A Context-Based Information Extraction based on a Query using deep learning model

D.S.Sameena Begum
PG Student, Department of CSE, Jawaharlal Nehru Technological University College of Engineering (Autonomous) Anantapuramu

Abstract **- Question Answering systems (QA) technology that provides the answer, rather than a list of possible answers, can be referred to as an algorithm. Text similarity and questions asked in natural language are the primary concerns for QA systems in this scenario. Several deep learning models for answering questions have been proposed. Local maxima corresponding to incorrect answers cannot be recovered because of their single-pass nature. So, we're going to use an algorithm called the Dynamic Coattention Network (DCN) to answer questions. DCN combines question and document representations that focus on the most relevant parts of each. Decoders iterate over possible answer spans using dynamic pointing. An initial local maximum associated with incorrect answers can be recovered using this iterative procedure. For Stanford question answering, this DCN ensemble model scores 80.4 percent accuracy.**

Index Terms **- Natural Language Processing, deep learning model, Dynamic Coattention networks, bidirectional LSTM.**

I.INTRODUCTION

Our sentences are input and the machine/computer responds in the context-based information extraction process. Deep learning algorithms, a subset of neural networks, are used to achieve the highest level of accuracy and precision. Neural language processing researchers are working on an intriguing problem. The Question Answering (QA) System is extremely beneficial because it can be used to model most deep learning-related problems. Because of this, it is one of the most heavily researched areas of computer science today. Many of the recent advancements in the field can be attributed to the rise of Deep Learning in the last few years. Finally, the recently proposed Deep Learning methods are addressed in this paper. Implementation and algorithmic tweaks that improved results have also been covered.

A critical part of natural language processing (NLP) is question answering (QA), which necessitates knowledge of the world around us as well as natural language understanding. QA datasets tend to be high-quality but small in size because of the human annotation. The result was that models like deep neural networks, which require a lot of data to train, could not be used. Using semi-automated techniques, researchers have been able to overcome this problem. To train more expressive models, these QA datasets can be used instead of smaller, hand-annotated datasets. Following the completion of this study, it was discovered that different types of reasoning were necessary for the answers to the questions (Chen et al., 2016).

To be RC, a machine must be able to comprehend and apply real-world knowledge, as well as be able to process and answer questions about documents' texts. Many applications are possible, from making it easier to find information to improving artificial intelligence. The majority of natural language processing was done with probabilistic models before they were introduced. Recent advances in deep learning, which have shown to produce superior results, have led researchers to rely more and more on neural networks.

End-to-end neural network called "Dynamic Coattention Network" is used to answer questions. What is encoded in the model a coattentive encoder and dynamic decoder that alternately estimates beginning and end points of the answer span are included in this tool. The F1 of this model is 74.9 percent greater than the best previously published results (Yu et al., 2016). The F1 of our ensemble model is 80.4 percent, which is higher than the second-best SQUAD result.

Stanford University researchers released the SQuAD dataset, which is several orders of magnitude larger than any previous hand-annotated dataset and contains a wide range of characteristics that make QA an obvious task (2016). Each SQuAD response is contained within a section of an answer document. As a result, all possible answers are constrained to this range. According to Rajpurkar et al., the dataset contains many different answers, including multi-sentence reasoning (2016).

II.LITERATURE SURVEY

Traditional methods of research have primarily focused on the syntactic matching of questions and responses. According to Punyakanok et al., dependency tree models were used in order to match questions and answers. The probabilistic tree edit algorithms proposed by Heilman and Smith [2] and Khan et al. [3] can both be used to model sentences. It was used by Yao et al. [4] to extract tree editing sentence answer sequence labelling from the TREC-QA dataset. Based on word relationships, Zhou et al. [5] selected the answer sentences. If you've ever tried it, you know how laborious it can be to hand-label data the old-fashioned way.

In recent years, the use of in-depth learning techniques has grown in popularity. Semantic parsing frameworks based on semantic similarity models and convolutional neural networks were developed by Yih et al. [6] and Wang et al. [7]. There was no syntactic parsing or external knowledge resources like WordNet used in Wang and Nyberg's stacked Bi LSTM study [8]. The questions and answers in these models were not interdependent. Attention is built into our deep neural networks from the ground up.

Rule-based or linear classifiers, which use hand-engineered feature sets, have been used in the past to answer questions. Richardson et al. (2013) suggested two possible baselines: a sliding window and word-distances between words in a question and a document, for example. The method proposed by Berant et al. (2014), on the other hand, is based on constructing a knowledge base of the document's entities and relations first, and then creating a structured query to match the database's content. Frame semantic features, as well as syntactic features such as speech tags and dependency parses, are used in a statistical model. For competitive analysis, Chen and colleagues (2016) developed a statistical foundation using various carefully crafted lexical, syntactic, and word order features.

