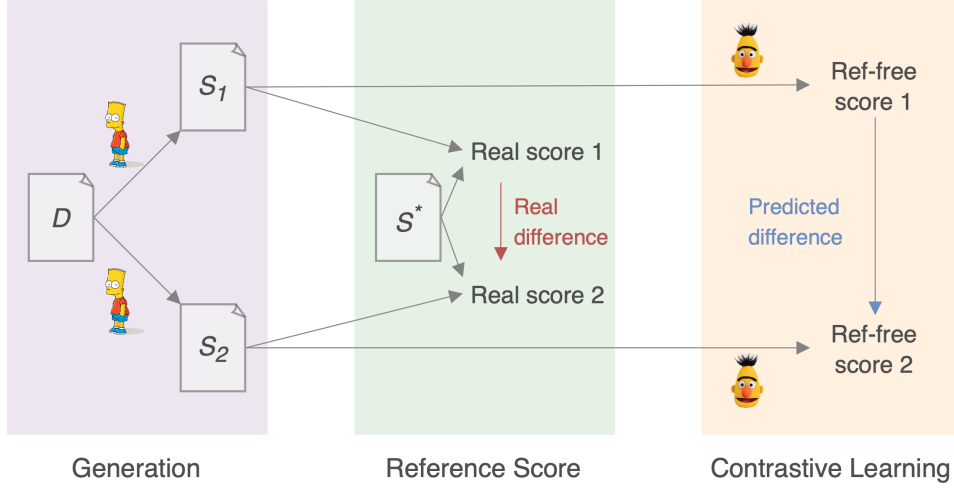


Figure 3.1: SimCLS framework. At the first stage (far left), BART generates candidate summaries. At the second stage (far right), RoBERTa ranks them against a source document.

$$\begin{aligned}
 L = & \sum_i \max(0, h(D, S_i) - h(D, \hat{S})) \\
 & + \sum_i \sum_{j>i} \max(0, h(D, S_j) - h(D, S_i) + \lambda_{ij}),
 \end{aligned} \tag{3.2}$$

where S_1, \dots, S_w are ordered descendingly by ranking score, which is ROUGE, and λ is a hyper-parameter as defined in [Zhong et al. \(2020\)](#).

For experimental setup, the authors used BART as the summary generator, RoBERTa as the ranking model, and cosine similarity as the scoring function h . For decoding, they used diverse beam search ([Vijayakumar et al., 2016](#)) with 16 beam widths to generate 16 summaries. For training, they used Adam with learning rate scheduling:

$$\alpha = 0.002 \times \min(t^{-0.5}, 1000^{-0.5}t), \tag{3.3}$$

where α is the learning rate and t is the update step number.

The authors then tested this setup against previous baselines on the CNN/DM and XSum datasets. It improved upon PEGASUS by roughly two points on CNN/DM on every ROUGE metric. On XSum, though, there was practically no difference (less than half a point). This may be because reference summaries in XSum only have one sentence, which limits semantic diversity. Exact performance is on Table 3.2.

R1/R2/RL	XSum	CNN/DM
RoBERTa	41.45/18.79/33.90	40.31/18.91/37.62
BART	45.14/22.27/37.25	44.16/21.28/40.90
PEGASUS	47.21/24.56/39.25	44.17/21.47/41.11
SimCLS	47.61/24.57/39.44	46.67/22.15/43.54

Table 3.2: Abstractive summarization performance.

3.2 Retrieval Augmentation for Text Generation

Pre-trained language models already retain significant knowledge within their parameters and without any access to external memory (Petroni et al., 2019). However, this knowledge is frozen in time and cannot be easily updated. To remedy this, there’s *retrieval augmentation*, which upgrades models with the ability to retrieve information from external, non-parametric databases.

Previous work shows that these retrieval-augmented models perform better across tasks like question answering (Chen et al., 2017), fact checking (Thorne et al., 2018), and dialogue (Dinan et al., 2018) than static models. It also shows reduced hallucination in dialogue agents (Shuster et al., 2021). Traditionally, this retrieval step consisted of term matching with metrics like BM25 (Robertson and Zaragoza, 2009) and TF-IDF (Jones, 1972). Lately, retrievers based on neural networks have become prevalent.

Lewis et al. (2020) presented a general-purpose fine-tuning recipe for models with such retrievers and called it RAG. A RAG model takes an input sequence x , retrieves a set of k documents $Z = (z_1, \dots, z_k)$, and uses both to generate a target sequence y of length n . The authors propose two approaches to marginalize a document z_j : RAG-Sequence, which uses the same document to predict all target-sequence tokens, and RAG-Token, which uses different documents for different tokens. Both feature a retriever, $P_\eta(z_j|x)$, which accesses a non-parametric database, and a generator, $P_\theta(y_i|x, z_j, y_{1:i-1})$, which contains the parametric memory. η and θ are parameter sets.

In RAG-Sequence, the retriever fetches the top k documents, and the generator produces an output sequence probability for each like so:

$$P_{\text{RAG-Sequence}}(\mathbf{y}|\mathbf{x}) \approx \sum_{z_j \in Z} P_\eta(z_j|\mathbf{x}) \prod_i^n P_\theta(y_i|\mathbf{x}, z, \mathbf{y}_{1:i-1}). \quad (3.4)$$

In RAG-Token, the retriever works the same way, but the generator picks content from several documents by producing a distribution over each for the next output token:

$$P_{\text{RAG-Token}}(\mathbf{y}|\mathbf{x}) \approx \prod_i^n \sum_{z_j \in Z} P_\eta(z_j|\mathbf{x}) P_\theta(y_i|\mathbf{x}, z, \mathbf{y}_{1:i-1}). \quad (3.5)$$

Following this recipe, the authors made their own RAG model. For the generator, they used BART, and, when producing the output, they combine the input sequence x with a retrieved document z_j through a simple concatenation. For the retriever, they followed DPR (Karpukhin et al., 2020) and combined two BERT models like so:

$$\begin{aligned} P_{\eta}(z_j|x) &\propto \exp(d(z_j)^{\top} q(x)), \\ d(z_j) &= \text{BERT}_d(z_j), \\ q(x) &= \text{BERT}_q(x), \end{aligned} \tag{3.6}$$

where d is the document encoder and q is the query encoder.

The retriever and generator train together by modeling documents as latent variables. For fine-tuning, the authors use stochastic gradient descent with Adam. For decoding, they use beam search.

As their retrieval knowledge source, the authors use the December 2018 Wikipedia dump. Each article is cut into 100-word documents, which results in 21M documents.

The authors tested it on open-domain question answering, which is knowledge-intensive, with the Natural Questions (NQ) dataset (Kwiatkowski et al., 2019). NQ contains over 300,000 training pairs, 7,000 validation pairs, and 7,000 testing pairs. Each pair comprises a Google query and the corresponding Wikipedia page. On this dataset, RAG-Token scored 44.1 points and RAG-Sequence scored 44.5.

3.2.1 Retrieval-Enhanced Transformer

The previous model retrieves information from databases with up to billions of tokens. In contrast, Borgeaud et al. (2021) designed a retrieval-augmented model capable of retrieving from a database with up to trillions of tokens. It features the typical Transformer architecture (see Fig. 3.2 for an overview) and incorporates retrieved text with a chunked cross-attention module. They call it Retrieval-Enhanced Transformer (RETRO).

The authors first build an external database with a key-value memory. Each value is a pair of neighboring text chunks (n_j, f_j) . The key is the BERT embedding of n_j , or $\text{BERT}(n_j)$. For a chunk c_i of an input sequence x split into l chunks of size m , the retriever fetches the k -nearest neighbors (kNN) from the database with L_2 distance as the metric:

$$\begin{aligned} d(c_i, n_j) &= \|\text{BERT}(c_i) - \text{BERT}(n_j)\|_2^2, \\ \text{Ret}(c_i) &= ((n_1, f_1), \dots, (n_k, f_k)), \end{aligned} \tag{3.7}$$

where $d(c_i, n_j)$ is the L_2 distance and $\text{Ret}(c_i)$ are the retrieved documents. For a database of t documents, retrieval time is $O(\log t)$.

A Transformer encoder ingests these documents and yields the encoded neighbors set E . Then, a

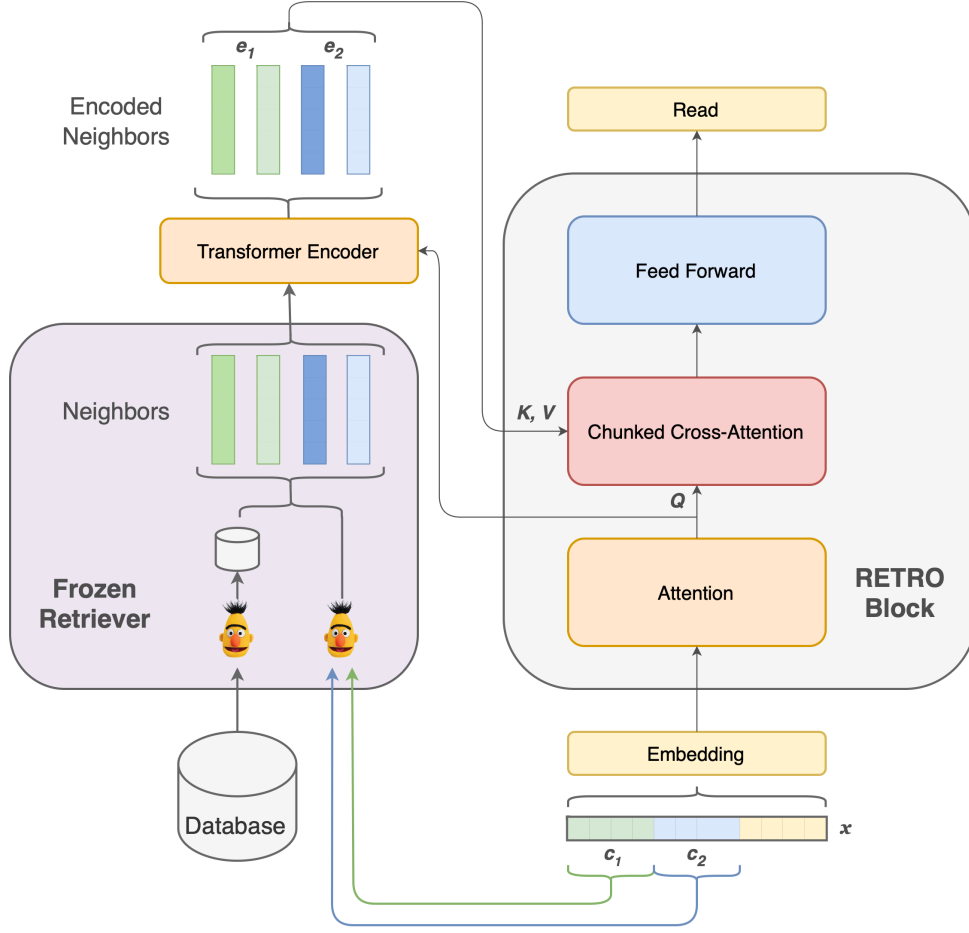


Figure 3.2: RETRO architecture.

