

Machine Reading Comprehension using Bidirectional Attention in Deep Neural Networks

Project Report Submitted in Partial Fulfilment of the Requirements for the
Degree of

Bachelor of Technology

in

Computer Science and Engineering

Submitted by

Anusha Porwal (Roll No. 1710110069)

Under the Supervision of

Dr Ketan Bajaj

Visiting Faculty, SNU

Strategic Management, Enterprise Technology Office

HCL Technologies

The logo of Shiv Nadar University, featuring the name "SHIV NADAR UNIVERSITY" in a blue, serif font, with a decorative horizontal line above it.

Department of Computer Science and Engineering

May, 2020

Contents

Declaration.....	4
Acknowledgement	5
Abstract.....	6
List of Figures	7
List of Tables	8
List of Symbols and Abbreviations.....	9
Chapter 1 Introduction	10
1.1 What is Machine Reading Comprehension?	10
1.2 Motivation	10
Chapter 2 Background work/ Starting Point.....	11
Chapter 3 Literature Review	12
Chapter 4 Bi-directional Attention Flow Model.....	16
4.1 About Bidirectional Attention Flow.....	16
4.1.1 What is the Attention Mechanism?	16
4.1.2 Attention Mechanisms in previous models	17
4.1.3 What is Bi-Directional Attention?.....	17
4.1.4 Older models don't have Bidirectional Attention Flow – what makes this better?. 17	
4.2 Tokenisation and Creation of Embedding Matrices.....	18
4.3 Highway Layer.....	21
4.4 Contextual Meaning Layer.....	23
4.5 Similarity Matrix	26
4.6 Context to Query Attention	29
4.7 Query to Context Attention	30
4.8 Mega Merge	33
4.9 Modelling Layer.....	34
4.10 Output Layer and Answer Span Probabilities	36

4.11 Training Functions	39
4.11.1 Accuracy Function.....	39
4.11.2 Loss Function	41
4.12 Bidirectional Attention Flow Model using TensorFlow	42
Chapter 5 Requirements.....	45
5.1 Datasets	45
5.2 Technical	50
Chapter 6 Results: Training and Testing	51
Chapter 7 Future Scope of the Project	55
References.....	56
Other References	59

Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Name of the Student Anusha Porwal Signature and Date Anusha 19/05/21

Acknowledgement

I would like to sincerely thank my project advisor, Professor Dr Ketan Bajaj for guiding me through this project. His enthusiasm, immense patience, insightful comments and advise, ideas and motivation have helped me greatly throughout this project. His vast knowledge in the field of machine learning helped me in bringing this project to a successful end.

I would also like to thank Professor Dr Dolly Sharma for coordinating the Final Semester Project. Her availability and support helped me in keeping up with the deadlines and completing the project on time.

I would also like to extend my gratitude the entire Computer Science Department of Shiv Nadar University for motivating me and providing me with enough options so I could find a field that I enjoy working in.

Last but not the least, I would like to thank my Parents and Friends for motivating me throughout this journey and keeping my spirits up, enabling me to finish my project.

Abstract

The task of teaching a machine to go through a passage and answer a question, like a human would is called Machine Reading Comprehension (MRC). It is a new and challenging topic in Machine Learning. This project uses the technique of Bi-directional Attention in Deep Neural Networks, originally put forward by the people of Allen Institute for Artificial Intelligence, University of Washington. Previous models using Attention, would use it to summarise the passage into a fixed size vector, or take only Uni-directional Attention of Query on the Context. The advantages of this technique, along with the pipeline of this model are explored in great detail. The model is trained on the SQuAD dataset, and implemented using TensorFlow (Version 2.3).

List of Figures

Figure 4.1 Creating Embedding Matrices for Context and Query	20
Figure 4.2 Information flow in Highway Layer.....	22
Figure 4.3 Highway Layer - Code Snippet 1	22
Figure 4.4 Highway Layer - Code Snippet 2	23
Figure 4.5 BiLSTM for Contextual Meaning of Context and Query Matrices	25
Figure 4.6 BiLSTM Code Snippet.....	26
Figure 4.7 Creating Similarity Matrix from Query and Context Matrices	27
Figure 4.8 Similarity Matrix - Code Snippet 1	27
Figure 4.9 Similarity Matrix - Code Snippet 2	28
Figure 4.10 Similarity Matrix - Code Snippet 3	28
Figure 4.11 Context to Query Attention Computation.....	29
Figure 4.12 Context to Query Attention - Code Snippet 1.....	30
Figure 4.13 Query to Context Attention Computation.....	31
Figure 4.14 Query to Context Attention - Code Snippet 1.....	32
Figure 4.15 Query to Context Attention - Code Snippet 2.....	32
Figure 4.16 Mega - Merge operation to obtain the Merged Context	33
Figure 4.17 Mega Merge - Code Snippet 1	34
Figure 4.18 Modelling Layer using BiLSTMs	35
Figure 4.19 Modelling Layer - Code Snippet 1	36
Figure 4.20 Finding Start and End Span of Answer.....	37
Figure 4.21 Start Span - Code Snippet 1	38
Figure 4.22 End Span - Code Snippet 1	38
Figure 4.23 Accuracy Calculation.....	40
Figure 4.24 Accuracy Function - Code Snippet 1	40
Figure 4.25 Loss Calculation	41
Figure 4.26 Loss Function - Code Snippet 1	42
Figure 4.27 Bidirectional Attention Flow Model - Code Snippet 1.....	43
Figure 4.28 Bidirectional Attention Flow Model - Code Snippet 2.....	43
Figure 4.29 Model Summary	44
Figure 4.30 Number of trainable parameters in the model	44
Figure 5.1 Example from SQuAD Dataset ^[31]	45
Figure 5.2 Modified Dataset - Context File.....	46

<i>Figure 5.3 Modified Dataset - Question File.....</i>	<i>47</i>
<i>Figure 5.4 Modified Dataset - Answer File.....</i>	<i>47</i>
<i>Figure 5.5 Modified Dataset - Answer Span File.....</i>	<i>47</i>
<i>Figure 5.6 Data Generator - Code Snippet 1.....</i>	<i>48</i>
<i>Figure 5.7 Data Generator - Code Snippet 2.....</i>	<i>49</i>
<i>Figure 6.1 Training for 1 epoch.....</i>	<i>51</i>
<i>Figure 6.2 Training for 3 additional epochs.....</i>	<i>52</i>

List of Tables

<i>Table 3.1 Literature Review of Various MRC Techniques.....</i>	<i>15</i>
<i>Table 6.1 Model Summaries.....</i>	<i>52</i>
<i>Table 6.2 Comparing Test Results.....</i>	<i>54</i>

List of Symbols and Abbreviations

1	BiDAF ^[1]	Bi-directional Attention Flow Model proposed by Minjoon Seo et al [1]
1	$numContext$	Number of words in the Context.
2	$numQuery$	Number of words in the Query.
3	$numWords$	Number of words in the input matrix. Could be for Context or Query.
4	emb_dim	Stands for Embedding Dimension. Represents the length of the vector.
5	$T(x, W_T)$	Transform gate in the Highway Layer.
6	$C(x, W_C)$	Carry gate in the Highway Layer.
7	W_A	Trainable weight vector. Subscript A represents the Layer or gate where the weight is being used.
8	BiLSTM	Bi-directional Long-Short Term Memory Layer
9	$[;]$	Represents vector concatenation (row-wise).
10	*	Element wise multiplication of 2 vectors.
11	SQuAD	Stanford Question Answering Dataset

Chapter 1 Introduction

1.1 What is Machine Reading Comprehension?

Given some text and a question, Machine Reading Comprehension is supposed to answer the question from the given passage. There are different ways of classifying MRC models:

- *Extractive and Abstractive*: Extractive is when you look for the answer in the text and return that simply. You usually find the answer within a single sentence. Abstractive is a little more complex than extractive. It paraphrases the answer extracted from the context and returns it.
- *Open and Closed Domain*: Open Domain model has access to a lot of information, and answers the question based on this pre-existing knowledge. Closed Domain models only works on the context given to them.
- *Factoid/Non-factoid Question Answering*: Factoid questions are those which have short factual answers. For non-factoid question answering, the model has to read, analyse and inference and then answer the query.

For this project, we are looking at Extractive, Closed Domain and Factoid question answering models. We'll be looking at Bi-directional Attention Flow, a model proposed by the people of Allen Institute for Artificial Intelligence, University of Washington ^[1].

1.2 Motivation

The problem of Machine Reading Comprehension is still an open problem in the Computer Science domain, with solutions that don't have the best accuracies. It is because of this that MRC is currently a very active field of research. It is a complicated task to solve and requires a lot of subtasks to be performed. I picked this problem because it is very interesting and an upcoming field. This project would serve as my first step into the world of Machine Reading Comprehension. I would learn about the different techniques and improvements over time and the complexities, intricacies and problems of the models. Implementing my own version of one of the models would give me a greater understanding of the technical requirements and the work that goes into making a machine comprehension model, and I would be equipped to work on a novel solution in the future.

Chapter 2 Background work/ Starting Point

In the previous semester, I did a Project for the Natural Language Processing course taught by Dr Ketan Bajaj. It was a project on Document Searching. Given some texts and a query word or phrase, this tool would search for the query and also for words and phrases that are semantically similar to the query. Finally, it would return the top 7 matches from all the texts, with some context so the user can understand what that paragraph is talking about. This was done on Python, with the help of GloVe Vector Embeddings and TF-IDF. Such a tool can be very useful for the question answering problem, when the word in the query isn't present in the context. In this case we need to look at synonyms or semantically similar words/phrases. It is also quite similar to the task of MRC, as we need to pick out the correct parts from the given document, just like we need to pick out the correct answer span from the context in MRC based on the query.

Chapter 3 Literature Review

The first step in this project was reading a literature review paper on MRC techniques, to gather some knowledge on the existing models. 10 recent models were found, that gave good results. They were: ReasoNet, RNet, QANet, Gated Attention Reader, BiDAF, FastQA, BERT, Pointer Networks, mLSTM and Dynamic Co-attention.

Information was collected on the headings: They types of neural networks used, advantages and disadvantages, accuracy and how dependent is the model on training data.

A summary of the models is given below:

Name + Year	Neural Networks Used	Advantages	Disadvantages	How dependent is it on the training data	Accuracy
RNet ^[2, 3] , 2017	BiDRN (GRU), gated attention-based RNs, self-matching gated attention RNNs, pointer networks	<ul style="list-style-type: none">• Question-aware passage representations are made. They basically give a higher weight to words that relevant to answering the question and lesser weight to words that are irrelevant to the question. It's like a mask.• RNNs can only memorize limited passage context, so to take care of that, self-matching is done so that the passage representations are refined and have information from the full passage.• GRUs are used instead of LSTM because they perform the same but are computationally cheaper	<ul style="list-style-type: none">• Complex inferences are not possible using this model.• Since you're making question specific passage representations, you would have to make different representations for the same passage if the question is changed• Additional features like Sentence Ranking, POS tags, NER results, PCFG tags, etc don't have any effect on the accuracy of the model.	Increase in data increases accuracy, but takes time and slows down inferencing. Also, additional features don't have any positive affect on the model.	72.3% on SQuAD test set 76.9% using an ensemble model.
QANet ^[4, 5] 2018	CNN and Self attention. (Feed forward model only)	<ul style="list-style-type: none">• Faster and can process tokens in parallel too, so the model can be trained for a greater number of iterations as well as with more data, and can be easily deployed in real time too• Make query aware context vectors for each position in	<ul style="list-style-type: none">• Query aware context vectors would mean that if the query changes for the same passage/context, the whole process would have to be done again.	Increasing the amount of data, increases the model's accuracy. So, for that they proposed a data augmentation technique. This technique translates the original sentence into a different	Test set accuracy of 76.2% for an exact match on SQuAD dataset with augmented data, and 84.6%

		<p>the context paragraph, which is used in later layers.</p> <ul style="list-style-type: none"> • Use Convolution and Self attention instead of RNNs, that is very popular in NLP models. This feed forward model is what makes its training and inferencing faster than other models 		<p>language then back to English. The meaning remains the same, but it is paraphrased.</p>	<p>accuracy for F1 score (measures the overlap between predicted answer and actual answer.</p>
ReasonN et ^[6] 2017	LSTM, BiGRU, Bidirectional RNN	<ul style="list-style-type: none"> • Instead of having a fixed number of iterations of going over the passage, there is a termination state. This state can decide whether it should continue reading and proceed to the next iteration, or should it terminate because the existing information is enough to answer the question. 	<ul style="list-style-type: none"> • This model is slow to train and infer from. 		<p>74.7% on CNN dataset, 76.6% on Daily Mail dataset, 69.1% on SQuAD, and 73.4% on SQuAD using ensemble learning</p>
Gated Attention Reader ^[7] 2017	CNN, Multihop propagation (which uses Gated Relational Graph Convolutional Network (Gated-RGCN))	<p>Existing neural network readers are restricted to either attend to tokens or entire sentences, with the assumption that certain sub-parts of the document are more important than others. But GA weigh individual components of the vector representation of each token in the document separately</p>	<p>GA when used with any other work like CNN or CBT, couldn't find disadvantages</p>	<p>Adding featured increases accuracy by around 3.2% - 3.5%</p>	<p>Ranges from 65-78% depending on the datasets.</p>
BiDAF ^[1, 7] 2017	CNN and RNN	<p>It is a multi-stage hierarchical process that represents the context at different levels of granularity and uses a bi-directional attention flow mechanism to achieve a query-aware context representation without early summarization</p>	<p>It doesn't incorporate multiple hops of the attention layer. For attaining more accuracy, it needs to implement hyper parameter tuning and monitoring gradient updates in the different layers of the network. Some models implemented the hyper parameter tuning but it slows down the speed of the model which in turns</p>	<p>Increasing the dataset on increasing the embedding improves the accuracy but on the multiple layered module (the ensemble one) but the simple model got slowed down and accuracy gets reduced on increasing the dataset.</p>	<p>Single Model-72% (approx.) and Ensemble model-76% (approx.)</p>

			reduces the accuracy to a great level.		
FastQA [7] 2017	BiRNN and Embeddings, LSTMs, RNN	FastQA is very competitive with existing, complex neural models. FastQA model to address linguistically motivated error types, e.g., fine-grained understanding and distinction of answer types or learning to respect syntax.	There are added complexities in FastQA model in order to get better results but it was observed that additional complexity introduced by the interaction layer is not necessarily justified by the incremental performance improvements.	A particular checkpoint is set, having dataset below the checkpoint keeps on increasing the accuracy but in order to get an efficient model, the models have been cut into 400 tokens.	78.90%
BERT [7, 8] 2018	Transformers and RNN	It has gained extreme popularity because here a pre-trained model with little fine tuning can be used for a specific task. Most of the papers till now read the words in the input text sequentially. A transformer reads all of the words at once which gives a context of the word based on the surroundings.	This drawback exists with all ML models, if your situation is very specific then BERT would require a lot of training data and would end up being computationally intensive.	BERT being pre-trained on a massive corpus, which lets it stay independent of the training data, but the last couple of layers are fine-tuned which will benefit from the rich representations of the corpus. Accuracy might not show a lot improvement with large dataset (other than pre-trained model)	On SQuAD v1.1, it achieves 93.2% F1 score, surpassing the previous state-of-the-art score of 91.6% and human-level score of 91.2%:
Pointer Networks [7] 2015	RNN based sequence to sequence models.	Pointer-Nets not only improve over sequence-to-sequence with input attention, but also allow us to generalize to variable size output dictionaries.	Pointer Networks tackle problems where input and output data are sequential data, but can't be solved by seq2seq type models because discrete categories of output elements depend on the variable input size (and are not decided in advance).		Ptr-Net model for n = 5, obtained an accuracy of 80.7% and triangle coverage of 93.0%.
m-LSTM [7] 2016	hidden-to-hidden transition of m-RNNs with the gating framework	This makes m-LSTM more easily parallelizable than these approaches. Additionally, it also shows that a large depth is not necessary to achieve competitive results on	These networks still carry on the disadvantages of LSTMs, they solve the problem of vanishing gradients but do not remove them		88% accuracy on wikitext-2 dataset.