Recently, many complex queries and matching tasks have relied on cosine similarity as a metric because it has been shown to be an effective one in previous studies like those by Liu [9] and He et al. To classify new data points and support vectors from various categories, Lee et al. (10) used the Euclidean distance to measure the average distance between them [10]. The GESD (Geometric mean of Euclidean and Sigmoid Dot product) and the AESD (absolute error standard deviation) metrics for answer selection were proposed by Feng et al. [10]. (Arithmetic mean of Euclidean and Sigmoid Dot product). This team has developed a set of metrics that they believe to be superior to the others. It is possible to measure the semantic distance between sentences using the cosine similarity and Euclidean distance. When used in conjunction with other evaluation mechanisms, the cosine similarity metric performs better. We are able to improve and optimise previous methods by combining the two functions. This strategy appears to work, according to our research.

NLP frequently employs neural attention models to aid in machine comprehension and question answering. An Attentive Reader model was proposed by Hermann et al. (2015) after releasing the Questions and answers collected from the Questions and answers collected from the CNN/Daily Mail website. For the second time, Hill and colleagues (2016) presented data from the book and proposed a window-based memory network. A single attentional step is performed by this pointer-style attention mechanism, according to Kadlec et al (2016). Machine comprehension tasks were applied to a neural attention model by Sordoni and his colleagues (2016).

Rajpurkar et al. recently released the SQuAD dataset. (2016). Non-entities and longer phrases can be included in answers to real-world questions rather than cloze-style queries. We found that Wang and Jiang (2016b) proposed an end-to-end SQUAD-specific neural network model, and we found that Yu et al. (2016) presented an end-to-end SQUAD-specific neural network model that extracts and ranks a set answer candidate of different lengths from documents in order to respond to questions. As Vinyals et al. (2015) stated, "

A hierarchical co-attention model for visual question answering was presented with the COCO-VQA dataset by Lu and colleagues (2016). (Antol et al., 2015). An image representation and a question representation are computed by the co-attention mechanism in a similar fashion (Lu et al., 2016).

Model	Dev EM	Dev F1	Test EM	Test F1
Ensemble				
DCN (Ours)	**70.3**	**79.4**	**71.2**	**80.4**
Microsoft Research Asia *	–	–	69.4	78.3
Allen Institute *	69.2	77.8	69.9	78.1
Singapore Management University *	67.6	76.8	67.9	77.0
Google NYC *	68.2	76.7	–	–
Single model				
DCN (Ours)	65.4	**75.6**	**66.2**	**75.9**
Microsoft Research Asia *	65.9	75.2	65.5	75.0
Google NYC *	**66.4**	74.9	–	–
Singapore Management University *	–	–	64.7	73.7
Carnegie Mellon University *	–	–	62.5	73.3
Dynamic Chunk Reader (Yu et al., 2016)	62.5	71.2	62.5	71.0
Match-LSTM (Wang & Jiang, 2016b)	59.1	70.0	59.5	70.3
Baseline (Rajpurkar et al., 2016)	40.0	51.0	40.4	51.0
Human (Rajpurkar et al., 2016)	81.4	91.0	82.3	91.2

Fig 1. Performance of DCN Model across various Dev set and Test set which shows great results.

Our dynamic coattention model (DCN) is based on the aforementioned research and includes a new coattentive encoder and a dynamic decoder. Iterative conditional modes (Wang & Jiang, 2016b) are comparable to our model in that they update the start and end positions iteratively to estimate the answer span's start and end positions (Besag, 1986).

III.DYNAMIC COATTENTION NETWORKS

The diagram below depicts the overall process of DCN. The dynamic decoder, which generates the answer span, and the document and question encoders, which contain coattention mechanisms, are both demonstrated here.