decoder, with intermediate activation functions \mathbf{H} , built of alternate layers of RETRO blocks, $\text{RETRO}(\mathbf{H}, \mathbf{E})$, and standard Transformer decoders, $\text{LM}(\mathbf{H}, \mathbf{E})$, predicts the output sequence:

$$\begin{aligned} \text{RETRO}(\mathbf{H}, \mathbf{E}) &= \text{FeedForward}(\text{CCA}(\text{Attention}(\mathbf{H}), \mathbf{E})), \\ \text{LM}(\mathbf{H}) &= \text{FeedForward}(\text{Attention}(\mathbf{H})), \end{aligned} \quad (3.8)$$

where $\text{CCA}()$ is the chunked cross-attention module.

To perform the cross-attention step (Figure 3.3), an intermediate activation function \mathbf{h} split into l chunks h_i is further split into $l - 1$ chunks h_i^+ . Each h_i^+ contains the last token of h_i and the first $m - 1$ tokens of h_{i+1} . The model then computes cross-attention between h_i^+ and e_i like so:

$$\text{CA}(h_i^+, e_i) = \text{softmax}(e_i K Q^\top h_i^+) e_i V, \quad (3.9)$$

where K, Q, V are parameter matrices.

For experimental setup, the authors built a baseline model and a RETRO model. The baseline

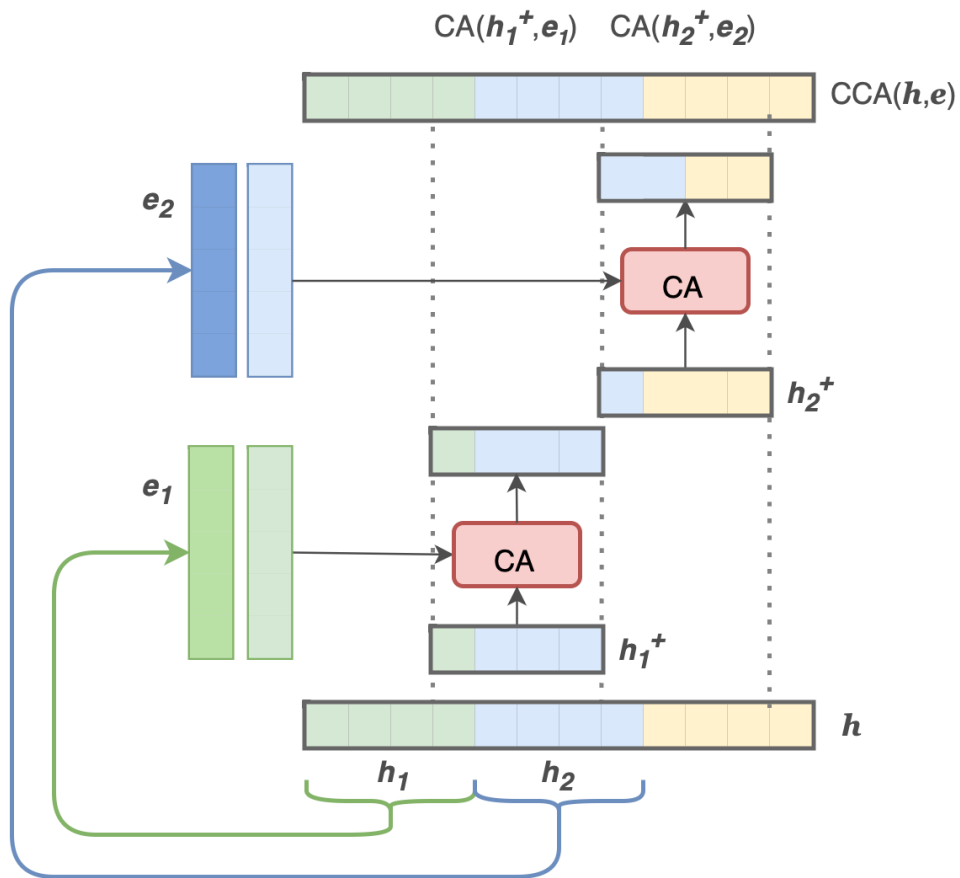


Figure 3.3: Cross-attention step.

is the original Transformer with minimal changes: RMSNorm (Zhang and Sennrich, 2019) instead of LayerNorm, and relative position encodings (Shaw et al., 2018) instead of absolute position encodings. The RETRO model uses a two-layer encoder for retrieval data, which adds 19M parameters over the baseline, and a decoder that contains one RETRO-block every three layers, which adds a further 4M parameters per block.

For pre-training and retrieval data, the authors rely on MassiveText (Rae et al., 2021), which comprises text documents with over five trillion tokens in total and which they tokenize with SentencePiece (Kudo and Richardson, 2018). For testing on language modelling tasks, they use C4 and Wikitext103 (Merity et al., 2016). For fine-tuning and testing, they used the NQ dataset.

RETRO scored one point higher than RAG and 15.1 points higher than the baseline.

3.2.2 Atlas

Izacard et al. (2022b) present a thorough study on how to design and train retrieval-augmented models that includes joint retriever and language model training procedures and efficient document index

updates. The authors then combine their findings and build Atlas, the current SOTA on retrieval augmentation.

For training the retriever, the authors discuss several loss functions which rest on the assumption that, if the language model deems a document useful, then the training objective should nudge the retriever to fetch said document. Two important loss functions for this project are Attention Distillation and Perplexity Distillation.

Attention distillation is based on the attention scores of the language model. These scores function as a proxy of the importance of each document and can be distilled into the retriever by minimizing the Kullback-Leibler divergence with the probability distribution over the top- k retrieved documents $D = (d_1, \dots, d_k)$, which is:

$$P_{\text{retrieval}}(d_i|q) = \frac{\exp(d_i \cdot q)}{\sum_{j=1}^k \exp(d_j \cdot q)}, i \in \{1, \dots, k\}, \quad (3.10)$$

where q is the query for the retrieval of D .

The attention scores are then summed across attention heads, layers, and tokens into a single score for every top- k document. To obtain the distribution $P_{\text{attention}}(d_i)$, we apply the softmax operator over the resulting k scores.

The loss function then becomes:

$$\text{KL}(P_{\text{retrieval}} || P_{\text{attention}}) = \sum_{j=1}^k P_{\text{attention}}(d_j) \log \left(\frac{P_{\text{attention}}(d_j)}{P_{\text{retrieval}}(d_j)} \right). \quad (3.11)$$

Perplexity distillation trains the retriever to predict the impact of each document on the language model perplexity. For this, the retriever minimizes the Kullback-Leibler divergence between the distribution in Equation 3.10 and the documents distribution according to the language model with a uniform prior output a :

$$P_i = \frac{\exp(\log P_{\text{LM}}(a|d_i, q))}{\sum_{j=1}^k \exp(P_{\text{LM}}(a|d_j, q))}, i \in \{1, \dots, k\}. \quad (3.12)$$

Then the authors turn to the problem of efficient document index updates. When fetching a document, the retriever uses a document index, which is simply a collection of embeddings for the available documents in the retrieval database. When training and fine-tuning, this index should be frequently updated to avoid documents going stale. However, it can add a huge computational overhead. To alleviate this, the authors compare two update strategies.

The first is the naive full index update. For computing its overhead, they assume that performing a forward pass on a document with a model of p parameters takes $O(p)$ time. With k as the number of retrieved documents, P_{LM} the number of parameters in the language model, and b as the batch size,

the complexity for each training step is $4 \times b \times k \times P_{\text{LM}}$. (The scalar 4 accounts for the backward pass and activation checkpointing.) With n as the number of retrieval documents and P_{Ret} as the number of parameters in the retriever, the complexity for full index computing is $n \times P_{\text{Ret}}$. Then, the total overhead for updating the index every t training steps is:

$$\frac{n \times P_{\text{Ret}}}{4 \times b \times k \times P_{\text{LM}} \times t}. \quad (3.13)$$

The authors give an example: if the index contains 37M documents (the size of Wikipedia), and the model trains with a batch size of 64, retrieves 20 documents, and updates the index every 1000 steps, the overhead accounts for 30% of the total computational cost.

The second is re-ranking. Here, the retriever fetches a larger number l of documents, re-embeds and re-ranks these documents, and hands the language model the top- k . The complexity of re-ranking these l documents is $b \times l \times P_{\text{Ret}}$. The total overhead for performing it every training step is:

$$\frac{l \times P_{\text{Ret}}}{4 \times k \times P_{\text{LM}}}. \quad (3.14)$$

With the same example as before, and assuming that $l = 10k$, the overhead would account for only 10%.

