Develop New Transformer Architecture For Question and Answering

Nirbhay P. Tandon

June, 2021

Contents

List of Figures

List of Tables

List of Abbreviations

ALBERT	A LITE BERT
BERT	Bidirectional Encoder Representations from Transformers
BiLSTM	Bidirectional LSTM
BPPTs	Back-Propagation Through Time
CEC	Constant Error Carousel
CNNs	Convolutional Neural Networks
DistilBERT	Distilled Version of BERT
EM	Exact Match
LSTMs	Long Short-Term Memory Architectures
MCTest	Machine Comprehension of Text
NLP	Natural Language Processing
NLU	Natural Language Understanding
QASENT	A challenge dataset for open-domain question answering
QnA	Question and Answering
RACE	ReAding Comprehension Dataset From Examinations
RNNs	Recurrent Neural Networks
RoBERTa	Robustly Optimized BERT
SQuAD	Stanford Question Answering Dataset

Abstract

Attention-based Transformer architectures have become the norm of current Natural Language Processing applications. Google began this trend back in 2017 with their paper *Attention Is All You Need*, by introducing the Transformer architecture that works solely on attention mechanisms. The purpose of our work will be to create a new kind of Transformer architecture. Compare and contrast its performance against other architectures such as BERT, DistilBERT, ALBERT etc. using the SQuAD 2.0 Dataset. Through our research, we aim to produce a new architecture that has a better performance than existing models specifically for Question Answering based applications.

Chapter 1

Introduction

⟨c1introduction⟩ One of the most philosophical questions that anyone can encounter is *"Can an answer exist without a question?"*. While that question exists to be solved by philosophers and thinkers, we shall aim to address the *answer* that can be provided to a question in terms of the context that has been asked. While the field of Natural Language Processing(NLP) is vast and many unexplored areas exist, in this thesis the focus is entirely on developing a novel architecture that provides *highly contextual and relevant answers* to the questions asked of it.

The earliest examples of Question-Answering(QA) systems can be traced back to 1961, in the form of BASEBALL,(Green Jr et al., 1961), a system developed by Bert F. Green and team to answer basic questions posed using punch cards in ordinary English language about stored data on baseball games. The team implemented a dictionary structure to allow for answers to be looked up and printed. For example, the system could answer very direct questions such as: "Where did each team play on July 7?". A highly impressive feat for the early days of neural networks in the 1960s, the work done by Green paved the way for systems like LUNAR,(Woods and WA, 1977), answered geological questions for the rocks brought back from the Apollo moon missions. LUNAR, which was first presented in 1971, had an astounding 90% accuracy rate for questions that were asked by people who didn't know how the system worked! Question-Answering based systems have gained a lot of popularity, especially in the form of "chatbots". These systems depend highly on contextual understanding of the input, the training corpus and the question asked. They use this knowledge to output an answer that can help the user with whatever their query is. Recurrent neural networks and architectures based on them, have been able to provide great advancements in the field of Question-answering and chatbots in general. However, there is a behaviour of over-fitting and a lack of contextual understanding of the

question. This, coupled with long training times and extremely complex mathematical model designs, have often kept the field of Natural Language Processing slightly obscured from the masses.

The task of Question-Answering is a machine comprehension problem. If we dig deeper, we can classify it even as a sequential problem. One might wonder how it is a sequential problem? The answer to that is simple, traditionally QA systems work by "sequentially reading" the input and In a more mathematical context, we can say that the problem we face is of *answer selection* and not *answer provision*. Through numerous studies that have been done, some of which we shall look into detail in chapter **??**, we see that the problem is not how to provide an answer to a question, it is if the answer provided is the correct one or not in the *context* in which the question has been posed.

Through the work done in this research, the author aims at creating a sustainable, fast, easy to understand Transformer architecture for Question Answering. An advancement on the work done by the team at Google (Vaswani et al., 2017). To be able to do so, let us first understand what a *Transformer* is. A Transformer is a form of transduction model that relies solely on self-attention to figure out how to represent its inputs and outputs. It does so without the use of any sequence aligned recurrent neural networks(RNNs) or convolutions.

1.1 Background Of The Study

⟨c1background⟩ In this section, we shall highlight what has led us this far and some of the interesting challenges that it poses. In **??**, we briefly look at the history of Natural Language Processing and how some of the challenges were addressed. In **??**, we look at the latest research that has gone into creating the Transformer architecture, identify some of the common patterns and use that information to strategise our model in later sections.

The area of Natural language Processing has taken significant leaps in the last two decades. Work done towards improving the ability of machine learning models to first recognize words, then sentences, followed by contextual understanding has led to several interesting and novel approaches in the field. From early on neural networks to creating Long Short-Term Memory architectures (Schmidhuber and Hochreiter, 1997) by Sepp Hochreiter and Jurgen Schmidhuber in the mid-'90s that resolved the vanishing gradient problem of classical neural networks, we have come a long way.

The latest advancements in this field come from Google's research lab in the form of *Transformers*. We look into this in a bit more detail later.

However, no model can be successful without a good dataset to train on. This is where the SQuAD 2.0 dataset (Rajpurkar et al., 2018) comes in. This dataset is what forces the machine learning models to do contextual understanding. One might even say that it forces the models to "think" for themselves before answering a task.

Let us now look at some of the key research that has been done in this regard.

1.2 Aims And Objectives

⟨12⟩ The main aim of this research is to propose a new transformer architecture that can perform better at Q&A using the SQuAD 2.0 dataset. We shall:

1. Implement the existing models that are available via libraries such as HuggingFace (Team), PyTorch and Tensorflow on the dataset
2. Obtain F1, Exact Match(EM), etc. scores for existing models and treat them as our benchmark scores
3. Identify drawbacks of the current architectures
4. Design our architecture and evaluate its performance
5. Fine-tune the architecture, re-evaluate and report improvements
6. Compare the results of our Transformer model with the benchmark scores.

1.3 Scope Of The Study

?⟨13⟩? This research work is entirely focused on the area of Question-Answering based networks. This focus helps in significantly narrowing down the scope and critically assessing the problem that needs to be solved. The practical applications of the field of QA are endless. From chatbots to QA systems on domain-specific knowledge or conversational applications for voice assistants, a light, robust, highly accurate Transformer based architecture can help ease the daily lives of millions of users.

1.4 Significance Of The Study

?⟨14⟩? The current state of the art systems, some of which have been highlighted in section 1.1, suffer from various drawbacks and even the best models have only been able to achieve an EM score of 90.871 with an F1 score of 93.183 on the SQuAD 2.0 leaderboard (Rajpurkar, 2021).

