DEEP LEARNING IN NLP - CONSTRUCTING A MACHINE QUESTION ANSWERING MODEL

Sam Cheung Hiu Fan[*]
Department of Statistics and Actuarial Science
The University of Hong Kong
chfsam@hku.hk

ABSTRACT

Machine Question Answering (QA), a machine's ability to comprehend textual information and respond to either relevant or irrelevant questions, remains one of the most challenging yet compelling tasks in the Natural Language Processing (NLP) domain. Its difficulty centers not only in deep natural language understanding but also accurate logical reasoning. This research study exploits some of the state-of-the-art language models and deep learning models, as well as the popular Stanford Question Answering Dataset (SQuAD) to train a QA model. The best model, which is an ensemble of Bidirectional Encoder Representations from Transformers (BERT) with additional gating mechanism and residual learning blocks in the classification layer, yields the highest F1 score of 82.294 along with the Exact Match score of 79.542 in the SQuAD Development dataset.

***K*eywords** Question Answering · Natural Language Processing · GloVe · ELMo · BERT · BiDAF · Neural Networks · Gating · Residual Learning

1 Introduction

Natural Language Processing (NLP) is prevailing in recent deep learning research, fuelled by the huge potentials of applying conversational machines to solve problems in different industry verticals, as well as the advancement of machine computational capabilities and efforts across the academia and the industry. Typical applications of NLP include sentiment analysis, named entity recognition (NER), machine translation, etc. Machine Question Answering (QA), a machine's ability to comprehend textual information and properly respond to either a relevant or irrelevant question, remains one of the most challenging tasks in the NLP domain. Its difficulty centers in not only deep natural language understanding but also accurate logical reasoning. A promising and intelligent QA model would mean a great leap forward in machine-human interaction.

Riding on various state-of-the-art word representation models and deep learning models, this research study aims at constructing a Machine QA model to comprehend textual information and answer questions that are either answerable or unanswerable. Possible applications of the model include a Q&A tool in documents, chat bot for answering customers' enquiry, etc.

2 Data

The version 2 of the Stanford Question Answering Dataset (SQuAD) is used to train QA models throughout the study. It contains about 150,000 question-answer pairs [1] created by crowd workers from over 400 English Wikipedia articles. For each question, the answer is restricted to be either a segment of texts in the given paragraph or no answer. Two examples for answerable and unanswerable questions are given below respectively.

[*]Acknowledgement to supervisor Dr. Gilbert Lui

- **Answerable**

  Context: "Spanish mendicants in the sixteenth century taught indigenous scribes in their communities to write their languages in Latin letters and there is a large number of local-level documents in Nahuatl, Zapotec, Mixtec, and Yucatec Maya from the colonial era, many of which were part of lawsuits and other legal matters . Although Spaniards initially taught indigenous scribes alphabetic writing, the tradition became self-perpetuating at the local level. The Spanish crown gathered such documentation and contemporary Spanish translations were made for legal cases. Scholars have translated and analyzed these documents in what is called the New Philology to write histories of indigenous peoples from indigenous viewpoints."[2]

  Question: "What were a large number of the local documents in regards to?"

  Answer: "lawsuits and other legal matters"

- **Unanswerable**

  Context: "One challenge to the traditional concept of matter as tangible "stuff" came with the rise of field physics in the 19th century. Relativity shows that matter and energy (including the spatially distributed energy of fields) are interchangeable. This enables the ontological view that energy is prima materia and matter is one of its forms. On the other hand, the Standard Model of Particle physics uses quantum field theory to describe all interactions. On this view it could be said that fields are prima materia and the energy is a property of the field." [3]

  Question: "When did the field of physics stop being taught?"

  Answer: Nil

The dataset is partitioned into 3 subsets, including the train, development and test datasets. The organizer of SQuAD keeps the testing dataset from the participants with a view to preserve the integrity of the test results. Participants have to submit the code and trained model to the organizer and let them run the model on the testing dataset. The model evaluation process following the submission takes approximately 1-2 weeks time. Owing to considerable evaluation time for the testing dataset, in this study the performance of the models are evaluated on the *Development* dataset only. Table 1 tabulated the summary statistics of the SQuAD.

Table 1: Statistics of SQuAD

	Train	Development	Test
Total examples	130319	11873	8862
Negative examples*	43498	5945	4432
Total articles	442	35	28
Articles with negatives*	0	35	28
Range of number of context tokens	[23, 408]	[27, 448]	-
Mean number of context tokens	116	122	-
Percentage of examples with number of tokens > 300	0.9%	3.5%	-
Range of number of question tokens	[4, 28]	[4, 17]	-
Mean number of question tokens	10	10	-

Note
1. "Negative examples" stands for the number of questions that are unanswerable while "Articles with negatives" refers to the number of articles which all questions are unanswerable.
2. The statistics related to number of tokens in the test dataset are not available.

3 Methodology

The QA Models proposed in this research study comprise of two key components, including a *word representation model* and a predictive model which is indeed a *neural network*. The word representation model converts each word token in both the context paragraph and the question into fixed-length numeric vectors. The vectors are then fed into the neural network to predict the starting and ending positions in the context paragraph. The span of text bounded by the two positions is the predicted answer to the question. If the neural network's predicted starting and ending positions do not surpass certain threshold (the *no answer threshold*), the model will conclude that the question has no answer. Further details of the no answer threshold will be articulated in later sections. Figure 1 presents the high level view of

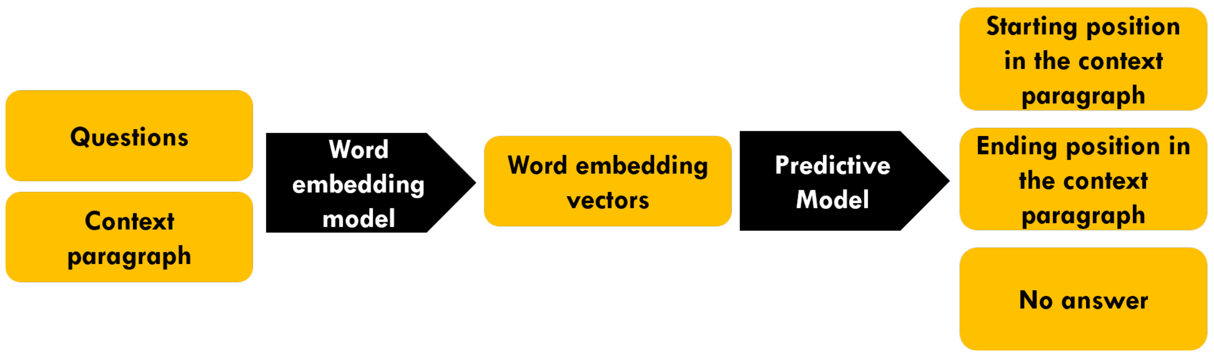

Figure 1: Flow diagram of QA model

the QA model. Various combinations of word representation models and neural network architectures are tested in order to find the best performing model.[2]

3.1 Word Representation Model

The class of word representation model adopted in this study is the word embedding, which is a collective name for models that transform words tokens into fixed-length numeric vectors. Compared to the simplest and traditional way to represent words in numeric forms (i.e. the *one hot vectors* representation), word embedding outperforms by efficient numeric representation of words and the capability in capturing semantic relationships amongst words, by giving similar vector representation to semantically similar words. This is achieved through the use of a relatively lower dimensional vector to represent a word.

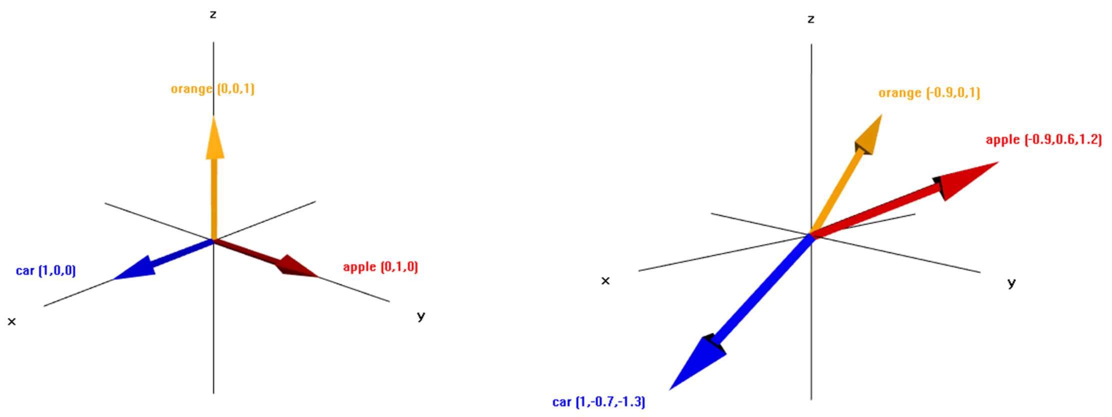

Figure 2: Visual representation of one-hot (left) and word embedding (right) vectors

Under one hot vectors representation (i.e. using 0 or 1 to represent the absence and presence of a word in a vector respectively), with the entire text dataset containing 100000 unique vocabularies, a 100000-dimensional vector is required to represent each vocabulary. Processing of such vectors is memory consuming and computationally expensive. In contrast, the dimension of word embedding vectors usually ranges from few hundreds and up to one or two thousand. Another merit of using word embedding vectors over one hot vectors is that they can encode semantic relationship amongst words. Figure 2 respectively plot the one hot vectors (left) and word embedding (right) vectors of the words "apple", "orange" and "car" in a 3-dimensional vector space. The angles between each pair of one hot vectors are 90° and one cannot tell which pairs of word are more similar or dissimilar to each other. Meanwhile on the right of Figure 2, the inner angle between word embedding vectors for fruit-related words "orange" and "apple" is smaller than that of "orange" and "car". Such nice property enables the neural network models easily differentiate words of different meaning. In this study, 3 different word embedding models, namely *Global Vectors (GloVe)*[4] , *Embeddings from Language Models (ELMo)*[5] and *Bidirectional Encoder Representations from Transformers (BERT)* [6] are employed to represent text before predictive modeling.

