Gradual Unfreezing Transformer-based Language Models for Biomedical Question Answering

By

Urvashi Khanna

A thesis submitted to Macquarie University

for the degree of Master of Research

Department of Computing

April 2021

Examiner's Copy

Typeset in LaTeX 2_ε.

Statement of Originality

This work has not previously been submitted for a degree or diploma in any university. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made in the thesis itself.

Urvashi Khanna

Acknowledgements

I would like to convey my sincerest gratitude to my supervisor Dr. Diego Mollá Aliod for his continued support and guidance throughout the course of this thesis.

This research was undertaken with the assistance of resources from the National Computational Infrastructure (NCI Australia), an NCRIS enabled capability supported by the Australian Government.

I dedicate this thesis to my loving and supportive husband, Sandeep, who has always been a source of inspiration and encouragement. To my wonderful daughters, Asmi and Saanvi, for their enduring love and for being my enthusiastic cheerleaders.

Abstract

Pretrained transformer-based language models have achieved state-of-the-art results on various Natural Language Processing (NLP) tasks. These models can be fine-tuned on a range of downstream tasks with minimalistic modifications. However, fine-tuning a language model may result in the problem of catastrophic forgetting and tend to overfit on smaller training datasets. Therefore, gradually unfreezing the pretrained weights is a possible approach to avoid catastrophic forgetting of the knowledge learnt from the source task. Multi-task fine-tuning is an intermediate step on a high-resource dataset that yields good results for low-resource tasks. In this project, we will be investigating the strategies of multi-task fine-tuning and gradual unfreezing on DistilBERT, which have not yet been applied for biomedical domain. First, we explore whether DistilBERT improves the accuracy of a low-resource dataset, BioASQ, with question answering (QA) task as our NLP use-case. Second, we investigate the effect that gradual unfreezing has on the performance of DistilBERT. We observe that despite being 40% smaller and without any domain-specific pretraining, DistilBERT achieves comparable results to a larger model, BERT on smaller BioASQ dataset. However, we observed that gradually unfreezing DistilBERT has no significant impact on the results of our QA task in comparison to standard non-gradual fine-tuning.

Contents

List of Figures

List of Tables

List of Abbreviations

NLP	Natural Language Processing
CV	Computer Vision
RNN	Recurrent Neural Networks
LSTM	Long Short Term Memory
seq2seq	sequence-to-sequence learning
BERT	Bidirectional Encoder Representations from Transformers
GPT	Generative Pre-Training
GPT-2	Generative Pre-Training 2
ELECTRA	Efficiently Learning an Encoder that Classifies Token Replacements Accurately
ALBERT	A Lite BERT
RoBERTa	Robustly optimized BERT approach
DistilBERT	Distilled BERT
ELMo	Embeddings from Language Models
LM	Language Model
MLM	Masked Language Model
NLI	Natural Language Inference
QA	Question Answering
CLR	Cyclical Learning Rate
STLR	Slanted Triangular Learning Rates
GLUE	General Language Understanding Evaluation
SQuAD	Stanford Question Answering Dataset
GPU	Graphics Processing Unit
NQ	Natural Questions

CoQA Conversational Question Answering

MeSH Medical Subject Headings

URL Uniform Resource Locator

SAcc Strict Accuracy

LAcc Lenient Accuracy

MRR Mean Reciprocal Rank

TREC Text REtrieval Conference

NCI National Computational Infrastructure

IoT Internet of Things

1

Introduction

Humans have the inherent ability to transfer the skills learned from one task to another related task. For example, the road rules learnt when learning to ride a bike can be reused for driving a car. This transfer of knowledge across tasks has relieved us from the hassle of acquiring the knowledge from scratch each time. Along similar lines, machine learning has also transitioned from the isolated paradigm of learning to transferring knowledge across tasks [7, 27].

In the context of Natural Language Processing (NLP), pretrained language models enable this transfer of knowledge. Language models that are pretrained on vast volumes of data have become a norm these days [8, 24, 28]. These pretrained models can be fine-tuned for a variety of tasks with minor modifications like adding a single additional output layer instead of major task-specific architectural changes. The fine-tuning techniques are quintessential for better adaptation of the language model and to learn the distributions of target tasks. Different fine-tuning techniques like gradual unfreezing, discriminative fine-tuning have yielded improvements on various NLP tasks [17].

Language models are pretrained on general language and adapted to the target tasks of

different domains. These domain-specific tasks have the universal problem of limited availability of manually labelled data. For better adaptation to these target tasks, additional signal in the form of multi-task fine-tuning has improved the quality of different systems [60, 11, 19]. This intermediary step of fine-tuning on a large related dataset prior to fine-tuning on target data has proven to be effective on various NLP tasks.

In this thesis, we focus on a low-resource dataset from the biomedical domain with question answering task as our NLP use case. We apply and investigate the techniques of gradual unfreezing and multi-task fine-tuning on the pretrained transformer-based language model, DistilBERT. We are also interested in understanding how a compact language model, DistilBERT, adapts to both the low-resource BioASQ dataset and the fine tuning techniques mentioned above.

1.1 Objectives

Our primary goal is to improve the accuracy of biomedical question answering task using fine-tuning techniques. Our secondary goal is to analyse the impact of these fine-tuning techniques on a smaller transformer-based language model for a low resource dataset.

Our thesis is built upon two research questions:

1. Can a smaller model, DistilBERT, achieve good results on a low-resource dataset, BioASQ?
2. Does gradual unfreezing of DistilBERT improve the quality of biomedical question answering task?

1.2 Contributions

In this thesis, we provide a comprehensive literature review on sequential transfer learning that covers all the current research and key developments. We detail the fine-tuning strategies and provide a manual that other niche domains with smaller datasets can use. We explore an unfreezing approach that makes use of 34% fewer trainable DistilBERT parameters. We discover that smaller language models viz. DistilBERT is a good alternative to large models for smaller datasets. From our experiments, we observe that DistilBERT achieved comparable results to a larger model, BERT.

1.3 Outline

This thesis has been structured in the following format. Chapter 1 provides a brief introduction along with our objectives and contribution. Chapter 2 details our background and literature review. Chapter 3 provides the details of the methodology used to answer both our research questions. Chapter 4 details our experimental setup and discusses the results of the experiments of fine-tuning the BioASQ task. Chapter 5 discusses the results of the experiments of gradual unfreezing the BioASQ task. Lastly, Chapter 6 provides a conclusion to our research and lists the future pathways of research.

2

Background and Literature Review

The traditional machine learning models were initially developed to work in isolation on specific tasks. They were usually designed from scratch for a particular task and trained to achieve good results. In recent years, complex deep learning algorithms require huge amount of labelled data to improve the learning, which might not be available for some tasks and domains. Manually annotating the data can be an exhaustive, time consuming and expensive process. Thus, the need of the hour was to forgo an isolated learning process and allow the transfer of knowledge acquired from one task to another. This utilisation of what has been learned in one setting to improve the ability to generalise better in another setting is Transfer Learning [13].

Motivated by the success of transfer learning in Computer Vision (CV), NLP research has also moved in this direction. The availability of huge amounts of raw text on the web and the ever increasing computational power are the driving forces behind the use of unsupervised pretraining for language understanding tasks.

In this chapter, we will discuss the advancements of transfer learning particularly in the context of deep learning and the language understanding task of Biomedical Question Answering.

2.1 Sequential Transfer Learning

Transfer learning is the process of leveraging the knowledge acquired from one system to other system to improve the performance and efficiency of the target system as illustrated in Figure 2.1.

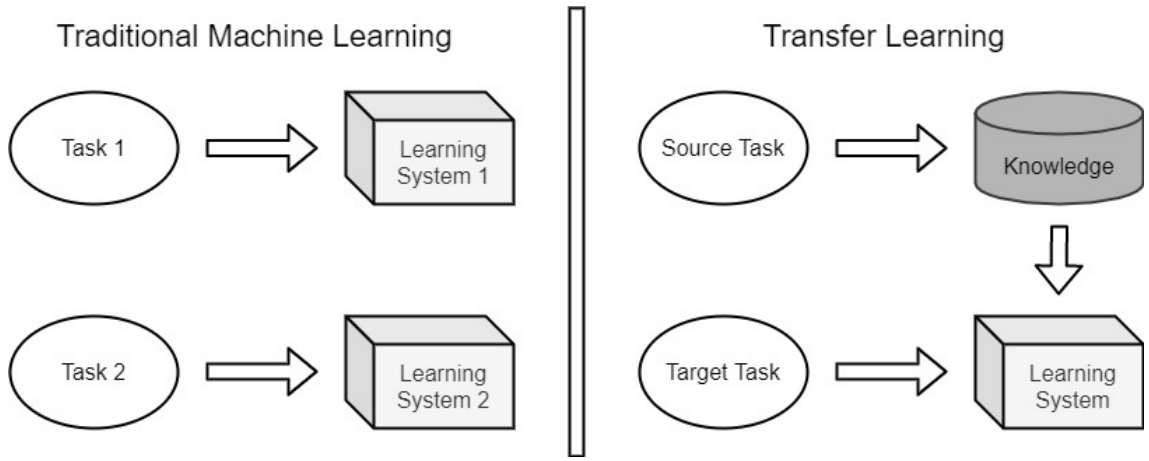

FIGURE 2.1: Learning process comparison of traditional machine learning and transfer learning.

Transfer learning can be broadly classified based on the three scenarios listed below [53]:

- whether the source and the target domains are same or different.
- the type of source and target task.
- the sequence in which the tasks are learnt.

As depicted in the Figure 2.2, *Inductive Transfer Learning* is the setting where the target domain has labelled data, and source and target tasks are different. On the other hand, *Transductive Transfer Learning* involves the scenario where the source and target task is the same and only the source domain has labelled data.

Depending on the order in which the tasks are learnt, inductive transfer learning can further be categorised into *Multi-task Learning* and *Sequential Learning*. Multiple tasks learnt at the same time is multi-task learning and when tasks are learnt sequentially it is sequential transfer learning. In this report, we focus on sequential transfer learning [53], particularly in the context of deep learning.

Computer Vision (CV) is one of the latest success stories of sequential transfer learning. Most machine learning models for CV tasks like object recognition, image classification and

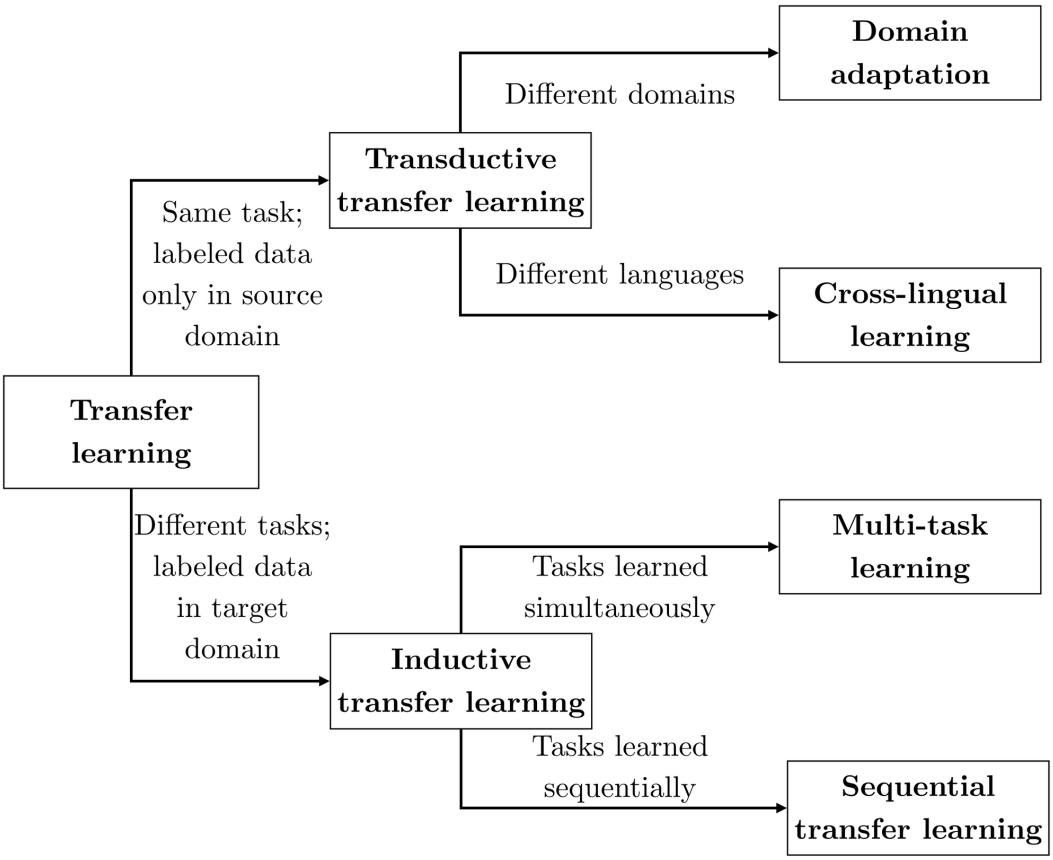

FIGURE 2.2: Categorization of NLP transfer learning [53].

image segmentation are not trained from scratch but they reuse the knowledge learnt from already existing pretrained models on large datasets like Imagenet Dataset [7] and MS-COCO [27].

Sequential learning is the transfer learning setting where the source and target tasks are different and the training process is carried out sequentially. This implies that the models are trained and learn separately. The ultimate aim of sequential transfer learning is that the target model generalises better on the target task by using the information acquired from the model trained on the source task [53]. It is found to be quite effective in scenarios where the source task has huge amount of data compared to target task and many target tasks are available to adapt to the representations learnt from the source task.

Sequential transfer learning [39] consists of two phases: *pretraining* and *adaptation* [53]. In the first phase, the model is pretrained on the source task; and in the adaptation phase the knowledge gained by the pretrained model is passed on to the target task. The pretraining phase usually has a substantial cost associated with it while the adaptation of the trained model on the target task is quicker.

2.2 Pretraining

Pretraining the model on the source task is costly, has high environmental costs and a considerable amount of carbon footprint associated with it [59]. Due to the high compute power required to pretrain the source models, pretraining is often done only once. Often researchers make available their pretrained models so that other people with less resources can use them for subsequent fine-tuning. The choice of the source task needs to be a well thought-out decision. The pretraining task should be able to capture all the properties of the language for its usefulness across many target NLP tasks [53].

The pretraining task can either be a large dataset of unlabelled text or human-crafted labelled dataset. Based on the type of pretraining task used, the source models can be pretrained in a supervised or unsupervised way. In this report, we focus on unsupervised pretraining and its adaption techniques.

Unsupervised pretraining leverages the representations or knowledge learnt from unlabelled data to the target task. It is a more accessible and adaptable approach in comparison to supervised pretraining. Unsupervised pretraining not only adds robustness to a deep architecture but also captures more intricate dependencies between parameters [9].

