Have You Tried Turning It Off and On Again? Pressing reset with a novel framework for the use of deep learning chatbots in education

Submitted August 2023, in partial fulfilment of
the conditions for the award of the degree **MSc Data Science.**

Joel Pointon
10286413

Supervised by Tissa Chandesa

School of Computer Science
University of Nottingham

I hereby declare that this dissertation is all my own work, except as indicated in the text:

Signature

Date 11/08/2023

Abstract

The recently released ChatGPT stunned users with its human-sounding content and ability to assist with a wide range of tasks. However, Large Language Models (LLMs) such as GPT generate content in a probabilistic manner and so, at present, cannot guarantee factual answers or provide sources. This paper presents a framework to achieve this, by using a document store to pass relevant information to a chatbot which has been fine-tuned for question-answering. The document store is generated automatically from chosen documents which are split into chunks of 800 tokens.

This paper takes a novel approach to creating a question-answering framework. By adapting existing summarisation fine-tuning techniques, and enhancing Google's long-form Natural Questions dataset using GPT-3.5 Turbo, a model is fine-tuned to create a generative (and not extractive) question-answering chatbot. Further, this paper contributes a long-form, generative answer variant of the Natural Questions dataset to the literature.

The proposed framework and provided model can, with future development, be used in countless applications. The long-form, closed-domain question-answering chatbot created by this paper can be used to answer questions from any domain, provided that context is given from a document store. Therefore, future adaptations to complete the model could enable it to be used in production, to benefit academics and students by streamlining Q&A processes. Other applications such as customer service, legal advice, and other applications are also possible, though not explored.

The model achieves excellent results on the validation set: ROUGE scores of between 43 and 45, a BLEU score of 34.2, and a METEOR score of 0.405. Validation answers attain an average cosine similarity of 0.763, with a 10.4% false negative rate and 30.2% false positive rate.

All code can be found in the following GitHub repository: `https://github.com/pointonjoel/MSc-Diss`, and the proposed model can be found in the following HuggingFace repository: `https://huggingface.co/psxjp5/mt5-small`.

Keywords: LLM; Natural Language Processing (NLP); Generative Pre-trained Transformer (GPT); Masked Language Modelling (MLM); Fine-Tuning

Acknowledgements

To my beautiful fiancée - the soon-to-be Lily Pointon. You have been my biggest fan and supported me this summer as we prepare for our adventure together! Thank you for listening to me, letting me complain about why things aren't working, and learning enough to write your own dissertation on this... I can't wait for the next stage of life together!

My unreserved gratitude and thanks go to the spirited and supportive Tissa Chandesa. Your supervision sessions and guidance have been very much appreciated - they inspired this project and helped me to overcome various obstacles. Thank you for challenging me and being a very attentive supervisor.

I also owe Irfan Yaqub many thanks for his expertise and availability to help with debugging and hyperparameter-tuning at various hours of the day. Your assistance has been very much appreciated and I wish you all the best with your PhD.

A huge thanks to my future in-laws, Richard and Julianne, for graciously allowing me to check the training progress of my models during mealtimes, and enabling me to work as much as possible during the time I spent in Canterbury. Thanks for welcoming me into your family!

My final thanks go to my wonderful parents, Mum and Dad - your constant support, encouragement and love have helped me get to not only get onto this Master's course, but to thrive on it. Thanks for all your help with planning the wedding and finding a flat, I can't wait to celebrate with you - I love you both!

Contents

List of Tables

List of Figures

Acronyms

AI Artificial Intelligence. vii, 1, 4, 6, 20, 21, 33, 35

ANN Artificial Neural Network. iii, 6, 7, 8

API Application Programming Interface. 20, 21, 23, 29, 34, 36

BART Bidirectional and Auto-Regressive Transformer. 13, 14, 17, 33, 36

BERT Bidirectional Encoder Representations from Transformers. 5, 12, 13, 14

BLEU Bilingual Evaluation Understudy. i, 25, 30, 31

CBOW Continuous Bag-of-Words. 5

CPU central Processing Unit. 18

FAQ Frequently Asked Question. 2, 33

GloVe Global Vectors for Word Representation. 5

GPT Generative Pre-trained Transformer. i, iv, v, vi, vii, 13, 14, 15, 17, 18, 19, 20, 21, 23, 26, 28, 29, 30, 32, 33, 34, 35, 46

GPU Graphics Processing Unit. 12, 18, 33, 35, 36

GRU Gated Recurrent Unit. iv, vii, 11, 12

HTML HyperText Markup Language. 22

LLM Large Language Model. i, 1, 24, 36

LSTM Long Short-Term Memory. iv, vii, 10, 11, 12

METEOR Metric for Evaluation of Translation with Explicit ORdering. i, 25, 30, 31

MIT Massachusetts Institute of Technology. 6

MLM Masked Language Modelling. i, v, vi, vii, 15, 17, 19, 26, 27, 36, 46

NaN Not a Number. 10

NLP Natural Language Processing. i, 4, 6, 12

PCA Principal component analysis. 27, 29

PDF Portable Document Format. 17, 19, 20

Q&A Question and Answer. i, 2, 3

RAM Random-Access Memory. 18, 19, 22

ReLU Rectified Linear Unit. 7

RGB red, green and blue. vii, 28, 29, 30, 32

RNN Recurrent Neural Network. iii, iv, vii, 9, 10, 11, 12

ROUGE Recall-Oriented Understudy for Gisting Evaluation. i, 25, 30, 31

SEAT Sentence Encoder Association Test. 14

SOTA state-of-the-art. 15, 17, 24, 36

WMT Workshop on Statistical Machine Translation. 25

Chapter 1

Introduction

1.1 Motivation

In late 2022, ChatGPT was released, and shocked many users with the quality of human-like responses and vast number of possible applications. The mass use of generative chatbots represents a new phase in Artificial Intelligence (AI). Rather than simply regurgitating facts from a knowledge base, chatbots can now learn from a corpus of text and generate their own human-like output.

A chatbot is an AI model which attempts to simulate human speech in a written manner, without a physical presence (Nee et al., 2023). The increased availability of high-quality, "excellent prose" generative chatbots has several benefits (Floridi and Chiriatti, 2020), particularly in education, which this paper will focus on. These include literature review assistance, data analysis, summarisation of documents, and question answering (Lund and Wang, 2023). (ChatGPT was even used to assist in the generation of some BibTeX citations for this paper, to speed up referencing.) However, there are several drawbacks, particularly surrounding the perpetuation of bias, privacy breaches, intellectual property rights and environmental costs associated with training a Large Language Model (LLM) (Jungherr, 2023; Bender et al., 2021). When it comes to education specifically, generative chatbots pose an elevated risk of students cheating in assessments, as well as the inability and unreliability of current methods (and even ChatGPT itself) to detect AI-generated text (Susnjak, 2022; Cotton et al., 2023).

The increased use of AI within education is "one of the most important contents of future education development" (Brown et al., 2020a) and is likely to continue to be for many years (Zawacki-Richter et al., 2019). While there is a need to improve current technology and mitigate the risks and any negative impacts, this paper seeks to focus on the opportunities available to education providers. Historically, Q&A in the classroom has been difficult and (in higher education) is often constrained to 'office hours' or 'consultation hours' where lecturers can spend 1-1 time with students to assist with specific queries. However, this can mean that academics repetitively answer similar questions posed by different students. The duplication of questions can be assisted by using platforms such as Piazza (n.d.), that can enable students to see all queries and post their own anonymously. However, this can also lead to duplicated questions as previous questions and answers are not always seen. Additionally, the professor may need to spend a significant amount of time typing up specific answers. Consequently, there is scope for the learning experience of students to be improved.

There are numerous benefits that chatbots can bring to the classroom (Ondáš et al., 2019). They can provide answers to FAQs and explain difficult concepts in plain language, which makes them popular among students. Furthermore, students can obtain answers almost instantaneously and at any hour of the day, rather than waiting hours or days for a response from an academic. Furthermore, students' research can be streamlined by pointing them to high-quality sources and finding the most relevant parts, particularly where they may have been overwhelmed by the quantity of information available on search engines (Chen et al., 2022). For academics, chatbots can answer the FAQs and more simple, repetitive questions that students have. This can free up academics so that they can spend their time more efficiently, to "reinforce the learning of students", improve their teaching methods (Pérez et al., 2020), and "focus on new experimental designs" (van Dis et al., 2023).

However, generative chatbots such as ChatGPT can produce errors (Marcus, 2018; Bender et al., 2021; van Dis et al., 2023), particularly when seeking to summarise and abstract documents (Durmus et al., 2020), and can perpetuate any bias in the training data (Geva et al., 2019; Brown et al., 2020b). This means that students could learn inaccurate or irrelevant information, particularly when answers sound plausible and confident but are inaccurate. ChatGPT itself, when asked about the drawbacks of chatbots in education, said that "Chatbots may lack accountability for errors or malfunctions, which could result in incorrect or harmful guidance to students" (ChatGPT, 2023). Additionally, the most advanced chatbots often lack knowledge after a specific date as the training process is resource-intensive and very time-consuming (Jungherr, 2023).

1.2 Aims and Objectives

The objective of this paper is to contribute to the field of long-form, closed-domain generative question answering. The aim is to discover whether this can be accomplished by fine-tuning a pre-trained Transformer model. Additionally, this paper will outline which fine-tuning method leads to the best outcomes, and how these methods compare with the use of existing fine-tuned models. Achieving these research aims will enable this paper to propose an efficient framework for academics to create a domain-specific, knowledgeable Q&A chatbot, where academics can have high confidence in the accuracy of any answers. Importantly, it will seek to be a “supplement to teaching and learning,” rather than replace current teaching methods entirely (Nee et al., 2023).

1.3 Paper Structure

This paper will begin by outlining the development of chatbots and the latest state-of-the-art technology used. After this, three potential models will be chosen and justified, the results of their performance will be discussed, and the best-performing model will be outlined. Finally, the findings of this paper will be summarised and complemented by a discussion of the recommended direction of future research. All code is available in the following GitHub repository: `https://github.com/pointonjoel/MSc-Diss`.

Chapter 2

Background and Literature Review

2.1 Embeddings

For AI models to be able to process text, a numerical representation is required. Improvements in these numeric representations have significantly contributed to the field of Natural Language Processing (NLP), particularly in recent years. Initial models used integer IDs to represent words by deriving an integer value from the frequency of a word occurring. This is known as a bag-of-words approach and is based on the work of Harris (1954). Although variations of this have been employed, such as 'n-grams' (collections of n successive words), this approach is limited in its ability to capture the semantic information within a language (Kanakaraj and Guddeti, 2015). In addition, these methods face challenges when dealing with large corpora as the model dimensionality is dependent on the number of unique words.

2.1.1 Word2vec

An alternative approach known as word2vec was developed by Mikolov et al. (2013) and aimed to overcome both of these limitations by building on the work of Bengio et al. (2000). It is a technique which uses two-layer neural networks (see Section 2.3) to produce learned word embeddings so that words that have similar usage in the training corpus are close (have a similar cosine similarity/small cosine distance) in the embedding vector space. Therefore, these vectors are superior to integer IDs in containing semantic detail. We can observe this by considering a common example using the vector representations of the words king, man, and woman (w_K, w_M, and w_W respectively), it can be shown that:

$$w_K - w_M + w_W \approx w_Q$$

$$King - Man + Woman \approx Queen$$

Where w_Q denotes the vector representation for 'Queen', which is the word with the smallest cosine distance to the above calculation (Allen and Hospedales, 2019).

Mikolov et al. (2013) proposed two possible approaches: one is to use the surrounding words to predict the 'current' one (Continuous Bag-of-Words (CBOW)) and the other is where the 'current' word is used to predict the surrounding ones (Skip Gram). A key feature of this approach is that each vector characterises the context of the word by considering neighbouring tokens, as opposed to just the word itself (Li, 2018). However, this approach cannot handle words that are not in the training corpus. Additionally, words with contrasting sentiments, such as "good" and "bad", are closely located in the vector space (Sivakumar et al., 2020) as they are often used in similar contexts, highlighting a key limitation of the CBOW or Skip Gram approach.