[2]The metrics for evaluating the model performance will be further elaborated in Section 4.1.

3.1.1 Global Vectors (GloVe)

Global Vectors (GloVe) for word representation is an unsupervised method to train fixed length vectors for word tokens. The model is trained based on the non-zero entries of the word-word co-occurrence matrix obtained from the training text corpus. The essence of the model is that the ratios of word-word co-occurrence probabilities should have encoded some form of semantics amongst the word tokens.

Assume there are N unique words in the text corpus and denote the matrix of word-word co-occurrence counts as **X**, and let X_{ij} be the number of times that the word w_j occurs in the context of the target word w_i. Note that a word w_i can either be a target word or an neighboring word for other target words, the GloVe model trains 2 word vectors for the same word w_i, i.e. v_i and $\tilde{v_i}$ respectively. The algorithm eventually boils down to a weighted least square regression problem to find a set of **v** and $\tilde{\mathbf{v}}$ that minimize the following loss function J.

$$J = \sum_{i=1}^{N} \sum_{j=1}^{N} f(X_{ij})(v_i \tilde{v_j} + b_i + \tilde{b_j} - \log X_{ij})^2, \text{where } X_{ij} \neq 0. \tag{1}$$

In the above formula, b_i and $\tilde{b_j}$ are the bias term associated to the words i and j respectively. $f(X_{ij})$ is the weight assigned to X_{ij} and the function f is parametrized as:

$$f(x) = \begin{cases} \left(\frac{x}{x_{\text{max}}}\right)^{\alpha} & \text{if } x < x_{\text{max}}, \\ 1 & \text{otherwise .} \end{cases} \tag{2}$$

The main objective of the weighting function is to prevent some frequently co-occurred word pairs, such as "this is", "and the", etc. to be over-weighed in the loss function. In the original paper written by Pennington et al.(2014) [4], they set $x_{\text{max}} = 100$ and $\alpha = 3/4$ and obtained satisfactory results.

In this research study, the pre-trained GloVe vectors from Wikipedia 2014 + Gigaword 5 are used. These GloVe vectors was trained on 6 billion tokens covering 400000 English vocabulary and the dimension for each word vector is 300. Assuming that each context paragraph is fixed with K word tokens while each question is fixed with V word tokens, after mapping there would be K 300-dimensional vectors representing the K context paragraph word tokens and V 300-dimensional vectors representing the V question tokens. They are the inputs to predictive modeling in Section 3.2.

One short-coming of GloVe, however, is that it cannot handle polysemy, i.e. the coexistence of multiple meanings for the same word under different context. For example, the word *jam* can come up with very different meanings under context like *strawberry jam* and *traffic jam*, yet the GloVe model gives the same vector to *jam* in both phrases regardless of its neighbouring words. In light of GloVe being a *context-free* embeddings, *contextualized embeddings* including **Embeddings from Language Models (ELMo)** and **Bidirectional Encoder Representations from Transformers (BERT)** are also considered in this project. Both of them output embedding vectors by conditioning on the contexts to the left and right of the target word. In addition, GloVe is a word based language representation model which words unseen in the training text corpus, i.e. *out-of-vocabulary tokens* would not have word embedding vectors trained. In case a rare or domain specific term is the most crucial word in either the context paragraph or the question, the use of GloVe in QA would have left out such important information.

3.1.2 Embeddings from Language Models (ELMo)

Embeddings from Language Models (ELMo) is a contextualized embedding model that represents each word token in vector form by taking the entire input context into consideration. The ELMo vectors obtained are functions of the intermediate states of a deep bidirectional language model (biLM). One of the differentiating features of ELMo when compared to GolVe is that it is a character-based language model and is capable of outputting proper representation vectors for out-of-vocabulary tokens based on the morphological clues.

Being a biLM, ELMo models the sequence of N word tokens $w_1, w_2, ..., w_N$ in both the forward and backward directions. A forward language model (LM) is trained to maximize the probability of the sequence, which is equivalent to maximize the product of the conditional probabilities of the word token w_j given the preceding word tokens $w_1, w_2, ..., w_{j-1}$.

$$\Pr\big(w_1, w_2, ..., w_N\big) = \prod_{j=1}^{N} \Pr\big(w_j | w_1, w_2, ..., w_{j-1}\big). \tag{3}$$

Likewise, a backward LM is trained to maximize the product of the conditional probabilities of the word token w_j given the future word tokens $w_{j+1}, w_{j+2}, ..., w_N$.

$$\Pr\big(w_1, w_2, ..., w_N\big) = \prod_{j=1}^{N} \Pr\big(w_j | w_{j+1}, w_{J+2}, ..., w_N\big). \tag{4}$$

The biLM in ELMo is essentially a L-layers of bidirectional Long Short Term Memory (biLSTM) [7] network. Each word token w_j is first represented as a context insensitive vector x_j after going through the character n-gram convolutional filters, highway layers [8] and a linear projection. Then, in the forward direction, the initial representation vectors $x_1, x_2, ..., x_N$ are fed into the LSTM layers to obtain a sequence of context-dependent hidden states $\overrightarrow{\mathbf{h}}_{1,k}, \overrightarrow{\mathbf{h}}_{2,k}, ..., \overrightarrow{\mathbf{h}}_{N,k}$, where $k = 1, ..., L$. The LSTM output for the word token w_j at the top layer, $\overrightarrow{\mathbf{h}}_{j,L}$, is used to predict the next word token w_{j+1} via a softmax layer. Similarly, the sequence of initial representation vectors **x** is passed into the backward LSTM and transformed to $\overleftarrow{\mathbf{h}}_{1,k}, \overleftarrow{\mathbf{h}}_{2,k}, ..., \overleftarrow{\mathbf{h}}_{N,k}$, where $k = 1, ..., L$. The graphical representation of the biLSTM is shown in Figure 3 (only displaying one biLSTM layer for simplicity).

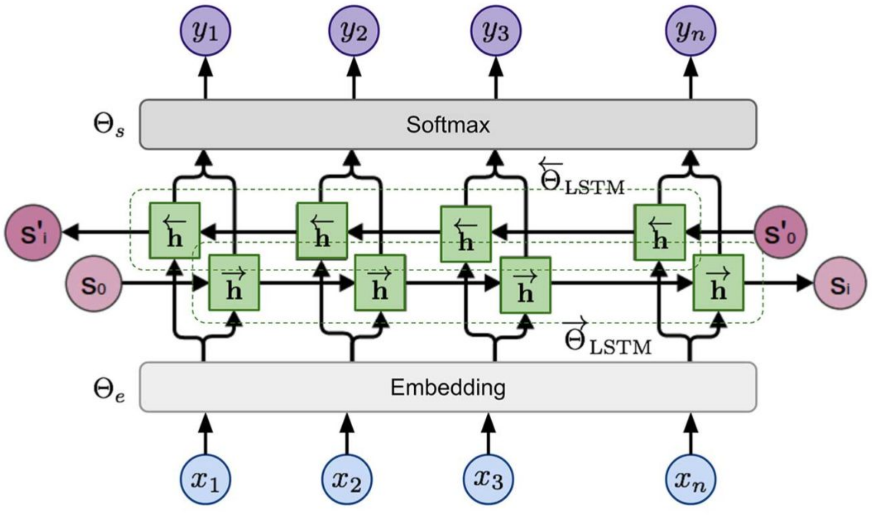

Figure 3: Model architecture of biLSTM (Source: Lilian Weng, 2019) [9]

Speaking of model optimization, the objective function of the biLM ℓ is to maximize the sum of log-likelihood of the word tokens predicted from the forward and backward language model respectively, mathematically,

$$\ell = \sum_{j=1}^{N} \Big[\log\Pr\big(w_j | w_1, ..., w_{j-1}; \Theta_e, \overrightarrow{\Theta}_{LSTM}, \Theta_s\big) + \log\Pr\big(w_j | w_{j+1}, ..., w_N; \Theta_e, \overleftarrow{\Theta}_{LSTM}, \Theta_s\big) \Big] \tag{5}$$

where the parameters for the word token representation Θ_e and softmax layer Θ_s are shared between the forward and backward LM, while the parameters in the forward LSTM $\overrightarrow{\Theta}_{LSTM}$ and backward LSTM $\overleftarrow{\Theta}_{LSTM}$ are independent to each other.

