Extreme Abstractive Text Summarization

by
Simran (MT21146)

Under the supervision of
Dr. Tanmoy Chakraborty and Md. Shad Akhtar

Submitted in partial fulfillment of the requirements for the degree of Master of Technology, CSE-AI

Indraprastha Institute of Information Technology - Delhi
December, 2022

Certificate

This is to certify that the thesis titled *"Extreme Abstractive Text Summarization"* being submitted by **Simran** to the Indraprastha Institute of Information Technology Delhi, for the award of the Master of Technology, is an original research work carried out by her under my supervision. In my opinion, the thesis has reached the standards fulfilling the requirements of the regulations relating to the degree.

The results contained in this thesis have not been submitted in part or full to any other university or institute for the award of any degree/diploma.

December,2022

Dr Tanmoy Chakraborty, Md. Shad Akhtar

Department of CSE
Indraprastha Institute of Information Technology Delhi
New Delhi 110 020

Acknowledgements

I would like to thank god and my family and definitely my supervisors Dr. Tanmoy Chakraborty and Md. Shad Akhtar, and PhD mentor Yash Kumar Atri. My supervisors had always pushed me to do work and maintain speed. I was carrying out research for the very first time and it wouldn't be possible at all without Dr. Tanmoy, Md. Shad and Yash. Due to their constant encouragement and guidance, I was able to submit and defend my thesis. All three of them have shown empathy and cooperated absolutely well. My supervisors used to hold weekly meetings and invest time in my work, I am grateful to them. Thanks to my PhD mentor Yash Kumar Atri for clearing even trivial doubts and being available always for help.

I would also like to thank my friends who have always given support and helped me out every time. Carrying out research had risks but they helped me to deal with tough times.

My family has always been a big support. Even when I felt like I am drastically failing they pumped me up and consoled me. My father, mother, and younger brother are more than friends. I shared all my insecurities regarding research and they pushed me in hard times and refueled me.

I also thank all the faculty members and staff of the Department of CSE and IIIT Delhi for always helping me throughout my college journey.

Abstract

Extreme Abstractive Summarization of long scientific papers requires domain knowledge and a concise summary maintaining faithfulness to the source and covering novel aspects presented in the paper. Human annotations are indeed expensive for the task, so we propose ExGrapf2, a novel encoder architecture that uses **fractality, FFT, and Graph Convolution Network** as its strong foundation to address the challenge. We observed that when the model is presented with different views of the source, it extracts more information from the same amount of data. ExGrapf2 successfully accomplishes the objective and beats the state-of-the-art models on SciTLDR dataset without any data augmentation. We also used the contrastive loss to enhance the performance further. The novelty is not only for the modules but also for how we fuse them.

ExGrapf2: Fractality-infused Graph Embeddings with FFT Transformer for Extreme Abstractive Text Summarization

Simran (MT21146)

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem Statement

We worked out the task of TLDR[1] generation for the scientific papers. The significance of the problem cannot be denied, given the implausibly prolific amount of papers published each year [IKB18]. The human annotations for the same would be overly tedious and impractical. The considerable time a human annotator spends to create a concise summary is accompanied by the expert domain knowledge required to write down the TLDR while maintaining faithfulness to the source and correctness of the written summary. No surprise that the only dataset to date for extreme summarization of scientific papers is SciTLDR, introduced in [CLCW20]. The dataset is relatively small, especially for a generation task, with only 1992 training samples (for AIC version)[2].

1.2 Our Idea

This calls for the need to create robust systems that can draw out more information given the small amount of data. In this work, we study this objective for extreme abstractive text summarization of long scientific documents and evaluate the proposed model on the downstream dataset: SciTLDR [CLCW20]. An attractive option may be to train the model for more prolonged periods. This leads to overfitting. Is there a way to use every piece of information to make the model more powerful? We discovered that using distinct modules to encompass different views of "the data" really assists the learning for the downstream task, even with less data.

[1] TLDR is an acronym that stands for "too long; didn't read," which is often used in informal online discussions (e.g., Twitter or Reddit) about scientific papers. This term was introduced in the TLDR paper. [CLCW20]

[2] The dataset has three versions: Abstract Only, AIC, Full Text, more details are provided in the Dataset section.

We used DistilBART and integrated four major modules that form the strong foundations of our proposed architecture, **ExGrapf2** that stands for **Ex**treme abstractive summarization using **Grap**h embeddings fused with **f**ractality; and **f**ft transformer. The first is the **FFT**[3]. FFT effectively mixes the tokens and provides the feed-forward sublayers sufficient access to all of the transformer tokens [LTAEO21] (FNet paper). The authors of FNet replaced FFT with attention, while ExGrapf2 accompanies it with the attention mechanism. We observed that FFT could find out some key terms of the source. The intuition comes from the fact that FFT, on a basic level, given a mixed signal, simplifies the components and extracts signal properties and features. Document embedding is also a mixed signal where the simplified components may be sentences that form the document or the words that appear in the source. This is also illustrated in the results of our experiment. While the word-level view (FFT) may seem to find keywords, what about the context? We also need to look for keywords based on "how the words are used and their distribution over the scientific paper." This can be accomplished with **fractality**. For years, fractals have astonished the world of mathematics because of their ability to create complex patterns despite their simple formulation. The self-similarity structure of fractals aligns with how the text is presented in the scientific paper. The paper can be thought of as a structure with "one central concept," which repeatedly occurs in the paper in one form or the other, one way or the other. We are the first to apply the concept of fractality for abstractive text summarization. To date, we could only see fractality used primarily for keyword extraction or extractive summarization. Once done with word-level views, the time calls for sentence-level view. A **Sentence Relation Graph (SRG)** is constructed using MPNet [STQ+20], and SciBERT [BLC19]. **Graph Convolutional Network (GCN)** applied on the **SRG** provides a sentence view of the document and apprehends how a sentence is related to the other sentences of the same document. Both ideas, fractality and GCN, can be combined to give the word and sentence info together. Finally, yet importantly, we adopted the idea from the BRIO [LLRN22] and used a *non-deterministic distribution*, unlike one-point target distribution, for the summary generation so that the different candidate summaries are assigned probability mass according to their quality and using the contrastive loss.