The authors then conduct an ablation study to compare the aforementioned technical choices. For the experimental setup, they used the Contriever model (Izacard et al., 2022a) as the retriever, and T5 as the language model. For pre-training and retrieval data, they resorted to Wikipedia and the Common Crawl¹. They pre-trained their model for 10,000 iterations with the Adam optimizer with a batch size of 128. They concluded that using Perplexity Distillation as loss function (with an MLM task) and re-ranking as document index updating strategy produced the best results. More specifically, the retriever fetches 20 documents, updates the index every 2,500 steps, and re-ranks the top-100 documents.

Finally, the authors evaluated this experimental setup on NQ. It scored 60.4 points, a whopping 15-point increase over RETRO.

3.3 Dialogue Summarization

Most research on abstractive text summarization has focused on single-speaker documents, where the information flow is clear and structured throughout sentences and paragraphs. In sharp contrast, there is dialogue, which is informal, repetitive, drenched with hesitations and topic drifts, and can involve many participants and interaction signals. Furthermore, information is jumbled all over the conversation, often with no clear thread to unravel it. Suffice it to say that dialogue summarization calls for new approaches.

¹<http://commoncrawl.org>

R1/R2/RL	AMI	R1/R2/RL	SAMSum
Sentence-Gated	49.29/19.31/24.82	TGDGA	43.11/19.15/40.49
DDAMS	51.42/20.99/24.89	S-BART	50.70/25.50/48.08
BART	51.77/18.83/49.67	BART	52.98/27.67/49.06
DialogLM	54.49/20.03/51.92	MV-BART	54.05/28.56/50.57

Table 3.3: Performance for meeting summarization (left) and chat summarization (right) models.

In this section I peruse the literature of this field with the help of the survey by [Feng et al. \(2021\)](#). I will also cover in detail two models that are relevant for this report.

Previous works on dialogue summarization are usually pigeonholed according to the domains of dialogue they address. [Feng et al. \(2021\)](#) identifies five domains: meeting, chat, email thread, customer service, and medical dialogues. I will focus on the two most relevant for this report: meeting and chat.

Meeting summarization is helpful for non-participants to capture the highlights of a meeting. The benchmark dataset for this is AMI ([Carletta et al., 2006](#)), which is a small-scale meeting corpora comprising a few hundred dialogues and corresponding extractive summaries. The crude approach is to apply off-the-shelf pre-trained neural language models. However, and despite having powerful modeling abilities ([Zhu et al., 2020](#)), these models pre-train on rough text data with across-the-NLP-board objectives and ignore the many idiosyncrasies of dialogue. Some authors acknowledged this and developed tools to deal with it. [Goo and Chen \(2018\)](#) introduced Sentence-Gated, which structures a conversation with dialogue acts, [Feng et al. \(2020\)](#) presented DDAMS, which features dialogue discourses, and [Koay et al. \(2020\)](#) publicly released domain terminology. Nonetheless, meetings can be lengthy, which can be a problem for Transformer-based methods because of their input-length restrictions. (BART, for example, limits its input to 1024 tokens.) In the following section, I will review DialogLM ([Zhong et al., 2022](#)), which is SOTA for handling long meeting transcripts.

Chat applications are now mainstream vessels for communication. This has led to gigantic amounts of chat messages, which can pose a challenge for the new chat participant. Because of this, chat summarization is now a vital endeavor. [Gliwa et al. \(2019\)](#) sparked research in this field by presenting the first high-quality corpus, SAMSum, which contains over 16k chat dialogues with manually annotated summaries. Many of the current works focus on modeling interactions in a dialogue. To do this, graph modeling strategies are commonplace. [Zhao et al. \(2020\)](#) introduces TGDGA, which uses topic words as bridges between dialogue turns to build a guided dialogue graph; [Chen and Yang \(2021\)](#) present S-BART, which models inter-utterance structure and intra-utterance action. For this report, a particularly interesting model for capturing rich conversation structures is MV-BART ([Chen and Yang, 2020](#)), which I will cover in detail after DialogLM.

Table 3.3 contains the performance of all models discussed in this section.

3.3.1 DialogLM

The aforementioned meeting-summarization models are skilled at tackling the many idiosyncrasies of dialogue. Nonetheless, they are mostly limited to short conversations (less than 200 words) and cannot handle long texts (more than 5,000 words). This is not a new problem, and previous research on it has achieved remarkable success (Beltagy et al., 2020; Kitaev et al., 2020; Wang et al., 2020). However, the resulting models are not designed for dialogue. To bridge this gap, Zhong et al. (2022) present DialogLM.

This model features a window-based denoising pre-training task on a large dialogue corpus which selects a window containing several dialogue utterances, injects dialogue-related noise into it, and trains with the objective of decorruping this window. This technique combines the strengths of BART and PEGASUS; it allows for corrupting windows longer than PEGASUS's one-sentence window, which is important to recognize the format of dialogue, and it requires less computational resources than BART's full-text denoising.

The authors design five types of dialogue-based noise that corrupt not only text but also the ordering of dialogue turns, which in turn challenges DialogLM to completely understand the intricacies of dialogue. *Speaker masking* consists of replacing 50% of the speaker names with a [MASK_SPEAKER] token. *Turn splitting* selects the turn with the most sentences, splits it, keeps the speaker name of the first split, and masks the names in all the other splits. *Turn merging* merges multiple consecutive turns into one turn, which is then assigned to the speaker of the first turn. The other two are *text infilling* and *turn permutation*, which are similar to BART's.

For the model architecture, the authors use a standard decoder and leverage a hybrid-attention approach based on the Transformer for the encoder. The latter combines layers with sparse Sinkhorn attention (Tay et al., 2020), which reduce the computational burden but also lose global information, and layers with global self-attention, which keep the full dialogue semantics intact.

As experimental setup, they initialize DialogLM with UniLMv2 (Bao et al., 2020) and keep the fourth, eight, and twelfth encoder layers with global self-attention. This model trains on the MediaSum (Zhu et al., 2021) and OpenSubtitles (Lison and Tiedemann, 2016) datasets for 200,000 steps, including 20,000 for warm-up, with batch size at 64, maximum learning rate at $2e^{-5}$, and window size at 10% of the input length and limited to 512 tokens. For noise, they use, in this order, speaker masking, turn splitting or turn merging, text infilling for 15% of the tokens, and turn permutation.

They evaluated DialogLM on the AMI dataset and beat all the models mentioned in the previous section (see Table 3.3).

3.3.2 MV-BART

MV-BART rests on the assumption that a conversation can be perceived through many views, and that each view supplies the language model a unique aspect of the conversation. Besides the much-studied generic views — *turn view*, where each participant utterance is one dialogue unity; and *global view*, which perceives the whole conversation as the only dialogue unity — they propose and enumerate two new views. *Topic view* comprises the topic flow of a conversation. For example, a telephone call could feature a “greetings → invitation → party details → confirmation” pattern from a topical perspective. The second view is *stage view*, and it supplies functions or goals for the different topics. A stage view of the previous example would be “introduction → problem exploration → problem solving → wrap-up.” Formally, the authors define a conversation view as:

$$\begin{aligned} \mathbf{C}_k &= (\mathbf{b}_1^k, \dots, \mathbf{b}_n^k), \\ \mathbf{b}_j^k &= (x_{1,j}^k, \dots, x_{m,j}^k), \end{aligned} \quad (3.15)$$

where \mathbf{b}_j^k are the (vector) blocks comprising a view k , and $x_{i,j}^k$ are the tokens comprising block j . The authors use the C99 (Choi, 2000) algorithm to find $\mathbf{C}_{\text{topic}}$, and extract stages through a Hidden Markov Model (HMM) to build \mathbf{C}_{view} .

MV-BART extends standard pre-trained seq2seq models with the ability to encode and combine several conversation views. It consists of BART-initialized conversation encoder and multi-view decoder. The encoder encodes each block \mathbf{b}_j^k into a hidden representation \mathbf{h}_j^k with tokens $h_{i,j}^k$.

A view k , or \mathbf{V}_k , encompasses all blocks through LSTM layers (Hochreiter and Schmidhuber, 1997) like so:

$$\begin{aligned} \mathbf{S}_j^k &= \text{LSTM}(\mathbf{h}_j^k, \mathbf{S}_{j-1}^k), j \in \{1, \dots, n\}, \\ \mathbf{V}_k &= \mathbf{S}_n^k. \end{aligned} \quad (3.16)$$

The multi-view decoder determines how much attention each view deserves, combines them, and produces a concise summary. It differs from a canonical Transformer decoder by introducing a multi-view attention layer. This layer decides the importance α_k of a view \mathbf{V}_k :

$$\begin{aligned} \alpha_k &= \frac{\exp(\mathbf{u}_k^\top \mathbf{v})}{\sum_i \exp(\mathbf{u}_i^\top \mathbf{v})}, \\ \mathbf{u}_k &= \tanh(\mathbf{W}\mathbf{V}_k + \mathbf{b}), \end{aligned} \quad (3.17)$$

where \mathbf{v} is a context vector, \mathbf{W} is a parameter matrix, and \mathbf{b} is a parameter vector.

Then, it computes multi-head attention for each view \mathbf{V}_k , weighs them with α_k , and sums them for forward passing:

$$\mathbf{A} = \sum_k \alpha_k \text{Attention}(\mathbf{V}_k). \quad (3.18)$$

For the experimental setup, the authors embedded the input tokens with Sentence-BERT (Reimers and Gurevych, 2019), set window size at 4 and standard coefficient at 1 for the C99 algorithm, the number of hidden states at 4 for the HMM, beam width at 4, and the learning rates for encoder and decoder at $3e^{-3}$.

