UNIVERSITY OF COPENHAGEN
FACULTY OF SCIENCE

MSc Thesis

# Summarisation with pixels

Zuzanna Maria Dubanowska

**Supervisor: Desmond Elliot**
**Number of characters: 71906**
**Number of standard pages: 60**

**Submitted: May 30, 2023**

This thesis has been submitted to The Faculty of Science, University of Copenhagen

# Contents

**1 Introduction** **2**

1.1 Motivation . . . 3

1.2 Objectives . . . 4

1.3 Structure of the report . . . 5

**2 Theoretical Background** **6**

2.1 Language Modelling . . . 6

2.1.1 Sequence-to-Sequence Modelling . . . 6

2.1.2 Attention . . . 7

2.1.3 Transformers . . . 8

2.1.4 Pre-trained Language Models . . . 9

2.1.5 Visual Language Modelling . . . 13

2.2 Automatic Text Summarisation . . . 16

2.2.1 Extractive summarisation . . . 16

2.2.2 Abstractive summarisation . . . 17

2.2.3 Extractive vs. Abstractive . . . 18

2.3 Evaluation methods . . . 19

2.3.1 Human Evaluation . . . 20

2.3.2 Automatic Evaluation Methods . . . 20

2.4 Datasets . . . 22

**3 Proposed Approach** **24**

3.1 Architecture . . . 24

3.2 Experiments . . . . . 25

**4 Related work** **28**

**5 Experimental Setup and Results** **30**

5.1 Experimental Setup . . . . . 30

5.2 Results . . . . . 31

5.2.1 How does fine-tuning cross-attention only vs. the whole decoder affect performance? How do different decoder architectures affect it? . . . . . 31

5.2.2 How does fine-tuning the whole network affect performance, as compared to training the decoder only? . . . . . 33

5.3 Qualitative Analysis . . . . . 34

5.3.1 Coherence . . . . . 34

5.3.2 Factual consistency . . . . . 34

5.3.3 Fluency . . . . . 38

5.3.4 Relevance . . . . . 38

**6 Discussion** **39**

**7 Conclusions** **47**

**Bibliography** **54**

## Abstract

The internet has revolutionised how we create, publish and access content offering an instant supply to limitless knowledge online. Finding relevant information, though, proves challenging amidst this sea of data, thus it became crucial to find ways to easily identify, extract and condense valuable knowledge. Automatic text summarisation (ATS) is the process of generating short, informative accounts of larger texts using techniques from Natural Language Processing, Machine Learning and statistics without human help. It has potential to improve every area where we encounter unstructured textual data, from education [1], academia [2], the legal industry [3, 4] to healthcare [5]. We distinct two approaches to ATS: extractive and abstractive. In the former, the summary is a concatenated collection of vital sentences from the original document. In the latter, the system generates new text which paraphrases the sense of the original document.

Since the advent of the Transformer [6], Large Language Models have been continually improving state-of-the-art in abstractive text summarisation, however research efforts are still needed to bridge the gap between human- and machine-generated summaries. Vocabulary bottlenecks are a known issue LLMs suffer from, which can hinder their performance in summarisation tasks as finite vocabularies limit the amount of words a model can use to generate text.

In this work, we investigate the potential of Pixel-based Encoder of Language (PIXEL) [7] for text summarisation. PIXEL frames language modelling as a visual recognition task and operates on visual text representations instead of discrete tokens, mitigating the bottleneck issues. We propose PIXELSum, an abstractive summarisation Seq2Seq model with a pre-trained PIXEL encoder and pre-trained GPT2 decoder. We examine how using different sizes / architectures of the decoder and freezing different parts of the network affects performance when fine-tuned on a summarisation dataset.

The PIXELSum variant where GPT2-medium decoder was used performs the best, with two best performing models depending on the metric: *GPT2-medium* where

the whole decoder was fine-tuned, achieving a ROUGE1 of 16.23, and *GPT2-medium* where only cross-attention was fine-tuned, achieving a BERTScore precision of 61.19%. The generated text for all model variants is fluent, but factually inconsistent and irrelevant, injecting information not present in the document.

Our experimental results give rise to a number of observations, shedding light on what is the most significant in PIXEL-based summarisation models. Evidence shows that when updating the weights of cross-attention only, the model learns steadily, in other cases it overfits. There are indications that larger decoders (300M+ params.) were over-parameterised for the task. Evidence points to that the decoder did not know how to attend to encoder outputs. Taking these observations together, we suggest, that to leverage PIXEL for text generation successfully, the full resulting Seq2Seq model should be pre-trained on a generalised summarisation-alike task, then the cross-attention layers fine-tuned on specific summarisation tasks.

**Keywords**: Language Modelling, Text Summarisation, Visual Text Representations

# List of abbreviations

| | |
|---|---|
| ATS | Automatic Text Summarisation |
| BERT | Bi-Directional Encoder Representations from Transformers |
| GRU | Gated Recurrent Unit |
| FNN | Feedforward Neural Network |
| ML | Machine Learning |
| LM | Language Model |
| LLM | Large Language Model |
| LSTM | Long-Short Term Memory |
| NLP | Natural Language Processing |
| Seq2Seq | Sequence to Sequence |
| RNN | Recurrent Neural Network |
| ViT | Vision Transformers |
| ViT MAE | Masked Autoencoding Vision Transformers |

# Chapter 1

# Introduction

The internet has brought about a content revolution. Everyone, in a matter of a couple clicks can create, publish and access resources like never before. Such access to information is undoubtedly beneficial: it provides an instant supply of limitless knowledge to all. But identifying relevant content and grasping it can prove difficult in this swarming sea of unstructured information. Thus, to benefit from it in the best way, it became crucial to efficiently identify, extract and condense useful knowledge.

Summarisation is the process of generating a brief account of larger texts while preserving their key ideas and important information. Manual summarisation entails a human reading textual entries to be summarised, then identifying key points and producing concise digests that preserve these key points. The process is time consuming [8], and can lead to inconsistencies and inaccuracies as people can perceive different parts of the text as most important [9] rendering this approach to summarisation unfit for the staggering amounts of data that the internet has to offer.

Automatic text summarisation is the task of creating summaries using techniques from Natural Language Processing, Machine Learning and statistics without human help. Since mid-20th century, it is attracting considerable interest due to its widespread benefits. Automatic summarisation systems help save time and effort people spend on manual summarisation of large documents, e.g. summarising legal

documents [4, 3]. They increase efficiency e.g. in education, by outlining key concepts that require most focus [1] or in academia by generating TL;DR (Too Long; Didn't Read) summaries [2]. Summarisation tools help extract most critical information and identify patterns in big (textual) data, enabling better decision-making, e.g. in healthcare to summarise clinical trials [5] or in businesses to make more data-informed decisions. Automated text summarisation algorithms ensure greater consistency and accuracy across same types of text as they follow rigit rules and criteria to produce summaries.

## 1.1 Motivation

In this work, we focus on single document abstractive automatic text summarisation. That is, the task of producing shorter, informative digests that paraphrase the content of individual textual entries, like newspaper articles, blog posts or research articles. The past decade has seen a rapid improvement in the performance of abstractive summarisation systems, due to the invention of Seq2Seq [10] networks. Ever since, the methodologies in this domain have largely been dominated by supervised text-to-text sequence modelling using NNs that learn to produce novel summaries given documents and human-generated reference summaries. Currently, the state-of-the-art in open-ended NLP tasks, amongst others abstractive text summarisation, is continually occupied by Transformer [11] Seq2Seq models, otherwise broadly referred to as Large Language Models, which owe their success to the attention mechanism [12].

The accomplishments of recent research show great promise in abstractive text summarisation, but the results are far from perfect. Significant research efforts are still needed to bridge the gap between human- and machine-generated summaries. The automatically generated summaries can be untrustworthy, as LLMs fail to discern important information or alter the information whilst generating text. Automatic abstractive summarisation systems fail to produce coherent and fluent summaries, as these concepts are difficult to formalise and include in the model's learning process explicitly.

Vocabulary bottlenecks are a known issue LLMs suffer from and another factor, that can negatively affect the quality of the produced summary. If the text contains out-of-vocabulary words, the model will replace them with unknown tokens or apply a suboptimal fallback strategy to find a similar word. This in turn could lead to ambiguities and inaccuracies in produced summary, especially if the particular word was crucial in the document. Finite vocabularies the model has access to, imply a limited amount of words the model can use to generate a summary. This can be especially harmful in summarisation tasks, where a lot of domain-specific nomenclature is used.

## 1.2 Objectives

In this thesis, we investigate the potential of PIXEL [7] (**Pix**el-based **E**ncoder of **L**anguage), a pre-trained language model rendering text as images, for sequence generation, more precisely abstract text summarisation. By substituting the vocabulary embedding layer, and instead rendering text as sequences of patches, the PIXEL model avoids the vocabulary bottleneck issue, allowing it to adapt to unseen scripts and handling out-of-vocabulary words [7].

In its current setup, however, PIXEL is not able to produce discrete words from the pre-trained encoder. We explore leveraging PIXEL for text generation, by creating an encoder-decoder text summarisation, with pre-trained PIXEL encoder and a pre-trained autoregressive model as the decoder.

More precisely, our objectives are to:

- Create an encoder-decoder summarisation model, with PIXEL as the encoder, the decoder of a pre-trained Transformer model and cross-attention binding the two;
- Explore and evaluate how different variants of the decoder or training configurations affect performance of the proposed abstractive summarisation model. Amongst other, address the following: *How do models perform when fine-tuning the whole decoder vs. cross-attention only? Does updating the weights*

*vs. freezing the encoder affect performance, and how? How do different decoder architectures affect performance?*

- Evaluate the model variants using appropriate task-specific metrics and compare them against state-of-the art results on the given benchmark dataset;
- Identify, based on the results, what is needed to leverage PIXEL for text summarisation.

By means of these objectives, we aim to determine how PIXEL can used to best advantage for text summarisation, and more broadly, text generation tasks.

## 1.3 Structure of the report

This paper is organised as follows. Chapter 1 gave a brief introduction to the topic, outlined the motivations and objectives of this thesis. Relevant theoretical background is introduced to the reader in Chapter 2. In Chapter 3 we outline our proposed method. Related work is discussed in Chapter 4. In Chapter 5 we describe the experimental setup and present results of experiments. We evaluate our results in the light of existing research in Chapter 6. Some conclusions and future research directions are outlined in the last chapter.

# Chapter 2

# Theoretical Background

## 2.1 Language Modelling

### 2.1.1 Sequence-to-Sequence Modelling

A sequence is an ordered data structure whose successive datapoints are correlated [13]. In the context of language modelling, sequences implicitly mean textual data sequences, e.g. a sequence of words in a sentence or sequence of characters in a word. Each element of the sequence consists of a feature vector and context, which is provided by the neighbouring elements.

Sequence-to-sequence models [10] map input sequences $x = (x_1, x_2, ..., x_n)$ onto output sequences $y = (y_1, y_2, ..., y_k)$. This model class is a type of an encoder-decoder architecture. The encoder network maps the input sequence $x = (x_1, x_2, ..., x_n)$ onto a lower dimensional latent representation vector $\mathbf{c}$, in a way that all information about the input is retained. Then the decoder network attempts to reconstruct the input from the latent, otherwise context, vector $\mathbf{c}$. More formally, the context vector is calculated in the encoder by [13]:

$$c = h_{e,T}, \quad \text{where} \quad h_{e,i} = f\left(x_i, h_{e,i-1}\right) \quad \forall i \in [1, T]$$

where f represents a recurrent gate, such as an LSTM, GRU, RNN, etc., $h_{e,i}$ is

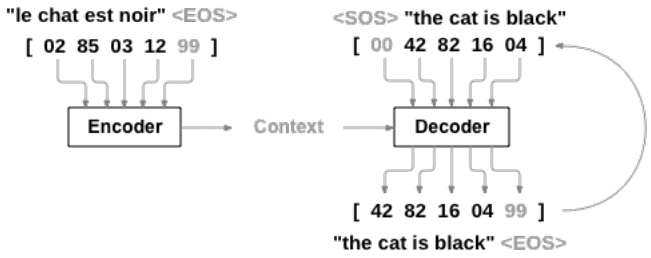



Figure 1: Seq2Seq model applied to machine translation from French to English. The encoder condenses the tokenised input sentence in French into a context vector. The context vector is passed to the decoder, which predicts the next token in English until the end-of-sequence token. Source: Pytorch NLP From Scratch: Transtion with Sequence to Sequence and Attention

referred to as the hidden state of the encoder at time $i$.

The decoder is trained to predict the next token $y_i$, given the context vector $\mathbf{c}$, the previous token $y_{i-1}$, the hidden state of the decoder $h_{d,i}$ at time $i$:

$$p\left(y_i \mid \{y_1, \ldots, y_{i-1}\}, c\right) = g\left(y_{i-1}, h_{d,i}, c\right) \tag{2.1}$$

where the hidden state of the decoder at time $i$ is calculated by:

$$h_{d,i} = k\left(y_{i-1}, h_{d,i-1}, c_i\right) \quad \forall i \in [1, T] \tag{2.2}$$

where g and k are realised by recurrent units, such as an LSTM, GRU, RNN. The output of g() is a probability distribution over the tokens in the vocabulary. Seq2Seq models are widely used to solve complex language modelling tasks, like machine translation, text summarisation or question answering. A machine translation example with the sequence-to-sequence architecture is illustrated in Figure 1.

### 2.1.2 Attention

The encoded context vector in Seq2Seq2 networks is fixed length. This implies that the dimensionality of long/complex sequences and short/simpler sequences is forced to be the same. This causes the hidden state to contain more information about short, than long term dependencies, which is especially problematic in language

modelling, where context often depends on long term dependencies. *Attention* [14] was proposed to mitigate this issue and enable the encoder network to learn the relevance of all elements of a sequence. In this approach, the context vector at each timestep is redefined as a linear combination of the inputs and attention weights $\mathbf{c_i} = \sum_{j=1}^{T} a_{i,j} h_{e,i}$. The higher $a_{i,j}$, the more focus will be paid to that element in the sequence at timestep $i$ and the higher the relative importance of this element to the meaning of the sequence. Attention weights differ from the usual definition of weights in NNs, as they vary with each input.

The probability distribution of next token $y_i$ given context vector is as in Equation 2.1. The context vector is $\mathbf{c_i} = \sum_{j=1}^{T} a_{i,j} h_{e,i}$. The learned attention weights $a_{i,j}$ reflect the alignment of two tokens in the sequence:

$$a_{i,j} = \frac{\exp\left(e_{i,j}\right)}{\sum_{k=1}^{T_x} \exp\left(e_{i,k}\right)}$$

Where $e_{i,j} = a(h_{d,i-1}, h_{e,j})$ is an "alignment model" scoring how well the j-th input and the i-th output match [13].

### 2.1.3 Transformers

The *attention* mechanism was introduced to enable the networks to grasp long-term dependencies as good as short-term dependencies, which improved the performance of Seq2Seq networks on language tasks [15]. The main drawback of these recurrent networks, however, is that they need to process the inputs sequentially, so the operations cannot be parallelised. This, combined with the huge amounts of data NNs need to be trained on became an obstacle to further improving performance of these models. The Transformer [11] model was introduced to mitigate this issue.

**Transformers** are a class of multi-layer encoder-decoder models in which the recurrent connections are removed and the only way units pass information to one another is through self-attention mechanisms.

In the attention layer, the alignment model $e_{i,j} = a(h_{d,i-1}, h_{e,j})$ scores how well the i-th output and the j-th input match. In self-attention, the alignment model is

redefined to $e_{i,j} = a(h_i, h_j)$, and reflects how well different elements within sequence match.