1.5 Structure Of The Study

?⟨15⟩? We have divided this report into 3 main chapters. Each chapter is broken down into sections and subsections to localize the reference and use it further in the global context of this research. In chapter 1, an introduction to the concept of Question-Answering, a brief look at the problem, the motivation, the scope of this research and its significance have been provided. Chapter 2, looks in detail at the background studies that have been briefly mentioned in section 1.1. In this chapter critical analysis is conducted how some of the work that has been selected is relevant to what this research aims to highlight, it helps us understand the merits and shortcomings of current state of the art approaches. Finally, Chapter 3, details the research methodology used by the author. Chapter 3 helps the reader understand various nuances of the dataset, evaluation metrics, pre-processing techniques, benchmarks from existing systems etc. Appendix A is where the reader can refer to the Gantt chart for the thesis work done and the updates to various stages.

Chapter 2

Literature Review

⟨c2litrev⟩ To be able to effectively deliver on the aims and objectives we have defined in Section 1.2, we must first critically analyse the work that has been done in the field of NLP and, in particular, the field of Question-Answering based networks. In this section, we will critically analyse a few key model architectures and how we reached Transformers, which form the base of our proposed architecture.

2.1 Long Short-Term Memory

?⟨22⟩? In 1997 Schmidhuber and Hochreiter (1997) introduced to the world a gradient descent based model in the form of Long Short-Term Memory. They set out to solve the problem of the vanishing gradient which was often seen in "Back-Propagation Through Time"(BPPTs) based neural networks. Extensive studies done on this, some by Hochreiter himself, showed that the problem of vanishing gradients is a real one. It was also seen that in case of BPTTs, the error back-flow mechanisms would either blow up or also suffer from vanishing gradients leading to either oscillating weights or learning to bridge long time gaps would not work. To remediate this, the authors introduced the concept of a *constant* error flow through the internal states of special units.

Their paper, Long Short-Term Memory, (Schmidhuber and Hochreiter, 1997), highlights that the work previously done using various gradient-descent variants, time-delays, time constants, use of higher order units to influence connections when the network receives conflicting error signals, Kalman filters, Second order nets etc. still suffer from some of the same problems that we have earlier. In the case of Ring's approach (Ring, 1993), the network can be really fast but also suffers from memory constraints and lack of gen-

eralization. This happens because to bridge n time lags, the model adds n additional units.

It is essential for us to understand how LSTMs function as it forms the basis of Transformers to exist. Hochreiter and Scmidhuber designed a *memory cell* that allows for a constant error flow. This is achieved by introducing *constant error carousel*(CEC), a multiplicative input gate unit and a multiplicative output gate unit. These *multiplicative* units help shield the contents of the memory cell from irrelevant inputs. The CEC, input and output gates together form the *memory cell*, as seen in Fig. 2.1. The central feature for

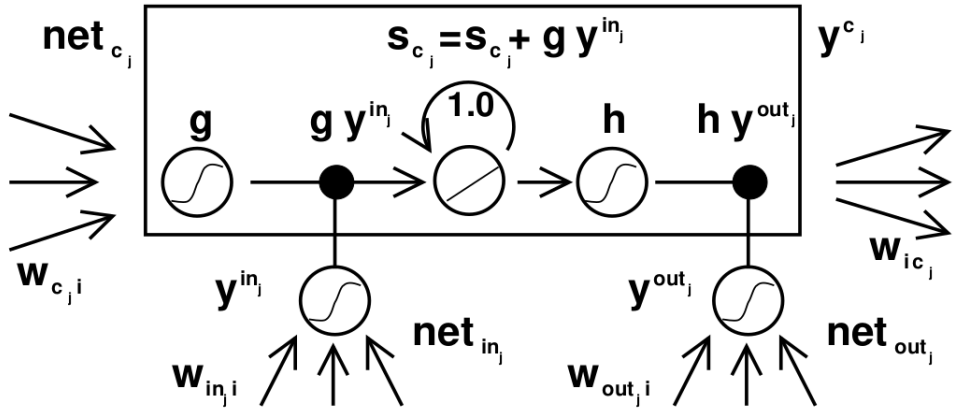

Figure 2.1: LSTM Memory cell with a Constant Error Carousel having fixed weight 1.0, (Schmidhuber and Hochreiter, 1997)

⟨lstmCEC⟩

LSTM's memory cell is the CEC. The self-recurrent connection, where weight is 1.0, is what helps create a time-delayed feedback loop by 1.

The CEC uses gate units to circumvent the issue with conflicting weights in the input and output layers. The input layer has control over when to use and when to override the information in the CEC but has no control over the error signals that are in the memory cell. The output gate, however, uses a feedback mechanism to assess when to superimpose different error signals. It also falls on the output gate to *learn* which errors to trap in the CEC by appropriately scaling them. The gates have to learn when to release and trap errors therefore controlling the access to the CEC, helping maintain constant error flow. To mathematically understand the memory cell from fig. 2.1, Schmidhuber and Hochreiter proposed the following set of equations 2.1 - 2.3. More details on this can be found by referring to the appendix section A.1 in (Schmidhuber and Hochreiter, 1997).

$$y^{out_j}(t) = f_{out_j}\left(net_{out_j}(t)\right); y^{in_j}(t) = f_{in_j}\left(net_{in_j}(t)\right) \tag{2.1}$$ eq1

where

$$net_{out_j}(t) = \sum_u w_{out_j u} y^u(t-1) \tag{2.2}$$?eq2?

and

$$net_{in_j}(t) = \sum_u w_{in_j u} y^u(t-1) \tag{2.3}$$ eq3

A generalised form of this equation can be represented as:

$$net_{c_j}(t) = \sum_u w_{c_j u} y^u(t-1) \tag{2.4}$$?eq4?

At time t, c_j's output $y^{c_j}(t)$ is computed as:

$$y^{c_j}(t) = y^{out_j}(t) h(s_{c_j}(t)), \tag{2.5}$$?eq5?

where the "internal state" $s_{c_j}(t)$ is:

$$s_{c_j}(0) = 0, s_{c_j}(t) = s_{c_j}(t-1) + y^{in_j}(t) g\left(net_{c_j}(t)\right) \text{ for } t > 0 \tag{2.6}$$?eq6?

Where u may stand for input, output or gate units, memory cells or even conventional hidden units, if they are being used.

In fig. 2.2, we see the internal workings of how an LSTM network could look like.

