Develop New Transformer Architecture For  
Question and Answering

Nirbhay P. Tandon

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.

# Introduction

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 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.

We wish to change that. Through our work, we would like to aim at creating a sustainable, fast, easy to understand Transformer architecture for Question Answering. An advancement on the work done by the team at Google . 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.

Through our research proposal we wish to highlight why we are going to be performing this research and putting in the effort to devise a new architecture. We have divided our research proposal into 7 sections. In Section [[introduction]](#introduction), we shall take a look at briefly introducing the concept and why our work is necessary. Next, in Section [[backRR]](#backRR), we outline the background work that has already been done in this field and how some of the papers relate to the work that has been done. We use this as an opportunity to highlight some of the shortcomings in current architectures and modelling techniques. In Section [[aims]](#aims), we briefly outline the aim of our proposed research. In Section [[researchMeth]](#researchMeth), we define in some detail the work that we will do to establish our research and how we plan to quantify the work that shall be done. Section [[expectedoutcomes]](#expectedoutcomes), highlights our goal, which is to produce a new transformer architecture that performs better at Question Answering based tasks. In Section [[resources]](#resources), we have outlined the minimum hardware requirements along with the resources available to the author that will be used to conduct this research. Finally, in Section [[plan]](#plan), we submit a Gantt Chart to outline our plan against the number of weeks.

## Background Of The Study

In this section, we shall highlight what has led us this far and some of the interesting challenges that it poses. In [1.1.1](#back), we briefly look at the history of Natural Language Processing and how some of the challenges were addressed. In [1.1.2](#rr), 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.

### Background

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 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 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.

### Related Research

Outlined below are some of the most important pieces of research that relate directly to the work done for Transformers and Q&A based systems.

## Aims And Objectives

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 , PyTorch and Tensorflow on the dataset
2. Obtain F1, validation, 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.

## Scope Of The Study

## Significance Of The Study

## Structure Of The Study

# Literature Review

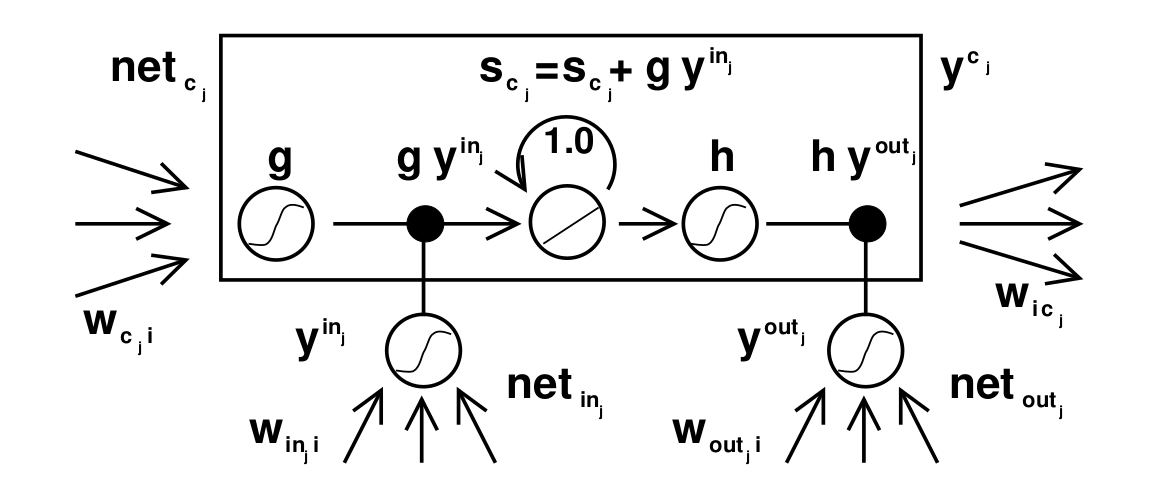
To be able to effectively deliver on the aims and objectives we have defined in Section [1.2](#Xb52009b64fd0a2a49e6d8a939753077792b0554), we must first critically analyse the work that has been done in the field of NLP and, in particular, the field of Question-Answering based networks. In this section, we will critically analyse a few key model architectures and how we reached Transformers, which form the base of our proposed architecture.

