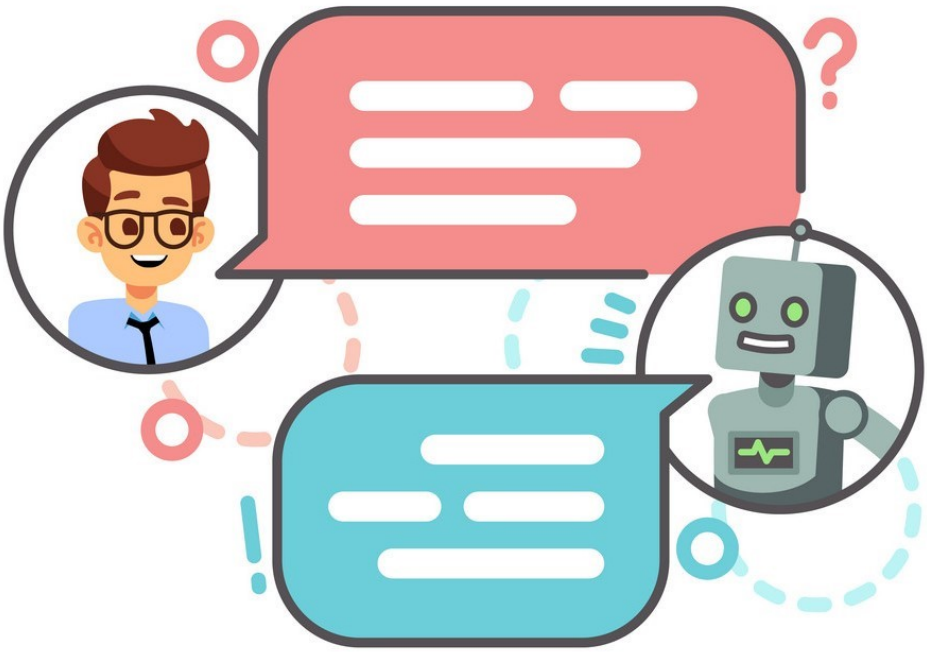
CHALMERS
UNIVERSITY OF TECHNOLOGY



# Question Answering In Conversational Context

## Using FlowQA and BERT for modelling conversations in QuAC

Master's thesis in Computer science and engineering

VISHNU RAVEENDRA NADHAN

SOUMYADEEP MONDAL

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2019

Master's thesis 2019

# Question Answering in Conversational Context

Using FlowQA and BERT for modelling conversations in QuAC

VISHNU RAVEENDRA NADHAN
SOUMYADEEP MONDAL


CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
Gothenburg, Sweden 2019

Question Answering In Conversational Context



Supervisor: Olof Mogren, PhD., RISE AI, Research Institutes of Sweden
Examiner: Richard Johansson, PhD., Department of Computer Science and Engineering, Chalmers University of Technology and University of Gothenburg

Master's Thesis 2019
Department of Computer Science and Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: A user engaged in a conversation with a QA system.

Typeset in LaTeX
Gothenburg, Sweden 2019

Question Answering In Conversational Context
Using FlowQA and BERT for modelling conversations in QuAC

Vishnu Raveendra Nadhan, Soumyadeep Mondal
Department of Computer Science and Engineering
Chalmers University of Technology

# Abstract

In this era of digital technology when people are busy with their daily life, they look for methods to learn quickly with minimal effort. Today, people more often depend on machines to store and retrieve information. Soon, they will interact with a machine to seek information conversationally by asking questions to establish a continuous dialogue based on the information gained through the conversation. This thesis aims to study existing models created to help such machines, for a popular dataset QuAC (Question Answering in Context) [1]. Furthermore, this thesis looks to reduce the gap between the state-of-the-art $F_1$ score of 64.1% (achieved by FlowQA [2] at the beginning of this thesis) and the human performance of 81.1%. In this thesis, we mainly focused on experimenting with FlowQA by (1) replacing the attention mechanism with multi-head attention, (2) integrating BERT (**B**idirectional **E**ncoder **R**epresentation from **T**ransformer) [3]. In every experiment, there was a considerable amount of increment in the $F_1$ score, with the highest score being 66.4% achieved by a novel combination of FlowQA and BERT along with the concept of obtaining contextualized word embeddings using a combination of dialog history and a moving window. Moreover, this thesis also developed a model using BERT alone that delivered an accuracy of 43.4% on the QuAC dataset.

Keywords: Machine Learning, Deep Learning, Neural Networks, Machine Comprehension, QuAC, Question Answering, Transformer, BERT, FlowQA, Recurrent Neural Network (RNN), Natural Language Processing (NLP)

# Acknowledgements

We would like to thank and express our gratitude to our supervisor Olof Mogren for giving us an opportunity to work on this challenging thesis, and for offering us a collaborative environment at RISE Research Institutes of Sweden, to perform our thesis. His continuous feedback, assistance and encouragement during the entire project has kept us going till the finish line. We would also like to thank him for being our advisor at Chalmers University of Technology. Special mention to two researchers at RISE, John Martinsson and Edvin Listo Zec for their valuable inputs and for allowing us access to more computational resources to execute our thesis.

We are thankful to the authors of FlowQA for open-sourcing the codebase of FlowQA and for permitting us to re-use the image of FlowQA architecture in this thesis.

We further want to thank our examiner Richard Johansson for his support and his ideas on further experiments to improve our thesis.

Finally, we would like to thank our family and friends for the constant encouragement and support thoroughout the project.

Vishnu Raveendra Nadhan & Soumyadeep Mondal, Gothenburg, September 2019

# Contents

# List of Figures

# List of Tables

# 1
# Introduction

The present year, a part of the 21st century has seen an increase in the volume of available information and its constituent data. Besides this, internet users are increasing in huge numbers and everyone looks up answers to their questions on the web. They also come across tons of information on popular websites like Wikipedia, Reddit, Quora, to name a few. Researchers proposed a problem called Question Answering (QA), which comes under Natural Language Processing (NLP), to encourage building solutions that could assist these users to find information by interacting with a machine. NLP consists of techniques used to analyze, understand, and derive meaning from human language in a smart and useful way. Using NLP, one can perform tasks such as automatic text summarization, translation, speech recognition, to name a few. A QA implementation typically involves building a system that automatically answers a user's questions and that can engage in a conversation with the user in the form of natural language-based dialogue. Since the 1960's, QA systems were progressively developed to be successful in chosen domains. More recently, there has been a spike in the research on QA systems, thanks to the advances in Machine Learning (ML) and Deep Learning (DL). QA can broadly be divided into two categories: One approach focuses on semantic parsing, where answers are retrieved by turning a question into a database query and subsequently applying that query to an existing Knowledge Base (KB). The other category is more closely related to the field of Information Retrieval (IR) and Machine Comprehension (MC).

The latter category consists of a passage (or context) and a set of questions for which a QA system predicts either descriptive answers or a span of text directly from the passage. A QA system belonging to this category uses machine comprehension techniques to understand the passage and the question, and uses IR algorithms to extract a span of text as an answer to the question. To support the development of the state-of-the-art ML models for machine comprehension, several large datasets were published by researchers. These include QuAC [1], SQuAD [4], and CoQA [5] for automated question answering, Trivia QA for complex compositional answers and multi-sentence reasoning, CNN/Daily Mail and Children's Book Test dataset for cloze-style reading comprehension, and many more. For machine comprehension, the state-of-the-art models have recently been able to outperform human benchmarks except for the QuAC dataset which has several differences compared to other datasets.

The focus of this thesis is on modelling a conversational QA system where the conversations are mostly multi-turn as opposed to single-turn where the questions,

concerned with a passage, are independent. The QuAC dataset is the preferred dataset for this thesis, as it ticks off all the requirements for an almost perfect conversational system. These requirements will be listed and briefed upon in Section 3.2. Recent research has culminated in QA systems that have certain common components, the most important among them being Recurrent Neural Networks (RNN), word embeddings, and attention mechanisms. RNNs are the de facto neural framework for QA, as these networks are designed to deal with sequential information, such as sentences where inputs depend on each other. By storing various parts of the sequence in the network's memory, RNNs can model the contextual relationship between words, phrases, and sentences to perform better IR and MC. Word Embeddings are an essential component of any deep neural network, where the large inputs are projected to a higher dimensional space. The RNN networks are usually combined with attention mechanisms to model the complex interaction between a question and a context. There has been much notable research that resulted in new attention mechanisms such as Bahdanau Attention [6], Dynamic Co-Attention [7], Attention-Over-Attention [8], and many others.

Using RNNs, it is tricky to model long-range dependencies and they do not exhibit parallelization which in turn increases the amount of resources required for large RNN-based networks. Recently, Vaswani et. al. proposed a novel model, Transformer [9], that solely depended on attention mechanisms rather than on RNNs. It was aimed at common applications under NLP such as Machine Translation, Text Classification, Question Answering, and others. The Transformer has a sequence-to-sequence architecture and the authors claim improved results on several tasks under NLP. One major research following Transformer resulted in BERT [3], which is a language representation model that can be used to obtain pre-trained word embeddings and can also be fine-tuned by merging it with additional task-specific layers.

At the time of writing of this thesis, no research was found to have used BERT for QuAC. As per the QuAC leaderboard statistics[1], the state-of-the-art model was FlowQA [2] which depended heavily on flavours of RNNs and different attention mechanisms. This model achieved an $F_1$ score of 64.1% as against the human benchmark $F_1$ of 81.1%. However, during the thesis period, many other models were published by innovators and the most recent one scored an $F_1$ of 72.9%. This thesis aims to create a novel architecture combining the benefits of both RNN and BERT to surpass the currently established $F_1$ scores.

## 1.1 Problem Formulation

The focus of this thesis is on modelling a multi-turn conversational system to seek information about a context. The term "context" used in this thesis, is also known as "passage" which refers to a random continuous block of text sourced from Wikipedia. The system is given a passage and a user poses a question one after another. For

[1] https://quac.ai/

a given question, the system predicts an answer span from the context, after which it waits for the next question from the user. The system involves three main tasks and each has several sub-tasks as follows:

1. Text Encoding
   - Encode the passage, the question, and the conversation history
2. Reasoning
   - Use the words in the question and in the answers to the previous questions, to attend to certain sections of the context.
   - Analyze the attention scores to rank the words in the context.
3. Answer Prediction
   - Predict the start and end index of the answer span.

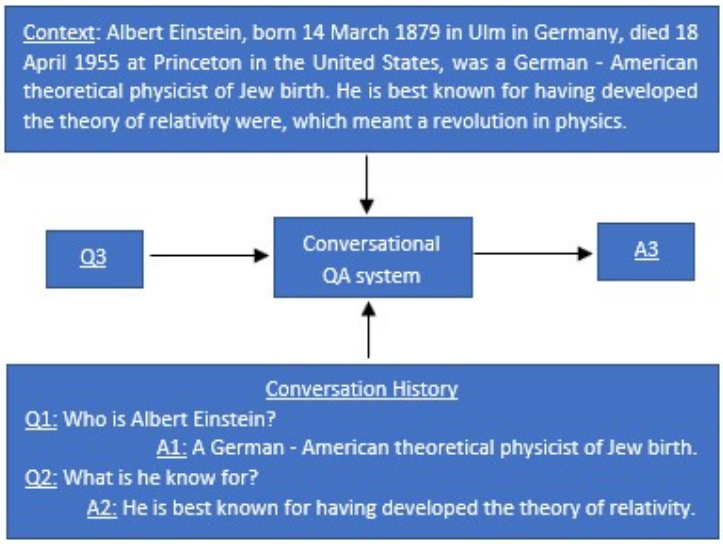



**Figure 1.1:** Illustration of a conversational QA system with an example

## 1.2 Project Purpose

QA is a hot topic in the AI research community and for the QuAC dataset there is a big gap between the human benchmarks and the accuracy scored by the latest QA system. This thesis aims to conduct experiments to improve the established accuracy by integrating novel methods and inspirations from other research papers. The core focus of this thesis is on FlowQA [2] and BERT [3]. The former is the current state-of-the-art model for QuAC and the latter is a breakthrough model in the domain of language representation. Experiments are conducted to find out if BERT can be used in some way to achieve better accuracy for the QuAC dataset.