The **self-attention** mechanism takes the input vector $x_i = (x_1, x_2, ..., x_n)$ and learns three versions of $x_i$:

- **Key** $k_i = W_k x_i$, where $W_k$ is a learnable matrix, which is used to establish the weights for the j-th output vector $y_j$;
- **Query** $q_i = W_q x_i$, where $W_q$ is a learnable matrix, which is used to establish the weights for it's own output $y_i$;
- **Value** $v_i = W_v x_i$, where $W_v$ is a learnable matrix, which is used as a part of the context vector's weighed sum to compute the output vector.

We then compute the output $y_i$ from the input $x_i$ as:

$$\begin{aligned} a'_{i,j} &= q_i^T k_j \\ a_{i,j} &= \text{softmax}\left(a'_{i,j}\right) \\ y_i &= \sum_j a_{i,j} v_j \end{aligned} \tag{2.3}$$

**Multi-head attention** mechanism is a major component of the Transformer model. Instead of computing the attention once over the input, the attention mechanism is split into multiple heads, each of which attends to a different subset of the input. This has shown to significantly improve the performance across a wide range of complex NLP tasks [16]. The Transformer architecture and the mechanism of multi-head attention are discussed in Figures 2 and 3 respectively.

### 2.1.4 Pre-trained Language Models

Pre-trained language models are large neural networks that are trained on very large amounts of textual data in a self-supervised or unsupervised manner. The pre-trained models are capable of understanding natural language and can be used for, or fine-tuned to, downstream NLP tasks.

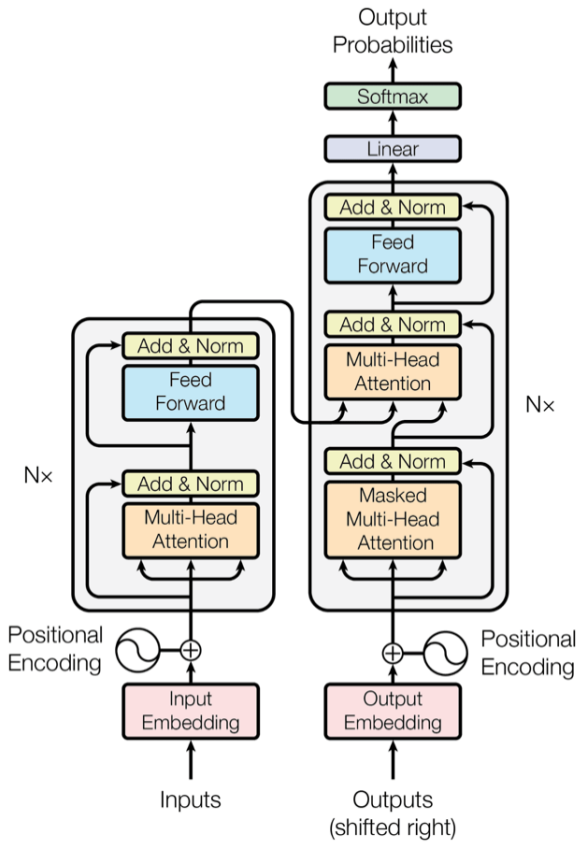



Figure 2: The Transformer [12] architecture. The sequential input is converted into embeddings and positional information is injected. The positional embeddings are fed to the encoder. The encoder consists of a stack of N identical layers which consist of a multi-head attention sublayer, output of which is fed to a fully-connected feed-forward sublayer consisting of two linear transformations with ReLU activation. Each of the sublayers has a residual connection around it. As the last operation, the sum between the output of each sublayer and the sublayer's input is normalised. The decoder cosists of a stack of N identical layers as well. The first block applies multi-head self-attention over the previous output of the decoder. The following decoder block receives the queries (Q) from the first decoder sublayer and the keys (K) and values (V) from the encoder's output. The last sublayer of the decoder is identical to the last sublayer of the encoder, implementing a fully connected feed-forward network. A dense layer with softmax is applied to compute the most probable next word of the sequence. Source of figure: *Attention is all you need, Vaswani et al.* [12].

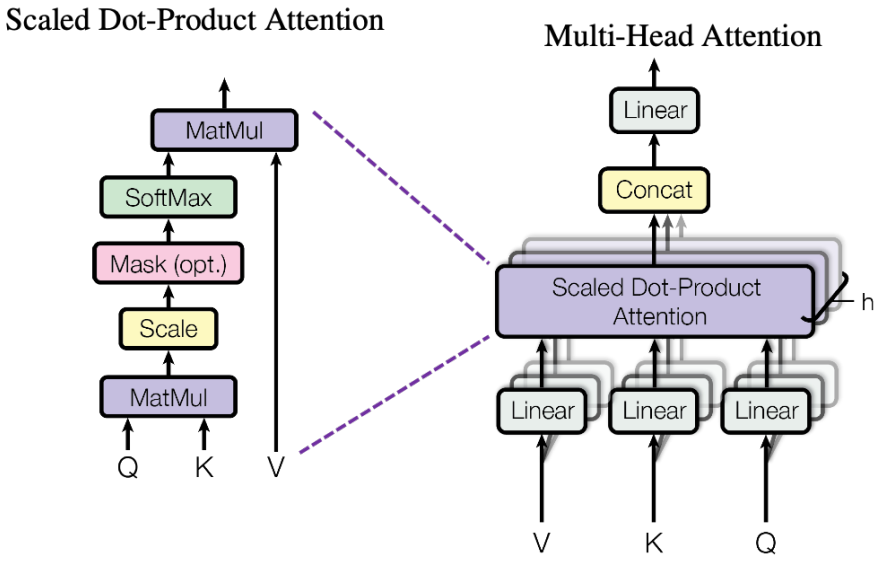



Figure 3: Architecture of the Multi-Head attention block from Figure 2. Each depicted layer is it's own attention mechanism ("head") computing it's own scaled dot-product attention defined as in Equation 2.3. The output from each head is then concatenated and projected back into the model's expected output shape. By allowing the model to attend to different parts of the input sequence with different queries, multi-head attention can capture different aspects of the input sequence simultaneously, improving the model's overall performance. Source: [17]

**BERT** (**B**idirectional **E**ncoder **R**epresentations from **T**ransformers) [18] is an encoder-only Transformer, which learns by randomly masking certain input tokens and predicting them from context. It has been pretrained on a large corpus of textual data, including the entire Wikipedia and physical books and has a total of 340 million parameters. Since BERT is an encoder-only model, it's output is the hidden state vector of the encoder. To use it for downstream tasks, task-specific data is applied to fine-tune all the parameters, and a classification layer is used to the make predictions from model outputs.

**GPT, GPT-2, GPT-3** are consecutive generations of the **G**enerative **P**re-trained **T**ransformer LM, which in contrast to BERT are decoder-only. The GPT models are *autoregressive*, meaning given a sequence of tokens $(x_1, ..., x_n)$, the model it predicts the following token $x_{n+1}$. The consecutive iterations of the model are scale-ups and improvements of their predecessors. The models have been pretrained on web data, including BookCorpus (GPT) [19], WebText and Reddit posts (GPT-2) [20], Common Crawl, books and Wikipedia (GPT-3) [21]. GPT has 117 million parameters, GPT-2, which was introduced as a direct 10-fold scale-up to its predecessor has 1.5B parameters, GPT-3 features 175B parameters. The GPT models are primarily meant to be used as zero- or one-shot learners, GPT and GPT-2 can be fine-tuned to downstream tasks, GPT-3 does not allow it [13].

**BART** (**B**idirectional and **A**uto-**R**egressive **T**ransformer) [22] is an example of an encoder-decoder pretrained language model. BART differs from the aforementioned models, in that it uses two objectives in its training. The denoising autoencoder objective is to learn to reconstruct corrupt input sentences, i.e. mask certain input tokens (like BERT), but also shuffle them randomly. The autoregressive objective, like in GPT models, is to generate the next token in the sequence conditioned on previous tokens. BART can be fine-tuned and leveraged in downstream NLP tasks.

### 2.1.5 Visual Language Modelling

#### Vision Transformers

The Transformer model was developed for sequence-to-sequence tasks in NLP. Due to its success it has been adopted across other Artificial Intelligence domains, including Computer Vision. Vision Transformers (ViT) [23] are Transformer model architectures capable of processing image data. The input image is a 2D-array of pixel values, which is split into a sequence of fixed size patches and flattened. These patches are then treated as tokens and processed in the same way as sequences of textual tokens in NLP tasks: the Transformer encoder is used generate a sequence of feature vectors and the resulting sequence is fed into a classification head for classification. Similarly to the success of Transformer architectures in NLP, the ViT achieves state-of-the-art results on many image classification benchmarks [23].

ViT-MAE (**V**ision **T**ransformer with **M**asked **A**uto**E**ncoder) [24] is a computer vision architecture that combines the Transformer-based ViT model with a Masked Autoencoder (MAE) for unsupervised pre-training. The ViT component is defined as above. The MAE component of ViT-MAE is a type of autoencoder trained on masked input images to reconstruct the original image. In the MAE training process, a subset of the input patches is randomly masked, and the model is trained to predict the original pixel values of the masked patches. ViT-MAE leverages the power of Transformer-based models and the MAE unsupervised pre-training stage to achieve state-of-the-art results on a variety of image recognition tasks.

#### Language Modelling as Visual Recognition

Encoder language models with finite vocabularies suffer from a vocabulary bottleneck at two levels: encoding of tokens the model has not seen before, and estimating the probability distribution over the entire vocabulary.

PIXEL [7] (**Pix**el-based **E**ncoder of **L**anguage) is a language model which reframes language modelling as a visual recognition task. PIXEL is based on a ViT-MAE architecture: it renders text as images and processes them in the same way as the


Figure 4: An example of text processed by the PIXEL text renderer, before patching. Black patches serve as right end-of-sequence markers. All patches after the end-of-sequence marker are ignored by the model for attention and loss computation Source: [7]

Masked Autoencoding Vision Transformer does. It's pre-traning objective is to reconstruct the pixels in masked pixel patches. This approach mitigates the bottleneck issues that models defined over finite vocabularies face. Instead of implementing a vocabulary embedding layer, rendering text as pixel-patches allows the model to process any words, or more generally glyphs, regardless if seen before. It's training objective is also less computationally expensive than predicting a distribution over tokens in the vocabulary [7].

PIXEL consists of three major components: the text renderer, which creates an image representation of the text, the encoder, which encodes the unmasked pixel patches and a decoder, which reconstructs the masked patches. The pretrained PIXEL encoder can be finetuned to downstream tasks in a similar fashion to BERT-like encoder-only transformers, the decoder is only used in pre-training.

**The text renderer** takes in a piece of text as input and produces an RGB image $\mathbf{i} \in R^{HxWxC}$, where H is the height of the image, W is the width of the image and C is the number of channels. The renderers standard setting is H = 16px, W = 8484px, C = 3, which is equivalent to 529 pixel patches, each 16 by 16 pixels. Figure 4 provides an illustration of rendered text and patched input can be seen in Figure 5. For further technical details, please refer to the original paper.

### Model architecture

PIXEL is a 112M parameter ViT-MAE architecture, featuring a 86M parameter 12-layer ViT encoder and a 26M parameter 8-layer Transformer decoder. The architecture of the model for pretraining and finetuning is depicted in Figure 5. First, the text processed by the renderer is projected onto a linear sequence of square 16 x 16 pixel patches of maximum length 529 patches and positional embeddings

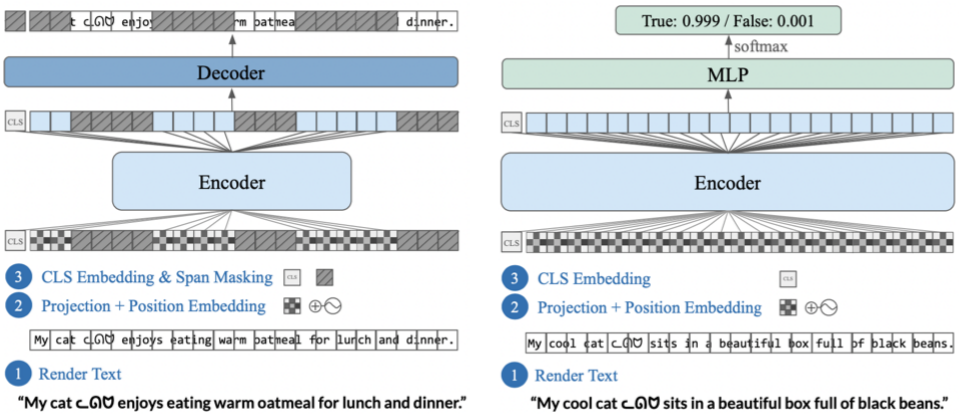
are appended. Then, PIXEL uses a span masking algorithm (see original paper for in-depth explanation), which masks sequences of up to 6 consecutive patches with a dynamically set number of masked patches in between them, to mask $\approx 25$ % of the tokens. The unmasked patches with special CLS tokens preprended are processed by the encoder and representation vectors are output. In pre-training, the 8-layer Transformer decoder is used to reconstruct masked patches at pixel-level. In fine-tuning, the PIXEL decoder can be substituted with a classification head for downstream tasks. PIXEL has been pretrained on data similar to BERT [18].

#### Performance

PIXEL was evaluated on numerous downstream classification NLP tasks, including part-of-speech tagging, question answering and language understanding. The model displayed better ability than BERT to transfer to scripts not found in pretraining data (i.e. non-Latin scripts) by a margin, however falls behind BERT when processing Latin scripts, including English. PIXEL is found to be more robust to noisy text inputs, including low-level orthographic noise. These results show clear benefits of visual language modelling using pixel-based representations.

#### Current limitations for text generation

PIXEL renders text as images and learns to reconstruct the masked patches at *pixel level* during pretraining. In the basic setting, where the image slices are 16 by 16 pixels, there is no bijective mapping between patches and words and the model's current pre-training objective is not to reconstruct patches autoregressively. Thus, in this setup, PIXEL cannot be used for language generation as it is not possible to produce discrete words from the pretrained decoder. This could be achieved by redefining how the masked pixels are reconstructed in the decoder, so it autoregressively reconstructs patches one at a time. This however falls beyond the scope of this thesis.




(a) PIXEL pretraining (b) PIXEL finetuning

Figure 5: Architecture of PIXEL for (a) pretraining and (b) fine-tuning. Source of figure: [7]

## 2.2 Automatic Text Summarisation

Automatic text summarisation (ATS) is the process of generating a brief account of larger text while preserving its key ideas and important information without human help. The goal is to extract salient information and present it in an orderly manner using human language. We identify the following parts of automatic summary generation [25]. First part is content selection, where key information is chosen and redundancies are discarded. Text contains a lot of information and the content of the summary needs to be chosen based on it's relevance for the main points of the document. The second part is information ordering, where the key ideas extracted from the document are ordered to form a coherent whole. The last part is sentence generation, where the aim is to create a clear and human-readable summary. ATS employs techniques from NLP, ML and statistics to execute all the aforementioned parts and generate summaries [25].

### 2.2.1 Extractive summarisation

There are two main approaches to ATS: extractive and abstractive. Extractive summarisation systems select vital sentences from the original document and concatenate them to create the digest version [26]. Content of the summary chosen based on significance scores assigned to sentences. These are calculated from sta-

tistical and linguistic word and sentence level features. Word level features could include, but are not limited to:

- Keyword features, where the sentences are scored based on how many keywords are present[1];
- Title word features, where the sentences containing title words are ranked higher;
- Named entities, where the sentences containing named entities are treated as more important.

Sentence level features could include:

- Sentence location, where the sentences in the beginning and the concluding part of the document are considered more important;
- Sentence length, where longer sentences are assumed to carry more information and prioritised over shorter sentences.

To generate the summary from identified important sentences, the sentences are ordered based on their significance score in a descending order and concatenated, with no changes to their structure or wording.