	from LSTMs	character level language modelling. successful for t have achieved success at character level language modelling	completely. They are prone to overfitting and are affected by random weight initializations		
Dynamic Co-attention Networks ^[7] 2016	Co-attention encoder, LSTM	The DCN has the capability to estimate the start and end points of the answer span multiple times, each time conditioned on its previous estimates. By doing so, the model is able to explore local maxima corresponding to multiple plausible answers the co-attentive encoder is largely agnostic to long documents, and is able to focus on small sections of relevant text while ignoring the rest of the (potentially very long) document.		it was found that training the embeddings consistently led to overfitting and subpar performance, and hence only report results with fixed word embeddings were shown in the paper, this suggests that there was not a lot of improvement in terms of the training data.	On SQuAD, a single DCN model improves the previous state of the art from 71.0% F1 to 75.9%, while a DCN ensemble obtains 80.4% F1.

Table 3.1 Literature Review of Various MRC Techniques

After the extensive reading on the topic, it was decided to move forward with the Bi-directional Attention Flow (BiDAF) model, and to build a version of my own.

In the following section the steps of the Bidirectional Attention Flow neural network model and the concepts behind them will be explained in detail.

Chapter 4 Bi-directional Attention Flow Model

In this section, the steps of the Bidirectional Attention Flow model will be explained in detail, and along with that, the code snippets of each step will be provided too, so it is easy to see what is going on.

The inputs to the model are two strings: The Context and the corresponding Query. And the output to the model is the Answer Span, i.e., the token number of the beginning of the answer and that of the end of the answer.

The model's accuracy was evaluated on 2 metrics: Exact Match (EM) and F1 Score. Exact Match, as the name suggests looks at whether or not the model predicts the exact answer or not. F1 ^[9] score is a little more lenient in the calculation of accuracy. It looks at the weighted average of precision and recall. Precision is the fraction of true positives among all examples that were classified as positive ^[9]. Recall is the fraction of examples classified as positive among all the positive examples in the dataset ^[9].

The Ensemble Bi-directional Attention Flow Model has an EM accuracy of 73.3% and a F1 Score of 81.1% on the SQuAD dataset, and an accuracy of 76.9% on the CNN test dataset, and an accuracy of 79.6% on the DailyMail test dataset ^[1]. They had trained the model for 12 epochs, with a batch size of 60, on single Titan X GPU ^[1].

4.1 About Bidirectional Attention Flow

4.1.1 What is the Attention Mechanism?

In the most basic terms, attention mechanism can be explained as the action of selectively concentrating on certain fewer, more important parts of the data, while ignoring the rest of it ^[10].

Typically, in a Sequence to Sequence model (for tasks like translation), we see that the words of the input are passed one by one through the encoder, and the final output from the encoder is a fixed size vector that is supposed to contain the encoded information of the full text that was passed. This vector is then passed to a decoder and the task at hand is completed ^[10]. So, this means that the entire input, whatever its size is, is compressed into a fixed length vector. This won't work out for longer sentences, because the meaning might get lost, as there is too

much information to compressed into a fixed length vector, and also because every word is given equal importance. In these models, the intermediate outputs produced by the memory cells in the encoder, are all discarded, and only the final one is used.

This is where the Attention model ^[11], proposed by Bahdanau et al, comes in. The idea suggests that not only should all words be considered while creating the input vector, but also that certain words must be given higher importance with respect to the rest of the words ^[12]. This solves the problem for longer sentences/texts as now the important parts are highlighted and sent forward. Apart from this, in an Attention Model, the intermediate outputs produced by the encoder aren't discarded. Instead they are used to create context vectors which are later passed on to the decoder to generate the final output sequence ^[10, 12].

4.1.2 Attention Mechanisms in previous models

With the specific task of Machine Reading Comprehension, Attention Mechanisms were typically used to just extract the most important part of the Context that would be useful to answer the Query. This means that the attention was Uni-directional, where only the projections of the query on the context are obtained ^[1]. This means that only those words are given more precedence in the Context, that are important to the Query.

This is where Bi-directional Attention steps in.

4.1.3 What is Bi-Directional Attention?

It is the usage of the attention mechanism in both directions, query to context, and context to query. This helps the model understand which query words are most important to the context and which context words are most important to the query. Information from both sides is obtained, not just one like in the previous models ^[1].

4.1.4 Older models don't have Bidirectional Attention Flow – what makes this better?

In models that don't have any attention mechanism, for every word of the Query, every word of the Context would be looked at and they would be given equal importance. This is time consuming and not very efficient or useful. With attention mechanisms in the model, words

that are important to the Query in the Context are given more importance or precedence, making the task of finding the Answer easier and faster. This explains why Uni-directional attention makes the model better.

Apart from that, the fact that the attention in this model is bi-directional and not Uni-directional ^[1] is another step forward. Along with looking at the Context words that are important to the Query words, we are also looking at the Query words that are important to the Context words. This is what makes the model Bi-directional.

Last but not the least, the model doesn't use the attention model to summarize the context ^[1] to find out the most important parts to answer the query. Using this method could lead to loss of information and hence would not yield the best answer.

4.2 Tokenisation and Creation of Embedding Matrices

The first thing that is done when the input is received, is to make it ready for a computer to process. The two input strings are split by spaces into tokens, so that they are ready to be converted into embedding matrices.

We use two kinds of pretrained word embedding models here: GloVe (Global Vectors for Word Representation) and Fasttext.

GloVe ^[13] is an unsupervised learning algorithm for obtaining vector representations of words in high dimensional space. The dimension of the word vector represents the length of each word vector. Larger word vectors can store more information. Training of GloVe is performed on global word-word co-occurrence from a large corpus. For this project, 100-dimensional GloVe word vectors are used. So, for all words in the GloVe dictionary, we have a unique 100-dimensional vector representation. When GloVe encounters an Out-Of-Vocabulary token, it assigns a random vector to it. This can be a little harmful to the training of the model, and this is where Fasttext word embeddings step in.

Fasttext ^[14, 15] is another algorithm for obtaining word vectors, but it is slightly different from how the GloVe embeddings are computed and obtained. Here, instead of giving the entire word as input to the neural network, the words are broken down into several n-grams or sub-words, and these sub-words are fed to the neural network. This is helpful for when you see a word you

haven't seen before. The model can make the embeddings of this new word using the vectors of the trained n-grams and so newer words can have a better representation than a randomly initialised vector (as in GloVe embeddings). This model is also good for getting more accurate representations for words whose occurrence is comparatively rarer in the training data. Since these words are seen less frequently in the training data, the model may not be able to learn the best vector to represent it.

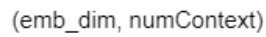
Here we are using 300-dimensional FastText word embeddings that have 1 million words in its dictionary and is trained on Wikipedia's news corpus, using sub-word information.

For each word, we will now have a GloVe embedding and a FastText embedding concatenated together to represent it. Each word will have an embedding dimension of 400.

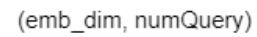
For the context and the query, we will get the embeddings for each token, and create two 2-dimensional matrices out of them. The context matrix will have dimension: $(400, numContext)$ where, $numContext$ is the number of tokens in the Context string, and 400 is the embedding dimension. Similarly for Query, we'll have a matrix of dimension $(400, numQuery)$.

The following image, Figure 4.1 shows an example of how this is done. Note that *emb_dim* in the image will be 400 in our case.

The island has two local newspapers, both of which are available on the internet. The St Helena Independent has been published since November 2005. The Sentinel Newspaper was introduced in 2012.



Since when has the St. Helena Independent been published?



(Token count starts from 0)

20

4.3 Highway Layer

A highway layer is very similar to a feed forward neural network. Here, only a fraction of the input is transformed by the network ^[16]. The rest of the information/data is passed through unchanged. We use the highway network here because we want our model to learn and adjust the contributions for each word, from the GloVe and Fasttext embeddings ^[17]. We want the model to assign more importance to the Fasttext embeddings if the word is an Out-Of-Vocabulary token, because Fasttext embeddings are better at dealing with words never seen before. If in case it is a word that exists in the vocabulary of both kinds of embeddings, we want the model to assign equal importance to the embeddings from both models.

The fractions of data being transformed and passed forward unchanged are maintained by the Transform and Carry gates respectively. The formula of the Highway layer looks like ^[16, 18]:

$$y(x) = T(x, W_T) * H(x, W_H) + (C(x, W_C)) * x \quad \text{Equation 4.1}$$

$T(x, W_T)$ represents the Transform Gate, where x represents the input, and W_T represents the trainable weight vector of this gate. $C(x, W_C)$ is the Carry Gate, where x is the input to the layer, and W_C is the trainable weight vector of this gate. The carry gate is set to:

$$1 - T(x, W_T) \quad \text{Equation 4.2}$$

So now the equation of the Highway layer ends up looking like:

$$y(x) = T(x, W_T) * H(x, W_H) + (1 - T(x, W_T)) * x \quad \text{Equation 4.3}$$

$H(x, W_H)$ and $T(x, W_T)$ are neural network outputs of Fully Connected or Dense Layers ^[16]. The transform gate uses Sigmoid activation so the output is between 0 and 1, and $H(x, W_H)$ gate used ReLU activation ^[16].

The input and output dimensions of the highway layer remains the same ^[16]. So, the output of the Context matrix passed through the Highway layer will have the same dimension, $(400, numContext)$. And similarly for the Query matrix, it'll have dimension $(400, numQuery)$.

The image below describes the flow of information through a highway layer, just like in equation (4.1).

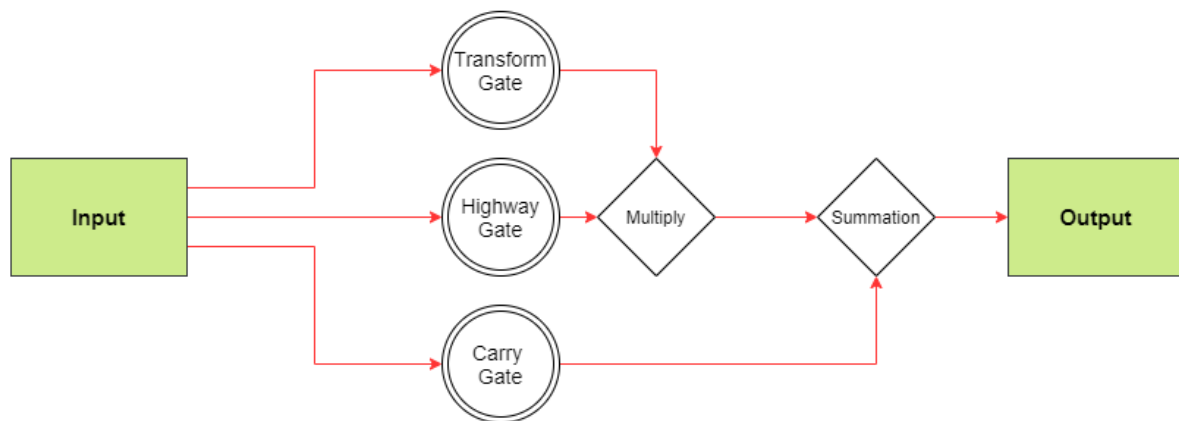


Figure 4.2 Information flow in Highway Layer

Below are the Code Snippets of the Highway Layer used in the model implementation.

```

2  """
3  Created on Thu Apr 15 18:44:37 2021
4
5  @author: Anusha
6  """
7
8  import tensorflow as tf
9  from tensorflow.keras.layers import Layer, Dense
10
11  class Highway(Layer):
12      """
13      ...  $y(x) = T(x) * H(x) + [1 - T(x)] * x$ 
14      ...
15      def __init__(self, name, transform_gate_bias=-1):
16          """
17          Parameters
18              input_shape : tuple
19                  input_shape should be of the format: (numExamples, num_words, emb_dim)
20              transform_gate_bias : int, optional
21                  Bias of the transform gate. The default is -1.
22          """
23          super(Highway, self).__init__()
24          self.transform_gate_bias = transform_gate_bias
25          self._name = name
26
27      def build(self, input_shape):
28          self.dim = input_shape[-1]
29
30          self.dense_1 = Dense(units=self.dim, bias_initializer=tf.keras.initializers.Constant(self.transform_gate_bias))
31          self.dense_2 = Dense(units=self.dim)
32

```

Figure 4.3 Highway Layer - Code Snippet 1

```

32
33     def call(self, x):
34         '''
35         Parameters
36             x : tf.Tensor: shape=(numExamples, num_words, emb_dim)
37                 input matrix
38         Returns
39             value : tf.Tensor: shape=(numExamples, num_words, emb_dim)
40                 output matrix
41         '''
42
43         #Transform Gate: Fully connected layer with sigmoid activation
44         T_x = self.dense_1(x)
45         T_x = tf.keras.activations.sigmoid(T_x)
46
47         #carry gate: C(x) = 1 - T(x)
48         C_x = tf.keras.layers.Lambda(lambda x: 1.0 - x, output_shape=(self.dim,))(T_x)
49
50         #H(x) Gate: Fully Connected layer with ReLU activation
51         H_x = self.dense_2(x)
52         H_x = tf.keras.activations.relu(H_x)
53
54         #Element wise multiplication:
55         transformed_gated = tf.math.multiply(T_x, H_x)
56         identity_gated = tf.math.multiply(C_x, x)
57
58         #Element wise addition:
59         value = tf.math.add(transformed_gated, identity_gated)
60
61         return value

```

Figure 4.4 Highway Layer - Code Snippet 2

4.4 Contextual Meaning Layer

Up until this point, each word had a vector representation, but the word vectors don't have knowledge of their surrounding words. There is no contextual meaning represented in the vectors ^[17]. Right now, if we see the word 'train', we don't know if it is the train we travel in or the activity of training to get better at something. So, to alter that, we pass our embeddings through a Bi-directional LSTM sequence.

