Conversational Question Answering with DrQA & Seq2Seq Model

Team Alpha-Targaryen

Rahul Patil
Data Science
University at Buffalo
State University of New York
rpatil@buffalo.edu

Eshwar N Kumar
Department of Computer Science
University at Buffalo
State University of New York
eshwarna@buffalo.edu

Soumith Reddy Chinthalapally
Department of Computer Science
University at Buffalo
State University of New York
soumithr@buffalo.edu

ABSTRACT

Conversations make up an important part of human speech and communication. Humans gather information through conversations involving a series of interconnected questions and answers to either seek or test the knowledge of an individual on a particular subject. Depending on the answer for the prior question, the latter is framed, and the conversation builds up on what has already been discussed. This incremental aspect makes human conversations more succinct. An inability to build and maintain common ground in this way is part of why virtual assistants do not usually seem to be competent conversational partners. In this baseline report, we propose a system for conversational question answering using the CoQA dataset[1]. The questions in the dataset are of conversational in nature, and consists of freeform texts and singular word answers such as 'Yes', 'No', 'Unknown', etc. Here, we formulate the Question-Answering task as predicting the start and end position of the answer in the given contextual paragraph and question.

1. INTRODUCTION

Machine Comprehension (MC) of text is an important task in Natural Language Processing. The task of Machine Comprehension has its origins along with the first concepts of Artificial Intelligence. The brilliant Alan Turing proposed in his famous article 'Computing Machinery and Intelligence' what is now called a 'Turing Test' as a criterion of intelligence. Almost 70 years later, Question Answering still remains one of the most difficult tasks of MC. The task is for the machine to be able to understand blocks of text and comprehend it. The machine is given a question and a paragraph serving as a context of the question and is asked to predict the answer based on its understanding of the question and the context. Prior to the advent of deep neural networks, the efficiency of such systems was based on the feature engineering and the models would often result in low prediction rate. The task of a network learning from a given context and answering based on it involves several steps which are explained further. With the advent of Recurrent Neural Networks (RNN) and attention mechanism has led to these models being as natural as a human. In this paper we implement a DrQA system to first derive the span of text from the given passage that has the most relevance to the question. The uniqueness of the dataset, the model implementation details, and further improvements are as follows.

2. PROBLEM STATEMENT

A passage is given as input to our chatbot. The chatbot must analyze the passage and provide answers to questions asked to it in the form of a conversation. It must also provide the text string in the passage from which the text string can be inferred as a rationale.

In addition, there are three main objectives to be achieved by this chatbot. First, each question in the paragraph after the first must be dependent on the conversation history and the bot must be able to answer the question of such kind. The conversation will have two annotators in which the chatbot will act as a second annotator/answerer. Second, the bot must ensure the presence of naturalness of the conversation with coreferences. The answer must be free form text which is a modified version of the text span obtained from the passage (which is quoted as a proof for the answer). The conversation system must work across domains like the field of literature, medicine, science, news, etc.

3. RELATED AND EXISTING WORK

Since the time of advent of Natural Language Processing, question-answering has been a task that is looked upon to be solved. Today, there are various methods with which the question-answering task is achieved but with more advanced methods coming up every day, the need to make machines answer with a naturalness arises. We looked into various models which can achieve this task, some of them are as follows:

Document Reader Question Answering (DrQA) uses bigram hashing and tf-idf to efficiently return subset of articles from the Wikipedia data store. Then it uses RNN to detect answer spans. This model combines a search component based on bigram hashing and TF-IDF matching with a multi-layer recurrent neural network model trained to detect answers in Wikipedia paragraphs However, it is applicable for Wikipedia articles only and there is no coreference and natural form in answers.

A simple but effective way to implement multi-turn context with BERT for conversational machine comprehension performs contextual encoding of previous questions, previous answers and current question and then applies SoftMax to find the start and end index of the answer span. However, this model lacks coreference and natural form in answers.

Stanford Question Answering Dataset (SQuaD) have models defined to answer the necessary objectives of our chatbot except that free-form nature and referencing are absent in the output.