As mentioned above, ELMo is a combination of the interim layer outputs in the biLM. For each word token w_j, a L layer biLM computes a set of $2L + 1$ representations:

$$R_j = \Big\{ \mathbf{x}_j, \overrightarrow{\mathbf{h}}_{j,k}, \overleftarrow{\mathbf{h}}_{j,k} | k = 1, ..., L \Big\} \tag{6}$$
$$= \mathbf{h}_{j,k} | k = 0, ..., L, \tag{7}$$

where $\mathbf{h}_{j,0}$ equals to x_j and $\mathbf{h}_{j,k} = \Big[\overrightarrow{\mathbf{h}}_{j,k}, \overleftarrow{\mathbf{h}}_{j,k} \Big]$ for $1 \leq k \leq N$ (the symbol "," refers to concatenation of 2 vectors) for each biLSTM layer. To adopt ELMo in downstream NLP tasks, a set of task specific weightings for all biLM layers is trained together with the specific task and the final representation for word token w_j is a linear combination of the weightings and the $L + 1$ biLM representations.

$$\mathbf{ELMO}_j = \gamma^{task} \sum_{k=0}^{L} s_j^{task} \mathbf{h}_{j,k}, \tag{8}$$

where $\mathbf{s}^{task}$ are softmax-normalized weights and the scalar parameter γ^{task} is used by the task model to scale the entire ELMo vector.

The "small" pre-trained ELMo model is used in this research study and the dimension of each ELMo word vector is 1024. The model is pre-trained on the 1 billion Word Benchmark [10] while the number of parameters is 13.6 million. In training the QA model, however, only 4 parameters in the small ELMo model are finetuned (the weights for the embedding layer and the ouput from 2 biLSTM layers as well as the scale parameter γ^{task}).

Assuming each context paragraph is fixed to have K word tokens V question tokens, the ELMo model outputs K 1024-dimensional representation vectors for the context tokens and V 1024-dimensional vectors for the questions tokens for subsequent predictive modeling.

3.1.3 Bidirectional Encoder Representations from Transformers (BERT)

Bidirectional Encoder Representations from Transformers (BERT) is a recent state-of-the-art language model which significantly outperforms peer word representation models in a number of prevalent and well-recognized NLP tasks, for example the Multi-Genre Natural Language Inference (**MNLI**) [11], The Semantic Textual Similarity Benchmark (**STS-B**) [12], etc. Similar to ELMo, BERT is also a contextualized embedding model and it is bidirectional, yet there are two major differences between the 2 models: BERT is more "bidirectional" than ELMo (to be explained in subsequent paragraphs) and the attention mechanism is adopted in BERT but not in ELMo.

The back-bone of BERT is Transformer[13], which harnesses the so called "multihead self-attention mechanism" and residual learning [14] blocks to create deep understanding of word tokens. Different from the popular Recurrent Neural Network (RNN) based language models, Transformers can be easily parallelized, enabling it to be trained more efficiently. Owing to the extensive details about Transformer's model architecture, please refer to Appendix for further reading.

BERT accepts either a single sentence or a pair of sentences (e.g. packing a context paragraph and the question into one sentence) in one token sequence as an input. For a given token, its input representation is constructed by summing 3 embeddings, including token, segment and position embeddings. Figure 4 is a visual representation of the BERT's inputs.

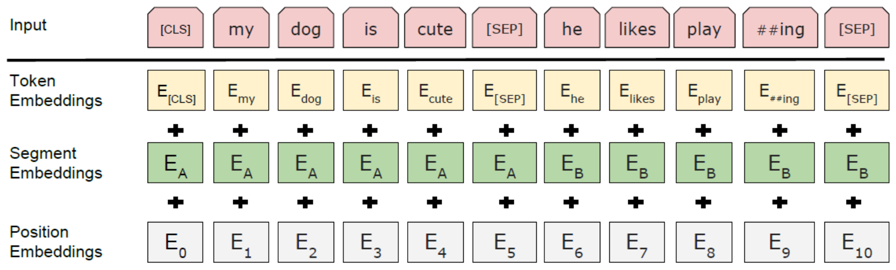
Input
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} E_{dog} E_{is} E_{cute} $E_{[SEP]}$ E_{he} E_{likes} E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Segment Embeddings
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Position Embeddings
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

Figure 4: Input representation of BERT (Source: Devlin, 2018) [6]. It is the sum of token, segment and position embeddings

The BERT model is also characterized with the following features:

- The token embedding is essentially WordPiece embeddings [15] with a 30000 token vocabulary. The split word pieces are denoted as ##. For instance the word "largely" is split as "large" and "##ly". Under this sub-word tokenization, the problem out-of-vocabulary token can still be properly addressed.
- The position embeddings $E_0, E_1, \ldots$ are included to capture the positional relationship amongst the word pieces. The maximum sequence length that the position embeddings can handle is 512.

- The first token of each word pieces sequence has to be the [CLS] label (CLS is abbreviated for classification). The final hidden state (i.e. output of Transformer) of the [CLS] token is regarded as representation of the entire sequence in classification tasks.
- As mentioned above, two sentences (context paragraph and question) can be concatenated as one sequence input in BERT. To be distinguished by the BERT model, they are separated by another special token [SEP]. Meanwhile, every token of the first sentence will have a learned segment A embedding E_A as input while each token of the second sentence have a segment B embedding E_B as input.

The BERT model is trained by two novel prediction tasks, namely *Masked Language Model* (MLM) and *Next Sentence Prediction*. Recall that the biLM structure in ELMo composes of a stack of left-to-right LSTM layers and a stack of right-to-left LSTM layers, yet in reality a word token's meaning should be conditioned on both previous word tokens and future word tokens simultaneously instead of independently comprehending the forward and backward context. To address this issue, 15% of word piece tokens in the training corpus for BERT are randomly masked and the BERT model is trained to predict the missing word tokens. The second task in BERT model training is the *Next Sentence Prediction*, which trains the model to understand the relationship between sentence pairs. It is basically a binary classification task, given an input sentence A, the model has to predict whether sentence B is the next sentence to A. In addition, training corpus for BERT include the concatenation of BooksCorpus (800 million words) and English Wikipedia (2500 million words). 2 BERT models, namely BERT-Base Uncased and BERT-Large Uncased are used in this research study:

Table 2: Pre-trained BERT models used in QA model

Name	No. of layers in Transformer	Hidden size (dimension of output vector)	No. of self-attention heads (A)	Total no. of parameters
BERT-Base	12	768	12	110M
BERT-Large	24	1024	16	340M

Following the Google AI Research team's SQuAD model[16], the maximum sequence length of question word pieces is set at 64 and the maximum length of each sequence of question-context word pieces is fixed at 384. Note that the Transformer architecture in BERT already captures deep understanding about the semantics and interaction between the question and answer word pieces, hence after generating N BERT vectors representing the N word piece tokens, the QA model can be simply set up by appending an extra classification layer for training and performing BERT fine-tuning.

3.2 Predictive Model

After transforming the word tokens/pieces into GloVe/ELMo/BERT vectors, they are fed into the predictive model, which is a neural network, for predicting the starting and ending position of the answer phrase in the context paragraph. If the prediction probabilities do not exceed certain threshold, the question is considered to be unanswerable. In this research study two streams of predictive model are proposed. One is BiDirectional Attention Flow (BiDAF) [17], which can be applied to all 3 word embedding models. Meanwhile another model stream is BERT finetuning with modified classification layer, which is only applicable to BERT input.

3.2.1 BiDirectional Attention Flow (BiDAF)

BiDirectional Attention Flow (BIDAF) network is a multi-stage model architecture that can be adopted in Question Answering task. It utilizes the *attention* mechanism bidirectionally in order to obtain the question-aware representation of tokens in the context paragraph as well as the context-aware representation of the question tokens for subsequent modeling. The bidirectional attention mechanism enables the model to better learn the relationship and interaction between the context and question word tokens. Meanwhile, it is computationally less expensive than the conventional recurrent neural network (RNN) based model since it can be easily parallelized. The BiDAF model experimented in this study is slightly modified based on the original model[17] in order to increase efficiency during model training. For simplicity, the term BiDAF is equivalent to the modified BiDAF model hereinafter. The visual representation of the modified BiDAF is as shown in Figure 5.