1.3 Summarize Contributions

To summarize our contributions:

- We propose a novel encoder architecture **ExGrapf2**, which can readily be unified with existing encoder architectures and furnish them the capability

[3]Fast Fourier Transform

to draw out more information from the same amount of data.

- ExGrapf2 uses *Fractality-infused Graph Embeddings of sentences and 2D FFT* to apprehend salient details at different levels. It views the task of extreme summarization at the level of words, sentences, and the document.
  - **Word View and its interaction with other words: 2D FFT** (Fast Fourier Transform) effectively mixes the transformer tokens, which provides the feed-forward sublayers sufficient access to all tokens [LTAEO21]. It also can capture keywords (one of the findings of our work).
  - **Individual Word View: Fractality** also caters to keywords capturing but based on the context and its distribution over the entire document.
  - **Sentence View: GCN** finds out the graph embeddings of the sentences from Sentence Relation Graph to yield a holistic view of sentences and the relation of the sentence w.r.t.[4] the other sentences.
  - **Document View: Transformer** The transformer generates embedding for the entire document hence encapsulating the document-level information too.
- We also took the idea from the BRIO [LLRN22] and used a non-deterministic distribution for the summary generation. It used contrastive loss, unlike traditional MLE[5] loss.

[4] with respect to
[5] Maximum Likelihood Estimation

Chapter 2

Related Work

2.1 FFT

The vision of applying DFT[1] instead of attention came from the Google Research Team in FNet paper [LTAEO21] where they showcased that FFT could also enable token mixing effectively and is a potential substitute for attention, while ExGrapf2 explored using attention and 2D-FFT together and see if that makes a difference. We found out that FFT catches the keywords. MHA and FFT seize complementary information and can provide distinct views when used together.

2.2 Fractality

For fractality, we are the first to use fractality for abstractive summarization. [ND15] used fractality to only find the fractal dimension of the words and return top fractal words as keywords. This worked for them because they had a large book corpus, but the exact implementation failed for the SciTLDR dataset. We modified the fractality calculation a bit and used it for small data. We accompanied it with Graph Convolution Network (GCN), to make it more robust.

2.3 Graph Convolution Network (GCN)

GCN was used by [YZM+17] for extractive summarization. They also used handcrafted features which are difficult to decide and fetch. We never know how many features are sufficient. ExGrapf2 uses GCN on the sentence-relation graph formed by sentence transformers, with no reliance on handcrafted features, and is also used for abstractive summarization.

[1] Discrete Fourier Transform, which can be calculated using FFT algorithm.

Chapter 3

Dataset

The dataset used for experimentation is SciTLDR [CLCW20]. It consists of source and target, a TLDR summary. The dataset comes in three types: *Abstract Only*, *AIC (Abstract, Introduction, Conclusion)*, *Full Text*. We used the *AIC* dataset for experimental purposes as it is not as short as *Abstract Only* and not as long as *Full Text*. It is observed that most of the paper's novelty, the problem it tries to solve, are mentioned in the Abstract, Introduction, and Conclusion. Hence we go ahead with this setup. The details of the dataset (train, validation, and test) are stated in Table 3.1.

Table 3.1: Statistics for SciTLDR: Training, Validation, and Test Set. Here, the source length and target length correspond to the length covering 97% to 98% of documents to avoid noise and not take max length.

	Train	**Validation**	**Test**
source length	2000	2000	2000
target length	50	50	50
num samples	1992	619	618

The training data has only one target summary corresponding to one source instance. For validation and test set, some samples have one or more target summaries, and the rouge score can be calculated as the maximum rouge of the generated summary and candidate summaries. Having multiple gold summaries per paper is essential for evaluation because of variability in human-written gold summaries. The results reported in this paper are for the AIC dataset only.

Chapter 4

Methodology

We introduce ExGrapf2 (Extreme abstractive summarization using Graph embeddings fused with fractality; and fft transformer), a novel yet intuitive method for learning to generate TLDRs. Our approach addresses the major challenge of limited training dataset size.

We propose using four distinct modules of ExGraf2 to pull out more information from small data to address the challenge. The proposed architecture is shown in Figure 4.1. As a base model, we used DistilBART [LLG+19]. The different modules are explained below.