## Question Answering Using LSTMs

In 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, , 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 , the network can be really fast but also suffers from memory constraints and lack of generalization. This happens because to bridge time lags, the model adds 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](#lstmCEC).



LSTM Memory cell with a Constant Error Carousel having fixed weight 1.0,

The central feature for 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](#lstmCEC), Schmidhuber and Hochreiter proposed the following set of equations [[eq1]](#eq1) - [[eq3]](#eq3). More details on this can be found by referring to the appendix section A.1 in .

where

and

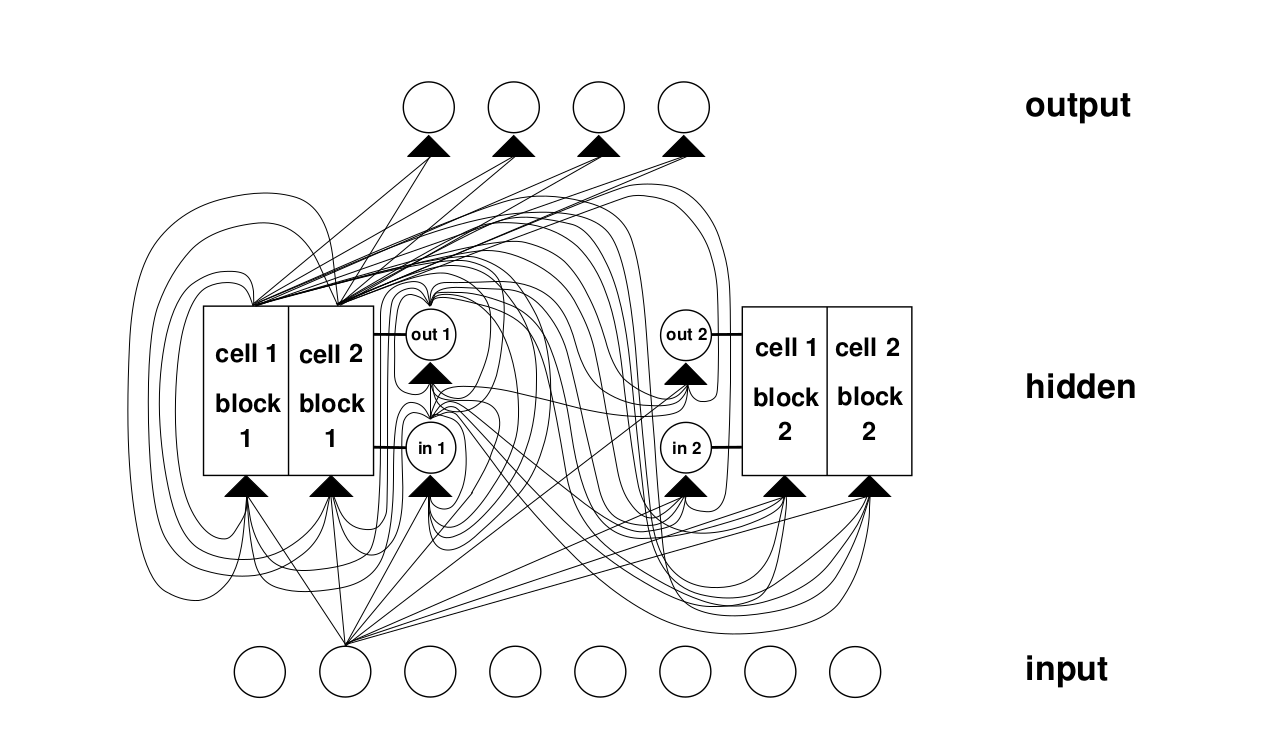
A generalised form of this equation can be represented as:

At time , ’s output is computed as:

where the “internal state” is:

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

In fig. [2.2](#lstmNetwork), we see the internal workings of how an LSTM network could look like.