### 2.2.2 Abstractive summarisation

Abstractive summarisation systems interpret and analyse the content of the document to select the relevant parts, then generate a novel summary, rather than simply extracting the important sentences from the text. Advanced NLP techniques, such as word embeddings, are used to create a vector representation of the text, capturing the semantic relationships in it. Seq2Seq modelling based on encoder-decoder architectures, like RNNs and more recently Transformers, has become the dominant technique for abstractive text summarisation [28]. The encoder is trained to capture the overarching idea of the text into the context vector, then the decoder learns to generate the summary word by word given that context vector.

[1] Where keywords are identified using NLP techniques, e.g. TF-IDF [27]

### 2.2.3 Extractive vs. Abstractive

Both approaches to summarisation have their strengths and weaknesses. Extractive summarisation systems create summaries by identifying, extracting and concatenating vital sentences from the original text, but without "understanding" the meaning behind them. Extractive summaries are faithful, they preserve the original information and do not skew meaning, as the system retains unchanged sentences from the source text [26]. For the same reason, extractive systems are well-fitted for tasks, where documents that need to be summarised contain a lot of factual information, like legal documents or scientific papers [29, 30]. Extractive summarising (non-neural) models are straightforward to implement and require comparatively less data than neural abstractive systems [31], making them the preferred option for highly specialised task or tasks with scarce data.

While generally successful, these systems have their limitations. Extractive summaries can be incoherent or unreadable for the target audience. Concatenating sentences from the original document does not imply the summary will form a cohesive whole [32]. Extractive summarisation systems can fail to be informative and reliable, as they rely on statistical patterns in text, and may not always capture the nuance and context of the original document or miss important information that is not explicitly stated in the text [33]. This limits the usefulness of this summarisation technique when applied to complex text i.e. where information is convoluted or unstructured.

Abstractive summarisation systems generate novel text to form a meaningful summary. Contrary to extractive systems, these models do not rely on actual sentences from the text and base the content of the summary on the learned meaning of the document. Abstractive summarisation models generally distill complex, or implicitly stated, information and capture nuance of the original document better than extractive systems [33], as they analyse the semantic relationships in the passage. This approach to summarisation generates more human-like summaries, which increases the readability and coherence of abstractive summaries as compared to their

extractive counterparts [30].

Abstractive models are the preferred approach to summarisation in research nowadays. They are considered to hold a lot of potential and be more useful than extractive models [34]. However, at present time, they still suffer from shortcomings. These summarisation systems can be untrustworthy as they are prone to introduce new, alter existing or omit information when generating a summary [35, 36, 9]. For the same reason, these models can lose the context of the summary. Abstractive summarisation algorithms are based on neural networks, hence require large amounts of training data to work effectively [31], so they may not be suitable for tasks in languages or domains with limited available data [37]. Lastly, extractive summarisation algorithms choose candidate sentences for the summary based on clearly defined metrics. Abstractive algorithms rely on the internal representations of the model, which is difficult to interpret and understand [8]. The resulting lack of transparency makes it challenging to determine how the model produced a particular summary, how to evaluate the quality of the generated summary or how to improve the model's performance [38].

## 2.3 Evaluation methods

Evaluating machine-generated summaries is a challenging and very complex task. We observe that there might not exist a unique possible output, as text is open to multiple interpretations. Instead, there are multiple comparable outputs the quality of which also depends on the purpose of the generated summary, the context in which it will be used, the background of the person using it, etc. Additionally, the concept of "goodness" of the summary itself is ambiguous and can be highly subjective among different assessors [38].

There are many different aspects of summaries that need to be evaluated. Various dimensions include [9], but are not limited to the linguistic quality of the summary, i.e. it's grammatical correctness; summary's coherence, i.e. whether it forms a well-structured, logical whole; it's readability, i.e. if it can be easily comprehended by the

intended recipient; faithfulness, i.e. if all the information presented in the summary can be found in the document; informativeness, i.e. if the summary contains all the important information from the document. These aspects all come with their own challenges and require individual methods to formalise and standardise [38].

### 2.3.1 Human Evaluation

Human evaluation of summaries is crucial to development and evaluation of summarisation algorithms as it directly measures the satisfaction of the target users. It is currently considered the most rigorous approach to evaluation [39]. Human evaluation can focus on assessing the intrinsic qualities of the summary, i.e. it's creadability, correctness, cohesiveness. It can evaluate the usefulness, faithfulness and informativeness with respect to the original text. Manual evaluation can also involve comparing multiple versions of summaries, to determine which one is the best for the end user. This could involve evaluating the accuracy, relevance and informativeness but also preference [9].

### 2.3.2 Automatic Evaluation Methods

Most automatic metrics so far focus on content of summary and rely on comparing the machine-generated with human-made summaries. The content can be evaluated in terms of syntactical similarity, or semantic similarity between the generated and reference summary (i.e. human generated).

**ROUGE** (**R**ecall-**O**riented **U**nderstudy for **G**isting **E**valuation) is a set of syntactic similarity metrics that measures the overlap between words, phrases and sentences between the generated and reference summary. Several ROUGE metrics can be calculated. ROUGE-1 to ROUGE-4 compute the recall of the overlap of uni-gram to four-grams between the automatic and gold standard (i.e. reference) summary, while ROUGE-L focuses on longest common sequence [38]. We formally define the metric:

$$\textbf{ROUGE} = \frac{\sum_{S \in \{GoldStandard\}} \sum_{ngram \in S} Count_{match}(ngrams)}{\sum_{S \in \{GoldStandard\}} \sum_{ngram \in S} Count_{all}(ngrams)}$$

where $Count_{match}(ngrams)$ denotes the number of ngrams that co-occurred both in the generated and gold standard summary.

ROUGE is the most widely used metric to evaluate automatic summarisation. The metric only assesses the syntactic similarity, i.e. content of the sentence, and does not account for quality aspects such as grammatical correctness, coherence, or readability. It relies on lexical overlap, although an abstractive summary could express they same content without any lexical overlap. To mitigate the effects of subjectiveness of summarisation and generally low agreement between annotators, the metric was constructed to evaluate a generated summary against a set of reference summaries. However, recent datasets in text summarisation usually only provide one golden standard summary, which leads to misuse of the metric. Due to it's serious drawbacks, it's adequacy for performance evaluation is questionable. Despite that, the metric is used in most research studies in the domain to enable comparison between papers.

Another family of metrics has been proposed to mitigate some of the aforementioned issues, which focuses on semantic similarity instead of syntactic similarity between tokens in summaries. To evaluate the goodness of a generated summary, the summaries' embeddings[2] are computed and the cosine similarity between the embedding vectors of the golden standard and generated summary are a direct measure of it's quality.

**BERTScore** [40] is a recently introduced language generation evaluation metric based on pretrained BERT [18] embeddings. Given a tokenized reference sentence $x = (x_1, x_2, ..., x_n)$ and a candidate sentence $\hat{x} = (\hat{x}_1, \hat{x}_2, ..., \hat{x}_k)$ BERT contextual embeddings are used to represent the tokens and the matching is computed using cosine similarity. Formally, we define the recall, precision and F1 scores of BERTScore as follows [40].

[2]Sentence embeddings are vector representations of sentences that capture their meaning.

$$R_{\text{BERT}} = \frac{1}{|x|}\sum_{x_i \in x} \max_{\hat{x}_j \in \hat{x}} \mathbf{x}_i^\top \hat{\mathbf{x}}_j, \quad P_{\text{BERT}} = \frac{1}{|\hat{x}|}\sum_{\hat{x}_j \in \hat{x}} \max_{x_i \in x} \mathbf{x}_i^\top \hat{\mathbf{x}}_j, \quad F_{\text{BERT}} = 2\frac{P_{\text{BERT}} \cdot R_{\text{BERT}}}{P_{\text{BERT}} + R_{\text{BERT}}}.$$

## 2.4 Datasets

Supervised ATS relies on availability of large, high quality datasets of documents and summaries. Data collection for such datasets is an expensive and laborious task, since manual human labelling is often necessary to assure high quality of summaries. There are several large text summarisation datasets available, we summarise information about them in Table 1.

| Dataset | Summary type | Source | Size | Length | Best model | ROUGE1 |
|---|---|---|---|---|---|---|
| CNN / DailyMail [41] | Abstractive | News stories | 286,817 / 13,368 / 11,487 | 766 / 53 | MoCa [42] | 48.88 |
| NEWSROOM [43] | Both | News stories | 1,321,995 | 658.6 / 26.7 | PEGASUS-large [28] | 45.15 |
| WikiHow [44] | Abstractive | WikiHow website | 230,000 | 579.8 / 62.1 | BertSum [45] | 35.91 |
| New York Times [46] | Abstractive | News stories | 650,000* | 500.74 / 50.74 | N/A | N/A |
| Gigaword [47] | Abstractive | News articles | 3,803,957 / 189,651 / 1,951 | 579.8 / 62.1 | PEGASUS + DotProd [48] | 40.60 |
| XSum [49] | Abstractive | News articles | 204,045 / 11,332 / 11,334 | 431.07 / 23.26 | PEGASUS 2B + SLiC [50] | 49.07 |
| Webis-TLDR-17 [51] | Abstractive (TL;DR**) | Reddit | 3,848,330 | 270 / 28 | Transformer + Copy | 22.00 |

Table 1: Text summarisation datasets. If the dataset size is given as three separate numbers, the numbers correspond to train / validation / test splits that the dataset already contains. Length is represented by two numbers and corresponds to document / summary. *The New York Times dataset is larger than given in the table, but only 650,000 datapoints have summaries. **TL;DR stands for Too Long; Didn't Read and is a more subjective way to summarise text.

**CNN / Daily Mail** [41] contains news stories and articles from CNN and Daily Mail with human generated abstractive summaries. The dataset contains more than 300,000 unique document-summary pairs in total.

**NEWSROOM** [43] is a large dataset containing 1.3 million articles and summaries collected across newsrooms in 38 major publications from 1998 to 2017. The dataset contains professionally made extractive, abstractive and mixed summaries.

**WikiHow** [44] is a large-scale collection of instructions from the WikiHow.com knowledge base. Each of the 230,000 examples contains a step-by-step how-to instruction paragraphs and a summarising sentence.

**New York Times** [46] contains almost 2 million articles published by New York Times between 1987 and 2007. Amongst that 650,000 manually summarised articles, making this subset useful for development and evaluation of ATS systems.

**GigaWord** [52] is a large headline-generation summarisation dataset with more than 6 million data points, based on the GigaWord corpus [27]. It uses first sentence and abstractive headline from articles as document-summary pair.

**XSum** [49] is an extreme summarisation dataset, consisting of more than 220,000 news articles from BBC covering a wide variety of domains, accompanied by professionally written one-sentence summaries.

**Webis-TLDR-17** Corpus [51] is a social media summarisation dataset consisting of almost 4 million posts and human generated TL;DR summaries from a social news forum Reddit. Both the textual entries and summaries are subjective, contrary to aforementioned datasets.

# Chapter 3

# Proposed Approach

Transfomer-based sequence-to-sequence models consisting of an encoder and an autoregressive decoder initialised from checkpoint and then finetuned have shown to be very effective for language generation tasks [53]. We leverage this approach to create a novel language model and explore the potential of *modelling language with pixels* for text summarisation.

## 3.1 Architecture

PIXELSum is a sequence-to-sequence model for abstractive text summarisation. The proposed model is an encoder-decoder architecture featuring a pre-trained PIXEL encoder and pre-trained GPT2 as decoder as shown in Figure 5. Input documents are rendered as images and patched using PIXEL's text renderer. Then the encoder processes the input and the learned representation, otherwise last hidden state, is passed to the GPT2 decoder transformer. Since these two models work with different vector spaces, to pass the data, we connect them together using multi-head cross attention, enabling each of the decoder layers to attend to the encoder outputs. Lastly, the decoder outputs the summary of the input document. The objective is to minimise the cross entropy loss between the generated output and reference summary. We evaluate the generated summaries in terms of ROUGE score and BERTScore against the reference summaries.

## 3.2 Experiments

We initialise the proposed PIXELSum model as an EncoderDecoderModel from pre-trained PIXEL and GPT2 configurations. The objective function is `cross entropy loss`, adjusting the weights of the network w.r.t to the generated output according to the formula:

$$L_{CE} = -\sum_{1}^{n} y_i * \log \hat{y}_i \tag{3.1}$$

where $y_i$ is the i-th true token and $p_i$ is the i-th predicted token.

Our setup is highly experimental, as PIXEL has not been used for text generation before. Our aim is to evaluate PIXEL's capability to be used in a text summarisation model, which could give an initial overview of it's potential for text generation. We investigate how the summarisation model performs given:

- We only update the cross attention layer weights during fine-tuning on text summarisation data, while the encoder and decoder remain frozen. In this configuration the weights of the cross attention layer get updated using aforementioned `cross entropy loss` and for the remaining weights the gradient is not computed i.e. they are frozen;
- We fine-tune the whole decoder for text summarisation and the encoder remains frozen. In this configuration the weights of the decoder (including cross attention) get updated and the weights of the encoder remain frozen;
- We fine-tune the whole model for text summarisation. In this variant, the weights of the entire model: the encoder, cross attention and decoder get updated aiming to minimise the `cross entropy loss`.

In the experiments, we try out different versions of GPT-2, which vary in their number of parameters, number of transformer layers and number of heads in multi-head attention, summarised in Table 2. We skip the largest variant of the decoder,

| **Decoder model** | *GPT-2* | *GPT-2 medium* | *GPT-2 large* |
|---|---|---|---|
| **Architecture** | 12-layer, 768-hidden 12-heads | 24-layer, 1024-hidden 16-heads | 36-layer, 1280-hidden 20-heads |
| **Trainable part** | *Whole decoder* | | |
| **Trainable params.** | 117M parameters | 345M parameters | 774M parameters |
| **Trainable part** | *Cross attention only* | | |
| **Trainable params.** | 28M parameters | 101M parameters | 237M parameters |

Table 2: GPT-2 decoder variants with their respective architectures and number of trainable parameters w.r.t which parts of the decoder are trained. Source of data: [6]

GPT-2 XLarge, in our experiments, as we did not have sufficient computational resources to train it.

Our approach is loosely inspired by *Ramos et al.*'s model SmallCap [54] who introduced a Vision-to-Text CLIP-GPT2 model for image captioning. Authors results suggest that Seq2Seq models initialised from pre-trained Transformers can be leveraged without pre-training as a whole, and instead just fine-tuning the newly introduced cross-attention layers and achieve competitive results. This prompted us to train the *cross attention only* PIXELSum variant and not pre-train PIXELSum as a whole.

## Dataset

The encoder can take input sequences up to 529 patches in length. Given this constraint, we aimed to choose a dataset in the lower range of average document lengths (see example benchmarks in Table 1). We initially investigated the Webis-TL;DR-17 [51] corpus, however the character of the dataset made it unfit for the task. Webis-TL;DR-17 is a social media dataset, covering a very broad range of topics with highly specific nomenclature. The entries are written by Reddit's users, not professionals, thus lack consistent structure across the corpus, contain slang, highly subjective entries, and figurative speech. The nature of TL;DR (too long, didn't read) summaries were found polarised and subjective to summarisers' own opinions. We found the dataset contains too much noise to enable us to effectively evaluate the proposed model.