4.1 2D - Fast Fourier Transform (Word View and its interaction with other words)

The first module is 2D-FFT[1]. ExGrapf2 explored using attention and 2D-FFT together and see if that makes a difference. This will also help us to know if they produce at least some complementary information. 2D FFT means DFT is calculated twice, 1D DFT along the sequence dimension and 1D DFT along the hidden dimension. We viewed FFT beyond token mixing and expected it to give some keywords of the scientific paper. The intuition comes from the fact that when applied to a mixed signal, the FFT simplifies the components and extracts signal properties and features. A scientific paper can also be viewed as a mixed signal where simplified components are the words or sentences that form the document. Note that the novel concept of the paper is repeated and used more often than the less-important terms, which FFT could intuitively catch. We kept only the real part of FFT, keeping the rest of the architecture constant. The equation for the same is

$$y = \Re(F_{seq}(F_h(x))) \tag{4.1}$$

[1]Fast Fourier Transform

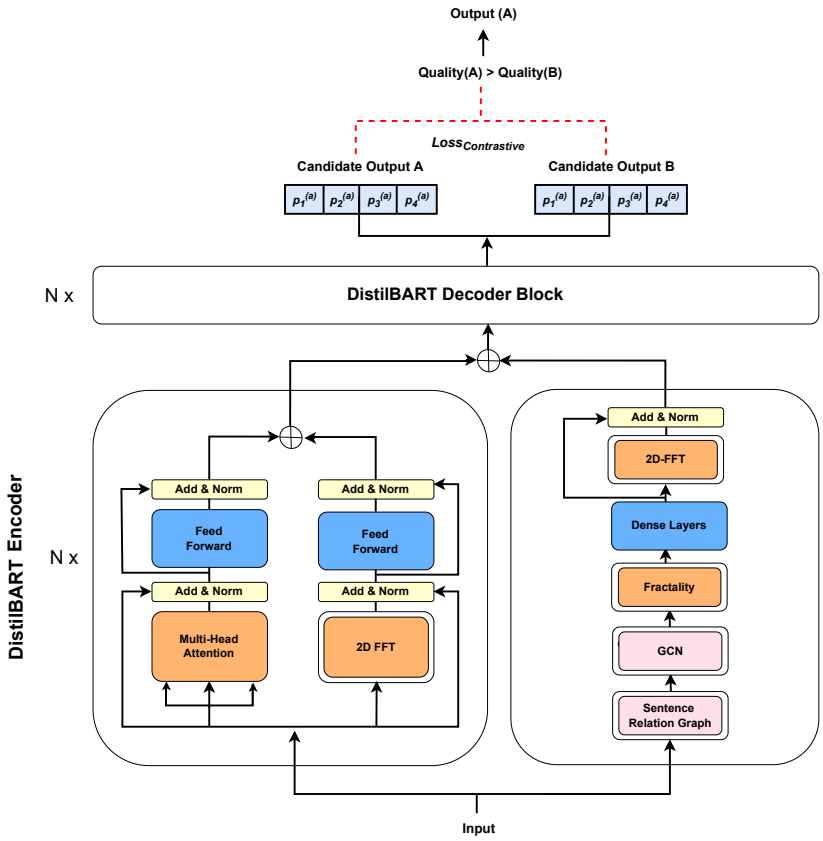

Figure 4.1: ExGrapf2: Model Architecture. The encoder architecture of DistilBART is altered, and the decoder remains unchanged. Newly-added modules in the encoder are shown in double-rounded boxes (2D-FFT, Sentence Relation Graph (SRG), Graph Convolution Network (GCN), Fractality). The contrastive loss is used at the decoder end. The details for SRG and GCN are given in Figure 4.2.

The FFT indeed provides some complementary information[2]. The rationale is that the attention looks at the relationship of one token with the others, which is extensively rich, while FFT could potentially provide some keywords, though not as rich as attention, but is more helpful for NLP tasks, as it offered a "word-level" view of the document. Surprising to know that a standard, unparameterized Fourier Transform could be suitable for the task.

[2]more details are provided in the Experiments and Analysis Section

4.2 Fractality (Word-View)

Having read about 2D-FFT, a fascinating idea of fractality gives an assorted view of the data. Fractals are beautiful to look at and powerful to work with. Behold a fern that consists of many small leaves that branch off a larger one or romanesco broccoli that consists of smaller cones spiraling around a large one; they both are fractals. In simpler terms[3], fractals have self-similarity property. There would be some repeating or iterative procedure that generates fractals. There need not be an exact match; approx similarity also works. An interesting fact about fractals is that even if there is the same repeating unit, it can lead to different structures depending on various factors. For instance, a fern and a tree have the same repeating unit, but they are distinct structures based on the position of the branch and the angle. A term in the scientific paper repeated again and again but with a different context will form a separate unit. For instance, backpropagation may occur multiple times. For Recurrent Neural Networks, in some places, it would be standard backpropagation; in other places, backpropagation would be discussed w.r.t. backpropagation over time. Fractals can inherently identify repeating units with tweaks.

To implement fractality, we attempted using Mandelbrot or Julia set [Bra89] formulations, but they didn't align with the text since they just looked at the bounded or unbounded behavior of a number. We modeled to find the fractal dimension of a word through the box-counting method. The fractal dimension calculation is done similarly to that of [ND15].