2.1.2 Context

Word2Vec and other similar embeddings such as GloVe (Pennington et al., 2014) have long been the industry standard. However, these embeddings fail to take context into account. For instance, such models provide one vector per word and, consequently, cannot differentiate between a 'fun fair' and a 'process being fair', for example. This stems from disregarding word order and, importantly, generating a sole vector representation per word. Therefore, it is essential to generate contextualised embeddings to represent words. Transformer models such as Bidirectional Encoder Representations from Transformers (BERT) were developed by Devlin et al. (2019) in response to the original 'Attention is All You Need' paper (Vaswani et al., 2017) to overcome this limitation, taking into account both left- and right- context (i.e. words before and after the current word).

Entire sentence and paragraph embeddings can also be generated, which can be used for sentiment analysis. Averaging non-contextual word embeddings is a crude way to achieve this, but Transformer models typically used attention mechanisms to extract the semantic meaning of a sentence or paragraph and generate a high-dimension embedding vector (Chase et al., 2023).

2.2 Early Chatbots

The earliest chatbots used rule-based logic to respond to textual inputs. One of the earliest chatbots was Eliza, created in 1966 by researchers at MIT to pass the Turing Test (ZEMČÍK, 2019). It used pattern matching to be able to construct human-like replies (Bradeško and Mladenić, 2012). However, the responses were often formulaic and predictable. These systems faced limitations due to the intricate nature of language, as it is highly difficult and inefficient to generate rules to handle every possible query. Many developments have been made since these early frameworks, with the adaptations outlined below.

2.3 Artificial Neural Networks

Artificial Neural Networks (ANNs) are part of a branch of machine learning called deep learning, with machine learning itself being a branch of Artificial Intelligence. ANNs seek to provide solutions to a wide range of classification, pattern recognition and prediction problems and are used extensively in image recognition and Natural Language Processing (NLP) tasks (Abiodun et al., 2018). Inspired by the human brain, they are analogous to the nervous system; they take an input and, using a complex set of neurons, seek to identify an output response (Bishop, 1994). They do this by learning from examples, in a similar way to humans. For example, ANNs can be used to predict whether an image contains a pizza or a football.

2.3.1 Structure

Neural networks take a series of inputs (via the input layer) and seek to predict the output (via the output layer). To achieve this, they often contain several ‘fully connected’ hidden layers of a pre-determined size consisting of neurons and nodes that hold weights and biases. The size of the input layer is determined by the number of attributes that the model has available to it, and the size of the output layer is determined by the classification/prediction problem.

This structure is shown in Figure 2.1, where each grey circle denotes a node, and each interconnecting line denotes a weight between two nodes. As can be seen, each node is connected to every node in the next layer (in a fully connected layer), and the value of each node in the next layer is a weighted sum of the values of the nodes in the previous layer and their corresponding weights (and sometimes a bias term) (Bishop, 1994). The weights, therefore, determine how much information is passed on to each node in the next layer. The weights are analogous to the strength of the connection of biological neurons, and the bias is analogous to the firing threshold in the human brain (Rathi and Roy, 2020).

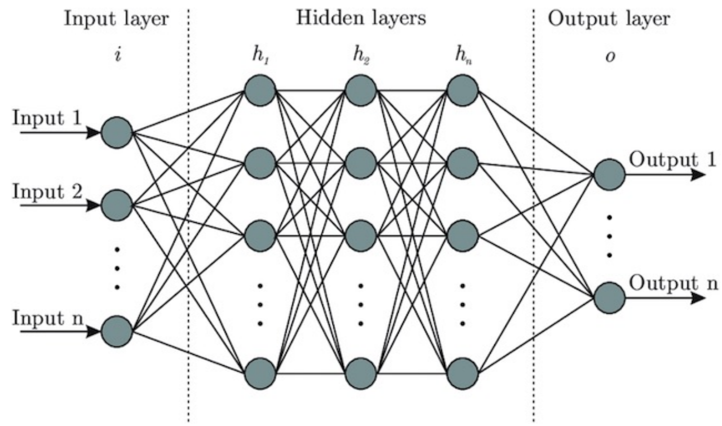


Figure 2.1: Typical Neural Network Structure (Shukla, 2019)

2.3.2 Activation Functions

The value in each node of a neural network is typically transformed by an activation function, often Sigmoid or Rectified Linear Unit (ReLU). The former ensures that the values are non-linearly scaled to be between 0 and 1 using $f(x) = \frac{1}{1+e^{-x}}$, whereas the latter truncates values that were below 0 to be 0 using $f(x) = max(0, x)$. Consequently, the range of a sigmoid activation function is $(0, 1)$ and the range of a ReLU activation function is $[0, \infty]$. These activation functions, along with some other common ones, are shown in Figure 2.2.

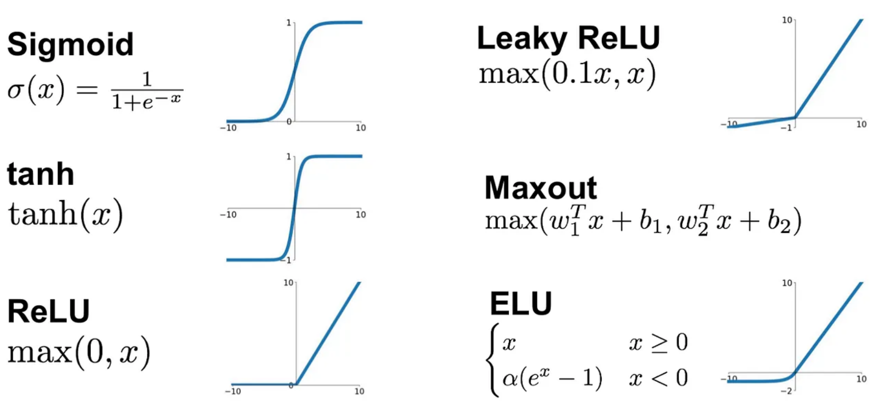


Figure 2.2: Activation Functions Used in Neural Networks (Udofia, 2018)

This process is continued for each of the (fully connected) hidden layers until the network has calculated the values for each of the nodes in the output layer. This will typically be a proportion which, in the case of next sentence prediction, denotes the likelihood of that specific word being the next word in the sentence.

2.3.3 Learning

The learning in neural networks occurs in training the weights that connect each of the nodes. Artificial Neural Networks use a technique known as Forward Propagation to calculate the predicted output from a given input. Initially, the weights in the network are randomly assigned (Bishop, 1994), meaning that the model has essentially no prior predictive power. For each example data point the network calculates the values for each of the nodes in the output layer; these denote “the probability that the given input fits into each of the pre-set categories” (Yathish, 2022). This process is repeated for each of the training examples and is known as Forward Propagation.

The predicted values are subsequently compared against the expected (true) value, and the model computes the error (the difference between the predicted and true values). This error is fed into a loss function (E), which is a measure of the inaccuracy of the model; the aim is to minimise the loss function. For most classification problems, a cross-entropy (log loss) function is used to compare the difference between the actual value (y_i) and the predicted value ($\hat{y_i}$) for each of the prediction classes (of size $N = n^L$, where L is subscript and not an exponent):

$$E = -\sum_{i=0}^{n^L-1} y_i * log(\hat{y_i})$$

Once the value for the loss function has been calculated, the model seeks to update weights and biases in each layer, using a process called backpropagation (Rumelhart et al., 1986; Nielsen, 2015). We can define the following notation:

a_j^L: *The output value of the jth node in the Lth layer (once the activation function has been applied)*

w_{jk}^L: *The weight connecting the jth node in the Lth layer and the kth node in the (L-1)th layer*

b_j^L: *The bias applied to the jth node in the Lth layer*

$z_j^L = w_j^L a_j^{L-1} + b_j^L$: *The value of the jth node in the Lth layer*

We can therefore say that $a_j^L = \sigma(z_j^L)$, where σ denotes the activation function.

The backpropagation process begins by computing the rate of change of the cost function with respect to each of the weights (holding other weights constant) because we are seeking to minimise the cost function. This is given by $\partial E/\partial w_{jk}^L$. By using the chain rule, we can expand this expression:

$$\frac{\partial E}{\partial w_{jk}^L} = \frac{\partial z_j^L}{\partial w_{jk}^L} \frac{\partial a_j^L}{\partial z_j^L} \frac{\partial E}{\partial a_j^L}$$

Similarly, we can calculate $\partial E/\partial b_j^L$ and $\partial E/\partial a_j^{L-1}$. Using these, the gradient of the cost function can be computed and used to update the parameters above so that the cost function is reduced. Gradient Descent is the process of adjusting the parameters in the 'direction' indicated by the gradient of the cost function, such that the loss function is reduced. The size of the adjustment is called the 'learning rate', and this affects how quickly the model adjusts its parameters.

This adjustment is applied to all of the network's layers as part of the backpropagation process and for each of the training examples. One iteration of the combined training process is known as an Epoch (Forward Propagation, Cost Calculation, and Backpropagation using Gradient Descent) (Sharma, 2017), and this is applied recursively until the loss function is sufficiently small. The smaller the learning rate, the more epochs are typically required to reach the minimum loss required. Once all of the epochs are completed, and the training process has reached a local minimum, the model is ready to produce predictions. However, it may not necessarily have reached a global minimum, meaning that predictions may not be completely optimal.

2.4 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are a form of neural network used for sequential data. They take each element of the sequence one at a time and use the current and previous values to predict future ones. They can crudely be thought of as "very deep feedforward networks in which all the layers share the same weights" (LeCun et al., 2015). This is depicted in Figure 2.3, where the input is processed sequentially $(x_{t-1}, x_t, x_{t+1}, \ldots)$, and information from the previous state is used to make the prediction in the current state (h_t). However, when backpropagation is used to train the network, problems are often encountered.

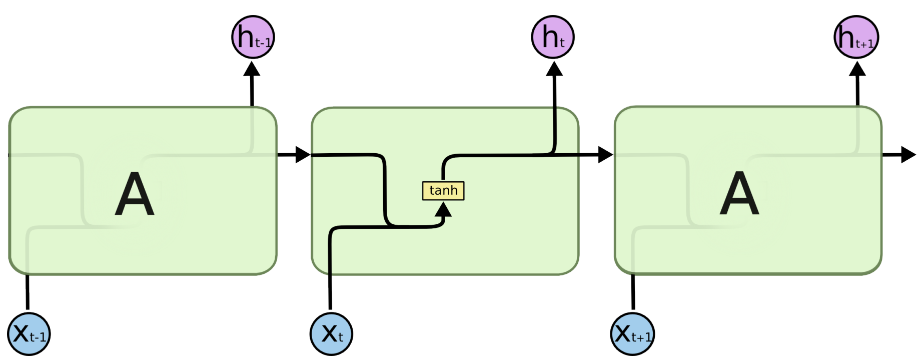


Figure 2.3: Architecture of a Recurrent Neural Network (Olah, 2015)

A common problem with training neural networks is the vanishing/exploding gradient problem (Hochreiter and Schmidhuber, 1997). This can occur in deep neural networks but is particularly common in RNNs, as the same weights are used in each iteration. The exploding gradient problem is where the model weights become exponentially large, which causes the model weights to become NaN. Alternatively, because of the recurrent structure of the model, there can be a tendency for model weights to 'vanish' and tend to 0. This causes the model to have short-term memory because it fails to capture long-term dependencies (Chung et al., 2014). In addition to this, in both cases, the loss function is not minimised because the weights cause the loss function to either overshoot or never reach the global/local minimum.

2.4.1 Long Short-Term Memory Cells

Long Short-Term Memory (LSTM) networks are a type of Recurrent Neural Network which were developed by Hochreiter and Schmidhuber (1997) to overcome the vanishing/exploding gradient problem. They are depicted in Figure 2.4 where, instead of one simple layer (as in Figure 2.3), there are three layers with different activation functions. LSTM networks use a sigmoid (σ) activation function for each of their three 'gates' (forget gate, input gate, and output gate) to determine how much of the long-term memory is maintained and to update both the long-term and short-term memory in each cell. The first (most left-wise neural network layer in Figure 2.4 is the 'forget gate'. The 'input gate' refers to the second layer, which is subsequently combined with a layer with a tanh activation function to update the long-term memory. The final layer is the 'output gate', which also uses a sigmoid activation function. By using this structure, LSTM networks overcome the vanishing and exploding gradient problem because they control how much the gradient vanishes using the 'forget gate' (Gers et al., 2000).