Most of the basic approaches used in NLP fall under the category of unsupervised pretraining. The knowledge acquired from the unsupervised pretraining process is usually transferred to the target tasks either in the form of features [35, 41] or fine-tuned weights [17, 8]. In the following sections, we describe the evolution of unsupervised pretraining from the shallow word embeddings to deep conceptualized word representations and eventually towards the pretrained language models.

2.2.1 Pretrained Word Embeddings

Word embeddings are an integral part in most of the NLP systems. Word embedding is a feature engineering and dimension reduction technique where the words or phrases from the vocabulary are mapped to vectors of real numbers [71].

One of the early approaches of word representations in NLP was one-hot vectors. A One-hot vector is a $1 \times N$ vector that is used to distinguish words in the vocabulary N. The vector has zeros in all the cells except the cell with the word that has the value 1 [70]. However, traditional word representations such as one-hot vectors are sparse and suffer from the curse of

dimensionality. Also, they are not good at representing the relations between different words. For example, the words “carrots” and “parsnips” belong to a root vegetable category, but their word representation using one-hot vector does not capture this relationship.

Due to the above shortcomings of the one-hot vectors, distributed word representations came into existence. These distributed word vectors are dense, have less dimensions and are more expressive than the traditional word representations [53]. Mikolov et al. [36] used two neural network architectures to produce distributed representations of words by pretraining a large corpus of unlabelled text. These continuous representations of words called Word2vec capture the exact word relationships both syntactically and semantically. The words encoded using these approaches are such that similar words are placed closer to each other in the vector space and have different levels of similarity. Apart from Word2vec, Glove [40] and fastText [3] are the most popular pretrained neural word embeddings in the NLP community. These embeddings are most commonly used as features in the majority of NLP tasks. The pretrained word embeddings significantly improve the accuracy of the baseline NLP systems over the embeddings learnt from scratch [64].

2.2.2 Language Model Pretraining

Pretrained neural language models can be regarded as these black boxes that learn the language representations and then can be fine-tuned on different NLP tasks in that language. These models have significantly improved the accuracy of the systems for different NLP tasks. Pretrained language models can capture the semantics and the syntax of the language and can learn both the word and the sentence representations from large corpus of unannotated data [54]. Moreover, due to the increase in computational power, it has become easier to pretrain deep neural language models.

In this section, we cover the background of the underpinning NLP advancements that paved the way for the pretrained language models that we all know today: Encoder-Decoder Architecture, Attention and Transformers.

Language modelling has been the focal point for many natural processing tasks. It has lead to many subsequent advancements like sequence-to-sequence models and different pretraining approaches for transfer learning. Language modelling predicts the next word in a sequence for a given sequence of previous words. In 2003, Bengio et al. [2] introduced the first shallow neural language model using a single hidden layer of a feed forward neural network. Later, Recurrent

Neural Networks (RNNs) [21, 34, 67] and Long Short Term Memory (LSTM) networks [14, 76, 33] based language models became more common.

Concurrently, the advancements in the field of machine translation has also led to new architectures of pretrained language models. Sutskever et al. proposed a new neural framework called *sequence-to-sequence learning (seq2seq)* [61] which is extensively used for machine translation and text generation tasks. The sequence-to-sequence model takes a sequence of words as input and outputs another sequence of words as shown in the Figure 2.3.

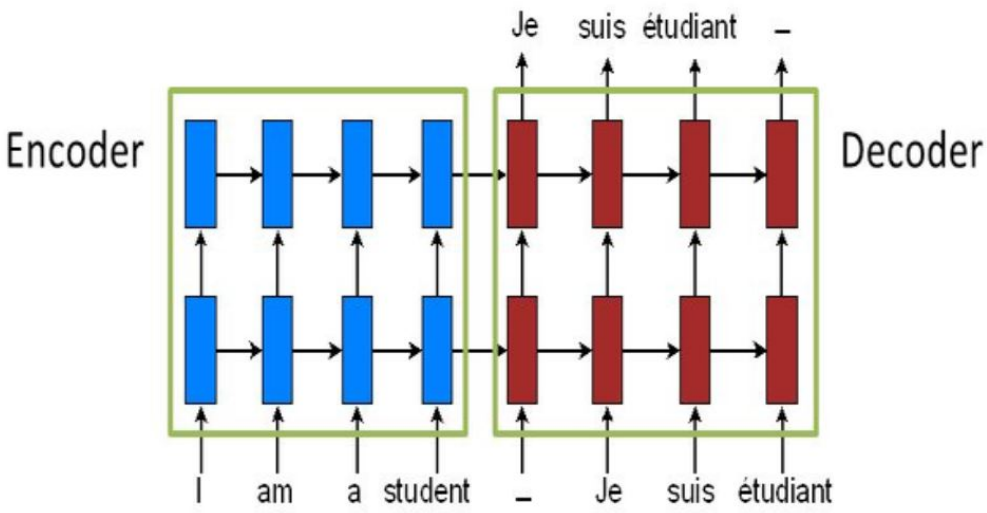

FIGURE 2.3: Diagram showing Sequence-to-Sequence model [31].

In the context of machine translation, both the sequences are processed word by word and can be of different lengths. Underneath the covers, seq2seq models consist of an *encoder* and a *decoder*. The encoder takes each word in the input sequence and compresses the information into a context vector, which is typically a vector of numbers. Once all the words in the input sequence are processed by the encoder the final context is passed the decoder. The decoder then outputs the prediction word by word taking into account the encoder context, and uses the previously predicted word as input at each step (Figure 2.4). Typically, both the encoders and decoders are RNNs. In recent architectures, RNNs are replaced with deep LSTMs [75], Convolutional encoders [12], and Transformers [65].

One of the drawbacks of the encode-decoder architecture is its inability to handle long sentences. Usually, the fixed-length content vector forgets the starting part of the sentence by the time it reaches its end. Bahdanau et al. [1] proposed *Attention*, a ground-breaking innovation

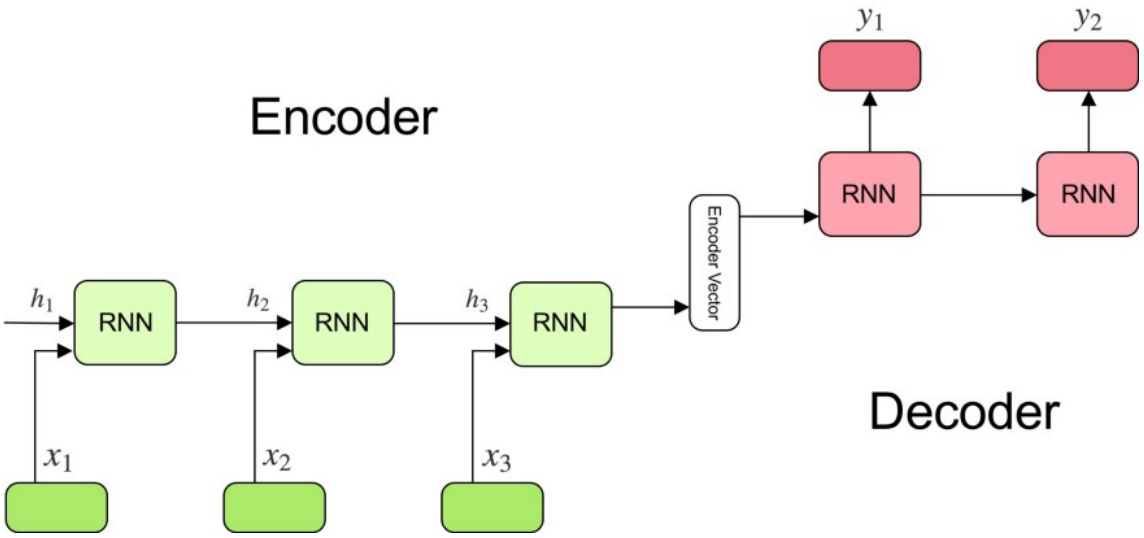

FIGURE 2.4: Diagram illustrating the Encode-Decoder Framework [56].

that changed the landscape of machine translation. Attention is paying more interest to relevant words in the input sequence. For example: Consider the sentence "The girl is eating a red apple". When you see the word "eating" you expect a category of food directly after it instead of colour. Thus each word "attends" to other word differently in the same sentence.

In Bahdanau et al.'s architecture [1], the context vector has information from all the hidden states of the encoder, decoder and the alignment information between input and output sequence. For each word that the decoder translates, it soft-searches for the positions of the most relevant parts in the input sentence as depicted in the Figure 2.5. Then, it predicts the output word based on the context vector associated with these positions and all the target words previously predicted. Attention is not only limited to looking for relevant information in between the input and output sentences. *Self-attention* looks at the neighbouring words in a sentence to get representations of the same sentence.

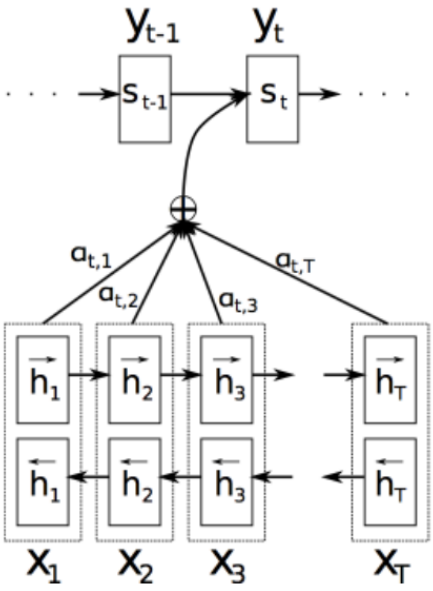

FIGURE 2.5: The graphical illustration of Attention mechanism [1].

Encoder-decoder models that use either RNNs and LSTMs have memory constraints due to the sequential nature of their computations thereby hindering parallelization. Attention mechanism used in conjunction with RNNs or LSTMs did not help the cause much. Thus, Vaswani et al. [65] came up with transformers for better parallelization compared to already existing machine translation and language modelling models.

The *Transformer* architecture [65], the latest state-of-the-art paradigm for neural machine translation has multiple layers of self-attention as its centerpiece, completely discarding recurrent and convolutional networks. Transformers are based on an encoder-decoder architecture with both encoder and decoder having 6 similar layers stacked over each other and adding position information through the positional encoding, displayed in the left and right side of the Figure 2.6. Each encoder layer has two components: multi-head attention and position-wise fully connected feed-forward network. Each component in turn is passed through a layer of residual connection followed by normalisation. The decoder is similar to the encoder except it has an additional component in between that processes' multi-head attention with the output from the encoder side.

Depending on the choice of the encoder and the training objective used there are many language models like BERT [8], GPT [45], GPT-2 [46], ELECTRA [5], ALBERT [24], RoBERTa [28], DistilBERT [55] and so on. In this section, we shed some light on the most prominent pretrained language models.

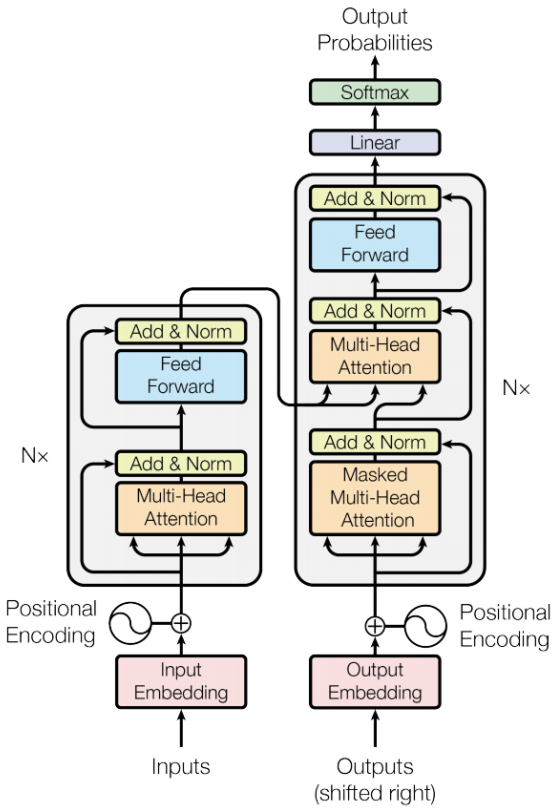

FIGURE 2.6: Diagram showing the Transformer Architecture [65].

Early Pretraining Approches

Dai et al. [6] were the first to come up with an idea to pretrain a sequence auto-encoder or language model on unlabelled data and then use the parameters from these models to train other supervised models. Their seq2seq model was similar to Sutskever et al. [61], except that their model predicted the input sequence as output, as shown in the Figure 2.7. They demonstrated that LSTMs pretrained on recurrent language models or seq2seq models performed better in comparison to LSTMs initialized with random weights.

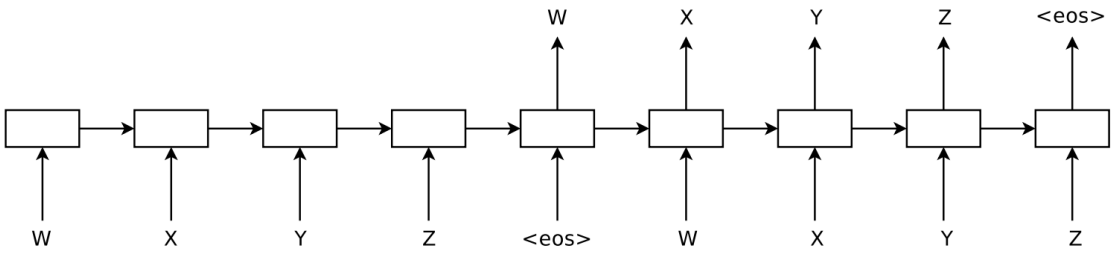

FIGURE 2.7: Figure showing Sequence Autoencoder [6].

Deep Contextual Word Representations

Consider the two statements: “The grey car is mine” and “This is a gold mine”. The word “mine” has different meanings depending on its context in the sentence. It is vital to capture the complex characteristics of the word use and how these characteristics change with the context when encoding the word vectors [41]. However, the distributed word representations using neural networks discussed so far are shallow and applied in a context free manner [53]. To resolve this problem, Peters et al. [41] introduced a new type of deep contextualized word representations called *ELMo (Embeddings from Language Models)*. They use the entire input sentence as a function to derive the representation of each token. Bidirectional LSTMs are trained with a language model objective on a large text corpus to extract contextual representations. ELMo representations are deep since they are derived as a function from all the internal layers of a bidirectional language model. These representations can be easily added to already existing architectures and improve the accuracy of the state-of-the-art benchmarks in every language understanding task.

Peters et al. [42] also conducted multiple experiments on three different bidirectional language model architectures to conclude that the LMs equip themselves with a high quality hierarchy of contextual data at both word and token level. Also, the models learn representations that differ with the depth of the network ranging from morphological based at embedding layer to syntactical information at lower layers and semantics and language modelling task related at top layers.