- **Word Embedding Layer**
  This layer outputs the GloVe/ELMo/BERT vectors for the word tokens in both the context paragraph and question. For GloVe vectors, they are simply looked up from the pre-trained embedding matrix according to the word tokens. Meanwhile, ELMo or BERT vectors are generated on the fly when the word tokens/pieces

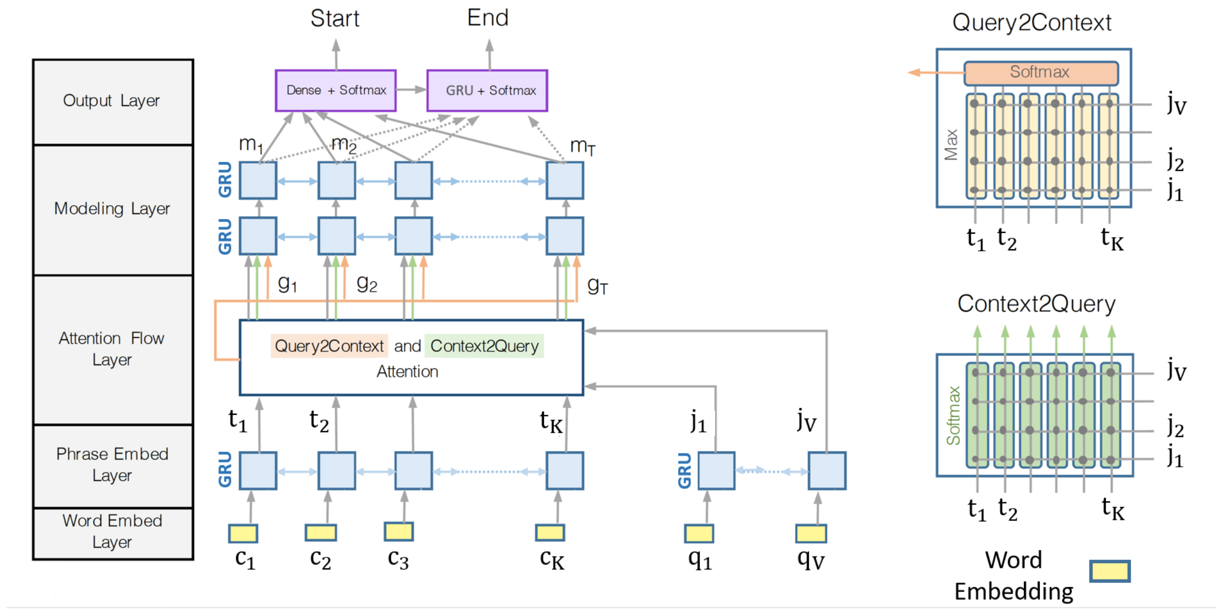

Figure 5: Visual representation of the modified BiDAF model (modified based on Seo et al., 2017) [17]

are passed into the pre-trained ELMo or BERT model[3], such that the vector representation of the word tokens can condition on the left and right context. Assuming the length of context token sequence be K, the length of question token sequence be V and the dimension of each word representation vector be d, then each context paragraph is represented by the matrix $\mathbf{C} \in \mathbb{R}^{d \times K}$ and each question is represented by the matrix $\mathbf{Q} \in \mathbb{R}^{d \times V}$. They are the input to the contextual embedding layer.

- **Contextual Embedding Layer** The contextual embedding layer is a bidirectional Gated Recurrent Unit (GRU) [18] instead of a biLSTM proposed in the original paper, given that it is computationally more efficient. This layer models the sequential relationship amongst the word tokens in both the context paragraph and the question. After that we obtain $\mathbf{T} \in \mathbb{R}^{2f \times K}$ and $\mathbf{J} \in \mathbb{R}^{2f \times V}$ respectively. The dimension of each contextualized context or question token vector is $2f$ as a result of the concatenation of f-dimensional hidden states from the forward and backward GRU.
- **Attention Flow Layer** The attention flow layer is used for associating and comprehending the relationship between the tokens in the context paragraph and question. Taking the contextual representations of the context paragraph $\mathbf{T}$ and the question $\mathbf{J}$ as input, the layer performs a bidirectional attention mechanism to obtain the question-aware vector representations of the word tokens in the context paragraph, $\mathbf{G}$.
  Firstly, a similarity matrix $\mathbf{S} \in \mathbb{R}^{K \times V}$ is derived and $\mathbf{S}_{kv}$ stands for the similarity between the k-th context word token and the v-th question word token. The computation of the similarity matrix is given by:

$$\mathbf{S}_{kv} = \mathbf{w}_{\mathbf{S}}^{\top} \left[\mathbf{T}_{:k}; \mathbf{J}_{:v}; \mathbf{T}_{:k} \odot \mathbf{J}_{:v} \right], \tag{9}$$

  where $\mathbf{w}_{\mathbf{S}}^{\top} \in \mathbb{R}^{6f}$ is a trainable weight vector, $\mathbf{T}_{:k}$ is the k-th column in matrix $\mathbf{T}$, $\mathbf{J}_{:v}$ is the v-th column in matrix $\mathbf{J}$, $\odot$ stands for the component-wise multiplication, and [A;B] means concatenating vector A and vector B horizontally (across the row). The similarity matrix $\mathbf{S}$ is then used to obtain the attentions and the attended vectors in both directions.
  - **Context-to-query Attention**
    Context-to-query (C2Q) attention is responsible for identifying question word tokens that are the most relevant to each context word token. Let $\mathbf{a}_k \in \mathbb{R}^V$ represents the attention weights on the question words by the k-th context word, and so $\sum \mathbf{a}_{kv} = 1$ for all k. The attention weight is computed by $\mathbf{a}_k = \text{softmax}(\mathbf{S}_{k:}) \in \mathbb{R}^V$. Then each attended question vector is $\tilde{\mathbf{J}}_{:k} = \sum_v \mathbf{a}_{kv} \mathbf{J}_{:v}$. Hence $\tilde{\mathbf{J}}$ is a matrix of dimension $2f \times K$ which consists of the attended question vectors for the entire context paragraph.

[3]To apply BiDAF with BERT, the word pieces of the context paragraph and question do not pack together as one sequence. Instead, the 2 sequences of word pieces are fed into BERT model separately as like ELMo does.

– **Query-to-context Attention**
On the other hand, query-to-context (Q2C) attention is responsible for identifying and assigning heavier weights to the context word that have the closest similarity to one of the question words, which is important to answer the question. The attention weights on the context words is obtained by $\mathbf{b} = \text{softmax}\big(\max_{col}(\mathbf{S})\big) \in \mathbb{R}^{K}$, where the function $\big(\max_{col}\big)$ is performed across the column. Then the attended context vector is $\tilde{\mathbf{t}} = \sum_k \mathbf{b}_k \mathbf{T}_{:k} \in \mathbb{R}^{2f}$. This vector can be view as the weighted sum of the most important context word tokens with respect to the question. By copying and stacking the vector $\tilde{\mathbf{t}}$ K times across the column, $\tilde{\mathbf{T}} \in \mathbb{R}^{2f\times K}$ is obtained.
In the final step, the contextual embeddings $\mathbf{T}$, the attention vectors $\tilde{\mathbf{T}}$ and $\tilde{\mathbf{J}}$ are combined to form $\mathbf{G}$, where each column vectors can be considered as the question-aware representation or each context word tokens. The k-th column vector of $\mathbf{G}$ is defined as:

$$\mathbf{G}_{:k} = \big[\mathbf{T}_{:k}, \mathbf{T}_{:k} \odot \tilde{\mathbf{J}}_{:k}, \mathbf{T}_{:k} \odot \tilde{\mathbf{T}}_{:k}\big] \in \mathbb{R}^{8f\times K}. \tag{10}$$

- **Modeling Layer** The modeling layer consists of two layers of bidirectional GRU, it takes $\mathbf{G}$ (which encodes the question-aware representations of context words) as the input and output the matrix $\mathbf{M} \in \mathbb{R}^{2f\times K}$. $\mathbf{M}$ is then relayed to the output layer for final prediction. Each column of $\mathbf{M}$ carries contextual information for each word tokens, i.e. each column vector have incorporated the important concepts in both the context paragraph, the question as well as their interactions to the word token itself.