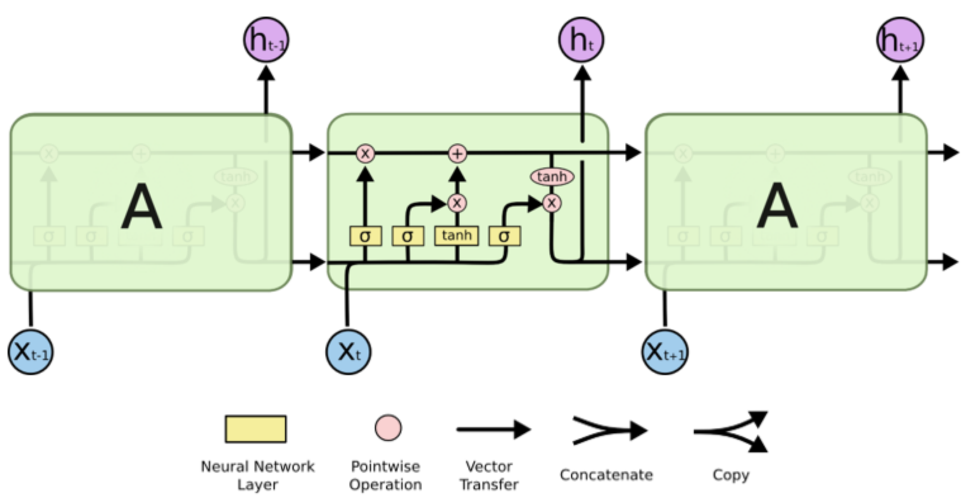


Figure 2.4: Architecture of a Long Short-Term Memory Cell (Olah, 2015)

2.4.2 Gated Recurrent Units

Another similar model to LSTMs is the Gated Recurrent Unit (GRU), with an architecture based on just two gates (reset gate and update gate). It was developed in 2014 by Cho et al. (2014) and provides a simpler architecture than the LSTM model. This model is shown in Figure 2.5, which shows how the flow of information is held using a 'hidden state' and the two gates (the neural network layers with sigmoid activation functions) determine how much information it remembers or forgets. The update gate determines how much of the memory it retains, and the reset gate determines how much of the memory it forgets.

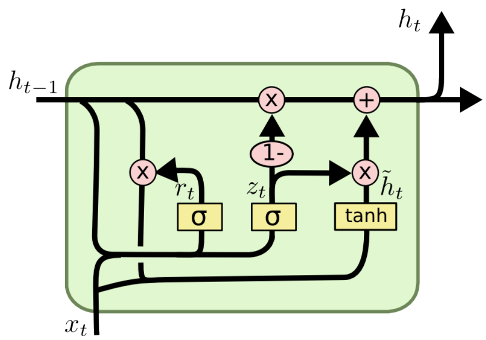


Figure 2.5: Architecture of a Gated Recurrent Unit (Olah, 2015)

The Gated Recurrent Unit has become popular due to its simplicity relative to the LSTM architecture, but LSTM cells and GRUs often perform similarly effectively. However, it has been noted in the literature that GRUs generally outperform LSTM networks on sequences that are short and less complex, whereas LSTM models are typically preferred for longer and more complex sequences (Cahuantzi et al., 2023). This is often attributed to the LSTM model's ability to better capture long-term dependencies in sequences, which often means it is preferred for language modelling (Irie et al., 2016). However, both models (and vanilla RNNs generally) can only capture forward dependencies, due to their sequential nature. For example, with the sentence 'Joel read a book about a bass that was owned by a fisherman', using only the first seven words, you would not know whether the word 'bass' refers to the fish or the instrument. It is only with the latter parts of the sequence that you can determine the context and therefore the bass was owned by the fisherman and not the musician. Therefore, models which only capture forward dependencies will miss any potential inference based on future words.

2.4.3 Bidirectional Recurrent Neural Networks

To overcome this limitation, Bidirectional RNNs were developed by Schuster and Paliwal (1997) and are a combination of two RNNs (see Section 2.4). One RNN processes information in the usual chronological manner, while the other processes it in reverse time order. The model is trained simultaneously on both of these and seeks to minimise the loss function for both time directions concurrently. This allows the model to capture the future context in sequences, which is particularly important in NLP implementations because the context of words is typically derived from future words.

All of the above models (RNNs, LSTMs, and GRUs) require sentences to be processed sequentially, so can take a long time to train especially when there are long strings to process, and can have convergence issues due to vanishing/exploding gradients (Vaswani et al., 2017; Lipton et al., 2015). We will now explore an alternative model which seeks to solve this.

2.5 Transformers

Transformers were introduced in Vaswani et al.'s 'Attention Is All You Need' paper (2017). The authors proposed a sequence-to-sequence model which uses six stacked encoders and decoders, though subsequent models often use varying numbers of encoders and/or decoders. The strength of Transformers lies in their ability to process all words in a sentence simultaneously (the sequential nature of LSTM networks made them slow to train), and their ability to retain context from much further back in the sequence (Vaswani et al., 2017).

While there are many variations, the initial model architecture is shown in Figure 2.6. It begins by taking initial embeddings such as word2vec (Google's BERT uses wordpiece tokenisation to easily handle out-of-vocabulary words (Wu et al., 2016)), and applies a vector containing positional encodings for each word (using *sine* and *cosine* functions). These positional encodings are then passed into the encoder block.

2.5.1 Encoder

An encoder consists of a multi-headed attention block combined with a feed-forward neural network, both of which are followed by layer normalisation. Firstly, the positional encoding vectors are used to calculate self-attention vectors, which specify how each word relates to every other word in the sequence. Each self-attention vector can be independently passed into the neural network, enabling the model to leverage GPU parallelisation and significantly reduce training time. Finally, the encoder outputs a contextualised vector, which can be intuitively conceptualised as containing the 'meaning' of the word within its context.

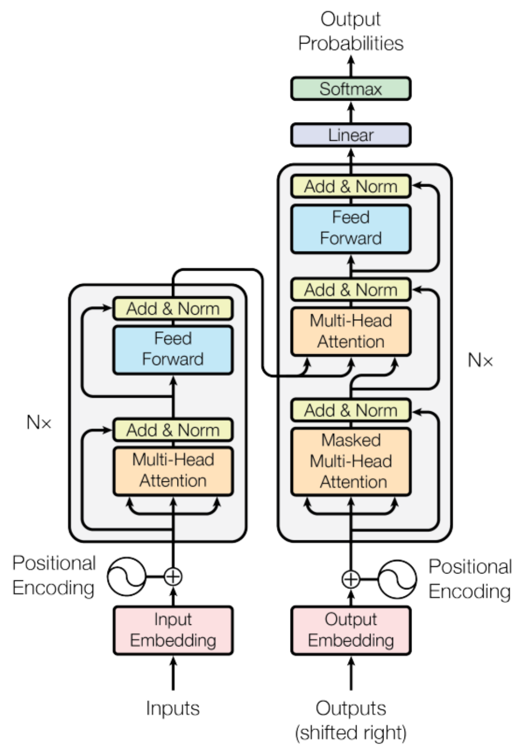


Figure 2.6: Architecture of a Transformer model (Vaswani et al., 2017)

2.5.2 Decoder

The decoder uses a similar architecture to the encoder but uses masking to prevent the model from 'seeing ahead' in the sentence. For example, to translate the phrase "Esta es una tesis fantástica" from Spanish to English, the model should not see any word "tesis fantástica" when calculating the self-attention for the phrase "Esta es una". This means that the model has unidirectional-context (typically only left-context). Once the masked multi-headed attention vectors have been calculated, the decoder combines this with the output from the encoder using another multi-headed attention block. Finally, this information is passed into a feed-forward neural network to produce the prediction for the next word in the sequence until an end-of-sentence token (`<EOS>`) is produced.

2.5.3 Variations

There are multiple variations of the Transformer architecture. They broadly fit into three categories. The first is autoencoding (bidirectional) models which use stacked encoders (e.g. BERT: Devlin et al. (2019)); the second is autoregressive (left-context) models which use stacked decoders (e.g. GPT: Radford et al. (2018)); the third is a combination of both, using both stacked encoders and decoders (e.g. BART: Lewis et al.

(2019)), and these are known as sequence-to-sequence models.

BERT excels at text classification and question answering. The model was released by Google in October 2018, and its success comes from utilising between 12 and 24 stacked encoders ($\sim$110 and $\sim$340 million parameters, respectively). These enable the model to extract high-dimensional semantic information about the text, and therefore either classify the sentiment or extract words which answer a question.

GPT was released after BERT, with the first version of GPT (GPT-1) released by OpenAI in June 2018. The model excels at being able to generate contextually-aware text in natural, human-sounding form, due to its use of many stacked decoders. Subsequent versions, such as GPT-2 and GPT-3, were improvements on the original model and used to develop ChatGPT, which was released in November 2022 (OpenAI, 2022b). However, the strength of GPT is also one of its main criticisms; it is trained to generate text using statistical patterns and distributions in its training data, but this can result in the generation of plausible but false content. This method of training also results in the perpetuation of bias. For example, OpenAI ran the SEAT (May et al., 2019) and Winogender (Rudinger et al., 2018) benchmarks to identify any potential bias in their model and found that GPT tends to associate positive sentiment with European American names when compared to African American names, and can be prone to generating content associated with negative stereotypes of black women (OpenAI, 2022a). This is caused by the model learning patterns from its training data which itself contains harmful content.

BART was released after these models by Facebook AI Research in October 2019. BART incorporates both bidirectional encoders and auto-regressive decoders, making it one of the best models for extractive question answering (Pearce et al., 2021). However, BART also suffers from the same criticisms as GPT, resulting in potentially false yet plausible content. BART, along with the above variations of the Transformer architecture, is typically pre-trained using a very large corpus of information, and can subsequently be fine-tuned for downstream tasks, such as question answering, translation, or text generation. This is known as transfer learning, where fine-tuning pre-trained models can be completed using much smaller custom datasets, as the model utilises existing knowledge gained during the pre-training phase.

Transformer models are highly effective in being able to generate natural-sounding content and can be fine-tuned to complete a variety of tasks. They are more effective at retaining context than previous models and can process entire sentences simultaneously as opposed to sequentially (see Sections 2.4.3 and 2.5.1). However, they still require many improvements to ensure that they generate factually accurate and safe content.

2.6 Implementations

There have been several recent attempts to produce reliable chatbots using state-of-the-art (SOTA) Transformer technology. The literature identifies two angles of approach. The first is to use fine-tuning, typically Masked Language Modelling (MLM), to embed the knowledge into the model weights. The second is to find relevant paragraphs of information and subsequently provide this to the model by using the input message. It is important to note that, at the time of writing, GPT-3.5 Turbo and GPT-4, which power ChatGPT (OpenAI, 2022b) are not available for open-source fine-tuning. Only GPT-2 its predecessors are available on open-source platforms. Therefore, while Khan Academy and a limited number of other companies are in the process of creating and trialling a knowledgeable chatbot (Khan, 2023), this will not discussed in detail as it will not be accessible for most applications.

The first of the above two approaches is Masked Language Modelling (MLM). MLM is used to extract the semantic information contained within the training text by updating model parameters to accurately predict masked words given the context in surrounding non-masked words. The partially masked sentence is taken as an input (e.g. "I love natural [MASK] processing") and the model seeks to accurately predict the masked token (which in the above example is "language"). However, this is not designed to be efficient for factual recall. OpenAI in their documentation note that "fine-tuning is better suited to teaching specialized tasks or styles, and is less reliable for factual recall" (OpenAI, 2022). Furthermore, models that rely only on model weights to provide factual, accurate answers often suffer from "hallucination", which is a term the literature uses to refer to model responses which are confident and realistic-sounding but inaccurate (Manakul et al., 2023). As a result, this approach is not recommended for 'teaching' the model specific information.

The second approach involves using a document store, which can be implemented as part of either the pre-training or fine-tuning process. Guu et al. (2020) proposed a retrieval-augmented pre-training model which uses relevant texts (from Wikipedia) to assist with the prediction of masked tokens, and penalised any poor document retrieval by the model. This is outlined in Figure 2.7, where the retrieved document is used to pre-train the model to predict the masked token. However, this approach is not possible for most applications, because it requires huge amounts of data and processing power. Instead, most applications will require the use of models that have already been pre-trained.