## 1.3 Project Objectives and Limitations

The thesis work intends to answer the following questions:

1. Can parts of the Transformer [9] be used in FlowQA [2], for better performance? For example, replace a single attention layer [10] with a multi-head attention inspired from Transformer to obtain better attention results. Will this increase the performance of FlowQA?
2. BERT [3] is a recently published model for language representation to create word embeddings. Can this be used in the place of RNN-based ELMo word embeddings [11] to obtain a better representation of the words in both the context and the question?
3. BERT solely depends on a stack of attention layers instead of RNNs. Recent results [2] have established that the BERT model can be used to create models with better performance for datasets like SQuAD and CoQA. Can the same be proved for QuAC dataset?

The architecture of FlowQA is complex due to the involvement of comprehensive mechanisms to encode conversation history to answer a question. This complex structure is important to address the co-reference between a (context, question, answer) at a point of time in a conversation. Co-reference is indeed an important factor in QuAC dataset, which is essential to build better conversational models. Due to the architectural complexity of FlowQA, more time is required to be spent on the research on FlowQA. Due to time constraints, less time will be spent on developing better mechanisms that deal with co-reference and also on modelling of conversation history in BERT.

[2]The research that achieved these results are not open-source and the results are available on the leaderboard of SQuAD and CoQA datasets.

# 2
# Theory

This chapter introduces deep learning in general and covers the theoretical concepts, terminologies, and components that are relevant to this thesis. Many of these were used to construct the model architectures for the methods in the thesis, while some of them were experimented with to gain further understanding and the learning from such experiments is covered in the discussion section.

## 2.1 Deep Neural Network

Artificial Neural Networks (ANN) are systems inspired by the biological neural networks which constitute the animal brain. These systems can learn to recognize and predict patterns by considering examples and can progressively improve their abilities. An ANN is a collection of connected units called neurons which are organized in layers. It is widely used in various applications such as image recognition, character recognition, stock market prediction, etc. The most common types of neural networks are Feed-Forward Neural Network (FFNN), Convolutional Neural Network (CNN), and Recurrent Neural Network (RNN).

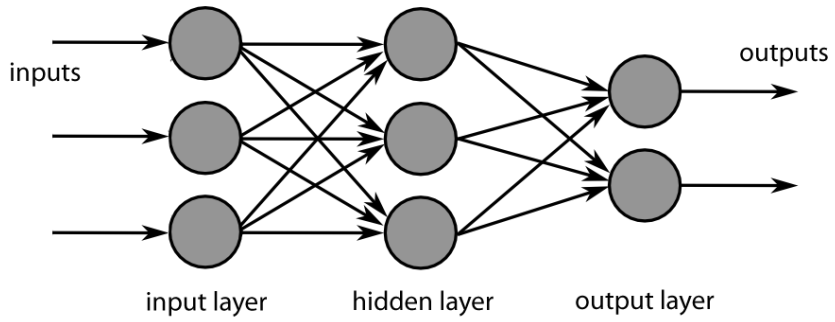



**Figure 2.1:** Deep Neural Network (Diagram of a multi-layer feedforward artificial neural network by Christoph Burgmer, licensed under CC BY-SA 3.0-migrated)

A Deep Neural Network (DNN) is an ANN with multiple layers between the input and output layers, hence the word "Deep" in DNN. A DNN can model a complex linear or non-linear relationships between the input and the desired output. These are typically feed-forward networks where the data flows from input to output layers without looping back. Figure 2.1 represents a deep neural network with one hidden

layer. The input layer receives the input data features, which are then processed by single/multiple hidden layers and then finally sent to the output layer. This output layer can represent the output based on specific tasks.

### 2.1.1 Artificial Neuron

The smallest component of a neural network is the artificial neuron or unit, which takes the weighted inputs, processes it and is also capable of performing binary classifications. Each node in Figure 2.1 represents a unit. Its mathematical equation is:

$$\boldsymbol{y} = f(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) \tag{2.1}$$

Here, $\boldsymbol{W}$ and $\boldsymbol{b}$ is the weight matrix and bias vector respectively and they are parameters learned during training, $\boldsymbol{x}$ is input, $\boldsymbol{y}$ is output and $f$ is the activation function. The activation function can be linear or non-linear, which defines the output of a unit given an input or set of inputs. Most commonly used activation functions are Sigmoid, Tanh, ReLU, Leaky ReLU which differs based on their range of the curves.

### 2.1.2 Forward Propagation

The training of the model starts by initializing all the parameters i.e. weights and biases of all units, randomly. During the forward propagation, the input data travels through the network from the input layer to the output layer. The mathematical calculations for each layer are given below:

$$\boldsymbol{z_i^{l+1}} = \boldsymbol{W^{l+1}}\boldsymbol{y^l} + \boldsymbol{b^{l+1}} \tag{2.2}$$

$$\boldsymbol{y^{l+1}} = f(\boldsymbol{z_i^{l+1}}) \tag{2.3}$$

Here, $f$, $\boldsymbol{W^l}$, $\boldsymbol{b^l}$ and $\boldsymbol{y^l}$ are respectively the activation function, weights, biases and outputs of the layer $l$.

### 2.1.3 Backward Propagation

After the forward propagation, the final layer outputs $\boldsymbol{y^L}$ are used to assess and improve the performance of the model. A loss function is defined to calculate the error between the model output and ground truth. Some of the most commonly used loss functions are mean square/L2 loss, mean absolute/L1 loss, SVM loss, cross-entropy loss, etc. The model performance is improved by minimizing this loss value, by calculating the gradient for all parameters in the model. This is done using the following formula:

$$\boldsymbol{W^l} = \boldsymbol{W^l} - \alpha\frac{\delta \boldsymbol{Loss}}{\delta \boldsymbol{W^l}} \tag{2.4}$$

$$\boldsymbol{b}^l = \boldsymbol{b}^l - \alpha \frac{\delta \boldsymbol{Loss}}{\delta \boldsymbol{b}^l} \tag{2.5}$$

Here, $\boldsymbol{W}^l$, $\boldsymbol{b}^l$ are the weight matrix and bias vector of all units in layer $l$ and $\alpha$ is a scalar value called learning rate or step size which indicates the amount of reduction to be done in each step. As this gradient calculation and adjustment of the parameters flow from the final to the input layer, this step is called backprop.

**Exploding/Vanishing Gradient problem**

The exploding or vanishing gradient problem arises when a neural network is trained with gradient-based learning algorithm and backpropagation, where the network weights are updated in proportion to the partial derivative of the error function with respect to the current weights in each iteration of training. If the gradient with respect to the weights in the network becomes really high, the update of the weights becomes higher. This, in turn, makes the network unstable, resulting in an exploding gradient problem. Similarly, if the gradient with respect to the weights in the earlier network becomes very small, the update of the weights becomes smaller. This will make the backpropagation ineffective, resulting in a vanishing gradient problem.

## 2.2 Recurrent Neural Network

Recurrent Neural Network (RNN) is a special type of ANN with directed cycles in memory for fixed-size input and output vectors. Unlike FFNN, RNN can use their internal state or memory ($\boldsymbol{h_{t-1}}, \boldsymbol{h_t}, \boldsymbol{h_{t+1}}$.. in figure 2.2) to process sequences of inputs which makes them applicable to inputs like continuous, connected handwriting or speech. RNN comes with few limitations, which are overcome with different variants called LSTM and GRU.

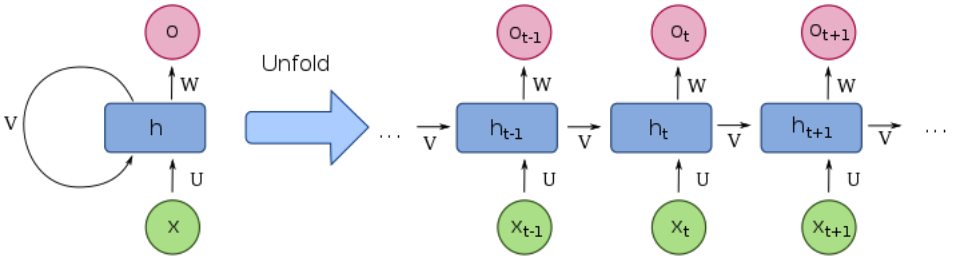



**Figure 2.2:** Unrolled Recurrent Neural Network. (Recurrent neural network unfold by François Deloche, licensed under CC BY-SA 4.0 International)

### 2.2.1 Long Short-Term Memory Networks (LSTM)

Due to the simple structure of RNN units, it comes with the limitation of inability to hold long-term dependencies. LSTM [12] overcomes this by using four different interacting network layers. LSTMs are designed to handle long-term dependencies by remembering only essential information for long periods using cell states ($\boldsymbol{c_t}$) by adding or removing the essential information through pointwise operations like vector additions or multiplications.

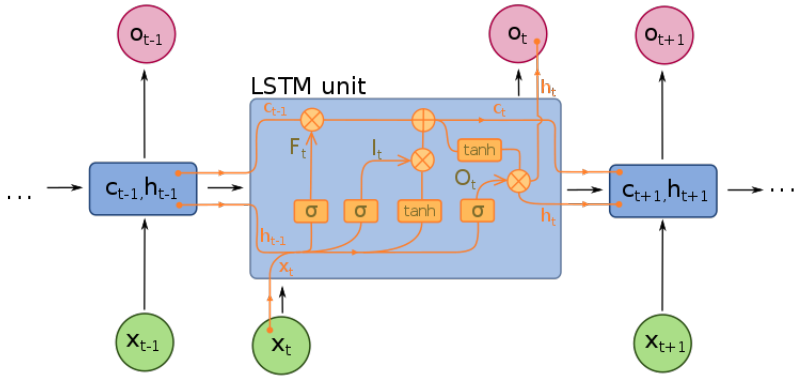



**Figure 2.3:** LSTM Cell (The LSTM cell by Guillaume Chevalier, licensed under CC BY 4.0 International )

The first step in a LSTM cell is to decide what information is not going to be remembered using a sigmoid layer (equation 2.6) known as "forget gate layer".

$$\boldsymbol{F_t} = \sigma(\boldsymbol{W_f}[\boldsymbol{h_{t-1}}, \boldsymbol{x_t}] + \boldsymbol{b_f}) \tag{2.6}$$

Next, it decides which information will be stored in cell state in two steps. A sigmoid layer called "input gate layer" (equation 2.7) specifies the values to be updated and a tanh layer (equation 2.8) creates the new candidate values. Once all these updates are ready, the final cell state ($\boldsymbol{C_t}$) is calculated.

$$\boldsymbol{I_t} = \sigma(\boldsymbol{W_i}[\boldsymbol{h_{t-1}}, \boldsymbol{x_t}] + \boldsymbol{b_i}) \tag{2.7}$$

$$\boldsymbol{C'_t} = tanh(\boldsymbol{W_C}[\boldsymbol{h_{t-1}}, \boldsymbol{x_t}] + \boldsymbol{b_C}) \tag{2.8}$$

$$\boldsymbol{C_t} = \boldsymbol{F_t} * \boldsymbol{C_{t-1}} + \boldsymbol{I_t} * \boldsymbol{C'_t} \tag{2.9}$$

Finally, the cell output ($\boldsymbol{h_t}$) is generated based on the cell state using a sigmoid ($\boldsymbol{O_t}$) and a tanh layer.

$$\boldsymbol{O_t} = \sigma(\boldsymbol{W_o}[\boldsymbol{h_{t-1}}, \boldsymbol{x_t}] + \boldsymbol{b_o}) \tag{2.10}$$

$$\boldsymbol{h_t} = \boldsymbol{O_t} * tanh(\boldsymbol{C_t}) \tag{2.11}$$

### 2.2.2 Gated Recurrent Unit (GRU)

GRU [13] solves the vanishing gradient problem of RNN using different types of gates. The update gate ($Z_t$) determines how much of the previous information need to be passed for future work and the reset gate ($R_t$) decides how much of the past information to forget.