The mathematical formulation for the same is given below. In the box-counting method, the number of boxes touched by the document, N, is inversely proportional to box size s, raised to the power D, where D is fractal dimension.

$$N \propto (1/s)^D \tag{4.2}$$

Equation 4.2, can be written in the following fashion:

$$N = c\ (1/s)^D \tag{4.3}$$

The equation 4.3 follows power-law, so applying log on both sides.

$$log\ N = log\ c + D\ log\ (1/s) \tag{4.4}$$

The equation 4.4, is a Linear Regression Equation. The slope of the line will give the fractal dimension. For our problem, the window size is the box size, and the box is considered to be touched if the word under consideration appears in that window. For instance, consider the sentence,

The brown fox jumps over the brown tree with the leaves of brown color. (The word under consideration: **brown**)

[3] a little caveat, more precisely a fractal is by definition a set for which Hausdorff-Besicovitch dimension (D) strictly exceeds the topological dimension

Box size = 2, no. of boxes touched = 3
*The **brown** —— fox jumps —— over the —— **brown** tree —— with the —— leaves of —— **brown** color.*

Box size = 8, no. of boxes touched = 2
*The **brown** fox jumps over the **brown** tree —— with the leaves of **brown** color.*

Box size = 13, no. of boxes touched= 1
*The **brown** fox jumps over the **brown** tree with the leaves of **brown** —— color.*

The fractal dimension of the word can be calculated as:

$$fracDim(word) = \frac{D_{shuffled}(word)}{D_{non-shuffled}(word)} \tag{4.5}$$

The $D_{shuffled}(word)$ is calculated as follows where T is the number of shuffled experiments:

$$D_{shuffled} = \frac{\sum_{i=1}^{T} D_i}{T} \tag{4.6}$$

The reason for this formulation is stated in [ND15]. We know that a sentence's sequence of words matters a lot. If the text is shuffled, then the words won't make sense. Usually, the unimportant words are uniformly distributed over the document, while the important words are present in a clustered fashion. If we have some measure calculated before shuffling, say x, and measure after shuffling, say y, then the degree of fractality can be given as y/x. Reason: After shuffling, the important words won't be clustered, so y will have more value than x. But for the unimportant words, this would remain the same. Hence, the above formulation.

This was for the keyword extraction, but we have never seen fractality applied for abstractive summarization. To view how it's done, we unfold the mystery in Section 4.4.

4.3 Sentence Relation Graph and Graph Convolutional Network (Sentence-View)

To instill the sentence-level view, we constructed Sentence Relation Graph on which the GCN can be applied. The steps to generate the graph are stated below, while Figure 4.2 depicts the same pictorially.

1. For each sentence in a document, compute their MPNet Embeddings.

   Why MPNet? A graph consists of edges and nodes, nodes can simply be sentences with some features, but the question is how to create the edges. It is intuitive to think that edges must be based on some similarity. Hence we computed MPNet embeddings and used them to

calculate sentence similarity (MPNet is shown to be the benchmark for semantic text similarity).

The sentences are the nodes of the graph.

2. An edge is created between two sentences if the cosine similarity of the two sentences is greater than a certain threshold; if the edge is created, then edge weight is merely the cosine similarity value itself.

3. Reinitialize the node features with the SciBERT embedding of the sentence.

   Why SciBERT? SciBERT [BLC19] is a BERT model pre-trained on scientific papers and would give rich embeddings for the task.

4. Apply GCN [ZTXM19] on this graph to compute the embeddings of the sentence based on other similar sentences, thereby incorporating the sentence view of the data. The embedding of the sentence now has its information and the information of its neighboring sentences.

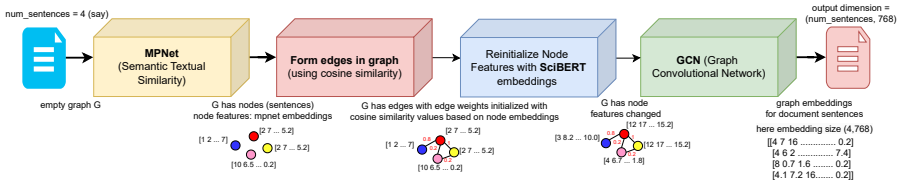

Figure 4.2: SRG and GCN Module of ExGrapf2: Formation of Sentence Relation Graph (SRG) for sentences in the document and applying Graph Convolution Network (GCN) to obtain graph embeddings for document sentences.

4.4 Combining Graph and Fractality: (Word View with Sentence View combined)

Rather than having GCN and Fractality applied individually, we fused them to form a single robust unit. Now the fractal dimension of the word conveys the paper's keywords. We calculated the number of top fractal words present in the sentence of the document and assigned weight to their graph sentence embeddings based on the overlap of words in the sentence with top fractal words. More the amount of top fractal words present in the sentence, the higher the weight of the corresponding graph embedding. We applied this weighting mechanism after graph embeddings were obtained from GCN and before feeding into the encoder layer. We can say that fractality is applied on top of GCN-

induced embeddings. This implies that the sentences with keywords would have more say in the process. [4]

4.5 Contrastive Loss

Finally, adding one more in the bag, the contrastive loss. The notion for the contrastive loss and having a non-deterministic target distribution came from BRIO paper [LLRN22], where the system generates multiple candidate summaries and then learns to rank them too. The idea aligns with our notion of extracting more information from the same data. Hence we used this training regime for ExGrapf2.

4.6 Putting all together

Having stated all the modules, how well they come up as a single unit is the magnificence of ExGrapf2. Figure 4.1 depicts the fusion of these modules into the encoder architecture. FFT has a parallel branch with Multi-Head Attention, and in each encoder layer, they add up after the application of the Feed-Forward Layer. Now, the embeddings add up in the last encoder layer for the graph and fractality combined module. We used static embeddings because they are already SciBERT embeddings fused with fractality, and keeping it as a trainable parameter would make it lose the charm. Hence we added them only with the last encoder layer's output. Since the dimensions aren't the same (the dimensions of the graph depend on the number of sentences in the document), we used Dense Layers to keep the consistency and also used 2D-FFT to boost the performance further and compensate for not making it a trainable parameter. The decoder remains unchanged. The training regime is contrastive learning adopted from BRIO [LLRN22].