Graph Flow: Most of the existing approaches do not effectively capture conversation history and thus have trouble handling questions involving coreference. In addition, when reasoning over passage text, most of them simply treat it as a word sequence without exploring rich semantic relationships among words. This techniques Exploits Conversation Flow with Graph Neural Networks for conversational machine comprehension that makes use of graphical neural network mechanism to process the paragraph over the traditional Integration Flow (IF) mechanism. Handles the case of abstractive answers in the passage. The graph flow model can effectively capture conversational flow in a dialog and as well shows competitive performance compared to existing state-of-the-art methods like QuAC, DoQA.

4.DATASET

Although there have been several datasets in the past where there is given information and questions based on that information, what makes CoQA dataset different is the proceeding questions for a given passage are based on the conversation history for that passage. The CoQA follows setting as follows: (1) Model first select a span as rationale. (2) Edit the rationale text to obtain free form answers. The dataset contains of 127k conversation turns obtained collected from 8k different conversations across 7 different domains of Children stories, literature, Middle and High School exams, CNN News, Wikipedia, Reddit threads and Science. In the text passages collected from 7 different domains, five are used for in-domain evaluation and two account for out-of-domain evaluation. The average length of each conversation is 15 turns and for each turn consists of a question, rationale, start span, end span and a generated free form answer. Each free form answer can be related to a rationale in the passage. Almost half the questions refer back to the conversation history. For each of the above mentioned domains, there are 100 passages in the development, 100 in the test and the remaining in the training set. For some questions, there are multiple instances of answers. In order to account for the variations in the answers, the dataset contains 3 additional questions in the test and development set. Since the data is conversational in nature, the questions influence answers which in turn influence the next question. In this dataset, the conversation history is indispensable for answering of most questions. The questions are on average 5.5 words long. Certain questions are just 1 word and certain questions contain context words that appear in the passage, we classify such words as a lexical match. About 29% of the dataset contains of such questions If questions do not have a direct relation to the words in the context, but paraphrase of the rationale, they are classified as paraphrasing. The rest are classified as pragmatics. These questions contain phenomena such as synonymy, antonymy, hypernymy and/or negation. Around 33.2% of the questions do not overlap with the given passage. Unlike other datasets such as SQuAD, which is dominated by what questions, CoQA consists of questions where coreferences are the major key (he, him, she, it, they). Apart from the free-form answers that are generated from the given rationale, the dataset also comprises questions with just one word answer as ‘Yes’, ‘No’, ‘Unknown’, some of which does not have a rationale.

5. PROPOSED FINAL ARCHITECTURE & BASELINE

5.1. MODELS

Given a passage P and the conversation history $\{q_1, a_1, \ldots. q_{i-1}, a_{i-1}\}$ and a question q_i, the task is to predict the answer. The gold answers $a_1, a_2, a_3, \ldots a_{i-1}$ are used to predict answer a_i[1]. This task can be modeled either as a conversational response generation or a reading comprehension problem.

5.1.1. CONVERSATIONAL MODEL

An example for conversation generation model is Sequence-to-Sequence model where we append the given passage and the first question as input to the encoder network which is a Bi-LSTM network. The following question would have the context passage and the question and answer as input to the

bi-directional LSTM. The decoder network with a slight modification to the network is used for generation where copy mechanism is used as the answer contains words from the passage. The drawback of this network is, the network is a generational model which would generate answers from the vocabulary of the passage, therefore, making the task of detecting the answer span difficult as the answer may not have the exact rationale from the passage.

5.1.2. READING COMPREHENSION MODEL

The reading comprehension model, a kind being implemented here, focus on finding the span in the answer which matches the question best. Since their answers are limited to spans, they cannot handle questions whose answers do not overlap the passage. This limitation makes them effective learners than the conversational models which generate answer from a large vocabulary space. Since these models require text span as answers, we select the span with highest overlap as gold answer. We then prepend each question with the past question and answer to account for conversation history.