LSTM ^[19, 20] stands for Long-Short Term Memory and it is a type of Recurrent Neural Network (RNN). Their speciality is to memorise dependencies (long-term and short-term) in the sequence, as the name suggests. This means that as each word is passed into the LSTM, the LSTM encodes and incorporates information of the words seen before it into the outputs of each word. Hence, the output embedding of each word contains contextual information of the words that came before it. To account for the words that come after a certain word, we make it a Bi-directional LSTM. This way, the meanings are encoded both ways.

The diagram below shows us how the information passed through a BiLSTM. The embedding for each word is passed through a unit or a memory cell, the memory cell gets information from the previous memory cell. It combines the information of the current word and whatever was obtained from the previous cell and gives an output for that word. It also passes some information to the next memory cell for the computations of the future words.

The Context and Query matrices are passed separately through the BiLSTM. This is because we want the Context and Query to capture their own meanings independent of each other, as they will be combined in future steps.

NOTE: *numWords* mentioned in the image means that it is *numContext* when the Context matrix is passed and *numQuery* when the Query matrix is passed.

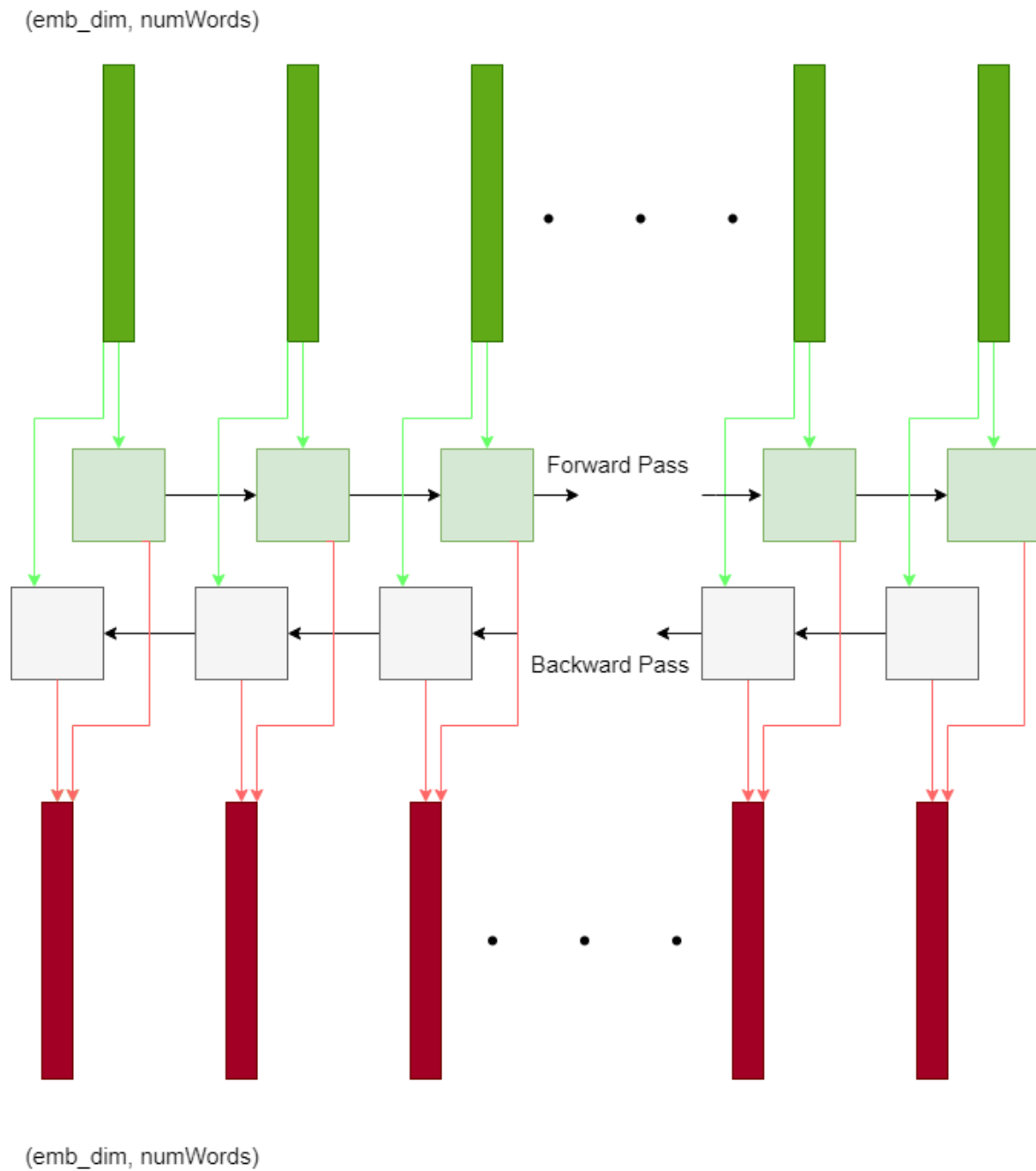


Figure 4.5 BiLSTM for Contextual Meaning of Context and Query Matrices

The code snippet in Figure 4.6 shows how the BiLSTM was implemented. The Bidirectional and LSTM methods are predefined in the TensorFlow library. Here, `return_sequences` is set to `True` because we want the output of each unit, and not just the final output of the BiLSTM.

```
#Contextual Meaning Layer
encoder_layer = Bidirectional(LSTM(emdim, recurrent_dropout=0, return_sequences=True), name='bidirectional_encoder')
encoded_question = encoder_layer(question_embedding)
encoded_passage = encoder_layer(passage_embedding)
```

Figure 4.6 BiLSTM Code Snippet

4.5 Similarity Matrix

This is the step where the contextual Context and Query matrices are combined to make a Similarity Matrix ^[1, 21]. The final matrix will represent the degree of similarity between each word of the Context matrix and each word of the Query Matrix. The dimensions of the Similarity Matrix are $(numContext, numQuery)$.

Here, the i^{th} row and j^{th} column of the similarity matrix represents the similarity between the i^{th} context word and the j^{th} query word. This is the first Attention step in our process, it will be used in the upcoming steps to calculate attention between the two sequences ^[17].

The Similarity matrix is calculated by the following rule ^[21]:

$$S(a, b) = W_T * [a ; b ; a * b] \quad \text{Equation 4.4}$$

Where, W_T is our trainable weight vector of length $3 * emb_dim$

$[;]$ represents vector concatenation (row-wise)

$*$ represents element-wise multiplication of two vectors

This single computation yields a scalar element, which represents the similarity between the words represented by vectors a and b .

The figure below represents the whole process of the creation of the Similarity Matrix.

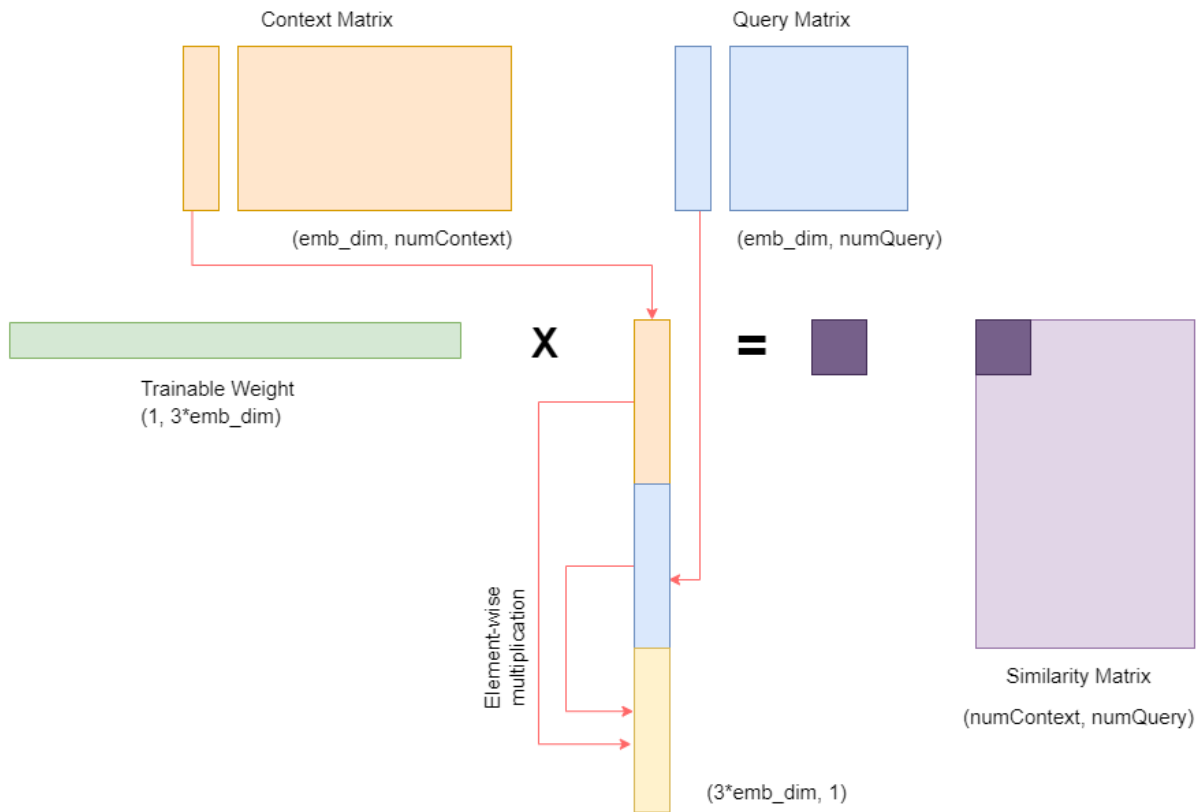


Figure 4.7 Creating Similarity Matrix from Query and Context Matrices

Figure 4.8, Figure 4.9 and Figure 4.10 show the code snippets that were used to implement the Similarity Matrix Layer in the Model.

```

2  """
3  Created on Sat Apr 10 17:36:36 2021
4
5  @author: Anusha
6  """
7  import tensorflow as tf
8  from tensorflow.keras.layers import Layer
9  from tensorflow.keras import backend as K
10
11  class SimilarityMatrix(Layer):
12      def __init__(self, name):
13          super(SimilarityMatrix, self).__init__()
14          self.name = name
15
16
17      def computeSimilarity(self, repeated_context_matrices, repeated_query_matrices):
18          """
19          repeated_context_matrices, repeated_query_matrices are of shape (None, numContext, numQuery, emb_dim)
20          returns a matrix of shape (None, numContext, numQuery)
21          """
22          #element wise mult, to give output of shape (None, numContext, numQuery, emb_dim)
23          element_wise_multiplication = tf.math.multiply(repeated_context_matrices, repeated_query_matrices)
24
25          #concat context, query and element-wise-mult vectors to give output of shape (None, numContext, numQuery, 3*emb_dim)
26          concat_matrices = tf.concat([repeated_context_matrices, repeated_query_matrices, element_wise_multiplication], axis=3)
27
28          #dot product of concatenated vector and the kernel(weights) to give output of shape (None, numContext, numQuery)
29          mult_with_weights = tf.tensordot(concat_matrices, self.kernel, axes=1)
30
31          #Add bias and return matrix
32          #shape=(None, numContext, numQuery)
33          return tf.keras.activations.linear(mult_with_weights + self.bias)
34

```

Figure 4.8 Similarity Matrix - Code Snippet 1

```

35
36 def build(self, input_shape):
37     '''
38     Parameters
39     -----
40     input_shape : (2, num_words, emb_dim)
41         input will be a list of matrices. [encoded_passage, encoded_question]
42         the matrices are of shape (None, num_words, emb_dim)
43     '''
44     #[0] indexes encoded passage matrix, [-1] indexes the embedding dimension
45     emb_dim = input_shape[0][-1]
46     weight_dim = emb_dim * 3
47
48     self.kernel = self.add_weight(name='similarity_weight', shape=(weight_dim), initializer='uniform', trainable=True)
49     self.bias = self.add_weight(name='similarity_bias', shape=(), initializer='ones', trainable=True)
50

```

Figure 4.9 Similarity Matrix - Code Snippet 2

```

51
52 def call(self, inputs):
53     #context_matrices: shape=[None, numContext, emb_dim]
54     #query_matrices: shape=[None, numQuery, emb_dim]
55     context_matrices, query_matrices = inputs
56
57     num_context_words = tf.shape(context_matrices)[1] #numContext
58     num_query_words = tf.shape(query_matrices)[1] #numQuery
59
60     context_dim_repeat = tf.concatenate([[1, 1], [num_query_words], [1]], 0) #[1, 1, numQuery, 1]
61     query_dim_repeat = tf.concatenate([[1], [num_context_words], [1, 1]], 0) #[1, numContext, 1, 1]
62
63     exp_context = tf.expand_dims(context_matrices, axis=2) #[None, numContext, 1, emb_dim]
64     exp_query = tf.expand_dims(query_matrices, axis=1) #[None, 1, numQuery, emb_dim]
65
66     '''
67     repeated_context_matrices: shape(None, numContext, numQuery, emb_dim)
68         each word's embedding is repeated numQuery times
69     repeated_query_matrices: shape(None, numContext, numQuery, emb_dim)
70         the full query vector is repeated numContext times
71     '''
72     repeated_context_matrices = tf.tile(exp_context, context_dim_repeat)
73     repeated_query_matrices = tf.tile(exp_query, query_dim_repeat)
74
75     #get Similarity Matrix
76     simMatrix = self.computeSimilarity(repeated_context_matrices, repeated_query_matrices)
77
78     return simMatrix
79

```

Figure 4.10 Similarity Matrix - Code Snippet 3

4.6 Context to Query Attention

This next step is where the Context to Query Attention ^[1, 21] is calculated. The aim of this layer is to find which Query words are most relevant to which Context words. The input to this step is the Similarity Matrix computed in the previous layer, and the contextual meaning Query matrix. The steps to calculate Context to Query Attention are as mentioned below:

- (1) The Similarity Matrix is put through row-wise SoftMax. (each row now adds up to 1)
- (2) Each row is taken one by one
- (3) Scalar multiplication of elements from step (2) row with corresponding column vectors of the contextual meaning Query Matrix.
- (4) All these multiplied vectors are summed up together to make a vector of dimension $(emb_dim, 1)$. This vector corresponds to the C2Q attention vector for that corresponding Context word.
- (5) Step (2) to (4) is repeated for each row taken in step (2)

The output of this step is a matrix of dimension $(emb_dim, numContext)$.