$$Z_t = \sigma(\boldsymbol{W_z} * \boldsymbol{x_t} + \boldsymbol{U_z} * \boldsymbol{h_{t-1}}) \tag{2.12}$$

$$R_t = \sigma(\boldsymbol{W_r} * \boldsymbol{x_t} + \boldsymbol{U_r} * \boldsymbol{h_{t-1}}) \tag{2.13}$$

Finally, the current state memory ($\boldsymbol{h_t}$) is calculated using a tanh layer and an update gate output.

$$\boldsymbol{h'_t} = tanh(\boldsymbol{W}[\boldsymbol{r_t} * \boldsymbol{h_{t-1}}, \boldsymbol{x_t}]) \tag{2.14}$$

$$\boldsymbol{h_t} = (1 - \boldsymbol{z_t}) * \boldsymbol{h_{t-1}} + \boldsymbol{z_t} * \boldsymbol{h'_t} \tag{2.15}$$

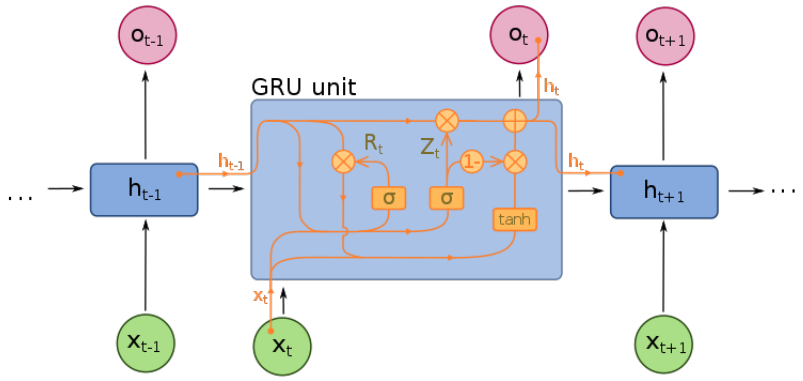



**Figure 2.4:** GRU Cell (The GRU cell by François Deloche, licensed under CC BY-SA 4.0 International)

## 2.3 Word Embeddings

Word embeddings are learned representations for a text where words that have the same meaning have a similar representation. Through embeddings, a word is represented in a $n$-dimensional space which is useful for solving most Natural Language Processing (NLP) based problems.

Mikolov et. al. proposed Word2Vec [14], which is one of the most used forms of word embeddings. It takes a large corpus of text as input to produce a vector space of several hundred dimensions. In this vector space, each unique word in the corpus is assigned to a corresponding vector, which is called as *word embedding* or *word vector*. The word vectors are positioned in the vector space in such a way that, words with common contexts from the corpus are located close to one another. Figure 2.5

shows a visualization of the word vector space and their relationships to one another in 2D, where each vector is represented using a colored circle.

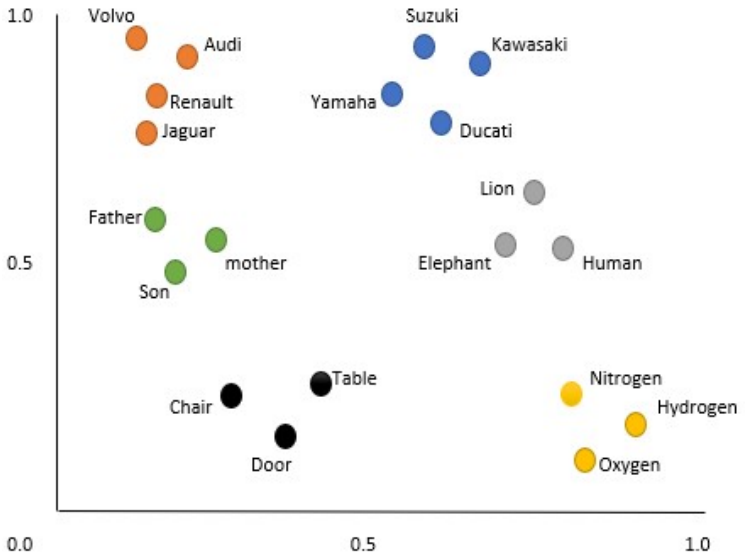



**Figure 2.5:** Illustration of Word Embeddings in 2D

These word vectors can be obtained using two different types of shallow (2 layers) neural networks: Skip Gram and Common Bag Of Words (CBOW).

- Skip-Gram: This model takes a single word (hereby called a focus word) from the corpus paired with some other words surrounding the focus word and after training, it will be able to predict the probability of other words in the corpus to appear around the focus word.

- CBOW: The CBOW model is just the mirror of Skip-Gram. It takes the surrounding words from the corpus as inputs and tries to predict the target word.

Some of the other types of embeddings used in this thesis are described in the following sections. These embeddings are generated using large neural networks and their pre-trained models can be used to initialize word vectors for words in a text. These neural networks can also be fine-tuned, but it will add to the overall size of the models developed in this thesis.

### 2.3.1 Global Vectors

Global vectors (GloVe) [15] is an unsupervised learning algorithm for obtaining the vector representations for words. The representations are obtained using the relationship between the corpus words through their co-occurrence probabilities. It uses a matrix where each entry $X_{ij}$ represents the number of times word $j$ occurs in

the context of word $i$, and $P_{ij} = X_{ij}/X_i$ to represent the probability of the word $j$ appearing in the context of word $i$. Now, the relationship of these words is studied by the ratio of their co-occurrence probabilities.

| **Probability** | $k = solid$ | $k = gas$ | $k = water$ | $k = fashion$ |
|---|---|---|---|---|
| $P(k\|ice)$ | $1.9 * 10^{-4}$ | $6.6 * 10^{-5}$ | $3.0 * 10^{-3}$ | $1.7 * 10^{-5}$ |
| $P(k\|steam)$ | $2.2 * 10^{-5}$ | $7.8 * 10^{-4}$ | $2.2 * 10^{-3}$ | $1.8 * 10^{-5}$ |

**Table 2.1: Example of GloVe embeddings:** Table showing probability of a random word $k$ being related to the words "ice" and "steam". Higher the probability, higher is the chances of the word $k$ appearing in the context of word "ice" or "steam".

### 2.3.2 Contextualized Word Vectors

Contextual Word Vectors (CoVe) [16] is a type of word embeddings learned by an encoder in an attention-based seq-to-seq machine translation (MT) model. The hidden states of MT encoder are defined as context vectors. It improves the performance over the approach that uses only unsupervised word vectors and character vectors.

MT models encode words in context and decode them into another language. These models usually contain an LSTM-based encoder (MT-LSTM [17]) whose outputs are considered as the context vectors. If $s$ is the sequence of input words and **GloVe($s$)** is the sequence of word vectors, then **CoVe($s$)** is the sequence of context vectors produced by MT-LSTM.

$$\textbf{CoVe}(s) = \text{MT-LSTM}(\textbf{GloVe}(s)) \tag{2.16}$$

### 2.3.3 Embeddings from Language Model

Embeddings from Language Model (ELMo) [11] also learns contextualized word representation by pre-training a language model in an unsupervised way.

A bi-directional language model is a combination of forward and backward language model. The forward pass computes the probability of a token $k$ provided the history of $k-1$ tokens. The backward pass is similar to the forward pass, but it runs through the sequence in the reverse order by predicting the previous token provided the future context.

ELMo stacks up all the hidden states across all the layers together on top of a L-layer bidirectional language model (bi-LM) to learn a task-specific linear combination. ELMo is applied on semantic-intensive and syntax-intensive tasks using the top-layer and the first-layer representations of the bi-LM respectively.

## 2.4 Attention

Attention mechanism enables a neural network to focus on relevant parts of the input. It can be categorized in the following ways:

- Self-Attention: It is also known as intra-attention, where the attention is performed at different positions of a single sequence to compute a representation of the same sequence. In the case of machine comprehension, the self-attention mechanism enables us to learn the correlation between the current words and the previous part of the sentence.

- Soft vs Hard Attention: This category is based on accessibility on input for performing attention. The soft attention [6] is performed over the complete input which results in a smooth and differentiable model, Whereas, the hard attention [18] is performed on one part of the input at a time, which requires less calculation at inference time due to lesser input, but it increases the complexity of training due to the non-differentiable model.

- Global vs Local Attention: Global attention [18] is similar to soft attention. Whereas, the local attention [18] is an interesting blend between hard and soft, an improvement over the hard attention to make it differentiable: the model first predicts a single aligned position for the current target word and a window centered around the source position is then used to compute a context vector.

In 2014, Bahdanau et. al. proposed an attention mechanism for neural machine translation [6] consisting of a bidirectional-RNN encoder and decoder. During decoding, the conditional probability, $p$ is obtained for each output word, $y_i$

$$p(y_i|y_1, ..., y_{i-1}, x) = g(y_{i-1}, \boldsymbol{s_i}, \boldsymbol{c_i}) \tag{2.17}$$

where $\boldsymbol{s_i}$ and $\boldsymbol{c_i}$ are the hidden states and context vectors at the $i^{th}$ position of the input sequence. The context vector $\boldsymbol{c_i}$ depends on a sequence of annotations $(\boldsymbol{h_1}, ..., \boldsymbol{h_{T_x}})$, each of which contains information about the input sequence with a focus on the neighbors of the $i^{th}$ word in the input sequence. $\boldsymbol{c_i}$ is computed as a weighted sum of these annotations $\boldsymbol{h_i}$.

$$\boldsymbol{c_i} = \sum_{j=1}^{T_x} \alpha_{ij} \boldsymbol{h_j} \tag{2.18}$$

where the attention weight $(\alpha_{ij})$ is calculated as follows,

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{T_x} exp(e_{ik}))} \tag{2.19}$$

where

$$e_{ij} = score(\boldsymbol{s_{i-1}}, \boldsymbol{h_j}) \tag{2.20}$$

is an alignment model which scores how well the inputs around the $j^{th}$ position and the $i^{th}$ position match in the output. The alignment model *score* is a feedforward neural network which is jointly trained with other components.

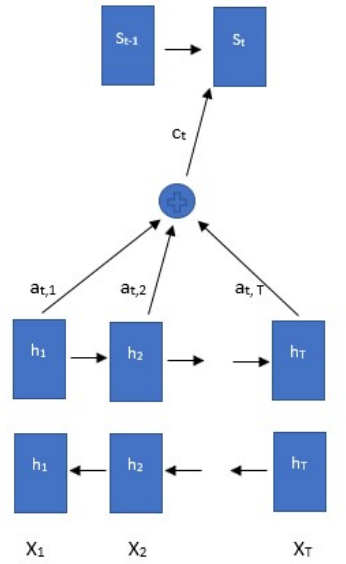



**Figure 2.6:** Attention calculation for Machine Translation

### 2.4.1 Scaled Dot-Product Attention

The paper *Attention Is All You Need* [9] published by Vasvani et. al., introduces a particular type of attention called the Scaled Dot-Product Attention. The attention on the keys ($\boldsymbol{K}$) (vector representation of all words in sequence) is calculated using dot-product, from queries ($\boldsymbol{Q}$) and keys ($\boldsymbol{K}$) of dimension $d_k$ and values ($\boldsymbol{V}$) of dimension $d_v$. As for large values of $d_k$, the dot-product magnitude can grow faster resulting in extremely small gradient, the dot-product is scaled by a factor of $\frac{1}{\sqrt{d_k}}$.

$$Attention(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = softmax(\frac{\boldsymbol{Q}\boldsymbol{K}^{\boldsymbol{T}}}{\sqrt{d_k}})\boldsymbol{V} \tag{2.21}$$

### 2.4.2 Multi-head attention

To achieve higher performance, the authors of *Attention Is All You Need* [9] replaced a single head attention with a multi-head attention by concatenating all the single heads. It allows the model to jointly attend the information from different representation sub-spaces at different positions. In this multi-head attention, the $1^{st}$ dimension of the parameter matrices ($\boldsymbol{W}_i^{\boldsymbol{Q}}, \boldsymbol{W}_i^{\boldsymbol{K}}, \boldsymbol{W}_i^{\boldsymbol{V}}$) must be the product of the head count and the the $2^{nd}$ dimensions of them.

$$MultiHead(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = Concat(head_1, ..., head_h)\boldsymbol{W}^{\boldsymbol{O}} \tag{2.22}$$

$$head_i = Attention(\boldsymbol{QW}_i^{\boldsymbol{Q}}, \boldsymbol{KW}_i^{\boldsymbol{K}}, \boldsymbol{VW}_i^{\boldsymbol{V}}) \tag{2.23}$$

## 2.5 Transformer

The architecture of the Transformer [9] is similar to other sequence-to-sequence models that have an encoder and a decoder, except the basic units that constitute these two. Instead of a LSTM or a GRU, the Transformer uses several attention-based components, which eliminates the dependencies on the length of the inputs and outputs. Higher parallelism can also be achieved with significantly less training time. The architecture of the Transformer includes the following two parts:

**Encoder:** The encoder is a stack of 6 identical layers. Each layer consists of two sub-layers, a multi-headed self-attention mechanism (see Section 2.4.2) and a fully connected Feed-Forward layer. Each sub-layer that produces an output of dimension 512, is surrounded by a residual connection and is followed by a normalization layer.