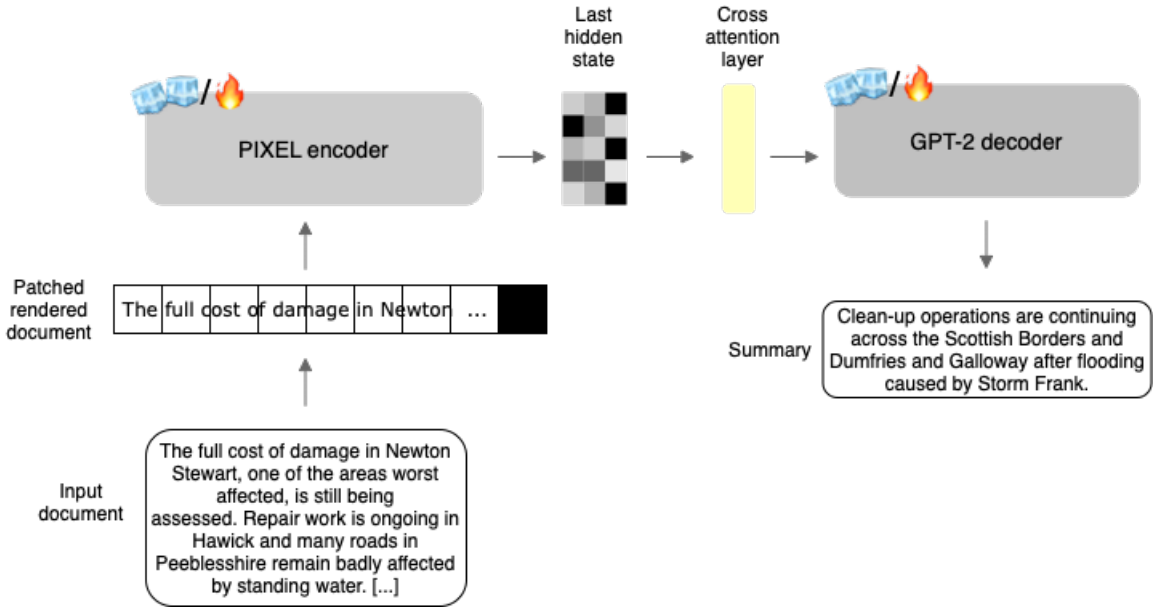



Figure 6: The architecture of PIXELSum, the proposed abstractive text summarisation model. For brevity's sake, the input document depicted in the figure has been truncated. The example is taken from XSum dataset [49]. The ice cubes and fire symbols in encoder and decoder blocks denote the block is frozen or trainable respectively, i.e. it's parameters are not updated / updated during fine-tuning.

We use XSum [49] dataset in our experiments. This corpus is a collection of news articles from BBC, which provides a level of consistency in the writing style, while still covering a large variety of topics to assure good generalisation capabilities of the model. The one-sentence abstractive summaries were professionally written, mostly by the authors of the articles [49]. XSum is a popular benchmark dataset for evaluating Transformer-based abstractive summarisation systems which allows us to compare our model with previous studies.

# Chapter 4

# Related work

With the advent of pre-trained transformer-based language models, the state-of-the-art performance in abstractive text summarisation has advanced significantly. Due to their successes, encoder-decoder transformer architectures are the most widely used approach for this task nowadays.

We summarise related work in terms of characteristics of proposed models and present our model in Table 3. We proceed by discussing some key similarities and differences between state-of-the-art and our proposal.

### Type of language model

BART, T5, BRIO and Uni-LM are general LLMs aimed to be finetuned for various language tasks. Their pre-training objectives were not designed to prepare the model for a particular task. PEGASUS, on the other hand, is an instance of LLM specialised for text summarisation and aimed to be fine-tuned on specific summarisation tasks. It's pretraining objective was intentionally designed to resemble summarisation. Our model comprises of general purpose LLM encoder and decoder, aimed to be fine-tuned for a specific summarisation task. The PIXEL pre-training objective is to reconstruct masked pixel-patches at pixel level, while GPT2's objective is autoregressive, i.e. to predict the next token in a sequence. The important difference between the state-of-the-art and our model, is that our proposed architecture

| Model | Characteristics |
|---|---|
| BART [22] | - Large Pre-Trained LM, aimed to be fine-tuned for natural language generation, translation and comprehension<br>- Denoising autoencoder with Tranformer-based machine translation architecture for pre-training Seq2Seq models<br>- Two pre-training objectives: denoising autoencoder and autoregressive<br>- Bi-directional training: considers both left and right hand side contexts when generating output |
| T5 [55] | - Large Pre-Trained LM, aimed to be fine-tuned for various downstream NLP tasks<br>- Text-to-text Transformer-based architecture<br>- Pre-training objective: Unified text-to-text framework to transform an input sequence into an output sequence<br>- Uni-directional training: considers only left context for output generation |
| PEGASUS [28] | - Large Specialised LM for text summarisation, aimed to be fine-tuned on specific summarisation tasks<br>- Transformer-based encoder-decoder architecture with gap-sentence pre-training<br>- Pre-training objective intentionally alike summarisation task: mask important sentences from documents and learn to generate them from unmasked sentences<br>- Bi-directional training: considers both left and right contexts for summary generation |
| BRIO [56] | - Large Pre-Trained LM, aimed to be fine-tuned on downstream NLP tasks, such as language modeling, translation, and text classification<br>- Bidirectional Recurrent Inner-Outer Transformer architecture, combining recurrent and transformer layers<br>- Pre-training objective: predict the next word in a sequence in both directions (Bidirectional Recurrence Prediction Task)<br>- Bi-directional training: considers both contexts for output generation, as per the objective function |
| Uni-LM [57] | - Large Pre-Trained LM, aimed to be fine-tuned on various downstream tasks<br>- Unified Transformer-based architecture, that can be used for encoder-only and encoder-decoder tasks<br>- Pre-training objective: multi-task, that combines both masked language modelling (encoder-only) and Seq2Seq prediction tasks (encoder-decoder)<br>- Bi-directional training: processes input starting from both left and right side |
| PIXELSum | - PIXEL: Large Pre-Trained encoder-only LM which can be fine-tuned on downstream classification tasks in a similar way to BERT-like encoder transformers; GPT-2: Large Pre-Trained decoder-only LM meant to be fine-tuned for sequence generation or zero-, one- or few-shot learning; the PIXELSum model is to fine-tuned for abstractive text summarisation from the pre-trained components<br>- Encoder-Decoder Transformer-based architecture, with a ViT-MAE encoder and Transformer autoregressive decoder with cross attention<br>- Pre-training objectives: PIXEL's pre-training objective is to reconstruct masked pixel-patches at pixel level, where the patches refer to text rendered as images and pieced; GPT-2's pre-training objective is to predict the next word in the sequence from previous context<br>- Training objective: PIXELSum's objective is to generate abstractive summaries of documents by minimising the cross-entropy loss between reference and generated summaries without pre-training the model as a whole. |

Table 3: State-of-the-art and our model in terms of characteristics.

comprises of two models pre-trained with independent objectives.

### Modalities

In contrast to state-of-the-art, our model employs two modalities for text summarisation: vision and text. The other models work in a single embedding space and process text as the only modality, while PIXEL-GPT2 combines two independent embedding spaces, processes text as images and outputs text.

### Architecture

When it comes to architecture, both state-of-the-art models and our algorithm are Transformer-based architectures. Some proposed models enhance their architecture with additional components, such as recurrent units on BRIO or denoising autoencoder in BART. All the architectures, however, are single-modal. PIXEL-GPT2, in contrast, combines a Vision Transformer (ViT-MAE [23]) with an autoregressive Transformer decoder.

# Chapter 5

# Experimental Setup and Results

## 5.1 Experimental Setup

The model's encoder is initialised from `PIXEL-base` and decoder from `GPT2` pre-trained checkpoints publicly available from HuggingFace [58]. We set-up the project environment according to PIXEL's guidelines. We use a pre-processed version of XSum dataset, available via HuggingFace's datasets Python module. The code base is written fully in Python 3.9 [59], and modelling is done using HuggingFace `transformers` [58] compiled with PyTorch [60].

For the text renderer, we use PIXEL's pre-training configuration unchanged. We use Google Noto Sans font collection, font size of 13.33 at 120DPI and render with PangoCairo backend in 8-bit greyscale.

The models are trained with a standard cross-entropy loss, using an AdamW [61] optimiser. The initial learning rate is 0.001, with a linear scheduler and no warm-up steps. The per GPU training batch size is set to 8, with 4 gradient accumulation steps, giving a total batch size of 32 per GPU. The overall total batch size depends on the computational resources used.

We train the models with early stopping based on ROUGE1 metric, where the training stops if the metric doesn't increase on the validation set for 3 consecutive

evaluations. The evaluation period was set to 200 steps, equal to one epoch. The best performing model, in terms of ROUGE1, is chosen for inference and tested on the test data. For text generation we use beam search with beam size 3. We infer using ROUGE and BERTScore metrics.

We have conducted the experiments with varying computational resources, with 3 - 8 NVIDIA A100 GPUs, with 40GB of memory each, and 24 - 128 CPUs with 80GB memory. Training takes from 10 to 30 hours, depending on number of trainable parameters and computational resources used.

## 5.2 Results

Through our experimental results, we wish to determine what is required to successfully use PIXEL for text generation. We break this aim down into a few experimental questions and discuss findings below. In the analysis we refer to the models or model groups frequently. To ease describing them, we refer to models where only cross-attention was fine tuned as "cross attention only" or "xa only", we refer to models where the encoder was frozen and the weights of the entire decoder were updated during fine-tuning as "whole decoder". We refer to models where the whole model was fine-tuned (i.e. no parts were frozen) as "whole model".

### 5.2.1 How does fine-tuning cross-attention only vs. the whole decoder affect performance? How do different decoder architectures affect it?

We present the results pertaining to this research question in Table 4. We observe two trends emerging depending on the method of evaluation. In terms of ROUGE score, training the whole decoder, as compared to cross-attention layers only improves performance. Within their architecture types, whole decoder models show improved ROUGE scores, as compared to their cross-attention only counterparts. In each case, the performance improved by a similar margin of 3-5pp for ROUGE1, 1pp

| Model configuration | ROUGE1 | ROUGE2 | ROUGE-L | BERTScore | | |
|---|---|---|---|---|---|---|
| *Range / submetric* | *(1 - 100)* | | | *F1 (0 - 1)* | *Recall (0 - 1)* | *Precision (0 - 1)* |
| *Whole decoder* | | | | | | |
| PIXEL + GPT-2 small | 13.15 | 3.29 | 9.80 | 0.594 | 0.6803 | 0.5401 |
| PIXEL + GPT-2 medium | **16.23*** | **4.50*** | **12.32*** | **0.6366** | **0.6972** | **0.5887** |
| PIXEL + GPT-2 large | 14.28 | 3.42 | 10.53 | 0.601 | 0.6821 | 0.5269 |
| *Cross-attention only* | | | | | | |
| PIXEL + GPT-2 small | 9.91 | 2.75 | 8.94 | 0.5997 | 0.6902 | 0.5653 |
| PIXEL + GPT-2 medium | **13.29** | **3.68** | **11.26** | **0.6704*** | **0.7214*** | **0.6119*** |
| PIXEL + GPT-2 large | 11.95 | 3.11 | 10.13 | 0.6412 | 0.6804 | 0.5879 |

Table 4: Evaluation results on XSum test set in terms of ROUGE and BERTScore metrics, w.r.t. different model architecture and parts of model trained. Bolded font denotes best model in its group (cross-attention only / whole decoder) and asterisk (*) denotes best performance overall.

for ROUGE2 and ROUGE-L.[1]. Within their categories, whole decoder vs. cross-attention only, *PIXEL+GPT-2 medium* outperforms the other two model configurations by a similar margin, better than *PIXEL+GPT-2 small* by 2-4pp, 1pp and 2-3pp for ROUGE1, ROUGE2 and ROUGE-L respectively. The *PIXEL+GPT-2 medium* achieves better ROUGE score than *GPT-2 large* by 2pp (ROUGE1), 0.5pp (ROUGE2), 1.5pp (ROUGE-L). The best performing PIXELSum variant overall in terms of ROUGE score is *whole decoder PIXEL+GPT-2 medium* achieving 16.23 ROUGE1, 4.50 ROUGE2 and 12.32 ROUGE-L.

When comparing BERTScore results, the cross-attention only models outperform the whole decoder category. Within their architecture type, *cross attention only PIXEL+GPT-2 small* outperforms *whole decoder PIXEL+GPT-2 small* in terms of recall and precision by 1-2pp., *PIXEL+GPT-2 medium cross-attention only* outperforms it's whole decoder counterpart by a margin of 3pp. across all three BERTScore metrics. *GPT-2 large cross-attention only* achieves better F1 and precision and scores similarly on recall when compared to *GPT-2 large whole decoder*. *PIXEL+GPT-2 medium cross-attention* achieves the best score overall in terms of BERTScore.

Regardless of the metric, however, we observe that the *PIXEL+GPT-2 medium* performs the best, followed by *PIXEL+GPT-2 large* and lastly *PIXEL+GPT-2 small* within their respective training categories.

[1]pp. stands for percentage points

| Model | ROUGE1 | ROUGE2 | ROUGE-L |
|---|---|---|---|
| BART [22] | 45.14 | 22.27 | 37.25 |
| BRIO [56] | 49.07 | 25.59 | 40.40 |
| Uni-LM [62] | - | | |
| PEGASUS [28] | 47.21 | 24.56 | 39.25 |
| T5 [55] | 36.76 | 14.69 | 30.07 |
| PIXELSum (GPT2 medium whole decoder) | 16.23 | 4.50 | 12.32 |

Table 5: A comparison of best performing PIXELSum in terms of ROUGE with other models fine-tuned on XSum.

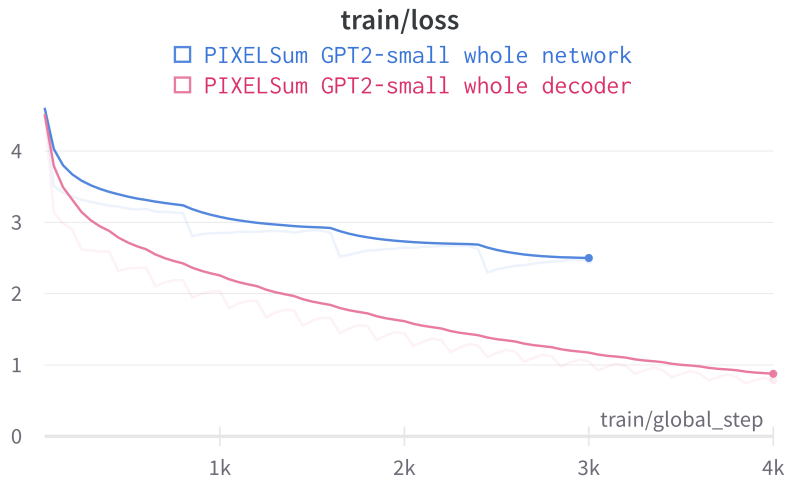



Figure 7: Comparison of training losses of two PIXELSum model variants with the same architectures when the encoder is frozen and enabled to update. The difference in number of global steps is due to early stopping.

### 5.2.2 How does fine-tuning the whole network affect performance, as compared to training the decoder only?

We investigated whether fine-tuning the whole network, i.e. the encoder, cross-attention and the decoder improve the performance of the model. We performed a PIXELSum experiment with a GPT-2 small architecture and compared the training losses of the models, as depicted in Figure 7.

By comparing the training loss functions of PIXELSum GPT2-small where the whole network was trained vs. the whole decoder, we observe that enabling the encoder's weights to update does not lead to improved performance. When the whole network is fine-tuned, the learning abilities are worse than when the encoder is frozen.

The evaluation results are also in favour of this observation, the model achieved a ROUGE1 score of **0.0726** and BERTScore of **0.5571** F1, **0.6263** recall and **0.5061** precision respectively. These are the weakest results we obtained and all models, whose performance results are in Table 4 outperform this model. From this experient we conclude that fine-tuning the whole network does not lead to performance improvements. We have not performed further experimentation with this network configuration as it seems not to be a prominent direction.