OpenAI Generative Pre-Training (GPT)

Radford et al. [45] proposed *universal language representations* which can be adapted easily with task-specific fine-tuning while requiring few changes to the model architectures. This method uses multi-layer *transformer [65] decoders* due to their ability to capture long term dependencies and add robustness to the model. These transformer decoders use multi-head self attention to train with a language modelling objective during the pretraining phase. Later, the pretrained parameters can be fine-tuned for diverse NLP tasks.

BERT

One of the limitations of the previous approaches is that they pretrain with uni-directional language modelling as their objective to derive the representations. In the case of ELMo,

independently trained left-to-right and right-to-left LSTMs are concatenated shallowly to produce the representations that are used for downstream tasks. Similarly, GPT uses left-to-right transformers in their architecture as depicted visually in Figure 2.8.

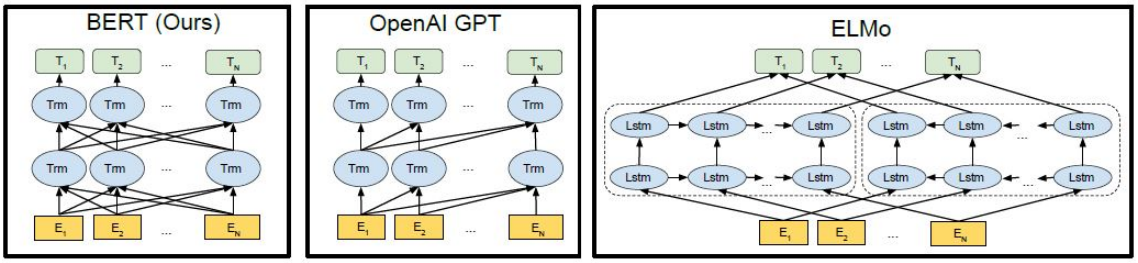

FIGURE 2.8: Comparing different pretraining model architectures [8].

To overcome this drawback, *BERT (Bidirectional Encoder Representations from Transformers)* [8] produces representations such that they are jointly conditioned on both left and right context in all layers. To achieve this, they use Masked Language Model (MLM) and Next Sentence Prediction tasks as the pretraining objective. In MLM, few words are randomly masked from the input tokens and later only those masked words are predicted. This masking strategy allows the model to look in both directions (left and right) and to be aware of the full context of the sentence when predicting the masked word. The second pretraining task of next sentence prediction captures the intrinsic relationships between two sentences useful for downstream tasks like Natural Language Inference (NLI) and Question Answering (QA). The next prediction task is the simple binary task of predicting whether two given sentences are next to each other or not.

BERT has been pretrained on large corpus of data from BooksCorpus [78] and Wikipedia with the two pretraining tasks as their objective described above. It took 4 days for the model to complete the pretraining.

Owing to the multi-layer bidirectional transformer encoder [65] architecture, BERT has outperformed all the previous state-of-the-art models in eleven token-level and sentence-level NLP tasks like language inference and question answering. The pretrained representations of BERT can be easily fine-tuned with only one output layer to achieve outstanding results on various NLP tasks without any modifications to the existing task-specific architectures. There are two variants of BERT depending on the model size:

- **BERT$_{\text{BASE}}$**: This model has 12 transformer blocks with 768 as hidden size and 12 self-attention heads that account to 110 million parameters in total.
- **BERT$_{\text{LARGE}}$**: This model has 24 transformer blocks with 1024 as hidden size and 16 self-attention heads that account to 340 million parameters in total.

DistilBERT

Following the current trend, pretrained language models in deep learning are getting deeper and more complex each day. Language models are being scaled up to gargantuan proportions to achieve outstanding performance on many target tasks. Deploying these bulky models on edge devices and to a large number of users has its own set of challenges. They have millions (occasionally billions) of parameters for example, GPT-2 [46] has 1.5 billion parameters. This leads to increased environmental costs [59] and computational constraints in terms of memory, budget and time, thereby impeding broader adoption of these models in real-time applications.

Sanh et al. [55] in their paper demonstrate that a smaller general purpose language model, *DistilBERT* can reach comparable performance of large models on different downstream tasks using information distillation. These compact models are lighter and faster in terms of inference time and have low latency when deployed on edge devices without impacting the performance. Sanh et al. apply the compression technique called *Knowledge Distillation* first introduced by Hinton et al. [15] as one of the ways to achieve this. Knowledge Distillation is the compression technique where a bigger cumbersome model – the teacher, is used to distill its knowledge to a smaller compact model – the student. In transfer learning, the weights are transferred from a pretrained model to a target model, which can be as complex as the pretrained model. However, in knowledge distillation, the goal is to transfer the ability to generalise from a bigger model to a compact model.

One way to transfer the ability to generalise from a teacher model to a student model is by utilising the class probabilities generated by the teacher model as "soft targets" for the training of a smaller student model as illustrated in Figure 2.9. The teacher model has the softmax with temperature T at the output layer, while the student model has two outputs, one with usual softmax function at the output layer and the other with the softmax with temperature T same as the teacher model. The smaller student model is trained to model the "soft targets", which are the outputs of the teacher model.

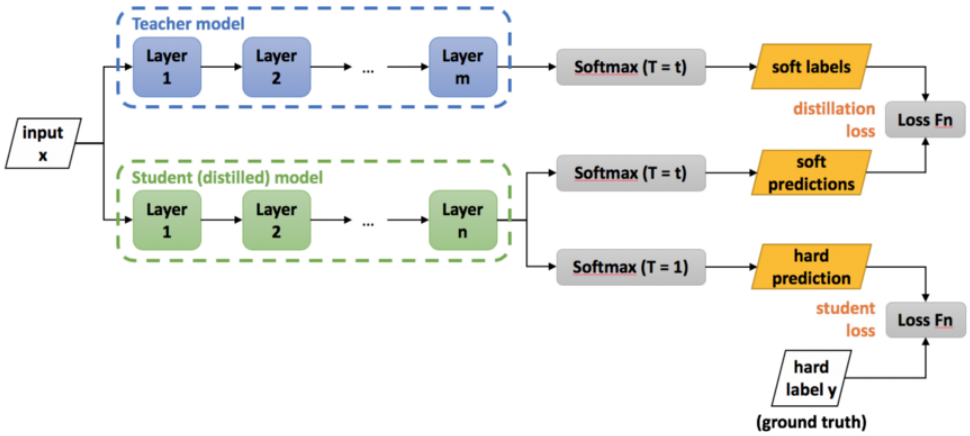

FIGURE 2.9: Diagram depicting Teacher Student architecture [38].

Sanh et al. [55] use the same *softmax-temperature* as Hinton et al. [15] given by equation 2.1.

$$p_i = \frac{\exp(\frac{z_i}{T})}{\sum_i \exp(\frac{z_j}{T})} \tag{2.1}$$

where a typical neural network that outputs the class probabilities using the "softmax" output layer transforms z_i, the logit, for each class into a probability p_i by comparing z_i with other logits. The temperature value T is ideally set to 1 to restore the standard softmax. The student is trained on the distillation loss shown in the equation 2.2 where t_i (resp. s_i) is a probability estimated by the teacher (resp. the student).

$$L_{ce} = \sum_i t_i * \log s(i) \tag{2.2}$$

Also, they set the architecture of the student model (DistilBERT) same as the teacher model (BERT) except that the number of layers is halved. Their end training objective, "triple loss" which is a linear combination of *distillation loss* L_{ce} with supervised training loss, *masked language modeling* loss L_{mlm}, in this case, and finally the *cosine embedding* loss L_{cos} between the hidden states of student and teacher models.

DistilBERT has half the total number of parameters and reports just about 5% drop in the efficiency of natural language understanding task benchmarks in comparison to BERT.

2.3 Adaptation of Pretrained Models

In the previous section, we have discussed the pretraining phase in sequential transfer learning. Research in the adaptation phase is in its nascent form. During the adaptation phase, the below directions form the basis for the decisions to be taken [54]:

- How much change does the pretrained model architecture require for adaptation?
- Which pretrained weights should we update?
- How and when should the weights be updated?
- Do we need additional supervision signals?

2.3.1 Whether to Tune or Not to Tune?

There are mainly two techniques used to adapt the pretrained representations on the target task: ***feature extraction*** and ***fine-tuning***.

In feature extraction, the pretrained model's weights are 'frozen'. In this paradigm, the pretrained representations are used as input features to the downstream model. Alternatively, a linear classifier or a linear combination of layers is trained on the top of the pretrained representations [41]. Feature extraction enables the use of already existing task-specific model architectures and is cost efficient as features are computed only once [43].

In contrast, the pretrained representations are updated in fine-tuning and the whole pretrained architecture is trained directly on the target task data. The pretrained weights are used as initialization for the parameters of the downstream model on the target task. One of the major benefits of using fine-tuning is that it allows us to adopt a general purpose representation to a diverse number of tasks [54].

Peters et al. [43], compare the two adaptation techniques, feature extraction and fine-tuning on different tasks with two state-of-the-art pretraining representations. The results show that the similarity between source and target tasks plays an important role in their relative performance. When the source and target tasks are similar, fine-tuning gives better results and in the case of dissimilar tasks, feature extraction outperforms the other. The decision to fine-tune or not also depends on other trade-offs like space, time and performance.

2.3.2 Fine-tuning Settings

The fine-tuning settings are quintessential for better adaptation of the language model representations. Fine-tuning involves updating the pretrained weights. However, the schedule in which these weights are updated is critical to avoid catastrophic forgetting of the knowledge learnt from the source task [54].

After pretraining the Language Model (LM), fine-tuning enables the LM to adapt to the different distribution of the target task data. Since all the layers of pretrained model have already learnt the general language features, fine-tuning the pretrained LM only involves adapting to the peculiar behavior of the target data. One of the guiding principles of fine-tuning is to update the weights from top to bottom progressively in time and intensity [54].

Training all layers at the same time on data of different target task might result in instability and poor results. The solution to this is to train all the layers individually giving it time to adapt to the new task and data [54]. There are many unfreezing schedules worth mentioning like ***freezing all the layers except the top layer*** [29], ***chain thaw*** [10], ***gradual unfreezing*** [17]. In the chain thaw approach, there is sequential unfreezing and fine-tuning of a single layer at a time. In gradual unfreezing, Howard et al. propose to first unfreeze the last layer and fine-tune the unfrozen layer and then gradually add a layer at a time to the set of unfreezed layers and fine-tune all layers until convergence.

Recently, Chronopoulou et al. in their paper [4] proposed a simple and efficient approach to address the problem of catastrophic forgetting by training the entire model end-to-end in a single step. They used ***sequential unfreezing*** using hyper-parameters that determine the length of fine-tuning process. They first fine-tune only the extra, randomly initialised LSTM and the output layer for n - 1 epochs. Exactly at the nth epoch, they unfreeze the pretrained hidden layers. Then, they let the model fine-tune, until epoch k - 1. Finally, at epoch k, they also unfreeze the embedding layer and let the network train until convergence. Grid search is used to obtain the values of n and k.

Prior work shows that different layers capture diverse information. The upper layers capture task-specific features and the lower layers have the general language representations. During the early stage of learning, the model needs to adapt to the target task and thus we can afford to use bigger steps of learning rates. Then late in the training process, as the model converges, a lower learning rate should be preferred to avoid overwriting useful information. In their paper, Howards et al. [17] propose ***discriminative fine-tuning*** which tunes each layer with different

learning rates such that the learning rate decreases from top to bottom layer.

Also, when adapting the parameters to the task-specific features, the model needs to quickly move to a suitable region in the parameter space in the early stage of training and then slowly converge over time. To achieve this behaviour, Smith et al. [57] proposed ***Cyclical Learning Rate (CLR)***, where they estimate the boundary learning rates by running the training in mini batches of few epochs with increasing learning rate for the CLR policies. Then, they devise a policy where the learning rate cyclically varies between these bounds to achieve optimal results [57]. They adopted a triangular window (linearly increasing then linearly decreasing) as the policy in their approach.

Howard et al. adapted a similar strategy called ***Slanted Triangular Learning Rates (STLR)***, in which they first linearly increase the learning rate and then linearly decay it as shown in Figure 2.10 [17]. This approach has a shorter increase and a longer decay period when compared to the triangular learning rates approach [57].

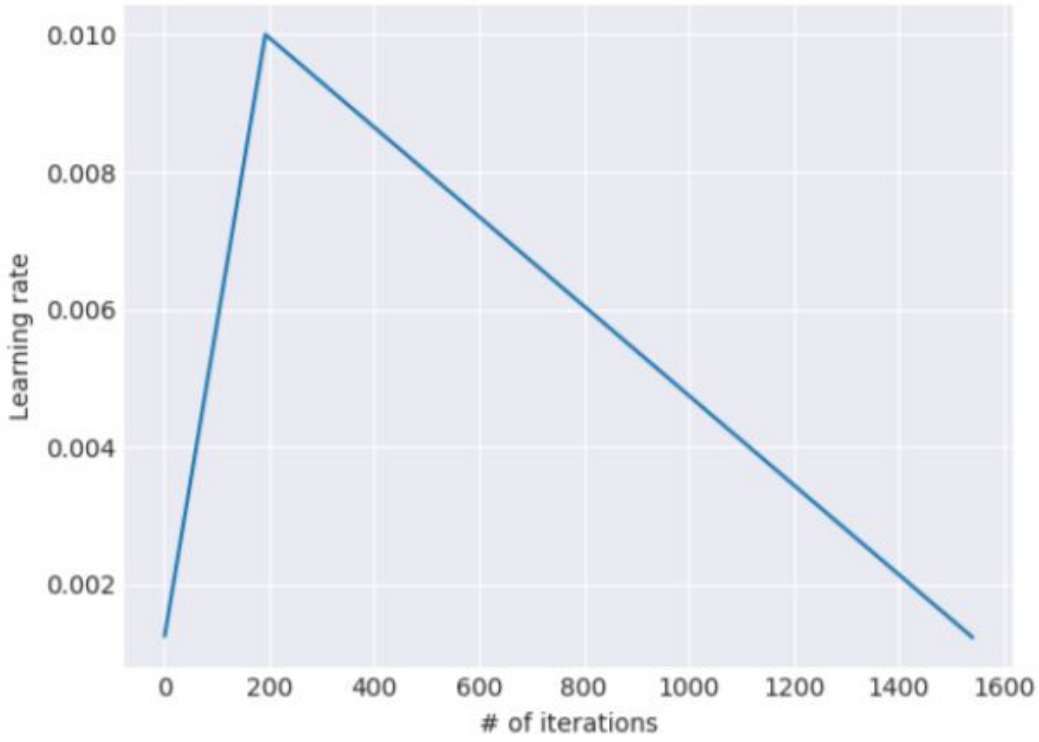

FIGURE 2.10: ULMFiT as a function of training iterations and with slanted triangular rate schedule [17].