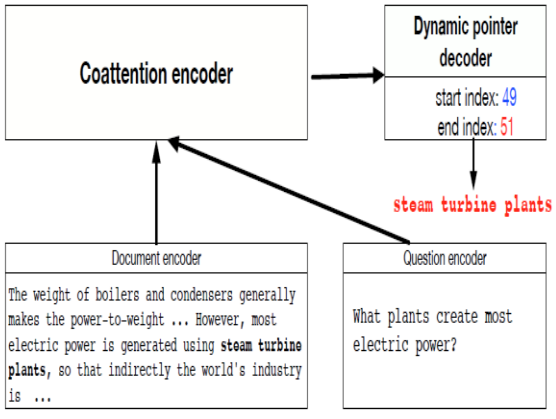

Fig 2. Overall Architechture

B. DOCUMENT AND QUESTION ENCODER

Here, two-word vectors have been taken corresponding to the words in both document and question. Let (x_1^Q, x_2^Q,…., x_n^Q) word vectors in the question are denoted by a sequence of words (x_1^D, x_2^D,…., x_m^D) indicate the same for the text's words and phrases. An LSTM can be used for this purpose, the document is encoded as a: $d_t = LSTM_{enc}(d_{t-1}, x_t^D)$. The Document Encoding Matrix is defined as below $D = [d_1 \dots d_m \; d_\emptyset] \in R^{l\times(m+1)}$. It also consists a sentinel vector $d_\emptyset$ (Merity et al., 2016), in which the model can ignore any particular word in the input, as later demonstrated.

A single LSTM model is used to compute the embeddings for all questions:

$q_t = LSTM_{enc}\left(q_{t-1}, x_t^Q\right)$. We define an intermediate question representation $Q' = [q_1 \dots q_n \; q_\emptyset] \in R^{l\times(n+1)}$. In order to accommodate differences between the encoding spaces for questions and documents, a non-linear projection layer is added to the question encoding. The question can now be expressed as follows:

$$Q = tanh\, tanh\left(W^{(Q)}Q' + b^{(Q)}\right) \in R^{l\times(n+1)}.$$

A.COTTENTION NETWORKS:

A coattention mechanism, similar to that proposed by (Lu et al., 2016), was put forth to pay attention to both the question and the document at the same time. The coattention encoder can be seen in the figure below.

Affinity matrices are calculated for each pair of document words and query words.

$L = D^\top Q \in R^{(m+1)\times(n+1)}$. This matrix is normalised row-wise and column-wise for each word in a question to produce A^Q and A^D for each word:

$$A^Q = (L) \in R^{(m+1)\times(n+1)}$$

$$A^D = softmax\,(L^\top) \in R^{(n+1)\times(m+1)} \qquad (1)$$

The document's summaries or attention contexts are calculated here based on each word of the Question.

$$C^Q = DA^Q \in R^{l\times(n+1)} \qquad (2)$$

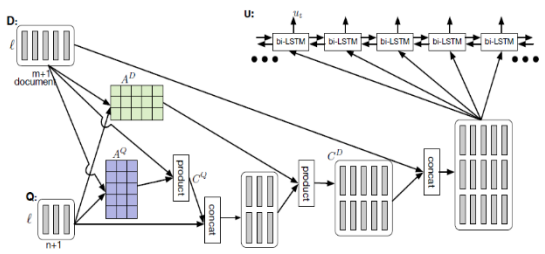

Fig 3. The above figure describes the Coattentive Encoder. Here, A^D and A^Q represents the normalised attention weigths A^Q and A^D.

For each word in the document, we compute the QA^D summaries of the question. We also compute the $C^Q A^D$ attention context summaries in light of each word in this document's content like Cui et al. (2016). Eq. 3 shows that these two operations can be performed in parallel. Question encoding is translated into document encoding space, which is one possible interpretation of the operation $C^Q A^D$.

$$C^D = [Q; C^Q]A^D \in R^{2l\times(m+1)} \quad (3)$$

The question and the document are depicted in a co-dependent manner is defined as C^D in this context. A and B are concatenated horizontally using the [a; b] notation.

The final step is to use a bidirectional LSTM to combine the temporal information with the coattention context:

$$u_t = Bi - LSTM\ (u_{t-1}, u_{t+1}, [d_t; c_t^D]) \in R^{2l} \quad (4)$$

Let $U = [u_1, \dots, u_m] \in R^{2l\times m}$, be the equation the basis for selecting, as a coattention encoding, which span could be the most appropriate response.

C. DYNAMIC COATTENTION NETWORKS

Because of the SQuAD structure, it is possible to generate the answer span by making educated guesses about its beginning and ending points (Wang & Jiang, 2016b). A single document may contain multiple intuitive answer spans, each of which corresponds to a different local maximum. Iteratively switching between predicting the beginning and ending points of the answer span. Incorrect answer spans' initial local maxima can be recovered with this iterative procedure. An LSTM-based sequential model depicted in the figure below maintains the Dynamic Decoder state machine. Using with the help of a multi-layered neural network, the decoder creates new estimations for the start and end positions.

Iteration I of the LSTM is represented by hi, si, and ei, which denote hidden state, position estimate, and end position estimate. Eq. 5 is used to describe the LSTM state change.

$$h_i = LSTM_{dec}\left(h_{i-1}, \left[u_{s_{i-1}}; u_{e_{i-1}}\right]\right) \quad (5)$$

where $u_{s_{i-1}}$ and $u_{e_{i-1}}$ depictions of the coattention encoding U that correspond to previous estimates of its start and end positions.