A series of experiments were performed to prove that LSTMs are indeed "state-of-the-art" and provide an improvement over then existing RNN based architectures. Experiments around temporal order, which is defined as a series of events that happen in a particular order, that were previously unsolved by RNNs, were successfully solved by LSTMs. This was possible because of how LSTMs address time lags vs. how RNNs suffer from vanishing gradients. Details of the analysis and experiments conducted are available in section 5 of the paper "Long Short-Term Memory" by Schmidhuber and Hochreiter (1997), with experimental summaries in tables 10 and 11 of the same. Being able to solve for temporal order based problems allowed LSTMs to be capable of solving various real world problems that we will discuss subsequently. Temporal orders are key in contextual understanding for NLP based tasks and especially in the case of QA based networks. Let us now look at some advantages and limitations of LSTMs, before moving on to understand how they have been applied in the field of QA.

Advantages of LSTMs

- LSTMs deal with long time lags by bridging the gaps using constant error flow in the memory cells.

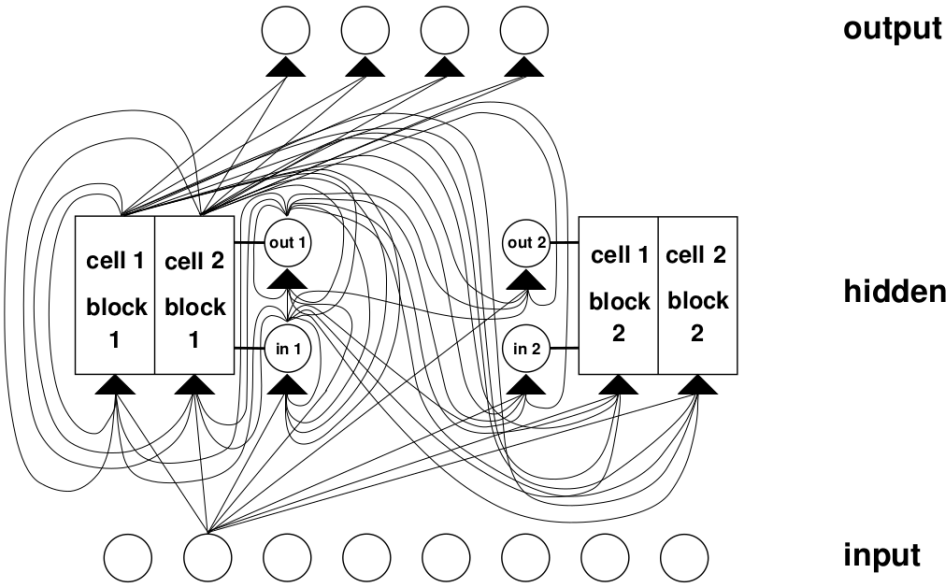

Figure 2.2: LSTM network with 8 input cells, 4 output cells and 2 memory cells of block size 2. Here $in1$ marks the input gate, $out1$ marks the output gate and $cell/block1$ marks the first memory cell block 1. The internal architecture of $cell/block1$ is similar to fig. 2.1. (Schmidhuber and Hochreiter, 1997)

⟨lstmNetwork⟩

- LSTMs respond well to generalization problems, even if the input sequences are widely separated.
- They work well with a variety of braod range hyper-parameters and don't usually require tuning.
- Their time and weight update complexity is the same as BPTT, ie. $O(1)$. The advantage for LSTMs however, is that it is local in both space and time.

Limitations of LSTMs

- LSTMs use memory cells which require an additional input and output unit. This *hidden unit* is replaced by 3 units in an LSTM architecture, compared to 9 in an RNN. A fully connected LSTM will have 3^2 connections however.
- LSTMs dont work well when they get the entire input in one go. The architecture is unable to generalize well by random weight guessing in

this case.

- They do not have the ability to "natively" count discrete time steps which might be useful in some applications. This isn't a big issue for the case of QA based problems.

2.1.1 BiLSTMs

?⟨c2bilstm⟩?

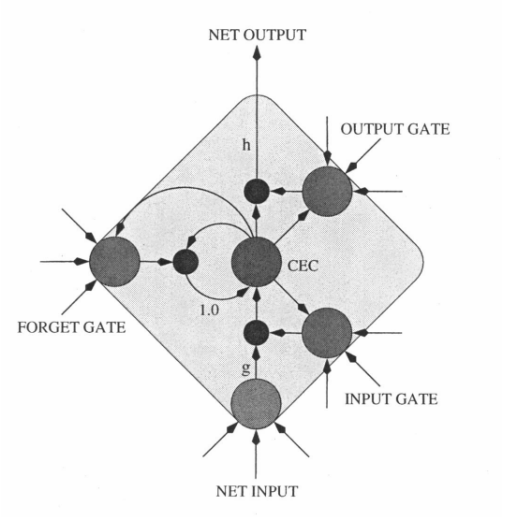

Figure 2.3: A memory cell for a biLSTM highlighting a *forget gate*. This forget gate helps in scaling the internal state of a memory cell, for example by resetting the state to 0. We can see that it is different to fig. 2.1, with the implementation of a connection to the forget gate, while still having an internal weight of 1. (Graves and Schmidhuber, 2005)

⟨lstmBilstm⟩

While LSTMs have found various applications across the field of machine and deep learning, QA seems to be one where being able to achieve consistent results of accuracy as well as having an exact match seems to elusive. Research carried out at IBM by Tan, Santos and co. looked at approaching this problem by using something called Bidirectional LSTMs or BiLSTM, (Graves and Schmidhuber, 2005). Introduced by Graves and Schmidhuber in their paper *Framewise Phoneme Classification with Bidirectional LSTM Networks*, the biLSTM implements a *forget gate* as seen in fig. 2.3. An advantage of biLSTMs over traditional LSTMs is that you feed the data twice, once from the beginning to the end and then from the end to the beginning. This helps the model contextualize and adjust weights better, therefore creating a more robust model that learns and performs better overall when compared to traditional LSTMs and RNNs.

	Questions	Answers	Question Word Count
Train	12887	18540	92095
Dev	1000	1454	7158
Test1	1800	2616	12893
Test2	1800	2593	12905

Table 2.1: InsuranceQA Corpus statistics: first two columns are the question and answer quantity; notice there could be multiple answers for some questions so that the answer quantity is larger than the question quantity; third column is the question total word count. The total number of answers is 24 981 and the whole answer text contains 2 386 749 words(Feng et al., 2015).

stmInsuranceQATable⟩

BiLSTMs were introduced as an approach towards phoneme classification for speech recognition applications, however, in their paper "LSTM-BASED DEEP LEARNING MODELS FOR NON -FACTOID ANSWER SELECTION", (Tan et al., 2015) have shown that extending the biLSTM model 2 different approaches for QA has its advantages. One approach, using a more composite method of using CNNs for representing questions as well as answers. The other, by implementing an *attention mechanism* that helps in generating an answer representation according to the question context. Their framework is based on developing a biLSTM model for both questions and answers, applying a connective pooling layer and using a similarity metric to measure the degree of matching answers. They utilized the InsuranceQA(Feng et al., 2015) dataset that was created in 2015 and tested primarily on CNN based networks, details of the dataset can be found in table 2.1.