2.3.3 Architectural Modifications and Additional Signals

Depending on the similarity between pretrained and target tasks, the pretrained model architecture can change or remain the same. If the target task is structurally similar to the pretrained task, then the pretrained model internals remain unchanged with minor target-task-specific modifications on top of the architecture. Usually, NLP practitioners remove the pretraining task head if it is not required for the target task. For example, removing a softmax classifier from the pretrained language model. Then, target task specific layers like linear layers are added on top or bottom of the pretrained models, keeping the model internals unchanged [54].

However, when the pretrained representations are adapted to a structurally different target task, the pretrained model's internal architecture needs to be changed. For example, when adapting a pretrained task of single sequence to a multiple sequence target task. Different capabilities or modules are added to the pretrained model's architecture to increase its usefulness for the target task [54]. Adding residual connections between layers, using multi-layer attention [49] and adapters [50] [58] are a few of the approaches used recently.

Using the ***Residual Adapters*** [50] module is a recent strategy in which task-specific parameters are inserted in between the layers of the pretrained model instead of adding them on top of the pretrained model architecture. These adapters are layers of different designs and contain a small fraction of parameters in comparison to the entire model. They are connected by means of residual connections with the rest of the network [54]. Adapters facilitate the reuse of the same pretrained parameters for many target tasks. Only the adapters are fine-tuned during the adaptation phase. In case the adapters are not useful for the downstream task, the residual connection enables the model to ignore them [54].

Inspired by residual adapters, Stickland et al. [58] add a low-dimensional multi-head attention layer in parallel to normal BERT layers. These adaptation modules allow a high degree of parameter sharing between the target tasks by adding the task-specific parameters to every layer of the pretrained BERT model. They get the same results as fine-tuned BERT on the General Language Understanding Evaluation (GLUE) benchmark with around 7 times less parameters.

To improve the performance of transfer learning on low-resource target data, we could use a combination of different strategies to provide additional signals. Some of these strategies are listed below [54]:

- ***Sequential Adaption:*** intermediate fine-tuning on related, high-resource datasets before

fine-tuning on the target dataset. Sequential adaptation is also referred to as multi-task fine-tuning in some literature.

- ***Ensembling:*** combining the predictions of different models fine-tuned with various hyper-parameters.
- ***Multi-tasking:*** fine-tuning the model jointly on related tasks.

2.4 Transfer Learning for Biomedical Question Answering

Question Answering (QA) is an NLP task of retrieving answers to a question given set of contexts. A typical span-extractive question answering task consists of a given passage P and question Q, the system needs to locate an answer span A (a_{start}, a_{end}) in P.

Transfer learning has been extensively used for transferring the knowledge from one domain to another. The lack of large scale domain-specific datasets and the cost associated with manually annotating them is the reason behind this trend. In this literature review, we particularly focus on biomedical question answering task of ***BioASQ*** [62].

BioASQ [62] is the first international challenge for generic biomedical question answering. It organises challenge tasks on biomedical question answering and semantic indexing. Question Answering (QA) is a task of retrieving answers to a question given set of contexts. The BioASQ dataset consists of a small collection of list, factoid and yes/no questions.

Wiese et al. [69] were the first to explore transfer learning techniques on biomedical question answering. They employ domain adaptation techniques to transfer knowledge from an already existing neural QA system (Fast QA [68]) trained on large SQuAD dataset to a smaller BioASQ dataset. Their approach falls under the category of supervised domain adaptation since the source task is labelled, target and source domains are different but tasks are similar. During the fine-tuning phase, they initialise the model parameters with the parameters of the pretrained models. The combination of fine-tuning and use of biomedical Word2vec embeddings produced state-of-the-art results in the 5th BioASQ challenge. Also, to avoid catastrophic forgetting, they deployed optimization techniques like forgetting cost term and L2 weight regularization to further boost the results.

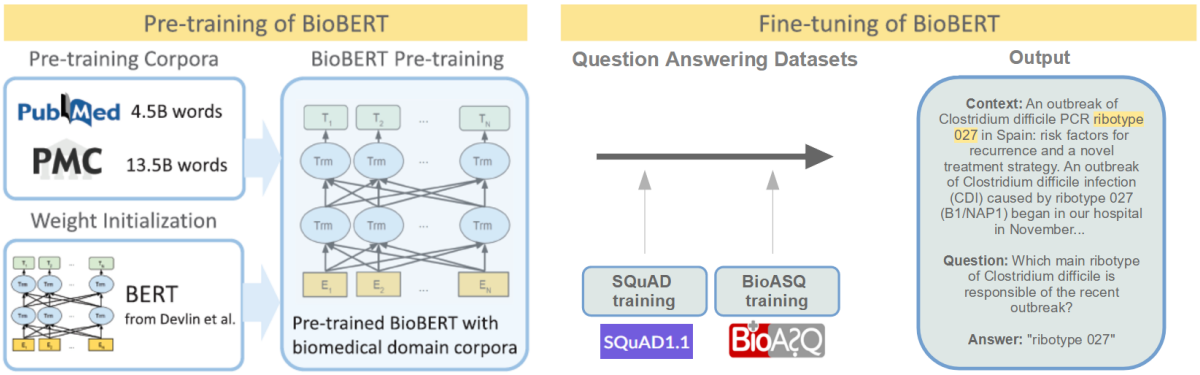

FIGURE 2.11: The pretraining and fine-tuning process of BioBERT. Diagram adapted from [25].

With the onset of pretrained language models like BERT for general domain, Lee et al. [25] found the potential to adapt BERT for the biomedical domain. They introduce ***BioBERT*** which is the first pretrained language representation model for the biomedical domain. Figure 2.11 illustrates an overview of the pretraining and the fine-tuning process of BioBERT. As shown in Figure 2.11, during the pretraining step, BioBERT is first initialized with the weights of BERT, a state-of-the art model which is pretrained on general domain corpora. Then, BioBERT is pretrained on biomedical domain corpora. It took nearly 23 days for them to pretrain BioBERT v1.1 using 8 NVIDIA V100 GPUs. BioBERT has yielded state-of-the-art results on various biomedical text mining tasks like named entity recognition, relation extraction and question answering tasks.

In the **7th BioASQ challenge**, Yoon et al.'s submission [77] for the task 7b was at the top of the leaderboard. They employed the sequential adaptation strategy where pretrained BioBERT was first fine-tuned on SQuAD [48] and later on BioASQ dataset.

On the same lines, ***BioELMo*** [18], which is a biomedical version of ELMo, when used as features extractor, outperforms BioBERT on the probing tasks designed by the authors. However, as expected, the fine-tuned BioBERT performs better than BioELMo on named entity recognition and NLI tasks.

In their paper [16], Hosein et al. detail their experiments and evaluation results of their submission to the BioASQ Task 7b. They measure the domain portability of BERT based question answering models that are pretrained and fine-tuned on general domain with models in biomedical domain. The results conclude that large scale general domain models have good portability to a new domain. They also observe that pretraining is more vital than fine-tuning

to improve the domain portability of BERT based question answering models. Interestingly, one of their system uses a sequential adaptation strategy in which BERT is first fine-tuned on SQuAD and later on Natural Questions (NQ) [23] and Conversational Question Answering (CoQA) challenge [51] datasets.

Resta et al. [52] explored the use of an ensemble of classifiers whose input was obtained through different transformer based language models. They used contextual embeddings from different pretrained language models like ELMo and BERT as features to capture long term dependencies. Their submission ranked top of the leaderboard in two batches for yes/no questions in the 7th BioASQ challenge.

The **8th BioASQ Challenge** in 2020 saw a total of 34 teams from around the world participate in the three shared tasks of the challenge [37]. In their participating system for the challenge, Jeong et al. [19] extend their previous work on BioBERT models [25, 77] to demonstrate that multiple levels of sequential adaptation is effective in improving the performance of biomedical question answering. They first fine-tune BioBERT on the NLI dataset [72] and then on the SQuAD [48] dataset and finally on the BioASQ Task 8b dataset. Their results establish that NLI task, which learns the relationships amongst sentence pairs, enhances the accuracy of biomedical question answering systems. Furthermore, they also report an analysis of the number of the unanswerable questions from BioASQ Task 7b Phase B test data in the extractive question answering (QA) setting.

The transformer-based system of Kazaryan et al. [20] has ALBERT [24] as its underlying neural network that is first fine-tuned on SQuAD v2.0 [47] and then on the training dataset of BioASQ 8b [24].

2.5 Conclusion

In recent years, there has been a major paradigm shift. We are beginning to realise the true potential of transfer learning in NLP. This realisation stems from the fact that generalisation from an existing task is any day better than starting from scratch. We discuss transfer learning with the main focus on sequential transfer learning and also describe in detail the two phases of sequential learning: pretraining and fine-tuning. Furthermore, we detail the current developments of applying transfer learning in the biomedical domain.

There is a great deal of scope for applying fine-tuning techniques like residual adapters

and optimization techniques on the pretrained models to generalise better in the biomedical domain. Currently, there is no research on the effects of using different fine-tuning techniques like gradual unfreezing and discriminative fine-tuning on BioBERT. In this project, we will investigate the effect of using fine-tuning technique of Gradual Unfreezing on DistilBERT.

3

Approach

As stated in Chapter 1, this thesis aims to answer two main research questions. First, we explore whether a compact model like DistilBERT can achieve comparable performance to a larger model, BioBERT. We report our results on the data from BioASQ with question answering task as our NLP use case. Second, we investigate the effect gradual unfreezing has on the performance of a smaller transformer-based model, DistilBERT, using a low-resource dataset such as BioASQ.

In this chapter, we explain our approach to answering our two research questions. Section 3.1 describes the BioASQ shared task, a question answering dataset used in our experiments. Section 3.2 describes the evaluation metrics used to evaluate the quality of our question answering systems. In Section 3.3 we lay out our main approach used for our first line of inquiry and our reasoning behind the algorithms chosen. In section 3.4 we propose an approach to tackle our second research question of gradual unfreezing.

3.1 BioASQ

BioASQ [62] is the first international challenge for generic biomedical question answering started in 2013. They organize annual challenge tasks on biomedical question answering and semantic indexing. The seventh edition of the BioASQ competition consists of two tasks: Task 7a, a biomedical semantic indexing task and Task 7b, a biomedical question answering task. Task 7a aims to classify the new articles from the PubMed [44] digital library into concepts of the Medical Subject Headings (MeSH) hierarchy. Task 7b involves a comprehensive question answering challenge designed for systems to tackle four categories of questions from biomedical domain namely, *yes/no, list, factoid and summary* questions. The participants of Task 7b are provided with questions along with related articles and snippets for each question. The participants' systems in turn output either ideal answers (a short paragraph for summary questions) or exact answers (e.g., list of named entities for factoid questions). Task 7b is released in five test batches over a period of two months with only 24 hours gap given to submit the answers after releasing the questions.

In this project we focus on factoid questions from the BioASQ 7b dataset. The training dataset of BioASQ 7b consists of 2747 questions in total out of which 779 are factoid questions. Each question from the Task 7b of the BioASQ training dataset has many fields, the most prominent ones are:

- *type*: type of question i.e., factoid, list, yes/no or summary
- *snippets*: multiple snippets extracted from PubMed articles relevant to the question along with the URLs of the PubMed documents used
- *ideal answer*: a short summary for summary questions
- *exact answer*: an entity or list of entities for factoid question; yes/no for yes/no type questions; list of answers for list questions

Question: Which is the third subunit of the TSC1-TSC2 complex upstream of mTORC1?
Ideal Answer: TBC1D7 was identified as a stably associated and ubiquitous third core subunit of the TSC1-TSC2 complex...
Exact Answer: TBC1D7
Type: Factoid
Snippet 1: The tuberous sclerosis complex (TSC) tumor suppressors form the TSC1-TSC2 complex...
Snippet 2: Here, we identify and biochemically characterize **TBC1D7** as a stably associated and ubiquitous third core subunit of the TSC1-TSC2 complex...

FIGURE 3.1: An example of factoid question from the BioASQ 7b training dataset. The gold standard answer is in bold. Note that the gold standard answer is present in snippet 2 and not in snippet 1.

Our system returns exact answers to the factoid question from the BioASQ 7b dataset. An exact answer to the factoid question could be either a single entity (it could be the name of a gene, disease, medicine, virus or even a number) or a list of 3-4 entities. An example of one of the factoid questions from the BioASQ 7b training dataset is shown in Figure 3.1. As seen in the example, answer to the question is extracted from the relevant snippet (passage) provided. Therefore, we consider the BioASQ 7b task as an extractive question answering task.

3.2 Evaluation Metrics

In this section, we list the metrics used to evaluate our system. Our system returns a list of up to five entities for each factoid question in decreasing order of their probability. The performance of our system is evaluated on the same three metrics used by the BioASQ organisers: *Strict Accuracy (SAcc), Lenient Accuracy (LAcc) and Mean Reciprocal Rank (MRR).*

For each question, if the golden entity name (or its synonym) is the first element of the list returned by the participant's system, strict accuracy marks that particular question as correctly answered. On the other hand, lenient accuracy marks a question as being correctly answered if the golden entity name (or its synonym) is included in the list returned by the system, not necessarily as the first element.

The equations 3.1, 3.2 below define the formula to calculate the strict and lenient accuracy for factoid questions [62].

$$SAcc = \frac{c_1}{n} \tag{3.1}$$

$$LAcc = \frac{c_5}{n} \tag{3.2}$$

where c_1 is the number of factoid questions answered correctly when considering only the first element from the list of answers returned by the systems, c_5 is the number of questions counted as correct when all the five answers in the list submitted are considered and n is the total number of factoid questions to be evaluated.

MRR is the most common evaluation metrics used for factoid question answering, proposed in the Text REtrieval Conference (TREC) question answering track in 1999 [66]. MRR tends to punish systems that produce the correct answers that are ranked lower in the returned list of answers.

MRR scores each question as the reciprocal of the rank of the first correct answer. Suppose a system returns a list of five answers for a particular question, and the correct answer is ranked third in the list, then the reciprocal rank score for that question is $\frac{1}{3}$. Any question that has no correct answers in the returned list of answers is marked as zero. Then MRR for the system is the average of the score of each question.

In the formal definition 3.3 below, for each factoid query q_i, we scan through the returned list for the top position containing the golden entity name (or one of its synonyms). Let's assume that the *j*-th element is that topmost position, then *r(i) = j;* otherwise $r(i) \rightarrow +\infty, i.e., \frac{1}{r(i)} = 0$ [62].

$$MRR = \frac{1}{n} \cdot \sum_{i=1}^{n} \frac{1}{r(i)} \tag{3.3}$$

In the lines of BioASQ, we employ the strict and lenient accuracy for measuring completeness. We use the **mean reciprocal rank** as the main criteria when evaluating our systems for factoid questions since it is the main metric used by BioASQ organisers.