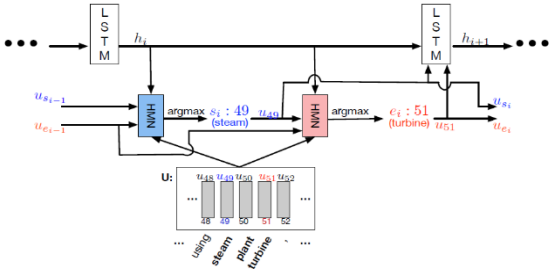

Fig 4. Dynamic Decoder.

When Starting from a starting point, determining the variables, functions, and variables. in blue are indicated by the colour blue, while the variables and functions in red are indicated by the colour red.

D. HIGHWAY MAXOUT NETWORKS

This is based on a previous starting position of hi and current hidden state $u_{s_{i-1}}$ and previous end position $u_{e_{i-1}}$ Equations 6 and 7 estimate the current start and end positions.

$$s_i = argmax_t(\alpha_1, \dots, \alpha_m) \quad (6)$$

$$e_i = argmax_t(\beta_1, \dots, \beta_m) \quad (7)$$

There is a starting score and an ending score for each a word from the text. Here, α_t and β_t are computed using distinct systems of the brain. Despite the fact that they share an overall design, these networks do not share any specific parameters.

Maxout Networks (Goodfellow et al., 2013) and Highway Networks' strong empirical performance (Srivastava et al., 2015), the Highway Maxout Network (HMN) is used to compute the value of t (Eq. 8). The goal of employing such a model is to make QA tasks more diverse by including a variety of question types and document subjects. Different models for estimating the answer span may be required to account for these variations. The use of Maxout Layers makes it easy to pool data from various model variants.

$$\alpha_t = HMN_{star}\left(u_t, h_i, u_{s_{i-1}}, u_{e_{i-1}}\right) \quad (8)$$

According to the coattention algorithm, this word has been encoded in the form: ut. Figure 4 depicts the beginning of an HMN. A separate HMN end is used to calculate the end score, β_t, rather than the start score α_t.

The HMN Model is described below:

$$\left(u_t, h_i, u_{s_{i-1}}, u_{e_{i-1}}\right) = max\left(W^{(3)}\left[m_t^{(1)}; m_t^{(2)}\right] + b^{(3)}\right) \quad (9)$$

$$r = tanh\left(W^{(D)}\left[h_i; u_{s_{i-1}}; u_{e_{i-1}}\right]\right) \quad (10)$$

$$m_t^{(1)} = max\left(W^{(1)}[u_t; r] + b^{(1)}\right) \quad (11)$$

$$m_t^{(2)} = max\left(W^{(2)} m_t^{(1)} + b^{(2)}\right) \quad (12)$$

Where $r \in R^l$ It's a current state parameterized non-linear projection $W^{(D)} \in R^{l \times 5l}, m_t^{(1)}$ consists of the parameters of the first maxout layer $W^{(1)} \in R^{p \times l \times 3l}$ and $b^{(1)} \in R^{p \times l}$ and $m_t^{(2)}$ The second maxout layer's parameters are the output $W^{(2)} \in R^{p \times l \times l}$ $b^{(2)} \in R^{p \times l}$. $m_t^{(1)}$ and $m_t^{(2)}$ parameters are fed into the final maxout layers $W^{(3)} \in R^{p \times 1 \times 2l}, b^{(3)} \in R^p \cdot p$ is the pooling size of each maxout layer. When a maximum is computed over the first dimension. The first and final maxout layers are linked by a highway.

The iterative procedure comes to an end when the estimated start and end positions don't change any more after a certain number of iterations. All iterations' cumulative softmax cross entropy of the start and end positions are minimised when training the network.

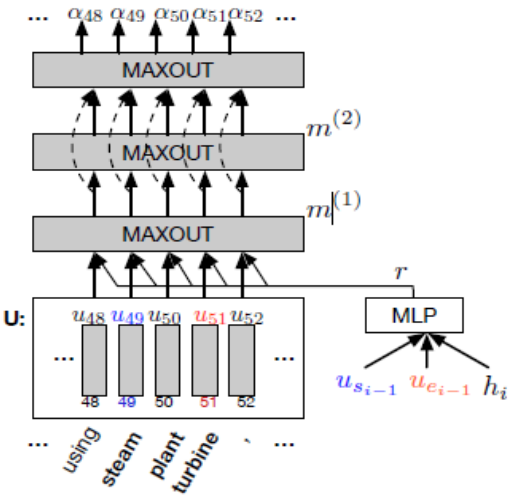

Fig 5. The Road Maxout Network. In highway signage, dotted lines denote where two or more roads merge or diverge.