## 5.3 Qualitative Analysis

Below we present examples of documents, reference summaries and summaries generated during evaluation by different variants of PIXELSum and evaluate them across four dimensions: coherence, consistency, fluency, relevance as outlined in [9].

### 5.3.1 Coherence

Coherence, according to the DUC 2005 guidelines [63] is defined as: "The summary should be well-structured and well-organized. The summary should not just be a heap of related information, but should build from sentence to sentence to a coherent body of information about a topic." XSum, the dataset we used in our experiments, was constructed with the goal to create short, one-sentence summaries and, with some exceptions, the generated summaries are also one-sentence. Coherence pertains to the organisation of a paragraph and thus we cannot evaluate all our results uniformly using this metric.

### 5.3.2 Factual consistency

We define a factually consistent summary, as one which contain only statements, entities, etc. present in the original document [64]. We evaluate consistency on two levels: word level, i.e. whether or not detailed descriptions in text match these in the summary (e.g. quantities of money, location, etc.) and summary level, i.e. whether the generated excerpt tells a story contained in the text.

| | |
|---|---|
| **Document** | The 3kg (6.6lb) dog is set to become part of a search-and-rescue team used for disasters such as earthquakes.<br>Its small size means it will be able to squeeze into places too narrow for dogs such as German Shepherds.<br>Chihuahuas, named after a Mexican state, are one of the the smallest breeds of dog.<br>"It's quite rare for us to have a chihuahua work as a police dog," said a police spokeswoman in Nara, western Japan.<br>"We would like it to work hard by taking advantage of its small size." Momo, aged seven, will begin work in January. |
| **Reference** | A chihuahua named Momo (Peach) has passed the exam to become a dog in thepolice force in western Japan,<br>in what seems to be a first. |
| **GPT2-small entire decoder summary** | A giant yellow rubber dog has been seized from a Pacific island by police who say it is the first of its kind to go missing.<br>As the search continues in the Seychelles. As the search continues for a new home for the American owner,<br>there have been calls for it to be listed. |
| **GPT2-medium entire decoder summary** | A dog called Sinacurry has been born at the Sea Life Centre in Birmingham.<br>The tiny marsupial has found a new home. It turns out to be a pretty impressive looking one.<br>As it can't be called. It looks like something straight out of a science fiction novel.<br>It turns out to be a pretty impressive looking one. As you probably wouldn't know.<br>It's a special dog. As you probably wouldn't recognise it. It's called nanawesome. It's called nanawesome. It's whacky. |
| **GPT2-large entire decoder summary** | A dog called Lofana has been given a set of spring decorations by the Japanese port city of Okinawa. |
| **GPT2-small xa only summary** | A new breed of dog has been named after the Japanese city of Shibuya. |
| **GPT2-medium xa only summary** | The first Chihuahua police dog in the world has been saddled with a new name.<br>He is called Atlas, or Shorty. He is short, short and very thin.<br>He is very well built but very well trained. So what does he mean? |
| **GPT2-large xa only summary** | A giant dog called Pepper called is being used by Japan to help build a city record for the largest number of dogs in the world.<br>ukan city of Haru Kagawa. |
| **GPT2-small whole model summary** | A man has been jailed for life for murdering his ex-partner in a "brutal" and "brutal"attack.<br>The victim's family have said they are "devastated" by his "brutal" actions.<br>They have lost their appeal against his conviction. |

Table 6: An example of document, reference summary and PIXELSum generated summaries from XSum test set. Factual inconsistencies, particularly, where the model introduced explicitly false new named entities or descriptors, between documents and summaries are denoted in red. Correctly identified named entities etc. from the original document (highlighted in yellow) in generated summaries are highlighted in green.

In the example in Table 6, every generated summary introduces new, false information into the text. These mostly concern location, e.g. *"Pacific Island"* or *"port city of Okinawa"*, the dog's name, e.g. *"Sinacurry"* or *"Atlas"* or the dogs size, e.g. *"giant"*. Only the excerpt generated by PIXELSum with GPT2-medium decoder cross-attention only correctly replicates factual information in the summary. In the following example in Table 7 we observe a similar pattern, all generated summaries introduce false information. They involve fictitious locations, e.g. *"Wirral"* or *"island of Ireland"*, the subject of the text, e.g. *"Viking treasure"* or *"100 huge standing stones"* (the real subject is coins) or specifying the material of the coins: *"gold and silver"*, which is not explicitly stated in the document. In the last example 8, the model's captured the object of the passage, *"Royal Dutch Shell"*, correctly in all except one case, where it thought it to be *"Oil and gas giant BP"*, Shell's rival mentioned in the document. The models introduce some information, which is not present in the document, such as e.g. monetary quantities *"$7bn"* or named entities

| | |
|---|---|
| **Document** | The conservationists have described it as a major milestone, there are thought to be about 70,000 coins in the hoard. By the end of the day on Friday the team had removed 55,014 coins, 63 more than the previous largest Roman Cunetio hoard found in Wiltshire. The team are clearing the coins in view of the public in a special laboratory at La Hougue Bie Museum in Grouville. The Jersey Heritage team has spent two years removing the coins one by one from the hoard discovered by two metal detector enthusiasts in 2012. The value of the hoard will not be known until next year when the States will have to decide whether to pay to keep it in the island. |
| **Reference** | The team removing coins from the Jersey Celtic hoard have gathered more than in any other British hoard to date. |
| **GPT2-small entire decoder summary** | A hoard of Roman gold and silver coins, described by an expert as the most important in the history of the art, have been discovered in Wirral. |
| **GPT2-medium entire decoder summary** | The largest of the largest Celtic hoards of Viking treasure ever found have been unearthed on the island of Ireland. |
| **GPT2-large entire decoder summary** | Metal Age- Age chess and other artefacts from the early 18th Century have been found in an East Yorkshire museum. |
| **GPT2-small xa only summary** | The world's largest hoard of gold and silver coins has been handed over to the Isle of Wight. |
| **GPT2-medium xa only summary** | Treasure hunters have found around 100 huge standing stones buried near Stonehenge. All this year have been hidden in the ground. Cheers have been enjoying the hunt. Laurence Cawley has been keeping an eye out for them. |
| **GPT2-large xa only summary** | Treasure hunters in Scotland have uncovered a hoard of gold coins from Stonehenge which they believe may be worth more than 31m. |
| **GPT2-small whole model summary** | A man has been arrested on suspicion of murder after a woman was found dead at a house in Kent. |

Table 7: Another example of document, reference summary and PIXELSum generated summaries from XSum test set. Factual inconsistencies, particularly, where the model introduced explicitly false new named entities or descriptors, between documents and summaries are denoted in red. Correctly identified named entities etc. from the original document (highlighted in yellow) in generated summaries are highlighted in green.

*"BG Group"*.

The summaries are largely false and misalign with the stories in the documents to varying extents. The passage in Table 6 talks about a small Chihuahua dog joining the police force in western Japan and briefly mentions the benefits of it's size for the tasks it will do. The summaries speak of *"a rubber dog being seized and going missing"*, to *"a dog being born in Birmingham, UK"*, to *"a breed of dog being named after a Japanese city"*. There is some overlap in the topic, as all summaries mention a dog, it's name or qualities, or the country of Japan, however the entire summaries are fictional and false. In the following example in Table 7 the pattern is similar, the passage updates on the conservationists' progress gathering coins from a hoard discovered by some enthusiasts in the past. The models, while correctly recognising the hoard of coins as the main object of the passage, inject false information not present in the article whatsoever, e.g. "*described by an expert as the most important in the history of art*" or "*treasure hunters have found around 100 [...] stones buried*

| | |
|---|---|
| **Document** | The Anglo-Dutch Giant said profits on a current cost of supply measure (CCS) - which strips out price fluctuations - jumped to $3.4bn from $1bn last year. A 55% rise in oil prices in the first quarter of 2017 compared with a year earlier was the main driver of profits. Excluding one-off items, profits on the CCS measure were $3.86bn, up 136%. Shell joins rivals BP, Exxon Mobil, Chevron and Total in reporting better-than-expected results. More than $1bn in cost savings and budget cuts made over the past three years from cost-cuts and assets sales have also helped to increase cash flow and boost profits. But Shell chief executive Ben van Beurden said the company would be making $25bn worth of investments this year in new oil and gas projects. "We continue to reshape Shell's portfolio and to transform the company," he said in a statement. Oil and gas production, known as upstream, rose 2% in the quarter to 3.752 million barrels of oil equivalent from 3.905 million in the fourth quarter of 2016. A number of new fields continued to ramp up output, particularly in Brazil and Kazakhstan. Earnings from refining, marketing and chemicals also rose 20% to $2.5bn. "We saw notable improvements in upstream and chemicals, which benefited from improved operational performance and better market conditions," Mr van Beurden said. Investors welcomed the results, marking Shell's shares up 3% at the start of trading. Energy producers across the world are reaping the benefits of higher oil prices, which have strongly increased their revenues and profits. Crude has recovered thanks to the Opec oil producers' cartel adhering to a production cut agreed late last year. However, the oil price has eased back recently. |
| **Reference** | Royal Dutch Shell's profits surged in the first three months of the year on the back of rising oil prices. |
| **GPT2-small entire decoder summary** | Royal Dutch Shell has reported a sharp fall in profits as a result of lower oil prices and weaker demand. It warned about its threat to cut more than 10,000 jobs. |
| **GPT2-medium entire decoder summary** | Oil giant Royal Dutch Shell has reported a sharp rise in profits thanks in part to a recovery in the value of its stake in the BG Group. |
| **GPT2-large entire decoder summary** | Royal Dutch Shell has reported a drop in profits just a year after setting aside $7bn of savings to cover the cost of its 2013 emissions scandal. |
| **GPT2-small xa only summary** | Oil and gas giant BP has reported a rise in pre-tax profits for the first time in more than a decade. Analysts expect the company to report a rise in profits for the first time in more than a decade. |
| **GPT2-medium xa only summary** | Oil giant Royal Dutch Shell has reported a sharp fall in first-quarter profits, compounding recent troubles for the energy giant. |
| **GPT2-large xa only summary** | Royal Dutch Shell has reported a drop in annual earnings, after being hit by a fall in the price of oil and gas. |
| **GPT2-small whole model summary** | A man has been jailed for life for murdering his ex-partner in a "brutal" and "brutal" attack. The victim's family have said they are "devastated" by his "brutal" actions. |

Table 8: Another example of document, reference summary and PIXELSum generated summaries from XSum test set. Factual inconsistencies, particularly, where the model introduced explicitly false new named entities or descriptors, between documents and summaries are denoted in red. Correctly identified named entities etc. from the original document (highlighted in yellow) in generated summaries are highlighted in green.

*near Stonehenge*" or the alleged price of hoard of coins "*may be worth more than 31m*". The last document in Table 8 speaks of Royal Dutch Shell's increasing profits in first quarter of the year, largely due to rise in oil prices. All except one (GPT2-medium entire decoder) summaries mention a drop in profits over various spans of time, e.g. "*drop in annual earnings*" or "*a rise in [...] profits for the first time in more than a decade*". The summaries mention false reasons for the alleged fall in profits, e.g. "*[rise] thanks to a recovery in the value of its stake in BG Group*" or "*fall in the price of oil and gas*". The generated excerpts also introduced false consequences of the fall in profits, e.g. "*threat to cut more than 10,000 jobs*" or "*compounding recent troubles for the energy giant*". The model also introduced fake information as a reference to point in time "*just a year after setting aside $7bn [...] to cover its 2013 emissions scandal*"

We have intentionally left out the summaries generated by the PIXELSum variant where the whole model was fine-tuned from this analysis. The model generated a similar summary across all three examples, suggesting it has not been able to learn the task during training and instead memorised some data it had exposure to. This further reinforces the observations we made based on quantitative performance of this model in the previous section. Namely, that fine-tuning the whole network does not improve performance. This model will be omitted in further qualitative analysis.

### 5.3.3 Fluency

Drawing again from DUC 2005 guidelines [63], a fluently written summary "should have no formatting problems, capitalization errors or obviously ungrammatical sentences (e.g., fragments, missing components) that make the text difficult to read."

Overall, the models produce fluent summaries. Formatting the summaries falls beyond our scope, as the only post-processing step applied is decoding the tokenized outputs from the decoder. The models capitalise named entities, do not introduce capitalisation otherwise and add commas where needed. Apart from a few verb form mismatches (have instead of has) in examples from Table 7, there are no other outstanding grammatical errors. The sentences are written in a clear manner and are easy to read, thus we consider PIXELSum capable of producing fluent summaries.

### 5.3.4 Relevance

Lastly, we deem a summary relevant if it contains only important information from the source document and conveys the core ideas of the document clearly. The examples from Tables 6, 7, 8 show models fail to produce relevant summaries. While clearly the models are somehow capable of identifying the topic or general ideas of the documents, f.ex the PIXELSum generated summaries in Table 7 all talk about a hoard of coins being found, they fail to identify the information and relevant links between them. The generated summaries do not reflect the content of the document in a satisfactory manner.

# Chapter 6

# Discussion

The experimental results gave rise to a number of observations, useful in addressing our research objective, which is to determine to best leverage Pixel Encoder of Language, PIXEL, for text summarisation and shed light on the right direction to use it for text generation. We discuss the observation in order on how they build upon each other and can offer an explanation to each other and lastly suggest what matters most in PIXELSum and why.

## ROUGE and BERTScore results divergence

Based on the results in Table 4 two contradicting trends emerge. In terms of ROUGE score, the whole decoder models perform better, outperforming all cross-attention only models. The best performing model overall, with respect to ROUGE, is *PIXEL + GPT-2 medium whole decoder*. In terms of BERTScore, an opposite trend emerges: the cross-attention only group performs better than whole decoder, outperforming all whole decoder models. The best PIXELSum variant in terms of this metric is *PIXEL + GPT-2 medium cross-attention only*.

ROUGE score is focused on whether a word in a sentence in the exact form matches between reference and generated text, while BERTScore operates on word embeddings and calculates semantic similarity of the reference and generated text, which is a more lenient metric in abstractive summarisation as it gives credit to sentences

that convey the same meaning using different words.

Based on the aforementioned lack of agreement, we hypothesize that the whole decoder model family overfits and learns to repeat keywords in the text instead of generating excerpts. This could offer an explanation to why words would overlap but semantic quality (in terms of BERTScore) decreases.

We compare the loss functions of the best performing cross-attention only and whole decoder model, as depicted in Figure 8. The evaluation loss of all whole decoder PIXELSum briefly decreases until the 500th step, then starts increasing while the training loss continues to converge, which suggests the model stopped learning meaningful patterns and started memorising i.e. is overfitting. The case is different for cross-attention only PIXELSum, where the evaluation loss keeps decreasing steadily, suggesting the model is improving. This further confirms our hypothesis, that overfitting occurs whilst training whole decoder models. This in turn suggests, that the decoder does not know how to read encoders outputs, thus starts memorising. The difference between these models lays in the number of trainable parameters (see Table 2), and experimental evidence points to that whole decoder models are overparameterised for the summarisation task on the XSum dataset.

## Fine-tuning the whole network

Through the comparison of training loss functions previously mentioned in Figure 7 and further reiterated by the training loss and evaluation loss respectively in Figure 9 we obtain strong indication that the model did not have sufficient capabilities to understand the task on hand and learn from the data. The training loss stopped decreasing before the model achieved performance comparable to it's encoder-frozen counterparts, while evaluation loss is fluctuating. This is another strong indication that the model had too many trainable parameters for this task and memorised the training set instead of learning, hence started overfitting.