- **Output Layer** The QA model is required to determine whether the question is answerable or not, followed by outputting a text segment in the context paragraph if it is answerable. The answer is generated by predicting the starting and ending positions of the answer in the context paragraph. Assume all context-question pairs in the training dataset are answerable, the probability of each position in the context paragraph being the starting position of the answer is computed by:

$$\mathbf{p}^s = \text{softmax}\big(\mathbf{w}_{(p^s)}^{\top}\big[\mathbf{G};\mathbf{M}\big]\big), \tag{11}$$

  where $\mathbf{w}_{(p^s)} \in \mathbb{R}^{10f}$ is a trainable vector weight.
  Due to the presence of unanswerable questions, an extra starting position is appended to the original K positions of the context paragraph. It is equivalent to concatenate a trainable scalar weight $w_s \in \mathbb{R}$ to the vector $\mathbf{w}_{(p^s)}^{\top}\big[\mathbf{G};\mathbf{M}\big] \in \mathbb{R}^{K}$ before taking softmax, i.e.

$$\mathbf{p}^s = \text{softmax}\big(\mathbf{w}_{(p^s)}^{\top}\big[\mathbf{G};\mathbf{M}\big]; w_s\big), \tag{12}$$

  Regarding the ending position of the answer in the context paragraph, matrix $\mathbf{M}$ computed above is passed into another bidirectional GRU to obtain $\mathbf{M}^e \in \mathbb{R}^{2d\times K}$. After that $\mathbf{M}^e$ is used to compute the probability of each position being the ending position in a similar manner.

$$\mathbf{p}^e = \text{softmax}\big(\mathbf{w}_{(p^e)}^{\top}\big[\mathbf{G};\mathbf{M}^e\big]; w_e\big), \tag{13}$$

  where w_e is the trainable scalar weight for the extra position created for unanswerable questions. In another perspective, w_s and w_e are the trainable thresholds for determining whether the question is answerable or not.
  After obtaining a list of starting and ending position probabilities, the pairs with ending position greater than the starting position are eliminated. Meanwhile, given that the majority of answer tokens' length is less than 20, for computational efficiency the maximum length of answer tokens is restricted to be 16. After filtering away invalid answers, the starting and ending position pair with the maximum value of $p^s \times p^e$ is chosen as the answer. In addition, if any of the starting and ending position in the pair falls into the no answer position, the question is determined as unanswerable.

- **Training** The objective function (to be minimized) is defined as the sum of the negative log likelihood of the ground truth starting and ending positions as computed in the predicted probability vectors, and taking average on all N examples:

$$\ell = -\frac{1}{N}\sum_{i}^{N}\big[\log\big(\mathbf{p}_{y_i^s}^s\big) + \log\big(\mathbf{p}_{y_i^e}^e\big)\big], \tag{14}$$

  where y_i^s and y_i^e are the ground truth starting and ending position of the i-th example respectively and $\mathbf{p}_k$ represents the k-th entry in the vector $\mathbf{p}$.

Table 3 tabulates the key hyperparameters used in the BiDAF model with different word embedding inputs. Note that the value for K and V for BERT input are slightly greater than that of GloVe and ELMo (which is the default size in the BiDAF paper) because BERT adopts words piece tokenization and creates a longer token sequence in general. The training of BiDAF models are done using a Tesla K80 GPU card with 12GB RAM. GloVe vector inputs can be trained on a batch size of 60 due to the relatively lower vector dimensionality and no finetuning required. With the increase in model complexity, ELMo and BERT-Base can only be trained with a maximum batch size of 32 and 16 respectively. BERT-Large is not experimented with BiDAF since its maximum batch size for training on GPU is too small which may harm the classification result. In addition, the learning rate for GloVe and ELMo follows that in the BiDAF paper while that for BERT is referenced from the BERT model[16]. All three models are trained by Adam optimizer.

Table 3: Hyperparameters of BiDAF model using GloVe/ELMo/BERT input

	GloVe	ELMo	BERT-Base
Sequence length of context paragraph tokens (K)	300	300	320
Sequence length of question tokens (V)	30	30	40
Dimension of word vector (d)	300	1024	768
Dimension of contextualized vector (f)	150	150	150
Learning rate	0.001	0.001	3e-5
Batch size	60	32	16

3.2.2 BERT Finetuning

Next, the modifications made to the classification layer of the QA model with BERT input are elaborated. Unlike BiDAF, which separate context paragraph representation matrix **C** and question representation **Q**, under BERT the context paragraph word pieces are concatenated behind the question word pieces to form one single sequence. The BERT model computes the attention between the context paragraph and question and output $\mathbf{X} \in \mathbb{R}^{d \times K}$, where d is the dimension of the BERT vector and K is the sequence length of the word piece tokens.

Feedforward Neural Network(FFN) According to BERT's GitHub repository [16], the classification for the QA model is simply 1 feedforward neural network layer, i.e.

$$\mathbf{Y} = \mathbf{W}\mathbf{X} + \mathbf{b}. \tag{15}$$

where $\mathbf{Y} \in \mathbb{R}^{2 \times K}$ and it is referred as *logits* hereinafter, $\mathbf{W} \in \mathbb{R}^{2 \times d}$ and $\mathbf{b} \in \mathbb{R}^{2}$. The first row of **Y** is the predicted logits of the starting position while the second row is the predicted logits for the ending position.

Prediction Similar to BiDAF, some invalid cases have to be thrown away (e.g. predicted ending position is before starting position) and the maximum length of answer word pieces is set to be 30. The answer with the largest sum of starting and ending logits is set as the preliminary answer. Before finalizing the prediction, the sum of starting and ending logits of the best answer is compared against the *null score*, which is the minimum of sum of starting and ending logits of the **[CLS]** token amongst all testing examples. If the difference between the null score and the sum of starting and ending logits for the best non null answer is greater than the pre-determined threshold, the question is determined as unanswerable.

Aside from using 1 feedforward neural network only, 2 feedforward neural network layers with and without ReLU activation function are also experimented in this study.

Meanwhile, several mechanisms that route the information through the neural network in different manner, namely *Gating*, *Highway networks*[8] and *Residual learning*[14] are separately incorporated into the classification layer. It is hypothesized that they can guide the model to select information from BERT output that is useful for predicting the starting and ending positions of the answer phrase.

Gating The gating mechanism is originally adopted in recurrent neural networks such as LSTM and GRU in processing long sequential data. Xue and Li (2018)[19] proposed a model based on convolutional neural networks and a special gating mechanism called *Gated tanh-ReLU Units (GTRU)* in aspect based sentiment analysis and achieved both effectiveness and efficiency in the prediction task. The GTRU unit can selectively output features that are crucial to the prediction task and increase model performance. Given this, it is used to replace the vanilla feedforward neural network layer in this study. The GTRU unit is defined as:

$$\mathbf{Y} = \mathrm{ReLU}(\mathbf{W_R}\mathbf{X} + \mathbf{b_R}) \odot \tanh(\mathbf{W_T}\mathbf{X} + \mathbf{b_T}). \tag{16}$$

where $\mathbf{W_R} \in \mathbb{R}^{2\times d}$, $\mathbf{W_T} \in \mathbb{R}^{2\times d}$, $\mathbf{b_R} \in \mathbb{R}^2$ and $\mathbf{b_T} \in \mathbb{R}^2$.

Recall that $\text{ReLU}(x) = \max(0, x)$, therefore a zero score at a particular output position from the ReLU unit means that feature at the same output position of the tanh unit will be blocked since the product of zero and any real number is zero. Otherwise, if the value from the ReLU unit is positive, the score at the output position will be magnified. In this way, important features for predicting the starting and ending positions for answer are routed through the gate.

Highway Network Srivastava et al.(2015) [8] proposed the Highway networks in attempt to optimize the training of deep neural networks. They allow seamless information flow to the latter layers in the deep networks. The architecture makes use of gating units which learn to regulate the flow of information through a network. The Highway network is defined as

$$\mathbf{Y} = H(\mathbf{XW_H} + \mathbf{b_H}) \odot T(\mathbf{XW_T} + \mathbf{b_T}) + \mathbf{X} \odot C(\mathbf{XW_C} + \mathbf{b_C}). \tag{17}$$

where $\mathbf{W_H} \in \mathbb{R}^{d\times d}$, $\mathbf{W_T} \in \mathbb{R}^{d\times d}$, $\mathbf{W_C} \in \mathbb{R}^{d\times d}$, $\mathbf{b_H} \in \mathbb{R}^d$ and $\mathbf{b_T} \in \mathbb{R}^d$ and $\mathbf{b_C} \in \mathbb{R}^d$.

In this study H is chosen to be ReLU function, T is the sigmoid function and $C = 1 - T$, hence

$$\mathbf{Y} = \text{ReLU}(\mathbf{XW_H} + \mathbf{b_H}) \odot \sigma(\mathbf{XW_T} + \mathbf{b_T}) + \mathbf{X} \odot [\mathbf{1} - \sigma(\mathbf{XW_T} + \mathbf{b_T})]. \tag{18}$$

Note that when $T(\mathbf{XW_T}+\mathbf{b_T}) = 1$, the output of the Highway network is $H(\mathbf{XW_H}+\mathbf{b_H})$ whilst when $T(\mathbf{XW_T}+\mathbf{b_T}) = 0$, the output of the Highway network is $\mathbf{X}$. In this way, the networks control the proportion of $H(\mathbf{XW_H} + \mathbf{b_H})$ and $\mathbf{X}$ passing into the next layer. In the QA model context, it is to select the features from the BERT output that are important to the prediction of the starting and ending position of the answer. With an additional plain feedforward neural network, the logits for the starting and ending positions are predicted. Due to time constraint, only 1 Highway network is experimented in this study.