3.3 Fine-tuning DistilBERT

We apply sequential Transfer learning that has proven to be successful on extractive question answering tasks [77, 16, 8]. The small size of the BioASQ dataset makes it the ideal scenario for using transfer learning approaches. In line with the sequential methodology, a pretrained language model is first selected which is trained in an unsupervised manner on a large data corpus. The knowledge from this pretrained model is then adapted to biomedical domain. Taking cue from the success of using pretrained transformer-based models like BERT, BioBERT, ALBERT on different question answering datasets [8, 25, 24], we choose DistilBERT as our pretrained transformer-based model. Our choice of DistilBERT is due to the fact that it is architecturally similar to BERT and has less parameters than BERT. To the best of our knowledge, DistilBERT has not yet been used as pretrained model for the BioASQ dataset.

For the adaptation phase, we prefer to use the fine-tuning technique to adapt the pretrained representations on the target task. Sequential adaptation (multi-task fine-tuning), intermediate fine-tuning on high-resource datasets, has improved the performance of low-resource target tasks [77, 20, 19]. This second step of fine-tuning has yielded benefits on answer sentence selection and text classification tasks [11, 60]. However, there are no large scale factoid question

answering datasets readily available in biomedical domain. Thus, SQuAD v1.1 [48] and SQuAD v2.0 [47], general domain QA datasets are widely used as intermediary datasets for sequential adaptation [22, 77] in the biomedical domain. In our system, we use SQuAD v1.1 to provide the additional signal required to improve the performance of biomedical QA systems. To summarize, we use DistilBERT as our pretrained language model which is first fine-tuned on SQuAD v1.1 and then on BioASQ 7b dataset shown in Figure 3.2. Further details of the experiments and the fine-tuning settings used are elaborated in chapter 4.

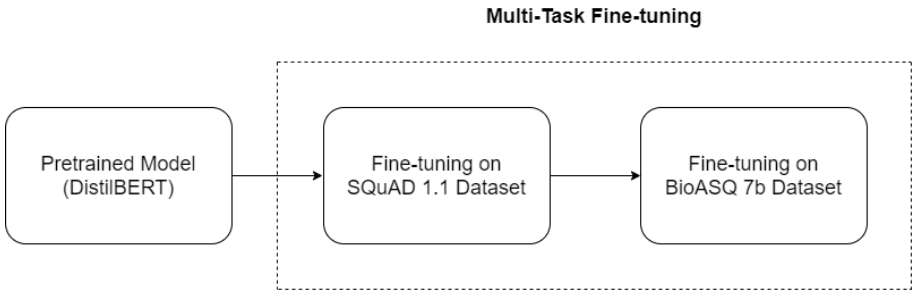

FIGURE 3.2: Diagram depicting the fine-tuning approach used by our system. DistilBERT is first fine-tuned on span-extractive QA task – SQuAD v1.1 and then on biomedical BioASQ 7b QA dataset.

3.4 Gradual Unfreezing

In typical end-to-end fine-tuning, all the layers of the pretrained model and the task-specific layers are trained simultaneously on the target data. The pretrained weights that are initialised at the start of the fine-tuning process are all updated. However, in the process of learning new knowledge, there is a risk that the model might forget the knowledge it has learned from the pretrained task. Hence the schedule for updating the weights plays an important role in avoiding this catastrophic forgetting. Howard et al. [17] introduced the gradual unfreezing technique on a regular LSTM without any attention mechanism. Instead of fine-tuning all the layers together, they gradually unfreeze one layer starting from the top layer, each time fine-tuning the set of unfrozen layers for one epoch until all the layers are fine-tuned as illustrated in the Figure 3.3. Gradual unfreezing has not yet been applied to the low-resource BioASQ dataset in combination with a compact transformer-based model such as DistilBERT.

On similar lines, in our approach, we start from the top task-specific layer and allow the model to learn the varied representations of the target task first for one epoch while all the remaining layers are frozen. Then we gradually unfreeze the lower top layer and continue

fine-tuning unfrozen layers for one epoch until all the layers except the bottom embedding layer are fine-tuned.

The initial embedding layer of the model is never fine-tuned in our approach. The decision to remove the embedding layer from the fine-tuning phase was taken on the basis of the preliminary investigations discussed in Chapter 5. The intuition behind our top to bottom fine-tuning approach is that top layers are task-related, domain-specific and require more fine-tuning on target task. Also, the topmost task-specific layer of the pretrained model is the only layer that is initialized with random weights during the fine-tuning process and therefore needs to capture the distributions of the target task first.

We implement the same sequential adaptation approach described in the Section 3.3. In short, we follow multi-task fine-tuning in which DistilBERT is first fine tuned on the SQuAD dataset and then on the data from BioASQ 7b. Gradual unfreezing is employed only when fine-tuning DistilBERT on the target dataset i.e., BioASQ 7b.

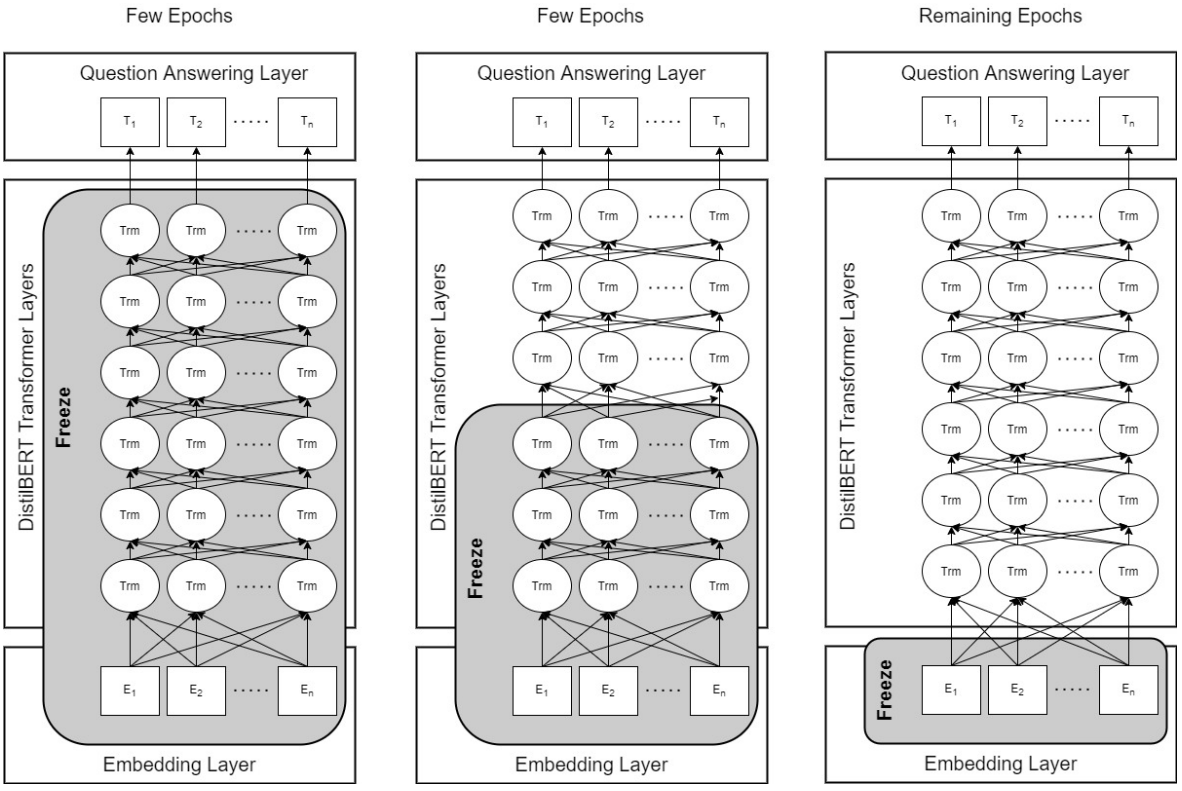

FIGURE 3.3: Diagram depicting the gradual unfreezing approach when unfreezing three transformer layers at a time.

The DistilBERT architecture consists of three blocks: an embedding layer, 6 transformer layers, and an additional final question answering layer. We also investigate the effect of gradual

unfreezing when the transformer layers are fine-tuned together in groups. Again, we start from the top question answering layer, which is first fine-tuned for few epochs while the other layers are frozen. Next, we unfreeze transformer layers consecutively in groups of either 3 or 6 and fine-tune them for a few epochs until all the layer except the embedding layer are fine-tuned. Figure 3.3 shows the unfreezing process when three transformer layers are grouped together. All experiments of gradual unfreezing are detailed in chapter 5.

4

Fine-tuning DistilBERT

In Chapter 3, we discussed the approach that we use to answer our research questions. In this chapter, we describe our experiments using the multi-task fine-tuning methodology discussed in Section 3.3 to answer our first research question as to whether a smaller pretrained model, DistilBERT, can improve the quality of a low-resource biomedical question answering task. We begin this chapter by describing the inner workings of our model. Then we detail the pre-processing and post-processing steps involved in transforming the BioASQ 7b dataset from an extractive QA task into a span prediction task.

This chapter's layout is as follows. Section 4.1 describes the details of our experiments. Section 4.2 describes the data processing steps involved on the dataset. Sections 4.3 and 4.4 describe the experimental setup and the hyper-parameters that we choose for our experiments. Section 4.5 describes and discusses the results.

4.1 Model

In this section, we detail the input representation of our model and the fine-tuning procedure adapted by BERT for the extractive question answering task.

The input representation of BERT can cater for single sentences and pairs of sentences in one sequence of tokens. The input representation for each token is the sum of its three corresponding embeddings: token, segment and position. WordPiece embeddings [75] is used for tokenization to avoid the problem of out-of-vocabulary and rare words. WordPiece is a segmentation technique that splits the words into sub-word units ("wordpieces") and the split wordpieces are denoted by ##. For example, the word "proteasome" is split into 4 wordpieces: "pro", "##te", "##as", "##ome". A classification embedding token, [CLS] is always added as the first token of every input sequence to cater for classification tasks. For sentence pair tasks such as QA and NLI, a special token [SEP] is added in between two sentences to separate them. To further differentiate the two sentences, a learned sentence embedding A is added to every token in sentence A and sentence embedding B to every token in sentence B. The position embedding represents the learned position of the token in the sequence. BERT can support sequence lengths up to 512 tokens.

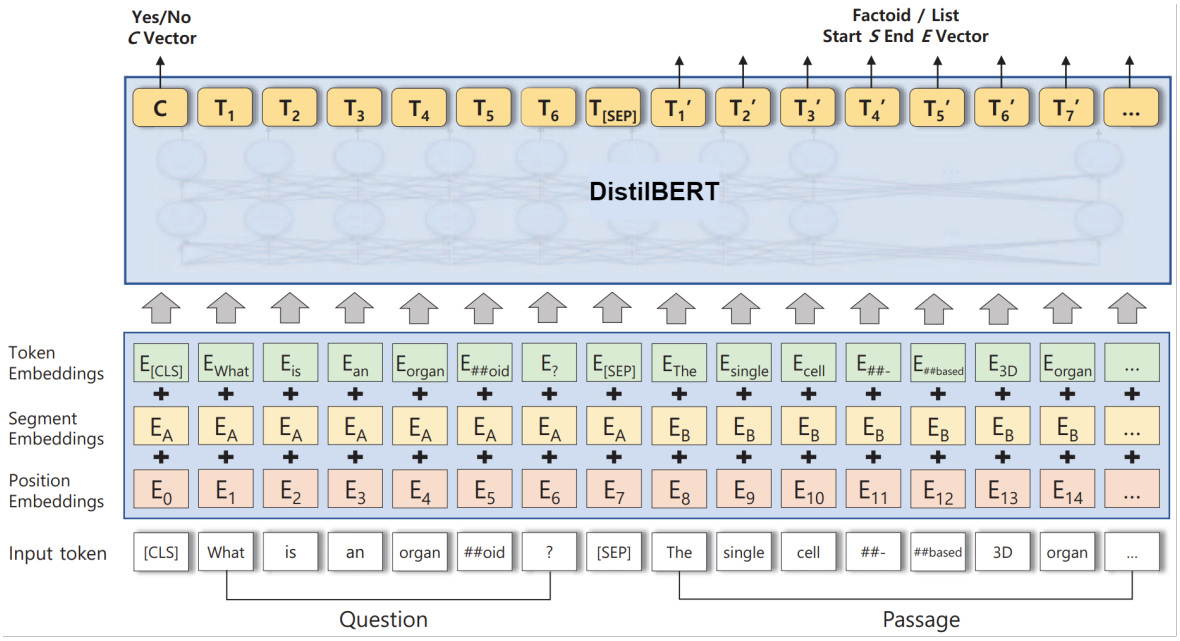

FIGURE 4.1: Example of a sequence of Question-Passage pair processed by BioBERT. Diagram adapted from [77].

Fine-tuning Procedure The fine-tuning process to adapt BERT for a span extraction task (or span-level prediction task) like SQuAD involves few task-specific modifications. The input question and context are fed into the model as a single sequence separated by a [SEP] token along with their corresponding segment embeddings to distinguish them.

During fine-tuning, the only new parameters that are learned are: a start vector $S \in \mathbb{R}^H$ and an end vector $E \in \mathbb{R}^H$. Let us assume $T_i \in \mathbb{R}^H$ and $T_j \in \mathbb{R}^H$ denote BERT's final hidden vectors for the i^{th} and j^{th} input tokens respectively. Then, the probability that the word i is the start of the answer span is calculated as a dot product between T_i and S followed by a softmax of all the words in the paragraph [8]. Similarly, the end of the answer is calculated. Equations 4.1, 4.2 for computing both the start and the end are shown below:

$$P_i^{start} = \frac{e^{(S \cdot T_i)}}{\sum_j e^{(S \cdot T_j)}} \tag{4.1}$$

$$P_i^{end} = \frac{e^{(E \cdot T_i)}}{\sum_j e^{(E \cdot T_j)}} \tag{4.2}$$

The score of a possible span from position i to position j is calculated as $S \cdot T_i + E \cdot T_j$. The candidate span with maximum score and where end position, j is greater than or equal to the start position i, ($j \geqslant i$) is the model's prediction. Therefore, the training objective is the sum of the log-likelihoods of the correct start and end positions.

4.2 Data Processing

This section discusses both the pre-processing and post-processing steps required to convert the BioASQ dataset into the SQuAD format and vice-versa. The SQuAD dataset is a large collection of question-answer pairs along with a passage (named *context* in the dataset) that answers the given question.

On the other hand, the BioASQ training dataset provided by the BioASQ organisers has a question, an exact answer and multiple relevant snippets. Refer to section 3.1 for an example of a factoid BioASQ question. First, as part of pre-processing, each snippet is paired with its question to transform the dataset into multiple question-snippet pairs [77].