For evaluation, they compared MV-BART with different view combinations on the SAMSum dataset. A mix of topic and stage views yielded the best result (see Table 3.3).

3.4 Summary

In this chapter, we reviewed important literature on the topics of single-document summarization and dialogue summarization, including the strengths and pitfalls of both areas. We also touched upon RAG techniques and implementations, which help to enlighten the way it will be used in the proposed dialogue-summarization approach.

4

Retrieval-augmented Dialogue Summarization

Contents

4.1 Component Overview	39
4.2 Augmenting a Dialogue Summarization Dataset	42
4.3 Retriever Fine-tuning	44
4.4 Retriever Inference	45
4.5 Summary	46

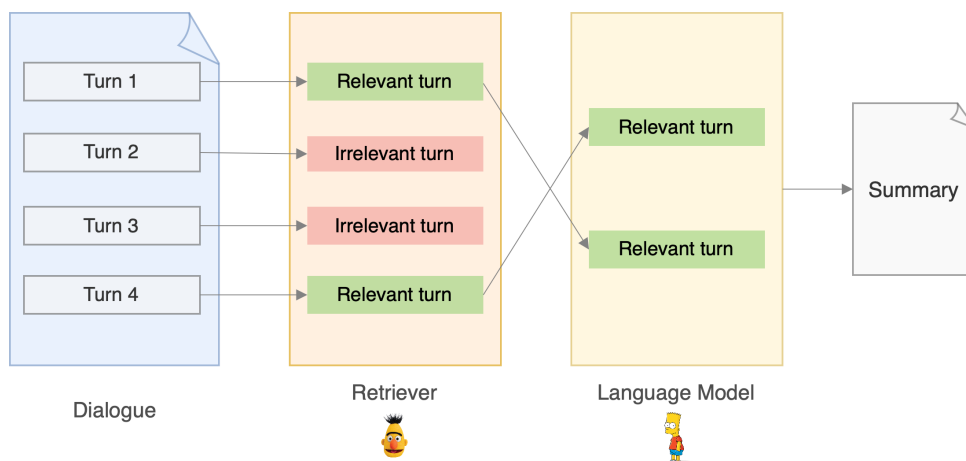


Figure 4.1: Component overview. The retriever classifies the turns from a dialogue as relevant or irrelevant, re-ranks them, and feeds them to a language model for summarization.

Most abstractive summarization models produce summaries based on the entire input text. This is true of all the models I have described in Chapter 3. Even retrieval-augmented models follow this; the only major difference is the addition of a retriever to fetch data from external databases. However, this approach generates and ignores major problems.

First, these models have a hard time with long inputs because of Transformer input-length restrictions. Second, topic drifts in the conversation motivate wrong summaries, and there is no mechanism to tackle them. These problems hint at a solution: the model should not take the whole text in. It needs to retrieve only the relevant chunks of text. This allows for long dialogues (the input length can be massively reduced), topic-drift management (unimportant chunks can and should be ignored). I propose an approach that incorporates this *retrieve-then-summarize* pipeline.

This chapter describes the aforementioned approach. Section 4.1 gives an overview of the components used, Section 4.2 describes the steps taken in augmenting an existing dialogue-summarization dataset, Section 4.3 and Section 4.4 explain the implementation of the retriever, and Section 4.5 presents a summary of this chapter.

4.1 Component Overview

The proposed architecture consists of a language model coupled with a retriever (see Figure 4.1). The retriever selects the relevant text from a dialogue and passes only those to the language model for generating a summary. This way we simplify the pipeline and define clear steps that can be improved. We will now look at both the retriever and language model in more detail.

4.1.1 Retriever

Before we proceed, we need to decide on the granularity of chunk of text to retrieve. We could choose to retrieve words, sentences, speaker turns, or blocks of text. For simplicity, I settled on speaker turns, but it definitely is a decision that can and should be reviewed in future work. Let us define a *turn* as a unit of conversation where one participant speaks and another listens containing one or more sentences. A turn can be *relevant* if it contains information essential to the summary or *irrelevant* if otherwise. Now we can zoom in on the retriever.

Ideally, the retriever fetches all relevant turns from a dialogue and only those. This immediately raises questions: how does the retriever compare turns? How many should it retrieve? Does the order of turns matter? The last two questions I will deal with in Chapter 5. Regarding the first question, a straightforward way of selecting turns is by comparing each to the entire dialogue, ranking their similarity, and selecting a few from the top of the list. For retrieval tasks, the similarity metric of choice is cosine similarity.

Cosine similarity determines how near two vectors are in a multidimensional space by computing the cosine of the angle between them. In our case, the vectors are the dialogue and the speaker turns. Before computing an angle, it is common to normalize the vectors to unit length so that only their directions are taken into account. The cosine similarity score ranges from -1 to 1 , with a value near 1 suggesting high similarity and a value near -1 suggesting dissimilarity.

Now we need to decide on the retrieval paradigm to use. The major options are sparse retrieval, dense retrieval, and use of heuristics. In sparse retrieval, text is represented by sparse vectors where each dimension corresponds to a term in the vocabulary. Documents and queries are matched based on how well their respective vectors align. Common techniques within this paradigm are TF-IDF and BM25, which rely on term frequency and inverse document frequency measures to assess document relevance to a query. In dense retrieval, we use Transformers to create dense vectors, i.e., embeddings, which, unlike sparse vectors, pack rich semantic information. For both cases, matching is often performed with cosine similarity. Then, we can use heuristics such as: given a dialogue turn, rank the remaining turns by matching objects of reference; choose a percentage of turns from all speakers; choose the longest turns; discard questions; etc. Preliminary experiments showed that sparse retrieval far outperforms using heuristics and that dense retrieval outperforms sparse retrieval. For this reason, I chose the dense retrieval paradigm for the retriever, and to decide which Transformer model to use for creating the word embeddings I resorted to the Massive Text Embeddings Benchmark (MTEB) ([Muennighoff et al., 2023](#)) leaderboard.

MTEB covers multiple embedding tasks on a total of 58 datasets. One of this tasks, and particularly relevant for our purposes, is retrieval. Datasets designed for retrieval contain a corpus, queries, and a mapping for each query to relevant documents from the corpus. The goal of the retrieval task is to

fetch these documents. Retrieval models embed both queries and documents from the corpus and compute cosine similarity scores. After ranking the documents for each query based on these scores, an evaluation metric like normalized Discounted Cumulative Gain (nDCG) ranks the document retrieval order.

A standard model for retrieval tasks in this leaderboard that combines score with speed is *all-mpnet-base-v2*¹. It uses the pre-trained MPNet model (Song et al., 2020) and fine-tunes it on a 1B sentence pairs dataset with a contrastive learning objective: given a sentence from a pair, the model should predict which out of a set of randomly sampled sentences was actually paired with it.

This text embedder uses the MPNet architecture instead of other alternatives for the following reason. BERT uses MLM for pre-training, which leverages bidirectional context of masked tokens efficiently, but it assumes that masked tokens are independent from each other and, therefore, ignores their interdependency. XLNet (Yang et al., 2019) solves this problem by introducing Permuted Language Model (PLM), which factorizes the predicted tokens with the product rule in any permuted order and can, therefore, better model the interdependency of masked tokens. Nevertheless, it presents a new drawback. Since a model can see the entire input sentence when fine-tuning for downstream tasks, it should also see as much as possible of the input sentences when pre-training, and PLM does not allow this; each predicted token can see the preceding tokens in a permuted sentence, but it does not know the position information of the full sentence.

MPNet combines the advantages of BERT and XLNet while avoiding their drawbacks. Its authors propose a unified view for MLM and PLM that rearranges and splits the input tokens into non-masked and masked parts. The non-masked tokens are put on the left, while the masked (to-be-predicted) tokens are put on the right, as shown in Figure 4.2.

I used this model as a baseline retriever and as the foundation for fine-tuned retrievers (see Section 4.3).

4.1.2 Language Model

The language model takes in the retrieved dialogue turns and generates a summary. It is far more crucial than the retriever since it is this component that performs the actual summarization. However, since the focus of this dissertation is on the retriever component, the choice for the language model was made mostly based on proven success in literature, online availability, ease of use, and computational efficiency. A clear contender that checks all these boxes is BART.

BART is readily available in popular NLP libraries like Hugging Face Transformers in many pre-trained and fine-tuned options. For this dissertation, I used a version called Meeting Summary², which is the original pre-trained BART fine-tuned on one abstractive single-document summarization dataset called

¹<https://huggingface.co/sentence-transformers/all-mpnet-base-v2>

²https://huggingface.co/knkarthick/MEETING_SUMMARY

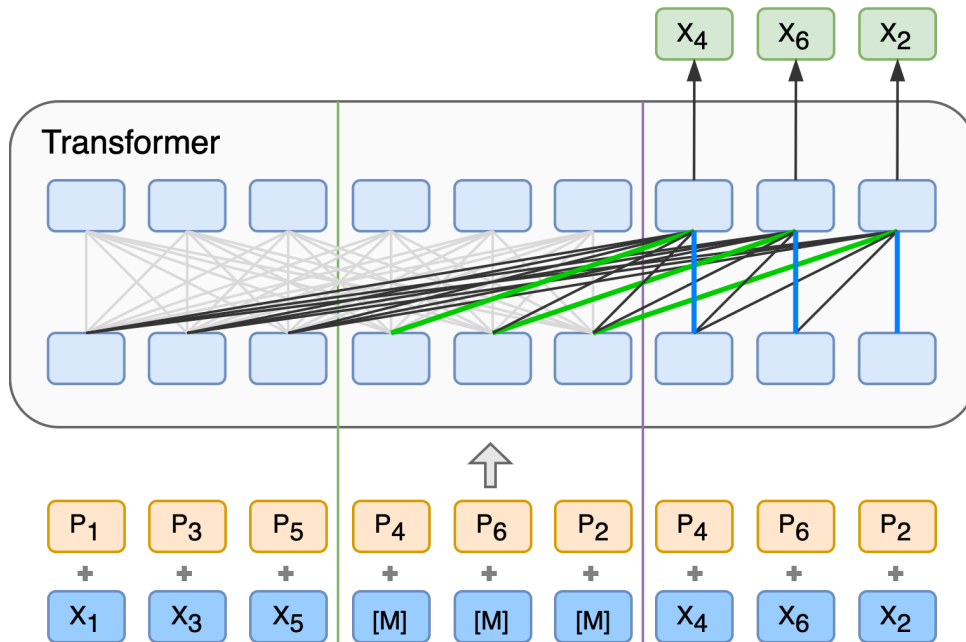


Figure 4.2: The MPNet architecture. The light grey lines represent the bidirectional self-attention in the non-masked part. The blue, green, and black lines represent self-attention in the masked part.