1V. LSTM AND BI-DIRECTIONAL LSTM

The inputs to LSTM are received by state boxes that change over time. This is how the LSTM output is calculated for each time step:

$$h_t = f_w(h_{t-1}, x_t) \quad (1)$$

Where x_t is the input vector , h_t and h_{t-1} are state vectors at time t and (t-1) , f_w weight vectors w are the nonlinear activation function Figure.1 is a common depiction of an LSTM.

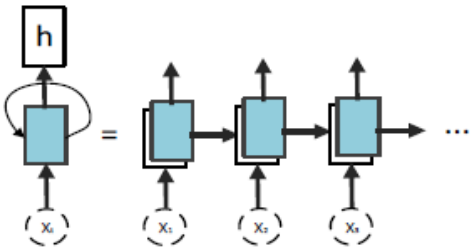

Fig 1: LSTM layer.

The unrolling depicted in Fig.1 can be expressed mathematically in (2), which is derived from [12] and [13].

$$\{C_t = f \odot C_{t-1} + i \odot g \; h_t = O \odot tanh\,(C_t)\,[i\; f\; o\; g\,] = (\sigma\; \sigma\; \sigma\; tanh\,) W(h_{t-1}\, X_t\,) \quad (2)$$

C_t is the cell's output, and forgetting gate f controls whether or not cells erase it; input gate I controls whether or not to write to cell; writing gate g controls function, which includes functions such as tanh and sigmoid. The input from X and the state h are, in general, seen as i⊙g in (2). In order to determine how much of a cell is revealed to the hidden state, ⊙tanh(C_t) represents the cell's current state. The unrolled LSTM and its repeating model (2) are shown in Fig. 2.

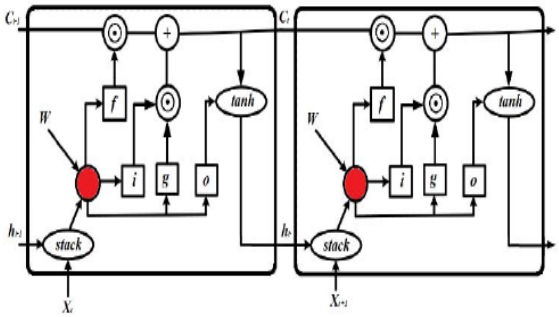

Fig 6. The in-depth description of the unrolled LSTM repeating model The tanh and sigmoid functions are represented by the red circles.

A. Bi- DIRECTIONAL LSTM:

Sequence classification problems benefit from the use of bidirectional LSTMs, which are an extension of the standard LSTM model. Using Bi-LSTMs, you can train two LSTMs instead of one LSTM on input sequences that have all time steps available. Analysis of the original input sequence and a reversed copy of it were carried out separately. Because the network now has more context, getting results is faster.

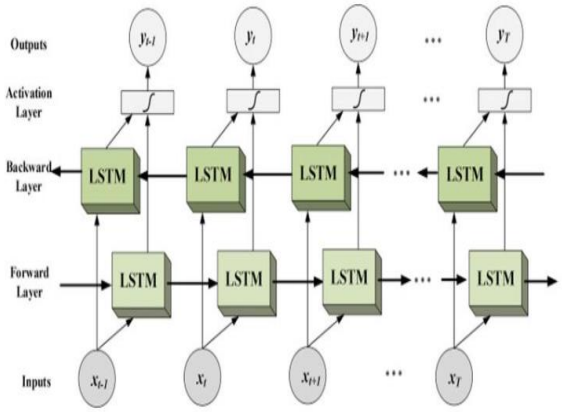

Fig 7. Bidirectional LSTM layer.

RNNs (Bidirectional Recurrent Neural Networks) have a very simple theory behind them. The input sequence is provided when it's used as input to layer 1, and a flipped copy of that input is provided to layer 2 when it's used in layer 2. We no longer have to worry about the drawbacks of the conventional RNN. It's possible to use the past and future to train a bidirectional recurrent neural network (BRNN). Regular RNNs divide state neurons into forward and backward states (positive and negative time directions) (negative time direction). Because the entire utterance is taken into account rather than a linear interpretation, the input sequence is fed both ways when it comes to speech recognition. This means that the network moves through the entire input sequence sequentially in both directions at the same time.