It is therefore more efficient for most applications to instead have a separate retriever which uses a semantic search (e.g. using the cosine similarity of sentence/paragraph embeddings). This approach is recommended by OpenAI for use with their GPT-3.5 Turbo model (OpenAI, 2022). They describe the general knowledge embedded in the model as "long-term memory" and outline how providing relevant passages within an input

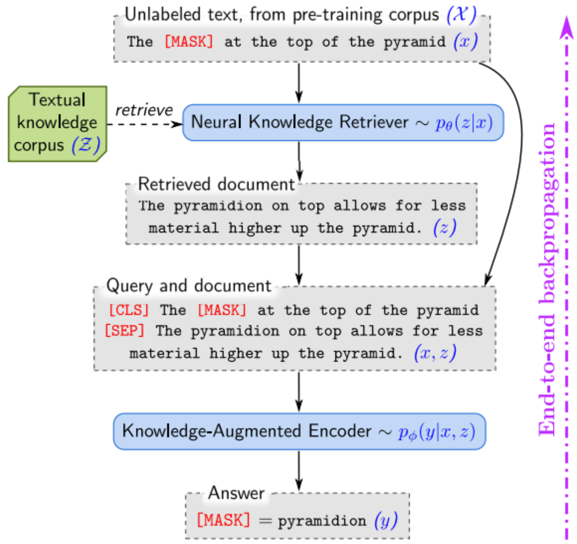


Figure 2.7: Pre-training of a Retrieval-Augmented Language Model (Guu et al., 2020)

message is akin to providing the model with "short-term memory" that yields more accurate and reliable answers. As a result of this, the development of relevant packages is underway, with early versions of an open-source Python package `LangChain` already available (Chase et al., 2023). By comparing the semantic similarity (e.g. cosine similarity) of a query and potentially thousands of relevant passages, the most relevant ones can be into a pre-trained model with a query, and the model can be fine-tuned to provide an answer.

There are limitations to this approach, however. Firstly, the accuracy of any model is limited to its ability to extract (natural-sounding) answers from a piece of text. Such a model can either be achieved by fine-tuning an open-source pre-trained model, or by using an already fine-tuned model that is adept at question-answering. Secondly, any model will be constrained by a maximum amount of text it can read at once, placing an upper bound on the amount of information it can use to produce an answer. However, this approach is still preferred in the literature, due to the frequency of "hallucination" as a result of relying solely on model weights.

Chapter 3

Methodology

3.1 Overview

Several possible approaches can be taken to achieve this paper's research aims. Current SOTA transformer models provide several methods. Firstly, a pre-trained model (such as BART, T5 or GPT) can be fine-tuned using Masked Language Modelling (MLM) on a specific document. Alternatively, a fully fine-tuned model can be used and simply fed the most relevant paragraphs from the 'knowledge' base. A third and final model is a combination of the other two approaches: a pre-trained model can be fine-tuned for general question-answering from a context, such that questions are posed to the model with the most relevant paragraphs from the 'knowledge' base included as context.

'Knowledge' in this context is used to refer to the information that the model aims to learn and will be used like this henceforth. This 'knowledge' can be in the form of websites, PDFs (e.g. textbooks), and other supporting documents used by a module. Pre-training a model provides more flexibility and opportunity for the model to become very good at this specific task, but is much more resource-intensive, may be difficult to achieve high-quality results, and can be less scale-able. I will begin by discussing the methodology behind creating the 'knowledge' base used by the transformer models, and then proceed to outline the different models which this paper will compare.

3.1.1 Knowledge

In order to encapsulate the 'knowledge' any chatbot should aim to contain, relevant documents were obtained from the University of Nottingham's COMP-2032 Computer Vision module. These included textbooks, lecture notes, and lecture transcripts. Due to the lecture notes containing only key headline information which was then discussed in audio format, lecture notes were not used to fine-tune the model. Furthermore, the transcripts were of poor quality and contained mistakes which could cause any model to perform badly. Subsequently, only textbooks were used to form the 'knowledge' base, because they were of high-quality and were around 500-1000 pages long. The two textbooks used to form the 'knowledge' base were '*Digital Image Processing textbook*' (Gonzalez and Woods, 2018) and '*Fundamentals of Digital Image Processing textbook*' (Solomon and Breckon, 2010).

PDF textbooks were parsed using Python's `Fitz` package (from `PyMuPDF`), with the text extracted using the package's *blocks* functionality, meaning that text was extracted line-by-line, or by paragraph depending on the context. This text was cleaned to remove websites, and then each block of text was merged with the next until the token limit (800) was reached. There is a trade-off between merging potentially irrelevant content, having large contexts to provide the model with as much background information as possible, and having section lengths that can be processed by multiple models. Additionally, the tokeniser used to determine the number of tokens per section was the 'all-mpnet-base-v2' model (Reimers and Espejel, 2021), as it typically produced higher token counts than GPT and T5, meaning that sections would never be larger than other models can handle.

Similar functionality was implemented for Wikipedia pages, to allow lecturers to provide specific pages that are relevant to a course. These pages are split into sections according to their headers and then split again using a delimiter if the section contained more than 800 tokens (as above). None of the models in this paper use Wikipedia, but the code has been included for completeness and future development.

3.1.2 Setup

All models below were fine-tuned and run using Google Colab (Bisong, 2019). Access to Graphics Processing Units (GPUs) was kindly provided by the University of Nottingham, but the vast size of the models in comparison to the quality of GPUs available meant that training could take up to 10 hours or more to complete and required more RAM than was available. Colab Pro provided access to Google's A100 GPU was available with 40GB GPU RAM, 80GB CPU RAM, and over 200GB of disk space. A virtual environment with the following deep learning packages was used: 🤗 `Transformers` (4.31.0), `PyTorch` (2.0.1+cu118), `Datasets` (2.13.1), and `Tokenizers` (0.13.3). The 🤗 `Transformers` library (specifically the `Seq2SeqTrainer` class) was directly used to fine-tune the models (Wolf et al., 2020). GPT's embedding model used the `Tiktoken` package (0.4.0) which requires Python >=3.8. Finally, all training, shuffling, and processes were conducted with a seed value of 9 to ensure reproducibility.

3.2 Masked Language Modelling

The simplest model produced in this paper is a fine-tuned version of a pre-trained model (GPT). The chatbot is fine-tuned to the module's domain (Computer Vision) by using Masked Language Modelling (MLM). By randomly 'hiding' (masking) a proportion of the tokens within the training data (e.g. a textbook), the model is trained to predict their value. This is discussed fully in Section 2.6.

3.2.1 Training

This model was trained using the Digital Image Processing textbook (Gonzalez and Woods, 2018) and validation metrics were computed using the Fundamentals of Digital Image Processing textbook (Solomon and Breckon, 2010). Both textbooks were imported into the 'knowledge' base as described in Section 3.1.1 and then exported into a `txt` file, with excess line breaks removed. The text was then additionally split into blocks of 512 tokens, using the GPT tokeniser. The model was trained with a batch size of 16 (due to RAM constraints) for 20 epochs. Regularisation was used to prevent overfitting, a with weight decay value of 0.01 and dropout of 0.6. Training used a learning rate of 3e-5, and took around 30 minutes to complete as the use of an early stopping callback (with a patience of 3 and a threshold of 0.05 for the validation loss) meant that the model finished training after almost 7 epochs.

3.2.2 Evaluation Metrics

The main metric used to evaluate this model was perplexity, which is defined as the exponent of the cross-entropy validation loss of the model. Perplexity values can be any number above 1, and most language models aim to have perplexity values in the single digits. This is because a lower perplexity indicates that the model is more confident and accurate in predicting the next word, as it better understands the domain and context of the text. This is the most important factor in determining whether the model has accurately 'learnt' sufficient information to be able to respond to queries and provide accurate responses.

3.3 Document Store With GPT-3.5 Turbo

The second approach used by this paper is the document store approach, combined with a fully fine-tuned GPT model. Given a set of documents (word documents, PDFs, Wikipedia pages), this framework chunks them into sections which contain sufficient context but are short enough to be processed by the model (and not use an inefficient or excessive number of tokens). These sections are stored in a data frame, along with the document name/section header, from which section embeddings are created.

The embedding model can be flexible, provided the same embedding model is used to also embed the student's question. GPT's embedding model is faster and can process much longer contexts, but these longer contexts cannot often be handled by other models (e.g. T5). Additionally, it has a small financial cost associated (Section 4.4). The GPT embedding model was used during model testing, but functionality for the use of an open-source alternative such as the 'all-mpnet-base-v2' model (Reimers and Espejel, 2021) was also implemented.

Once the embedding model has been chosen, the sections of 'knowledge' can be embedded. The user's question is then also embedded, with its cosine similarity between each section of 'knowledge' used to determine the top (n) relevant sections. These sections are fed into the AI model, along with the student's question, and it provides a response. If the model can't find an answer in the context provided, it responds appropriately. This framework is shown in Figure 3.1.

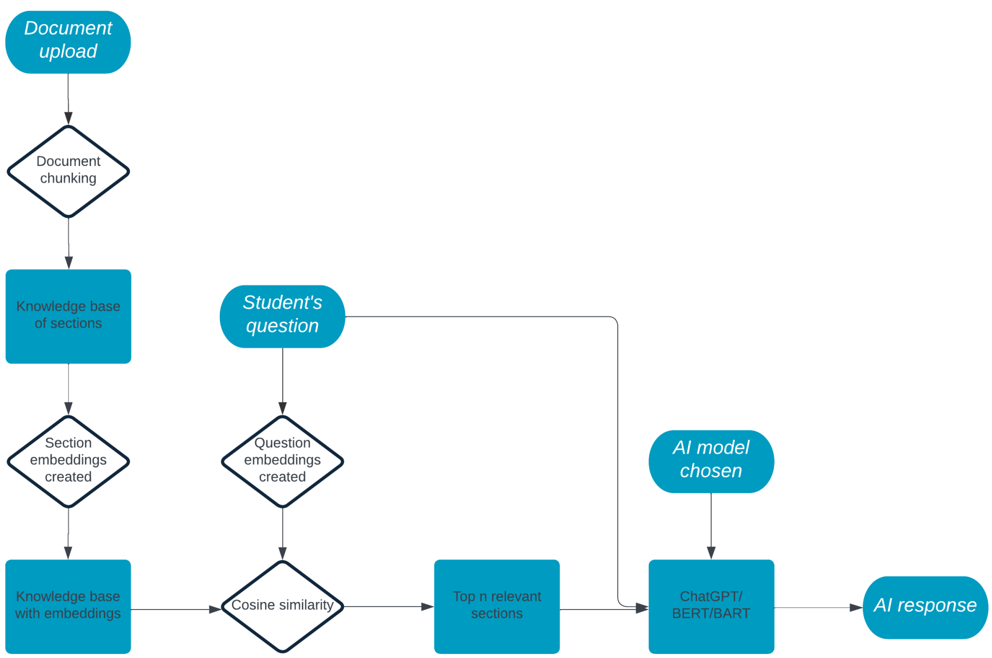


Figure 3.1: Document Store With AI Chatbot Framework

3.3.1 Implementation

The document store/'knowledge' base for this model used the two PDF textbooks outlined in Section 3.1.1. The top 5 sections were fed into GPT-3.5 Turbo using OpenAI's API, along with an instruction to:

> *Use the below article on Computer Vision, to answer the subsequent question. If the answer cannot be found in the article, write: I could not find an answer in the text I've been provided, sorry! Please try again.". If you are asked to produce any code, then decline the request and write "Sorry but I'm not allowed to do your assignments for you!".*

There was also a combined token limit of 3596, due to the GPT API token limit of 4096 being reduced by 500 tokens to allow for a response. This approach benefits from being efficient and less resource-intensive, as it uses the existing fine-tuned GPT model. Only the top 5 sections of knowledge were provided to the model, provided they had a cosine similarity of greater than 0.5. The limitations help to minimise any token wastage when using the API.

3.3.2 Potential Drawbacks

A key drawback of this model is the large cost. While the costs associated with this model will be discussed in Section 4.4, they are large and may be difficult for teaching departments to justify, given that ChatGPT is currently free. Additionally, the model would always be dependent on OpenAI's support for the API, along with its availability and stability. The reliability of the API is beyond the control and scope of this paper but could harm the feasibility of this setup in practice.