Residual Learning Similar to Highway networks, Residual learning [14] was proposed to combat the inefficient training of deep neural network, particularly in image classification. While deep neural network can learn deeper and more complex relationship amongst variables, vanishing or exploding gradient likely cause the divergence of network training and information loss during forward propagation. Residual learning is characterized by directly forwarding and adding the output $\mathbf{X}$ of from an earlier neural network layer L_k to the output of later layer L_{k+r}. The graphical representation of a building block of residual network is illustrated in Figure 6.

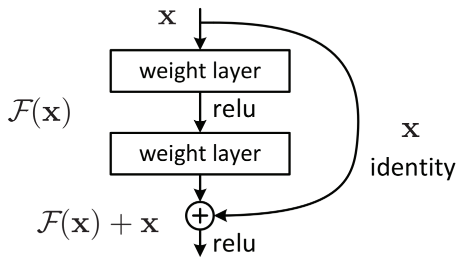

Figure 6: A building block for residual network (Source: He et al., 2016) [14]

As seen from Figure 6, the output $\mathbf{X}$ from an earlier layer proceed via 2 separate paths. One serves as the input to the next layer while another is directly added to the output of the layer at a later stage.

The merit of doing so is that the later layers only require learning the incremental/residual information $F(x)$ instead of learning the output from earlier neural network layers from scratch, since they are "starting" from where the earlier layers are. It mitigates the difficulties of neural network training as depth increases. In this study, 1 and 5 residual learning blocks are incorporated before the final classification layer respectively.

Loss function The loss function for the BERT finetuning models is the average of sum of categorical cross entropy for starting and ending positions, i.e.

$$J = -\frac{1}{2N}\Big\{\sum_{j=1}^{K}\sum_{i=1}^{K} \mathbb{1}_{PS_j \in GS_i} \log[\Pr(PS_j \in GS_i)] + \sum_{j=1}^{K}\sum_{i=1}^{K} \mathbb{1}_{PE_j \in GE_i} \log[\Pr(PE_j \in GE_i)]\Big\}, \tag{19}$$

where N is the number of question-answer pairs in the batch, K is the maximum sequence length of the question-context word pieces, PS and PE are the predicted starting and ending positions respectively, GS and GE are the ground truth starting and ending positions respectively.

The model training on both BERT-Base and BERT-Large are done using Google Cloud TPU, which allows the batch size to be set at 32. The learning rate is kept at 3e-5 while Adam optimizer is used during training as suggested by Google AI research team.

4 Experiments and Results

4.1 Evaluation Metrics

Exact Match (EM) and **F1** are defaulted by the organizer of SQuAD challenge to evaluate the predictive performance of a given QA model. Exact Match outputs 1 when the predicted answer exactly match the ground truth answer and 0 otherwise. F1 is the harmonic mean of Recall and Precision of the prediction. Assume there are n questions,

$$\text{EM} = \frac{1}{n}\sum_{i=1}^{n}\mathbb{I}\Big\{\text{predicted tokens for question } i = \text{ground truth answer tokens for question } i\Big\} \tag{20}$$

$$\text{F1} = \frac{1}{n}\sum_{i=1}^{n}\frac{2\times\text{Recall}_i\times\text{Precision}_i}{\text{Recall}_i+\text{Precision}_i} \tag{21}$$

$$\text{Recall}_j = \frac{\text{No. of predicted tokens for question } j \in \text{ground truth answer tokens for question } j}{\text{No. of ground truth answer tokens for question } j} \tag{22}$$

$$\text{Precision}_j = \frac{\text{No. of predicted tokens for question } j \in \text{ground truth answer tokens for question } j}{\text{No. of predicted answer tokens for question } j} \tag{23}$$

Formula (22) and (23) only apply to answerable question j. The values for Precision_j and Recall_j are both 1 when the question is unanswerable and the predicted answer is null, 0 otherwise.

The example below demonstrates the computation of EM and F1 score when the question is answerable.

- **Context:** "The only words of Jesus on the cross in the Mark and Matthew accounts, this is a quotation of Psalm 22. Since other verses of the same Psalm are cited in the crucifixion accounts, it is often considered a literary and theological creation. Geza Vermes, however, points out that the verse is cited in Aramaic rather than the Hebrew in which it usually would have been recited, and suggests that by the time of Jesus, this phrase had become a proverbial saying in common usage. Compared to the accounts in the other Gospels, which he describes as 'theologically correct and reassuring', he considers this phrase 'unexpected, disquieting and in consequence more probable'. He describes it as bearing 'all the appearances of a genuine cry'. Raymond Brown likewise comments that he finds 'no persuasive argument against attributing to the Jesus of Mark/Matt the literal sentiment of feeling forsaken expressed in the Psalm quote'." [20]
- **Question:** "What was the psalm said to be in Jesus' time?"
- **Ground truth answer:** "a proverbial saying in common usage"
- **Predicted answer 1:** "a proverbial saying in common usage"

  $\text{EM} = 1$
  $\text{F1} = 1$
- **Predicted answer 2:** "common usage"

  $\text{EM} = 1$
  $\text{Recall} = \frac{2}{6} = \frac{1}{3}$
  $\text{Precision} = \frac{2}{2} = 1$
  $\text{F1} = \frac{2\times\frac{1}{3}\times 1}{\frac{1}{3}+1} = 0.5$

4.2 Results and Discussion

The performance metrics of the various QA models are tabulated in Table 4. Each model in the table is ran 3 times with different initial randomization in order to obtain consistent results. It turns out that all models exhibit consistency in terms of F1 and EM scores in the development dataset after repeated model training. For each model, the F1 and EM scores on development set (*Dev F1* and *Dev EM*) for the trial attaining the highest F1 score are reported. Meanwhile, ensemble technique is applied on models using BERT-Large. It is done by summing the probabilities of the answers in each trial and applying either equal weight, weights based on Dev F1 or Dev EM of the model. The answer with the highest aggregated probability is considered as the final answer. The predictions from ensembles of BERT-Large model are also evaluated by F1 and EM scores and they are appended as the *Ensemble Dev F1* and *Ensemble Dev EM* column in the table respectively.

Table 4: Experiment Results

No.	Model	Dev F1	Dev EM	Ensemble Dev F1	Ensemble Dev EM
1	BiDAF + GloVe	54.622	50.678	-	-
2	BiDAF + ELMo	62.388	59.244	-	-
3	BiDAF + BERT Base Uncased	59.695	56.565	-	-
4	BERT-Base Uncased + 1 FFN	77.158	74.050	-	-
5	BERT-Large Uncased + 1 FFN	80.884	77.899	81.450*	78.573*
6	BERT-Large Uncased + 2 FFN	81.168	78.253	81.841#	79.121#
7	BERT-Large Uncased + 2 FFN + ReLU	80.628	77.756	81.429#	78.624#
8	BERT-Large Uncased + Highway networks	80.720	77.975	81.211*	78.691*
9	BERT-Large Uncased + 1 Residual learning block	81.243	78.371	81.914#	79.045#
10	BERT-Large Uncased + 5 Residual learning blocks	81.233	78.278	82.016&	79.272&
11	BERT-Large Uncased + Gating	81.404	78.169	82.001*	79.205*
12	Ensemble of 10 and 11 (6 models)	-	-	82.294#	79.542#

Note: # stands for taking equal weight, * refers to weighted by EM score and & means weighted by F1 score.

According to Table 4, it is obvious that the BERT finetuning approach significantly outperforms BiDAF with an improvement of about 20 F1 and EM scores on the development dataset respectively. Another observation is that the performance of BiDAF with BERT input trails far behind the BERT finetuning peers, which shows that the extensive attention mechanism in BERT does not necessarily synergize with the bidirectional attention flow mechanism in BiDAF.

Amongst the BiDAF models, GloVe delivers the lowest F1 and EM scores when compare to that of ELMo and BERT-Base. Its context-free nature may have limited its performance as it cannot cater possible polysemy found in the context paragraph and questions. Hence contextualized word embedding do have an edge in the question answering task.

Meanwhile, BERT-Large outperforms Bert-Base by 2 to 3 F1 and EM scores likely due to more attention layers and 2 times more parameters that can better capture the information in the context and question tokens. Comparing Model 5 to Model 6-11, the addition of various specific layer between the BERT output and the final classification neural network layer slightly improve the performance of the models. Amongst all single model, model 11 with BERT-Large and gating mechanism tops amongst other rival models in this study with a F1 score of 81.404 and EM score of 78.169.