Pooling layers are known to suffer from an incapability to retain local linguistic information. To combat this the team proposed an additional CNN layer on top of the biLSTM layer. Additionally, they added a simple and efficient attention model to help their biLSTM model better distinguish between candidate answers according to the question context. This study is significant in the field of both deep learning based models and for the field of Question-Answering because it requires little or no feature engineering to achieve significant results. We will also see later in section 2.2, how biLSTMs are implemented with Transformers to produce an even better model. In their work (Tan et al., 2015) have shown that biLSTMs are capable of exploiting long-range sequential context information. This is important as it is often seen that the answer is not directly semantically related to the question but more contextually related. Long-range exploitations help the model understand the question more holistically, therefore helping produce a more relevant answer. BiLSTMs have an advantage over traditional LSTMs

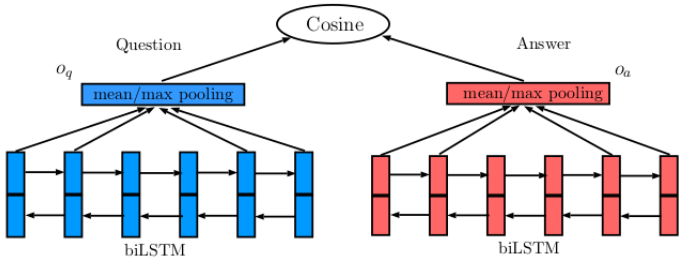

Figure 2.4: Basic QA-LSTM model depicting biLSTM implementation Tan et al. (2015)

⟨lstmhaig⟩

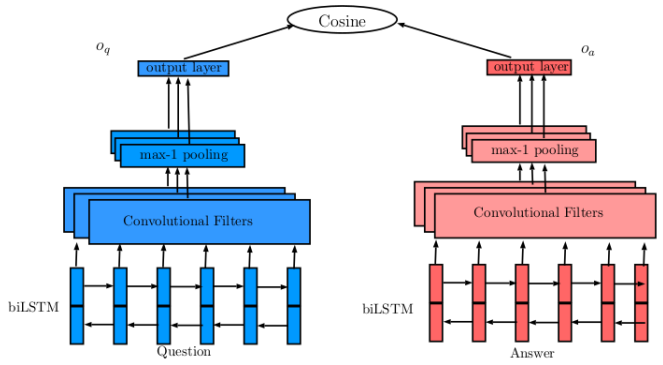

Figure 2.5: QA-LSTM with CNN layer (Tan et al., 2015)

⟨lstmhaigcnn⟩

in that they utilize both previous and future contextual information by processing the input in both directions, forward and backward. This leads to the generation of two independent sequences of output vectors form LSTMs. The final output is a concatenation of the outputs from both directions ie. $h_t = \overrightarrow{h_t}||\overleftarrow{h_t}$. The **QA-LSTM** model proposed by (Tan et al., 2015) can be seen in fig. 2.4. We can see that it generates a distributed representation for both questions(blue) and answers(red) before moving on to perform pooling and cosine similarity operations. In fig. 2.5, it can be seen that the CNN layer has been implemented after the biLSTM layers and before the pooling layers. The pooling layers have been modified to be max-k pooling layers, similar to standard CNNs.

The implementation of this convolution layer forces localized interactions between the inputs within a filter size of m. For each window of size m in biLSTM output vectors ie.

$$\mathbf{H}_m(t) = [\mathbf{h}(t), \mathbf{h(t+1)}, ..., \mathbf{h}(t+m-1)] \quad (2.7)\ \{?\}$$

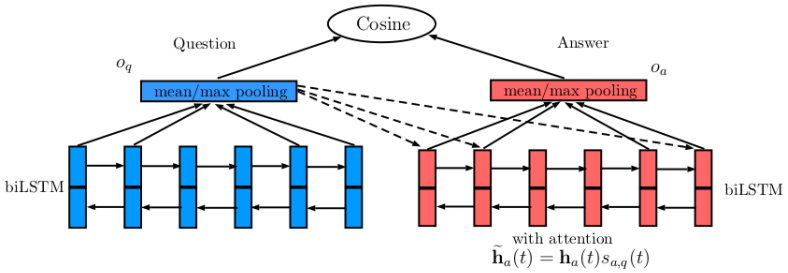

Figure 2.6: QA-LSTM with Attention layer (Tan et al., 2015)

⟨lstmhaigattention⟩

where t is a time step and the convolution filter will generate one value as follows

$$o_F(t) = tanh\left[\left(\sum_{i=0}^{m-1} \mathbf{h}(t+i)^T \mathbf{F}(i)\right) + b\right] \tag{2.8}$$ {?}

Using k-MaxPooling, the authors emphasized that a maximum of k values will be kept for one filter, thereby indicating the highest degree that a filter matches the input sequence. To end this approach, there are N parallel filters, with different parameter initialization which provide the CNN layer with N-dimension output vectors. This produces two output vectors of dimension kN for each question and answer. Their experiments intuitively took $k = 1$ as anything greater showed no improvements and it helped emphasize aspects of the answer such that this CNN based hybrid biLSTM could differentiate ground truths and incorrect answers efficiently.

The second approach adopted in this paper is actually inspired by another paper (Hermann et al., 2015) where attention based deep LSTM networks were used to enhance the capabilities of reading comprehension in LSTM based models. Here, in (Tan et al., 2015), a modification has been introduced in the form of attention-based connections along with biLSTM layers as depicted in fig. 2.6.

Before conducting a mean/average pooling operation on the output from every biLSTM, the output vectors are multiplied by a softmax weight that is determined by the question embeddings from the biLSTM. It was shown that given an output vector from a biLSTM for an answer at time $t, \mathbf{h}_a(t)$ and the question embedding, o_q, the updated vector $\tilde{\mathbf{h}}_a(t)$ for each answer token can be given as:

$$\mathbf{m}_{a,q}(t) = tanh(\mathbf{W}_{am}\mathbf{h}_a(t) + \mathbf{W}_{qm}o_q) \tag{2.9}$$ bilstmAttentioneq

$$s_{a,q}(t) \propto exp(\mathbf{w}_{ms}^T \mathbf{m}_{a,q}(t)) \tag{2.10}$$?bilstmattentioneq2