**Decoder:** The Decoder also consists of 6 identical layers similar to the encoder but with an extra sub-layer to perform multi-head attention on the outputs produced by the encoder.

In the next section, we will describe how the Transformer is used in BERT [3].

## 2.6 Bidirectional Encoder Representations from Transformers

BERT (Bidirectional Encoder Representations from Transformers) was introduced by the researchers at Google AI Language in 2018 [3], it brought an evolution to the machine learning community by presenting state-of-the-art results in different areas.

BERT uses WordPiece embeddings [19] with 30,000 token vocabulary. To generate WordPiece embeddings, a word is divided into a limited set of common sub-word units which provides a good balance between the flexibility of single characters and the efficiency of full words for decoding. It naturally handles the sidesteps needed for special treatment of unknown words. Besides WordPiece, the architecture of BERT is composed of a multi-layer bidirectional Transformer (2.5) encoder. It is designed to pre-train deep bidirectional representations by jointly conditioning on both left and right context in each layer.

The pre-training procedure of BERT is done using two novel unsupervised prediction tasks, as described in the next paragraph. This corpus for pre-training procedure involves BooksCorpus (800M words) and random content from English Wikipedia (2,500M words).

**Masked LM:** In this task, some percentage of the input tokens are masked at random and predictions are made only on those masked tokens. In the case of training of BERT, 15% of all WordPiece tokens in each sequence are masked at random and the final hidden vectors corresponding to the mask tokens are fed into an output softmax layer over the vocabulary.

**Next Sentence Prediction:** To make the model understand the sentences relationships, which is an important part for both QA and NLP, BERT is also pre-trained for a binarized next-sentence prediction task. For this training, a dataset is used with each data containing 2 sentences A and B, where for 50% cases, B is the next sentence, and for the rest 50% cases, B is a random sentence.

The authors of BERT released two pre-trained models with different sizes:

- $BERT_{BASE}$: It consists of 12 identical transformer encoder layer with 512 hidden size and 12 attention head. Total number of parameters is 110M.
- $BERT_{LARGE}$: With 24 identical transformer encoder layer, 1024 hidden size and 16 attention head. Total number of parameters is 340M.

The pre-trained BERT model can be fine-tuned by plugging in one additional output layer to create state-of-the-art models for a wide range of tasks such as Question Answering, Sentence Classification, Next Sentence Prediction, etc. without substantial task-specific architecture modifications.

As a summary of this section, BERT can be used in two ways for this thesis. First, either of the two available pre-trained models alone can be used for feature extraction. The output of feature extraction is the BERT embeddings of the input to the BERT model. The second way to utilize BERT is to fine-tune it with an additional output layer.

# 3
# Related Work

## 3.1 QA Systems using RNN and BERT

Zhu et. al. proposed a neural network, SDNet [20] that combined the benefits of both RNN and BERT. Similar to FlowQA (which will be detailed in Section 4.3), SDNet also leverages both inter-attention and self-attention to comprehend conversation context and extract relevant information from the context. It uses BERT with locked parameters to extract word embeddings for a passage/question/answer/question along with an answer. It obtains the output from BERT in quite a novel way, by taking the weighted sum of the outputs from all transformer layers in BERT. By default, the output returned by BERT is the output generated by the last transformer layer in BERT. Zhu et. al. evaluated their model on CoQA dataset, where they achieved an accuracy of 80.7, but they did not evaluate their model on QuAC dataset. Their performance on CoQA shows that BERT can indeed be used to obtain a better representation for the words in the question, answer and in the passage.

## 3.2 QA Datasets

As introduced in Section 1, the three well-known datasets for question answering is SQuAD, CoQA, and QuAC. Apart from these, a new dataset called Natural Questions [21], was recently published for research in question answering. Since this dataset is new, it will not be looked into in this thesis.

An important fact to be considered is the content of each dataset. In each of three datasets, an entry in the training/validation/test involves a passage and a set of questions with an answer to each question. The passages in both SQuAD and QuAC are based on random text from Wikipedia, whereas in CoQA, the content is from 7 domains: MCTest, RACE, news articles from CNN, Wikipedia, Project Gutenberg, AI2 Science Questions, and Reddit articles.

The three datasets have a few similarities and several dis-similarities as well. One major difference between the three is, the questions in SQuAD dataset does not form a dialog. In other words, each question is independent of the previous questions. Whereas, the other two datasets have questions that are similar to a dialog between two people. Due to this reason, SQuAD dataset was not evaluated in this thesis since the priority is to develop models that can emulate a natural conversation

between a human and a machine. Besides this difference, half of the questions in SQuAD are "what" questions, whereas the others have a much wider, more balanced distribution of types of questions ("what", "why", "who", "where", "when", etc.).

The datasets, QuAC and CoQA, both have several similarities such as (a) both have questions that form a dialog, (b) have single-worded questions such as "who?" or "why?", (c) multi-turn questions. They differ in that, QuAC has span-based answers, whereas CoQA expects abstractive answers.

QuAC is a challenging dataset in that, 54 percent of the questions are non-factoid which is different as compared to factoid questions whose answers are mere facts picked from the concerned passage. Answering non-factoid questions requires a thorough understanding of the concerned passage and identifying all important entities in the passage. Moreover, 86 percent of questions are contextual which require answers that involve a lot of co-reference resolution. An answer can refer to one of the several entities in the article, and can also refer to entities in the past dialog. These kind of questions are difficult to answer, even for humans! It requires a thorough understanding of the passage.

# 4
# Methods

This chapter gives a detailed insight into the selected dataset and the development environment. The architecture of the existing models chosen for modification will be explained in detail. Furthermore, this chapter will have a description of the experiments performed, along with the hyperparameters chosen to tune the models.

## 4.1 Platform

The code for all the experiments is written in PyTorch, an open-source machine learning library for Python, based on Torch. PyTorch is the preferred framework for this thesis due a number of reasons: (a) this thesis uses many data processing APIs of AllenNLP [22], an open-source NLP research library built on top of PyTorch, (b) it provides dynamic computational graphs instead of static graphs as provided by Tensorflow (useful especially when working with RNNs) and, (c) it is user friendly and simpler to use. The experiments were run on a local machine equipped with two NVIDIA 2080Ti GPUs of 11GB each.

## 4.2 QuAC Dataset

The QuAC (Question Answering in Context) dataset [1] contains 14K information-seeking QA dialogs with 100K questions in total. It is split into 10% validation (or dev) dataset and 10% test dataset, with the remaining as the training dataset. Questions in the training dataset have one reference answer, while the questions in the validation (or dev) and the test dataset have five references each. The test dataset is hidden from public to preserve the integrity of the test results. To run any model on the test dataset, the model and the code used to implement it has to be uploaded on the Codalab platform[1]. The challenging fact about QuAC dataset is that, its questions are often more open-ended, unanswerable, or only meaningful within the dialog context. To facilitate more natural interactions, the dataset introduces three types of dialog acts:

- Continuation: Indicates whether the user can ask a followup question based on the current question or not.
- Affirmation: Indicates whether the answer to the current question is yes/no type or not.

[1] https://codalab.org/

- Answerability: Indicates whether the current question can be answered or not.

The question $Q$ can be formulated only from the limited information that has been given to the user and the answer must be a contiguous span of text from the context. The dialog of conversation continues until either (1) twelve questions are answered or (2) user decides not to ask more questions, or (3) more than two unanswerable questions were asked.

| | **Train** | **Validation** | **Test** | **Overall** |
|---|---|---|---|---|
| Questions | 83568 | 7354 | 7353 | 98407 |
| Dialogs | 11576 | 1000 | 1002 | 13594 |
| Unique Sections | 6843 | 1000 | 1002 | 8854 |
| Tokens/Section | 396.8 | 440.0 | 445.8 | 401.0 |
| Tokens/Questions | 6.5 | 6.5 | 6.5 | 6.5 |
| Tokens/Answer | 15.1 | 12.3 | 12.3 | 14.6 |
| Questions/dialog | 7.2 | 7.4 | 7.3 | 7.2 |
| %Yes/No | 26.4 | 22.1 | 23.4 | 25.8 |
| % Unanswerable | 20.2 | 20.2 | 20.1 | 20.2 |

**Table 4.1: QuAC dataset statistics:** Table showing the data distribution across Training, Validation and Test set.

A randomly picked example from the training set of QuAC dataset is shown below. It has a context and set of questions that shows the kind of conversational questions present in QuAC dataset. For each question, apart from the answer there will also be a dialog act. The dialog act in the example is of type Continuation and it takes values such as (Followup) or (No-Followup).

**Context:** "Leadon was born in Minneapolis, one of ten siblings, to Dr. Bernard Leadon Jr. and Ann Teresa (nee Sweetser) Leadon, devout Roman Catholics. His father was an aerospace engineer and nuclear physicist whose career moved the family around the U.S. The family enjoyed music and, at an early age, Bernie developed an interest in folk and bluegrass music. He eventually mastered the 5-string banjo, mandolin and acoustic guitar. As a young teen he moved with his family to San Diego, where he met fellow musicians Ed Douglas and Larry Murray of the local bluegrass outfit, the Scottsville Squirrel Barkers. The Barkers proved a breeding ground for future California country rock talent, including shy, 18-year-old mandolin player Chris Hillman, with whom Leadon maintained a lifelong friendship. Augmented by banjo player (and future Flying Burrito Brother) Kenny Wertz, the Squirrel Barkers eventually asked Leadon to join the group, upon Wertz's joining the Air Force in 1963. His stint in the Scottsville Squirrel Barkers did not last long. In late 1963, his family once again relocated to Gainesville, Florida, when his father accepted a position as Professor of Aerospace Engineering at the University of Florida. Leadon attended Gainesville High School, where he met classmate and future Eagles lead guitarist Don Felder, whose band, the Continentals, had just lost guitarist Stephen Stills. Upon Leadon's joining the group, rechristened Maundy Quintet, they gigged locally, even sharing the bill with future Gainesville legend Tom Petty and his early

band the Epics (a band that also included Bernie's brother, musician Tom Leadon). A call from ex-Squirrel Barker Larry Murray in 1967, to join his fledgling psychedelic country-folk group, Hearts  Flowers, was enticing enough for Leadon to return to California, where he soon became involved with the burgeoning L.A. folk/country rock scene. Leadon recorded one album with the band, their second release Of Horses, Kids, and Forgotten Women for Capitol Records. The record was a local hit but failed to make much of a dent on the national album charts. Discouraged, the group disbanded the following year. CANNOTANSWER"

**Question:** "Where was Bernie born?"
**Answer:** "Leadon was born in Minneapolis," **(Followup)**

**Question:** "Who were his parents?"
**Answer:** "Dr. Bernard Leadon Jr. and Ann Teresa (nee Sweetser) Leadon," **(Followup)**

**Question:** "Did he have siblings?"
**Answer:** "one of ten siblings," **(Followup)**

**Question:** "How was his childhood?"
**Answer:** "As a young teen he moved with his family to San Diego, where he met fellow musicians Ed Douglas and Larry Murray of the local bluegrass outfit," **(Followup)**

**Question:** "Did he start music at this point?"
**Answer:** "The family enjoyed music and, at an early age, Bernie developed an interest in folk and bluegrass music." **(Followup)**

**Question:** "What instruments did he play?"
**Answer:** "5-string banjo, mandolin and acoustic guitar." **(Followup)**

**Question:** "Did he sing also?"
**Answer:** "CANNOTANSWER" **(No-Followup)**

**Question:** "What gigs did he do?"
**Answer:** "His stint in the Scottsville Squirrel Barkers did not last long." **(Followup)**

For evaluation and scoring, the core evaluation metric level $F_1$ score is used. For each question, $F_1$ scores are calculated by measuring the average overlap between the predicted answer and all the ground truth answers separately. Then the maximum of them is considered as the $F_1$ score for that question. Here the precision and recall are computed by considering the overlapping of word portions in both the predicted answer and in all the references. For questions with answer as "CANNOTANSWER", the $F_1$ score is 1 if predicted correct, else zero. For each question, the maximum $F_1$ score among all the references is considered.