Ensemble is a common technique in data mining since it help reduce the non-systematic error committed by individual models (e.g. different initial randomization, answers that are hard to differentiate from other candidates in terms of prediction probability) by aggregating and considering the predictions from a group of models, thus reducing variance. With several combinations experimented, the ensemble of BERT-Large + 5 Residual Learning blocks and BERT-Large + Gating, which contains a total of 6 models, combine to deliver the highest F1 score of 82.294 and EM score of 79.542.

4.3 Error Analysis

Understanding the errors committed by the trained model might provide insights for area of improvement and future research directions. Under time and manpower constraint, only a random sample of 50 incorrectly answered questions are selected from Model 12 for further inspection. Half of them are answerable questions while the other half are unanswerable. Not withstanding the relatively small sample size, few observations are still made and summarized below:

- **Failure to handle certain questions with lexical variation**

  Context: "There were many religions practiced during the Yuan dynasty, such as Buddhism, Islam, and Christianity. The establishment of the Yuan dynasty had dramatically increased the number of Muslims in China. However, unlike the western khanates, the Yuan dynasty never converted to Islam. Instead, Kublai Khan, the founder of the Yuan dynasty, favored Buddhism, especially the Tibetan variants. As a result, Tibetan Buddhism was established as the *de facto* state religion..."[21]

Question: "What was the Yuan's *unofficial* state religion?"

Ground Truth Answer: "Tibetan Buddhism"

Predicted Answer: Nil

In this question, the model fails to link the term "de facto" to its synonym "unofficial". It could be due to the finetuned BERT model does not have sufficient knowledge about the rare term "de facto". It is hypothesized that more training data, either from other dataset or adversarially created by computer program (e.g. swap the word "unofficial" with its other synonym such as "informal") could drive the model to learn the semantic relationship amongst the words more accurately.

- **Trapped by multiple sentence reasoning**

Context: "...In front of the field of macrocilia, on the mouth "lips" in some species of Beroe, is a pair of narrow strips of adhesive epithelial cells on the stomach wall that "zip" the mouth shut when the animal is not feeding, by forming intercellular connections with the opposite adhesive strip. This tight closure streamlines the front of the animal when it is pursuing prey."[22]

Question: "What does the beroe do when pursuing prey?"

Ground Truth Answer: ""zip" the mouth shut"

Predicted Answer: Nil

The model fails to fuse the meaning of 2 sentences to give the right answer. In particular, the model has to be able to associate the pronoun "it" right before "pursuing prey" to beroe, which requires a good comprehension of the 2 sentences.

- **Tricked by small twists in the questions**

Context: "Southern California is also home to a large home grown surf and skateboard culture. Companies such as Volcom, Quiksilver, No Fear, RVCA, and Body Glove are all headquartered here. Professional skateboarder Tony Hawk, professional surfers Rob Machado, Tim Curran, Bobby Martinez, Pat O'Connell, Dane Reynolds, and Chris Ward, and professional snowboarder Shaun White live in southern California..."[23]

Question: Where does professional surfer Tony Hawk live?"

Ground Truth Answer: Nil

Predicted Answer: "southern Califonia"

The model ignores the fact that Tony Hawk is not a *professional surfer* but a *professional skateboarder*. Aside from entity swap, the model is also fooled by situations like questions with negation word inserted or contradictory questions, etc. It has missed out the broader overall logical reasoning in such cases.

5 Limitations and Way Forward

Hyperparameters Tuning Hyperparameters tuning, one of the usual steps in deep learning to further boost the performance of the model, is limited in this project given the numerous trainable model parameters and considerable training time. In my point of view, rather than performing grid search, the benefits of hypothesizing and testing different model structures might outweigh the former's. Even so, it is still worth spending time to find the set of hyperparameters that can optimize the model performance going forward.

Design of dataset According to the investigation by Tom et al.(2019) [24], questions in SQuAD are unusually similar to the text containing the answer, due to priming effects in the question generation procedure. Even with the adversarially written unanswerable questions to penalize systems that rely heavily on context and type matching heuristics. They argued that identifying artificial questions in SQuAD requires less reasoning than determining whether a particular paragraph contains sufficient information to fully answer a question. Furthermore, human reading comprehension is indeed done in long passages instead of a short paragraph. The model trained in this project is trained on short paragraphs can might underperform when put in long passages. Hence a more down-to-earth dataset is needed in order to train a more robust Question Answering models. Google AI team spearheaded to create the Natural Questions corpus address the aforementioned limitations, which can be the paradigm for training question answering model in future.

Model Training There are infinite number of ways people can frame the same question, either be in a straight forward or indirect manner. Despite the ample number of question-answer pairs in the SQuAD dataset, it is natural to think that the more pairs to train the model the more natural language features the model can learn. Thus, to increase the training sample size and the variability in the type of question-answer pairs, data from other question answering datasets, for instances TriviaQA [25], WikiQA [26], etc. can also be used to train the question answering model. In this way the model may be able to understand more about the semantics, syntactic features and logical reasoning in natural language.

Additional variables Some more input variables, such as part-of-speech (POS) and named-entity tagging could supply extra information to the QA model for learning the syntactic and lexical characteristics in both the context paragraph and the question. It is worth trying in upcoming research.

6 Conclusion

In this research study, a combination of novel word embedding models and neural networks are deployed to construct a Question Answering model. Several notable takeaways include the outperformance of contextualized word embeddings over context-free embeddings in QA task, as well as the further improvement of QA model that built on BERT by modifying the classification layer and applying ensemble technique. After all, the ensemble of the state-of-the-art BERT-Large model with additional Gating mechanism and Residual learning blocks in the classification layer lead to the best F1 and EM scores in the experiment.

Going forward, there are rooms of improvement in building the QA model in terms of its training data, hyperparameters tuning and model architecture. In particular, the proven capability of question answering in long articles or documents instead of short paragraphs (as in SQuAD) should be taken as the long term goal of this research project.

References

[1] Pranav Rajpurkar, Robin Jia, and Percy S. Liang. Know what you don't know: Unanswerable questions for squad. In *ACL*, 2018.

[2] Wikipedia. Indigenous peoples of the Americas — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Indigenous%20peoples%20of%20the%20Americas&oldid=894586503`, 2019. [Online; accessed 29-April-2019].

[3] Wikipedia. Materialism — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Materialism&oldid=892571381`, 2019. [Online; accessed 29-April-2019].

[4] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[5] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proc. of NAACL*, 2018.

[6] 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.

[7] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, 1997.

[8] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *CoRR*, abs/1505.00387, 2015.

[9] Generalized language models: Cove, elmo & cross-view training, Apr 2019.

[10] Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, and Phillipp Koehn. One billion word benchmark for measuring progress in statistical language modeling. In *INTERSPEECH*, 2013.

[11] Nikita Nangia, Adina Williams, Angeliki Lazaridou, and Samuel R. Bowman. The repeval 2017 shared task: Multi-genre natural language inference with sentence representations. In *RepEval@EMNLP*, 2017.

[12] Daniel M. Cer, Mona T. Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *SemEval@ACL*, 2017.

[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[15] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Gregory S. Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016.

[16] Jacob Devlin. Tensorflow code and pre-trained models for bert. `https://github.com/google-research/bert`, 2018.

[17] Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *CoRR*, abs/1611.01603, 2017.

[18] Junyoung Chung, Çaglar Gülçehre, Kyunghyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014.

[19] Wei Wei Xue and Tao E Li. Aspect based sentiment analysis with gated convolutional networks. In *ACL*, 2018.

[20] Wikipedia. Crucifixion of Jesus — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Crucifixion%20of%20Jesus&oldid=894092396`, 2019. [Online; accessed 29-April-2019].

[21] Wikipedia. Yuan dynasty — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Yuan%20dynasty&oldid=894618426`, 2019. [Online; accessed 29-April-2019].

[22] Wikipedia. Ctenophora — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Ctenophora&oldid=894233868`, 2019. [Online; accessed 29-April-2019].

[23] Wikipedia. Southern California — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Southern%20California&oldid=888411908`, 2019. [Online; accessed 29-April-2019].

[24] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Matthew Kelcey, Jacob Devlin, Kenton Lee, Kristina N. Toutanova, Llion Jones, Ming-Wei Chang, Andrew Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: a benchmark for question answering research. *Transactions of the Association of Computational Linguistics*, 2019.

[25] Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[26] Yi Yang, , and Chris Meek. Wikiqa: A challenge dataset for open-domain question answering. ACL - Association for Computational Linguistics, September 2015.