5.2. PROPOSED MODEL

Here, to implement a model, we use a part of DrQA along with a Seq-to-Seq model. DrQA is an open-domain question answering system with primary task of large-scale Machine Reading by Facebook. DrQA will look for answers to the questions in a very large unstructured document corpus, i.e., the model comprises of a Document retriever and a Document reader. Since we will be using a single dataset, we will partly implement the DrQA, using only the document reader[2] and make certain changes to the model to make it suitable for CoQA. The goal of the DrQA is to predict the start and end span of the answer in the given context.

5.3. DOCUMENT READER

Given a question q consisting of l tokens and a paragraph p consisting of m tokens, we develop a RNN model that is applied to the paragraph and the aggregated answer is found.

5.4. DATA PREPROCESSING AND MODEL IMPLEMENTATION

The data is downloaded from the CoQA site which is in the form of JSON, consisting of the following fields: {Source, ID, Filename, Story, Questions: {Input_text, ID}, Answers: {Span start, Span_end, Span_text, Input_text, Turn_ID}. We use natural language toolkit and spacy for data preprocessing. The preprocessing steps are as follows:

1. Normalize text (Remove white spaces).

2. Gather vocabulary from passage, questions, and answers.

3. Build a vocabulary set from all the unique words across training and development data.

4. Tokenize the data and create word to integer mapping.

5. Get the context span for each word.

6. Create similar mapping for answers of training and validation sets.

7. Save the preprocessed training and validation data as pickle files since some of these preprocessing steps take over an hour and this eliminates the need for those steps to be run again.

8. Split the data into batches, pad the data to maximum length sequence in that batch and calculate the mask.

We extract 102126 sentences from the training and validation datasets which are unique. The raw vocabulary set consists of 79598 words.

Paragraph Encoding: We use this to represent all the tokens in a paragraph P as a sequence of feature vectors and then pass them as an input to the RNN. The encoder network uses a multilayer Bi-LSTM and take P_i as concatenation of each layer's hidden unit output. The feature vector consists of word embeddings formed using GLoVe[4] model.

We use word embeddings for the model to learn text where the words have a similar representation. This helps in paraphrasing. GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from the corpus, and the resulting representations showcase interesting linear substructures of the word vector space. In this model, we use a 6 Billion, 300-dimension vector GloVe word vectors which consists of 400k word vocabulary from Wikipedia. We have a total of 36590 words from our dataset that are found in the GloVe vocabulary.

We then use aligned question embedding function on context and questions which is the attention score calculated to capture the similarity between the paragraph and the questions. Specifically, the non-linear word mapping of word embeddings using ReLU. Here, we first compare the exact word match features and then add softmax alignments between non-identical words. This feature enables the model to understand what portion of the context is more important or relevant with respect to the question. The products of projections taken at token level ensure a higher value when similar words from the question and context are multiplied.

The paragraph encoding is then passed to a multilayer bidirectional LSTM. The best number of layers and hidden units are discussed further. Then an attention layer is used to encode the question and is much simpler than the previous layers. Given a set of vectors values, and a single vector query, attention is a method to calculate a weighted sum of the values, where the query determines which values to focus on. In general, there are 3 steps when calculating the attention.,

1. Calculating the attention score
2. Taking the softmax to get a attention distribution
3. Using the learning rate to get weighted sum.

Here, we use bilinear attention, a modified version of dot product proposed in the transformers paper. To implement this layer, we characterize W by a linear layer. First the linear layer is applied to the question, which is equivalent to the product W.q. This product is then multiplied by the context using torch.bmm. Here, at paragraph level, we predict the span of tokens most likely to be correct. We train two classifiers separately to predict each token to be start and end token. We then choose the start and end span such that P(Start)*P(End) is maximized. The whole model can be put together as shown in Fig1.