QuAC also introduces the human equivalence score (HEQ), a performance measure to judge the similarity of a system's output and that of an average human. It measures the percentage of examples for which, the system $F_1$ is greater than or equal to the human $F_1$. HEQ has two variants, (1) HEQ-Q: the percentage of questions for which HEQ is true, and (2) HEQ-D: the percentage of dialogs for which this is true for every question in the dialog. By definition, the average human performance has an HEQ-Q and HEQ-D of 100% each.

$$F_1 = \frac{2 * precision * recall}{precision + recall} \tag{4.1}$$

$$precision = \frac{number\ of\ tokens\ overlapped}{number\ of\ tokens\ in\ predicted\ answer} \tag{4.2}$$

$$recall = \frac{number\ of\ tokens\ overlapped}{number\ of\ tokens\ in\ reference\ answer} \tag{4.3}$$

## 4.3 Network Architecture

This section will describe the architecture FlowQA in detail. Moreover, this section will give a detailed description of the experiments performed using FlowQA and BERT.

### 4.3.1 FlowQA

For the experiments involving RNN and BERT, the baseline model is FlowQA [2], which incorporates the intermediate representations generated during the process of answering previous questions, through an alternating parallel processing structure. Here we will discuss the model architecture of FlowQA.

This model introduces a Flow mechanism to generate a sequence of latent representations based on the context tokens. As the topic being discussed changes over time through conversational progress, the answer to the same question can also differ significantly.

#### 4.3.1.1 Data Preprocessing

The raw dataset is processed through various ways to make it usable with the model. All the contexts and questions are tokenized to generate the vocabulary, which is then used to generate token ids, POS (parts of speech) tags, named entities and features. Both the POS tags and named entities are obtained from the open source library spaCy and are used as input to the 1st IF layer along with the features. The features include (1) the information about similarities between the question and context words, (2) an indication of, if a word is present in the answers to the two questions previous to the current question. Apart from these, some additional information is generated such as the start and end point of an answer and the answer type for each question, which will be used later during validation.

#### 4.3.1.2 Integration-Flow (IF) Layer

To achieve better parallelism, FlowQA introduces two layers:

- **Context Integration**: Processes the context sequentially in parallel with the question turn. Here the intermediate context representation $C_i^h$ (generated in $h^{th}$ context integration layer) for each question $i$ is sent to a Bi-LSTM layer. During training all questions $i$ ($1 \leq i \leq t$) are processed in parallel.

$$\acute{\boldsymbol{C}}_{\boldsymbol{i}}^{\boldsymbol{h}} = \acute{c}_{i,1}^h, ..., \acute{c}_{i,m}^h = \text{BiLSTM}([\boldsymbol{C}_{\boldsymbol{i}}^{\boldsymbol{h}}]) \tag{4.4}$$

  Here $m$ is the length of the context.
- **Flow**: Processes the question turns sequentially in parallel with context words. The output of the integration layer includes $t$ context sequences of length $m$, one for each question, which are reshaped to $m$ sequences of length $t$, one for each context word. Each of these sequences are then passed to a GRU layer, which helps in utilizing the entire intermediate representation for answering the previous questions when processing the current question. During training, all context words $j$ ($1 \leq j \leq m$) are processed in parallel.

$$f_{1,j}^{h+1}, ..., f_{t,j}^{h+1} = \text{GRU}(\acute{c}_{1,j}^h, ..., \acute{c}_{t,j}^h) \tag{4.5}$$

  The flow layer output is then reshaped back and concatenated with integration layer output, which is later used for further contextualization to predict the start and end answer span tokens.

$$\boldsymbol{F}_{\boldsymbol{i}}^{\boldsymbol{h+1}} = \{f_{i,1}^{h+1}, ..., f_{i,m}^{h+1}\} \tag{4.6}$$

$$\boldsymbol{C}_{\boldsymbol{i}}^{\boldsymbol{h+1}} = c_{i,1}^{h+1}, ..., c_{i,m}^{h+1} = [\acute{c}_{i,1}^h; f_{i,1}^{h+1}], ..., [\acute{c}_{i,m}^h; f_{i,m}^{h+1}] \tag{4.7}$$

#### 4.3.1.3 FlowQA Architecture

FlowQA architecture consists of three major modules:

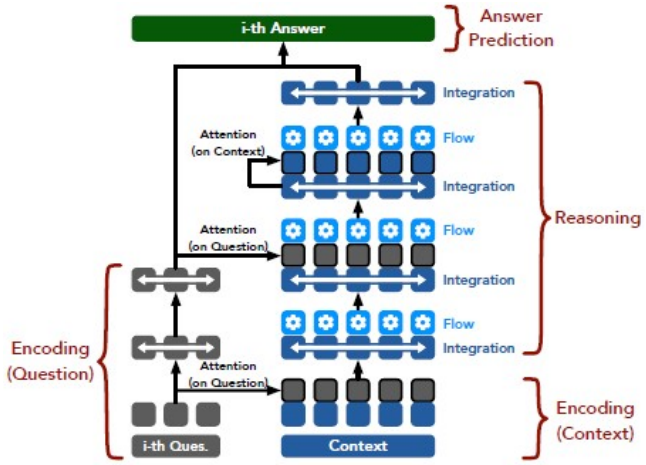



**Figure 4.1:** Architecture of FlowQA[2]

- Context/Question Encoding: In this part, each context and its questions are converted into a sequence of vectors $C_i = \{c_1, ..., c_m\}$ and $Q_i = \{q_{i,1}, ..., q_{i,n}\}$ respectively with pre-trained GloVe [15], CoVe [16], and ELMo [11] embeddings. FlowQA starts processing by calculating word level attention on the questions using DrQA mechanism [23], to enhance the context word embeddings and generates question-specific context input representations ($\boldsymbol{C}_i^0$).

$$g_{i,j} = \sum_k \alpha_{i,j,k} * g_{i,k}^Q \tag{4.8}$$

$$\alpha_{i,j,k} \propto exp(\text{ReLU}(\boldsymbol{W} g_j^C)^T \text{ReLU}(\boldsymbol{W} g_{i,k}^Q)) \tag{4.9}$$

$$\boldsymbol{C_i^0} = [c_1; em_{i,1}; g_{i,1}], ..., [c_m; em_{i,m}; g_{i,m}] \tag{4.10}$$

  Here $g_j^C$ is GloVe embedding for $j^{th}$ context word and $g_{i,k}^Q$ is GloVe embeddings $k^{th}$ question word in the $i^{th}$ question. The final representation ($C_i^0$) includes, (1) context word embeddings ($c_i$), (2) a binary indicator ($\text{em}_{i,j}$, $1 \leq j \leq m$ ) specifying whether that context word appears in a question or not, and (3) output from attention ($g_{i,j}$, $1 \leq j \leq m$).

  In parallel to this, contextualized embeddings for questions ($\boldsymbol{Q}_i^1, \boldsymbol{Q}_i^2$) are obtained using two layers of Bi-LSTM.

$$\boldsymbol{Q}_i^1 = q_{i,1}^1, ..., q_{i,n}^1 = \text{BiLSTM}(\boldsymbol{Q}_i) \tag{4.11}$$

[2]Image sourced from FlowQA [2]. Prior permissions obtained for re-use in this project.

$$\boldsymbol{Q}_i^2 = q_{i,1}^2, ..., q_{i,n}^2 = \text{BiLSTM}(\boldsymbol{Q}_i^1) \tag{4.12}$$

Finally, history aware question vectors $(\boldsymbol{p}_1, ..., \boldsymbol{p}_t)$ are generated, which are later used for answer predictions.

$$\boldsymbol{p}_1, ..., \boldsymbol{p}_t = \text{LSTM}(q'_i, ..., q'_t) \tag{4.13}$$

$$q'_i = \sum_{k=1}^{n} \alpha_{i,k}.q_{i,k}^2 \tag{4.14}$$

$$\alpha_{i,k} \propto exp(\boldsymbol{w}^T q_{i,k}^2) \tag{4.15}$$

- Reasoning: This part includes several IF layers inter weaved with attention calculations. Starting from the bottom, the first IF layer takes different (GloVe, CoVe, ELMo) embedding representations, POS tags, named entities and features generated during preprocessing for the context concatenated with question-specific context representations ($\boldsymbol{C}_i^0$) as input.

$$\boldsymbol{C}_i^1 = \text{IF}(\boldsymbol{C}_i^0) \tag{4.16}$$

$$\boldsymbol{C}_i^2 = \text{IF}(\boldsymbol{C}_i^1) \tag{4.17}$$

The outputs of the 2 IF layers along with the contextualized embeddings for questions $(\boldsymbol{Q}_i^1, \boldsymbol{Q}_i^2)$ are used to calculate fully-aware multi-level attention [24] on question for each context words, to capture the complete information about the questions layer-by-layer.

$$\grave{q}_{i,j} = \sum_{k=1}^{n} \alpha^{i,j,k}.q_{i,k}^2 \tag{4.18}$$

$$\alpha^{i,j,k} \propto exp(\boldsymbol{S}([c_i^0; c_{j,i}^1; c_{j,i}^2], [q_{j,k}; q_{j,k}^1; q_{j,k}^2])) \tag{4.19}$$

$$S(x, y) = \text{ReLU}(\boldsymbol{U}x)^T \boldsymbol{D}\text{ReLU}(\boldsymbol{U}y) \tag{4.20}$$

Here, $S(x, y)$ calculates the attention score between $x$, $y$, and $\boldsymbol{U}$, $\boldsymbol{D}$ are trainable symbol.

These attention scores $(\grave{q}_{i,j})$ along with the output of $2^n d$ IF layer are then passed as an input to the third IF layer.

$$\boldsymbol{C_i^3} = \text{IF}([c_{i,1}^2; \grave{q}_{i,1}], ..., [c_{i,m}^2; \grave{q}_{i,m}]) \tag{4.21}$$

Then, self-attention is performed on the context using fully-aware multi-level attention mechanism [24] using the all IF layer outputs.

$$\grave{c}_{i,j} = \sum_{k=1}^{m} \alpha^{i,j,k}.c_{i,k}^3 \tag{4.22}$$

$$\alpha^{i,j,k} \propto exp(\boldsymbol{S}([c^1_{i,j}; c^2_{i,j}; c^3_{i,j}], [c^1_{i,k}; c^2_{i,k}; c^3_{i,k}])) \tag{4.23}$$

Finally, the outputs of 3rd the IF layer is concatenated with the self-attention scores, which is fed to the last Bi-LSTM layer.

$$\boldsymbol{C^4_i} = \text{BiLSTM}([c^3_{i,1}; \grave{c}_{i,1}], ..., [c^3_{i,m}; \grave{c}_{i,m}]) \tag{4.24}$$

- Answer Prediction: For answer span prediction, FlowQA estimates the start and end probabilities $PS^S_{i,j}$, $PS^E_{i,j}$ of the $j^{th}$ context token of the $i^{th}$ question. For unanswerable questions, FlowQA also calculates the no answer probabilities.

#### 4.3.1.4 Loss and Score Calculation

The answer prediction module calculates the softmax score over all the context words indicating the start and end point of answers, which are then fed to the loss function, cross entropy in the case of FlowQA. The mathematical equation for the multiclass classification cross entropy in given below:

$$loss : -\sum_{c=1}^{M} y_c log(p_c) \tag{4.25}$$

where $M$ is the number of classes. With respect to question answering in context, $M$ indicates the number of context words. $y_c$ indicates (0 or 1) true observation of class $c$ and $p_c$ is predicted probability of class $c$.