In their work, Yoon et al. [77] introduce *Full Abstract Strategy* to pre-process the BioASQ dataset. In this strategy, in addition to the snippet provided, the complete abstract along with the title from PubMed articles related to that snippet (obtained using the links provided in the

BioASQ dataset) is appended to the original snippet to make a single passage. Finally, using the exact answer provided in the original dataset, the start position of the answer in the passage is identified and populated in the dataset to transform the BioASQ 7b dataset into a span prediction task. These pre-processing steps increase the total number of training samples from 779 to 5537. Table 4.1 shows the statistics of BioASQ training and test batches. Refer to Appendix A for an example of a factoid question before and after pre-processing.

In our experiments, we use the full abstract variant of pre-processed training and test batches of BioASQ 7b dataset that have been made publicly available by the authors [77] in their GitHub repository [74]. Our system outputs the prediction span for each question. However, since we have split snippets into multiple question-passage pairs in the pre-processing step, we now have several answer spans along with their probabilities as predictions for each question. We then select the top five answers in the decreasing order of their probabilities for a particular question as our submission, since the official evaluation of BioASQ requires each system to return a list of up to 5 answers.

Dataset	**Number of Factoid Questions (Provided by BioASQ Organisers)**	**Number of Factoid Questions After Pre-processing**
Training	779	5537
Test Batch 1	39	98
Test Batch 2	25	56
Test Batch 3	29	84
Test Batch 4	34	90
Test Batch 5	35	79

TABLE 4.1: Statistics of Factoid Questions BioASQ 7b Training and Five Test Batches

4.3 Experimental Setup

We are investigating different fine-tuning techniques on DistilBERT for extractive question answering setting. The BioASQ 7b training data is used for training our model and Test Batch 1 is to tune the hyper-parameters. The four batches of test data are evaluated on the gold-standard answers made available by the BioASQ organisers and are compared with other top-ranked systems on the BioASQ 7b leaderboard [63]. All the systems are evaluated for performance on

the three metrics: SAcc, LAcc and MRR (detailed in Section 3.2) used by the BioASQ organisers with an intent to compare results with the top ranked systems in the BioASQ challenge.

In all of our experiments (including chapter 5), we use pretrained models from the Huggingface Transformers Library [73] along with their model specific tokenizers. We choose AdamW optimizer [30] along with a linear scheduler with a warm-up in all our experiments. This linear scheduler linearly decreases from the initially assigned *learning rate* in the optimizer to zero until a warm up period after which it linearly increases from zero to the initially assigned learning rate in the optimizer.

We utilize one NVIDIA Volta GPU for all our fine-tuning experiments. This project was implemented using the resources from the National Computational Infrastructure Australia (NCI Australia).

4.4 Multi-task Fine-tuning

In accordance with multi-task fine-tuning, we are fine-tuning DistilBERT on two tasks: first on SQuAD v1.1 and then on our downstream task of biomedical question answering.

We employ *distilbert-base-cased* as our pretrained model with an additional task-specific layer (linear layer on top of the hidden-states output) to compute span start logits and span end logits). All the layers are trainable in the fine-tuning process. When fine-tuning the model on SQuAD v1.1, we use the default parameters recommended by the DistilBERT authors [55].

In the second phase of fine-tuning on BioASQ data, the model is fine-tuned with a batch size of 4 and learning rate in the range of 9e-6 to 3e-5. We perform an extensive hyper-parameter sweep on the low-resource BioASQ dataset in line with Devlin et al.'s [8] observation that smaller datasets are more sensitive to hyper-parameters than the larger datasets when fine-tuning the transformer-based models. All the other hyper-parameters are set to default values as suggested by the authors of DistilBERT [55] unless stated otherwise.

DistilBERT takes 512 as the maximum sequence length after WordPiece tokenization. Any input sequence longer than this set value will be truncated and shorter will be padded. This value can be set depending on the length of the input sequence.

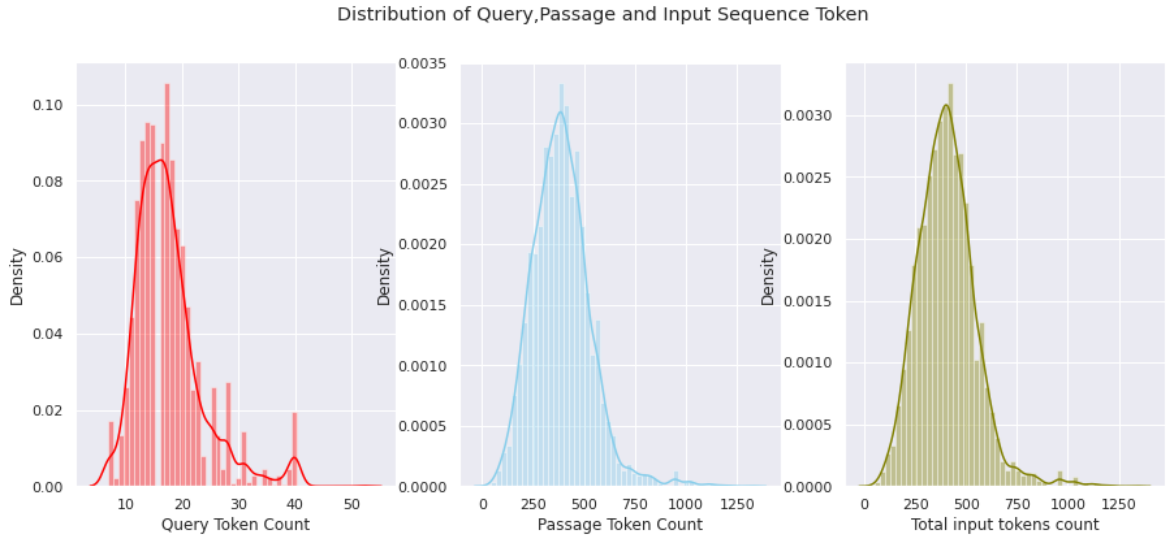

FIGURE 4.2: Distribution of the total number of tokens in Query (a), Passage (b) and Input sequence (c) of the BioASQ 7b Training Data after WordPiece Tokenization. Here the input sequence is [CLS]+[Question tokens]+[SEP]+[Passage tokens]. It is truncated for sequences longer than 512 and padded for shorter sequences.

In the context of a QA setting, the input sequence is the concatenation of the [CLS] token with the Question and Passage tokens separated by the [SEP] token. The full abstract version of the BioASQ dataset that we use for fine-tuning has the initial snippet sentence along with its full abstract and title as its passage. Using the full abstract dataset not only extends the search space for extracting the answer, but also increases the number of tokens in the passage and in turn the input sequence. From our initial analysis, we observe that the total number of tokens in the input sequence after WordPiece tokenization is in the range of 51-1329. The frequency distribution of the tokens in question, passage and input sequence are plotted using the histograms shown in Figure 4.2.

In Table 4.2, we report the evaluation results of two of our models: one fine-tuned with input sequence length of 512 and the other one fine-tuned with sequence length of 384. The sequence length value of 384 is chosen for comparison as it is the default value suggested by BERT authors [8]. All four BioASQ 7b test batches are evaluated on the three metrics: SAcc, LAcc and MRR for both the models. From the comparison, we observe that there is a 4-6% increase in all the three metrics when the model is fine-tuned with 512 input sequence tokens. Thus, we set the sequence length as 512, for all of our models to handle the long sequence of tokens in our BioASQ 7b full abstract training dataset.

System Name	Factoid		
	Strict Accuracy (SAcc)	**Lenient Accuracy (LAcc)**	**Mean Reciprocal Rank (MRR)**
DistilBERT (max_seq_length = 384)	0.3868	0.4902	0.4255
DistilBERT (max_seq_length = 512)	**0.4403**	**0.5588**	**0.4844**

TABLE 4.2: Comparison of systems with different maximum sequence lengths. The scores of all the four test batches are averaged for each metric. 'DistilBERT (max_seq_length = 512)' and 'DistilBERT (max_seq_length = 384)' indicate models fine-tuned with input sequence lengths of 512 and 384 respectively on BioASQ 7b full abstract dataset. The best score is in bold.

Additionally, we conduct experiments on BERT to perform a comparative analysis with DistilBERT. We adapt the same multi-task fine-tuning strategy and first fine-tune BERT on SQuAD v1.1 and then on the BioASQ 7b dataset. We use the batch size of 4, 8 and same learning rate range and maximum sequence length as DistilBERT.

We report our final results on the four BioASQ 7b test batches in Table 4.3. We compare our results with the top ranking systems of the BioASQ 7b (Phase B) Challenge [63]. The top ranking systems for each test batch along with their scores are listed in the table. The systems with highest score are in bold.

The top performing systems among those who participated in BioASQ 7b for four test batches use BioBERT as their pretrained language model. The systems that start with 'KU-DMIS' are BioBERT based QA systems [77] and are the top scorers in four out of the five test batches of BioASQ 7b Challenge. Our system 'DistilBERT fine-tuned' yields good results on Test Batch 2. DistilBERT obtains higher MRR score (+2%) than the top performing system for Test Batch 2. Both the fine-tuned BERT and DistilBERT models score the lower MRR scores on Test Batch 4 and Test Batch 5.

Test Batch Number	System Name	Factoid		
		Strict Accuracy (SAcc)	Lenient Accuracy (LAcc)	Mean Reciprocal Rank (MRR)
2	KU-DMIS-5	0.5200	**0.6400**	0.5667
	DistilBERT (fine-tuned)	**0.5600**	**0.6400**	**0.5867**
	BERT (fine-tuned)	**0.5600**	**0.6400**	0.5800
3	KU-DMIS-1	0.3793	**0.6207**	0.4724
	DistilBERT (fine-tuned)	**0.4138**	0.5172	0.4655
	BERT (fine-tuned)	**0.4138**	0.5517	**0.4770**
4	KU-DMIS-1	**0.5882**	**0.8235**	**0.6912**
	DistilBERT (fine-tuned)	0.5588	0.7353	0.6186
	BERT (fine-tuned)	0.5294	0.6176	0.5510
5	KU-DMIS-5	**0.2857**	**0.5143**	**0.3638**
	DistilBERT (fine-tuned)	0.2286	0.3429	0.2667
	BERT (fine-tuned)	0.2286	0.3714	0.2819

TABLE 4.3: System comparison run on four BioASQ test batches. The best score for each test batch is in bold.

4.5 Results and Discussion

This section discusses the results pertaining to the first research question of our thesis. Our first research question was whether a smaller pretrained language model like DistilBERT can improve the quality of current biomedical question answering systems.

Table 4.4 lists the results of our fine-tuning experiments along with the current state-of-the-art system. We use paired t-tests to know how significant the differences between the groups are and observe that there is no statistically significant difference between our models and the top ranked model. This might be due to the small sample used for t-tests. Thus, we conclude that our system does not improve the accuracy in comparison to the current state-of-the-art question answering system. In spite of being 40% smaller than other models listed and without domain specific pretraining, DistilBERT has achieved comparable results to BERT on a small dataset like BioASQ.

System Name	Mean Reciprocal Rank
KU-DMIS Team [63, 77]	0.5235
DistilBERT (fine-tuned)	0.4844
BERT (fine-tuned)	0.4725

TABLE 4.4: Results of DistilBERT and BERT systems. We include the results of 'KU-DMIS Team' system for comparison. The 'KU-DMIS Team' system is the top-ranked system on the BioASQ 7b leaderboard. The reported results are the average MRR of the four test batches of BioASQ 7b. MRR was the main metric used by the BioASQ organisers.

DistilBERT was 3.9 points behind BERT in test accuracy on a larger SQuAD dataset when evaluating on exact match and F1 score [55]. However, on low-resource BioASQ dataset, not only has DistilBERT closed the gap but also resulted in comparable performance to BERT. Thus using DistilBERT for small datasets is better considering there is no drop in performance and requires less compute power. Most of this improvement can be attributed to our fine-tuning strategies.

Based on our results, DistilBERT is a good alternative to BERT for small datasets like BioASQ. This paves the way for DistilBERT to be used for other niche domains that have less human annotated data. Furthermore, since the top ranked model, BioBERT [77] is pretrained on biomedical data, it would be worth exploring the performance of using distilled version of BioBERT.

5

Gradual Unfreezing Experiments

In this chapter, we describe the experiments that we run in order to answer our second research question (introduced in Section 1.1). Our second research question was whether gradual unfreezing can improve the performance of DistilBERT on a biomedical question answering task. We first lay out our baseline system that we use to compare our gradual unfreezing approach performance with. Later, we describe our experiments with the unfreezing approach and discuss the results of our experiments in detail.

The layout of this is as follows. Section 5.1 describes the baseline system which we use to compare our results. Section 5.2 describes the unfreezing approach that we use, and the hyper-parameters that we choose. Section 5.3 describes unfreezing when grouping transformer blocks. Section 5.4 presents and discusses the results.

5.1 Baselines

Catastrophic forgetting [32] has been detrimental when transferring knowledge learned from a pretrained task to a target task. The risk of forgetting the knowledge learned from earlier stages undermines the potential of transfer learning. Gradually unfreezing layers instead of fine-tuning all the layers of the pretrained model together helps overcome the catastrophic forgetting [17]. We test this fine-tuning approach on a transformer-based model, DistilBERT with biomedical extractive question answering as our NLP use-case.

Before we start our experiments of fine-tuning with gradual unfreezing, we need a baseline system to compare our approach with. We use our 'DistilBERT (fine-tuned)' system from Chapter 4 as the baseline system. 'DistilBERT (fined-tuned)' is a strong baseline due to our fine-tuning settings and its results are comparable to the current state-of-the-art systems on the BioASQ 7b dataset. To recall, we used a multi-task fine-tuning approach of fine-tuning first on SQuAD v1.1 and then on the BioASQ dataset. Both the fine-tuning stages use the standard fine-tuning process i.e., all the layers are trained at once.

5.2 Unfreezing Experiments

In this section, we describe the experiments that we ran to investigate the gradual unfreezing technique on DistilBERT for extractive QA setting. We use the full abstract version of BioASQ 7b training data to train our model and use Test Batch 1 as our validation data to tune our hyper-parameters. The remaining four test batches from BioASQ 7b are evaluated to provide a comparison against our baseline and systems that achieved the best score in the BioASQ challenge. All the systems are evaluated on the three metrics: SAcc, LAcc and MRR as discussed in Section 3.2.

During the fine-tuning on biomedical data, we apply the gradual unfreezing approach discussed in Section 3.4. We start from the top task-specific layer and train the model for one epoch in order to learn the distributions of the target task while freezing all the other layers. Then we gradually unfreeze the next top layer and continue training the unfrozen layers for one more epoch. This gradual fine-tuning is continued until all the layers of DistilBERT converge except the embedding layer. The embedding layer is never fine-tuned in our approach.