The image below shows this process in pictorial representation.

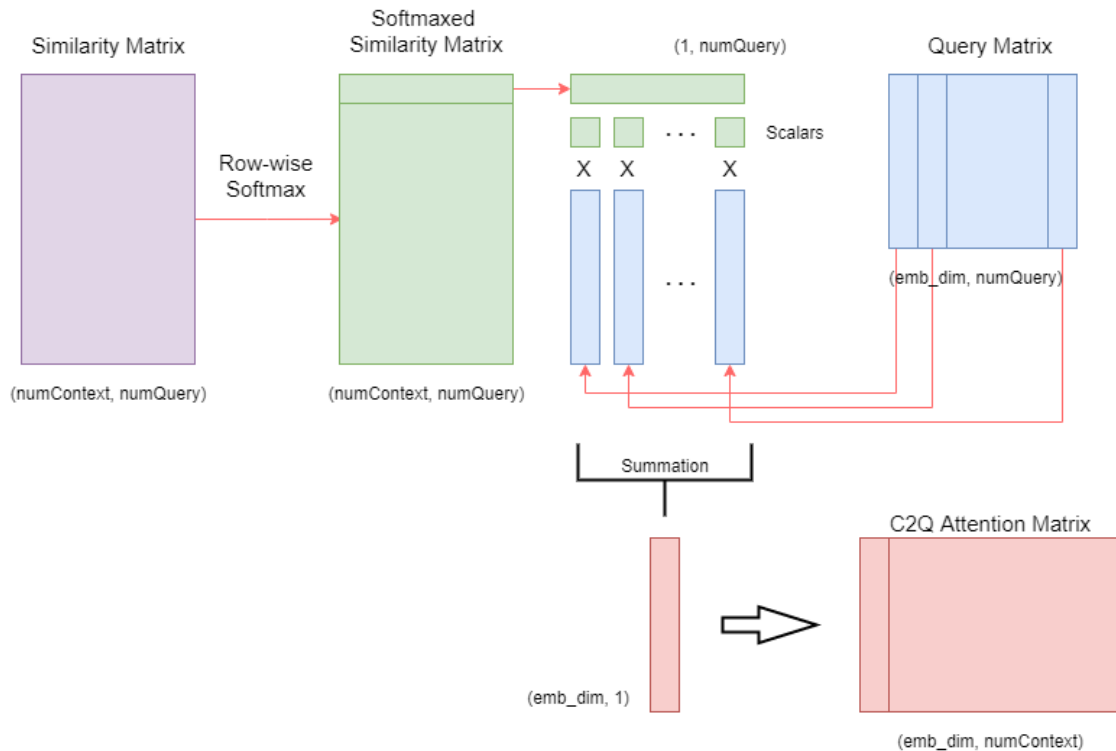


Figure 4.11 Context to Query Attention Computation

The Code Snippets are shown in Figure 4.12, that were used to implement this layer in the model.

```

3   Created on Mon Apr 12 12:39:08 2021
4
5   @author: Anusha
6   """
7   import tensorflow as tf
8   from tensorflow.keras.layers import Layer
9
10  class ContextToQueryAttention(Layer):
11      def __init__(self, name):
12          super(ContextToQueryAttention, self).__init__()
13          self._name = name
14
15      def call(self, inputs):
16          '''
17          Parameters
18          -----
19          inputs : [similarity_matrix, encoded_query]
20                  similarity_matrix of shape=(None, numContext, numQuery)
21                  encoded_query of shape=(None, numQuery, emb_dim)
22
23          Returns
24          -----
25          C2QAttention : shape=(None, numContext, emb_dim)
26                  Context to Query Attention
27          '''
28
29          similarity_matrix, encoded_query = inputs
30
31          #taking row-wise softmax of the similarity matrix
32          softmaxed_similarity_matrix = tf.nn.softmax(similarity_matrix, axis=-1)
33
34          #expanding dim to make of shape=(None, 1, numQuery, emb_dim)
35          encoded_query = tf.expand_dims(encoded_query, axis=1)
36
37          weighted_multiplication = tf.expand_dims(softmaxed_similarity_matrix, axis=-1) * encoded_query
38          # shape=(None, numContext, embdim)
39          sum_of_vectors = tf.sum(weighted_multiplication, axis=-2)
40
41          return sum_of_vectors

```

Figure 4.12 Context to Query Attention - Code Snippet 1

4.7 Query to Context Attention

This is the second kind of attention that is calculated in this process, which is what gives this model its name. Here, we want to find which context word is most similar to which query word to help in answering the question. The inputs to this layer are the Similarity matrix and the contextual meaning Context Matrix. The output is a matrix of dimension $(emb_dim, numContext)$.

The steps to calculate the Query to Context ^[1, 21] attention are as follows:

- (1) The Similarity matrix is taken row-wise, and the maximum element from each row is obtained. A column matrix of dimension $(numContext, 1)$ is obtained.

- (2) This matrix is put through SoftMax. (So now the full vector adds up to 1)
- (3) Scalar multiplication of elements from matrix in step (2) and corresponding column vectors of the contextual meaning Context Matrix.
- (4) All the resultant column vectors are summed up, to get a matrix of dimension $(emb_dim, 1)$
- (5) The matrix obtained in step (4) is duplicated $numContext$ times and concatenated to give a final Q2C Attention matrix of dimension $(emb_dim, numContext)$.

The image in Figure 4.13 shows the process of computing the Query to Context Attention Matrix in pictorial representation.

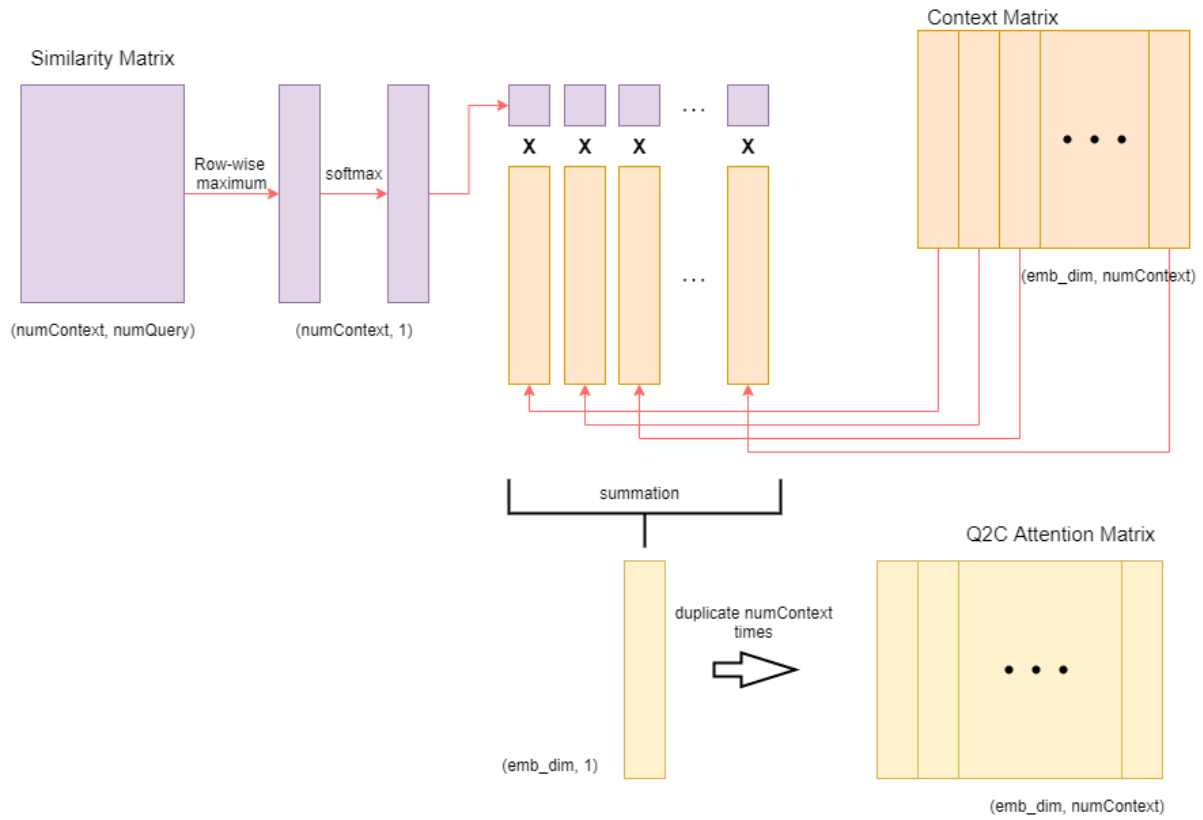


Figure 4.13 Query to Context Attention Computation

The Code Snippet are attached below that were used to implement this layer in the model.

```
2  """
3  Created on Mon Apr 12 18:21:54 2021
4
5  @author: Anusha
6  """
7
8  import tensorflow as tf
9  from tensorflow.keras.layers import Layer
10
11
12  class QueryToContextAttention(Layer):
13      def __init__(self, name):
14          super(QueryToContextAttention, self).__init__()
15          self._name = name
16
17      def call(self, inputs):
18          '''
19          Parameters
20          -----
21          inputs :[similarity_matrix, encoded_context]
22                  similarity_matrix of shape=(None, numContext, numQuery)
23                  encoded_query of shape=(None, numContext, emb_dim)
24
25          Returns
26          -----
27          C2QAttention : shape=(None, numContext, emb_dim)
28                  Context to Query Attention
29          '''
30          similarity_matrix, encoded_context = inputs
31
32          #get max value from each row
33          row_wise_max_similarity = tf.max(similarity_matrix, axis=-1)
34
35          #softmax of resultant (None, numContext,) matrix
36          softmaxed_row_wise_max_similarity_matrix = tf.nn.softmax(row_wise_max_similarity, axis=-1)
37
38          #shape=(None, numContext, 1)
39          expanded_softmax = tf.expand_dims(softmaxed_row_wise_max_similarity_matrix, axis=-1)
40
```

Figure 4.14 Query to Context Attention - Code Snippet 1

```
40
41      #shape=(None, numContext, emb_dim)
42      mult = expanded_softmax * encoded_context
43
44      #weighted sum of softmaxed_max_similarity_matrix and context matrix:
45      #shape=(None, emb_dim, 1)
46      weighted_sum = tf.sum(mult, -2)
47
48      #expand dims: shape=(None, 1, emb_dim)
49      expanded_weighted_sum = tf.expand_dims(weighted_sum, axis=1)
50
51      #need to repeat it numContext times:
52      num_of_repeat = encoded_context.shape[1] #numContext
53
54      #repeat numContext times: shape=(None, numContext, emb_dim)
55      t = tf.tile(expanded_weighted_sum, [1, num_of_repeat, 1])
56
57      return t
58
```

Figure 4.15 Query to Context Attention - Code Snippet 2

4.8 Mega Merge

The next layer of the model combines the contextual meaning Context Matrix, Query to Context Attention Matrix and the Context to Query Matrix to form 1 final Merged Context matrix. The dimensions of the Merged Context Matrix are $(4 * emb_dim, numContext)$. This merging takes place as follows ^[1, 21]:

$$Merge(a, b, c) = [a ; b ; a * b ; a * c] \quad \text{Equation 4.5}$$

Where, a , b and c are corresponding column vectors from the Context Matrix, Context to Query matrix and Query to Context matrix respectively.

$[;]$ is the concatenation operation (row-wise)

$*$ is the element-wise multiplication operation between two vectors

The mega-merge step is shown in the figure below

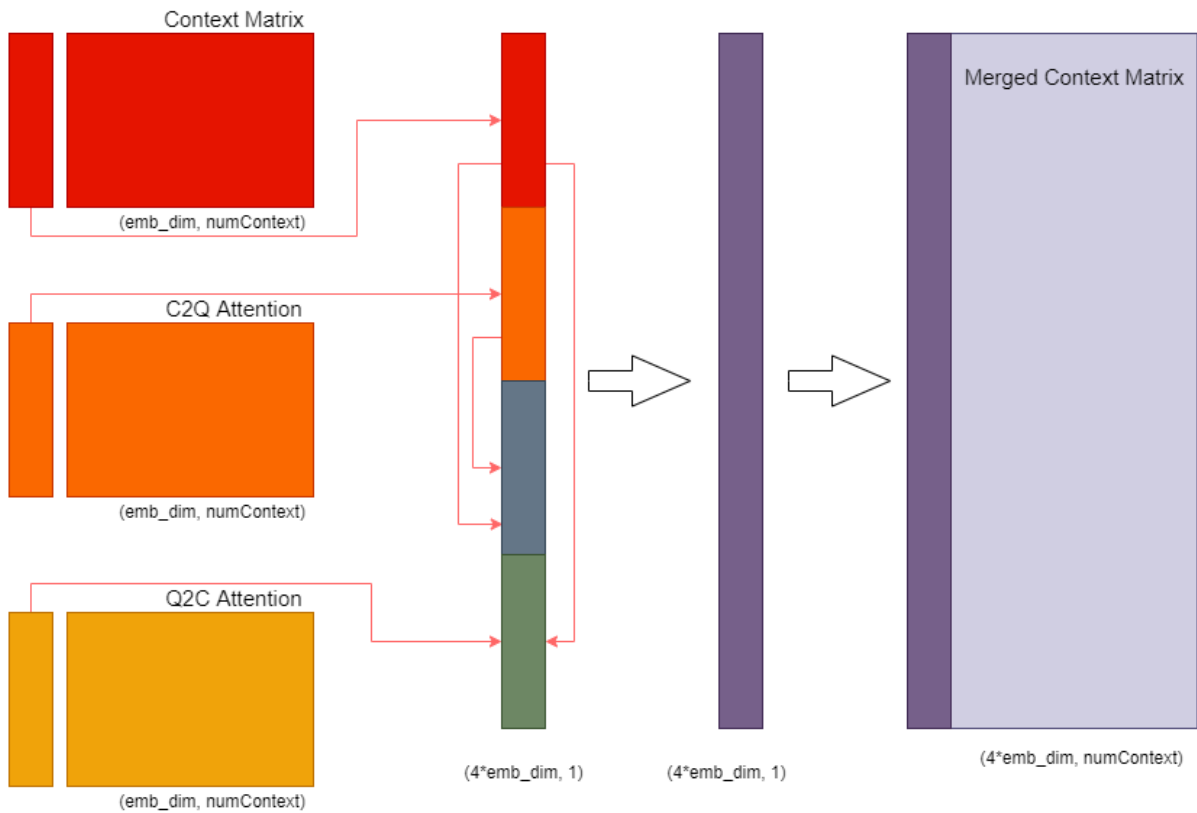


Figure 4.16 Mega - Merge operation to obtain the Merged Context

The code snippet of this layer is attached below.

```
2 """
3 Created on Mon Apr 12 19:07:54 2021
4
5 @author: Anusha
6 """
7 import tensorflow as tf
8 from tensorflow.keras.layers import Layer
9
10 class MegaMerge(Layer):
11     def __init__(self, name):
12         super(MegaMerge, self).__init__()
13         self._name = name
14
15     def call(self, inputs):
16         """
17         Merging the Context Matrix(with encoded contextual meanings) with the C2Q and Q2C Attention matrices.
18         """
19         #dimensions of all 3: shape=(None, numContext, emb_dim)
20         encoded_context, c2q_attention, q2c_attention = inputs
21
22         element_wise_multiply1 = encoded_context * c2q_attention
23         element_wise_multiply2 = encoded_context * q2c_attention
24
25         concatenated_tensor = tf.concat([encoded_context, c2q_attention, element_wise_multiply1, element_wise_multiply2], axis=-1)
26
27         #output dim: shape=(None, numContext, 4*emb_dim)
28         return concatenated_tensor
29
```

Figure 4.17 Mega Merge - Code Snippet 1

4.9 Modelling Layer

This is the layer before which we get our final answer – The Modelling Layer^[1, 22]. The purpose of the modelling layer is so that the interactions between the context words are captured, keeping in mind the Query words (or in other words, keeping in mind the Query that needs to be answered). This is different from the contextual meaning layer seen previously, because that layer captures the interaction among the Context and the interaction among the Query words^[1], independent of each other.