FlowQA uses the same evaluation metric level $F_1$ as in the QuAC dataset (see Section 4.2) for score calculation. The dataset contains more than one valid answer for each question. Once an answer is predicted, for each valid answer an $F_1$ score is calculated with that predicted answer. The $F_1$ score is calculated based on the occurrence of matching token counts from the valid and predicted answer. This results in a number of $F_1$ scores for each valid answer, from which the maximum is considered as the $F_1$ score for that predicted answer. Using this $F_1$ score, later the HEQ-Q and HEQ-D are calculated as specified in section 4.2.

#### 4.3.1.5 Training

For training, FlowQA used a dropout of 0.4 after the embedding layer and before applying any linear transformation. Adamax [25] optimizer was used with a learning rate $\alpha = 0.002$, $\beta = (0.9, 0.999)$ and $\epsilon = 10^{-8}$. Originally, FlowQA is trained for a maximum of 20 epochs with batch size of 3 dialogs.

### 4.3.2 Extensions to FlowQA

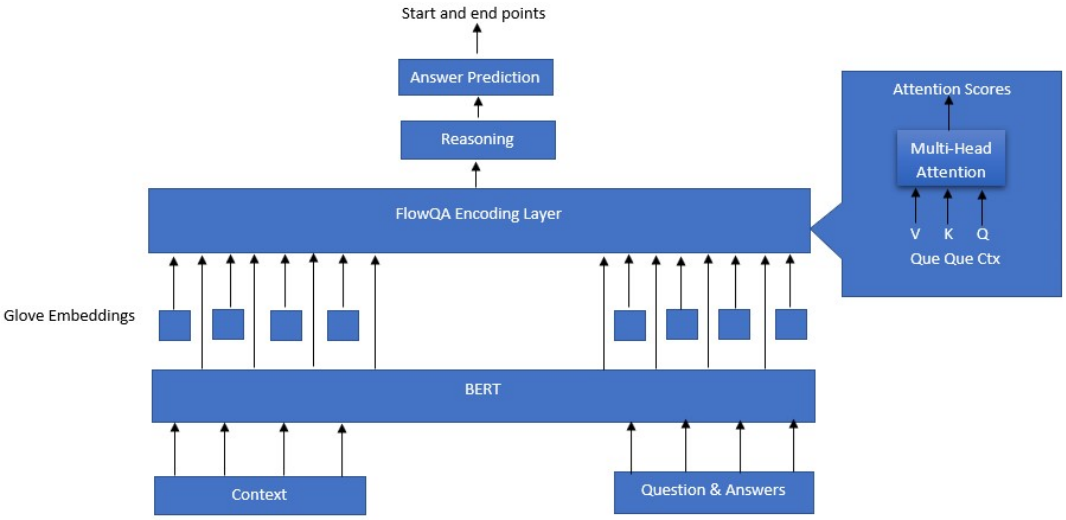



**Figure 4.2:** Schematic diagram of our extension of FlowQA

In this section, the modifications performed on FlowQA will be explained in detail. Four major modifications were performed on the open-sourced codebase of FlowQA. The first among the modifications utilizes the multi-head attention and the rest of them utilizes multi-head attention as well as BERT embeddings. These modifications were aimed at answering the first two research questions of this thesis.

**Multi-head attention**. Multi-head attention was briefly discussed in Section 2.4.2. As noted in the Section 4.3.1.3, FlowQA performs an initial word level attention on all question words for each word in the context. This computes a question specific context representation. This early attention, supported by DrQA [10], is the key for the remaining components in FlowQA. In this experiment, the attention is replaced with a multi-head attention (6 heads in total). This modification is also replicated in the models for the next few experiments described in this section.

The next three modifications described below uses BERT embeddings. The available pre-trained BERT model can be used directly to extract the BERT embeddings for a given input without the need for fine-tuning.

**BERT Embeddings**. In FlowQA, for the current dialog being processed, the contextualized word embeddings for the words in both the question and the context are provided by GloVe, CoVe and ELMo embeddings. The ELMo embeddings, based on RNNs, provides better word representations in a contextual embedding space. The ELMo embeddings for the words in both the context and the question are replaced with the BERT embeddings in the list of inputs for the encoding layer in FlowQA. It has to be noted that most of the contexts in QuAC have a length greater than 512. It was said earlier that BERT has a limitation, its input sequence should have a maximum length of 512. Therefore, to overcome this limitation, the context sequence

is split into mini-sequences of length 512. Each mini-sequences is fed as input to BERT and each of the respective output is concatenated to form the contextualized embedding of the entire context.

**Dialog History based BERT Embeddings**. In the previous experiment, the BERT embeddings for the current question was obtained independently. In this new experiment, the previous two rounds of questions and answers is prepended to the current question to incorporate dialog history. In case, for a given question, only 0 or 1 previous rounds of questions and answers exist, then the available rounds will be appended. Let the $i$-th question be a sequence of tokens $Q_i = \{q_1, q_2, ..., q_m\}$, where $m$ is the maximum length of all questions in the dialog, let the answer to the question $Q_i$ be a sequence of tokens $A_i = \{a_1, a_2, ..., a_n\}$.

As an example: for encoding, reasoning and answer prediction for the fourth question in a dialog, an input is formed by appending sequences of $Q_2$, $A_2$, $Q_3$, $A_3$ and $Q_4$. This resultant sequence is an input to the BERT feature extractor unit to obtain the contextualized embeddings of each token in the input. The exact embeddings for $Q_4$ is extracted from the last $n$ positions in the output of BERT feature extractor.

**Window Based BERT Embeddings**. In the first experiment, the context sequences are split into mini-sequences of length 512. This approach may result in improper embeddings for the words at the edge of a mini-sequence. To encounter this, a novel approach is performed to get a better BERT representation of every token in the context. We slide a window of size 384 and a stride of 128 over the context sequence to create a number of window spans. Each window span is given as an input to BERT to obtain the representation of every token in the window span. This creates multiple representations of a token in the context. To get the best among the multiple representations for a token in the context, a best window span is chosen based on the maximal context around the token in a span. This span is used to get the embedding for the token. In another variation of the sliding window approach, the embedding of a token is considered as an average of its different representations.

### 4.3.3 Standalone BERT

This section details the experiments which answers the third and final research question of this thesis. This experiment simply adapts the BERT model with specific inputs and a linear output layer (described in [3]). The output layer is to predict the start and end positions of the answer span for a given input which includes a context and a question. Basically, this experiment involves end-to-end fine-tuning of the pre-trained version of $BERT_{BASE}$ model fitted with a linear output layer.

The training data for this experiment is a set of <Context, Question> pairs. The context refers to a passage and for the question, the model aims to find the start and end points of answer span from the context. Experiments were also conducted with varied inputs. To consider the conversation history in answering the current ques-

tion, pairs of previous question and answer is appended to the input. As an example, for the $i^{th}$ question, $Q_i$, the training data will be <Context, $Q_0, A_0, Q_1, A_1, ..., Q_i$>.

To comply with the input format that BERT requires, the first token of every input sequence will be a special classification token `[CLS]` followed by the context and another special separator token `[SEP]`. This is then appended with the conversation history along with the current question, which is again followed by the special token `[SEP]`. As an example, the input will look like <`[CLS]`, Context, `[SEP]`, $Q_0, A_0, Q_1, A_1, ..., Q_i$, `[SEP]`>.

## 4.4 Experimental Setup

This section will list and describe the steps to run the experiments described in the previous section. Furthermore, the list of hyperparameters and corresponding values will be mentioned.

### 4.4.1 Data Preparation

As said in an earlier section, the QuAC dataset is an array of contexts (passages), corresponding questions and possible answers. This data needs to be preprocessed in either of the following two ways to create the training data for the experiments.

#### 4.4.1.1 FlowQA with BERT

For experiments involving FlowQA and BERT, the data is preprocessed according to the preprocessing steps performed by FlowQA (described in section 4.3.1.1). In addition to this, preprocessing is also performed for each question and context, before it is input to the BERT model to obtain respective contextualized embeddings. BERT uses WordPiece tokenization to segment a word (also known as token) into several valid sub-word levels. Therefore, every context and question undergoes WordPiece tokenization, before it is given as an input to BERT. The average length of a context in QuAC is 401 tokens which implies after BERT tokenization, there can be more than 512 tokens. For a single word in the context or question, its word embedding is the sum of the word embeddings of its sub-words.

#### 4.4.1.2 Standalone BERT

This section will describe the data preprocessing for experiments that involve fine-tuning the BERT model. Initially, all passages, its corresponding questions, and answers undergo WordPiece tokenization and this will be used in further steps involved. A sliding window is applied on each passage in the dataset using a window of size 450 with a stride of 128. It should be noted that the stride and window size are hyperparameters which can be experimented with different values. After window processing the passage, for each question (corresponding to the passage), a training example is created for each mini-sequence of the passage. E.g. for passage

C and corresponding question $Q_0$, suppose the sliding window produces 3 mini sequences, $C_1, C_2, C_3$, the preprocessed training data will have the entries $<C_1, Q_0>$, $<C_2, Q_0>$, $<C_3, Q_0>$.

### 4.4.2 Training & Validation

#### 4.4.2.1 FlowQA with BERT

The training and validation data is batched in terms of dialogs. A batch size of 2 dialogs is used, which created 5784 and 500 batches for the training set and the validation set respectively. For large-sized models (Dialog History and/or Window Based), the training and validation process consumed 2.5 days, while the others took almost 2 days. The experiments were run for 30 epochs. For each epoch, the training is performed over the entire training set after which a validation is performed on the validation set. A fixed random seed is used across all experiments.

#### 4.4.2.2 Standalone BERT

The training and the validation dataset after data preprocessing resulted in 83568 and 7354 examples. Both the data sets are batched on a batch size of 8, and the model is trained for 3 epochs. After training for 3 epochs, the model is validated on the validation data. Overall, an experiment takes close to 1.5 days to complete.

### 4.4.3 Evaluation

In our experimental study, the core evaluation metric used to evaluate the models is world-level $F_1$ score. We also report the Human Equivalence Score (HEQ) and its two variants, HEQ-Q and HEQ-D.

### 4.4.4 Hyperparameters

In our experiments, we use both $BERT_{BASE}$ with configuration (hidden_size = 768, hidden_dropout_prob = 0.1, intermediate_size = 3072, layers = 12) and $BERT_{LARGE}$ with configuration (hidden_size=1024, hidden_dropout_prob=0.1, intermediate_size=4096, layers=24). The authors of BERT, Devlin et. al. concluded that these values for the hyperparameters gives a better performance.

The experiments on FlowQA with BERT model were run for 30 epochs. Following the trajectory of the training and validation loss on each epoch, it was decided not to continue training for more number of epochs. After training on each epoch, a validation is performed and the evaluation metrics described Evaluation section is calculated. If the $F_1$ score for an epoch is the best found so far, then the corresponding model is saved as the best model.

Huang et. al. notes that the best optimizer (with corresponding values) for the FlowQA network is Adamax [25] with learning rate $\alpha = 0.002$, $\beta = (0.9, 0.999)$ and $\epsilon = 10^8$. This thesis conducted experiments on different values for the learning rate

($\alpha = 0.002, 0.001$), to conclude that the best performing learning rate is $\alpha = 0.001$ for models that have both FlowQA and BERT. The experiments were run with 3 different random seeds (100, 1023, 1024) resulting in almost the same accuracy.

The experiments on standalone BERT were run for 3 epochs. The model was run for more number of epochs, but this resulted in a decrease in values for the evaluation metrics. BERT uses Adam optimizer [25] with configuration: learning rate = $5e^{-5}$, adam_epsilon = $1e^{-6}$, warmup_proportion = 0.1, weight_decay = 0.01. The experiments were run for different values of learning rate ($5e^{-5}$, $4e^{-5}$, $3e^{-5}$) and the best performing learning rate was found to be $3e^{-5}$. Besides learning rate and the number of epochs, the other hyperparameters were the batch size, the window size, and window stride. Since BERT allows only a max of 512 tokens at a time as an input, a window size of length less than 512 can only be used. Different combinations of (window_size, stride) such as (384, 128), (390, 128), (420, 128), (450, 128), (480, 128) were experimented upon. The best performing combination is (450, 128). A batch size of more than 8 for training/validation could not be tested owing to memory limitations.