We fine-tune the pretrained model *distilbert-base-cased* [73] using AdamW optimizer along with a linear scheduler with a warm up. All the fine-tuning setting for SQuAD v1.1 are same

as before and are detailed in Section 4.4. During this stage of fine-tuning, all of the model's parameters are trained together. We use a batch size of 4 and learning rate in the range of 9e-6 to 3e-5 with a maximum sequence length of 512 when fine-tuning on BioASQ. It is worth noting that the gradual unfreezing approach is only applied when fine-tuning on the BioASQ dataset. In our experiments, in order to avoid overfitting on the training data, the total number of epochs the model is fine-tuned during gradual unfreezing is set to seven epochs. We choose seven as our upper limit for number of epochs considering the fact that there are seven layers in DistilBERT (six transformer layers and an additional question answering layer excluding the embedding layer) and each layer needs to be fine-tuned at least once. We do not change the learning rate for each layer and use the same set learning rate for each cycle of unfreezing.

In the first set of experiments, we compared the impact of applying gradual unfreezing with and without including the embedding layer in the list of layers to unfreeze. The results are shown in Figure 5.1. We observe an increase in the Mean Reciprocal Rank in the experiments that did not unfreeze the embedding layer. Thus, we conclude that removing the embedding layer during our unfreezing process has no negative impact on the performance of the system. Freezing the embedding layer throughout the fine-tuning process also reduced the number of trainable parameters from 65.19 million to 42.52 million, i.e., a 34.76% reduction in the number of trainable parameters. We therefore decide to keep the embedding layer frozen for all our experiments hereafter.

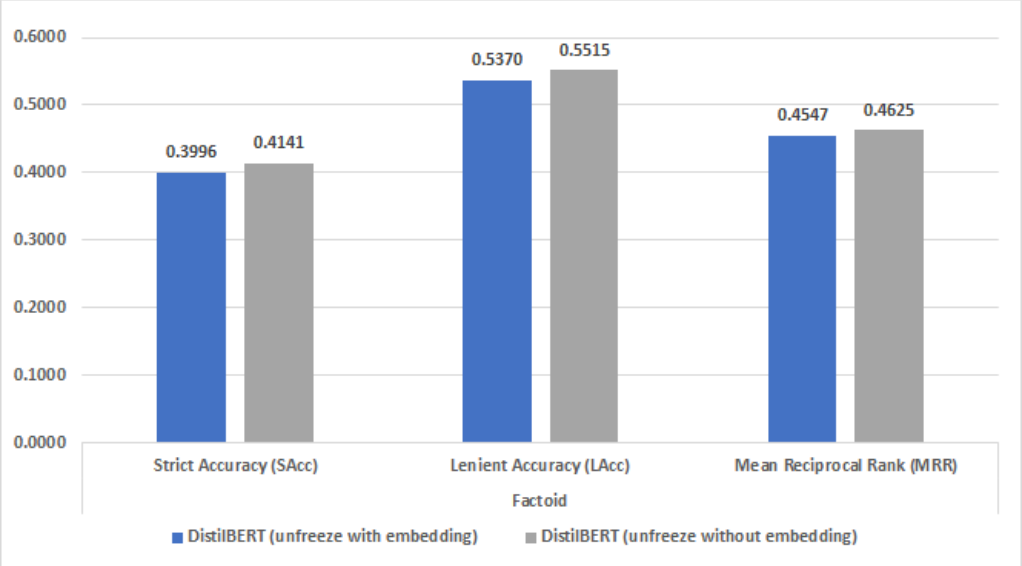

FIGURE 5.1: Comparison of systems when unfreezing one layer at a time with and without embedding layer. The scores of all the four test batches are averaged for each metric. 'DistilBERT (unfreeze with embedding)' and 'DistilBERT (unfreeze without embedding)' indicate models unfreezed with and without embedding layer respectively on four test batches from BioASQ 7b.

5.3 Grouping Blocks of Transformers

In addition, we also apply the gradual unfreezing approach when grouping the transformer layers. Figure 3.3 shows the gradual unfreezing process when three transformer layers are fine-tuned together. In this approach, we start from the top-most task-specific layer, and fine-tune it for few epochs while the other layers are frozen. Next, we unfreeze transformer layers consecutively in groups of either 3 or 6 and fine-tune them for a few epochs until all the layers except the embedding layer are fine-tuned. Our approach differs from the standard gradual unfreezing, as we have grouped the transformer layers and kept the embedding layer always frozen. Also, the number of epochs during each cycle of unfreezing is not always one in our approach as described.

We conducted an extensive search and selected the combination of epochs for unfreezing layers which resulted in the best MRR on validation data (Test Batch 1). We notice that the top task-specific layers required more epochs than the transformer layers (refer Table 5.1). This is understandable given that the task-specific layer weights are the only randomly initialised weights during fine-tuning and therefore require more epochs to learn the target task distributions.

DistilBERT (unfreeze 1)	**DistilBERT (unfreeze 3)**	**DistilBERT (unfreeze 6)**
1,1,1,1,1,1,1	3,1,1	5,1

TABLE 5.1: The table lists the sequence of epochs utilized by the models: 'DistilBERT (unfreeze 1)', 'DistilBERT (unfreeze 3)' and 'DistilBERT (unfreeze 6)'. We list the sequence of epochs used during the gradually unfreezing of layers. This combination of epochs was selected as it gave the best MRR on validation data (Test batch 1) of BioASQ data.

In Table 5.1, we report the sequence of epochs the model is fine-tuned for when unfreezing 1, 3 and 6 transformer layers at once. The sequence in the table represents the number of epochs the model is fine-tuned for during each cycle of unfreezing starting from the top task-specific layer. For example, the sequence 3,1,1 implies that model fine-tunes for 3 epochs after unfreezing question-answering layer, 1 epoch for the next unfrozen layers group (QA layer and top 3 transformer layers) and finally 1 epoch for all the remaining unfrozen layers (all layers except embedding layer).

Test Batch Number	System Name	Factoid		
		Strict Accuracy (SAcc)	Lenient Accuracy (LAcc)	Mean Reciprocal Rank (MRR)
2	DistilBERT (fine-tuned)	**0.5600**	0.6400	**0.5867**
	DistilBERT (unfreeze 1)	0.5200	0.6400	0.5633
	DistilBERT (unfreeze 3)	0.5200	0.6000	0.5500
	DistilBERT (unfreeze 6)	0.5200	**0.6800**	0.5833
3	DistilBERT (fine-tuned)	0.4138	**0.5172**	0.4655
	DistilBERT (unfreeze 1)	0.3793	**0.5172**	0.4425
	DistilBERT (unfreeze 3)	**0.4483**	**0.5172**	0.4741
	DistilBERT (unfreeze 6)	**0.4483**	**0.5172**	**0.4828**
4	DistilBERT (fine-tuned)	**0.5588**	0.7353	0.6186
	DistilBERT (unfreeze 1)	0.5000	0.7059	0.5647
	DistilBERT (unfreeze 3)	**0.5588**	**0.7647**	**0.6422**
	DistilBERT (unfreeze 6)	0.5294	0.7059	0.5917
5	DistilBERT (fine-tuned)	0.2286	0.3429	0.2667
	DistilBERT (unfreeze 1)	**0.2571**	0.3429	**0.2795**
	DistilBERT (unfreeze 3)	**0.2571**	0.3143	0.2700
	DistilBERT (unfreeze 6)	0.1714	**0.3714**	0.2448

TABLE 5.2: Results of gradual unfreezing approaches compared to the baseline. 'DistilBERT (fine-tuned)' depicts the baseline. 'DistilBERT (unfreeze 1)' denotes fine-tuned model when gradual unfreezing one layer at a time. 'DistilBERT (unfreeze 3)' and 'DistilBERT (unfreeze 6)' represents fine-tuned models when unfreezing 3 and 6 transformer layers at a time, respectively, on BioASQ 7b test batches. The best score for each test batch is in bold.

In Table 5.2, we report our final results when using gradual unfreezing on BioASQ 7b dataset. We compare all the three DistilBERT models that are fine-tuned using gradual unfreezing with our baseline model. We discover that 'DistilBERT (unfreeze 3)' achieves better results in three test batches with the exception of Test Batch 2 relative to the baseline in terms of MRR. Overall, the model that unfreezed one transformer layer at a time achieved worst results amongst all unfreezed models. Since the MRR for each test batch is a normal distribution as it is an average of about hundred samples, we use paired t-tests to let us know how significant the differences between groups are. We observe that there is no statistically significant difference between the three unfreezed models and the baseline. Therefore, our gradual unfreezing approach has not achieved any significant improvement over the baseline. Even though 'distilBERT (unfreeze 1)

shows worst results, the difference is still not statistically significant to confirm that this model performed worst.

5.4 Results and Discussion

In this section, we discuss the results of our experiments to answer our second research question. Our second question was to investigate whether gradual unfreezing can improve the accuracy of DistilBERT on extractive question answering task.

Table 5.3 summaries the results of all our gradual unfreezing experiments along with the current state-of-the-art BioBERT-based system [77] for the biomedical question answering task of BioASQ 7b. We conclude that our fine-tuning technique of gradual unfreezing does not improve the performance of DistilBERT over the baseline system, which uses typical end-to-end fine-tuning.

System Name	Mean Reciprocal Rank
KU-DMIS Team [63, 77]	0.5235
DistilBERT (fine-tuned)	0.4844
DistilBERT (unfreeze 3)	0.4841
DistilBERT (unfreeze 6)	0.4756
BERT (fine-tuned)	0.4725
DistilBERT (unfreeze 1)	0.4625

TABLE 5.3: Comparison of all unfreezing experiments run on the BioASQ dataset along with the top ranked BioBERT based 'KU-DMIS Team' system on BioASQ leaderboard [63, 77]. All the reported results are averaged MRR scores (the main metric used by BioASQ organisers) of four test batches of BioASQ 7b data.

Furthermore, we observe that the systems that have unfreezed transformer layers in the groups of three and six, have achieved comparable results than the 'BERT (fine-tuned)' model. This feat was achieved by the smaller 'DistilBERT' model and, on top of that, by not training the embedding layer during fine-tuning, 34% less trainable parameters of DistilBERT were used during fine-tuning.

6

Conclusion and Future Work

This thesis examined two research questions. First, we investigated whether a smaller pre-trained language model, DistilBERT, could improve the accuracy of a limited human annotated dataset, BioASQ. Second, we investigate whether gradual unfreezing can improve the quality of biomedical question answering task.

We have observed that DistilBERT did not result in any significant reduction of accuracy over the current benchmark model in BioASQ. From the results of our experiments, we discovered that DistilBERT is a better alternative for small datasets like BioASQ than BERT. We have provided a step by step guide to sequential adaptation of pretrained model that serves as the starting point for other question answering tasks with limited annotated data. From our experiments, we observe that DistilBERT achieved comparable results to a larger model, BERT. Thus, DistilBERT is a viable alternative option for deploying biomedical QA applications on edge, mobile and other Internet of Things (IoT) devices.

We conducted multiple experiments of gradual unfreezing on BioASQ dataset. From the results obtained, we can conclude that our gradual unfreezing has no significant impact on the

quality of biomedical question answering task. None of our unfreezed models performed better than the standard baseline fine-tuned model.

6.1 Future Work

In view of the limited availability of human annotated data in biomedical domain, additional fine-tuning on in-domain extractive question answering datasets could lead to an improvement in accuracy. Recently, Lewis et al. [26] devised an extractive question answering dataset from unsupervised data by applying cloze translation approach. We could try to create an exclusive biomedical extractive QA dataset from the unsupervised data from PubMed articles and documents for additional fine-tuning. Along with additional data for training, we would like to investigate the fine-tuning strategies like multi-tasking and residual adapters discussed in Chapter 2.

Language models are pretrained on the general domain corpus and are usually adapted to a target task from a completely different domain. While the general domain corpus used for pretraining can be from a variety of diverse sources, the target data will often have a different distribution. Therefore, Howard and Ruder [17] used an intermediate step of fine-tuning the language model on the target task data before fine-tuning the target task itself. This intermediary step of fine-tuning the language model with text data from BioASQ task before fine-tuning on the extractive question answering task holds potential and needs to be explored further.

References

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. “Neural machine translation by jointly learning to align and translate”. In: *arXiv preprint arXiv:1409.0473* (2014).

[2] Yoshua Bengio et al. “A neural probabilistic language model”. In: *Journal of machine learning research* 3.Feb (2003), pp. 1137–1155.

[3] Piotr Bojanowski et al. “Enriching word vectors with subword information”. In: *Transactions of the Association for Computational Linguistics* 5 (2017), pp. 135–146.

[4] Alexandra Chronopoulou, Christos Baziotis, and Alexandros Potamianos. “An Embarrassingly Simple Approach for Transfer Learning from Pretrained Language Models”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 2089–2095. DOI: 10.18653/v1/N19-1213. URL: https://www.aclweb.org/anthology/N19-1213.

[5] Kevin Clark et al. “Electra: Pre-training text encoders as discriminators rather than generators”. In: *arXiv preprint arXiv:2003.10555* (2020).

[6] Andrew M Dai and Quoc V Le. “Semi-supervised sequence learning”. In: *Advances in neural information processing systems*. 2015, pp. 3079–3087.

[7] Jia Deng et al. “Imagenet: A large-scale hierarchical image database”. In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[8] Jacob Devlin et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume*

1 (Long and Short Papers). Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. URL: https://www.aclweb.org/anthology/N19-1423.

[9] Dumitru Erhan et al. "Why does unsupervised pre-training help deep learning?" In: *Journal of Machine Learning Research* 11.Feb (2010), pp. 625–660.

[10] Bjarke Felbo et al. "Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, Sept. 2017, pp. 1615–1625. DOI: 10.18653/v1/D17-1169. URL: https://www.aclweb.org/anthology/D17-1169.

[11] Siddhant Garg, Thuy Vu, and Alessandro Moschitti. "Tanda: Transfer and adapt pre-trained transformer models for answer sentence selection". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 05. 2020, pp. 7780–7788.

[12] Jonas Gehring et al. "Convolutional Sequence to Sequence Learning". In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70*. ICML'17. Sydney, NSW, Australia: JMLR.org, 2017, pp. 1243–1252.

[13] Ian Goodfellow. *Deep learning / Ian Goodfellow, Yoshua Bengio, and Aaron Courville*. eng. Adaptive computation and machine learning. 2016. ISBN: 9780262035613.

[14] Alex Graves. "Generating sequences with recurrent neural networks". In: *arXiv preprint arXiv:1308.0850* (2013).

[15] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. "Distilling the knowledge in a neural network". In: *arXiv preprint arXiv:1503.02531* (2015).

[16] Stefan Hosein, Daniel Andor, and Ryan McDonal. "Measuring Domain Portability and ErrorPropagation in Biomedical QA". In: *arXiv preprint arXiv:1909.09704* (2019).

[17] Jeremy Howard and Sebastian Ruder. "Universal Language Model Fine-tuning for Text Classification". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, July 2018, pp. 328–339. DOI: 10.18653/v1/P18-1031. URL: https://www.aclweb.org/anthology/P18-1031.

[18] Qiao Jin et al. “Probing Biomedical Embeddings from Language Models”. In: *Proceedings of the 3rd Workshop on Evaluating Vector Space Representations for NLP*. Minneapolis, USA: Association for Computational Linguistics, June 2019, pp. 82–89. DOI: 10.18653/v1/W19-2011. URL: https://www.aclweb.org/anthology/W19-2011.