3.4 Document Store With Fine-Tuned T5

The third model considered by this paper is a combination of the above two models. The challenge is to achieve the accuracy of a fully fine-tuned model (such as GPT-3.5 Turbo) at a fraction of the cost. The framework in Figure 3.1 can be adapted to use any relevant AI model. For this model, only contexts with a cosine similarity of above 0.7 were provided as an input, to assist with detecting unanswerable questions. If the model is fine-tuned to answer a question using a provided context, it can excel at extracting answers and recognising when an answer cannot be found. This can be achieved by adapting an innovative method of training which is typically used for article summarisation tasks, involving sequence-to-sequence training.

3.4.1 Dataset

Training the model for question-answering used Google's Natural Questions dataset (Kwiatkowski et al., 2019). It is the only dataset known to contain naturally phrased questions with answers in natural language.

While the answers are extractive (i.e. directly contained within the context provided), they are long-form answers which can be used to fine-tune a model to write natural-sounding answers.

This dataset consists of 307,373 training examples and 7,830 examples for validation. Each example contains a question related to a specific Wikipedia page, which was given in both HTML and tokenised form. Both a short answer (at most five tokens) and a long answer were available, with some answers being descriptive and therefore only having a long answer.

3.4.2 Pre-Processing

The dataset requires 154GB of disk space and so only the relevant information was maintained. To clean and reduce the disk size of the dataset, only the question, long answer, and page content in tokenised form were retained. Answers were cleaned to remove any unnecessary whitespace, and the tokenised page content was cleaned of any references (e.g. [2]).

Additionally, only examples which have answers contained within a `<P>` tag were kept (approximately 72.9%), as the main aim of the model is to provide textual answers rather than use tables and lists. Answers using the following HTML tags were therefore removed: `<table>`, `<tr>`, `<th>`, `<td>`, `<ul>`, `<li>`, `<dl>`, `<dd>`, `<ol>`, `<dt>`. Furthermore, examples with a particularly short context (<100 tokens) were removed. Additionally, for examples with a context larger than 1024 tokens, any which contained the answer within the first 1024 tokens were retained and the context was truncated; any without the answer in the first 1024 tokens were removed. This upper limit was chosen because most models cannot process more than 1024 tokens. The model used for fine-tuning in this paper (T5, see section 3.4.5) can process more than 1024 tokens, but there is an efficiency/memory trade-off, as longer contexts require much more RAM.

One drawback of the dataset is that Wikipedia articles typically start with a summary paragraph which is unrealistic for this use case. Therefore, any model may be skewed to incorrectly return the first sentence as the answer. To combat this, any examples where the answer is simply the first line or paragraph were also removed (approximately 16%).

Additionally, the above pre-processing results in a very high proportion of unanswerable questions relative to answerable ones. As a result of this, the number of unanswerable questions was reduced to only 30% of the training and test sizes. This was achieved by removing a random sample of the unanswerable questions. Finally, the combined dataset was split into a training and validation set using an 80-20 ratio, with 22,512 training examples and 5,519 validation examples. A `[NO_ANS]` token was used to denote an unanswerable question.

3.4.3 Data Enhancement

A key limitation of the Natural Questions dataset is that it provides only extractive answers. The answers are a sentence or paragraph of information which can contain irrelevant information to the question. Therefore, fine-tuning a model using this data will not train it to produce succinct summary answers, and it may even struggle to identify correct answers as the training answers can be too large and contain ambiguous content.

To address the issue of poor-quality answers, a one-off enhancement to the dataset was conducted using GPT-3.5 Turbo. GPT was provided with each question-answer pair and asked to answer the question by only using the answer provided. The development and implementation of this had a one-off cost of only $20; however, GPT was unable to answer 25.6% of the questions. For example, GPT-3.5 Turbo was unable to find the answer to the question "charlie puth we don't talk anymore album" when given the following context:

> *"We Don't Talk Anymore" is a song produced and performed by American singer Charlie Puth from his debut studio album Nine Track Mind (2016). It contains guest vocals from Selena Gomez. Jacob Kasher co-wrote the track with the artists. The song was released on May 24, 2016, as the third and final single from the album. Musically, it is a pop song with tropical-inspired production.*

The correct answer of "Nine Track Mind" was available but not identified; although, this example highlights the necessity to succinctly rewrite answers, as the original answer contains irrelevant information about the type of song and its location on the album.

Therefore, around 1 in 4 question-answer pairs were removed as a result of this process, causing the resulting dataset to contain 10121 training examples and 2484 validation examples. While a reduced dataset is unfortunate, some questions were ambiguous and not necessarily answerable using the context. As such, the process of using GPT-3.5 Turbo to rewrite answers refined the dataset to only include high-quality examples. After this process, an additional random subset of unanswerable questions was removed to maintain the 30% percentage of unanswerable questions. This use of GPT-3.5 Turbo was a one-off requirement and therefore the ongoing use of this dataset has no reliance on the availability of the API. However, any model trained from this dataset may be limited in its capacity to learn, particularly if the answers are not as high-quality as human-written answers.

3.4.4 Data Format

Each example was manipulated to be in the following format:

> ***Input:*** *She is a song written by Charles Aznavour and Herbert Kretzmer and released by Aznavour as a single in 1974. ... Elvis Costello recorded a cover version of the song in 1999. This version, produced by Trevor Jones, was featured over the final sequence of the film Notting Hill, and charted throughout Europe. ... The song has been recorded by many different artists over the years. The most notable versions include: </s> who sings the song she from notting hill?*
>
> ***Answer:*** *Elvis Costello recorded a cover version of the song in 1999. This version, produced by Trevor Jones, was featured over the final sequence of the film Notting Hill, and charted throughout Europe.*

As can be seen above, the answer to each question is contained within the context, meaning that the model is trained to produce answers using the provided context. The input began with the context and was followed by the T5 separator `</s>` and the question. The target for the model was the encoded form of the long answer. This sequence-to-sequence fine-tuning aims to train T5 to understand the input and question and respond accordingly.

3.4.5 Training

An MT5 (small) model was trained using 22,512 examples, with a learning rate of 1e-3. Regularisation was used to prevent overfitting, with dropout of 0.4 and weight decay of 0.007. The fine-tuning was conducted using a batch size of 16 and gradient accumulation steps of 8 (due to RAM constraints), and almost 10 epochs, during which training took around 3 hours to complete.

T5 and other state-of-the-art models require huge amounts of computing power that were not available for this research. To combat this, gradient accumulation (using steps=4) and gradient checkpointing (activation checkpointing) were used to reduce RAM, at the expense of training time (Chen et al., 2016). These adaptations cause model gradients to be accumulated over several batches (4) before updating the weights, and for some intermediate activations to be recomputed during the backward pass instead of storing them in memory. While this does not directly affect model accuracy and performance, it can mean that convergence is slower and larger batches lead to different local minima. Overall accuracy should not be significantly affected (Huang et al., 2023).

3.4.6 Evaluation metrics

Evaluating Large Language Models is a difficult task due to their complexity, differing model objectives, and different forms of output; there are many facets to evaluate (Chang et al., 2023). Consequently, a variety of evaluation metrics will be used to evaluate model performance.

The first metric that will be used is the ROUGE score (Lin, 2004). ROUGE scores compare the n-gram overlap between the generated answer and the target answer, and also for the longest common subsequence. They are adept at measuring how much key information from the target answer is contained in the generated answer. However, there are many drawbacks to ROUGE scores, particularly that they are by nature difficult to compare, as the vocabulary and context vary in each application. Furthermore, a ‘perfect’ score of 100 is unlikely and generally unattainable even for humans, as two sentences with the same meaning can be written in different ways, using different words and connectives (Schluter, 2017). Typically a score of around 40 is considered reasonable.

The BLEU metric will also be used, which is similar to ROUGE but was initially developed for machine translation tasks. The BLEU score compares the precision of n-grams rather than the recall (Papineni et al., 2002), and the limitations discussed for ROUGE scores also apply. Additionally, METEOR scores will be used to evaluate the model (Banerjee and Lavie, 2005), which are similar but more complex than a ROUGE score. A METEOR score also aims to compare the accuracy of the generated answer to the target answer but additionally considers the alignment of words and phrases between the generated answer and the target answer. Neither BLEU nor METEOR scores have an ‘ideal’ target number, but range from 0 to 100, with the original transformer ‘Attention is All You Need’ paper achieving BLEU a score of 28.4 on the WMT 2014 English-to-German translation task, and 41.8 on the WMT 2014 English-to-French translation task Vaswani et al. (2017).

However, each of the above metrics fails to capture the semantic accuracy of answers. As a result, the cosine similarity will also be used to evaluate the model, which can range from 0 to 1. This metric will enable an evaluation of how accurately the answer encapsulates the target meaning. Furthermore, the model will also be evaluated using the percentage of unanswerable questions which are accurately identified. This will be calculated by taking the proportion of answerable questions which the model responds to by returning the unanswerable token, `[NO_ANS]`.

By using a combination of the above metrics, the overall model accuracy and performance, including word overlap, precision, recall, semantic meaning, and unanswerable response accuracy, will be evaluated.

Chapter 4

Results and Discussion

The results of each of the three models will now be compared, with discussion on both the qualitative answers and empirical evaluation where appropriate. The discussion will begin with the model trained using Masked Language Modelling (MLM), will be followed by the document store and GPT-3.5 Turbo model, and will end with the fine-tuned T5 model using the Natural Questions dataset. For ease of comparison, the same three questions will be used as a prompt for each model.

4.1 Masked Language Modelling

4.1.1 Empirical Results

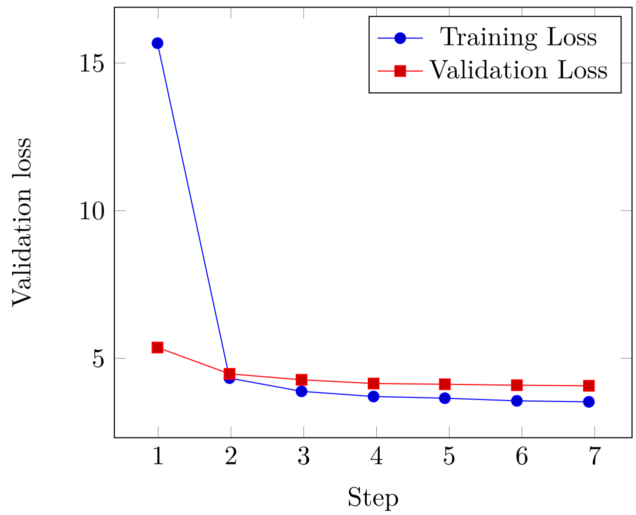
A GPT-2 model was fine-tuned using Masked Language Modelling as per Section 3.2. The validation metrics can be found in Table 4.1, with full training results and access to the Huggingface repo available in Appendix A.

Metric	**Value**
Training Loss	3.3968
Validation Loss	4.0711
Perplexity	58.62

Table 4.1: Training Results for GPT-2 Using MLM Fine-Tuning

The early-stopping callback was activated after almost 7 epochs, due to the validation loss failing to fall sufficiently further. The training loss and validation loss both finished the training process stubbornly high at 3.40 and 4.07 respectively. This is demonstrated graphically in Figure 4.1. The model had the potential to decrease the training loss further, but the validation loss reached a plateau, indicating possible overfitting to the training data. Furthermore, the associated perplexity value of 58.62 indicates the model's difficulty in accurately learning the context and semantic meaning of the text, or at least in learning topic-specific information as opposed to textbook-specific information.

Figure 4.1: Training and Validation Loss at Each Epoch using MLM



There were a number of factors causing the model's poor performance. Firstly, the model had only one textbook, consisting of 1000 pages, which included specific figures and interrelated chapters. Cleaning the dataset to remove these nuances might enhance training results, but it would require considerable effort and time constraints prevented this. Furthermore, the very poor results above did not justify further investigation. In particular, the training loss exhibited slow and limited improvements, indicating significant difficulties in learning from the training data.

4.1.2 Qualitative Output

To evaluate the model output and confirm the empirical results discussed above, a series of questions were given to the model.

As can be seen in Figure 4.2, the model answers using natural language which is grounded in the domain/context of the training data (textbook). However, unfortunately, the answer has several issues. Firstly, the model has failed to grasp the concept of Principal component analysis (PCA), and has given a vague and incorrect answer. Secondly, the answer contains words (tokens) which are not present in the training data (e.g. "NONAMBOW"). Finally, the model has not given a succinct answer and has been cutoff by the 150 token limit (introduced for demonstration purposes).