7 Appendix: Model Architecture of Transformer

The transformer was proposed by Vaswani et al (2017) [13] for tasks that require modeling sequential data such as language model and neural machine translation, etc. It consists of a encoder and decoder stack as shown in Figure 7. The encoder receives an input sequence of word tokens and transform each of them to a list of learned word embeddings. These word embeddings flow through the encoder followed by the decoder to predict the next word token. At each time step the output word token also serves as the input to the decoder for predicting the next word token. The structure of the encoder and decoder are further elaborated in the paragraphs below:

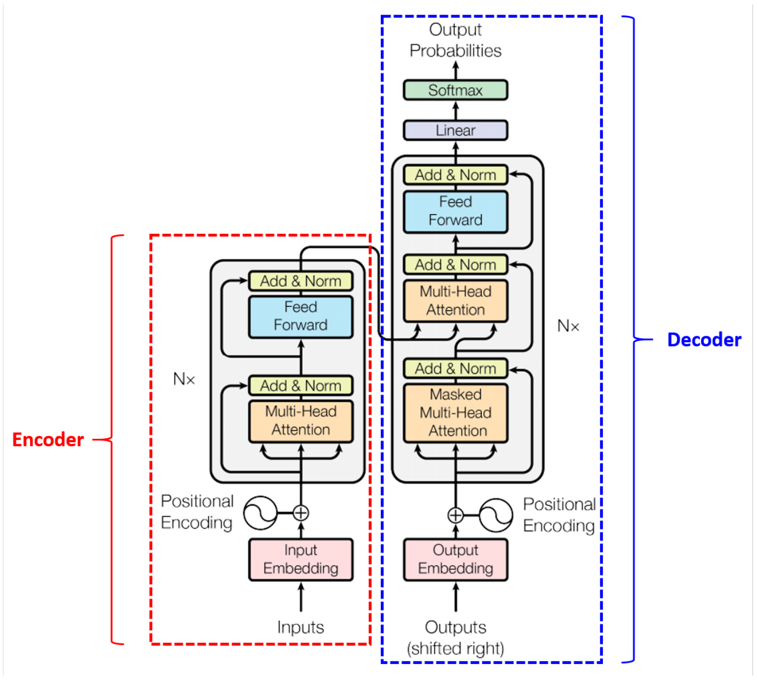

Figure 7: Model Architecture of the Transformer (Source: Vaswani et al., 2017) [13]

7.1 Encoder

There are N identical layers in the encoder, and each layer can be further split into 2 sub-layers. The first sub-layer operates the *multi-head self-attention*, while the second is a fully connected feedforward neural network (FFN). Residual connection is also applied to each of the two sub-layers before layer normalization. Hence, the output of each sub-layer is LayerNormoalization(x + Sublayer(x)), where Sublayer(x) is either multi-head self-attention or FFN. In the BERT model, the dimension of the outputs of all sub-layers and embedding layers are $d_{\text{model}} = 512$.

7.2 Decoder

There are also N identical layers in the decoder. Aside from the two sub-layers mentioned in encoder layer, the decoder has an additional third sub-layer, which performs multi-head attention over the encoder output. Meanwhile, residual connections followed by layer normalization are also deployed around each of the 3 types of sub-layers. Different from the encoder layer, the self-attention sub-layer in the decoder stack is modified to restrict positions to attend to previous positions only. It is done through applying masks with negatively large enough values to the positions behind. This ensures that the predictions for a position i can depend only on the outputs at previous positions.

7.3 Attention

- Assume the input at each encoder/decoder layer is fixed to have n word tokens, horizontally concatenating their respective d_{model}-dimensional input embedding vectors yields the input matrix $\mathbf{X} \in \mathbb{R}^{n \times d_{\text{model}}}$. The first step in the attention mechanism is to transform $\mathbf{X}$ into a set of 3 matrices, namely query $\mathbf{Q}$, key $\mathbf{K}$ and value $\mathbf{V}$. The transformation is done by multiplying $\mathbf{X}$ to the trainable weight matrices $\mathbf{W}^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $\mathbf{W}^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$

and $\mathbf{W}^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$ respectively. $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$ are abstractions that are useful for computing attention in subsequent steps.

$$\mathbf{XW}^Q = \mathbf{Q} \in \mathbb{R}^{n \times d_k}; \mathbf{XW}^K = \mathbf{K} \in \mathbb{R}^{n \times d_k}; \mathbf{XW}^V = \mathbf{V} \in \mathbb{R}^{n \times d_v} \tag{24}$$

- Next, the "Scaled Dot-Product Attention" in applied to $\mathbf{Q}$, $\mathbf{K}$, $\mathbf{V}$ as follows:

$$\text{Attention}\big(\mathbf{Q}, \mathbf{K}, \mathbf{V}\big) = \text{softmax}\big(\frac{\mathbf{QK}^\top}{\sqrt{d_k}}\big)\mathbf{V} \tag{25}$$

- Apart from just performing attention once at each sub-layer, researchers for Transformer found it beneficial to repeat the attention mechanisms simultaneously at the same sub-layer for h times, where the weight matrices $\mathbf{W}^Q$, $\mathbf{W}^K$ and $\mathbf{W}^V$ are distinct in each of the h attentions. In other words, the attention mechanism is simultaneously happening at h heads. Multi-head attention is intended to jointly attend to information from different representation subspaces at different positions. Therefore, we have

$$\text{MultiHead}\big(\mathbf{Q}, \mathbf{K}, \mathbf{V}\big) = \text{Concat}\big(\text{head}_1, ..., \text{head}_h\big)\mathbf{W}^O \tag{26}$$
$$\text{where head}_i = \text{Attention}\big(\mathbf{QW}_i^Q, \mathbf{KW}_i^K, \mathbf{VW}_i^V\big) \tag{27}$$

Where the trainable weight matrices are $\mathbf{W}_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $\mathbf{W}_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $\mathbf{W}_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$, $\mathbf{W}_i^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$.

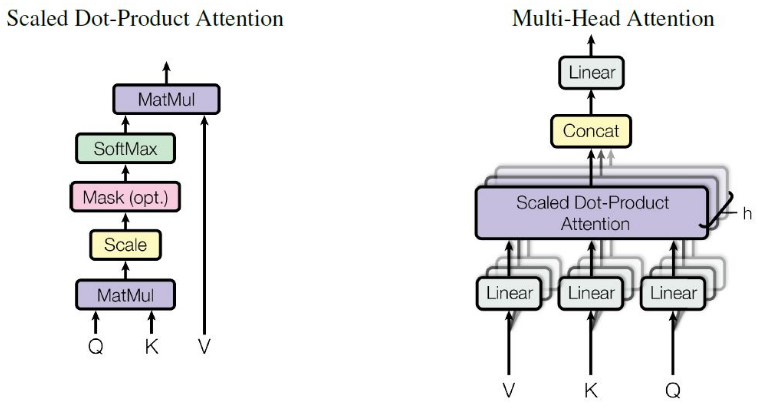

Figure 8: Attention Mechanism in the Transformer (Source: Vaswani et al., 2017) [13]

7.3.1 Position-wise Feed-Forward Networks (FFN)

After the attention sub-layers, each position in the encoder and decoder would be connected to a separate feedforward network, which is essentially two linear transformations with ReLU activation.

$$\text{FFN}(x) = \max\big(0, xW_1 + b_1\big)W_2 + b_2 \tag{28}$$

Despite the same linear transformations across different positions, different parameters are trained for the transformations.

7.3.2 Embeddings and Softmax

Speaking of the vector representation of the initial input and final output of the Transformer, the learned embeddings are used to represent the input tokens and output tokens in vectors of dimension d_{model}. A linear transformation with softmax activation is used to convert the decoder output to predicted probabilities of the next token. In Transformer, the same weight matrix is shared between the input and output token embedding layers and the pre-softmax linear transformation. In the embedding layers, the weights are multiplied by $\sqrt{d_{\text{model}}}$.

7.4 Positional Encoding

The Transformer also takes the order of the token sequence into consideration, some extra information about the relative or absolute position of the tokens in the sequence are added to the Transformer model. The extra information is indeed the "positional encodings" that are added to the input embeddings at the bottoms of the encoder and decoder stacks. The positional encodings have the same dimension d_{model} as the embeddings such that they can be summed.

Amongst ample of choices of positional encodings, Transformer adopts the sine and cosine functions as follows:

$$PE_{pos,2i} = sin\big(pos/10000^{2i/d_{\text{model}}}\big) \tag{29}$$

$$PE_{pos,2i+1} = cos\big(pos/10000^{2i/d_{\text{model}}}\big) \tag{30}$$

where pos is the position and i is the dimension. Each dimension of the positional encoding corresponds to a sinusoid. The wavelengths form a geometric progression from 2π to $10000\dot{2}\pi$. It is hypothesized that the functions would allow the model to easily learn to attend by relative positions, because for any fixed offset k, PE_{pos+k} can be represented as a linear function of PE_{pos}.