V.EXPERIMENTS

The DCN model described above was trained and tested on the SQuAD dataset. Prior to preprocessing, the corpus was run through Stanford Core NLP's tokenizers (Manning et al., 2014). The 840B Common Crawl corpus serves as a pre-training ground for GloVe's word vectors (Pennington et al., 2014). There are no embeddings for words that aren't in the Common Crawl corpus because the vocabulary is restricted to words that are already in the database. In order to avoid overfitting and poor performance, only results with fixed word embeddings are reported here. This is the maximum length of sequences that can be used in recurrent units, maxout layers, and linear layers during training. To begin, all LSTMs have zero initial states and random initialised parameters. During training, random initialization and optimization are applied to the sentinel vectors. We use a maxout pool size of 16 and a maximum number of iterations of 4 for the dynamic decoder. Dropout (Srivastava et al., 2014) is used to regularise our network during training, and ADAM is used to optimise the model (Kingma & Ba, 2014). Chainer is used to implement and train all models (Tokui et al., 2015).

VI.RESULTS

Two metrics are used for evaluation on the SQuAD dataset. Using the exact match score, you can determine how closely two answers predicted by a computer match those provided by human judgement in the real world (EM). The F1 score is based on how closely the predicted and actual answers sound. There may be multiple ground truth answers for a document-question pair, which is why EM and F1 are taken to be the maximum values across all of them. All document-question pairs are averaged together to arrive at the overall score. CodaLab hosts the official SQuAD evaluation. There is a public release of the training and development sets, but a private release of the test set.

When compared to the other models on the leaderboard, 3 is a significant improvement, the Dynamic Coattention Network's performance on the SQuAD dataset is shown in Table 1. This single-model DCN currently has the highest exact match and F1 scores of all single-model submissions (66.2 percent) as of the time of this writing. On the test data, the ensemble DCN comes in first place with an exact match rate of 71.6% and an F1 rate of 80.4%.

Multiple estimates of the answer span's start and end points can be made by the DCN, each time based on previous estimates. So, as shown in Figure 5, the model's ability to explore multiple plausible solutions is enhanced by doing so.

Fig. 8 shows an example of a situation in which the model makes an incorrect start point and an accurate end point guess in Question 1. After adjusting the start point in subsequent iterations, the model finally gets it right in iteration 3. The model gradually moves the end point in the direction of the correct word, similar to the probability mass.

Question 2 illustrates a situation in which both the start and end dates were incorrectly estimated. In the next iteration, the model reaches the correct answer.

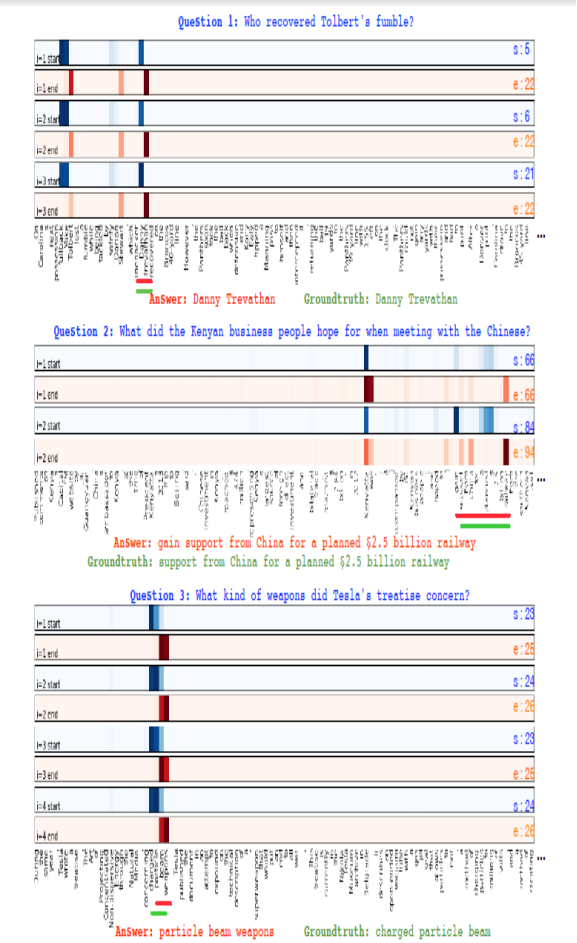

Fig 8. Examples of the The dynamic decoder generates start and end conditional distributions. Rows with odd numbers (blue) and even numbers (red) represent the start and end distributions respectively. The dynamic decoder's iteration number is indicated by the letter i. Darker areas indicate areas with a higher probability of mass. It is shown on the graph's right side, you'll see the word with the highest probability mass is offset. Underlined in red is the predicted span and underlined in green is the ground truth answer span.