The Merged Context, that was created in the mega-merge step, is now passed through a Bi-directional LSTM, to give us an output matrix M1 which is used ahead to obtain the Start Span of the Answer. The matrix M1 is passed through another Bi-directional LSTM to give us matrix M2. This matrix M2 is used ahead to find the End Span of the Answer. Both matrices M1 and M2 have information about the Context and Query embedded in them, which will help us find the final Answer Spans.

Each column in both matrices M1 and M2, representing each word of the Context, now contains information of that word with respect to all the words in the Context and the Query.

The output matrices of this layer, M1 and M2 have dimensions (*emb_dim*, *numContext*).

The modelling layer is illustrated in the image that follows.

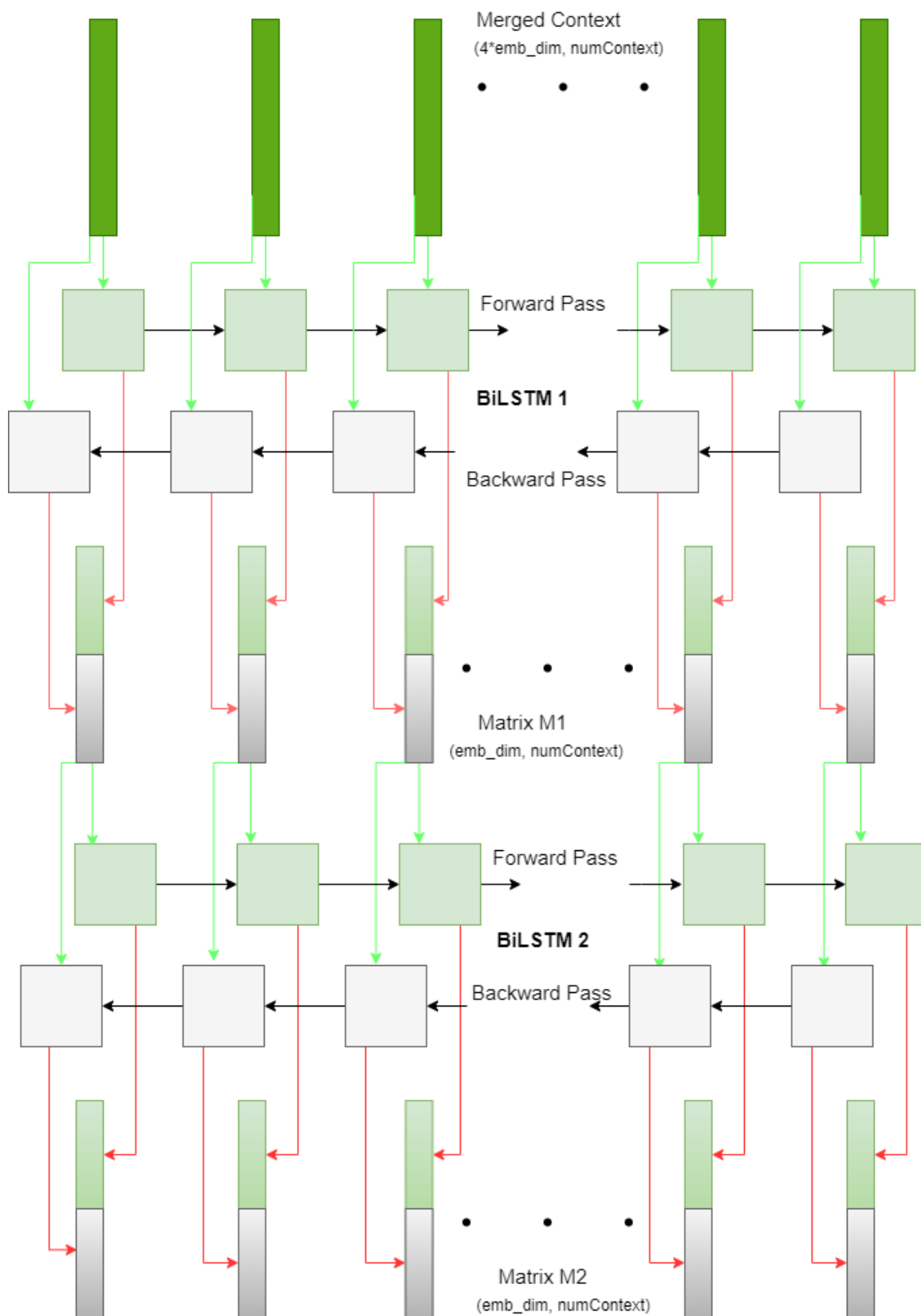


Figure 4.18 Modelling Layer using BiLSTMs

The code snippet for the Modelling layer is provided below. Bidirectional and LSTM Classes are predefined classes in the TensorFlow package, so have been used directly. The `return_sequences` variable is set to `True`, because we want the BiLSTM to return an output for each word we give it, not just one final output vector.

```
#Modelling Layer
modeled_passage = merged_context

hidden_layer1 = Bidirectional(LSTM(emdim, recurrent_dropout=0, return_sequences=True), name='modelling_layer_1')
M1 = hidden_layer1(modeled_passage)

hidden_layer2 = Bidirectional(LSTM(emdim, recurrent_dropout=0, return_sequences=True), name='modelling_layer_2')
M2 = hidden_layer2(M1)
```

Figure 4.19 Modelling Layer - Code Snippet 1

4.10 Output Layer and Answer Span Probabilities

This is the final layer of the model. This is the layer that converts our vectors into final probabilities that tell us which word is most likely to be the Start of the Answer and which word is most likely to be the End of the Answer.

As we have seen in the previous layer, each column vector in both matrices M1 and M2, (which represent each word of the Context), now contains information of that word with respect to all the words in the Context and the Query ^[1].

At the end of this layer, we get two vectors that represent the probabilities of the respective words being the Start of the Answer Span, or the End of the Answer Span, depending on the vector being looked at.

To obtain the Start Span of our Answer ^[1, 22], we take the Merged Context Matrix and the M1 Matrix. We concatenate those matrices and multiple it with a trainable weight vector, to get a vector of dimension $(1, numContext)$. This vector is passed through SoftMax, so that all the elements of this vector add up to 1, like probabilities do. Each element now corresponds to the probability of that word being the Start of the Answer Span.

Similarly, the probabilities corresponding to the End of the Answer Span ^[1, 22] are also found. The inputs are the Merged Context Matrix and the M2 Matrix. They are concatenated and multiplied with a trainable weight vector. We get a vector of dimension $(1, numContext)$.

This vector is passed through Softmax. Now, each element of this vector corresponds to the probability of that word being the End of the Answer Span.

The dimensions of the concatenated matrix in both cases are $(5 * emb_dim, numContext)$. The dimensions of the trainable weight vectors are $(1, 5 * emb_dim)$. The concatenated matrix had dimension $5 * emb_dim$ because the two matrices used to create this matrix have dimensions $(4 * emb_dim, numContext)$ and $(emb_dim, numContext)$. These two matrices are joined by placing one on top of the other. So, the dimension of the resultant matrix has size $5 * emb_dim$.

Now we have two vectors that tell us the start and end position of the Answer to our Query.

The pictorial visualization of this process is shown in the image below.

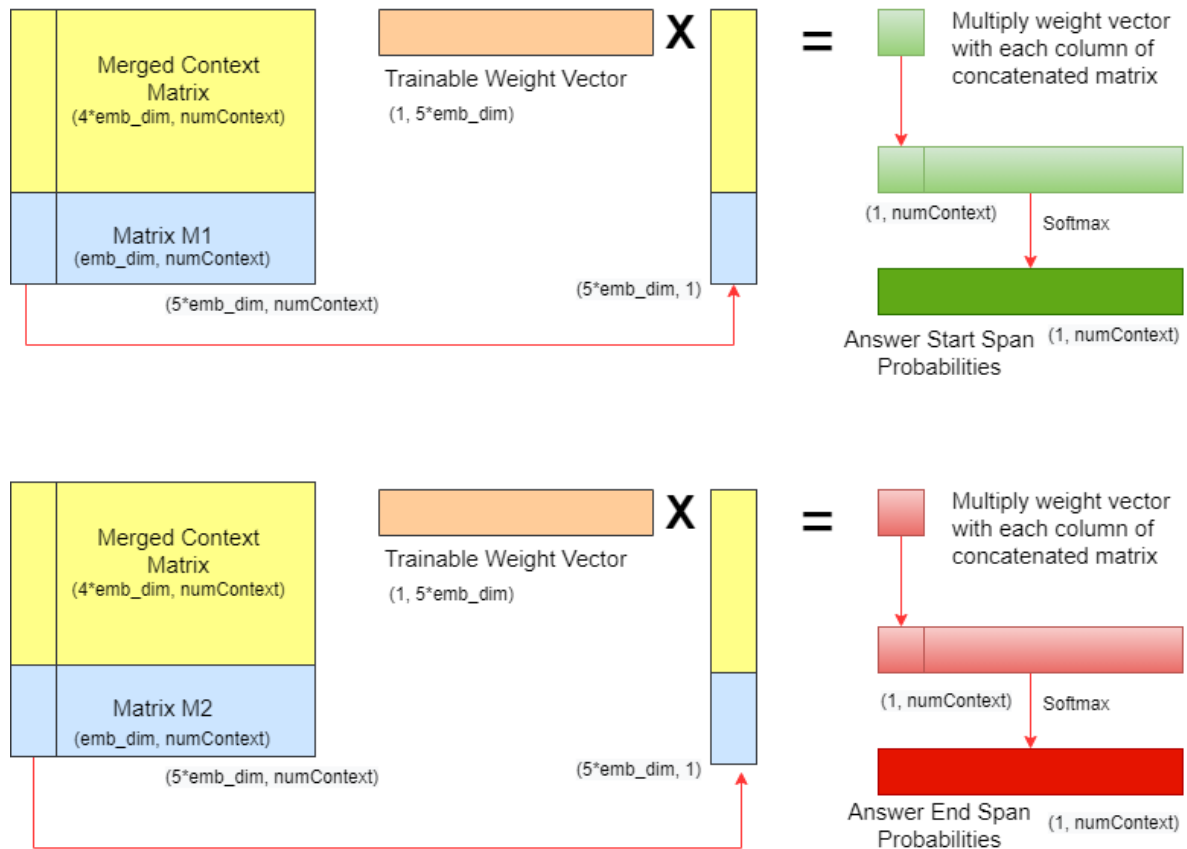


Figure 4.20 Finding Start and End Span of Answer

The code snippets of obtaining the start span are attached below.

```

2  """
3  Created on Tue Apr 13 12:28:30 2021
4
5  @author: Anusha
6  """
7
8  import tensorflow as tf
9  from tensorflow.keras.layers import Layer
10
11  class SpanBegin(Layer):
12      #p1 = softmax(WT(p1) * [megamerge:output_bilSTM_1])
13      def __init__(self, name):
14          super(SpanBegin, self).__init__()
15          self._name = name
16
17      def build(self, input_shape):
18          #should add up to 10*emb_dim
19          word_vector_dim = input_shape[0][-1] + input_shape[1][-1]
20          self.kernel = self.add_weight(name='span_begin_weight', shape=(word_vector_dim, 1), initializer='uniform', trainable=True)
21
22      def call(self, inputs):
23          merged_context, M1 = inputs
24
25          #concat to make of shape=(None, numContext, 10*emb_dim)
26          concat_tensor = tf.concat([merged_context, M1], axis = -1)
27
28          #MatrixMultiplication(concat_tensor, weights):shapes: (numContext, 10*emb_dim)*(10*emb_dim, 1)
29          #output shape = (None, numContext, 1)
30          mult = tf.matmul(concat_tensor, self.kernel)
31          #squeeze to make of shape=(None, 1, numContext)
32          mult = tf.squeeze(mult, axis=-1)
33          #expand dims to make of shape=(None, 1, numContext)
34          span_begin = tf.expand_dims(mult, axis=1)
35          #apply softmax to the vector:
36          span_begin = tf.nn.softmax(span_begin, axis=-1)
37          #squeeze to make of shape=(None, numContext)
38          span_begin = tf.squeeze(span_begin, axis=-2)
39
40      return span_begin