The output of this model is a start span and end span. These spans are then converted to a list of context ids in the range start and end span. These outputs are then converted into words using the id to context function which maps the id from vocabulary set to a list of ids. Thus, the output of this model is the ID of the context that is appended with the turn ID of the question. We separate the context ID from the turn ID and create a list of results then appended in json format. This result comprises if identifier context ID and turn ID for each question. However, these results are parts of the context which point towards the answer and are not in the perfect answer forms.

Therefore, we need to generate answers from the given answer using a Seq-to-Seq model.

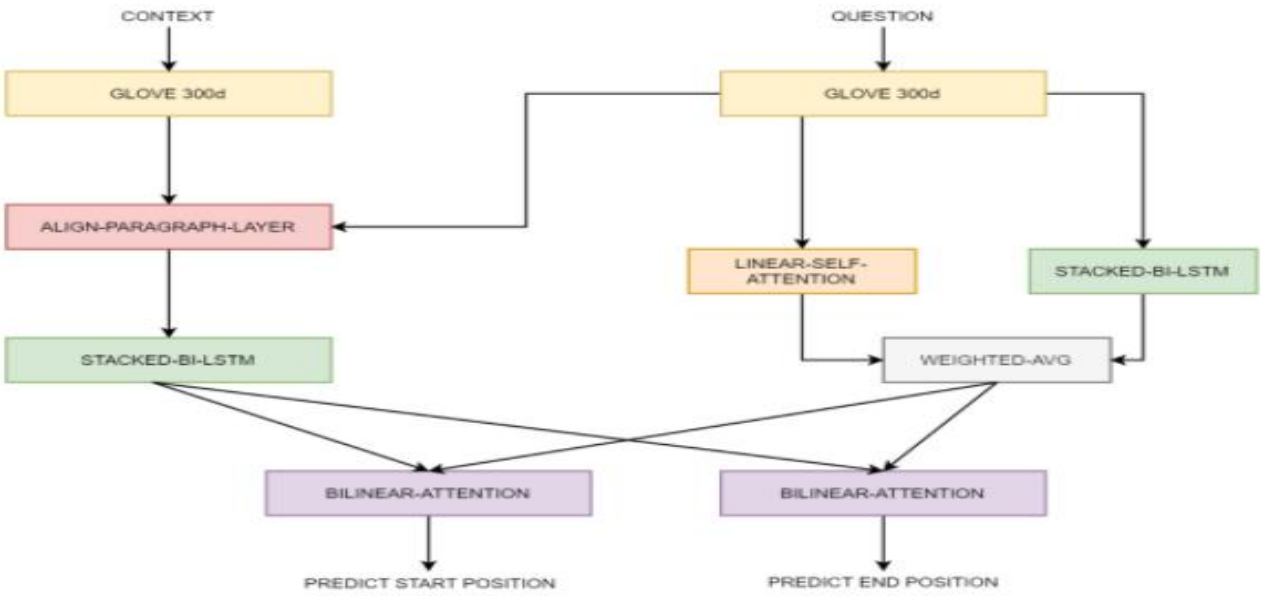

Figure 1 DrQA Model Architecture

Sequence-to-sequence model is a language model which assigns a probability for the likelihood of a given word (or a sequence of words) to follow a sequence of words. The block diagram depicting DrQA model Architecture and seq2seq model is shown in Figure 1 and Figure2 respectively. Sequence to sequence models are a special class of recurrent neural network architectures that is typically used to solve the problem of text generation. They are knowing to be behind a lot of applications used in day to day activities like online-chatbots and google translate, etc. The most common

architecture used to build Seq-to-Seq models is an encoder-decoder architecture. As the name implies, there are two components – an encoder and a decoder. The underlying idea here is to map the input sequence to a fixed size vector using one network, and then to map the vector to the target sequence with another network. The goal of the model is to estimate the conditional probability p (y1, . . ., $y^{T'}$ |x1, . . ., x_T) where (x1, . . . , x_T) is an input sequence and (y1, . . . , $y_{T'}$) is its corresponding output sequence whose length T′ may differ from T. It then computes this conditional probability by first obtaining the fixed dimensional representation v of the input sequence (x1, . . ., x_T) given by the last hidden state of the encoder network, and then computing the probability of y_1, . . ., $y_{T'}$ with a standard formulation whose initial hidden state is set to the representation v of x_1, . . ., x_T. To improve upon this model, we'll use an attention mechanism, which lets the decoder learn to focus over a specific range of the input sequence.