XSum and on three dialogue-specific summarization datasets: AMI, SAMSum, and DialogSum. For reproducibility, when generating a summary, the stock settings were preserved and a simple “summarize:” prompt was given to the model.

There are also other available options which check some of these boxes and provide much better results than BART. Namely, Large Language Model (LLM)s such as GPT4 (OpenAI et al., 2023), LLaMa (Touvron et al., 2023), FlanT5 (Chung et al., 2022), etc. Nonetheless, preliminary experiments showed that despite giving much better results than BART (sometimes more than ten ROUGE-1 points), these models were far too expensive with regards to the available computational resources. For this reason, I decided to keep on using BART-based models.

4.2 Augmenting a Dialogue Summarization Dataset

The next step is to consider the dataset used for fine-tuning and testing the retriever. Recall that we need instances that contain a dialogue, a summary, and the relevant dialogue turns. Any dialogue summarization dataset will contain the first two elements, but I have found none that also features the last element. Because of this we need to augment an existing dataset with it.

We could employ annotators to analyze each dialogue of a given dataset and decide which of the turns are relevant, but, unfortunately, this is a costly endeavor. However, we now have powerful LLMs at our disposal that can do this.

For ease of use and reproducibility, I used OpenAI's GPT4-Turbo (OpenAI et al., 2023) as the “annotator” to select these relevant turns. The temperature was set at 0.0 since the response should be based strictly on known facts (the input) without any speculative or random content. This is the prompt used:

```
1 Instruction: Below are a Dialogue and its respective Summary. I want
  you to look at both and extract from the Dialogue the turns that
  are immediately and really relevant for the generation of the
  Summary (i.e., the sentences without which a language model could
  not have generated the Summary). Put the extracted turns inside
  the brackets in 'Relevant Turns: {}' separated by a comma. Please
  keep the name of the person attached to the turn, like this '
  Relevant Turns: {'Ms. Dawson: Yes, sir...', 'Ms. Dawson: Yes, sir.
  Go ahead.'}'. The extracted turns, in their totality, must
  contain all the person names present in the Summary and, also in
  their totality, be under 1024 tokens.
2 Dialogue: {dialogue}
3 Summary: {summary}
4 Relevant Turns: {}
```

The addition of a command to keep the totality of the extracted turns under 1024 tokens is to avoid reaching BART's maximum number of input tokens, which, in turn, solves the aforementioned problem of long inputs.

On average, the resulting relevant turns for each dialogue compose 4.5 sentences, which is less than half of the average 9.8 sentences that the original dialogues contain. From this, and if the annotator can be trusted, we can infer that more than half of the original dialogues is not helpful for summarization. In the next chapter we verify if this is indeed true.

With this approach I was able to augment a dialogue summarization dataset with turns that the LLM deemed relevant within each dialogue. If they are indeed relevant or not is, of course, not guaranteed, and future work involving human annotators could help in this regard. However, in principle, if the summary generated with only these turns is of a higher quality than the summary generated with the entire dialogue, we can deduce that the turns are indeed relevant; at least, more relevant than the entire dialogue.

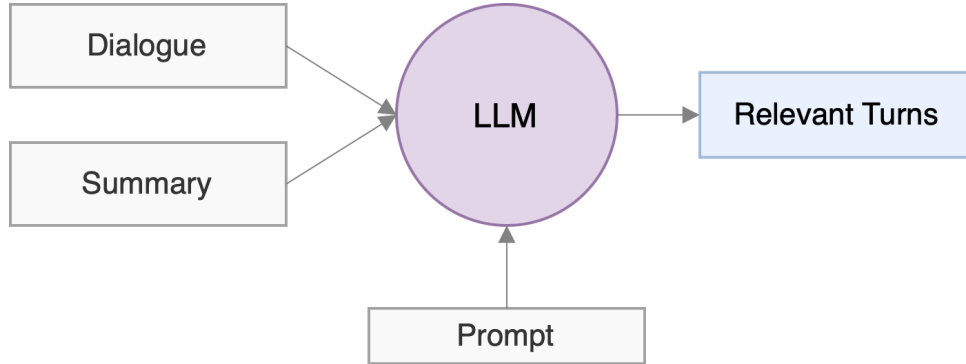


Figure 4.3: Dataset augmentation.

4.3 Retriever Fine-tuning

Now I will detail the steps I took in fine-tuning the pre-trained retriever. Namely, loss function selection, dataset preparation, hyperparameter tuning, and evaluation.

The goal of this fine-tuning is for the retriever to correctly classify the turns of a dialogue as relevant or irrelevant. For this, we need a loss function that clearly distinguishes relevant (positive) from irrelevant (negative) turns when juxtaposed with the dialogue in question. One function that does this particularly well is called Multiple Negatives Ranking (MNR) loss.

MNR achieves the aforementioned distinction by minimizing the distance between a query (a dialogue) and positive samples (relevant turns) and maximizing that of the query and multiple negative samples. Mathematically, this is achieved by computing

$$L = \sum_{i=1}^P \sum_{j=1}^N \max(0, s(q, p_i) - s(q, n_j) + m), \quad (4.1)$$

where P and N are the number of positive and negative samples, respectively, q is the query, p_i and n_j are positive and negative samples, respectively, s is the similarity between query and sample, and m is a margin for distinguishing positive from negative samples.

The loss function accepts as input a batch of triplets (q_i, p_i, n_i) , corresponding to a dialogue (q_i) , a positive sample to that dialogue (a relevant turn) (p_i) and a hard negative (an irrelevant turn) (n_i) . For a given dialogue-relevant turn pair, i.e., (q_i, p_i) , the loss considers all the relevant turns of the other pairs, i.e., p_j , $(j \neq i)$, and all the irrelevant turns, i.e., n_j , as negatives. It is important to note that each dialogue q_i has to be unique within a batch.

For this to work, we need to prepare the training dataset accordingly. With the newly generated augmented dataset (see previous subsection) and a batch size of 8 (due to limited computing power available), I shuffled the data so that each dialogue in every batch is unique, its positive pair is a relevant turn, and its hard negative an irrelevant one. There are as many batches as necessary for every dialogue

in the training set to be paired with every one of its relevant turns.

With regards to the hyperparameters, I followed those suggested in the BERT paper, since the retriever is a BERT-based model. This meant setting the learning rate at 3×10^{-5} , number of epochs at 3, and dropout probability at 0.1.

Finally, I did no intermediate evaluation, since what is ultimately important is the quality of the summary the language model generates with these retrieved turns. I leave a more comprehensive evaluation of the fine-tuning procedure for future work.

4.4 Retriever Inference

At inference time, the model embeds the entire dialogue and individual turns of the dialogue, computes cosine similarity for each *dialogue-individual turn* pair, and ranks the individual turns by descending score. Then, the goal is to select less than 100% of these dialogue turns to ensure that we mitigate topic shifts and long dialogue problems. There are two main ways of doing this: relevance score threshold and top-K.

With relevance score threshold, we only select the dialogue turns with score higher or equal to the threshold. This assures that (if the fine-tuning process was well done) we consistently get relevant turns. The drawback of this approach is that it may return no turns if all of them do not reach the relevance score threshold. This may happen if fine-tuning was sub-optimal.

Top-K selection avoids the previous drawback by ensuring that we always select a consistent number of dialogue turns. However, it is blind to the relevance score and may select low-score turns in order to complete K.

To avoid each of these drawbacks, the better approach would be to combine the two techniques: start by selecting dialogue turns based on the relevance score and, if none or too few are selected, use top-K.

Besides thinking about the selection strategy, let us focus on some more considerations to keep in mind at inference time. The first is the order in which we feed the turns to the language model. We can feed the turns based on relevance score or temporal order. Preliminary experiments showed that the latter works best since it captures the natural flow of the conversation and produces a more genuine conversation. Then, another consideration is whether we feed all the selected dialogue turns to the language model at same time or if we feed each one and summarize it independently. If we choose the latter, the resulting summary gets much larger than the ground-truth summary, which makes sense because the ground-truth summary has approximately the length of two dialogue turns. Because of this, I feed all the selected dialogue turns at the same time. Next we have to consider if these selected approaches mitigate what we set out to mitigate: topic shifts. Since we only selected the relevant dialogue turns, irrelevant topic shifts are discarded.

4.5 Summary

This chapter presented a new approach for retrieval-augmented dialogue summarization. It consists of combining a fine-tuned retriever that selects the relevant turns from a dialogue and feeds only those to a language model for summary generation. It reviewed the role of the retriever, how it works, how it is trained, and it also reviewed the role of the language model.

An augmentation of existing dialogue summarization datasets was also presented, which can be generally applied for further improving work in this area.