```

Figure 4.21 Start Span - Code Snippet 1

The code snippets of obtaining the end span are attached below.

```

2  """
3  Created on Wed Apr 14 13:21:25 2021
4
5  @author: Anusha
6  """
7
8  import tensorflow as tf
9  from tensorflow.keras.layers import Layer
10
11  class SpanEnd(Layer):
12      #p2 = softmax(WT(p2) * [megamerge:output_bilSTM_2])
13      def __init__(self, name):
14          super(SpanEnd, self).__init__()
15          self._name = name
16
17      def build(self, input_shape):
18          #should add up to 10*emb_dim
19          word_vector_dim = input_shape[0][-1] + input_shape[1][-1]
20          self.kernel = self.add_weight(name='span_end_weight', shape=(word_vector_dim, 1), initializer='uniform', trainable=True)
21
22      def call(self, inputs):
23          #None in shape is the numnber of examples passed
24          merged_context, M2 = inputs
25
26          #concat to make of shape=(None, numContext, 10*emb_dim)
27          concat_tensor = tf.concat([merged_context, M2], axis = -1)
28
29          #MatrixMultiplication(concat_tensor, weights):shapes: (None, numContext, 10*emb_dim)*(None, 10*emb_dim, 1)
30          #output shape = (None, numContext, 1)
31          mult = tf.matmul(concat_tensor, self.kernel)
32          #squeeze to make of shape=(None, numContext)
33          mult = tf.squeeze(mult, axis=-1)
34          #expand dims to make of shape=(None, 1, numContext)
35          span_end = tf.expand_dims(mult, axis=1)
36          #apply softmax to the vector:
37          span_end = tf.keras.activations.softmax(span_end, axis=-1)
38          #squeeze to make of shape=(None, numContext)
39          span_end = tf.squeeze(span_end, axis=-2)
40
41      return span_end

```

Figure 4.22 End Span - Code Snippet 1

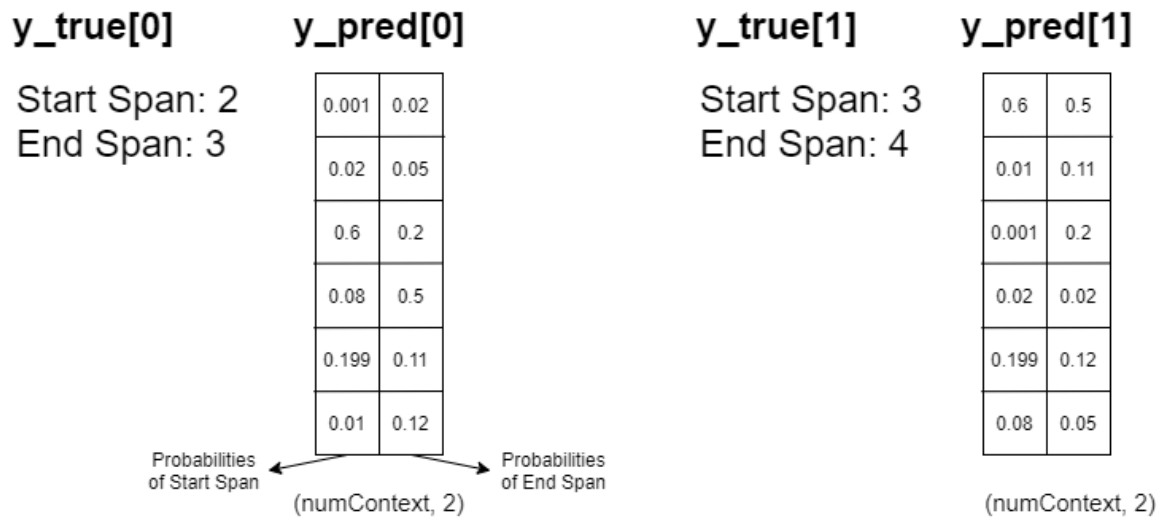
4.11 Training Functions

For the training process of the model, custom Accuracy and Loss functions need to be defined. The model is compiled using the Adam Optimizer that is predefined in the TensorFlow library. This optimizer is an extension of the stochastic gradient descent algorithm. They are used to update the weights in the network/model in an iterative way, based on the training data that is passed through the model. Adam is a popular algorithm used in neural networks, especially in Natural Language Processing problems, as it gives good results quickly. In this algorithm, each parameter has its own learning rate as compared to other optimizer algorithms which have only one learning rate for all the parameters in the model ^[23].

4.11.1 Accuracy Function

Accuracy here is seen as the sum of probabilities of the start and end spans. The higher the probabilities of the words that are the actual span, higher the accuracy comes out to be. The Accuracy function ^[24] looks at the probability of the actual span indices, takes those probabilities and takes their average. This is done for all examples in the batch and their average is taken again. This is what is returned as the accuracy metric.

The accuracy is calculated in an example in the diagram below. The batch has 2 data points. Here the number of context words is 6.



Accuracy Calculation

For example 0

Start Probability = 0.6
End Probability = 0.5

Average of Probs = $(0.6 + 0.5)/2$
= $(1.1)/2$
= 0.55

For example 1

Start Probability = 0.02
End Probability = 0.12

Average of Probs = $(0.02 + 0.12)/2$
= $(0.14)/2$
= 0.07

Accuracy returned = $\text{mean}(\text{AverageOfProbs}[0], \text{AverageOfProbs}[1])$
= $1/2(0.55 + 0.07)$
= 0.31

Figure 4.23 Accuracy Calculation

The code snippet is attached below.

```

2  """
3  Created on Thu Apr 15 16:37:03 2021
4
5  @author: Anusha
6  """
7  import tensorflow as tf
8
9  def accuracy(y_true, y_pred):
10
11      def calculate_accuracy(true_pred):
12          ytrue, pred_start, pred_end = true_pred
13
14          # incase that the start or end span is more than 200, then it won't be able to index in the predicted vectors.
15          # it would lead to an error. Hence for precaution we take the minimum with 199, since 200 is the max len of Context.
16          startProb = pred_start[tf.math.minimum(tf.cast(ytrue[0], dtype=tf.int32), tf.cast(199, dtype=tf.int32))]
17          endProb = pred_end[tf.math.minimum(tf.cast(ytrue[1], dtype=tf.int32), tf.cast(199, dtype=tf.int32))]
18          return (startProb + endProb) / 2.0
19
20      pred_start_span = y_pred[:, :, 0]
21      pred_end_span = y_pred[:, :, 1]
22
23      #applying the function on each example (y_true, y_pred)
24      acc = tf.map_fn(calculate_accuracy, (y_true, pred_start_span, pred_end_span), fn_output_signature=tf.float32)
25
26      #returning the mean of accuracy:
27      return tf.mean(acc, axis=0)
28

```

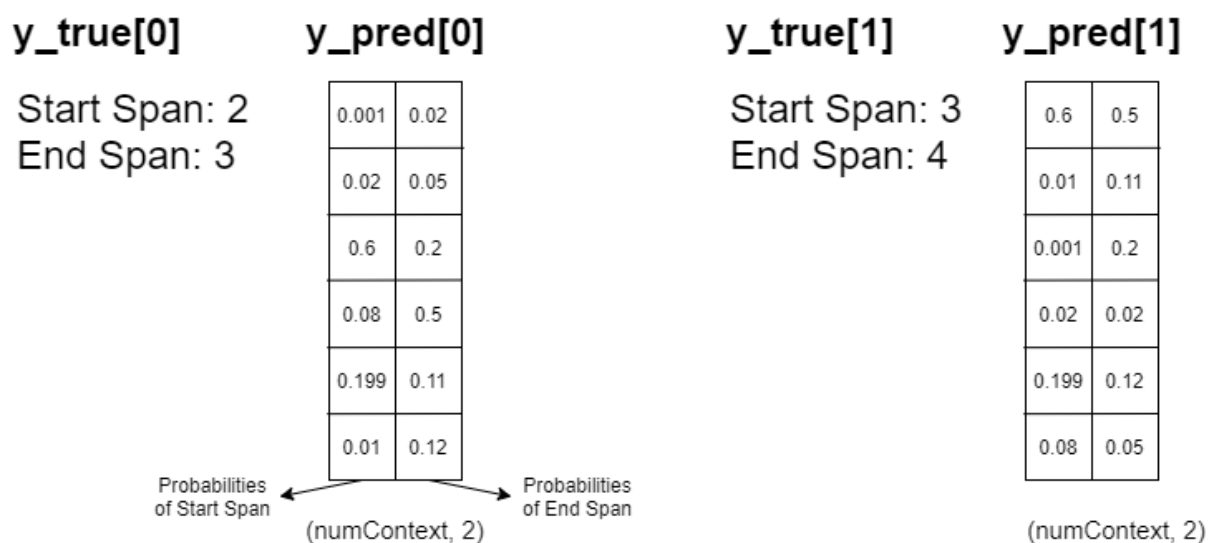
Figure 4.24 Accuracy Function - Code Snippet 1

4.11.2 Loss Function

The loss function ^[24] used here is the negative average log loss. This is calculated as follows: The actual span is the index where the probability value in the output spans given should be maximum. For each example, the log values of the index of the actual span are calculated, and the negative mean of that value is returned for each batch, which is how the loss is calculated.

Probability vectors of start words and end words are being passed to the loss function, instead of the predicted spans, because when the error is back propagated, it is required to go to the entire probability vector and make necessary changes to it. The whole vector needs to be altered in such a way that the actual answer span indices have the highest probability in the whole vector. So, when an answer needs to be predicted, the words with the highest probability can be picked out.

The loss is calculated in an example in Figure 4.25. The same example is used as used in the example of calculating the accuracy. The code snippet follows in Figure 4.26.



Loss Calculation

For example 0

Start Probability = 0.6
End Probability = 0.5

Sum Of Logs of Probs = $\log(0.6) + \log(0.5)$
= $-0.2218 - 0.3010$
= -0.5228

For example 1

Start Probability = 0.02
End Probability = 0.12

Sum Of Logs of Probs = $\log(0.02) + \log(0.12)$
= $-1.6989 - 0.9208$
= -2.6197

loss returned = $-\text{mean}(\text{sumOfLogsOfProbs}[0], \text{sumOfLogsOfProbs}[1])$
= $-1/2(-0.5228 - 2.6197)$
= 1.57125

Figure 4.25 Loss Calculation

```

2  """
3  Created on Thu Apr 15 11:00:01 2021
4
5  @author: Anusha
6
7  reference: https://towardsdatascience.com/custom-loss-function-in-tensorflow-2-0-d8fa35405e4e
8  """
9
10 import tensorflow as tf
11
12 def NegativeAvgLogError(y_true, y_pred):
13
14     def sum_of_log_probabilities(true_pred):
15         ytrue, pred_start, pred_end = true_pred
16
17         # incase that the start or end span is more than 200, then it won't be able to index in the predicted vectors.
18         # it would lead to an error. Hence for precaution we take the minimum with 199, since 200 is the max len of Context.
19         startProb = pred_start[tf.math.minimum( tf.cast(ytrue[0], dtype=tf.int32), tf.cast(199, dtype=tf.int32)) ]
20         endProb = pred_end[tf.math.minimum( tf.cast(ytrue[1], dtype=tf.int32), tf.cast(199, dtype=tf.int32)) ]
21
22         #returning the sum of the log of the start and end probabilities.
23         return tf.log(startProb) + tf.log(endProb)
24
25     #y_true = K.squeeze(y_true, axis=1)
26     pred_start_span = y_pred[:, :, 0]
27     pred_end_span = y_pred[:, :, 1]
28
29     #Calling the function for each example:
30     batch_probability_sum = tf.map_fn(sum_of_log_probabilities, (y_true, pred_start_span, pred_end_span), fn_output_signature=tf.float32)
31
32     #returning the negative of mean of all log probabilities:
33     return -tf.mean(batch_probability_sum, axis=0)
34

```

Figure 4.26 Loss Function - Code Snippet 1

4.12 Bidirectional Attention Flow Model using TensorFlow

TensorFlow is a free open-source library for implementing machine learning models, developed by Google Brain ^[25]. It can train and run deep neural networks for a variety of tasks. It has available modules for implementing for different kinds of machine learning models easily. If those are not enough, Tensorflow also gives you the power to create your custom models, layers, loss and accuracy functions, among other things ^[26].

Each layer in this model that was made up till now was a custom layer created on TensorFlow. They were created by inheriting the base Layer Class that is present in the tensorflow.keras.layers module ^[27, 28]. The weights of the layer are defined in the call method of each class ^[27]. They will be updated automatically according to our loss function during the training process ^[24, 29].

Now that all the layers are created and in order, they can finally be put together to be made into a model on TensorFlow. We create the model using the Functional API, that helps us define custom inputs and outputs ^[30] for the model.

Attached is the code snippet of the definition of the model.

```

#Input Layer
passage_input = Input(shape=(max_passage_length, emdim), dtype='float32', name="passage_input")
question_input = Input(shape=(max_query_length, emdim), dtype='float32', name="question_input")

#Highway Layer
highwayLayer = Highway(name='highway_1')
question_embedding = highwayLayer(question_input)
passage_embedding = highwayLayer(passage_input)

#Contextual Meaning Layer
encoder_layer = Bidirectional(LSTM(emdim, recurrent_dropout=0, return_sequences=True), name='bidirectional_encoder')
encoded_question = encoder_layer(question_embedding)
encoded_passage = encoder_layer(passage_embedding)

#Similarity Matrix
sim_layer = SimilarityMatrix(name='sim')
similarity_matrix = sim_layer([encoded_passage, encoded_question])

#C2Q Attention
c2q_attention = ContextToQueryAttention(name='c2q_attention')([similarity_matrix, encoded_question])

#Q2C Attention
q2c_attention = QueryToContextAttention(name='q2c_attention')([similarity_matrix, encoded_passage])

#Mega Merge
merged_context = MegaMerge(name='merged_context')([encoded_passage, c2q_attention, q2c_attention])

```

Figure 4.27 Bidirectional Attention Flow Model - Code Snippet 1

```

#Modelling Layer
modeled_passage = merged_context

hidden_layer1 = Bidirectional(LSTM(emdim, recurrent_dropout=0, return_sequences=True), name='modelling_layer_1')
M1 = hidden_layer1(modeled_passage)

hidden_layer2 = Bidirectional(LSTM(emdim, recurrent_dropout=0, return_sequences=True), name='modelling_layer_2')
M2 = hidden_layer2(M1)

#Span Begin
span_begin_probabilities = SpanBegin(name='span_begin')([merged_context, M1])

#Span End
span_end_probabilities = SpanEnd(name='span_end')([merged_context, M2])

#Combine Outputs
output = CombinedOutputs(name='combined_outputs')([span_begin_probabilities, span_end_probabilities])

#Defining the Model
bidaf_model = tf.keras.models.Model([passage_input, question_input], output)

```

Figure 4.28 Bidirectional Attention Flow Model - Code Snippet 2

Below attached is the summary of the model, and an image to show the number of trainable parameters in the model.