# 5
# Results

This chapter presents the results of all experiments and ablation studies performed in this thesis. Graphs showing the trajectory of validation loss and $F_1$ score are presented for the validation set of QuAC dataset. The tables list the performance of the best models obtained from each of the experiments. Qualitative and quantitative results of the best performing model obtained in this experiment are also presented for the validation set of the QuAC dataset.

## 5.1 Inspirations from the Transformer

The authors of FlowQA, Huang et. al. open-sourced the code for FlowQA, which is the model FlowQA Baseline in Table 5.1. The initial word level attention in FlowQA (see 4.3.1.3), supported by DrQA [10] is the key for the remaining components in FlowQA. Replacing this with multi-head attention (a core component of the Transformer [9]) increases the performance of FlowQA as shown in the table below:

| **Model** | **$F_1$** | **HEQ-Q** | **HEQ-D** |
|---|---|---|---|
| FlowQA with Multi-head attention (6 heads) | **64.57** | **60.25** | **7.1** |
| FlowQA Baseline | 64.05 | 59.85 | 6.0 |

**Table 5.1: Table with the values for the evaluation metrics for experiments on FlowQA with multi-head attention** Evaluation performed on the validation set. The evaluation metrics include $F_1$, HEQ-Q, and HEQ-D.

Due to this increase in model performance in terms of number of questions and dialogs understood (referring to the values of HEQ-Q and HEQ-D), further experiments involving FlowQA and BERT will include this crucial inspiration from Transformer.

## 5.2 FlowQA with BERT

Table 5.2 reports the performance of all the experiments that integrated BERT into FlowQA. It can be seen that all models obtained from these experiments have a performance that is much better than that of the baseline model. The highest $F_1$ obtained among all the models here is up by 2.35% and this is regarded as the highest performing model developed in this thesis.

| Experiment/Model | $F_1$ | HEQ-Q | HEQ-D |
|---|---|---|---|
| DHBE & BERT Best Window | **66.40** | **62.50** | **8.1** |
| Dialog Hist. w/ BERT Emb. (DHBE) | 65.82 | 61.81 | 6.0 |
| BERT Emb. | 65.33 | 61.30 | 6.5 |
| Window Based BERT Emb. (BERT Best Window) | 65.28 | 61.20 | 5.8 |
| FlowQA Baseline | 64.05 | 59.85 | 6.0 |

**Table 5.2: Table with the values of the evaluation metrics for the experiments on FlowQA with BERT included.** Evaluation performed on the validation set. In the table, BERT Emb. denotes BERT Embeddings.

Figure 5.1 shows the loss reported on the validation set over epochs (for all models). As seen, there is a considerable difference between the validation loss of the baseline model and that of the highest performing model. The trajectory of $F_1$ scores of all the models over epochs is depicted in Figure 5.2. All the models show a similar kind of path although with a good difference in the $F_1$ scores. In the case of FlowQA Baseline and the highest performing model, there is a steep decrease in the $F_1$ score after $12^{th}$ and $19^{th}$ epoch respectively. This is clearly a sign of overfitting due to the varied nature of the questions and expected answers in the dialogs in QuAC dataset.

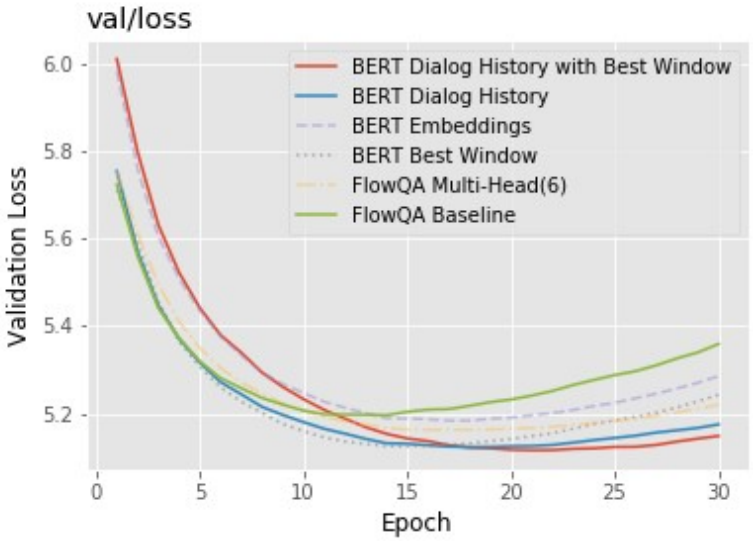



**Figure 5.1: Comparison of validation loss across all models involving FlowQA with BERT.** Results obtained using QuAC validation set. The models FlowQA Baseline and FlowQA Multi-head does not include BERT.

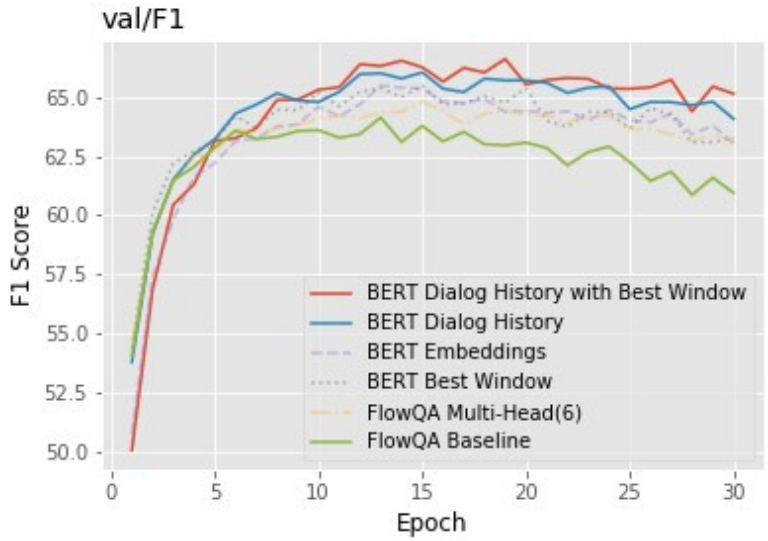



**Figure 5.2: Comparison of $F_1$ score across all models involving FlowQA with BERT.** Results obtained using QuAC validation set. The models FlowQA Baseline and FlowQA Multi-head does not include BERT.

### 5.2.1 Ablation Studies

In all the experiments involving FlowQA and BERT, ablation studies were conducted to find out the importance of using a pre-trained model of $BERT_{LARGE}$ as compared to $BERT_{BASE}$ model. The studies proved that using the former over the latter boosts the $F_1$ scores by atleast 1.5%. Further experiments were attempted to enable fine-tuning of the BERT model when it is used with FlowQA, but due to memory limitation, this experiment could not be completed.

Further ablation studies were conducted on the dialog history which is one of the most important factors to consider besides attention. For models that generate BERT embeddings of a question based on the Dialog history (see description in Section 4.3.2), ablation studies were performed on the numbers of previous (question, answer) pairs appended to the current question. The best performing model (Dialog History w/ BERT Embeddings & BERT Best Window) was chosen to conduct this study and their results are shown in Table 5.3.

| $N$ **previous rounds of (question, answer)** | $\mathbf{F_1}$ |
|---|---|
| 2 | 66.40 |
| 3 | 65.67 |

**Table 5.3: Ablation study on number of (question, answer) pairs in dialog history based BERT Embeddings.** The results were obtained for the validation dataset.

The results in Table 5.3 depicts that, at any point in time, the two previous rounds

of (question, answer) is crucial to obtain the best BERT-based word embeddings for the words in the current question.

## 5.3 Standalone BERT

As said in earlier sections, the experiments that involved the BERT model alone used the pre-trained model of $BERT_{BASE}$. Table 5.4 lists the results of the experiments involved.

| Experiment/Model | $F_1$ | HEQ-Q | HEQ-D |
|---|---|---|---|
| Fine-tuned $BERT_{BASE}$ with Dialog History | **43.4** | **35.4** | **1.8** |
| Fine-tuned $BERT_{BASE}$ Baseline | 38.4 | - | - |

**Table 5.4: Table with the values of the evaluation metrics for the experiments on fine-tuning BERT.** The dialog history contains the previous one round of (question, answer). Evaluation performed on the validation set.

The best performing model has a dialog history involved which performs better than the baseline model. In Section 4.4.1.1, the kind of input to the BERT model was described. To involve dialog history, for every example having a mini-sequence of a passage and a current question e.g. <$C_0$, $Q_1$> will be transformed to <$C_0$, $Q_0$, $A_0$, $Q_1$>, where $A_0$ is the answer to the previous question $Q_0$.

Ablation studies were conducted on the number of (question, answer) pairs ($N = 1, 2$) in the dialog history. The results are listed in Table 5.5.

| $N$ previous rounds of (question, answer) | $F_1$ | HEQ-Q | HEQ-D |
|---|---|---|---|
| 0 (Baseline) | 38.4 | - | - |
| 1 | **43.4** | **35.4** | 1.8 |
| 2 | 41.7 | 33.1 | **2.1** |

**Table 5.5: Ablation study on number of (question, answer) pairs in dialog history for BERT fine-tuning experiments.** The results were obtained for the validation set.

## 5.4 Example Results

This section presents the predictions for the first context from the QuAC dataset and its corresponding questions. The results are outputs of the best performing models from both FlowQA with BERT and Standalone BERT. Along with the predictions, the corresponding Gold Answers is also mentioned and the wrong answers are highlighted in **red**.

### 5.4.1 FlowQA with BERT

As said before, the best performing model that integrates both FlowQA and BERT is *Dialog History w/ BERT Embeddings & BERT Best Window*. Below is a context, its questions and corresponding predicted answers:

**Context:** In May 1983, she married Nikos Karvelas, a composer, with whom she collaborated in 1975 and in November she gave birth to her daughter Sofia. After their marriage, she started a close collaboration with Karvelas. Since 1975, all her releases have become gold or platinum and have included songs by Karvelas. In 1986, she participated at the Cypriot National Final for Eurovision Song Contest with the song Thelo Na Gino Star ("I Want To Be A Star"), taking second place. This song is still unreleased up to date. In 1984, Vissi left her record company EMI Greece and signed with CBS Records Greece, which later became Sony Music Greece, a collaboration that lasted until 2013. In March 1984, she released Na 'Hes Kardia ("If You Had a Heart"). The album was certified gold. The following year her seventh album Kati Simveni ("Something Is Happening") was released which included one of her most famous songs, titled "Dodeka" ["Twelve (O'Clock)"] and reached gold status selling 80.000 units. In 1986 I Epomeni Kinisi ("The Next Move") was released. The album included the hit Pragmata ("Things") and went platinum, becoming the best selling record of the year. In February 1988 she released her ninth album Tora ("Now") and in December the album Empnefsi! ("Inspiration!") which went gold. In 1988, she made her debut as a radio producer on ANT1 Radio. Her radio program was titled after one of her songs Ta Koritsia Einai Atakta ("Girls Are Naughty") and was aired every weekend. In the same year, she participated with the song Klaio ("I'm Crying") at the Greek National Final for Eurovision Song Contest, finishing third. In 1989, she released the highly successful studio album Fotia (Fire), being one of the first albums to feature western sounds. The lead single Pseftika ("Fake") became a big hit and the album reached platinum status, selling 180.000 copies and becoming the second best selling record of 1990. She performed at "Diogenis Palace" in that same year, Athens's biggest nightclub/music hall at the time. CANNOTANSWER

**Question:** what happened in 1983?"
**Answer:** In May 1983, she married Nikos Karvelas,
**Gold Answer:** In May 1983, she married Nikos Karvelas,

**Question:** did they have any children?
**Answer:** she gave birth to her daughter Sofia.
**Gold Answer:** she gave birth to her daughter Sofia.