5

Evaluation

Contents

5.1 Experimental Setup	49
5.2 Experimental Results	50
5.3 Summary	53

The previous chapter introduced a new approach for dialogue summarization with RAG that aims to help with long dialogues and topic drifts. This chapter's goal is to put that approach to the test. Section 5.1 covers the experiment setup I did for testing and evaluating the model, including dataset and evaluation metrics selection. Section 5.2 details the main findings of the evaluation.

5.1 Experimental Setup

This section describes the experimental setup used to evaluate several implementations of the proposed approach. This includes the choice of the dataset and the evaluation metrics I used for training and testing the models.

5.1.1 Dataset

For developing and testing retrieval-augmented approaches, I chose the DialogSum dataset. It is a large-scale summarization dataset containing around 15,000 multiple-domain, daily-life dialogues and their respective summaries.

For long dialogue summarization, it still is the only viable choice. There are many other abstractive summarization datasets, but, pertaining to multiple-domain dialogue, there are few. SAMSum is one that comes close. However, it consists of online chat messages instead of natural, spoken dialogue and its dialogues are 30% shorter than those of DialogSum. Other alternatives include MediaSum and TODSum, which include interviews (60 Minutes-style) and task-oriented (think restaurant, taxi, and hotel dialogue exchanges) dialogues, respectively. For an example of a DialogSum dialogue, check Chapter 1.

Furthermore, I resorted to three versions of the DialogSum dataset:

- **Original:** this is the unchanged, original version of DialogSum. Turns begin with “Person#X”.
- **Correct Names:** this version corrects the names of the speakers in each turn of the dialogues. It was introduced by [Wang et al. \(2023\)](#). Check Figure 5.1 for an example.
- **Augmented:** this version, which is built on the previous iteration, is the augmented dataset that was introduced in Section 4.2.

5.1.2 Evaluation Metrics

As mentioned before, the only evaluation performed was that of the generated summaries against the reference summaries. For this, the usual evaluation metric is ROUGE score. ROUGE corresponds to the F_1 scores for word overlap (ROUGE-1), bigram overlap (ROUGE-2), and longest matched sequence between reference summary and generated summary (ROUGE-L). The F_1 score combines Precision (P) and Recall (R) scores like so:

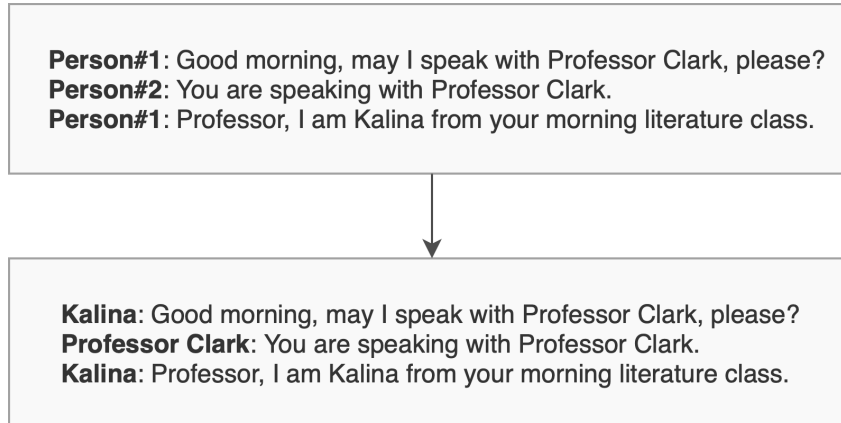


Figure 5.1: Excerpt of a dialogue with the original names (above) and the corrected names (below).

$$\begin{aligned}
 P &= \frac{TP}{TP + FP}, \\
 R &= \frac{TP}{TP + FN}, \\
 F_1 &= 2 \times \frac{P \times R}{P + R},
 \end{aligned}
 \tag{5.1}$$

where TP, FP, FN are the number of true positives, false positives, and false negatives, respectively, between candidate and reference summaries' tokens. Precision gives the proportion of n -grams (contiguous sequences of n words) in the generated summary that appear in the reference summary, while Recall measures the reverse, i.e., the proportion of n -grams in the reference summary that are also in the generated summary.

5.2 Experimental Results

This section details the experimental results obtained from the prior experimental setup. This includes an analysis of the augmented dataset, the answer to the research question introduced in Chapter 1, and the analysis of the proposed approach.

5.2.1 Augmented Dataset Analysis

Before diving in to the results of the new architecture, let us evaluate the summaries generated from the three versions of the DialogSum test set. The reference summaries (ground truth) are exactly the same in all three versions; what changes is only the text we feed into the language model. In 'Original' and 'Corrected Names', we feed the entire dialogue. In 'Augmented', we feed the relevant turns instead of the entire dialogue. Table 5.1 presents the ROUGE scores for the generated summaries. The same language model (introduced in Subsection 4.1.2) was used in the three runs. It is also important to

remember that no retrieval was used for this evaluation.

Dataset	R_1	R_2	R_L
Original	37.14	14.32	29.30
Corrected Names	39.19	15.08	30.60
Augmented	41.83	16.85	33.74

Table 5.1: ROUGE scores of the generated summaries for the different DialogSum versions.

With 'Corrected Names', we can check that just by fixing the speaker names in the dialogues, the generated summaries score one or two points higher. This was the reason for also including the names of the persons when generating the 'Augmented' dataset. More importantly, there is a considerable increase in 'Augmented', which confirms our research question: *summarizing only the relevant turns instead of the entire dialogue does indeed yield better results.*

Nevertheless, this Augmented dataset may be faulty. Ideally, the relevant turns should contain (1) all the names of the persons involved in the dialogue, (2) all the information in the summary, (3) only the information in the summary, and (4) the information rightly ordered. The hard part is how we can test if these relevant turns are ideal.

For the first point, we can simply check the percentage of Relevant Turns entries that contain all the speaker names. Upon verification, this comes at 82%, which means it is not perfect. For points (2) and (3) we can check Recall and Precision, respectively, between Relevant Turns and Summary and compare with the Recall and Precision between Dialogue and Summary. Table 5.2 contains these scores. We can check that, indeed, Precision is higher for relevant turns when compared to the original dialogue. This is expected, since, in Relevant Turns, we are focusing only on the information present in the summary. Nonetheless, Recall should have matched that of the Dialogue, and, since it does not, we can conclude that the information in Relevant Turns is, on average, not complete.

Text	Precision@1	Recall@1
Dialogue	11.14%	50.40%
Relevant Turns	17.76%	42.53%

Table 5.2: Average Precision and Recall scores at word level for Dialogue and Relevant Turns entries when compared to Summary in the Augmented dataset.

The last point, though, is harder; we could use a metric like nDCG to evaluate information order. nDCG considers the relevance of turns and their positions, reflecting the idea that highly relevant turns should appear higher up. nDCG calculates a normalized score by dividing the cumulative gain of the ranked turns by the ideal cumulative gain. A higher nDCG value, ranging from 0 to 1, indicates better ranking quality, with 1 representing perfect ranking alignment with relevance. Nonetheless, and despite

the adequacy of the metric, this analysis would involve non-trivial changes to the existing turns and summaries. I leave this fourth point for future work.

Overall, we do see considerable improvement with using the Augmented dataset when compared to the Original (more than three ROUGE points). However, upon examination, we also conclude that it is not optimal: not all speaker names are present in the relevant turns and not all of the summary information may be found there. For our purposes, this dataset augmentation suffices, but I leave the creation of a better one for future work, which could yield even better results.

5.2.2 New Architecture Results

Now the question we have to settle is, 'Can we train a retriever to emulate fetching the relevant turns of the Augmented dataset?' The 'Corrected Names' dataset performance now becomes our Baseline and the 'Augmented' dataset becomes the Target performance. If the retriever trained only on the Augmented train set, and if the training was optimal, its performance, in principle, would be equal to that of the pure Augmented test set.

Before moving onto the results of the new architecture, let us address the results of the same architecture but without the language model, i.e., without the summary generation. This allows us to check how the retrieved turns function as summary. Table 5.3 shows these results.

Model	DialogSum		
	R_1	R_2	R_L
Original Dialogue	19.32	6.08	14.72
Original MPNet Retriever			
- top 25%	26.88	7.61	20.86
- top 50%	24.73	7.31	18.37
- top 75%	21.72	6.65	16.27
Fine-tuned MPNet Retriever			
- top 25%	26.82	7.62	20.88
- top 50%	24.43	7.34	18.28
- top 75%	21.72	6.65	16.27
GPT4 Retriever	<u>30.73</u>	<u>10.57</u>	<u>23.95</u>

Table 5.3: ROUGE scores of the retrieved turns from the different retrievers.

The results indicate that the retrievers alone can extract relevant parts of the dialogues that are themselves better summaries (i.e., ROUGE scores) than the original dialogues are as summaries. The GPT4 retriever achieves the highest scores among the retrievers, suggesting its potential effectiveness in identifying relevant turns.

Next, we evaluate the complete architecture with the language model generating summaries based on the retrieved turns. Table 5.4 presents the ROUGE scores for the different model variations.

Model	DialogSum		
	R_1	R_2	R_L
Baseline	39.19	15.08	30.60
Baseline + original MPNet			
- top 25%	31.41	10.61	25.85
- top 50%	37.28	13.77	29.74
- top 75%	39.02	14.62	30.84
Baseline + fine-tuned MPNet			
- top 25%	32.47	11.21	26.88
- top 50%	38.06	14.34	30.29
- top 75%	39.29	15.06	31.19
Target	<u>41.83</u>	<u>16.85</u>	<u>33.74</u>

Table 5.4: ROUGE scores of the generated summaries for the different DialogSum versions.