	Model	Validation	Test1	Test2
A	QA-LSTM basic-model(head/tail)	54.0	53.1	51.2
B	QA-LSTM basic-model(avg pooling)	58.5	58.2	54.0
C	QA-LSTM basic-model(max pooling)	64.3	63.1	58.0
D	QA-LSTM/CNN(fcount=1000)	65.5	65.9	62.3
E	QA-LSTM/CNN(fcount=2000)	64.8	66.8	62.6
F	QA-LSTM/CNN(fcount=4000)	66.2	64.6	62.2
G	QA-LSTM with attention (max pooling)	66.5	63.7	60.3
H	QA-LSTM with attention (avg pooling)	**68.4**	**68.1**	62.2
I	QA-LSTM/CNN (fcount=4000) with attention	67.2	65.7	**63.3**

Table 2.2: Experimental results from (Tan et al., 2015), highlighting how QA-LSTM with attention outperforms its various modifications

experiementresults⟩?

$$\tilde{\mathbf{h}_a} = \mathbf{h}_a(t) s_{a,q}(t) \tag{2.11}$$ bilstmattentioneq3

where $\mathbf{W}_{am}, \mathbf{W}_{qm}$ and $\mathbf{w}_m s$ are attention parameters. Attention parameters show somewhat similar behaviour to tf-idf where some words get more attention or weights associated to them. The main difference being that the attention mechanism adjusts its weights according to question information.

Critically, this attention based approach from (Tan et al., 2015) emphasizes on attention-driven representations and uses that to calculate the distances between questions and their respective answers. A combination approach with CNNs and attention based connections helps in localizing various internal connections, correctly scaling the errors and reducing unnecessary noise. From eq. 2.9-2.11, (Tan et al., 2015) we can mathematically prove how the average distances are computed. Experiments conducted on the InsuranceQA dataset showed that the *QA-LSTM/CNN with Attention* model can outperform the taken baseline models, which were based purely on QAs using CNNs.

An improvement to this approach was done in 2018 by (Brahma, 2018) who proposed *Suffix Bidirectional LSTM* or *SuBiLSTM* adding prefixes and suffixes to the inputs over both directions. There are a few obvious differences between the biLSTM and SuBiLSTM like the doubling of parameters, increased time complexity which happens over quadratic time for worst case scenarios compared to linear in biLSTMs. The author of this paper has shown that despite higher time complexity, there are significant gains in accuracy in text encoding and classification in their experiments, of particular note is the Paraphrase Detection experiment which uses GloVe embeddings and shows an 88.2% score in the test set, which is marginally higher than other models implemented. The work presented in this thesis will also implement

paraphrase detection and GloVe embeddings to potentially improve model performance and answer detection.

2.1.2 LSTM And Question Answering

?⟨c2lstmqa⟩? In 2016, The SQuAD dataset (Rajpurkar et al., 2016) was released, that generated quite a buzz in the field of QA based NLP tasks due to its nature of being extremely versatile and crowdsourced. We will take a deeper look at it in section 2.3, where we discuss the various datasets in detail.

Using the SQuAD dataset (Wang and Jiang, 2016) developed a match-LSTM (Wang and Jiang, 2015) based model that they showed performed significantly better on the dataset than the work done by (Rajpurkar et al., 2016). Their work shows that using a match-LSTM approach combined with a Pointer Net model (Vinyals et al., 2015), that aids token predictions using only input sequences rather than using any form of large fixed vocabulary, therefore allowing for generation of answers with multiple tokens.

The approach by (Wang and Jiang, 2016) was tw-fold. The first approach implemented as a sequence model and the second as a boundary model, which was further extended with a search function.

The match-LSTM model architecture consists of:

- **LSTM Preprocessing Layer**: As we can see from 2.7, there are 2 LSTM preporcessing layers. This layer is used to integrate contextual information directly into the representation of how each token in the passage and question is represented. This is based directly on the simple LSTM (Schmidhuber and Hochreiter, 1997) and not the BiLSTM.

- **Match-LSTM Layer**: Here, the model treats the question as a premise and the passage as hypothesis. The layer sequentially reads the passage and calculates an attention weight vector $\vec{\alpha}_i \in \mathbb{R}^Q$ as:

$$\begin{aligned} \overrightarrow{\mathbf{G}}_i &= \tanh\left(\mathbf{W}^{\mathrm{q}}\mathbf{H}^{\mathrm{q}} + \left(\mathbf{W}^{\mathrm{p}}\mathbf{h}_i^{\mathrm{p}} + \mathbf{W}^{\mathrm{r}}\overrightarrow{\mathbf{h}}_{i-1}^{\mathrm{r}} + \mathbf{b}^{\mathrm{p}}\right) \otimes \mathbf{e}_Q\right) \\ \vec{\alpha}_i &= \mathrm{softmax}\left(\mathbf{w}^\top \overrightarrow{\mathbf{G}}_i + b \otimes \mathbf{e}_Q\right) \end{aligned} \tag{2.12}$$ eqmatchlstm

where $\mathbf{W}^{\mathrm{q}}, \mathbf{W}^{\mathrm{p}}, \mathbf{W}^{\mathrm{r}} \in \mathbb{R}^{l \times l}, \mathbf{b}^{\mathrm{p}}, \mathbf{w} \in \mathbb{R}^l$ and $b \in \mathbb{R}$ are parameters to be learned, $\overrightarrow{\mathbf{h}}_{i-1}^{\mathrm{r}} \in \mathbb{R}^l$ is the hidden vector of the one-directional match-LSTM at position $i-1$, and the outer product $(\cdot \otimes \mathbf{e}_Q)$ produces a matrix or row vector by repeating the vector or scalar on the left for Q times. To simply summarise, the above equation 2.12 produces $\vec{\alpha}_{i,j}$ that gives us the degree of matching between token i in the passage

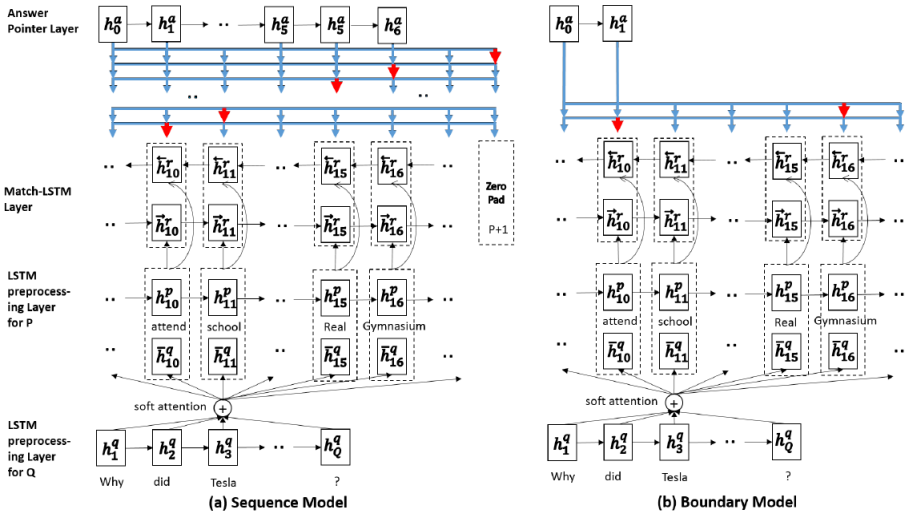

Figure 2.7: Overview of the 2 approaches by (Wang and Jiang, 2016), showing the Sequence Model on the left and the Boundary Model on the right.