This conclusion is further consolidated by the model's generated outputs. As can be seen from Tables 6, 7, 8 the *GPT2-small whole model* summary produces almost the

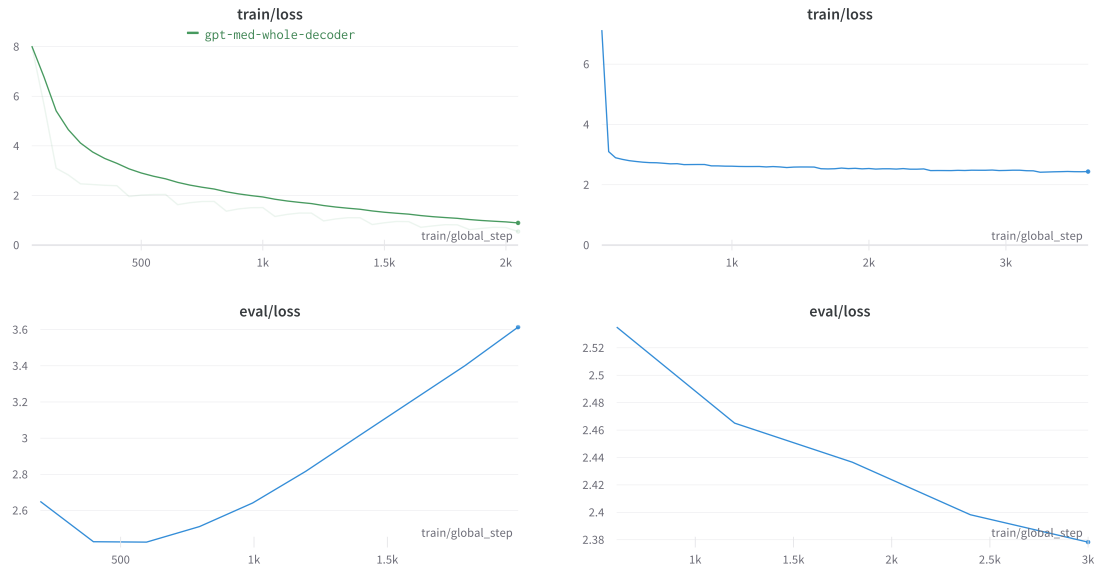



Figure 8: Loss functions of the best performing whole decoder model (left column, training on top, evaluation on bottom) and cross-attention only model (to the right, training on top, evaluation on bottom).

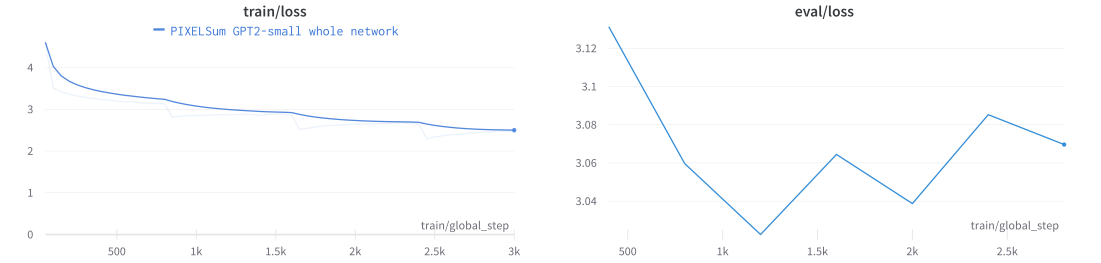



Figure 9: Training (to the left) and evaluation (to the right) losses of the PIXELSum GPT-2 small whole network variant.

same summary regardless of the input document. This suggests the model picked up on one of the training examples and memorised it rather than keep on learning from the data.

## Qualitative examples of good and bad behaviour in summaries

The examples from Tables 6, 7, 8 suggest that the models are somehow capable of identifying important information and general ideas of the documents. The generated text is fluent. For example, the PIXELSum generated summaries in Table 7 all talk about a hoard of coins being found which is the overarching idea of the article.

Moreover, the summaries in Table 8 suggest that models are capable of identifying key entities from the articles, mentioning *Royal Dutch Shell* as the object of the sentence.

However, the generated summaries display a lot of undesired behaviours as well. Most importantly, the summaries are not factually consistent or relevant. These two qualities can be treated together, as a text containing false information cannot be relevant to the document. This is well depicted in the example from Table 6: the object of the text is correctly identified to be a dog, but the details are false, e.g. name, size of the dog or even material it's made of. The summaries describe what happens to the dog, in no case however is it related to the document. The document mentions Japan and models pick that up, however mentioned cities are completely irrelevant and unrelated to the truth, with one city *"Haru Kagawa"* non-existent in reality. Overall, the reader would not be able to infer anything about the document from these summaries which renders them irrelevant.

We observe the models introduce new information to the summaries, which are not stated in the document, yet are somehow related to the entities in it. For example in Table 8: *"Oil giant Royal Dutch Shell has reported a sharp rise in profits thanks in part to a recovery in the value of its stake in the BG Group"* the summary quotes *BG Group* as responsible for rise in profits, which is false and not mentioned in the article, yet is indeed a company Shell acquired in 2015 [1].

This contextual information being injected into the summaries could suggest that the decoder attends uniformly to the whole input, i.e. does not regard the encoder's output as important in predicting the next token. We proceed by analysing the attention masks of an example document in Figures 10 and 11 from the best performing model. The reader can find more examples in Appendix 1. The attention masks further confirm our hypothesis, that the decoder does not know how to read encoder's outputs and attends almost uniformly to all the tokens in the document. This means the decoder relies on previous tokens in predicting the next one more heavily than the document itself. This could offer an explanation to why the outputs

[1] https://en.wikipedia.org/wiki/BG_Group

are largely irrelevant and introduce new information to the text. This is a strong indication, that cross attention failed to learn a conductive mapping between the encoder and the decoder.

## Our model vs. state-of-the-art

Table 5 gives a comparison of our best PIXELSum variant's results with state-of-the-art models in terms of ROUGE score, we omit BERTScore in this analysis since most of the models were not evaluated using this metric. Our model falls behind state-of-the-art by a substantial margin and there is certainly room for improvement. However, the aim of our research was to explore how to best use PIXEL for text summarisation, and not to create a model that beats state of the art in the aforementioned task and we have gathered a considerable amount of observations pointing towards the steps to improve PIXEL-based summarisation models' performance in the future. We proceed by discussing them in the next section.

## What matters in PIXELSum and why?

In our results and discussion, trends have emerged that suggest what is the most significant in PIXEL-based summarisation models.

Our model falls behind the state-of-the-art by a significant margin. As discussed in the Related Work, Chapter 4, the main difference between our model and the other models is that we concatenate a pretrained encoder and decoder, but do not pretrain the resulting model again. We hypothesize, that pretraining PIXELSum as a whole on a more general, summarisation-related task, alike PEGASUS [28] could improve the performance substantially. For example, the model could be pre-trained to predict the X last % of an L-length document from the first L-X % on a large dataset, then fine-tuned on a specific summarisation task.

Amongst tested decoders GPT-2 medium architecture yields the best results for the task. Empirical evidence suggests that models where the whole decoder or whole model are trained are overparameterised for the summarisation task, without

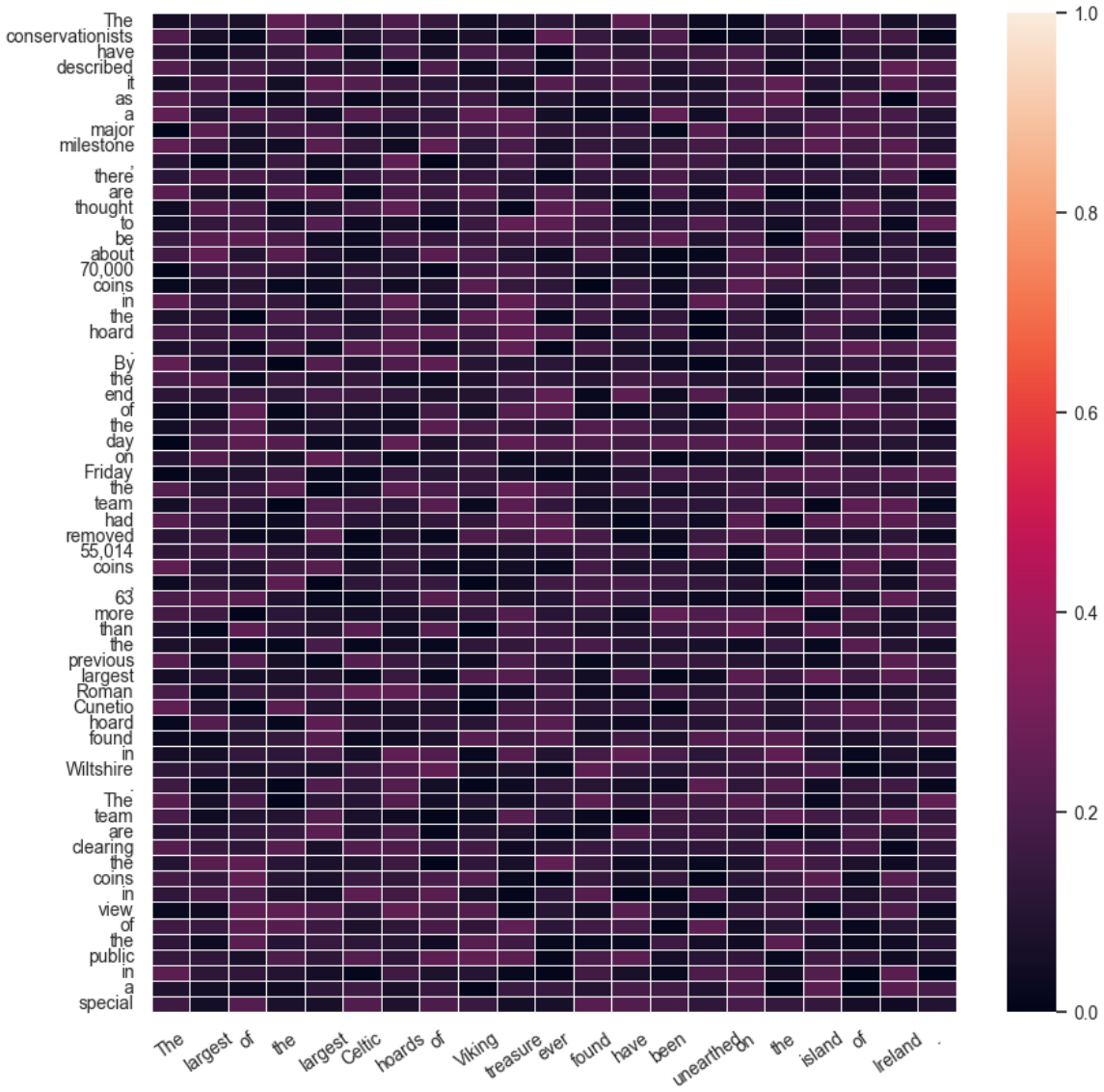



Figure 10: Part 1: Normalised attention heatmap for example document and summary from Table 7

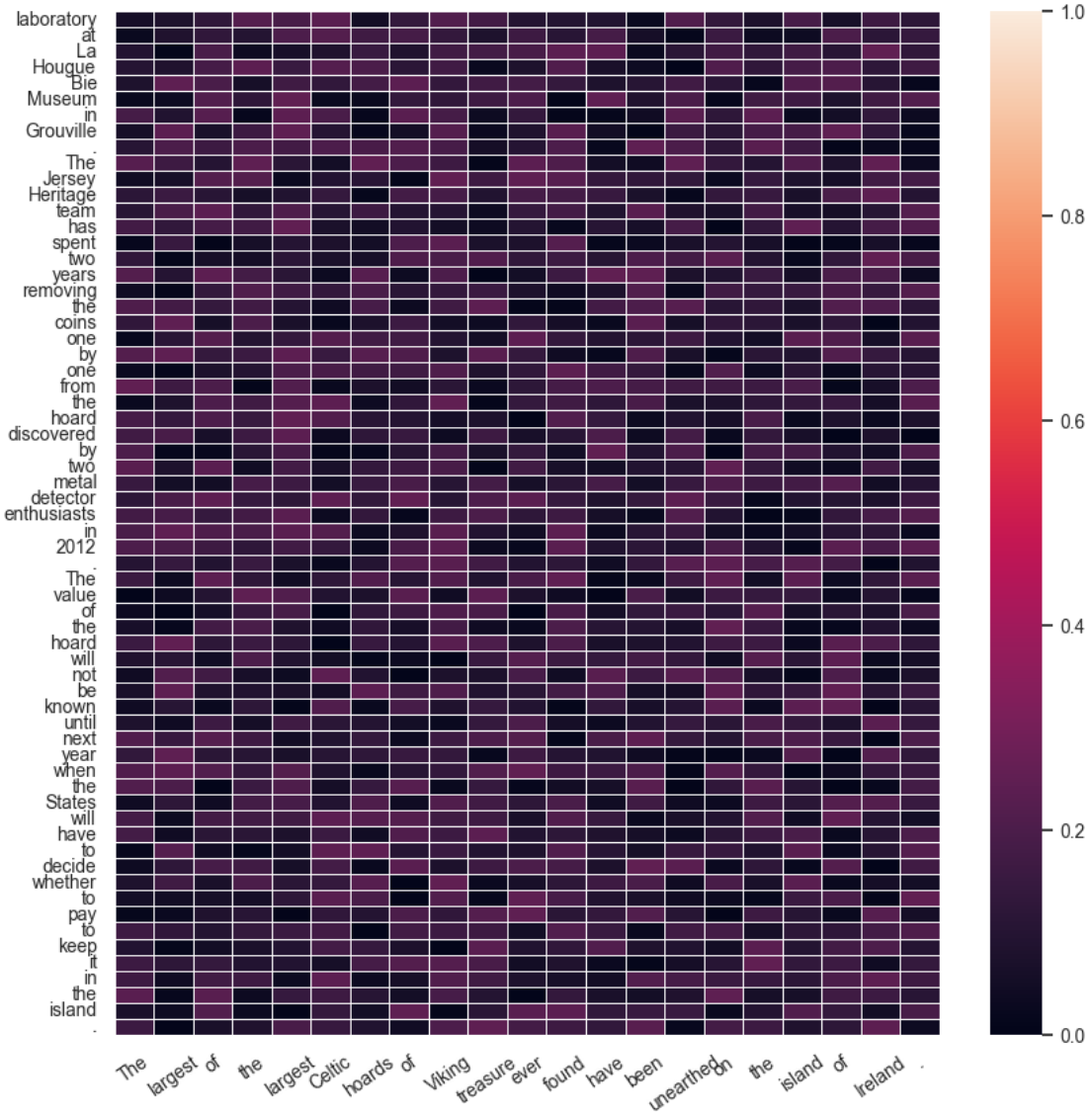



Figure 11: Part 2: Normalised attention heatmap for example document and summary from Table 7.

pretraining, on the XSum dataset. The cross-attention only models are the only variation, where the evaluation loss consistently keeps decreasing, which suggests that the number of parameters is more suitable for this particular task and prevents overfitting, however the performance of the model falls far behind state-of-the-art. Taking a look back at Table 5, we hypothesize that a mid-size range of decoder parameters between 90M to 250M yields the best results and training cross-attention only should be prioritised over training the whole decoder on the fine-tuning task. This both reduces the cost of model training and preserve generalisation capabilities [54].

Lastly, the decoder does not know how to read the encoder's outputs and attends almost uniformly to the whole input. We suggest pre-training the model on a more general, summarisation-related task in the way outlined previously could help with this. This approach has proved successful in the state-of-the-art models (see Table 3).

# Chapter 7

# Conclusions

This paper has investigated how to leverage the Pixel Encoder of Language, PIXEL, for text summarisation. We proposed PIXELSum, a sequence-to-sequence summarisation model comprising of PIXEL encoder and GPT2 autoregressive decoder connected with cross-attention. We trained multiple variants of the model on an abstractive summarisation dataset, XSum, with different variants of the decoder, and various subparts of the model frozen.