Layer (type)	Output Shape	Param #	Connected to
passage_input (InputLayer)	[(None, 209, 400)]	0	
question_input (InputLayer)	[(None, 16, 400)]	0	
highway_1 (Highway)	multiple	320800	question_input[0][0] passage_input[0][0]
bidirectional_encoder (Bidirect	multiple	2563200	highway_1[0][0] highway_1[1][0]
sim (SimilarityMatrix)	(None, 209, 16)	2401	bidirectional_encoder[1][0] bidirectional_encoder[0][0]
c2q_attention (ContextToQueryAt	(None, 209, 800)	0	sim[0][0] bidirectional_encoder[0][0]
q2c_attention (QueryToContextAt	(None, 209, 800)	0	sim[0][0] bidirectional_encoder[1][0]
merged_context (MegaMerge)	(None, 209, 3200)	0	bidirectional_encoder[1][0] c2q_attention[0][0] q2c_attention[0][0]
modelling_layer_1 (Bidirectiona	(None, 209, 800)	11523200	merged_context[0][0]
modelling_layer_2 (Bidirectiona	(None, 209, 800)	3843200	modelling_layer_1[0][0]
span_begin (SpanBegin)	(None, 209)	4000	merged_context[0][0] modelling_layer_1[0][0]
span_end (SpanEnd)	(None, 209)	4000	merged_context[0][0] modelling_layer_2[0][0]
combined_outputs (CombinedOutpu	(None, 209, 2)	0	span_begin[0][0] span_end[0][0]

Figure 4.29 Model Summary

```

=====
Total params: 18,260,801
Trainable params: 18,260,801
Non-trainable params: 0
=====

```

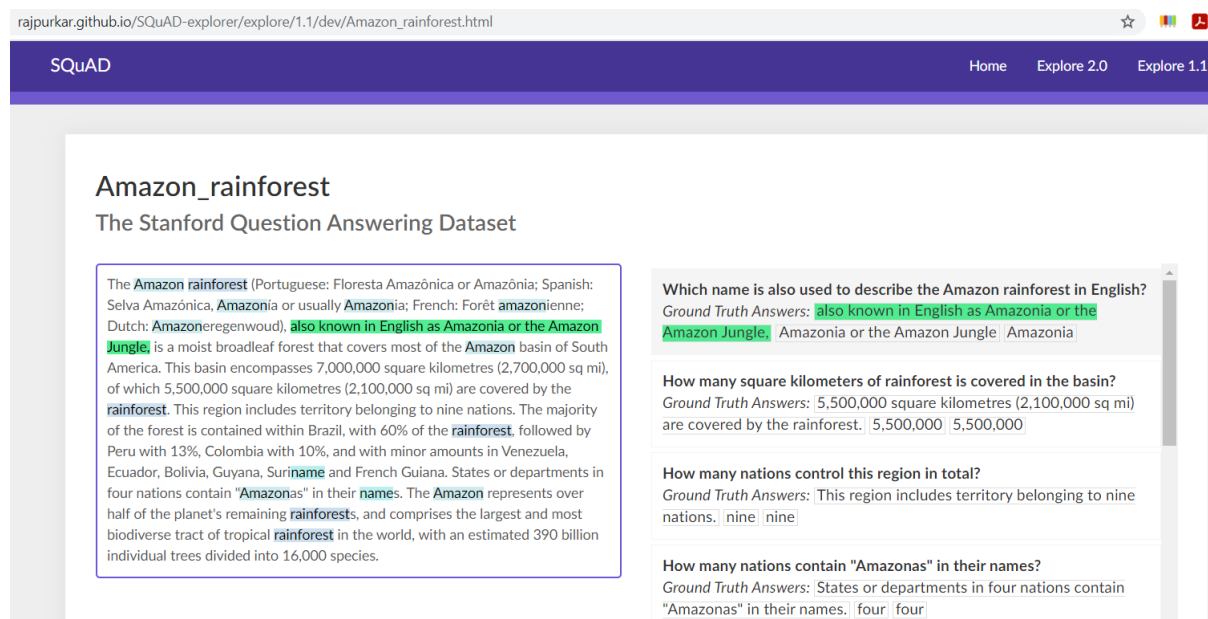
Figure 4.30 Number of trainable parameters in the model

Chapter 5 Requirements

5.1 Datasets

The dataset of Contexts, Queries and Answers used is the SQuAD ^[31, 32] Dataset. SQuAD stands for Stanford Question Answering Dataset. Version 1.1 has been used for this project. The Dataset was created by thousands of crowd workers, who asked questions based on a set of Wikipedia articles and answered them too. The answer is either a word or a span of words from the context, nothing that needs to be inferred from the passage (That is part of a different type of Machine Reading Comprehension - Abstractive). Either that, or the question is unanswerable. An example from SQuAD's website is attached in Figure 5.1 below. The part of the passage highlighted in green is the most accurate answer span, and the parts highlighted in blue are other satisfactory answers that are also correct and acceptable.

The dataset has been separated into 4 files: one for the Context, one for the Query, one for the Answers, and one for the Answer Span, which is the start and end token number of the Answer in the Context.



The screenshot displays the SQuAD dataset interface. The top navigation bar includes the SQuAD logo and links for Home, Explore 2.0, and Explore 1.1. The main content area is titled "Amazon_rainforest" and "The Stanford Question Answering Dataset".

Context Passage:

The Amazon rainforest (Portuguese: Floresta Amazônica or Amazônia; Spanish: Selva Amazónica, **Amazonia** or usually **Amazonia**; French: Forêt amazonienne; Dutch: Amazoneregenwoud), **also known in English as Amazonia or the Amazon Jungle**, is a moist broadleaf forest that covers most of the Amazon basin of South America. This basin encompasses 7,000,000 square kilometres (2,700,000 sq mi), of which 5,500,000 square kilometres (2,100,000 sq mi) are covered by the **rainforest**. This region includes territory belonging to nine nations. The majority of the forest is contained within Brazil, with 60% of the **rainforest**, followed by Peru with 13%, Colombia with 10%, and with minor amounts in Venezuela, Ecuador, Bolivia, Guyana, Suriname and French Guiana. States or departments in four nations contain "Amazonas" in their **names**. The Amazon represents over half of the planet's remaining **rainforests**, and comprises the largest and most biodiverse tract of tropical **rainforest** in the world, with an estimated 390 billion individual trees divided into 16,000 species.

Questions and Ground Truth Answers:

- Which name is also used to describe the Amazon rainforest in English?**
Ground Truth Answers: **also known in English as Amazonia or the Amazon Jungle**, Amazonia or the Amazon Jungle, Amazonia
- How many square kilometers of rainforest is covered in the basin?**
Ground Truth Answers: 5,500,000 square kilometres (2,100,000 sq mi) are covered by the rainforest. 5,500,000 5,500,000
- How many nations control this region in total?**
Ground Truth Answers: This region includes territory belonging to nine nations. nine nine
- How many nations contain "Amazonas" in their names?**
Ground Truth Answers: States or departments in four nations contain "Amazonas" in their names. four four

Figure 5.1 Example from SQuAD Dataset ^[31]

The Contexts are all of varying lengths, which could lead to problems while passing through the model, as only fixed length matrices are to be passed. To fix this, the maximum length of the passages is set to 200, if a passage is longer than 200 tokens, then that corresponding data

point is omitted from the training and validation sets. If the length is less than 200, then the vector will be padded to make it of length 200. The same thing is done for the queries too. If the length of a query is more than 50, then that corresponding example is omitted from the training/validation sets. This is done as a pre-processing step. The data is put into a CSV file, and then imported as a pandas dataset. The data is iterated through row wise and the lengths of the Query and Context are checked. A filter is created and examples where either the Context length is greater than 200 or the Query length is greater than 50, are marked and then removed. The full dataset had about 86 thousand examples in the training set and 10 thousand examples in the validation set. After the filtering, we are left with about 75 thousand examples in the training set and 9 thousand examples in the validation set.

Each example shown to the model has an upper token limit of 250. 200 from the context and 50 from the query. There are about 75 thousand examples. This gives us about 18.75 million tokens (as an upper limit) that the model sees while training.

The Figures shows a part of the modified dataset. The first 5 examples are shown – The Contexts, Queries, and the Answer Spans. The corresponding lines in each file together make up a data point or example. The 1st line of the Question File asks a question about the context in the 1st line of the Context File, and the 1st line of the Answer File answers that question, and the 1st line of the Answer Span File gives the location of the answer in the context.

```

1 during the perestroika era , the law on the status of the estonian language was adopted in january 1989 . the
collapse of the soviet union led to the restoration of republic of estonia 's independence . estonian went back to
being the only state language in estonia which in practice meant that use of estonian was promoted while the use of
russian was discouraged .
2 kathmandu is a center for art in nepal , displaying the work of contemporary artists in the country and also
collections of historical artists . patan in particular is an ancient city noted for its fine arts and crafts . art
in kathmandu is vibrant , demonstrating a fusion of traditionalism and modern art , derived from a great number of
national , asian , and global influences . nepali art is commonly divided into two areas : the idealistic
traditional painting known as paubhas in nepal and perhaps more commonly known as thangkas in tibet , closely
linked to the country 's religious history and on the other hand the contemporary western-style painting ,
including nature-based compositions or abstract artwork based on tantric elements and social themes of which
painters in nepal are well noted for . internationally , the british-based charity , the kathmandu contemporary art
centre is involved with promoting arts in kathmandu .
3 farming had been a traditional occupation for centuries , although it became less dominant in the 20th century with
the advent of tourism . grazing and pasture land are limited because of the steep and rocky topography of the alps
. in mid-june cows are moved to the highest pastures close to the snowline , where they are watched by herdsman who
stay in the high altitudes often living in stone huts or wooden barns during the summers . villagers celebrate the
day the cows are herded up to the pastures and again when they return in mid-september . the alpinschluss or
dÄsalpes ( " coming down from the alps " ) is celebrated by decorating the cows with garlands and enormous
cowbells while the farmers dress in traditional costumes .
4 temporal measurement , chronometry , takes two distinct period forms : the calendar , a mathematical tool for
organizing intervals of time , and the clock , a physical mechanism that counts the passage of time . in day-to-day
life , the clock is consulted for periods less than a day whereas the calendar is consulted for periods longer than
a day . increasingly , personal electronic devices display both calendars and clocks simultaneously . the number (
as on a clock dial or calendar ) that marks the occurrence of a specified event as to hour or date is obtained by
counting from a fiducial epochâ€a central reference point .
5 sir john call argued the advantages of norfolk island in that it was uninhabited and that new zealand flax grew
there . in 1786 the british government included norfolk island as an auxiliary settlement , as proposed by john
call , in its plan for colonisation of new south wales . the decision to settle norfolk island was taken due to
empress catherine ii of russia 's decision to restrict sales of hemp . practically all the hemp and flax required
by the royal navy for cordage and sailcloth was imported from russia .

```

Figure 5.2 Modified Dataset - Context File

```
1 what historical event once again freed estonia ?  
2 what do the tibetans call traditional idealistic paintings ?  
3 when are cows moved to the highest pastures close to the snowline ?  
4 what is a mathematical tool used for organizing intervals of time ?  
5 who proposed the idea to include norfolk island as a british auxiliary settlement in 1786 ?
```

Figure 5.3 Modified Dataset - Question File

```
1 collapse of the soviet union  
2 thangkas  
3 mid-june  
4 the calendar  
5 john call
```

Figure 5.4 Modified Dataset - Answer File

```
1 21 25  
2 93 93  
3 42 42  
4 11 12  
5 38 39
```

Figure 5.5 Modified Dataset - Answer Span File

Once this filtering of the data points was done, the next step was to prepare them into batches. Our dataset is huge, as mentioned above. We cannot possibly load the whole data into memory to train the model. If we do that, our RAM will get flooded and there would be no space to perform the computations of the model, and the system would hang. To avoid this situation, Data Generators have been used. They help in loading only a batch of data at a time, this way our memory isn't flooded.

When the data generator is obtaining the examples to be passed to the model, it also creates the embedding matrices and then passed that as input to the model.

Attached is the code snippet used to generate the data as batches.

```

7 from keras.utils import Sequence
8 import os
9 import numpy as np
10
11 class BatchGenerator(Sequence):
12     def __init__(self, ContextVectors, QueryVectors, batch_size, emdim, maxContextLen, maxQueryLen, shuffle, gen_type):
13         self.ContextVectors = ContextVectors
14         self.QueryVectors = QueryVectors
15
16         self.maxContextLen = maxContextLen
17         self.maxQueryLen = maxQueryLen
18
19         self.context_file = '/content/gdrive/MyDrive/BiDAF/Data/squad/' + gen_type + '.context'
20         self.question_file = '/content/gdrive/MyDrive/BiDAF/Data/squad/' + gen_type + '.question'
21         self.span_file = '/content/gdrive/MyDrive/BiDAF/Data/squad/' + gen_type + '.span'
22
23         self.batch_size = batch_size
24         i = 0
25         with open(self.span_file, 'r', encoding='utf-8') as f:
26
27             for i, _ in enumerate(f):
28                 pass
29         self.num_of_batches = (i + 1) // self.batch_size
30         self.indices = np.arange(i + 1)
31         self.shuffle = shuffle
32
33     def __len__(self):
34         '''Returns the number of batches per epoch'''
35         return self.num_of_batches

```

Figure 5.6 Data Generator - Code Snippet 1

Here, *gen_type* is the name of the file that we want to get the data from. The location is set to the location in Google Drive, as the training was taking place on Google Colab. ContextVectors and QueryVectors have the concatenation of the word embedding dictionary. They are defined separately because while initialization, the padding length needs to be defined, and the padding length of the context and query are different here.