⟨lstmMatch⟩

with token j in the question. This is the forward pass representation of the mathc-LSTM, the backward pass, which generates encodings for each token in the passage is given by:

$$\overleftarrow{\mathbf{G}}_i = \tanh\left(\mathbf{W}^{\mathrm{q}}\mathbf{H}^{\mathrm{q}} + \left(\mathbf{W}^{\mathrm{p}}\mathbf{h}_i^{\mathrm{p}} + \mathbf{W}^{\mathrm{r}}\overleftarrow{\mathbf{h}}_{i+1}^{\mathrm{r}} + \mathbf{b}^{\mathbf{p}}\right) \otimes \mathbf{e}_Q\right)$$
$$\overleftarrow{\alpha}_i = \mathrm{softmax}\left(\mathbf{w}^\top \overleftarrow{\mathbf{G}}_i + b \otimes \mathbf{e}_Q\right) \quad (2.13)$$ eqmatchlstmreverse

The equations above, 2.12 and 2.13 are presented in (Wang and Jiang, 2016).

- **Answer Pointer Layer**: This layer, the Answer-Pointer layer(Ans-Ptr), introduced by (Vinyals et al., 2015). It uses the sequence $\mathbf{H}^r$ as input.

It is important to highlight that the sequence model represents the answers as integers in a sequence that refer to the position of the selected token in the original passage. The Ans-Ptr layer models these sequentially as well. For the purposes of this thesis we will focus on the boundary model from the paper by (Wang and Jiang, 2016).

2.2 Transformers

⟨23⟩ Transformer based models have been become a very reliable way to implement various NLP based tasks since their introduction in the late 2017, (Vaswani et al., 2017). In this section we will critically evaluate a few key transformer based architectures & assess their application particularly on the SQuAD 2.0 dataset.

Work done in the field of Long short-term and gated recurrent (Schmidhuber and Hochreiter, 1997) and (Zhou et al., 2016) neural networks, in particular, has been established as a state of the art approach in sequence modelling, transduction problems such as language modelling and machine translation. Efforts have been made to push the boundaries of encoder-decoder based & recurrent language models.

Recurrent models usually adopt a combination approach where they take symbol position of the input and output positions along with the computation. This helps them align positions to steps in computation time and generate a sequence of hidden states h_t, which is a function of the previous hidden state h_{t-1} and the input function t. This inherently sequential nature of RNNs causes memory issues, leading to reduced batch sizes.

The architecture for a *Transformer* in this paper is outlined as having an

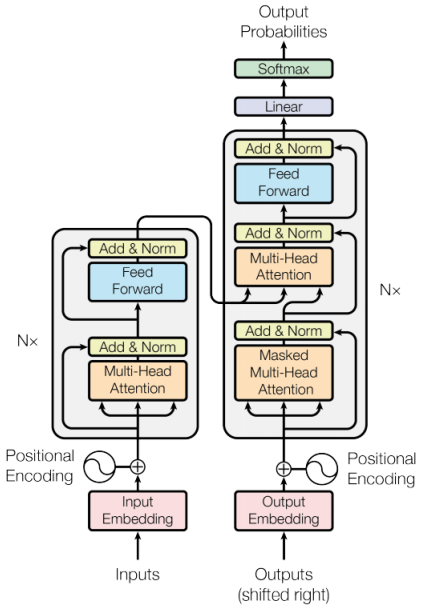

Figure 2.8: Transformer Architecture built by (Vaswani et al., 2017)

sformerArchitecture⟩

encoder that maps input sequences to a continuous representation. The ar-

chitecture can be seen in Fig. 2.8. This is then decoded into an output sequence of symbols one at a time. Each step is auto-regressive, ie. it consumes the previously generated symbols as additional input when creating the next. This is similar to an ensemble model. Stacks of 6 encoder and 6 decoder layers is used.
Each encoder layer has 2 sub-layers of a multi-head self-attention and the other a simple, position-wise fully connected feed-forward network layer. The output from each encoder layer can be represented as $LayerNorm(x + Sublayer(x))$, where $Sublayer(x)$ is the function implemented by the sub-layers in the model. Additionally, supporting residual connections is handled by producing outputs of dimension $d_{model} = 512$.

The decoder layer is similar to the encoder layer and has an additional 3rd sub-layer that performs multi-head attention over the output of the encoders. To ensure that the output layer isn't affected by the output of residual connections from the sub-layers, a normalization layer is implemented at the end. Masking the subsequent positions and offsetting the output by 1 position ensures that the predictions for position i can only depend on outputs from previous positions.

One of the key features in the Transformer architectures is *self-attention*. Let us look at this in detail below.

2.2.1 Self Attention

?⟨231⟩? The attention mechanism can be described as mapping a query to a set of key-value pairs. This can be seen from Fig. 2.9.
The evaluations performed on the Wall Street Journal dataset(Marcus et al., 1993), using 40k sentences, showed that even without task-specific tuning the model had better results with a fraction of the training cost.

2.2.2 Improvements of Transformer Architectures

?⟨232⟩? **BERT**

?⟨2321⟩? The paper on BERT, which is *Bidirectional Encoder Representations from Transformers* (Devlin et al., 2018), introduces a new language model. This model is truly fascinating in many ways. First and foremost it is designed to pre-train deep bidirectional representations using unlabelled data. This is done by jointly conditioning context in all layers to the right and left. This pre-training allows the model to be fine-tuned simply using one additional output layer. These features make this model conceptually simple and very powerful empirically.

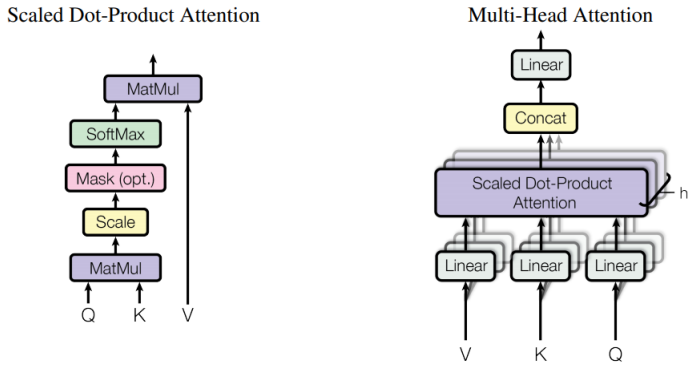

Figure 2.9: Scale Dot and Multi-Head Attention Models Vaswani et al. (2017)

⟨multiHeadAttention⟩

BERT employs language pre-training (Dai and Le, 2015) which has shown significant advantages in many applications e.g. paraphrasing, language level inference etc. These tasks aim to highlight the relationships between sentences through contextual understanding as well as by using tokenized outputs.