> ***What is principal component analysis?***
> *NONAMBOW: *The term principal component analysis is defined in the context of classifying a continuous graph as a matrix-by-matrix function of spatial and temporal dimensions. This approach is equivalent to a function of all principal components in a binary variable, but is more detailed. Lista: * Principal component analysis does not involve discrete spatial data, and spatial aspects of a continuous graph, but simply spatial images. NONAMB, M: *Principal, M, and T is the probability of defining a function from the binary variables. NONAMB: *Principal function analysis does not depend on discrete spatial data, but consists of discrete spatial features (such, pixels and color pixels*

Figure 4.2: Fine-tuned GPT-2's response to the prompt "What is principal component analysis?"

A second prompt asking about meaning of the acronym RGB is shown in Figure 4.3 and demonstrates a more promising response. The model provides a succinct natural-sounding answer grounded in the context of the training corpus but fails to give a correct answer. It also suffers from 'hallucinations' (see Section 2.6), providing a misleading colour range.

> ***What does RGB stand for?***
> *For example, on screen a black square with the color range 0, 1, 15, 60. Then you can see on retina, black squares are always a rectangle but there is no color in those black gray pixels.*

Figure 4.3: Fine-tuned GPT-2's response to the prompt "What does RGB stand for?"

It is also pertinent to ask a question which is not related to the training data, to observe how the model responds. This final example output is given in Figure 4.4, where the model gives an irrelevant, incorrect, and poorly formed answer. Additionally, it contains specific tokens from the training data, such as references and figure numbers. While the model could be improved by cleaning the training data to remove these artefacts, the model is still highly likely to respond poorly to irrelevant questions.

> ***Who is prime minister of the uk?***
> *(). These two concepts of representation are based on the representation of a set of subset of an integer or the set of elements defined in terms of [17] [20]. The function of representa d is the sum of every subsets of representatives in a set. All subset of an integer or set of elements is represented in an integer. EXAMPLE 54 Example 54a.1 (a) f (b) (c) (d) (e) (f) (g) (h) (i) (*

Figure 4.4: Fine-tuned GPT-2's response to the prompt "Who is prime minister of the uk?"

4.2 Document Store With GPT-3.5 Turbo

The framework using a document store and GPT-3.5 Turbo has no evaluation metrics, as the model has already been fine-tuned and is not available on open-source platforms. Therefore, only the qualitative output of the model will be discussed. The first question about PCA was posed to the model, with the response given in Figure 4.5.

> ***What is principal component analysis?***
> *Principal component analysis (PCA) is a statistical technique used for dimensionality reduction. It identifies the dominant subset of orthogonal directions in a feature space and discards the components that contribute little variance to the data. By using only the significant components, PCA can accurately approximate and describe the data in a more compact way. It is commonly used in multidimensional problems, such as analyzing digital images, where there may be tens, hundreds, or even thousands of variables. PCA calculates the principal axes, which are uncorrelated variables that maximize the variance of the data along each axis.*
>
> *To construct this answer, I used the following documents:*
>
> 1. *Fundamentals_of_Digital_Image_Processing_Textbook->Page(s)274/275: Depending on the precise context, there may be several closely related reasons for doing PCA, but...*
> 2. *Fundamentals_of_Digital_Image_Processing_Textbook->Page(s)265/266/267: To grasp the essence of PCA, suppose we have a large sample of M children and that we wish to ass...*
> 3. *Fundamentals_of_Digital_Image_Processing_Textbook->Page(s)267/268/269: Weight 90.57 73.43 w 1/4 69:42 Weight 1 0.57 Height 73.43 182.99 h 1/4 173:25 Heigh...*
> 4. *Fundamentals_of_Digital_Image_Processing_Textbook->Page(s)278/279/280/281: When we carry out a PCA on this ensemble of faces, we find that the eigenvalue spectrum dies off ...*

Figure 4.5: GPT-3.5 Turbo's Response to the Prompt "What is principal component analysis?"

The response is excellent for several reasons. Firstly, it succinctly provides an accurate answer using the sources. Secondly, the framework (outlined in Figure 3.1) enables the specific sources to be referenced so that any user/student can conduct further reading. Thirdly, and finally, the answer is grounded in the context of computer vision using a combination of the provided sources, such that the model does not suffer from 'hallucinations' and its output can be trusted.

The second example regarding the RGB colour model is shown in Figure 4.6 and demonstrates a good response, but has room for improvement.

While the model provided a correct answer, it stopped short of providing any additional detail, in part because of the phrasing of the question. Additionally, the model could have answered the question using only the first two sources, as the third and fourth sources contain no additional useful information, meaning that tokens were wasted which incurs a higher financial cost through using the API. Consequently, the framework could be improved by either increasing the cosine similarity needed to use a section of knowledge, reducing the token limit, or reducing the number of sources passed to the model.

The third and final example is the irrelevant question regarding the UK Prime Minister. The output in Figure 4.7 demonstrates how the framework/model can correctly identify questions which are beyond the

What does RGB stand for?
RGB stands for Red, Green, Blue.

To construct this answer, I used the following documents:

1. *Fundamentals_of_Digital_Image_Processing_Textbook->Page(s)28/29: RGB (or true colour) images are 3-D arrays that we may consider conceptually as three distinct 2-...*
2. *Digital_Image_Processing_Textbook->Page(s)408/409: In the RGB model, each color appears in its primary spectral components of red, green, and blue....*
3. *Digital_Image_Processing_Textbook->Page(s)412: As we have seen, creating colors in the RGB, CMY, and CMYK models, and chang- ing from one model ...*
4. *Digital_Image_Processing_Textbook->Page(s)409/410: The cube in Fig. 6.8 is a solid, composed of the ( ) 28 3 colors mentioned in the preceding para...*

Figure 4.6: GPT-3.5 Turbo's Response to the Prompt "What does RGB stand for?"

scope of the chatbot's topic and respond appropriately.

Who is prime minister of the uk?
I could not find an answer in the text I've been provided, sorry! Please try again.

Figure 4.7: GPT-3.5 Turbo's Response to the Prompt "Who is prime minister of the uk?"

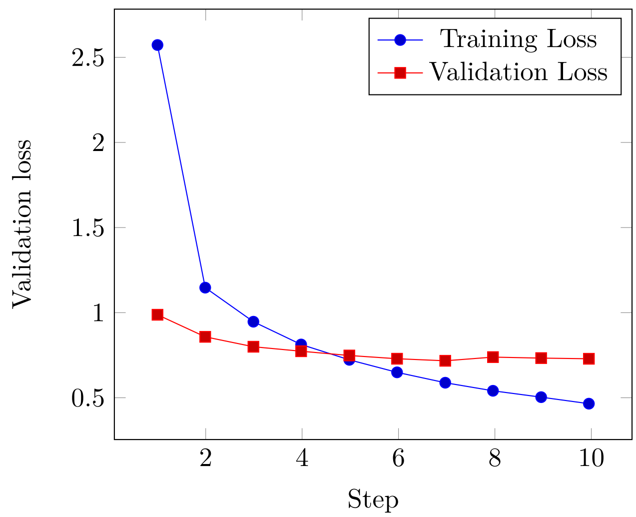
4.3 Document Store With Fine-Tuned T5

4.3.1 Empirical Results

The third and final model to discuss is the T5 model fine-tuned for question-answering. Figure 4.8 shows how the training loss had a steep decline in the first couple of epochs, and continued to fall even approaching 10 epochs. The validation loss, however, steadily declined until epoch 7. A small rise and plateau occurred after this, with the early stopping callback ending the training after almost 10 epochs. There is no clear sign of overfitting from this data, with the training and validation loss both below 1 and close together.

After almost 10 epochs, the model achieves excellent results. Table 4.2 shows that the model achieved an average cosine similarity of 0.762 between the generated answer and the target answer, with the ROUGE1 and ROUGE2 scores showing that uni- and bi-grams had an overlap of 44.4% and 38.8% respectively. Similar results using the BLEU and METEOR metrics confirm a significant overlap between generated answers and target answers. Encouragingly, the model can confidently distinguish between answerable and unanswerable questions, with only 10.4% of answerable questions being classified as unanswerable questions. However, there is room for improvement, with only 69.7% of unanswerable questions being detected; the model is

Figure 4.8: Training and Validation Loss at Each Epoch for Fine-Tuned T5



relatively likely (incorrectly) respond to a question which cannot be answered using the provided context. Additionally, the generated length of an answer is only 12.7 tokens, on average, indicating that the model may only produce short answers rather than provide explanatory content or reasoning.

Metric	**Value**
Training Loss	0.465
Validation Loss	0.729
ROUGE1	44.4
ROUGE2	38.8
ROUGEl	43.1
ROUGElsum	43.1
BLEU	34.2
Generated Length	12.7
METEOR	0.405
True negatives	69.7%
False negatives	10.4%
Cosine Similarity	0.763

Table 4.2: Validation Results for T5 Fine-Tuned for Question-Answering

4.3.2 Qualitative Output

While the evaluation results are very promising, most metrics are difficult to compare between different models, so the output from testing the model will provide the greatest insights. The first prompt and response is shown in Figure 4.9 and demonstrates the potential for false negatives. Even though the model was provided with the context to extract a correct answer, it failed to produce an answer.

The second prompt in Figure 4.10 demonstrates the model's ability to provide a factually accurate response

What is principal component analysis?
I could not find an answer in the text I've been provided, sorry! Please try again.

Figure 4.9: Fine-Tuned T5's Response to the Prompt "What is principal component analysis?"

regarding the acronym RGB. Furthermore, this framework benefits from accountability, with the source referenced for the user to explore. However, this answer corroborates the discussion in Section 4.3.1, with the answer being short and not identifying the correct intent of the question (the meaning of the acronym). This is a consequence of the training data and relatively succinct training answers that were created when enhancing the dataset (see Section 3.4.3).

What does RGB stand for?
RGB stands for the three primary colours mixed for display on a monitor or similar device.
To construct this answer, I used the following documents:

1. *Fundamentals_of_Digital_Image_Processing_Textbook->Page(s)28/29: RGB (or true colour) images are3-D arrays that we may consider conceptually as three distinct 2-...*

Figure 4.10: Fine-Tuned T5's Response to the Prompt "What does RGB stand for?"

The third and final prompt confirms the ability of the model to detect unanswerable questions, with it correctly identifying an irrelevant question given its 'knowledge' base, as shown in Figure 4.11

Who is prime minister of the uk?
I could not find an answer in the text I've been provided, sorry! Please try again.

Figure 4.11: Fine-Tuned T5's Response to the Prompt "Who is prime minister of the uk?"

4.4 Costs

GPT-3.5 Turbo demonstrates superior performance, but it comes at a higher cost compared to open-source models. As shown in Table 4.3, GPT-2 and T5 have no outright cost for usage (aside from the running costs: electricity, hardware, setup time etc.). The only model which has ongoing usage costs is GPT-3.5 Turbo. Although each query using this model is inexpensive, the cumulative cost for each module using the model may reach several hundred pounds per year. for each module that uses it, which might be unattainable for several departments. Additionally, departments would need to justify that this model is sufficiently more robust, reliable, and beneficial than ChatGPT, which is currently free (OpenAI, 2022b). Note that there is also a very small cost each time a query embedding is requested ($0.0004/1,000 tokens), but this is negligible compared to the cost of each query GPT, as the number of tokens per query is very small (typically under 50).

Table 4.3: Comparison of Estimated AI Costs

Model	Cost per query[1]	Cost per student[2]	Cost per module[3]
GPT-2	$0.000	$0.00	$0
GPT-3.5 Turbo	$0.004	$0.40	$120[4]
T5	$0.000	$0.00	$0

[1] Estimated cost using based on each query using 2000 tokens [2] Estimated cost using based on 100 queries per student
[3] Estimated cost using based on 300 students in a module [4] Excluding the small cost of embedding students' questions

4.5 Applications

The framework used by the two most impressive models, GPT-3.5 Turbo and fine-tuned T5, can be used in an almost limitless number of applications. Neither model is fined-tuned to work only on a specific domain, and rely solely on the input documents. In the context of educational use, academics need to only provide a few high-quality text-based documents (such as textbooks or detailed lecture notes) and the framework will chunk them into sections for use with the chosen AI model. Therefore, this framework can also be used in a variety of applications: customer service, general FAQ services, healthcare, legal advice, technical support and more.