The encoder of a Seq-to-Seq network is a RNN that outputs some value for every word from the input sequence. For every input word, the encoder outputs a vector and a hidden state, and uses the hidden state for the next input word. The decoder here is another network that takes the encoder vectors and translates them into words. Here, in the Seq-to-Seq decoder, we only use the last output from the encoder known as context encoder as it encodes context from the entire sequence. This context vector is used as the initial hidden state of the decoder. At every step of decoding, the decoder is given an input token and the hidden state. If only context vector is passed between the encoder and decoder, only that single vector carries the burden of encoding the entire sentence. Hence, we use attention.

Attention allows the decoder network to focus on a different part of encoder output for every step of decoder's own output. Here, the attention weights are multiplied by the encoder output to create a weighted combination. The result, thus contains information about specific part of the input and helps the decoder choose the right words. Apart from attention, we also use a method called teacher forcing. "Teacher forcing" is the concept of using the real target outputs as each next input, instead of using the decoder's guess as the next input. Using teacher forcing causes it to converge faster but when the trained network is exploited, it may exhibit instability.

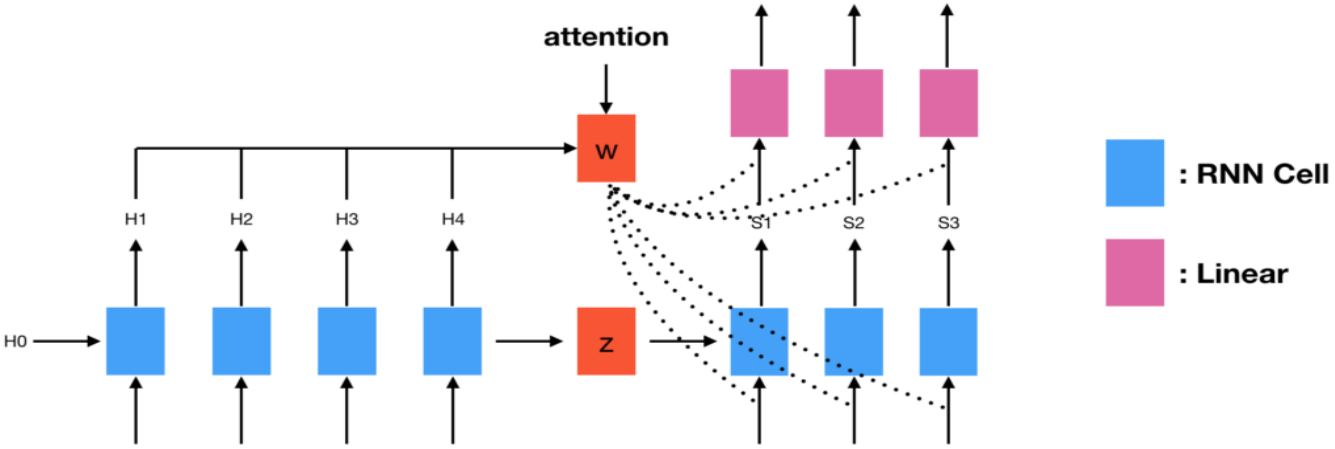

Figure 2.Block Diagram of Seq2Seq Model

We can observe outputs of teacher-forced networks that read with coherent grammar but wander far from the correct translation - intuitively it has learned to represent the output grammar and can "pick up" the meaning once the teacher tells it the first few words, but it has not properly learned how to create the sentence from the translation in the first place.