Our best model variant achieved ROUGE1, ROUGE2, ROUGE-L scores of 16.23, 4.50, 12.32 respectively and BERTScore precision of 0.5887. The models produced fluent, but factually inconsistent and irrelevant summaries. The evidence from the study suggests, that models where the whole decoder, or whole model, were fine-tuned are over-parameterised and over-fit to the data. On the other hand, models where only cross-attention only was fine-tuned learn from the data but not to an extent that yields satisfactory performance. We observe the decoder does not know how to attend to encoder outputs. Taken together, these results would seem to indicate PIXELSum should be pre-trained on a more general task, before fine-tuning for abstractive summarisation as the state-of-the-art models have.

The strength of our contribution lies in giving preliminary insight into how to best leverage PIXEL for summarisation, or more broadly, generation tasks. We are aware PIXELSum falls far behing state-of-the-art's performance, however, we believe our

work could be a starting point to creating successful pixel-based generative models. We have suggested the improvements that should be taken, based on emprical observations, in attempt to bridge the gap between PIXELSum and state-of-the-art. We discuss them further in the future research directions section.

# Future research directions

## Immediate

1. Research efforts should be directed towards investigating how pre-training the model on a general, summarisation-alike task, then fine-tuning on a paricular abstractive summarisation dataset influences performance;

2. Further studies should investigate how using different autoregressive decoders, such as e.g. OPT [65] influences performance.

## Future

1. Since the model renders text as images, studies could investigate whether visually enunciating important parts of the document, e.g. underlining named entities, affect quantitative and qualitative performance of the model;

2. PIXEL has shown great ability to transfer to scripts not found in pre-training data [7]. Future research could investigate how to create a multi-lingual pixel-based summarisation model;

3. Research efforts should be directed towards creating an autoregressive PIXEL decoder to reconstruct pixel patches one at a time. As far as we are concerned, work in this direction is underway from the research group that created PIXEL [7].

# Appendix 1

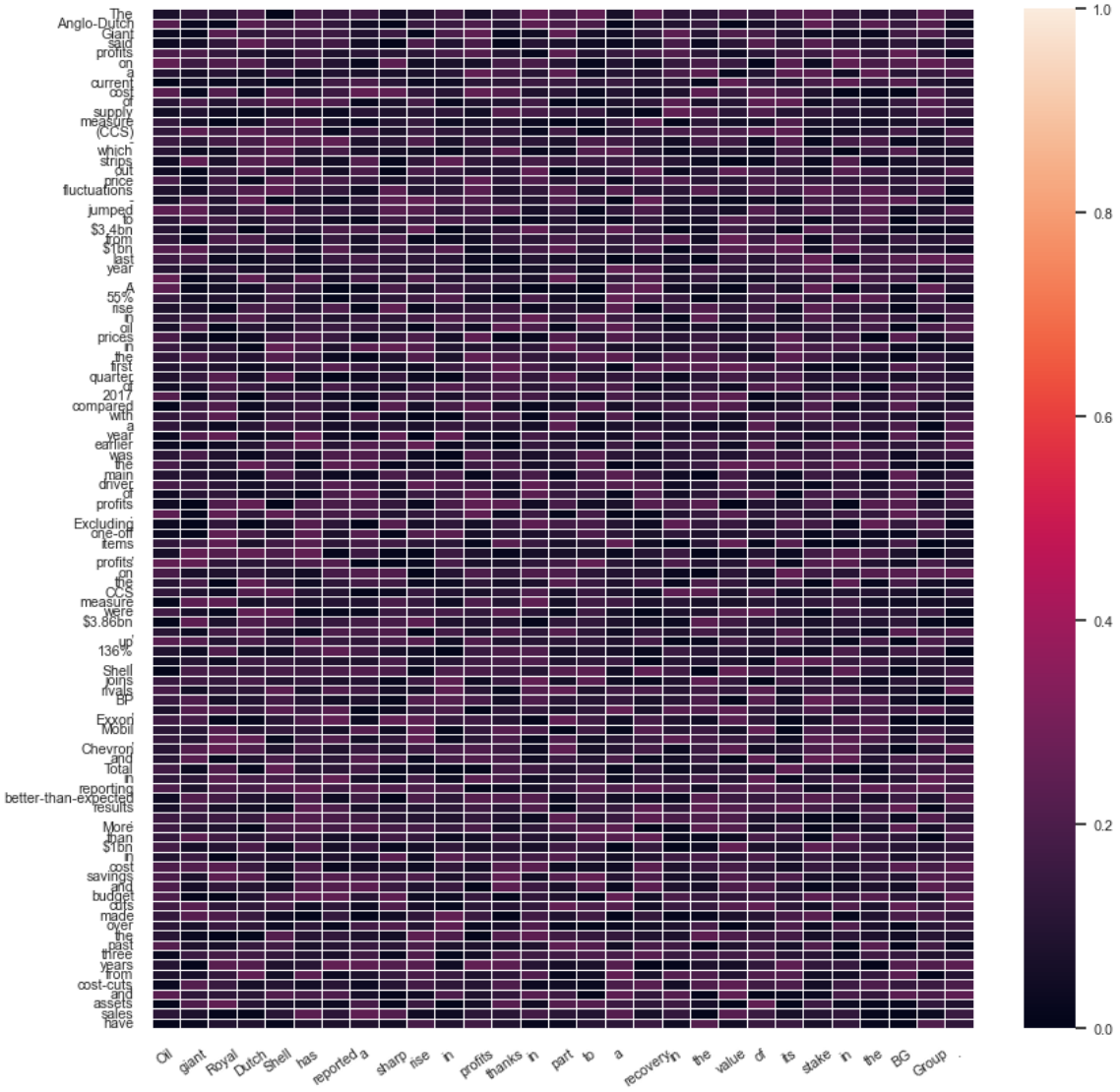



Figure 12: Part 1: Normalised attention heatmap for document example in Table 8.

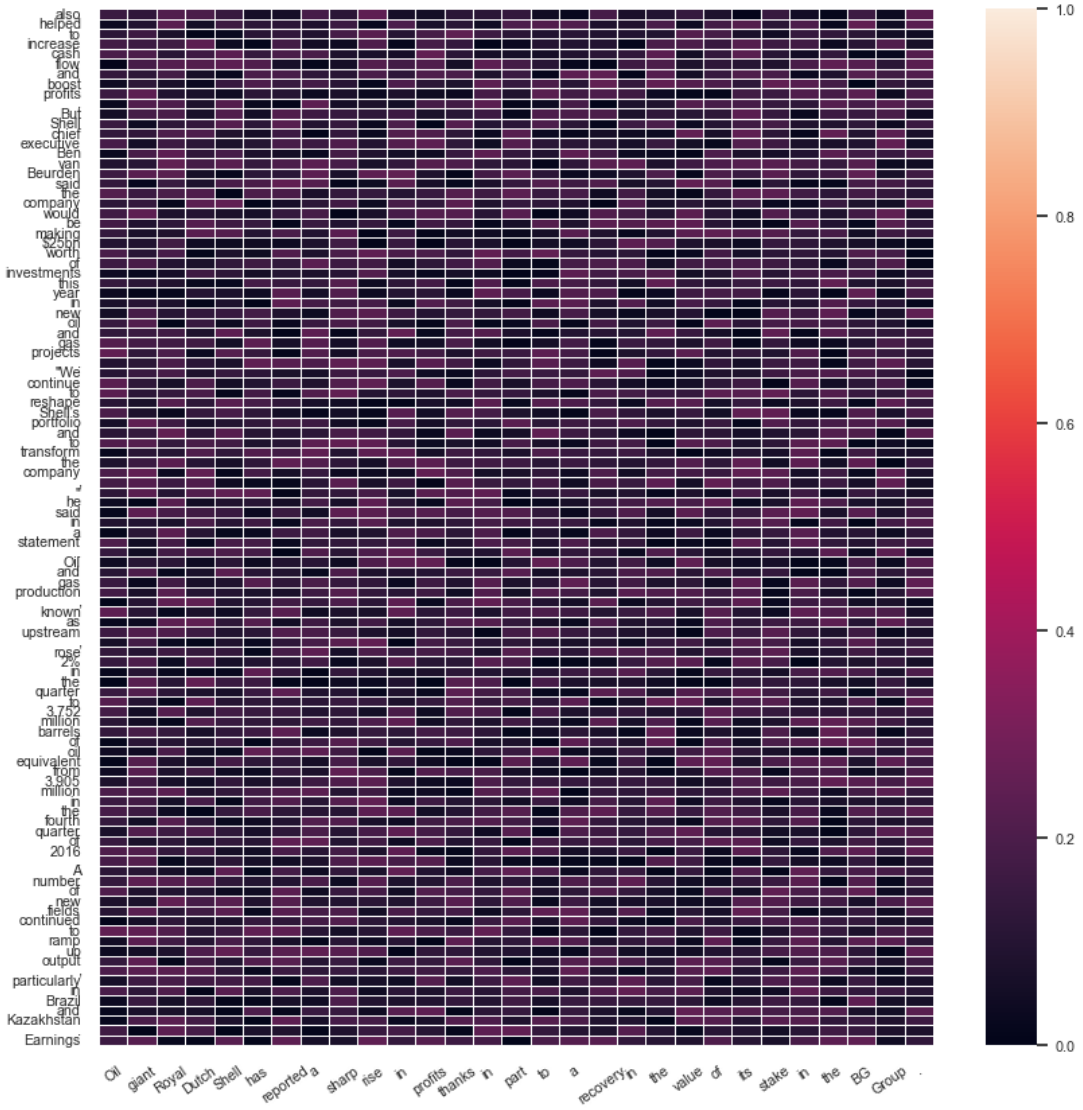



Figure 13: Part 2: Attention heatmap for document example in Table 8.

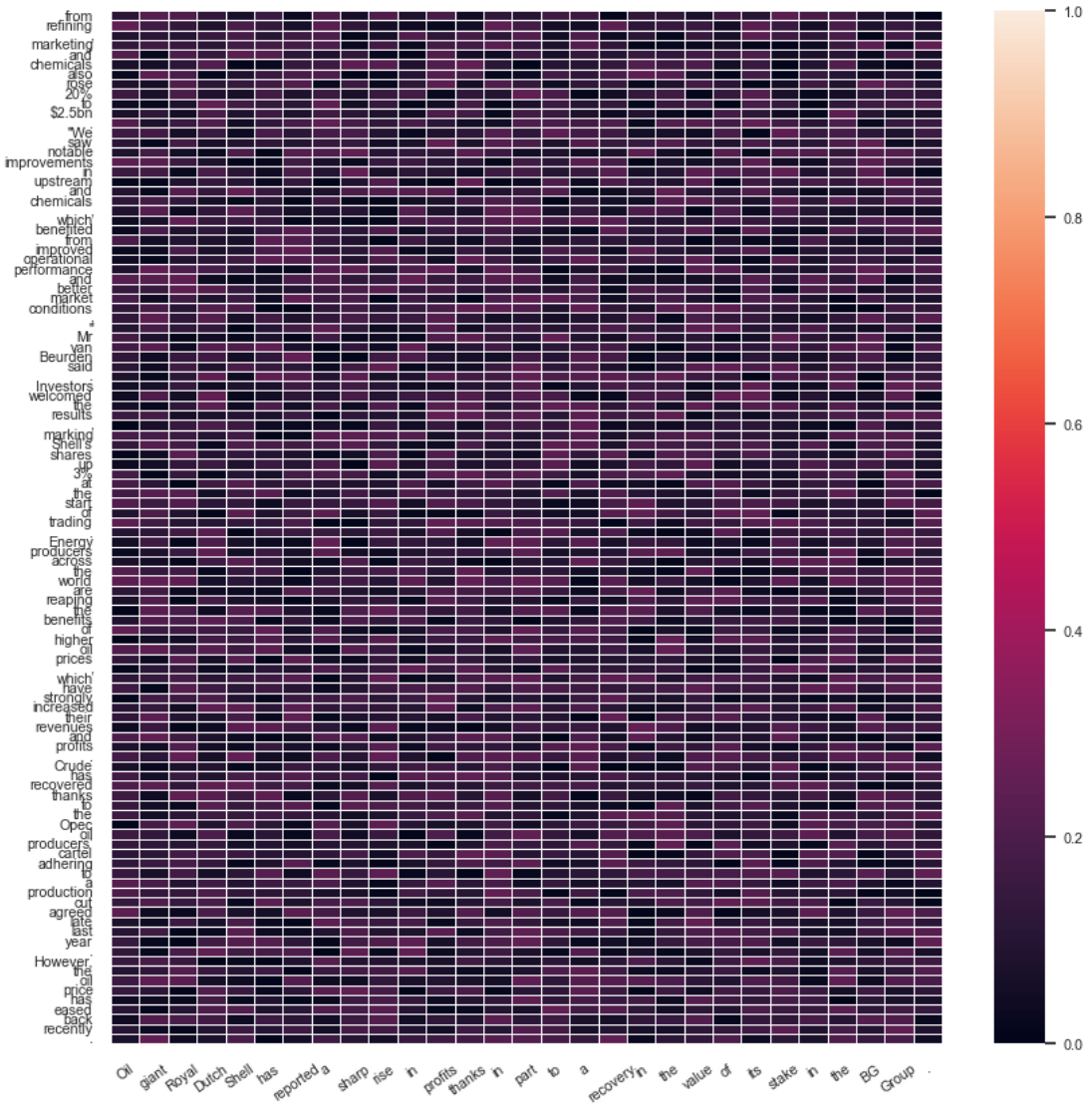



Figure 14: Part 3: Attention heatmap for document example in Table 8.

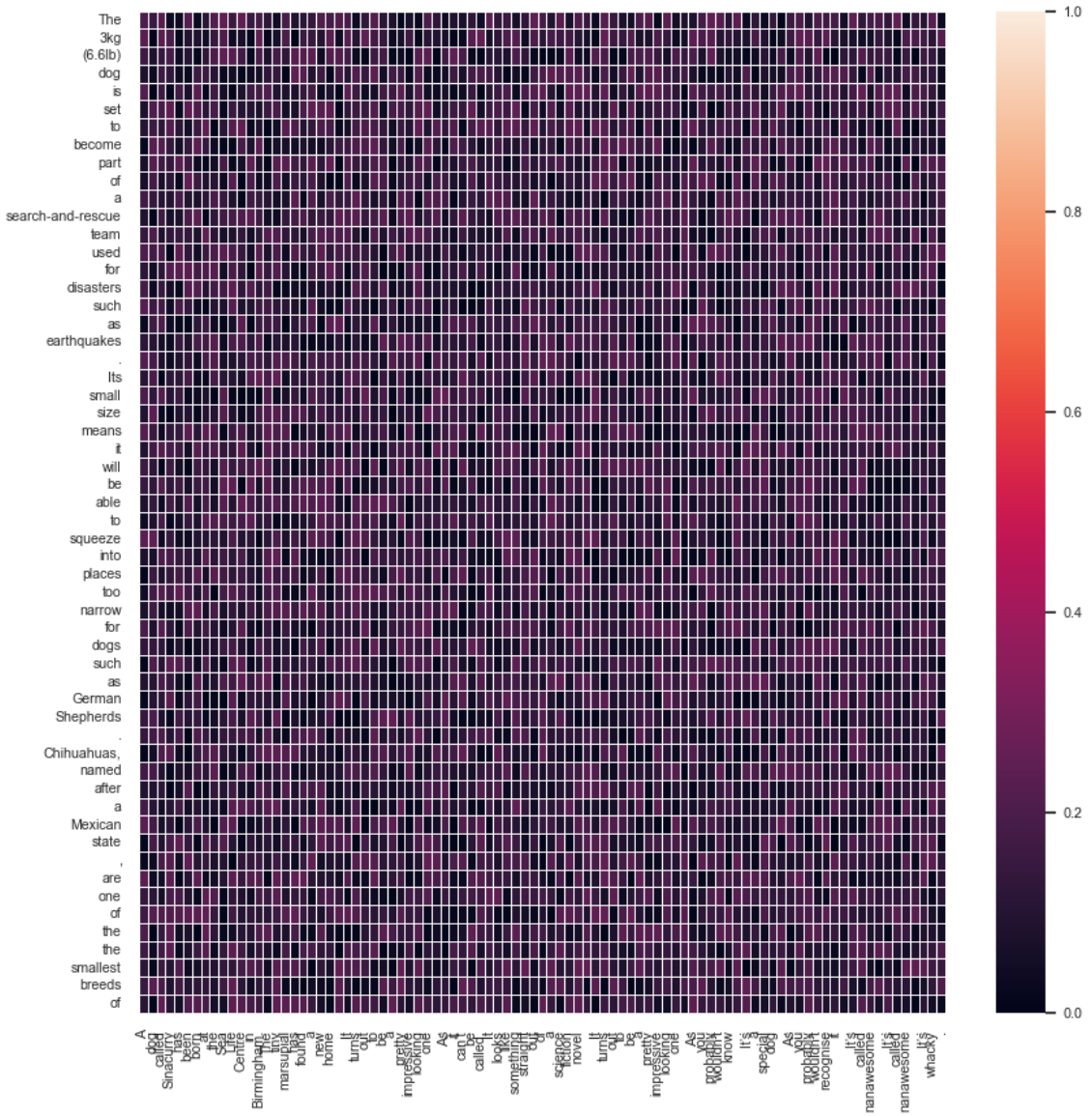



Figure 15: Part 3: Attention heatmap for document example in Table 6.