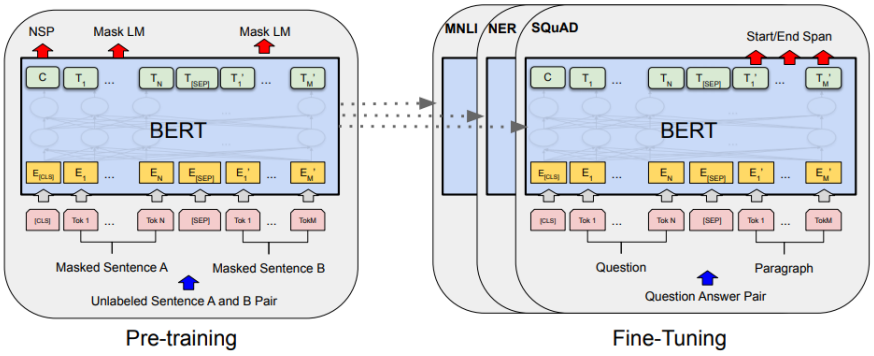

Figure 2.10: Pre-training and Fine Tuning procedures for BERT (Devlin et al., 2018)

?⟨bertPretraining⟩?

BERT was tested on The General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2018) which has a large number of diverse NLU tasks. BERT performed extremely well on the 11 NLP tasks that the authors ran it against. it showed an average accuracy improvement of 4.5% and 7% when compared to the previous state of the art models. Results of

BERT and its significant gains make it one of the best candidate models for NLU tasks.

ALBERT

?⟨2322⟩? Lan et al. (2019)

ROBERTA

?⟨2323⟩? Liu et al. (2019)

DISTILBERT

?⟨2324⟩? Sanh et al. (2019)

2.3 SQuAD Dataset

⟨24⟩ ## 2.4 Comparison of Techniques

?⟨25⟩? ## 2.5 Summary

?⟨26⟩? (Corsair, 2021)

Chapter 3

Research Methodology

⟨c3researchmeth⟩

3.1 Data Selection

?⟨c31⟩? mention datasets like Feng et al. (2015), We have selected the Stanford Question Answering Dataset (SQuAD). This is as a reading comprehension dataset based on Wikipedia articles. It is based on questions posed by crowdworkers on a set of articles. The answer to every question is a segment of text or span, from the corresponding reading passage, or the question might be unanswerable (Rajpurkar et al., 2018).

The dataset consists of over 150,000 questions. Split into 100,000 answerable and 50,000+ unanswerable question, which were written to look similar to unanswerable questions. The challenge being that a model should be able to correctly answer the answerable questions and abstain from answering the unanswerable ones. The dataset is freely available as a part of the Transformers package in python or it can be downloaded from the SQuAD 2.0 website (Rajpurkar, 2021).

To effectively use this dataset for our purposes, let us first take a look at what its contents look like below.

Context: *"The Normans (Norman: Nourmands; French: Normands; Latin: Normanni) were the people who in the 10th and 11th centuries gave their name to Normandy, a region in France. They were descended from Norse ("Norman" comes from "Norseman") raiders and pirates from Denmark, Iceland and Norway who, under their leader Rollo, agreed to swear fealty to King Charles III of West Francia."*
Question: *Who was the Norse leader?*
Answer: *Rollo*

The answer to the aforementioned question is quite simple for humans to comprehend. The challenge is for us to contextualize this and make it machine-understandable so that our model can answer it correctly.

The dataset consists of various kinds of English language examples like negation, antonyms, entity swaps, impossible conditions to answer, answerable, etc. making the dataset a well-balanced one.

To use this dataset correctly we shall perform the following pre-processing steps on it:

1. Data splitting into separate Question, Answer and Context lists.
2. Splitting the data into separate training and validation sets of question and answers using the 80/20 rule, also known as the Pareto principle. We will have 80% training data and 20% test data.
3. Tokenization of the split data to generate "context-question" pairs
4. Generating indexes for when an answer begins and ends in the dataset
5. Adding answer tokens based on their encoded positions

The SQuAD 2.0 Dataset (Rajpurkar et al., 2018), was developed with funding from Facebook to help address some major issues with existing datasets. Most datasets focus on questions that can be easily answered or use of automatically generated, unanswerable questions which are easily identifiable.
The SQuAD 2.0 dataset resolves this by combining the SQuAD dataset along with 50,000 crowd worker generated unanswerable questions. The key feature of these being that the unanswerable questions must look similar to answerable ones. For a model to be successful on this new dataset, it must be able to answer all possibly answerable questions as well as determine when no answers are provided for a question in the given paragraph and abstain from answering. A comparative study was done for a Natural Language Understanding(NLU) task that obtained an 86% score on SQuAD 1.1, only got 66% on the new 2.0 dataset. The dataset helps bridge the gap between true NLU and machine understanding by using the concept of Relevance. Through comparisons with various datasets such as RACE, MCTest, QASENT etc. they have identified the missing links like negative examples, antonyms and helped fill the gap. This dataset forces the models to understand whether a paragraph span has the answer to the question posed.

	SQuAD 1.1	SQuAD 2.0
Train		
Total Examples	87,599	130,319
Negative Examples	0	43,498
Total articles	442	442
Articles with negatives	0	285
Development		
Total Examples	10,570	11,873
Negative Examples	0	5,945
Total articles	48	35
Articles with negatives	0	35
Test		
Total Examples	9,533	8,862
Negative Examples	0	4,332
Total articles	46	28
Articles with negatives	0	28

Table 3.1: Dataset statistics of SQuAD 2.0, compared to the previous SQuAD 1.1(Rajpurkar et al., 2018).

datasetDescription⟩?

3.2 Data Pre-processing And Transformation

?⟨c32⟩? Techniques used. (1) Average pooling; (2) max pooling; (3) the concatenation of the last vectors on both directions. . Dropout operation is performed on the QA representations before cosine similarity matching. Use GlOVe embeddings and suffix-prefix encoding in both directions of self attention pass to show improvements.

3.3 Existing Models And Benchmarks

?⟨c33⟩?

Appendices

Chapter A
Gantt Chart

⟨cgc⟩

?\ganttChartMid\?