[4] this technique can also be used to combine extractive and abstractive summarization (future work). Extractive summarization gives a high rouge, while abstractive summarization generates new phrases and may get a low rouge. This idea would use the best of both worlds.

Chapter 5

Experiments

5.1 FFT

Our approach was to test the modules independently without pre-trained models to know their significance. We wanted to know if the output is the result of the module only, not the pre-trained model. Also, if they complement each other or produce some similar outputs, based on that, we placed all of them in the final architecture of ExGrapf2.

Firstly, to check the FFT module, we took the help of a standard transformer and placed FFT in three ways (Figures 6.1, 6.2, and 6.3, respectively in Section 6.3 (Appendix).

- **Way 1**: 2D FFT is placed in parallel to Multi-Head Attention (Figure 6.1) and they share same feed-forward network.

  Intuition: Multi-Head Attention (MHA) and FFT capture different aspects. MHA looks for relations among tokens, while FFT captures keywords.

- **Way 2**: 2D FFT is placed in parallel to Multi-Head Attention (Figure 6.1) and they have different feed-forward networks.

  Intuition: Both must have different feed-forward networks to give them a chance to learn.

- **Way 3**: 2D FFT is placed in after Multi-Head Attention.

  Intuition: FFT can use the information given by MHA.

The results for all three experiments are shown in Table 5.1. The results are produced when the difference between consecutive training losses is less than 0.001. These figures for rouge are reported for f-measure.

Analysis of the results: From the results in Table 5.1, we can see that all ways of FFT outperformed standard transformer except way 3, but to note that

transformer produces high rouge than way 3 transformer but the summaries for way 3 transformers make much more sense than standard transformer. Anyhow, for way 1 and 2, results also outperform, and the summaries contain some keywords. For more details, glance at Table 6.2 (given in Appendix Section 6.3). **Final Decision:** We go ahead with placement way 2 and used 2D-FFT in parallel with MHA in ExGrapf2. This also shows that MHA and 2D-FFT produce complementary information and give best results when used parallely.

5.2 Fractality

Followed by FFT, we performed an extensive evaluation for fractality. We plotted a curve (Figure 5.1) for the training dataset. The curve depicts the overlap between the top fractal words from the source and target summary. The curve increases till 100, and later, it begins to saturate. Therefore, we went ahead with the top 100 fractal words. GCN embeddings would be weighed according to the top 100 fractal words. We added fractal words as input along with the source in the standard transformer and further compared the results to validate fractality's usage. Table 5.2 confirms the importance of fractality. For divide + log_{10}(frequency) the graph may look better than divide only but the analysis shows that when frequency term is added then words like "the","and" also appears, hence we went ahead with "divide only".

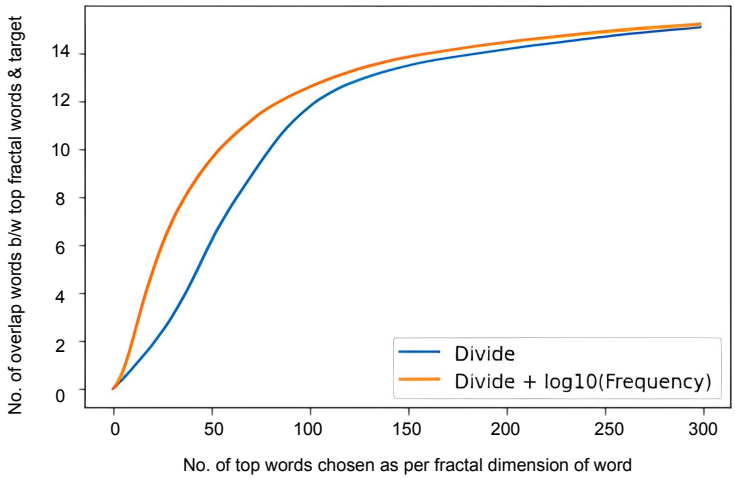

Figure 5.1: The plot of "Number of top fractal words chosen from source" (vs) "Average number of overlapping words between top fractal words and target summary" for the training dataset. The divide operation looks for the degree of fractality as the number of boxes touched after shuffling and before shuffling. In contrast, divide + $\log_{10}$(frequency) also gives importance to the frequency of words in the document.

The Dense Layers are added to keep the dimension consistent. We experimented with and without the 2D-FFT layer on the top of Dense Layers & with and without MHA, and the best setup came up as the one with 2D-FFT.

For the final architecture, ExGrapf2, we played with length penalty, beam width, learning rate, and the final results are illustrated in Section 6.

Table 5.1: Results on testing dataset for three ways of placement of 2D-FFT module. Way 1: 2D-FFT placed in parallel with MHA, and they share the same feed-forward networks, Way 2: Same as way1, except that they have different feed-forward networks, Way 3: 2D-FFT used after MHA.