However, despite several iterations, Question 3 shows that the Multiple local maxima cannot be chosen by the model. due to the model's dynamic nature, which allows the model to escape initial maximums associated with incorrect answers. Particle beam weapons and "charged particle beam weapons" are the only two answers that the model will ever accept. When trained with maximum iteration of 4, the model takes on average 2.7 iterations to converge upon an answer.

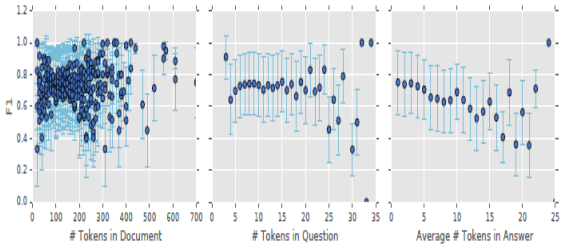

Fig 9. The blue dot shows the average F1 for the length of the line. An F1 at a given length has a standard deviation represented by the vertical bar shown here.

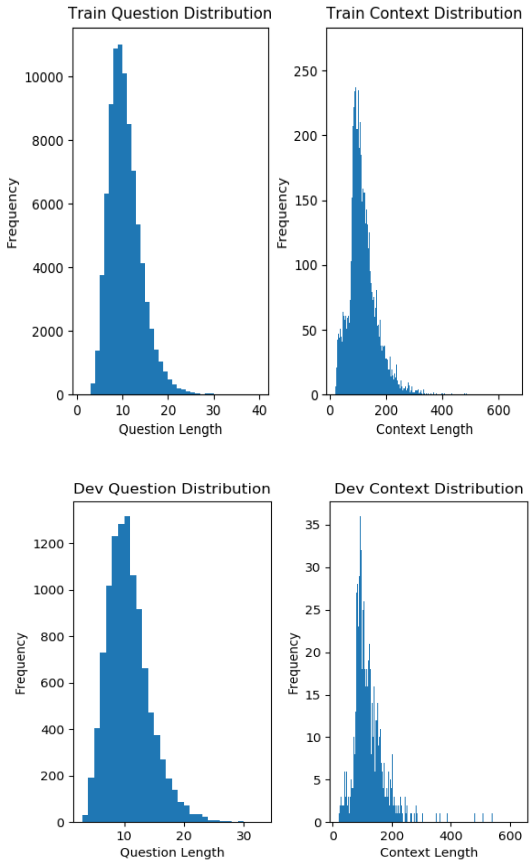

Fig 10. Results on both Dev set and Train set.

Our learning model will be trained and validated on the Stanford Question Answering Dataset (SQuAD). Questions on Wikipedia document texts are used to compile SQuAD, which is a collection of data on reading comprehension in which the answer to each question is an excerpt from the corresponding text. The training and validation sets for this project are 88581 and 5964 SQuAD entries, respectively. A test set of 10,654 data entries will be used to evaluate the learning model after sufficient training.

A. DROPOUT LAYER

The maxout activation layer has been removed from the highway network in our model. All highway network parameters except the last layer are dropped out with a keep probability of 0.9.

$$W^{(3)}\left[m_t^{(1)}; m_t^{(2)}\right] + b^{(3)}\, r$$
$$= tanh\left(W^{(D)}\left[h_i; u_{s_{i-1}}; u_{e_{i-1}}\right]\right) m_t^{(1)}$$
$$= W^{(1)}[u_t; r] + b^{(1)}\, m_t^{(2)} = W^{(2)} m_t^{(1)} + b^{(2)}$$

B. LOSS

It is calculated by averaging the loss of each prediction's cross-entropy loss.

$$loss_s = \frac{1}{N}\sum_{n=1}^{N} CE\left(y_{s_n}, \hat{y}_{s_n}\right) loss_e$$
$$= \frac{1}{N}\sum_{n=1}^{N} CE\left(y_{e_n}, \hat{y}_{e_n}\right)$$

To get an overall loss, simply add up each index's losses.

loss = $loss_s$ + $loss_e$

C .OPTIMIZATION

As annealing improves, the learning rate of our model decreases exponentially. Uses Adam's a stochastic gradient descent algorithm. Convex function Gradient Descent's output is the partial derivative of its input parameters. Scale: The more steep a slope, the greater the gradient. Gradient Descent is used iteratively for the parameters to have the best possible values in order to minimise the cost function's value.