The data being used here for training are the text span and the answer from the CoQA training dataset, where the text span is the training data X and the answer span are the labels Y. Similar to the character encoding in the DrQA model, we will be representing each word as a vector. We need unique index per word to use as inputs and targets of the network. We have word2index and index2word functions defined for the same. As with the DrQA model, the full process for preparing the data is read text lines and select relevant data, normalize text, filter by content, and make word lists from sentences in pairs.

Here, we use Gated Recurrent Units (GRU) for the network. GRUs as the name suggests is a variant of RNN architecture, and uses gated mechanism to control and manage the flow of information between cells in the neural network.

GRU was introduced in 2014 and can be considered a relatively new architecture, as compared to the widely adopted LSTM. The structure of GRU allows it to adaptively create dependencies from large sequences of data without disregarding information from its earlier parts of the sequence. This is achieved through its gating units, similar to the one in LSTM. Other than its internal gating mechanisms, the GRU functions just like an RNN, where sequential input data is consumed by the GRU cell at each time step along with the memory, or otherwise known as the hidden state. The hidden state is then re-fed into the RNN cell together with the next input data in the sequence. This process continues like a relay system, producing the desired output. While LSTMs have two different states passed between the cells — the cell state and hidden state, which carry the long and short-term memory, respectively — GRUs only have one hidden state transferred between time steps. This hidden state is able to hold both the long-term and short-term dependencies at the same time due to the gating mechanisms and computations that the hidden state and input data go through. This helps us in retaining the previous word history while retaining the history of the complete sentence. To train, for each pair we will need an input tensor (indexes of the words in the input sentence) and target tensor (indexes of the words in the target sentence). The whole training process for the Seq-to-Seq can be stated as: Initialize optimizers and criterion, set the number of epochs for the network to train, calculate the loss at each epoch, repeat till the loss in minimized.

6. RESULTS & ERROR ANALYSIS

6.1. RESULTS

Data Set	In Domain					Out Domain		In Domain	Out Domain	Overall
	Children stories	Literature	mid high school	news	Wikipedia	Reddit	Science			
Baseline Results	11.3	8.1	10.3	9.4	11.2	0.0	0.0	10.0	0.0	10.0
Final Model Results	19.5	14.7	19.1	17.9	19.2	0.0	0.0	18.0	0.0	18.0

Table 1: Baseline and Final Model results comparison

Now, let us analyze the results of our final model and compare it with the baseline.

As a part of our baseline model, we used only the vanilla version of the DrQA and received an over F1 score of 10. For our final model, we implemented DrQA and the Seq2Seq model. We obtained the best F1 score of 18.0 with 3 hidden layers, dropout rate of 0.4 and our model was run for about 20 epochs.

The Table 1 depicts the comparison of our final model results with that of the baseline.

The model has 25.2M trainable parameters when Num_layers are 1 and hidden_dim size of 128, with increase in number of layers and increase in hidden_dim to 300 the hyperparameters increased significantly to 42. 7M and the time to train the model increase.

6.2. ERROR ANALYSIS

Our model has several shortcomings when it comes to F1 score and accuracy. One of the important reasons for the loss of F1 score is from the questions for which the answers are either Yes/No. These questions constitute to about 20% of the total questions our model was trained on. Apart from this, we are also dropping certain questions, answers to which are not available/general questions in the training dataset as DrQA model is responsible for predicting the star5t span and end span of the relevant answers and cannot handle null values. Apart from that, we also do not process the additional answers for certain paragraphs for the given question as that would conflict the original answers extracted from the dataset.

The model is also not been architectured to derive answers from the context, i.e given a date of birth and current date,

the model cannot generalize the answers. Also, the model performs the worst in out-of-domain science and Reddit sections while predicting wrong answers for almost every question. Below is a snippet of examples extracted from the answers obtained by the model and comparison of those answers with the standard answers from the CoQA dataset. As we see, although the model is predicting the correct span of the answer, it is not able to work on the Yes/No section of the questions. We also see that in times where there is an assertion question, the model output span in indefinite, therefore, predicting the long span which finds not much relevance to the question.