	Rouge-1	**Rouge-2**	**Rouge-L**
Transformer	12.39	1.12	10.28
Transformer+ FFT (Way1)	13.43	1.12	10.70
Transformer+ FFT (Way2)	14.07	1.49	11.21
Transformer+ FFT (Way3)	11.04	0.63	9.20

Table 5.2: Results on testing dataset for three ways of placement of 2D-FFT module. Way 1: 2D-FFT placed in parallel with MHA, and they share the same feed-forward networks, Way 2: Same as way1, except that they have different feed-forward networks, Way 3: 2D-FFT used after MHA.

	Rouge-1	**Rouge-2**	**Rouge-L**
Transformer	12.39	1.12	10.28
Transformer + Fractality	13.54	0.81	10.96

Table 5.3: Test set max Rouge scores (rouge-1, rouge-2 and rouge-L) of extractive and abstractive baselines and ExGrapf2.

	R1	**R2**	**RL**
PACSUM [ZL19]	28.7	9.8	21.9
LexRank [ER04]	28.9	8.6	20.4
BERTSUMEXT [LL19]	36.2	14.7	28.5
T5 [RSR$^+$20]	37.4	15.4	25.8
BART [LLG$^+$19]	39.8	19.2	31.6
ExGrapf2 (Our Model)	43.7	20.5	34.3

Chapter 6

Evaluation and Analysis

6.1 Evaluation

The metric used for evaluation is Rouge-Score: rouge-1, rouge-2, and rouge-L f-measure. We tested extractive and abstractive baselines to evaluate the performance of our proposed model. Some of the baselines are exactly the same as SciTLDR paper [CLCW20]. Table 5.3 illustrates the performance of ExGrapf2 compared to other baselines. ExGrapf2 beats the best-performing model by 3% even though it uses disilBART.

6.2 Analysis

The system's summaries are shown in Table 6.1. We can see that generated summaries are way better than gold-summaries, but the problem lies in the rouge-score that measures the overlap between words.

Table 6.1: Summaries produced by ExGrapf2

Example 1 (Target Summary)
We propose using five deep architectures for the cybersecurity task of domain generation algorithm detection .
Example 1 (ExGrapf2 Summary)
comparing five different architectures for the cybersecurity problem of DGA detection : classifying domain names as either benign vs. produced by malware (i.e. , by a Domain Generation Algorithm) in terms of accuracy .
Example 2 (Target Summary)
Presents a variational autoencoder for generating entity pairs given a relation in a medical setting .
Example 2 (ExGrapf2 Summary)
we introduce a generative model called Conditional Relationship Variational Autoencoder (CRVAE) , which can discover meaningful and novel relational medical entity pairs without the requirement of additional external knowledge .
Example 3 (Target Summary)
This paper proposes the Recurrent Discounted Attention (RDA) , an extension to Recurrent Weighted Average (RWA) by adding a discount factor .
Example 3 (ExGrapf2 Summary)
Recurrent Weighted Average (RWA) unit captures long term dependencies far better than an LSTM on several challenging tasks .

6.3 Appendix

The three ways tried to check for the placement of FFT are given in Figures 6.1, 6.2, and 6.3, respectively. The results of the summaries generated in three ways are also shown in Table 6.2.