[19] Jaewoo Kang. “Transferability of Natural Language Inference to Biomedical Question Answering”. In: *arXiv preprint arXiv:2007.00217* (2020).

[20] Ashot Kazaryan, Uladzislau Sazanovich, and Vladislav Belyaev. “Transformer-Based Open Domain Biomedical Question Answering at BioASQ8 Challenge”. In: (2020).

[21] Stefan Kombrink et al. “Recurrent neural network based language modeling in meeting recognition”. In: *Twelfth annual conference of the international speech communication association*. 2011.

[22] Vaishnavi Kommaraju et al. “Unsupervised Pre-training for Biomedical Question Answering”. In: *arXiv preprint arXiv:2009.12952* (2020).

[23] Tom Kwiatkowski et al. “Natural questions: a benchmark for question answering research”. In: *Transactions of the Association for Computational Linguistics* 7 (2019), pp. 453–466.

[24] Zhenzhong Lan et al. “Albert: A lite bert for self-supervised learning of language representations”. In: *arXiv preprint arXiv:1909.11942* (2019).

[25] Jinhyuk Lee et al. “BioBERT: pre-trained biomedical language representation model for biomedical text mining”. In: *arXiv preprint arXiv:1901.08746* (2019).

[26] Patrick Lewis, Ludovic Denoyer, and Sebastian Riedel. “Unsupervised Question Answering by Cloze Translation”. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 4896–4910. DOI: 10.18653/v1/P19-1484. URL: https://www.aclweb.org/anthology/P19-1484.

[27] Tsung-Yi Lin et al. “Microsoft COCO: Common Objects in Context”. In: *European conference on computer vision*. Springer. 2014, pp. 740–755. ISBN: 9783319106021.

[28] Yinhan Liu et al. “Roberta: A robustly optimized bert pretraining approach”. In: *arXiv preprint arXiv:1907.11692* (2019).

[29] Mingsheng Long et al. “Learning transferable features with deep adaptation networks”. In: *arXiv preprint arXiv:1502.02791* (2015).

[30] Ilya Loshchilov and Frank Hutter. “Decoupled weight decay regularization”. In: *arXiv preprint arXiv:1711.05101* (2017).

[31] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. “Effective Approaches to Attention-based Neural Machine Translation”. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, Sept. 2015, pp. 1412–1421. DOI: 10.18653/v1/D15-1166. URL: https://www.aclweb.org/anthology/D15-1166.

[32] Michael McCloskey and Neal J. Cohen. “Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem”. In: ed. by Gordon H. Bower. Vol. 24. Psychology of Learning and Motivation. Academic Press, 1989, pp. 109–165. DOI: https://doi.org/10.1016/S0079-7421(08)60536-8. URL: http://www.sciencedirect.com/science/article/pii/S0079742108605368.

[33] Gábor Melis, Tomáš Kočiský, and Phil Blunsom. “Mogrifier lstm”. In: *arXiv preprint arXiv:1909.01792* (2019).

[34] Stephen Merity, Nitish Shirish Keskar, and Richard Socher. “An analysis of neural language modeling at multiple scales”. In: *arXiv preprint arXiv:1803.08240* (2018).

[35] Tomas Mikolov et al. “Distributed representations of words and phrases and their compositionality”. In: *Advances in neural information processing systems*. 2013, pp. 3111–3119.

[36] Tomas Mikolov et al. “Efficient estimation of word representations in vector space”. In: *arXiv preprint arXiv:1301.3781* (2013).

[37] Anastasios Nentidis et al. “Overview of BioASQ 8a and 8b: Results of the eighth edition of the BioASQ tasks a and b”. In: (2020).

[38] Nishant Nikhil. *Knowledge Distillation Article*. [Accessed 15-December-2020]. 2015. URL: https://towardsdatascience.com/knowledge-distillation-a-survey-through-time-187de05a278a.

[39] Sinno Jialin Pan and Qiang Yang. “A survey on transfer learning”. In: *IEEE Transactions on knowledge and data engineering* 22.10 (2009), pp. 1345–1359.

[40] Jeffrey Pennington, Richard Socher, and Christopher Manning. "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, Oct. 2014, pp. 1532–1543. DOI: 10.3115/v1/D14-1162. URL: https://www.aclweb.org/anthology/D14-1162.

[41] Matthew Peters et al. "Deep Contextualized Word Representations". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, June 2018, pp. 2227–2237. DOI: 10.18653/v1/N18-1202. URL: https://www.aclweb.org/anthology/N18-1202.

[42] Matthew Peters et al. "Dissecting Contextual Word Embeddings: Architecture and Representation". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, 2018, pp. 1499–1509. DOI: 10.18653/v1/D18-1179. URL: https://www.aclweb.org/anthology/D18-1179.

[43] Matthew E. Peters, Sebastian Ruder, and Noah A. Smith. "To Tune or Not to Tune? Adapting Pretrained Representations to Diverse Tasks". In: *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*. Florence, Italy: Association for Computational Linguistics, Aug. 2019, pp. 7–14. DOI: 10.18653/v1/W19-4302. URL: https://www.aclweb.org/anthology/W19-4302.

[44] Pubmed. *PubMed® comprises more than 30 million citations for biomedical literature from MEDLINE, life science journals, and online books.* [Online; accessed 1-December-2020]. 2020. URL: https://pubmed.ncbi.nlm.nih.gov.

[45] Alec Radford et al. "Improving language understanding by generative pre-training". In: (2018).

[46] Alec Radford et al. "Language models are unsupervised multitask learners". In: *OpenAI blog* 1.8 (2019), p. 9.

[47] Pranav Rajpurkar, Robin Jia, and Percy Liang. "Know What You Don't Know: Unanswerable Questions for SQuAD". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Melbourne, Australia:

Association for Computational Linguistics, July 2018, pp. 784–789. DOI: 10.18653/v1/P18-2124. URL: https://www.aclweb.org/anthology/P18-2124.

[48] Pranav Rajpurkar et al. "SQuAD: 100,000+ Questions for Machine Comprehension of Text". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 2383–2392. DOI: 10.18653/v1/D16-1264. URL: https://www.aclweb.org/anthology/D16-1264.

[49] Prajit Ramachandran, Peter Liu, and Quoc Le. "Unsupervised Pretraining for Sequence to Sequence Learning". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, Sept. 2017, pp. 383–391. DOI: 10.18653/v1/D17-1039. URL: https://www.aclweb.org/anthology/D17-1039.

[50] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. "Learning multiple visual domains with residual adapters". In: *Advances in Neural Information Processing Systems*. 2017, pp. 506–516.

[51] Siva Reddy, Danqi Chen, and Christopher D Manning. "Coqa: A conversational question answering challenge". In: *Transactions of the Association for Computational Linguistics* 7 (2019), pp. 249–266.

[52] Michele Resta et al. "Transformer models for question answering at bioasq 2019". In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2019, pp. 711–726.

[53] Sebastian Ruder. "Neural transfer learning for natural language processing". PhD thesis. NUI Galway, 2019.

[54] Sebastian Ruder et al. "Transfer learning in natural language processing". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials*. 2019, pp. 15–18.

[55] Victor Sanh et al. "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter". In: *arXiv preprint arXiv:1910.01108* (2019).

[56] Simeon Kostadinov. *Understanding Encoder-Decoder Sequence to Sequence Model Article*. [Accessed 16-December-2020]. 2019. URL: https://towardsdatascience.com/understanding-encoder-decoder-sequence-to-sequence-model-679e04af4346.

[57] Leslie N Smith. "Cyclical learning rates for training neural networks". In: *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE. 2017, pp. 464–472.

[58] Asa Cooper Stickland and Iain Murray. "BERT and PALs: Projected attention layers for efficient adaptation in multi-task learning". In: *arXiv preprint arXiv:1902.02671* (2019).

[59] Emma Strubell, Ananya Ganesh, and Andrew McCallum. "Energy and Policy Considerations for Deep Learning in NLP". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 3645–3650. DOI: 10.18653/v1/P19-1355. URL: https://www.aclweb.org/anthology/P19-1355.

[60] Chi Sun et al. "How to fine-tune bert for text classification?" In: *China National Conference on Chinese Computational Linguistics*. Springer. 2019, pp. 194–206.

[61] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. "Sequence to sequence learning with neural networks". In: *Advances in neural information processing systems*. 2014, pp. 3104–3112.

[62] George Tsatsaronis et al. "An overview of the BIOASQ large-scale biomedical semantic indexing and question answering competition". In: *BMC bioinformatics* 16.1 (2015), pp. 1–28. ISSN: 14712105. DOI: 10.1186/s12859-015-0564-6.

[63] Tsatsaronis et al. *BioASQ Participants Area Task 7b: Test Results of Phase B*. http://participants-area.bioasq.org/results/7b/phaseB/. [Online; accessed 17-January-2021]. 2019.

[64] Joseph Turian, Lev Ratinov, and Yoshua Bengio. "Word representations: a simple and general method for semi-supervised learning". In: *Proceedings of the 48th annual meeting of the association for computational linguistics*. 2010, pp. 384–394.

[65] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems*. 2017, pp. 5998–6008.

[66] Ellen M Voorhees and Donna Harman. "Overview of TREC 2003." In: *Trec*. 2003, pp. 1–13.

[67] Dilin Wang, Chengyue Gong, and Qiang Liu. "Improving neural language modeling via adversarial training". In: *arXiv preprint arXiv:1906.03805* (2019).

[68] Dirk Weissenborn, Georg Wiese, and Laura Seiffe. "Making Neural QA as Simple as Possible but not Simpler". In: *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*. Vancouver, Canada: Association for Computational Linguistics, Aug. 2017, pp. 271–280. DOI: 10.18653/v1/K17-1028. URL: https://www.aclweb.org/anthology/K17-1028.

[69] Georg Wiese, Dirk Weissenborn, and Mariana Neves. "Neural Domain Adaptation for Biomedical Question Answering". In: *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*. Vancouver, Canada: Association for Computational Linguistics, Aug. 2017, pp. 281–289. DOI: 10.18653/v1/K17-1029. URL: https://www.aclweb.org/anthology/K17-1029.

[70] Wikipedia. *One-hot — Wikipedia, The Free Encyclopedia*. [Online; accessed 9-February-2020]. 2020. URL: https://en.wikipedia.org/w/index.php?title=One-hot&oldid=936656342.

[71] Wikipedia. *Word embedding — Wikipedia, The Free Encyclopedia*. [Online; accessed 9-February-2020]. 2020. URL: https://en.wikipedia.org/w/index.php?title=Word_embedding&oldid=936593210.

[72] Adina Williams, Nikita Nangia, and Samuel Bowman. "A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, 2018, pp. 1112–1122. URL: http://aclweb.org/anthology/N18-1101.

[73] Thomas Wolf et al. "Transformers: State-of-the-Art Natural Language Processing". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45. URL: https://www.aclweb.org/anthology/2020.emnlp-demos.6.

[74] Wonjin Yoon. *Pre-trained Language Model for Biomedical Question Answering BioBERT*. https://github.com/dmis-lab/bioasq-biobert. [Online; accessed 13-January-2021]. 2019.

[75] Yonghui Wu et al. "Google's neural machine translation system: Bridging the gap between human and machine translation". In: *arXiv preprint arXiv:1609.08144* (2016).

[76] Zhilin Yang et al. "Breaking the softmax bottleneck: A high-rank RNN language model". In: *arXiv preprint arXiv:1711.03953* (2017).

[77] Wonjin Yoon et al. "Pre-trained Language Model for Biomedical Question Answering". In: *arXiv preprint arXiv:1909.08229* (2019).

[78] Yukun Zhu et al. "Aligning books and movies: Towards story-like visual explanations by watching movies and reading books". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 19–27.

A

Appendix: BioASQ Factoid Question

An example of factoid question from the BioASQ 7b dataset before pre-processing:

```
{
  "body": "Which is the third subunit of the TSC1-TSC2 complex
      upstream of mTORC1?",
  "documents": [
    "http: //www.ncbi.nlm.nih.gov/pubmed/22795129"
  ],
  "ideal_answer": [
    "TBC1D7 was identified as a stably associated and
       ubiquitous third core subunit of the TSC1-TSC2 complex
       .... "
  ],
  "exact_answer": [
```

```
    "TBC1D7"
  ],
  "concepts": [
    "http: //www.uniprot.org/uniprot/TSC1_HUMAN",
    "http: //www.biosemantics.org/jochem#4266396",
    "http: //amigo.geneontology.org/cgi-bin/amigo/term_details
       ?term=GO:0033596",
    "http: //amigo.geneontology.org/cgi-bin/amigo/term_details
       ?term=GO:0031931",
    "http: //www.uniprot.org/uniprot/TSC2_HUMAN"
  ],
  "type": "factoid",
  "id": "5319ac99b166e2b806000034",
  "snippets": [{
      "offsetInBeginSection": 0,
      "offsetInEndSection": 328,
      "text": "The tuberous sclerosis complex (TSC) tumor
         suppressors form the TSC1-TSC2 complex, which limits
         cell growth in response to poor growth conditions
         .....",
      "beginSection": "abstract",
      "document": "http: //www.ncbi.nlm.nih.gov/pubmed/2279512
         9",
      "endSection": "abstract"
    },
    {
      "offsetInBeginSection": 329,
      "offsetInEndSection": 633,
      "text": "Here, we identify and biochemically
         characterize TBC1D7 as a stably associated and
         ubiquitous third core subunit of the TSC1-TSC2 complex
         ..... ",
```

```
      "beginSection": "abstract",
      "document": "http: //www.ncbi.nlm.nih.gov/pubmed/22795129",
      "endSection": "abstract"
    }
  ]
}
```


An example of factoid question from the BioASQ 7b dataset after pre-processing into SQuAD format:


```
{
  "qas": [{
    "id": "5319ac99b166e2b806000034_001",
    "question": "Which is the third subunit of the TSC1-TSC2 complex upstream of mTORC1?",
    "answers": [{
      "text": "TBC1D7",
      "answer_start": 449
    }]
  }],
  "context": "The tuberous sclerosis complex(TSC) tumor suppressors form the TSC1 - TSC2 complex, which limits cell growth in response to poor growth........"
},
{
  "qas": [{
    "id": "5319ac99b166e2b806000034_002",
    "question": "Which is the third subunit of the TSC1-TSC2 complex upstream of mTORC1?",
    "answers": [{
      "text": "TBC1D7",
```

```
      "answer_start": 573
    }]
  }],
  "context": " Here, we identify and biochemically
     characterize TBC1D7 as a stably associated and ubiquitous
     third core subunit of the TSC1-TSC2 complex. We
     demonstrate that the TSC1-TSC2-TBC1D7 (TSC-TBC) complex is
      the functional complex..........."
}
```