LSTM network with 8 input cells, 4 output cells and 2 memory cells of block size 2. Here marks the input gate, marks the output gate and marks the first memory cell block 1. The internal architecture of is similar to fig. [2.1](#lstmCEC).

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. 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 QnA based networks

## Question Answering Using Transformers

Transformer based models have been become a very reliable way to implement various NLP based tasks since their introduction in the late 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 and 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 , which is a function of the previous hidden state and the input function . This inherently sequential nature of RNNs causes memory issues, leading to reduced batch sizes.

Transformer Architecture built by

Transformer Architecture built by

The architecture for a *Transformer* in this paper is outlined as having an encoder that maps input sequences to a continuous representation. The architecture can be seen in Fig. [2.3](#transformerArchitecture). This is then decoded into an output sequence of symbols one at a time. Each step is auto-regressive, i.e. 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 , where is the function implemented by the sub-layers in the model. Additionally, supporting residual connections is handled by producing outputs of dimension .

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 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.

### Self Attention

The attention mechanism can be described as mapping a query to a set of key-value pairs. This can be seen from Fig. [2.4](#multiHeadAttention).  
The evaluations performed on the Wall Street Journal dataset, using 40k sentences, showed that even without task-specific tuning the model had better results with a fraction of the training cost.

Scale Dot and Multi-Head Attention Models

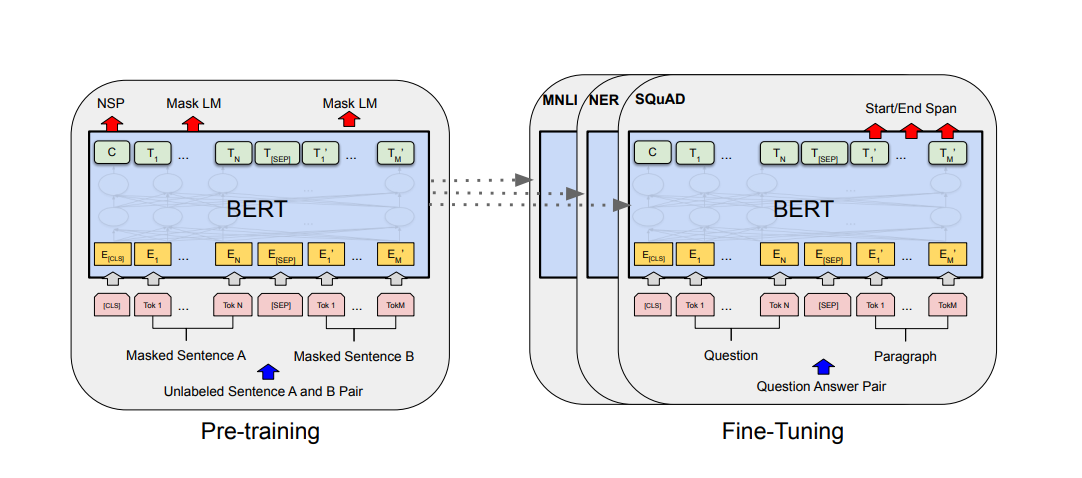
Scale Dot and Multi-Head Attention Models

### Improvements of Transformer Architectures

#### BERT

The paper on BERT, which is *Bidirectional Encoder Representations from Transformers* , 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.

BERT employs language pre-training 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.



Pre-training and Fine Tuning procedures for BERT

BERT was tested on The General Language Understanding Evaluation (GLUE) benchmark 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

#### ROBERTA

#### DISTILBERT

## Comparison Of Techniques

## Summary

# Research Methodology

**DONT DO ANY ANALYSIS HERE**

## Data Selection

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 crowd-workers 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 .

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 .

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

## Data Pre-processing And Transformation

## Existing Models And Benchmarks

# Architecture Creation

## Drawbacks Of Current Architectures

## Proposed Architecture Improvements

## Architecture Refinement