The results show that the integration of the language model with the retrievers significantly improves the ROUGE scores. Fine-tuning the MPNet retriever achieves the best results among the retrievers, especially when selecting the top 75% turns, which surpasses the baseline in some metrics. However, the Target performance remains higher, indicating room for further improvement in the retriever’s effectiveness.

While the architecture with fine-tuned MPNet retriever shows promising results, it is evident that further optimization and refinement can push the performance closer to the Target.

5.3 Summary

This chapter covered the experimental setup and results for the proposed dialogue-summarization approach and a thorough analysis of the augmented dataset DialogSum, which is presented as DialogSum Augmented.

With regards to the proposed approach, we confirmed that fine-tuning a retriever to select the relevant turns out of a dialogue does indeed produce better results, even though there is large room for improvement in future work.

More importantly, the analysis of DialogSum Augmented confirms the validity of our research question: summarizing only selected parts of the dialogue rather than the whole dialogue produces a better summary, i.e., one with higher ROUGE scores.

6

Conclusion and Future Work

Contents

6.1 Conclusion	57
6.2 Future Work	57

This chapter concludes the dissertation. It offers a general view of the work presented, the principal conclusions reached from the experimental results, and some routes that future work might take in order to further improve on the proposed approach.

6.1 Conclusion

This thesis presented a retrieval-augmented approach for dialogue summarization. Instead of feeding the entire dialogue to a language model for summarization, it consists of first selecting the relevant passages from it and feeding only those to the model. This is designed to mitigate the problems that appear when applying single-document summarization methods on dialogue; namely, topic shifts and long input. Practically, this mitigation would translate into higher ROUGE scores. Thus, the research question this thesis set out to answer can be put as, *will summary quality increase if we summarize only the important passages of a dialogue instead of its entirety?* The answer comes from the experimental results as *yes*; summarizing only the relevant parts of a dialogue instead of the whole dialogue results in a better summary, i.e., higher ROUGE score.

Another contribution of this dissertation is the introduction of the augmented version of the DialogSum dataset, DialogSum Augmented¹, which features not only dialogue and summary for each instance, but also the most relevant turns in the dialogue. This contribution aims to further encourage work in this area and bring about new approaches for dialogue summarization.

6.2 Future Work

Despite this work answering its research question, there is room for improvement in the area of retrieval-augmented dialogue summarization. In particular, each of the steps in the proposed pipeline – dataset augmentation, retrieval, summary generation – can be improved. I will now detail some possible paths for each.

When augmenting a dialogue summarization dataset, the most cost-efficient (and surprisingly good) tool is an LLM. Nonetheless, there are many other ways to augment the dataset besides the one this thesis presented. Instead of asking the LLM to return only the relevant turns of the dialogues, we could ask it to rank the turns, reorder the turns, or give a score to each, for example. This would enable other methods for the retriever fine-tuning and enable the use of nDCG as the evaluation metric for it.

For retriever training, instead of fine-tuning the entire model we could fine-tune only selected layers of it or choose an entirely different training paradigm like choosing to use Adapter layers (Pfeiffer et al., 2020) or LoRA (Hu et al., 2021) techniques. Then, there remains a plethora of different loss functions to

¹<https://huggingface.co/datasets/diogofouto/dialogsum-augmented>

choose from and numerous hyperparameters to tinker with. With regards to retriever inference, there is vast room for testing new retrieval strategies and for thoroughly evaluating the presented one – top k%.

Finally, in the summary generation step, there are many more models to choose from apart from BART. Namely, we could leverage the power of LLMs to further improve results. And, as with the retriever inference, we can tinker with the hyperparameters to maximize performance.

Bibliography

- Bao, H., Dong, L., Wei, F., Wang, W., Yang, N., Liu, X., Wang, Y., Gao, J., Piao, S., Zhou, M., et al. (2020). Unilmv2: Pseudo-masked language models for unified language model pre-training. In *International Conference on Machine Learning*, pages 642–652. PMLR.
- Beltagy, I., Peters, M. E., and Cohan, A. (2020). Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.
- Borgeaud, S., Mensch, A., Hoffmann, J., Cai, T., Rutherford, E., Millican, K., Driessche, G. V. D., Lespiau, J., Damoc, B., Clark, A., Casas, D. D. L., Guy, A., Menick, J., Ring, R., Hennigan, T., Huang, S., Maggiore, L., Jones, C., Cassirer, A., Brock, A., Paganini, M., Irving, G., Vinyals, O., Osindero, S., Simonyan, K., Rae, J. W., Elsen, E., and Sifre, L. (2021). Improving language models by retrieving from trillions of tokens. *ICML*.
- Carletta, J., Ashby, S., Bourban, S., Flynn, M., Guillemot, M., Hain, T., Kadlec, J., Karaiskos, V., Kraaij, W., Kronenthal, M., et al. (2006). The ami meeting corpus: A pre-announcement. In *International workshop on machine learning for multimodal interaction*, pages 28–39. Springer.
- Chen, D., Fisch, A., Weston, J., and Bordes, A. (2017). Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*.
- Chen, J. and Yang, D. (2020). Multi-view sequence-to-sequence models with conversational structure for abstractive dialogue summarization. *arXiv preprint arXiv:2010.01672*.
- Chen, J. and Yang, D. (2021). Structure-aware abstractive conversation summarization via discourse and action graphs. *arXiv preprint arXiv:2104.08400*.
- Choi, F. Y. (2000). Advances in domain independent linear text segmentation. *arXiv preprint cs/0003083*.
- Chung, H. W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, Y., Wang, X., Dehghani, M., Brahma, S., Webson, A., Gu, S. S., Dai, Z., Suzgun, M., Chen, X., Chowdhery, A., Castro-Ros, A., Pellat, M., Robinson, K., Valter, D., Narang, S., Mishra, G., Yu, A., Zhao, V., Huang, Y., Dai, A., Yu, H., Petrov,

- S., Chi, E. H., Dean, J., Devlin, J., Roberts, A., Zhou, D., Le, Q. V., and Wei, J. (2022). Scaling instruction-finetuned language models.
- Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
- Dinan, E., Roller, S., Shuster, K., Fan, A., Auli, M., and Weston, J. (2018). Wizard of wikipedia: Knowledge-powered conversational agents. *arXiv preprint arXiv:1811.01241*.
- Feng, X., Feng, X., and Qin, B. (2021). A survey on dialogue summarization: Recent advances and new frontiers. *arXiv preprint arXiv:2107.03175*.
- Feng, X., Feng, X., Qin, B., and Geng, X. (2020). Dialogue discourse-aware graph model and data augmentation for meeting summarization. *arXiv preprint arXiv:2012.03502*.
- Gliwa, B., Mochol, I., Biesek, M., and Wawer, A. (2019). Samsun corpus: A human-annotated dialogue dataset for abstractive summarization. *arXiv preprint arXiv:1911.12237*.
- Goo, C.-W. and Chen, Y.-N. (2018). Abstractive dialogue summarization with sentence-gated modeling optimized by dialogue acts. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 735–742. IEEE.
- Graff, D. and Cieri, C. (2003). English gigaword 2003. *Linguistic Data Consortium, Philadelphia*.
- Hermann, K. M., Kocisky, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., and Blunsom, P. (2015). Teaching machines to read and comprehend. *Advances in neural information processing systems*, 28.
- Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.
- Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., and Gelly, S. (2019). Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR.
- Howard, J. and Ruder, S. (2018). Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.
- Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. (2021). Lora: Low-rank adaptation of large language models.
- Izacard, G., Caron, M., Hosseini, L., Riedel, S., Bojanowski, P., Joulin, A., and Grave, E. (2022a). Unsupervised dense information retrieval with contrastive learning. *Transactions on Machine Learning Research*.

- Izacard, G., Lewis, P., Lomeli, M., Hosseini, L., Petroni, F., Schick, T., Dwivedi-Yu, J., Joulin, A., Riedel, S., and Grave, E. (2022b). Few-shot learning with retrieval augmented language models. *arXiv preprint arXiv:2208.03299*.
- Jones, K. S. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*.
- Joshi, M., Chen, D., Liu, Y., Weld, D. S., Zettlemoyer, L., and Levy, O. (2020). Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.
- Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., and Yih, W.-t. (2020). Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.
- Kim, B., Kim, H., and Kim, G. (2018). Abstractive summarization of reddit posts with multi-level memory networks. *arXiv preprint arXiv:1811.00783*.
- Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- Kitaev, N., Kaiser, Ł., and Levskaya, A. (2020). Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*.
- Koay, J. J., Roustai, A., Dai, X., Burns, D., Kerrigan, A., and Liu, F. (2020). How domain terminology affects meeting summarization performance. *arXiv preprint arXiv:2011.00692*.
- Kudo, T. and Richardson, J. (2018). Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*.
- Kwiatkowski, T., Palomaki, J., Redfield, O., Collins, M., Parikh, A., Alberti, C., Epstein, D., Polosukhin, I., Devlin, J., Lee, K., Toutanova, K., Jones, L., Kelcey, M., Chang, M.-W., Dai, A. M., Uszkoreit, J., Le, Q., and Petrov, S. (2019). Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.
- Lee, D. D. and Seung, H. S. (1999). Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791.
- Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. (2019). Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

- Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., et al. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.
- Lison, P. and Tiedemann, J. (2016). OpenSubtitles2016: Extracting large parallel corpora from movie and TV subtitles. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 923–929, Portorož, Slovenia. European Language Resources Association (ELRA).
- Liu, Y. and Liu, P. (2021). Simcls: A simple framework for contrastive learning of abstractive summarization. *arXiv preprint arXiv:2106.01890*.
- Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.
- Martins, P. H., Marinho, Z., and Martins, A. F. (2022). Chunk-based nearest neighbor machine translation. *arXiv preprint arXiv:2205.12230*.
- McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133.
- Merity, S., Xiong, C., Bradbury, J., and Socher, R. (2016). Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.
- Muennighoff, N., Tazi, N., Magne, L., and Reimers, N. (2023). Mteb: Massive text embedding benchmark. *arXiv preprint arXiv:2210.07316*.
- Nallapati, R., Zhou, B., Gulcehre, C., Xiang, B., et al. (2016). Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*.
- Narayan, S., Cohen, S. B., and Lapata, M. (2018). Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. *arXiv preprint arXiv:1808.08745*.
- OpenAI, :, Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Al-tenschmidt, J., Altman, S., Anadkat, S., Avila, R., Babuschkin, I., Balaji, S., Balcom, V., Baltescu, P., Bao, H., Bavarian, M., Belgum, J., Bello, I., Berdine, J., Bernadett-Shapiro, G., Berner, C., Bogdonoff, L., Boiko, O., Boyd, M., Brakman, A.-L., Brockman, G., Brooks, T., Brundage, M., Button, K., Cai, T., Campbell, R., Cann, A., Carey, B., Carlson, C., Carmichael, R., Chan, B., Chang, C., Chantzis, F., Chen, D., Chen, S., Chen, R., Chen, J., Chen, M., Chess, B., Cho, C., Chu, C., Chung, H. W., Cummings, D., Currier, J., Dai, Y., Decareaux, C., Degry, T., Deutsch, N., Deville, D., Dhar, A., Dohan, D., Dowling, S., Dunning, S., Ecoffet, A., Eleti, A., Eloundou, T., Farhi, D., Fedus, L., Felix, N., Fishman,

S. P., Forte, J., Fulford, I., Gao, L., Georges, E., Gibson, C., Goel, V., Gogineni, T., Goh, G., Gontijo-Lopes, R., Gordon, J., Grafstein, M., Gray, S., Greene, R., Gross, J., Gu, S. S., Guo, Y., Hallacy, C., Han, J., Harris, J., He, Y., Heaton, M., Heidecke, J., Hesse, C., Hickey, A., Hickey, W., Hoeschele, P., Houghton, B., Hsu, K., Hu, S., Hu, X., Huizinga, J., Jain, S., Jain, S., Jang, J., Jiang, A., Jiang, R., Jin, H., Jin, D., Jomoto, S., Jonn, B., Jun, H., Kaftan, T., Łukasz Kaiser, Kamali, A., Kanitscheider, I., Keskar, N. S., Khan, T., Kilpatrick, L., Kim, J. W., Kim, C., Kim, Y., Kirchner, H., Kiros, J., Knight, M., Kokotajlo, D., Łukasz Kondrasiuk, Kondrich, A., Konstantinidis, A., Kosic, K., Krueger, G., Kuo, V., Lampe, M., Lan, I., Lee, T., Leike, J., Leung, J., Levy, D., Li, C. M., Lim, R., Lin, M., Lin, S., Litwin, M., Lopez, T., Lowe, R., Lue, P., Makanju, A., Malfacini, K., Manning, S., Markov, T., Markovski, Y., Martin, B., Mayer, K., Mayne, A., McGrew, B., McKinney, S. M., McLeavey, C., McMillan, P., McNeil, J., Medina, D., Mehta, A., Menick, J., Metz, L., Mishchenko, A., Mishkin, P., Monaco, V., Morikawa, E., Mossing, D., Mu, T., Murati, M., Murk, O., Mély, D., Nair, A., Nakano, R., Nayak, R., Neelakantan, A., Ngo, R., Noh, H., Ouyang, L., O’Keefe, C., Pachocki, J., Paino, A., Palermo, J., Pantuliano, A., Parascandolo, G., Parish, J., Parparita, E., Passos, A., Pavlov, M., Peng, A., Perelman, A., de Avila Belbute Peres, F., Petrov, M., de Oliveira Pinto, H. P., Michael, Pokornyy, Pokrass, M., Pong, V., Powell, T., Power, A., Power, B., Proehl, E., Puri, R., Radford, A., Rae, J., Ramesh, A., Raymond, C., Real, F., Rimbach, K., Ross, C., Rotsted, B., Roussez, H., Ryder, N., Saltarelli, M., Sanders, T., Santurkar, S., Sastry, G., Schmidt, H., Schnurr, D., Schulman, J., Selsam, D., Sheppard, K., Sherbakov, T., Shieh, J., Shoker, S., Shyam, P., Sidor, S., Sigler, E., Simens, M., Sitkin, J., Slama, K., Sohl, I., Sokolowsky, B., Song, Y., Staudacher, N., Such, F. P., Summers, N., Sutskever, I., Tang, J., Tezak, N., Thompson, M., Tillet, P., Tootoonchian, A., Tseng, E., Tuggle, P., Turley, N., Tworek, J., Uribe, J. F. C., Vallone, A., Vijayvergiya, A., Voss, C., Wainwright, C., Wang, J. J., Wang, A., Wang, B., Ward, J., Wei, J., Weinmann, C., Welihinda, A., Welinder, P., Weng, J., Weng, L., Wiethoff, M., Willner, D., Winter, C., Wolrich, S., Wong, H., Workman, L., Wu, S., Wu, J., Wu, M., Xiao, K., Xu, T., Yoo, S., Yu, K., Yuan, Q., Zaremba, W., Zellers, R., Zhang, C., Zhang, M., Zhao, S., Zheng, T., Zhuang, J., Zhuk, W., and Zoph, B. (2023). Gpt-4 technical report.

Over, P., Dang, H., and Harman, D. (2007). Duc in context. *Information Processing Management*, 43:1506–1520.

Peters, M. E., Ruder, S., and Smith, N. A. (2019). To tune or not to tune? adapting pretrained representations to diverse tasks. *arXiv preprint arXiv:1903.05987*.

Petroni, F., Rocktäschel, T., Lewis, P., Bakhtin, A., Wu, Y., Miller, A. H., and Riedel, S. (2019). Language models as knowledge bases? *arXiv preprint arXiv:1909.01066*.

Pfeiffer, J., Rücklé, A., Poth, C., Kamath, A., Vulić, I., Ruder, S., Cho, K., and Gurevych, I. (2020). Adapterhub: A framework for adapting transformers.

- Qian, N. (1999). On the momentum term in gradient descent learning algorithms. *Neural Networks*, 12(1):145–151.
- Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. (2018). Improving language understanding by generative pre-training.
- Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.
- Rae, J. W., Borgeaud, S., Cai, T., Millican, K., Hoffmann, J., Song, F., Aslanides, J., Henderson, S., Ring, R., Young, S., et al. (2021). Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*.
- Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P. J., et al. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67.
- Reimers, N. and Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.
- Robertson, S. and Zaragoza, H. (2009). The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.
- Rush, A. M., Chopra, S., and Weston, J. (2015). A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*.
- Sennrich, R., Haddow, B., and Birch, A. (2015). Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.
- Shaw, P., Uszkoreit, J., and Vaswani, A. (2018). Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*.
- Shazeer, N. and Stern, M. (2018). Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning*, pages 4596–4604. PMLR.
- Shuster, K., Poff, S., Chen, M., Kiela, D., and Weston, J. (2021). Retrieval augmentation reduces hallucination in conversation. *arXiv preprint arXiv:2104.07567*.
- Song, K., Tan, X., Qin, T., Lu, J., and Liu, T.-Y. (2020). MpNet: Masked and permuted pre-training for language understanding. *arXiv preprint arXiv:2004.09297*.
- Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27.

- Sutton, R. S. (2019). The bitter lesson. <http://incompleteideas.net/IncIdeas/BitterLesson.html>.
- Tay, Y., Bahri, D., Yang, L., Metzler, D., and Juan, D.-C. (2020). Sparse sinkhorn attention. In *International Conference on Machine Learning*, pages 9438–9447. PMLR.
- Thorne, J., Vlachos, A., Christodoulopoulos, C., and Mittal, A. (2018). Fever: a large-scale dataset for fact extraction and verification. *arXiv preprint arXiv:1803.05355*.
- Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G. (2023). Llama: Open and efficient foundation language models.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.
- Vijayakumar, A. K., Cogswell, M., Selvaraju, R. R., Sun, Q., Lee, S., Crandall, D., and Batra, D. (2016). Diverse beam search: Decoding diverse solutions from neural sequence models. *arXiv preprint arXiv:1610.02424*.
- Wang, B., Liu, Z., and Chen, N. F. (2023). Instructive dialogue summarization with query aggregations. *arXiv preprint arXiv:2310.10981*.
- Wang, S., Li, B. Z., Khabsa, M., Fang, H., and Ma, H. (2020). Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*.
- Williams, R. J. and Zipser, D. (1989). A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280.
- Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al. (2016). Google’s neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.
- Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., and Le, Q. V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.
- Zhang, B. and Sennrich, R. (2019). Root mean square layer normalization. *Advances in Neural Information Processing Systems*, 32.
- Zhang, J., Zhao, Y., Saleh, M., and Liu, P. (2020). Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, pages 11328–11339. PMLR.

- Zhao, L., Xu, W., and Guo, J. (2020). Improving abstractive dialogue summarization with graph structures and topic words. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 437–449.
- Zhong, M., Liu, P., Chen, Y., Wang, D., Qiu, X., and Huang, X. (2020). Extractive summarization as text matching. *arXiv preprint arXiv:2004.08795*.
- Zhong, M., Liu, Y., Xu, Y., Zhu, C., and Zeng, M. (2022). Dialoglm: Pre-trained model for long dialogue understanding and summarization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 11765–11773.
- Zhu, C., Liu, Y., Mei, J., and Zeng, M. (2021). Mediasum: A large-scale media interview dataset for dialogue summarization. *arXiv preprint arXiv:2103.06410*.
- Zhu, C., Xu, R., Zeng, M., and Huang, X. (2020). A hierarchical network for abstractive meeting summarization with cross-domain pretraining. *arXiv preprint arXiv:2004.02016*.