The custom fine-tuned T5 chatbot (see Section 3.4) has no ongoing dependency on GPT or OpenAI in general. It is completely open-source (if GPT embeddings are not used) and can be run without a GPU.

4.6 Limitations

There are several limitations to the findings in this paper. Firstly, T5 was the only model fine-tuned for question-answering, and was chosen because of the availability of the 'small' variant. However, other models such as BART are available and could have produced better outcomes.

The second limitation pertains to the evaluation process. Due to limited access to the high-performance GPUs required to fine-tune the models, nested cross-validation was not used to provide a generalisation error. Therefore, the results may not be a reliable estimate of the generalisation error. If more resources were available for this research, the generalisation error and optimal hyperparameters could be found using grid search or otherwise, which could have also improved model performance.

Qualitative output from the T5 model demonstrates the model can claim that a question cannot be answered using the context, even when it can. Additionally, answers often fail to contain explanatory reasoning, which is likely to be a consequence of the dataset used and the quality of answers. Use of an improved dataset with more diverse, higher quality answers could improve the model further.

The final limitation is that the GPT-3.5 Turbo model requires ongoing use of the OpenAI API, which may be discontinued in the future.

Chapter 5

Summary and Reflections

This paper has outlined a pioneering framework that adapts fine-tuning techniques to produce a question-answering chatbot. It has produced and critically analysed three potential approaches to creating a chatbot which is both reliable and limited to answering from specific documents. The framework presented by this paper offers promising results and potential for fine-tuning a long-form, closed-domain question-answering chatbot, which has not yet been directly attempted by the literature.

The main challenge with the use of a novel fine-tuning approach was a lack of naturally-worded, long-form, generative (and not extractive) answers. By using existing fine-tuned Artificial Intelligence models, specifically GPT, to enhance the existing Natural Questions dataset, (Kwiatkowski et al., 2019), this paper generated and has contributed to the literature by providing a novel dataset of naturally worded questions with succinct answers written in natural language. An improved dataset with human-generated answers (as opposed to AI-generated ones) would likely bring improvements to this model.

While the use of pre-trained Transformer models for creating chatbots is not novel, the combination of fine-tuning techniques and document store approaches, along with the comparison of different models, brings originality to this work. By using an innovative application of a (typically) summarisation training method, this paper has shown that it is possible to fine-tune an open-source pre-trained model to achieve accurate and reliable answers. While the fine-tuned T5 model produces imperfect responses, further improvement of such models could be achieved by enhancing and expanding the training dataset to generate highly effective responses. Access to high-performance GPU resources could yield further improvements through the use of higher batch sizes, hyperparemeter tuning, and nested cross-validation.

The proposed framework has a vast number of applications. Due to the model fine-tuning emphasis being on question-answering, rather than the knowledge of a specific domain, it can very easily be used in a variety of contexts. This model input can be utilised to provide relevant ‘knowledge’ with each query, rather than attempting to retain information through the neural network weights. This approach benefits from more reliable answers and the ability to reference the sources used to provide an answer, unlike most state-of-the-art LLMs. These developments have the potential to improve the reliability and accuracy of chatbots, which can be used for students and academics, commerce, customer support, healthcare and more.

For applications of the framework proposed in this paper, the model choice depends on the value placed on the quality of human-like responses. The state-of-the-art API managed by OpenAI can produce excellent answers. While the fine-tuned T5 model produces very good numerical outcomes, the actual output from the model can lack substance, and will require an improved dataset to become reliable enough.

This paper has shown the potential to produce a reliable long-form, closed-domain chatbot and offered a framework to do so. Further development of such a model could have applications in a wide range of fields, and lead to vastly improved outcomes for its users.

5.1 Future Research

This paper has demonstrated promising results and the potential to use open-source Transformer models to create a long-form, closed-domain question-answering chatbot. However, there are a number of limitations and avenues for potential development. Firstly, the fine-tuned T5 question-answering model was trained on a synthetically enhanced dataset (see Section 3.4.3), which seemed to limit its ability to understand the full meaning of a question and consistently extract the correct answer. The creation of a human-written dataset, as opposed to an AI-generated one, could vastly improve the potential success of the model. Additionally, adapting the dataset and framework to include tables, lists, formulas, and images would present a difficult but valuable improvement.

Secondly, a key limitation of this paper pertains to the sequence-to-sequence fine-tuning of only one model family (T5). Time constraints, combined with limited access to GPU resources, prevented a comparison with other models, such as BART. However, the framework and code provided enables alternative models to be trained, making it simple to compare the performance of different models.

A potential area for improvement could be combining multiple fine-tuning approaches. Improved results could be achieved by fine-tuning for question answering in conjunction with additional Masked Language Modelling. However, this approach would mean that such a chatbot would need to be fine-tuned for each module, as opposed to having one fine-tuned chatbot that can be used in a variety of applications.

Finally, this paper does not address the impact of a chatbot in practice, particularly in the field of education. Additional research regarding the accuracy of answers (e.g. information about assessments and deadlines), whether it is possible to cheat, and whether it is used responsibly, is essential. Use of the framework in this paper requires rigorous testing to identify the reliability of any model's responses and their impact on users.

Bibliography

Abiodun, O. I., A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, and H. Arshad (2018): "State-of-the-art in artificial neural network applications: A survey," *Heliyon*, 4, e00938, `https://doi.org/10.1016/j.heliyon.2018.e00938`.

Allen, C. and T. Hospedales (2019): "Analogies Explained: Towards Understanding Word Embeddings," ArXiv preprint: `https://doi.org/10.48550/arXiv.1901.09813`.

Banerjee, S. and A. Lavie (2005): "METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments," in *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, Ann Arbor, Michigan: Association for Computational Linguistics, 65–72.

Bender, E. M., T. Gebru, A. McMillan-Major, and S. Shmitchell (2021): "On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? 🦜," in *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, New York, NY, USA: Association for Computing Machinery, FAccT '21, 610–623, `https://doi.org/10.1145/3442188.3445922`.

Bengio, Y., R. Ducharme, and P. Vincent (2000): "A neural probabilistic language model," *Advances in neural information processing systems*, 13.

Bishop, C. M. (1994): "Neural networks and their applications," *Review of Scientific Instruments*, 65, 1803–1832, `https://doi.org/10.1063/1.1144830`.

Bisong, E. (2019): *Google Colaboratory*, Berkeley, CA: Apress, 59–64, `https://doi.org/10.1007/978-1-4842-4470-8_7`.

Bradeško, L. and D. Mladenić (2012): "A Survey of Chatbot Systems through a Loebner Prize Competition," in *Proceedings of Slovenian language technologies society eighth conference of language technologies*, vol. 2, 34–37.

Brown, M., M. McCormack, J. Reeves, D. C. Brook, S. Grajek, B. Alexander, M. Bali, S. Bulger, S. Dark, N. Engelbert, K. Gannon, A. Gauthier, D. Gibson, R. Gibson, B. Lundin, G. Veletsianos, and N. Weber (2020a): "2020 Educause Horizon Report Teaching and Learning Edition," Tech. rep., EDUCAUSE, [online] Available at: `www.learntechlib.org/p/215670`.

Brown, T. B., B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei (2020b): "Language Models are Few-Shot Learners," ArXiv preprint: `https://doi.org/10.48550/arXiv.2005.14165`.

Cahuantzi, R., X. Chen, and S. Güttel (2023): "A comparison of LSTM and GRU networks for learning symbolic sequences," ArXiv preprint: `https://doi.org/10.48550/arXiv.2107.02248`.

Chang, Y., X. Wang, J. Wang, Y. Wu, K. Zhu, H. Chen, L. Yang, X. Yi, C. Wang, Y. Wang, W. Ye, Y. Zhang, Y. Chang, P. S. Yu, Q. Yang, and X. Xie (2023): "A Survey on Evaluation of Large Language Models," ArXiv preprint: `https://doi.org/10.48550/arXiv.2307.03109`.

Chase, H., D. Chase, Bagatur, Z. Chase, W. FH, and L. Ganeline (2023): "LangChain," Open source Python package with 1,000+ contributors [online] Available at: `https://github.com/langchain-ai/langchain`. [accessed: 15/06/2023].

ChatGPT (2023): "Ethical concerns associated with the use of chatbots in education," Retrieved on 21/04/2023, from `https://www.chatgpt.com`, unpublished raw data.

Chen, P., J. Zhong, and Y. Zhu (2022): "Intelligent Question Answering System by Deep Convolutional Neural Network in Finance and Economics Teaching," *Comput Intell Neurosci. 2022 Jan 21;2022:5755327. doi: 10.1155/2022/5755327. eCollection 2022.*, [`https://doi.org/10.1155%2F2022%2F5755327`].

Chen, T., B. Xu, C. Zhang, and C. Guestrin (2016): "Training Deep Nets with Sublinear Memory Cost," ArXiv preprint: `https://doi.org/10.48550/arXiv.1604.06174`.

Cho, K., B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio (2014): "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation," ArXiv preprint: `https://doi.org/10.48550/arXiv.1406.1078`.

Chung, J., C. Gulcehre, K. Cho, and Y. Bengio (2014): "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling," ArXiv preprint: `https://doi.org/10.48550/arXiv.1412.3555`.

Cotton, D., P. Cotton, and R. Shipway (2023): "Chatting and Cheating. Ensuring academic integrity in the era of ChatGPT," `http://doi.org/10.35542/osf.io/mrz8h`.

Devlin, J., M.-W. Chang, K. Lee, and K. Toutanova (2019): "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," ArXiv preprint: `https://doi.org/10.48550/arXiv.1810.04805`.

Durmus, E., H. He, and M. Diab (2020): "FEQA: A Question Answering Evaluation Framework for Faithfulness Assessment in Abstractive Summarization," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, `https://doi.org/10.18653%2Fv1%2F2020.acl-main.454`.

Floridi, L. and M. Chiriatti (2020): "GPT-3: Its Nature, Scope, Limits, and Consequences," *Minds and Machines*, 30, 1–14, `https://doi.org/10.1007/s11023-020-09548-1`.

Gers, F. A., J. Schmidhuber, and F. Cummins (2000): "Learning to Forget: Continual Prediction with LSTM," *Neural Computation*, 12, 2451–2471, `https://doi.org/10.1162/089976600300015015`.

Geva, M., Y. Goldberg, and J. Berant (2019): "Are We Modeling the Task or the Annotator? An Investigation of Annotator Bias in Natural Language Understanding Datasets," ArXiv preprint: `https://doi.org/10.48550/arXiv.1908.07898`.

Gonzalez, R. C. and R. E. Woods (2018): *Digital Image Processing*, Pearson Prentice Hall, 4th ed.

Guu, K., K. Lee, Z. Tung, P. Pasupat, and M. Chang (2020): "Retrieval Augmented Language Model Pre-Training," in *Proceedings of the 37th International Conference on Machine Learning*, ed. by H. D. III and A. Singh, PMLR, vol. 119 of *Proceedings of Machine Learning Research*, 3929–3938, `https://doi.org/10.48550/arXiv.2002.08909`.

Harris, Z. S. (1954): "Distributional Structure," *<i>WORD</i>*, 10, 146–162, `https://doi.org/10.1080/00437956.1954.11659520`.

Hochreiter, S. and J. Schmidhuber (1997): "Long short-term memory," *Neural computation*, 9, 1735–1780, `http://doi.org/10.1162/neco.1997.9.8.1735`.

Huang, Z., B. Jiang, T. Guo, and Y. Liu (2023): "Measuring the Impact of Gradient Accumulation on Cloud-based Distributed Training," in *2023 IEEE/ACM 23rd International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, 344–354, https://doi.org/10.1109/CCGrid57682.2023.00040.

Irie, K., Z. Tüske, T. Alkhouli, R. Schlüter, and H. Ney (2016): "LSTM, GRU, Highway and a Bit of Attention: An Empirical Overview for Language Modeling in Speech Recognition," 3519–3523, http://doi.org/10.21437/Interspeech.2016-491.

Jungherr, A. (2023): "Using ChatGPT and Other Large Language Model (LLM) Applications for Academic Paper Assignments," Otto-Friedrich-Universität https://doi.org/10.31235/osf.io/d84q6.

Kanakaraj, M. and R. M. R. Guddeti (2015): "NLP based sentiment analysis on Twitter data using ensemble classifiers," 1–5, https://doi.org/10.1109/ICSCN.2015.7219856.