Another reason for decreased accuracy can be attributed to the missing naturalness in then answers, as the answers are mostly excerpts from the context itself. Another frequently noticed misevaluation by the model is for the parts of the context which can be referred by the question multiple times. As we know, we use attention calculation for paragraph-question embedding, if there are multiple instances of the context to which the question can point, the model will be rendered inefficient. Therefore, this leaves us with a lot of scope for improvement, importantly with the model preprocessing part such that we handle the edge cases in the dataset.

Examples

Question: "Was Cotton happy that she looked different than the rest of her family?",

Our Model Answer: "She often wished she looked like the rest of her family"

Answer on the dataset: "no",

Question : "What color were her sisters?",

Our Model Answer: "mommy and 5 other sisters . All of her sisters were cute and fluffy , like Cotton . But she was the only white one in the bunch . The rest of her sisters were all orange"

Answer on the dataset: "orange and white",

Question : "Did they want Cotton to change the color of her fur?",

Our Model Answer : "I only wanted to be more like you ". Cotton 's mommy rubbed her face on Cotton 's and said " Oh Cotton , but your fur is so pretty and special , like you . We would never want you to be any other way " . And with that , Cotton 's mommy picked her up and dropped her into a big bucket of water . When Cotton came out she was herself again . Her sisters licked her face until Cotton 's fur was all dry . " Don't ever do that again , Cotton !" they all cried ." Next time you might mess up that pretty white fur of yours and we wouldn't want that"

Answer on the dataset: "no"

Figure 3. Examples showing discrepancies in result

7. CONCLUSION & FUTURE ENHANCEMENTS

CoQA dataset is conversational dataset which introduces a naturalness in the conversation by answering questions which closely relates to the conversations that humans have. Figure 3 provides a few examples indicating where our model provided erroneous results by comparing our models results with the expected results in the CoQA Dataset. We tried designing a network which first uses attention mechanism to relate the answer to the context and then another network that generates the perfect answer forms from the extracted answers. Apart from free form text, the model needs to be able to predict answers such as Unknowns, Yes and No for in-domain and out-of-domain questions. Initial trials with transformer models such as BERT were not computationally well handled and resulted in a resource constraint, therefore using transformers which are not as huge as BERT for embeddings can also be tried with. Apart from this, we do not have explicit function to handle the additional answers from validation data, which needs to be addressed as well. However, this model does pose a lot of challenges in terms of accuracy and the

specificity. However, with more upcoming methodologies and technologies like transformers where trainable parameters are over a billion. This enables a model to be much more accurate with the answers generated. Another reason for a lower accuracy is also that we are using two different models appended to each other and not inter-related to each other. This makes the second model learn from the output of first model, which means if the first model points to a wrong output, the output of the network is wrong as well. Also, the computational requirement was a constraint when it came to a more complex model which can be resolved using services like cloud, etc.

REFERENCES

[1] CoQA: A Conversational Question Answering Challenge. Siva Reddy, Danqi Chen, Christopher D. Manning (https://arxiv.org/abs/1808.07042)

[2] A simple but Effective Method to Incorporate Multi-turn Context with BERT for Conversational Machine Comprehension. Yasuhito Ohsungi, Itsumi Saito, Kyosuke Nishida, Hisako Asano, Junji Tomita (https://arxiv.org/abs/1905.12848)

[3] Technical Report on Conversational Question Answering. Ying Ju, Fubang Zhao, Shijie Chen, Bowen Zheng, Xuefeng Yang, Yunfeng Liu.

[4] GraphFlow: Exploiting Conversation Flow with Graph Neural Networks for Conversational Machine Comprehension (https://arxiv.org/pdf/1908.00059.pdf)

[5] CoQA official webiste (https://stanfordnlp.github.io/coqa)

[6] Attention is all you need, Ashish Vaswani (https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf)

[7] BERT: pre-training of deep bidirectional transformers for language understanding. Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. (https://arxiv.org/abs/1810.04805)

[8] The Illustrated Transformer, Jay Alammar (http://jalammar.github.io/illustrated-transformer/)