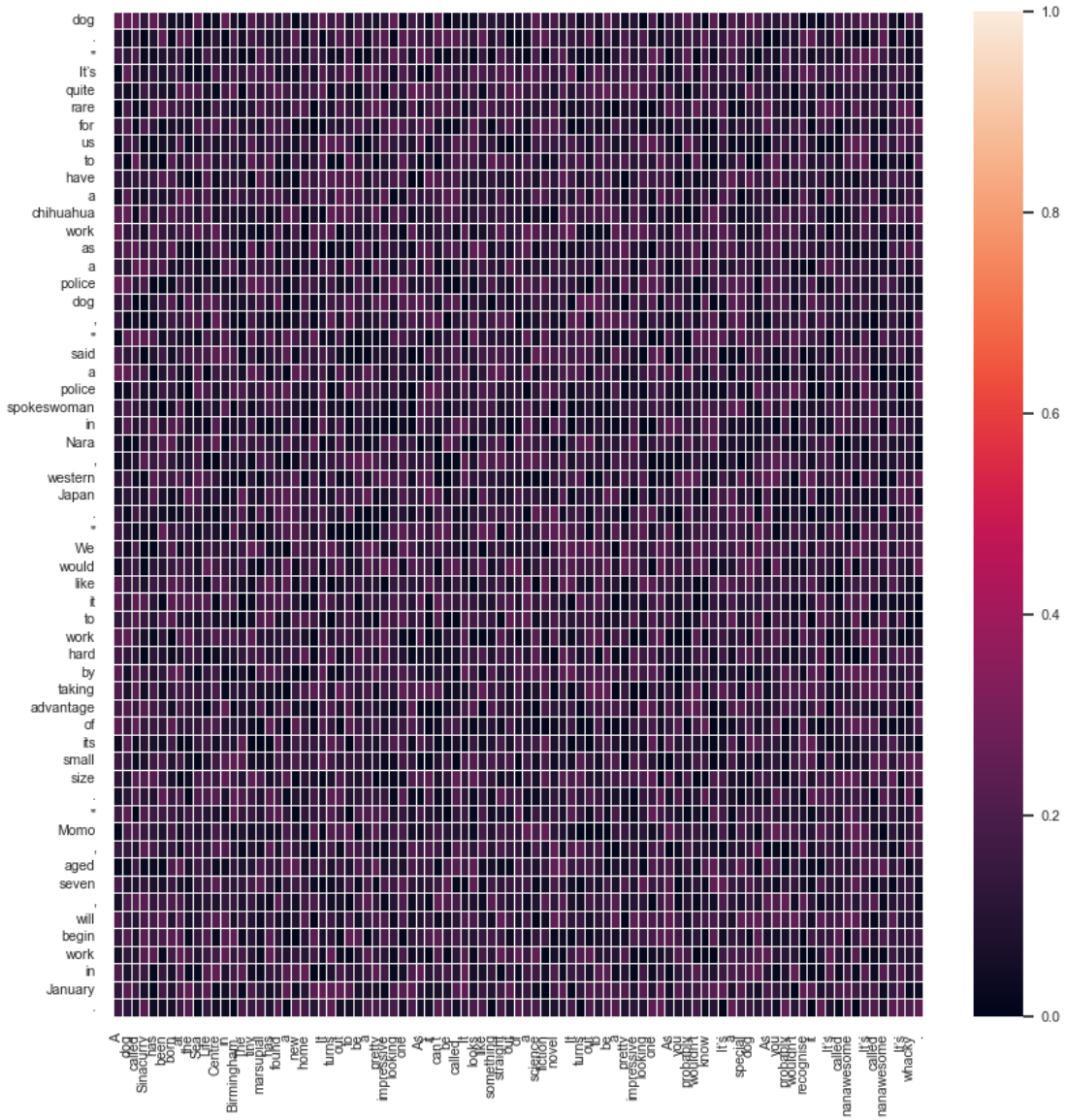



Figure 16: Part 3: Attention heatmap for document example in Table 6.

# Bibliography


[1] Makovhololo, P., Taylor, F. & Iyamu, T. Diffusion of abstractive summarisation to improve ease of use and usefulness. In *2018 Open Innovations Conference (OI)*, 257–263 (IEEE, 2018).

[2] Cachola, I., Lo, K., Cohan, A. & Weld, D. S. Tldr: Extreme summarization of scientific documents. *arXiv preprint arXiv:2004.15011* (2020).

[3] Vella, G. *Automatic summarisation of legal documents*. B.S. thesis, University of Malta (2010).

[4] Grover, C., Hachey, B., Hughson, I. & Korycinski, C. Automatic summarisation of legal documents. In *Proceedings of the 9th international conference on Artificial intelligence and law*, 243–251 (2003).

[5] Summerscales, R. L., Argamon, S., Bai, S., Hupert, J. & Schwartz, A. Automatic summarization of results from clinical trials. In *2011 IEEE International Conference on Bioinformatics and Biomedicine*, 372–377 (IEEE, 2011).

[6] Pretrained models - huggingface. URL `https://huggingface.co/transformers/v2.2.0/pretrained_models.html`.

[7] Rust, P. *et al.* Language modelling with pixels (2022). `2207.06991`.

[8] El-Kassas, W., Salama, C., Rafea, A. & Mohamed, H. Automatic text summarization: A comprehensive survey. *Expert Systems with Applications* **165**, 113679 (2020).

[9] Kryscinski, W., McCann, B., Xiong, C. & Socher, R. Evaluating the factual consistency of abstractive text summarization. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 9332–9346 (Association for Computational Linguistics, Online, 2020). URL https://aclanthology.org/2020.emnlp-main.750.

[10] Sutskever, I., Vinyals, O. & Le, Q. V. Sequence to sequence learning with neural networks (2014). URL https://arxiv.org/abs/1409.3215.

[11] Wolf, T. *et al.* Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 38–45 (Association for Computational Linguistics, Online, 2020). URL https://aclanthology.org/2020.emnlp-demos.6.

[12] Vaswani, A. *et al.* Attention is all you need. *CoRR* **abs/1706.03762** (2017). URL http://arxiv.org/abs/1706.03762. 1706.03762.

[13] Maucher, J. Machine learning book. URL https://hannibunny.github.io/mlbook/intro.html.

[14] Bahdanau, D., Cho, K. & Bengio, Y. Neural machine translation by jointly learning to align and translate (2014). URL https://arxiv.org/abs/1409.0473.

[15] Niu, Z., Zhong, G. & Yu, H. A review on the attention mechanism of deep learning. *Neurocomputing* **452**, 48–62 (2021).

[16] Weng, L. Attention? attention! *lilianweng.github.io* (2018). URL https://lilianweng.github.io/posts/2018-06-24-attention/.

[17] Tamura, Y. Multi-head attention mechanism: "queries", "keys", and "values," over and over again (2022). URL https://data-science-blog.com/blog/2021/04/07/multi-head-attention-mechanism/.

[18] Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[19] Radford, A., Narasimhan, K., Salimans, T., Sutskever, I. *et al.* Improving language understanding by generative pre-training (2018).

[20] Radford, A. *et al.* Language models are unsupervised multitask learners. *OpenAI blog* **1**, 9 (2019).

[21] Brown, T. B. *et al.* Language models are few-shot learners (2020). 2005.14165.

[22] Lewis, M. *et al.* Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension (2019). 1910.13461.

[23] Dosovitskiy, A. *et al.* An image is worth 16x16 words: Transformers for image recognition at scale (2021). 2010.11929.

[24] He, K. *et al.* Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 16000–16009 (2022).

[25] Widyassari, A. P. *et al.* Review of automatic text summarization techniques methods. *Journal of King Saud University - Computer and Information Sciences* **34**, 1029–1046 (2022). URL https://www.sciencedirect.com/science/article/pii/S1319157820303712.

[26] Moratanch, N. & Chitrakala, S. A survey on extractive text summarization. In *2017 International Conference on Computer, Communication and Signal Processing (ICCCSP)*, 1–6 (2017).

[27] Ramos, J. *et al.* Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, vol. 242, 29–48 (Citeseer, 2003).

[28] Zhang, J., Zhao, Y., Saleh, M. & Liu, P. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, 11328–11339 (PMLR, 2020).

[29] Shukla, A. *et al.* Legal case document summarization: Extractive and abstractive methods and their evaluation. *arXiv preprint arXiv:2210.07544* (2022).

[30] Yadav, D. *et al.* Qualitative analysis of text summarization techniques and its applications in health domain. *Computational Intelligence and Neuroscience* **2022** (2022).

[31] Sarker, I. H. Deep learning: a comprehensive overview on techniques, taxonomy, applications and research directions. *SN Computer Science* **2**, 420 (2021).

[32] Gupta, V. & Lehal, G. S. A survey of text summarization extractive techniques. *Journal of emerging technologies in web intelligence* **2**, 258–268 (2010).

[33] Papagiannopoulou, A. & Angeli, C. Social media text summarisation techniques and approaches: A literature review. In *Proceedings of the 2022 5th International Conference on Algorithms, Computing and Artificial Intelligence*, 1–6 (2022).

[34] Lin, H. & Ng, V. Abstractive summarization: A survey of the state of the art. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 9815–9822 (2019).

[35] Durmus, E., He, H. & Diab, M. Feqa: A question answering evaluation framework for faithfulness assessment in abstractive summarization. *arXiv preprint arXiv:2005.03754* (2020).

[36] Falke, T., Ribeiro, L. F., Utama, P. A., Dagan, I. & Gurevych, I. Ranking generated summaries by correctness: An interesting but challenging application for natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2214–2220 (2019).

[37] Liu, J. *Improving Text Summarization with Limited Resources*. Ph.D. thesis, Apple Inc (2020).

[38] Lloret, E., Plaza, L. & Aker, A. The challenging task of summary evaluation: an overview. *Language Resources and Evaluation* **52**, 101–148 (2018).

[39] Iskender, N., Polzehl, T. & Möller, S. Reliability of human evaluation for text summarization: Lessons learned and challenges ahead. In *Proceedings of the Workshop on Human Evaluation of NLP Systems (HumEval)*, 86–96 (2021).

[40] Zhang, T., Kishore, V., Wu, F., Weinberger, K. Q. & Artzi, Y. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675* (2019).

[41] Nallapati, R., Zhou, B., Gulcehre, C., Xiang, B. *et al.* Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023* (2016).

[42] Zhang, X. *et al.* Momentum calibration for text generation (2022). 2212.04257.

[43] Grusky, M., Naaman, M. & Artzi, Y. Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. *arXiv preprint arXiv:1804.11283* (2018).

[44] Koupaee, M. & Wang, W. Y. Wikihow: A large scale text summarization dataset. *arXiv preprint arXiv:1810.09305* (2018).

[45] Savelieva, A., Au-Yeung, B. & Ramani, V. Abstractive summarization of spoken and written instructions with bert. *arXiv preprint arXiv:2008.09676* (2020).

[46] Sandhaus, E. The New York Times Annotated Corpus (2008). URL https://hdl.handle.net/11272.1/AB2/GZC6PL.

[47] Graff, D., Kong, J., Chen, K. & Maeda, K. English gigaword. *Linguistic Data Consortium, Philadelphia* **4**, 34 (2003).

[48] Kedia, A., Chinthakindi, S. C. & Ryu, W. Beyond reptile: Meta-learned dot-product maximization between gradients for improved single-task regularization. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, 407–420 (2021).

[49] Narayan, S., Cohen, S. B. & Lapata, M. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. *arXiv preprint arXiv:1808.08745* (2018).

[50] Zhao, Y. *et al.* Calibrating sequence likelihood improves conditional language generation. *arXiv preprint arXiv:2210.00045* (2022).

[51] Völske, M., Potthast, M., Syed, S. & Stein, B. TL;DR: Mining Reddit to learn automatic summarization. In *Proceedings of the Workshop on New Frontiers in Summarization*, 59–63 (Association for Computational Linguistics, Copenhagen, Denmark, 2017). URL https://aclanthology.org/W17-4508.

[52] Rush, A. M., Chopra, S. & Weston, J. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685* (2015).

[53] Rothe, S., Narayan, S. & Severyn, A. Leveraging pre-trained checkpoints for sequence generation tasks. *Transactions of the Association for Computational Linguistics* **8**, 264–280 (2020). URL https://doi.org/10.1162%2Ftacl_a_00313.

[54] Ramos, R., Martins, B., Elliott, D. & Kementchedjhieva, Y. Smallcap: Lightweight image captioning prompted with retrieval augmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2840–2849 (2023).

[55] Raffel, C. *et al.* Exploring the limits of transfer learning with a unified text-to-text transformer (2020). 1910.10683.

[56] Liu, Y., Liu, P., Radev, D. & Neubig, G. Brio: Bringing order to abstractive summarization (2022). URL https://arxiv.org/abs/2203.16804.

[57] Dong, L. *et al.* Unified language model pre-training for natural language understanding and generation (2019). 1905.03197.

[58] Wolf, T. *et al.* Huggingface's transformers: State-of-the-art natural language processing (2020). 1910.03771.

[59] Van Rossum, G. & Drake, F. L. *Python 3 Reference Manual* (CreateSpace, Scotts Valley, CA, 2009).

[60] Paszke, A. *et al.* Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, 8024–8035 (Curran Associates, Inc., 2019). URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[61] Loshchilov, I. & Hutter, F. Decoupled weight decay regularization (2019). 1711.05101.

[62] Dong, L. *et al.* Unified language model pre-training for natural language understanding and generation. *CoRR* **abs/1905.03197** (2019). URL http://arxiv.org/abs/1905.03197. 1905.03197.

[63] Dang, H. T. Overview of duc 2005. In *Proceedings of the document understanding conference*, vol. 2005, 1–12 (Citeseer, 2005).

[64] Fabbri, A. R. *et al.* SummEval: Re-evaluating Summarization Evaluation. *Transactions of the Association for Computational Linguistics* **9**, 391–409 (2021). URL https://doi.org/10.1162/tacl_a_00373. https://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00373/1923949/tacl_a_00373.pdf.

[65] Zhang, S. *et al.* Opt: Open pre-trained transformer language models (2022). 2205.01068.