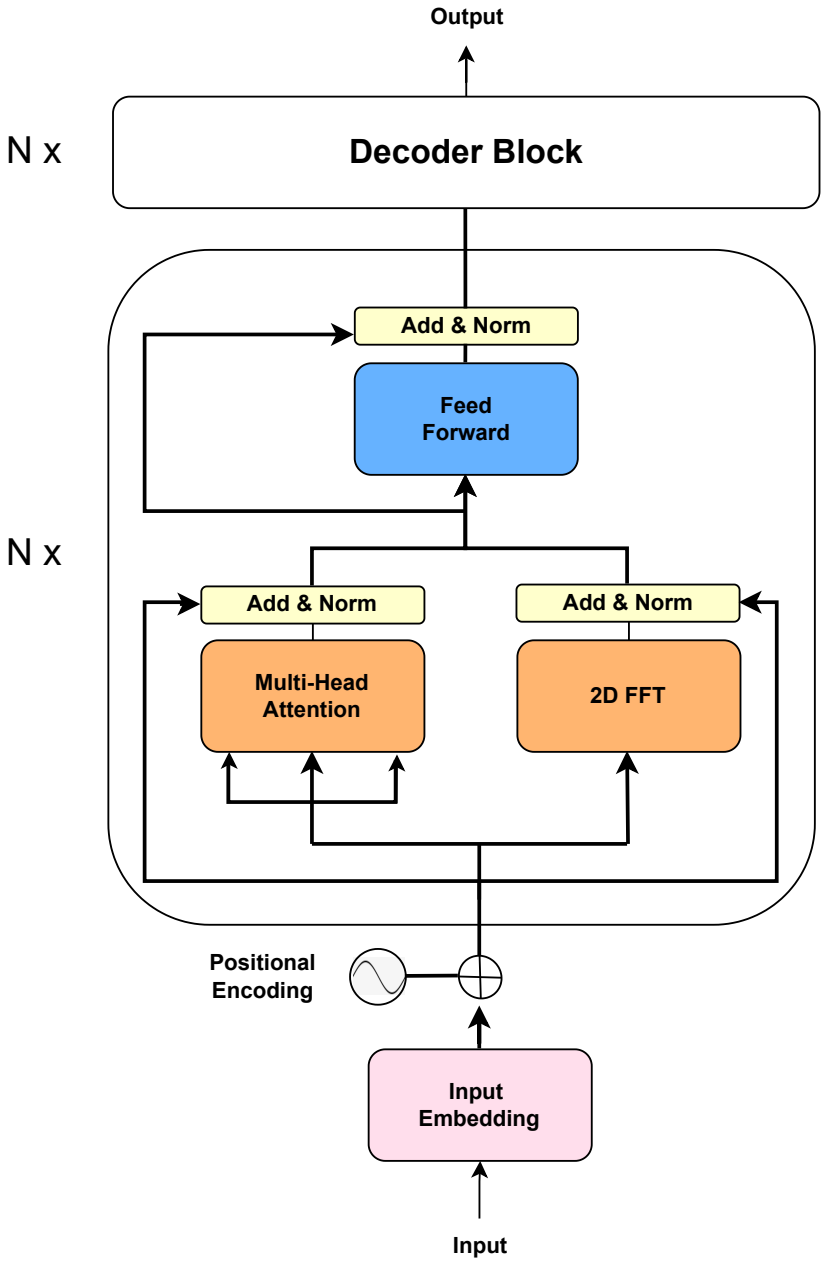

Figure 6.1: 2D FFT is used in parallel with MultiHead Attention, and they share the same Feed-Forward Network.

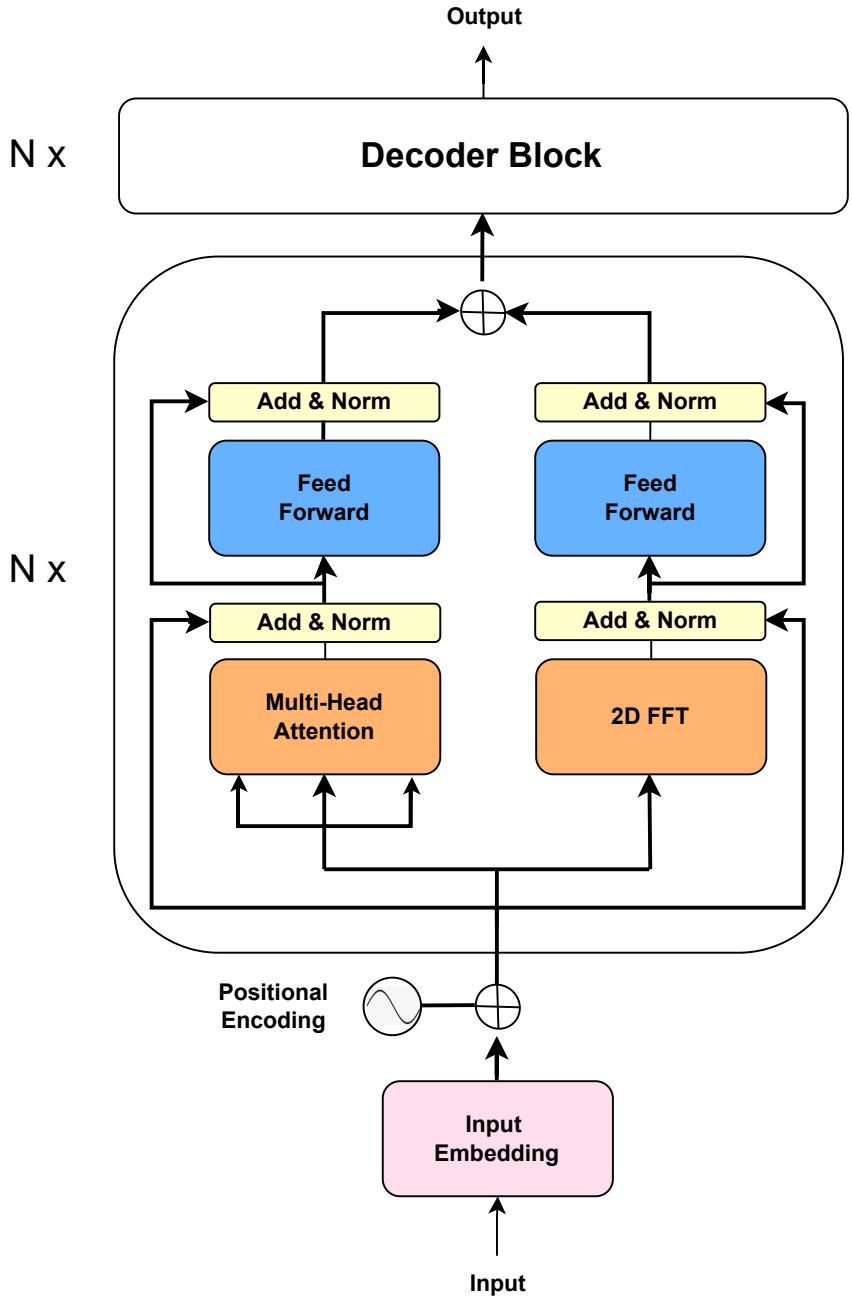

Figure 6.2: 2D FFT is used in parallel with MultiHead Attention, and they have different Feed-Forward Networks.

Table 6.2: Results on testing dataset for three ways of placement of 2D-FFT module. Way 1: 2D-FFT placed in parallel with MHA, and they share the same feed-forward networks, Way 2: Same as way1, except that they have different feed-forward networks, Way 3: 2D-FFT used after MHA.