Khan, S. (2023): "Harnessing AI So That All Students Benefit: A Nonprofit Approach for Equal Access," Khan Academy Blog [online] Available at: https://blog.khanacademy.org/harnessing-ai-so-that-all-students-benefit-a-nonprofit-approach-for-equal-access/, [accessed: 04/05/2023].

Kwiatkowski, T., J. Palomaki, O. Redfield, M. Collins, A. Parikh, C. Alberti, D. Epstein, I. Polosukhin, J. Devlin, K. Lee, K. Toutanova, L. Jones, M. Kelcey, M.-W. Chang, A. M. Dai, J. Uszkoreit, Q. Le, and S. Petrov (2019): "Natural Questions: A Benchmark for Question Answering Research," *Transactions of the Association for Computational Linguistics*, 7, 453–466, https://doi.org/10.1162/tacl_a_00276.

LeCun, Y., Y. Bengio, and G. Hinton (2015): "Deep Learning," *Nature*, 521, 436–44.

Lewis, M., Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer (2019): "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension," ArXiv preprint: https://doi.org/10.48550/arXiv.1910.13461.

Li, S. (2018): "Introduction to Word Embedding and Word2Vec," Towards Data Science [online] Available at: https://towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa, [accessed: 03/05/2023].

Lin, C.-Y. (2004): "ROUGE: A Package for Automatic Evaluation of Summaries," in *Text Summarization Branches Out*, Barcelona, Spain: Association for Computational Linguistics, 74–81.

Lipton, Z. C., J. Berkowitz, and C. Elkan (2015): "A Critical Review of Recurrent Neural Networks for Sequence Learning," *CoRR*, abs/1506.00019, arXiv preprint: https://dblp.org/rec/journals/corr/Lipton15.bib.

Lund, B. D. and T. Wang (2023): "Chatting about ChatGPT: How may AI and GPT impact academia and libraries?" *Library Hi Tech News*, 40, 20–27, [online] Available at: http://doi.org/10.2139/ssrn.4333415.

Manakul, P., A. Liusie, and M. J. F. Gales (2023): "SelfCheckGPT: Zero-Resource Black-Box Hallucination Detection for Generative Large Language Models," ArXiv preprint: https://doi.org/10.48550/arXiv.2303.08896.

Marcus, G. (2018): "Deep Learning: A Critical Appraisal," ArXiv preprint: https://doi.org/10.48550/arXiv.1801.00631.

May, C., A. Wang, S. Bordia, S. R. Bowman, and R. Rudinger (2019): "On Measuring Social Biases in Sentence Encoders," ArXiv preprint: https://doi.org/10.48550/arXiv.1903.10561.

Mikolov, T., K. Chen, G. Corrado, and J. Dean (2013): "Efficient Estimation of Word Representations in Vector Space," ArXiv preprint: https://doi.org/10.48550/arXiv.1301.3781.

Nee, C. K., M. H. A. Rahman, N. Yahaya, N. H. Ibrahim, R. A. Razak, and C. Sugino (2023): "Exploring the Trend and Potential Distribution of Chatbot in Education: A Systematic Review," *International Journal of Information and Education Technology*, http://doi.org/10.18178/ijiet.2023.13.3.1834.

Nielsen, M. A. (2015): "Neural Networks and Deep Learning," *Determination Press*.

Olah, C. (2015): "Understanding LSTM Networks," Colah's Blog [online] Available at: https://colah.github.io/posts/2015-08-Understanding-LSTMs/, [accessed: 02/05/2023].

Ondáš, S., M. Pleva, and D. Hládek (2019): "How chatbots can be involved in the education process," in *2019 17th International Conference on Emerging eLearning Technologies and Applications (ICETA)*, 575–580, https://doi.org/10.1109/ICETA48886.2019.9040095.

OpenAI (2022a): "Embeddings," OpenAI [online] Available at: https://platform.openai.com/docs/guides/embeddings/limitations-risks, [accessed: 29/05/2023].

——— (2022b): "Introducing ChatGPT," OpenAI [online] Available at: https://openai.com/blog/chatgpt, [accessed: 13/06/2023].

OpenAI (2022): "OpenAI Cookbook: Question Answering Using Embeddings," https://github.com/openai/openai-cookbook/blob/main/examples/Question_answering_using_embeddings.ipynb, gitHub Repository, [accessed 04/05/2023].

Papineni, K., S. Roukos, T. Ward, and W.-J. Zhu (2002): "Bleu: a Method for Automatic Evaluation of Machine Translation," in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, 311–318.

Pearce, K., T. Zhan, A. Komanduri, and J. Zhan (2021): "A comparative study of transformer-based language models on extractive question answering," ArXiv preprint: https://doi.org/10.48550/arXiv.2110.03142.

Pennington, J., R. Socher, and C. Manning (2014): "GloVe: Global Vectors for Word Representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar: Association for Computational Linguistics, 1532–1543, http://doi.org/10.3115/v1/D14-1162.

Pérez, J. Q., T. Daradoumis, and J. M. M. Puig (2020): "Rediscovering the use of chatbots in education: A systematic literature review," *Computer Applications in Engineering Education*, 28, 1549 – 1565, https://doi.org/10.1002/cae.22326.

Piazza (n.d.): "About Piazza: Our Story," [online] Available at: https://piazza.com/about/story, [accessed: 21/04/2023].

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

Rathi, N. and K. Roy (2020): "DIET-SNN: Direct Input Encoding With Leakage and Threshold Optimization in Deep Spiking Neural Networks," ArXiv preprint: https://doi.org/10.48550/arXiv.2008.03658.

Reimers, N. and O. Espejel (2021): "Huggingface (sentence-transformers/all-mpnet-base-v2)," https://huggingface.co/sentence-transformers/all-mpnet-base-v2.

Rudinger, R., J. Naradowsky, B. Leonard, and B. V. Durme (2018): "Gender Bias in Coreference Resolution," ArXiv preprint: https://doi.org/10.48550/arXiv.1804.09301.

Rumelhart, D. E., G. E. Hinton, and R. J. Williams (1986): "Learning representations by back-propagating errors." *Nature*, 323, 533–536, https://doi.org/10.1038/323533a0.

SCHLUTER, N. (2017): "The limits of automatic summarisation according to ROUGE," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, Valencia, Spain: Association for Computational Linguistics, 41–45, https://doi.org/10.18653/V1%2FE17-2007.

SCHUSTER, M. AND K. PALIWAL (1997): "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, 45, 2673–2681, https://doi.org/10.1109/78.650093.

SHARMA, S. (2017): "Epoch vs Batch Size vs Iterations," Towards Data Science [online] Available at: https://towardsdatascience.com/epoch-vs-iterations-vs-batch-size-4dfb9c7ce9c9, [accessed: 01/05/2023].

SHUKLA, L. (2019): "Fundamentals of Neural Networks," Weights & Biases [Online Image] Available at: https://wandb.ai/site/articles/fundamentals-of-neural-networks, accessed: 01/05/2023.

SIVAKUMAR, S., L. S. VIDELA, T. R. KUMAR, J. NAGARAJ, S. ITNAL, AND D. HARITHA (2020): "Review on word2vec word embedding neural net," in *2020 international conference on smart electronics and communication (ICOSEC)*, IEEE, 282–290, https://doi.org/10.1109/ICOSEC49089.2020.9215319.

SOLOMON, C. AND T. BRECKON (2010): *Fundamentals of Digital Image Processing: A Practical Approach with Examples in Matlab*, Wiley.

SUSNJAK, T. (2022): "ChatGPT: The End of Online Exam Integrity?" ArXiv preprint: https://doi.org/10.48550/arXiv.2212.09292.

UDOFIA, U. (2018): "Basic Overview of Convolutional Neural Network (CNN)," Medium [Online Image] Available at: https://medium.com/dataseries/basic-overview-of-convolutional-neural-network-cnn-4fcc7dbb4f17, [accessed: 01/05/2023].

VAN DIS, E. A. M., J. BOLLEN, W. ZUIDEMA, R. VAN ROOIJ, AND C. L. BOCKTING (2023): "ChatGPT: five priorities for research," *Nature, 614(7947), pp.224-226.*, https://doi.org/10.1038/d41586-023-00288-7.

VASWANI, A., N. SHAZEER, N. PARMAR, J. USZKOREIT, L. JONES, A. N. GOMEZ, L. KAISER, AND I. POLOSUKHIN (2017): "Attention Is All You Need," *CoRR*, https://doi.org/10.48550/arXiv.1706.03762.

WOLF, T., L. DEBUT, V. SANH, J. CHAUMOND, C. DELANGUE, A. MOI, P. CISTAC, T. RAULT, R. LOUF, M. FUNTOWICZ, J. DAVISON, S. SHLEIFER, P. VON PLATEN, C. MA, Y. JERNITE,

J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush (2020): "🤗 HuggingFace's Transformers: State-of-the-art Natural Language Processing," ArXiv preprint: https://doi.org/10.48550/arXiv.1910.03771.

Wu, Y., M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, Łukasz Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean (2016): "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation," ArXiv preprint: https://doi.org/10.48550/arXiv.1609.08144.

Yathish, V. (2022): "Loss Functions and Their Use In Neural Networks," Towards Data Science [online] Available at: https://towardsdatascience.com/loss-functions-and-their-use-in-neural-networks-a470e703f1e9, accessed: 01/05/2023.

Zawacki-Richter, O., V. I. Marín, M. Bond, and F. Gouverneur (2019): "Systematic review of research on artificial intelligence applications in higher education–where are the educators?" *International Journal of Educational Technology in Higher Education*, 16, 1–27, https://doi.org/10.1186/s41239-019-0171-0.

ZEMČÍK, M. T. (2019): "A brief history of chatbots," *DEStech Transactions on Computer Science and Engineering*, 10, http://doi.org/10.12783/dtcse/aicae2019/31439.

Appendix A

MLM Training Results

Training results for the model in Section 4.1. Full details including hyperparameters, virtual environment framework, and usage can be found at the Huggingface repo: `https://huggingface.co/psxjp5/mlm`.

Train Loss	**Epoch**	**Val Loss**	**Perplexity**
12.948	0.99	5.147	171.9
4.061	1.98	4.311	74.5
3.713	2.97	4.081	59.2
3.603	3.96	4.055	57.7
3.503	4.94	4.051	57.5
3.443	5.93	4.088	59.6
3.397	6.92	4.071	58.6

Table A.1: Full Training Results for the Fine-Tuned GPT Using Masked Language Modelling

Appendix B

Fine-tuned T5 Training Results

Training results for the model in Section 4.3. Full details including hyperparameters, virtual environment framework, and usage can be found at the Huggingface repo: `https://huggingface.co/psxjp5/mt5-small`.

Train Loss	**Epoch**	**Val Loss**	**Rouge1**	**Rouge2**	**Rougel**	**Rougel-sum**	**Bleu**	**Gen Len**	**Meteor**	**True Negatives**	**False Negatives**	**Cosine Sim**
2.572	1	0.988	18.8	15.6	18.2	18.3	7.7	7.8	0.163	72.9%	56.7%	0.400
1.147	1.99	0.858	36.8	31.3	35.5	35.5	25.7	12.0	0.331	62.8%	20.4%	0.665
0.947	2.99	0.800	40.4	34.7	39.1	39.1	29.3	12.4	0.366	63.4%	15.3%	0.711
0.813	3.98	0.773	42.7	36.7	41.2	41.3	32.1	12.9	0.387	62.2%	11.4%	0.743
0.723	4.98	0.748	42.9	37.0	41.5	41.5	32.5	12.9	0.391	63.3%	11.5%	0.747
0.649	5.97	0.729	40.3	35.0	39.1	39.1	28.8	11.7	0.367	73.9%	18.0%	0.707
0.588	6.97	0.717	42.7	37.1	41.4	41.4	32.1	12.5	0.389	70.0%	12.8%	0.739
0.541	7.96	0.739	44.7	38.8	43.3	43.3	34.5	12.9	0.408	66.3%	9.5%	0.766
0.504	8.96	0.733	43.5	38.0	42.3	42.2	32.6	12.3	0.398	72.6%	12.8%	0.745
0.465	9.95	0.729	44.4	38.8	43.1	43.1	34.2	12.7	0.405	69.7%	10.4%	0.763

Table B.1: Full Training Results for the Fine-Tuned T5