The number of batched is calculated by dividing the number of examples in the span file, with the batch size. The shuffle variable can be set to True or False. If set to True, it will shuffle the data at the end of every epoch.


```

36
37 def __getitem__(self, index):
38     '''Generate one batch of data'''
39     start_index = (index * self.batch_size)
40     end_index = ((index + 1) * self.batch_size)
41
42     inds = self.indices[start_index:end_index]
43
44     contexts = []
45     with open(self.context_file, 'r', encoding='utf-8') as cf:
46         for i, line in enumerate(cf, start=0):
47             if i in inds:
48                 contexts.append(line.split(' '))
49
50     questions = []
51     with open(self.question_file, 'r', encoding='utf-8') as qf:
52         for i, line in enumerate(qf, start=0):
53             if i in inds:
54                 questions.append(line.split(' '))
55
56     answer_spans = []
57     with open(self.span_file, 'r', encoding='utf-8') as sf:
58         for i, line in enumerate(sf, start=0):
59             if i in inds:
60                 answer_spans.append(line.split(' '))
61
62     context_batch = self.ContextVectors.query(contexts, pad_to_length=self.maxContextLen)
63     question_batch = self.QueryVectors.query(questions, pad_to_length=self.maxQueryLen)
64
65     if self.maxContextLen is not None:
66         span_batch = np.expand_dims(np.array(answer_spans, dtype='float32'), axis=1).clip(0, self.maxContextLen - 1)
67     else:
68         span_batch = np.expand_dims(np.array(answer_spans, dtype='float32'), axis=1)
69     span_batch = np.array(answer_spans, dtype='float32')
70     return [context_batch, question_batch], [span_batch]
71
72 def on_epoch_end(self):
73     if self.shuffle:
74         np.random.shuffle(self.indices)

```

Figure 5.7 Data Generator - Code Snippet 2

5.2 Technical

The technical (hardware/software) requirements for this project were:

- Python v3.8
- Jupyter Notebook and Spyder to write Python Code locally and check working of code and flow
- TensorFlow v2.3
- Numpy v1.19.5
- Pymagnitude v0.1.143 on Python for easily loading the word vector embeddings
- Google Colab with GPU (NVIDIA Tesla K80) enabled for training
- Other libraries used with python:
 - os for reading the files of data
 - tensorflow.keras.layers for predefined and custom layers
 - tensorflow.keras.models for using the functional API
 - tensorflow.keras.backend for matrix transformations
 - keras.utils.Sequence for the Data Generators
 - pandas for filtering the dataset

Chapter 6 Results: Training and Testing

The model was trained using different training strategies on Google Colab, using the GPU.

First it was trained for 1 epoch and the batch size was 16. Hence there were 4725 batches. The training loss came out to be 9.6661 and the training accuracy was 0.89%. The validation loss was 8.9407 and the validation accuracy was 1.35%, on the validation dataset. This training process took about 3 hours to complete.

```
[18] bidaf_model.compile(optimizer=tf.keras.optimizers.Adadelta(learning_rate=0.01),  
                        loss = NegativeAvgLogError,  
                        metrics=[accuracy])  
  
[19] history = bidaf_model.fit(train_generator, epochs=1, validation_data=validation_generator)  
  
4725/4725 [=====] - 11738s 2s/step - loss: 9.6661 - accuracy: 0.0089 - val_loss: 8.9407 - val_accuracy: 0.0135
```

Figure 6.1 Training for 1 epoch

This 1 epoch trained model was saved using the model.save method. Using this method, the training can be resumed from where the model was saved. Model checkpoints were also added to the model to save the progress of the training after every epoch. This was done in case the GPU access is lost in the middle, some progress can be saved.

Next, the 1 epoch trained model was loaded and it was trained for 3 additional epochs, and the batch size remained the same, 16. The model trained for about 9 hours, and the final model was saved again.

[33], to cross check with the predicted answers. Testing analysis with a large number of examples would be done after the training is complete for 12 epochs and the model has been optimized.

The table storing these results is attached below.

Context	Query	Answer by Google BERT pretrained model	Predicted Answer (1 epoch)	Predicted Answer (4 epoch)	e.g. seen during Training?
The following is the hierarchy of the Canadian armed forces. It begins at the top with the most senior-ranking personnel and works its way into lower organizations.	What is the hierarchy of the Canadian armed forces?	It begins at the top with the most senior - ranking personnel	Canadian armed forces. It begins at the top with the most senior-ranking personnel and works its way into lower organizations.	Canadian armed forces	Yes
There is a mango tree planted outside the house.	What is planted outside the house?	mango tree	There is a mango	mango tree planted outside the house	No
Anusha is a final year computer science student. She is enrolled at Shiv Nadar university. The university is located in greater Noida. her father's name is Dilip.	What is Anusha a student of?	Shi	Anusha is a final year computer science student. She is enrolled at Shiv Nadar	Shiv Nadar university. The university is located in greater Noida	No
He is going to Mumbai.	Where is he going?	Mumbai	He	Mumbai	No
I will drink tea today.	What will I drink today?	tea	I	drink tea	No

The effect of Luther's intervention was immediate. After the sixth sermon, the Wittenberg jurist Jerome Schurf wrote to the elector: oh, what joy has Dr. Martin's return spread among us! His words, through divine mercy, are bringing back every day misguided people into the way of the truth.	Who wrote a resounding letter to the elector praising Luther?	Jerome Schurf	Wittenberg jurist Jerome Schurf	sixth sermon, the Wittenberg jurist Jerome Schurf	Yes
The BBC television department headed by Jana Bennett was absorbed into a new, much larger group; BBC vision, in late 2006. The new group was part of larger restructuring within the BBC with the onset of new media outlets and technology.	What caused the move of BBC television into the BBC vision unit?	new media outlets and technology	Jana	Jana Bennett was absorbed into a new, much larger group; BBC vision, in late 2006	Yes
Valencian is classified as a western dialect, along with the north western varieties spoken in western Catalonia (provinces of Lleida and the western half of Tarragona). The various forms of Catalan and Valencian are mutually intelligible (ranging from 90% to 95%)	What forms are mutually intelligible?	Catalan and Valencian	Catalonia (provinces of Lleida and the western half of Tarragona)	90% to 95%	Yes

Table 6.2 Comparing Test Results

Chapter 7 Future Scope of the Project

The first point for future work in this project would be to make the accuracy of the model better. As we have seen in the previous section, moving from the first epoch to the second, the accuracy improves greatly. Training the model for longer can help in increasing the accuracy.

That can be done by using a more powerful system to train the model, using some compute resources on Google Cloud or AWS or some other cloud providers, would be helpful. The model needs to be trained for a greater number of epochs, and ideally on the full data, which includes examples with contexts longer than 200 tokens, and not just the filtered dataset used in this project.

Experimentation with different batch sizes and number of epochs and the size of the validation sets, could also provide some more insights into the working of the model.

Next, would be to review the implementation in more detail than what was done now, and find out places where the model can be optimized by making changes and see how the accuracy improves with each change. Another scope of improvement could be using ideas and inspiration from other popular Machine Reading Comprehension models and incorporating them into this model, to see how those ideas work in improving the accuracy.

Testing the trained models on a large number of examples needs to be done too, and analysing the answers obtained from multiple models. This is to be done after training is done for 12 epochs and the code has been optimized.

References

- [1] Minjoon Seo et al, University of Washington, Allen Institute for Artificial Intelligence, “*Bi-directional Attention Flow for Machine Comprehension*”, Published as a conference paper at ICLR 2017, 1611.01603v6[cs.CL] 21 Jun 2018, <https://arxiv.org/abs/1611.01603>
- [2] Natural Language Computing Group, Microsoft Research Asia, “*R-Net: Machine Reading Comprehension with Self-Matching Networks*”, presented in ACL 2017, <https://www.microsoft.com/en-us/research/wp-content/uploads/2017/05/r-net.pdf>
- [3] Martin Mirakyan et al, “*Challenges of reproducing R-NET neural network using Keras*”, Aug 2017, <https://yerevann.github.io/2017/08/25/challenges-of-reproducing-r-net-neural-network-using-keras/>
- [4] Adams Wei Yu et al, “*QANet: Combining Local Convolution with Global Self Attention for Reading Comprehension*”, 2018, <https://arxiv.org/abs/1804.09541>, Published as full paper in ICLR 2018
- [5] Min Sang Kim, “*Implementing QANet (Question Answering Network) with CNNs and self-attentions*”, Apr 2018, <https://towardsdatascience.com/implementing-question-answering-networks-with-cnns-5ae5f08e312b>
- [6] Yelong Shen et al, Microsoft Research, “*Reasonet: Learning to Stop Reading in Machine Comprehension*”, 2017, <https://arxiv.org/abs/1609.05284>, Published in KDD 2017
- [7] Xin Zhang et al, “*Machine Reading Comprehension: A Literature Review*”, 2019, <https://arxiv.org/abs/1907.01686>
- [8] Jacob Devlin et al, “*Bert: Pre-training of deep bidirectional transformers for language understanding*”, arXiv preprint arXiv:1810.04805 (2018), <https://arxiv.org/abs/1810.04805>
- [9] Thomas Wood, “*What is the F-score?*”, <https://deepai.org/machine-learning-glossary-and-terms/f-score#:~:text=What%20is%20the%20F%2Dscore,positive'%20or%20'negative'>.
- [10] Harshall Lamba, “*Intuitive Understanding of Attention Mechanism in Deep Learning*”, 2019, <https://towardsdatascience.com/intuitive-understanding-of-attention-mechanism-in-deep-learning-6c9482aecf4f>

- [11] Dzmitry Bahdanau et al, “*Neural Machine Translation by Jointly Learning to Align and Translate*”, 2015, <https://arxiv.org/abs/1409.0473>, Accepted at ICLR 2015 as oral presentation
- [12] American Express, “*A Comprehensive Guide to Attention Mechanism in Deep Learning for Everyone*”, 2019, <https://www.analyticsvidhya.com/blog/2019/11/comprehensive-guide-attention-mechanism-deep-learning/>
- [13] Jeffrey Pennington, et al, “*GloVe: Global Vectors for Word Representations*”, 2014, <https://nlp.stanford.edu/projects/glove/>
- [14] Kung-Hsiang, Huang (Steeve), “*Word2Vec and Fasttext Word Embedding with Genism*”, 2018, <https://towardsdatascience.com/word-embedding-with-word2vec-and-fasttext-a209c1d3e12c>
- [15] “*English Word Vectors – Fasttext*”, 2016, <https://fasttext.cc/docs/en/english-vectors.html>, <https://fasttext.cc/>
- [16] Praveen Narayanan, “*A note on Highway Layers*”, 2018, <https://pravn.wordpress.com/2018/02/28/a-note-on-highway-layers/#:~:text=Highway%20layers%20are%20an%20extension,through%20very%20deep%20network%20stacks>.
- [17] Meraldo Antonio, “*Word Embedding, Character Embedding and Contextual Embedding in BiDAF – An Illustrated Guide*”, Aug 2019, <https://towardsdatascience.com/the-definitive-guide-to-bidaf-part-2-word-embedding-character-embedding-and-contextual-c151fc4f05bb>
- [18] “*Highway Layer*”, <https://paperswithcode.com/method/highway-layer#>
- [19] Michael Phi, “*Illustrated Guide to LSTM’s and GRU’s: A step by step explanation*”, Sep 2018, <https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21>
- [20] Jason Brownlee, “*A Gentle Introduction to Long-Short Term Memory Networks by the Experts*”, May 2017, <https://machinelearningmastery.com/gentle-introduction-long-short-term-memory-networks-experts/>
- [21] Meraldo Antonio, “*Attention Mechanism in Seq2Seq and BiDAF – An Illustrated Guide*”, Aug 2019, <https://towardsdatascience.com/the-definitive-guide-to-bidaf-part-3-attention-92352bbdcb07>

- [22] Meraldo, Antonio, “*Modelling and Output Layers in BiDAF – An Illustrated Guide with Minions*”, Sep 2019, <https://towardsdatascience.com/modeling-and-output-layers-in-bidaf-an-illustrated-guide-with-minions-f2e101a10d83>
- [23] Jason Brownlee, “*Gentle Introduction to the Adam Optimization Algorithm for Deep Learning*”, Jul 2017, <https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/#:~:text=Adam%20is%20a%20replacement%20optimization,sparse%20gradients%20on%20noisy%20problems.>
- [24] Parikh Kadam, “*bidaf-keras*”, May 2019, <https://github.com/ParikhKadam/bidaf-keras>
- [25] “*Why TensorFlow*”, <https://www.tensorflow.org/>
- [26] Serdar Yegulalp, “*What is TensorFlow? The machine learning library explained*”, Jun 2019, <https://www.infoworld.com/article/3278008/what-is-tensorflow-the-machine-learning-library-explained.html>
- [27] Dr Kevin Webster, Imperial College London, “*Customising your models with TensorFlow 2*”, <https://www.coursera.org/learn/customising-models-tensorflow2>
- [28] Sadrach Pierre, “*Using Classes for Machine Learning*”, Feb 2020, <https://towardsdatascience.com/using-classes-for-machine-learning-2ed6c0713305>
- [29] Allen AI, “*bi-att-flow*”, <https://github.com/allenai/bi-att-flow>
- [30] Praphul Singh, “*Bi-directional Attention Flow Model for Machine Comprehension*”, Feb 2020, <https://towardsdatascience.com/bidirectional-attention-flow-model-for-machine-comprehension-d533f5600007>
- [31] Rajpurkar, et al, “*Explore SQuAD*”, 2016, https://rajpurkar.github.io/SQuAD-explorer/explore/1.1/dev/Amazon_rainforest.html
- [32] “*squad*”, <https://www.tensorflow.org/datasets/catalog/squad>
- [33] Chris McCormick, “*Question Answering with a Fine-Tuned BERT*”, Mar 2020, <https://mccormickml.com/2020/03/10/question-answering-with-a-fine-tuned-BERT/#:~:text=Super%20Bowl%2050.-,BERT%20Input%20Format,question%20from%20the%20reference%20text.>

Other References

- [34] Meraldo Antonio, “*An Illustrated Guide to Bi-directional Attention Flow (BiDAF)*”, Aug 2019, <https://towardsdatascience.com/the-definitive-guide-to-bi-directional-attention-flow-d0e96e9e666b>
- [35] <https://towardsdatascience.com/custom-loss-function-in-tensorflow-2-0-d8fa35405e4e>
- [36] Dr Kevin Webster, Imperial College London, “*Getting Started with TensorFlow 2*”, <https://www.coursera.org/learn/getting-started-with-tensor-flow2>
- [37] Mohammed Innat, “*Model Subclassing and Custom Training Loops from Scratch in TensorFlow 2*”, Nov 2020, <https://towardsdatascience.com/model-sub-classing-and-custom-training-loop-from-scratch-in-tensorflow-2-cc1d4f10fb4e>
- [38] Adrian Rosebrock, “*3 ways to create a Keras model with Tensorflow 2.0 (Sequential, Functional and model Subclassing)*”, Oct 2019, <https://www.pyimagesearch.com/2019/10/28/3-ways-to-create-a-keras-model-with-tensorflow-2-0-sequential-functional-and-model-subclassing/>
- [39] Andrew Ng, deeplearning.AI, “*Attention Model Intuition*”, Feb 2018, <https://www.youtube.com/watch?v=SysgYptB198>
- [40] Andrew Ng, deeplearning.AI, “*Attention Model Intuition*”, Feb 2018, <https://www.youtube.com/watch?v=quoGRI-1l0A>
- [41] Microsoft Research, “*Machine Reading Comprehension*”, July 2019, <https://www.microsoft.com/en-us/research/project/machine-reading-comprehension/>
- [42] Alvira Swalin, “*Building a Question Answering System from Scratch – Part I*”, May 2018, <https://towardsdatascience.com/building-a-question-answering-system-part-1-9388aadff507>