Target Summary: the paper presents a multi-view framework for improving sentence representation in nlp tasks using generative and discriminative objective architectures.
Transformer: a general and effective model for avoiding negatve transfer in neural network few shot learning
Transformer + FFT (Way 1): multi view learning improves the semi supervised learning improves training and dependency relationship
Transformer + FFT (Way 2): multi view learning improves unsupervised sentence representation learning
Transformer + FFT (Way 3): multi view learning improves unsupervised sentence representation learning

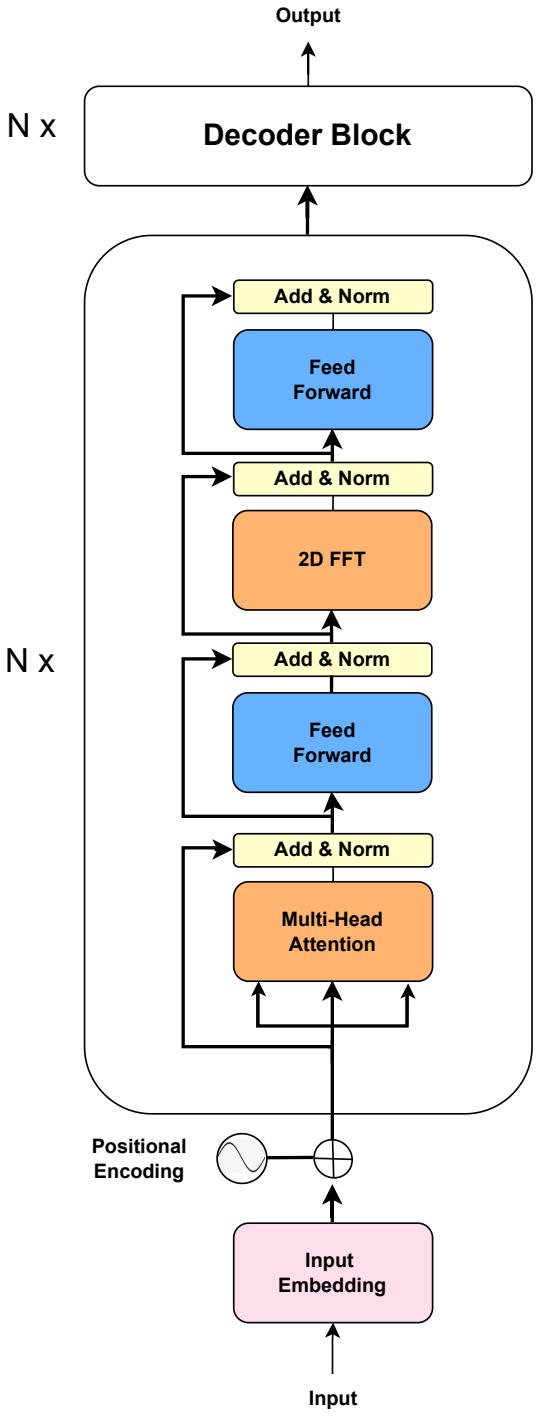

Figure 6.3: 2D FFT is used after MultiHead Attention.

Bibliography

[BLC19] Iz Beltagy, Kyle Lo, and Arman Cohan. Scibert: A pretrained language model for scientific text. *arXiv preprint arXiv:1903.10676*, 2019.

[Bra89] Bodil Branner. The mandelbrot set. In *Proc. symp. appl. math*, volume 39, pages 75–105, 1989.

[CLCW20] Isabel Cachola, Kyle Lo, Arman Cohan, and Daniel Weld. TLDR: Extreme summarization of scientific documents. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4766–4777, Online, November 2020. Association for Computational Linguistics.

[ER04] Günes Erkan and Dragomir R Radev. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of artificial intelligence research*, 22:457–479, 2004.

[IKB18] John Ioannidis, Richard Klavans, and Kevin W Boyack. Thousands of scientists publish a paper every five days, 2018.

[LL19] Yang Liu and Mirella Lapata. Text summarization with pretrained encoders. *arXiv preprint arXiv:1908.08345*, 2019.

[LLG+19] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

[LLRN22] Yixin Liu, Pengfei Liu, Dragomir Radev, and Graham Neubig. Brio: Bringing order to abstractive summarization. *arXiv preprint arXiv:2203.16804*, 2022.

[LTAEO21] James Lee-Thorp, Joshua Ainslie, Ilya Eckstein, and Santiago Ontanon. Fnet: Mixing tokens with fourier transforms. *arXiv preprint arXiv:2105.03824*, 2021.

[ND15] Elham Najafi and Amir H Darooneh. The fractal patterns of words in a text: a method for automatic keyword extraction. *PloS one*, 10(6):e0130617, 2015.

[RSR+20] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67, 2020.

[STQ+20] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. Mpnet: Masked and permuted pre-training for language understanding. *Advances in Neural Information Processing Systems*, 33:16857–16867, 2020.

[YZM+17] Michihiro Yasunaga, Rui Zhang, Kshitijh Meelu, Ayush Pareek, Krishnan Srinivasan, and Dragomir Radev. Graph-based neural multi-document summarization. *arXiv preprint arXiv:1706.06681*, 2017.

[ZL19] Hao Zheng and Mirella Lapata. Sentence centrality revisited for unsupervised summarization. *arXiv preprint arXiv:1906.03508*, 2019.

[ZTXM19] Si Zhang, Hanghang Tong, Jiejun Xu, and Ross Maciejewski. Graph convolutional networks: a comprehensive review. *Computational Social Networks*, 6(1):1–23, 2019.