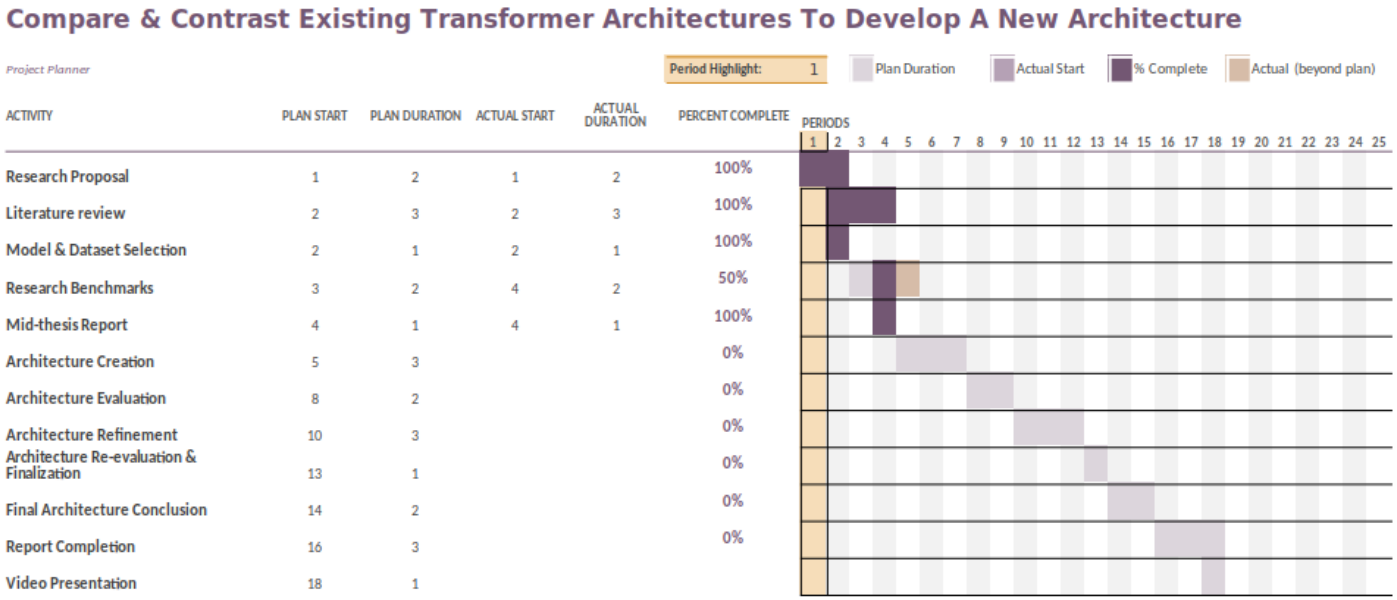

Figure A.1: Mid-Thesis report Gantt Chart

References

lstmoriginal — Jürgen Schmidhuber and Sepp Hochreiter. Long short-term memory. *Neural Comput*, 9(8):1735–1780, 1997.

lstmBiLSTM — A. Graves and J. Schmidhuber. Framewise phoneme classification with bidirectional lstm networks. 4:2047–2052 vol. 4, 2005. doi: 10.1109/IJCNN.2005.1556215.

ighextractive — Ming Tan, Bing Xiang, and Bowen Zhou. Lstm-based deep learning models for non-factoid answer selection. *CoRR*, abs/1511.04108, 2015. URL `http://arxiv.org/abs/1511.04108`.

atayl — Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

bert — Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

stmInsuranceQA — Minwei Feng, Bing Xiang, Michael R. Glass, Lidan Wang, and Bowen Zhou. Applying deep learning to answer selection: A study and an open task. *CoRR*, abs/1508.01585, 2015. URL `http://arxiv.org/abs/1508.01585`.

dataset — Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad. *CoRR*, abs/1806.03822, 2018. URL `http://arxiv.org/abs/1806.03822`.

en1961baseball — Bert F Green Jr, Alice K Wolf, Carol Chomsky, and Kenneth Laughery. Baseball: an automatic question-answerer. In *Papers presented at the May 9-11, 1961, western joint IRE-AIEE-ACM computer conference*, pages 219–224, 1961.

lunar — William A Woods and WOODS WA. Lunar rocks in natural english: Explorations in natural language question answering. 1977.

hfTransformers The Hugging Face Team. Transformers. URL https://huggingface.co/transformers/. Accessed: 2021-04-16.

squad P Rajpurkar. Squad2.0, 2021. URL https://rajpurkar.github.io/SQuAD-explorer/. Accessed: 2021-04-16.

lstmRing Mark B Ring. Learning sequential tasks by incrementally adding higher orders. *Advances in neural information processing systems*, pages 115–115, 1993.

bilstmherman Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. *CoRR*, abs/1506.03340, 2015. URL http://arxiv.org/abs/1506.03340.

lstmSubilstm Siddhartha Brahma. Suffix bidirectional long short-term memory. *CoRR*, abs/1805.07340, 2018. URL http://arxiv.org/abs/1805.07340.

dataset1 Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. *CoRR*, abs/1606.05250, 2016. URL http://arxiv.org/abs/1606.05250.

wu2016question Shuohang Wang and Jing Jiang. Machine comprehension using match-lstm and answer pointer. *CoRR*, abs/1608.07905, 2016. URL http://arxiv.org/abs/1608.07905.

lstmMatch Shuohang Wang and Jing Jiang. Learning natural language inference with LSTM. *CoRR*, abs/1512.08849, 2015. URL http://arxiv.org/abs/1512.08849.

lstmPointer Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. *arXiv preprint arXiv:1506.03134*, 2015.

recurrent Jie Zhou, Ying Cao, Xuguang Wang, Peng Li, and Wei Xu. Deep recurrent models with fast-forward connections for neural machine translation. *Transactions of the Association for Computational Linguistics*, 4:371–383, 2016.

wsj Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. 1993.

dai Andrew M Dai and Quoc V Le. Semi-supervised sequence learning. *arXiv preprint arXiv:1511.01432*, 2015.

wang Wei Wang, Ming Yan, and Chen Wu. Multi-granularity hierarchical attention fusion networks for reading comprehension and question answering. *arXiv preprint arXiv:1811.11934*, 2018.

albert Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

roberta Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

distil Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108, 2019. URL `http://arxiv.org/abs/1910.01108`.

RAM Corsair. Vengeance® lpx 8gb (1 x 8gb) ddr4 dram 2400mhz c14 memory kit - black, 2021. URL `https://www.corsair.com/uk/en/Categories/Products/Memory/VENGEANCE-LPX/p/CMK8GX4M1A2400C14`. Accessed: 2021-04-16.