**Question:** did she have any other children?
**Answer:** CANNOTANSWER
**Gold Answer:** CANNOTANSWER

**Question:** what collaborations did she do with nikos?
**Answer:** **After their marriage, she started a close collaboration with**

**Karvelas.**
**Gold Answer:** After their marriage, she started a close collaboration with Karvelas. Since 1975, all her releases have become gold or platinum and have included songs by Karvelas.

**Question:** what influences does he have in her music?
**Answer:** CANNOTANSWER
**Gold Answer:** CANNOTANSWER

**Question:** what were some of the songs?
**Answer:** Thelo Na Gino Star ("I Want To Be A Star"),
**Gold Answer:** Thelo Na Gino Star ("I Want To Be A Star"),

**Question:** how famous was it?
**Answer:** reached gold status selling 80.000 units.
**Gold Answer:** reached gold status selling 80.000 units

**Question:** did she have any other famous songs?
**Answer:** In 1986 I Epomeni Kinisi ("The Next Move") was released.
**Gold Answer:** In 1986 I Epomeni Kinisi ("The Next Move") was released.

### 5.4.2 Standalone BERT

The best performing model for Standalone BERT is the model which integrates a dialog history that has the previous (question, answer) pair appended to the current question being considered. The prediction for the same passage (context) referred in the previous section is as below:

**Question:** what happened in 1983?"
**Answer:** In May 1983, she married Nikos Karvelas, **a composer, with whom she collaborated in 1975**
**Gold Answer:** In May 1983, she married Nikos Karvelas,

**Question:** did they have any children?
**Answer:** she gave birth to her daughter Sofia.
**Gold Answer:** she gave birth to her daughter Sofia.

**Question:** did she have any other children?
**Answer:** **she gave birth to her daughter Sofia.**
**Gold Answer:** CANNOTANSWER

**Question:** what collaborations did she do with nikos?
**Answer:** **she collaborated in 1975**
**Gold Answer:** a composer, with whom she collaborated in 1975

**Question:** what influences does he have in her music?
**Answer:** CANNOTANSWER
**Gold Answer:** CANNOTANSWER

**Question:** what were some of the songs?
**Answer:** **she participated at the Cypriot National Final for Eurovision Song Contest with the song Thelo Na Gino Star ("I Want To Be A Star"), taking second place.**
**Gold Answer:** Thelo Na Gino Star ("I Want To Be A Star"),

**Question:** how famous was it?
**Answer:** **Since 1975, all her releases have become gold or platinum and have included songs by Karvelas.**
**Gold Answer:** reached gold status selling 80.000 units

**Question:** did she have any other famous songs?
**Answer:** **she participated at the Cypriot National Final for Eurovision Song Contest with the song Thelo Na Gino Star (Ï Want To Be A Star¨), taking second place.**
**Gold Answer:** In 1986 I Epomeni Kinisi ("The Next Move") was released.

### 5.4.3 Comparison of results

On comparing the results of Standalone BERT with those obtained by the model that uses both FlowQA and BERT, it can be induced that FlowQA does better modelling of the prior conversation history, to answer the current question. The absence of this in Standalone BERT can be noticed in the last two questions for which the predicted answers are quite different compared to their respective GOLD answers. Moreover, in certain cases, Standalone BERT returns more words per answer as noticed in the answers to the last three questions. These words look like unnecessary words and with their presence, the predicted answer does not sound like an answer to the question.

# 6

# Discussion

This chapter presents an interpretation of the results outlined in Chapter 5. It is important to note that a major limiting factor to the scope of this thesis was the computational requirements for training and evaluating deep learning models.

## 6.1 FlowQA with/without BERT

Following up on the results presented in Section 5.1 for multi-head attention in FlowQA, the performance increase is attributed to the ensemble-like setup of the multi-head attention unit. It is a general view that ensemble models often perform a bit better than a single model. Therefore, better attention in the earliest attention layer of FlowQA increases the performance of the overall model. In quantitative terms, there is an increase in the number of questions answered correctly and 1.1% more well-understood dialogs.

All the experiments that embed BERT in FlowQA produced models that outperformed the performance of FlowQA Baseline model. This can be attributed to the inclusion of BERT Embeddings instead of the default ELMo embeddings. It was mentioned in previous sections that, the $BERT_{LARGE}$ model was used for our experiments with BERT and FlowQA. Both $BERT_{LARGE}$ and FlowQA are large models and combining them would lead to a larger model that would need more time to train and validate. Therefore, in this thesis there was a trade-off between training time and accuracy and the latter was given more importance.

From among all these models, the best performing model is *Dialog History w/ BERT Embeddings & BERT Best Window*. To reiterate, this applies a window on the input to BERT and selects the best embedding for each token in the input, based on the maximal context around the particular token. This input is nothing but the passage in consideration by the model at any point of time. Moreover, the model also appends the previous two rounds of (question, answer) to the current question in consideration. Since the dialog conversations in the QuAC dataset are multi-turn, there can be a lot of Wh- questions and pronoun words in a question. Therefore, considering the dialog history will result in better representations for the words in the current question in consideration. There is also an increase in the number of well-understood dialogs as compared to the performance of FlowQA Baseline model.

### 6.1.1 Interpretation of model predictions

Upon deeper analysis of the prediction results and comparison of answers predicted by FlowQA Baseline and the best performing model, the following conclusions were noted:

- Most of the predicted answers have extra words (in certain cases, short of a few words) at the start or at the end of the answer which makes it incorrect. This may sometimes lead to the subsequent answers being incorrect.

- Compared to FlowQA, extended FlowQA predicts more precise answers. At times, these predicted answers are correct. In certain cases, when the GOLD answer has more words, precise answers will be deemed wrong by the prediction unit of extended FlowQA.

  An example: (the extra/missing words are highlighted)

  **Question:** What made Anna get into politics?
  **Answer:** Carroll entered the national political arena in the 1850s,
  **Gold Answer:** Carroll entered the national political arena in the 1850s, **following her father's appointment as Naval Officer for the District of Baltimore**

  **Question:** who were his parents?
  **Answer:** the son of Martha (nee Callari) and David Reivers
  **Gold Answer:** the son of Martha (nee Callari) and David Reivers (born 1958)**, an actor.**

  **Question:** When did the Greatest Hits come out
  **Answer:** **By the** beginning of 2004
  **Gold Answer:** beginning of 2004

- FlowQA baseline model returns CANNOTANSWER for more questions which have an answer. But in most of these cases, extended FlowQA provides an answer which is almost equal to the GOLD answer but, it still suffers from the problem of extra/shortage of words as noted in the previous point.

Referring to Figure 5.1, which shows the loss reported on the development set over epochs, it can be seen that there is a considerable difference between the validation loss of the baseline model and that of the best performing model, which implies a better understanding of the dataset using BERT representations. Moreover, it can be clearly seen in Figure 5.2 that the $F_1$ score over epochs for all experiments involving BERT representations steadily increases and remains on a certain level of $63\% - 67\%$, whereas there is a slow decrease in the case of FlowQA Baseline model's performance.

## 6.2 Standalone BERT

Looking at the results presented in Section 5.3, it can be induced that dialog history is indeed important for better performance in models that involve only fine-tuning of pre-trained BERT models. But, in the experiments, it was found that only one round of previous (question, answer) pairs as the dialog history helped in a performance increase. There was a decrease in model performance when two previous rounds were used, which should not happen ideally. Because, in QuAC dataset, in a dialog, there can often be a switch in the topic being discussed and usually atmost 3 consecutive questions can be about a particular topic in the dialog. This performance decrease can be attributed to the absence of a performant dialog history modelling mechanism (like in FlowQA) in BERT. Moreover, the answer span prediction is supported by a simple linear layer which could be improved to predict better answer spans.

# 7
# Conclusion

Based on the results and the discussion in the previous two chapters, this chapter relates back to the research questions and draws conclusions about the findings. Finally, suggestions for potential future work are discussed.

## 7.1 Research Questions

To answer the three research questions posed by this thesis, an approach was taken to build upon the two models, FlowQA and BERT. The goal was to achieve a performance that would surpass the state-of-the-art performance ($F_1$ = 64.05%, HEQ-Q = 59.85%, HEQ-D = 6.0%) at the time of writing of the thesis.

The thesis is concluded by reflecting back to the research questions. The first research question was

*Can parts of the Transformer be used in FlowQA for better performance?*

Using FlowQA as the baseline model, the thesis performed replaced the initial attention mechanism with a multi-headed attention unit inspired by the Transformer. Due to the ensemble-like setup of the multi-headed attention unit, a slightly better performance was achieved, but there was scope for more modifications on FlowQA which needed focus on the next research question. The second research question was

*Can BERT be used in the place of ELMo word embeddings to obtain a better representation?*

Using the updated model obtained from experiments for the first research question, the thesis attempted experiments on the type of contextual word embeddings and the way in which it is used. Following up on the recent research on BERT, it was decided to replace ELMo-based embeddings with BERT-based embeddings. Moving away from the traditional way of obtaining word embeddings for a question or a context at a point of time, the dialog history was embedded to the current question to obtain a better representation for the tokens in the question. For the context, a window was applied on the context to get a better representation of every token. Moreover, hyperparameters were modified to suit the updated models. These changes resulted in an $F_1$ score that surpassed the state-of-the-art $F_1$ score by 2.35%. Besides $F_1$, there was better performance on other evaluation metrics, HEQ-Q and

HEQ-D as well for which the scores were 62.50% and 8.1% respectively.

The third and the final research question was

*Following the impressive results of BERT on SQuAD and CoQA datasets, Can the same be proved for QuAC dataset?*

This research question looked to avoid RNNs to focus entirely on the BERT model. With the inclusion of dialog history to the input of BERT model, the experiments were able to surpass the $F_1$ accuracy of the baseline model by 15.0%, but it was not enough. The performance of the other evaluation metrics, HEQ-Q and HEQ-D were quite poor.

## 7.2 Future work

Question Answering is a challenging problem and is one of the most widely researched problems in the AI community. On the challenges posed by the QuAC dataset, some of the current research is focusing on using BERT embedded with sequential networks or better attention mechanisms to obtain a better performance on the datasets. Since time was limited for conducting this thesis, more work could not be done around BERT and co-reference resolution. As future work, the BERT model should be investigated in deeper detail and modified to encode the dialog history in greater detail, which can help to answer more questions and dialogs as well. This will require some form of sequential networks like RNN / CNN embedded in the BERT model. Besides this, the ability of the models to co-reference well between a (context, question, answer) should be investigated.

# Bibliography


[1] Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wen-tau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. Quac: Question answering in context. *arXiv preprint arXiv:1808.07036*, 2018.

[2] Hsin-Yuan Huang, Eunsol Choi, and Wen-tau Yih. Flowqa: Grasping flow in history for conversational machine comprehension. *arXiv preprint arXiv:1810.06683*, 2018.

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

[4] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad. *CoRR*, abs/1806.03822, 2018.

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

[6] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[7] Caiming Xiong, Victor Zhong, and Richard Socher. Dynamic coattention networks for question answering. *arXiv preprint arXiv:1611.01604*, 2016.

[8] Yiming Cui, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu. Attention-over-attention neural networks for reading comprehension. *arXiv preprint arXiv:1607.04423*, 2016.

[9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc., 2017.

[10] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*, 2017.

[11] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

[12] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[13] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase repre-

sentations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[14] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[15] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[16] Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*, pages 6294–6305, 2017.

[17] Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M Rush. Opennmt: Open-source toolkit for neural machine translation. *arXiv preprint arXiv:1701.02810*, 2017.

[18] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[19] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[20] Chenguang Zhu, Michael Zeng, and Xuedong Huang. Sdnet: Contextualized attention-based deep network for conversational question answering. *arXiv preprint arXiv:1812.03593*, 2018.

[21] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

[22] Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. Allennlp: A deep semantic natural language processing platform. *arXiv preprint arXiv:1803.07640*, 2018.

[23] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*, 2017.

[24] Hsin-Yuan Huang, Chenguang Zhu, Yelong Shen, and Weizhu Chen. Fusionnet: Fusing via fully-aware attention with application to machine comprehension. *arXiv preprint arXiv:1711.07341*, 2017.

[25] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.