D .MODEL ABLATION

An example of this model and its ablations in action is depicted in Fig. 11. To test the HMN decoder, we're experimenting with different pool sizes for the maxout layers, using MLPs instead of HMNs, and limiting the decoder iteration to one. Based on our results on our development set, we found that a deeper, more iterative decoder network consistently improves our model's performance. As the maximum number of iterations increases, performance improves, but after 4 iterations, there isn't much of an improvement. The encoder's F1 drops by 1.9 points when the coattention mechanism is replaced with an attention mechanism like Wang & Jiang (2016b) in equation 3. Using the coattention mechanism, the document and question sequences are encoded more effectively at the costs for a softmax calculation and a dot product. The appendix contains additional studies, such as how people perform without paying attention and how they perform on questions that require different types of reasoning.

Model	Dev EM	Dev F1
Dynamic Coattention Network (DCN)		
pool size 16 HMN	**65.4**	**75.6**
pool size 8 HMN	64.4	74.9
pool size 4 HMN	65.2	75.2
DCN with 2-layer MLP instead of HMN	63.8	74.4
DCN with single iteration decoder	63.7	74.0
DCN with Wang & Jiang (2016b) attention	63.7	73.7

Fig 11.Single model ablations on the development set

5.7 Performance across length

It's interesting to see how the DCN performs depending on the length of the document. As with neural machine translation, it's logical to assume that the model's performance will degrade with increasing sample size (Luong et al.,2015). There appears to be no noticeable performance degradation for documents and questions that are longer, as shown in Figure 6. Even in the case of very long documents, the encoder appears to be able to focus only on the most relevant text while overlooking the rest of the document. Our results show that longer answers have a negative impact on our overall performance. However, given the nature of the evaluation metric, this is logical. In other words, as the number of words grows, determining the correct word span becomes more difficult.

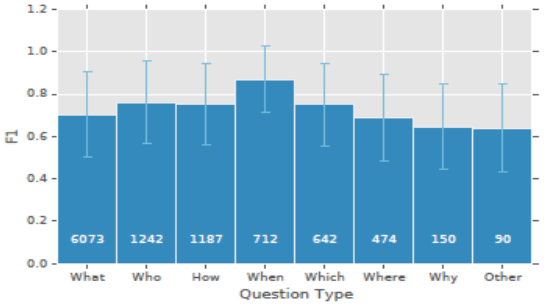

Fig 12. The DCN's ability to handle a variety of question types. There are a total of three bars for each question type, and their height represents the median score for that question type. The lower the number, the more frequently that particular question type has been encountered in the test data.

Performance across question type Analyzing the model's performance across different types of questions is a logical next step. The mean F1 of DCN (Wang & Jiang, 2016b) is higher than that of previous systems (Yu and Jiang, 2016) across all question types, as shown in Figure 7. "When" questions are easy for the DCN, but the more difficult "why" questions are beyond its capabilities.

Breakdown of F1 distribution DCN's performance is also highly bimodal, which is worth noting. It correctly predicts (100 percent F1) 62.2 percent of the examples on the development set and incorrectly predicts (0% F1) 16.3 percent of the examples. In other words, only 21.5 percent of the time does the model pick out partial answers. Some of the 0 percent F1 answers are shown in Appendix A.4; "answer type," such as "person" for "who" questions or "method" for "how" questions, tends to be correct when the model is incorrect, according to qualitative inspections of the answers.

E. ANALYSIS

Document text: In 1881, Tesla moved to Budapest to work under Ferenc Pusk's at a telegraph company, the Budapest Telephone Exchange. Upon arrival, Tesla realized that the company, then under construction, was not functional, so he worked as a draftsman in the Central Telegraph Office instead. Within a few months, the Budapest Telephone Exchange became functional and Tesla was allocated the **chief electrician** position.
Question: What position did Tesla accept at the exchange?
Correct answer: chief electrician
Predicted answer: chief ⟨unk⟩ position

Fig13. Sample prediction from dev evaluation.

To train the GloVe vectors, we used 6 billion tokens and 400,000 vocabulary from Wikipedia 2014 and Gigaword 5. The lack of a large vocabulary led to a large number of words being tokenized as . Table 3 shows how this impacted the accuracy of predicted answers. The Common Crawl GloVe vectors, which have a larger vocabulary, should be used instead to reduce the number of unknown words.

Document text: The First British Empire was based on mercantilism, and involved colonies and holdings primarily in North America, the Caribbean, and India. Its growth was reversed by the loss of the American colonies in **1776**. Britain made compensating gains in India, Australia, and in constructing an informal economic empire through control of trade and finance in Latin America after the independence of Spanish and Portuguese colonies about 1820.
Question: When did Great Britain lose its colonies in North America??
Correct answer: 1776
Predicted answer: 1776 . Britain made compensating gains in India , Australia , and in constructing an informal economic empire through control of trade and finance in Latin America after the independence of Spanish and Portuguese colonies about 1820

Fig 14. Sample prediction from dev evaluation

When reading the text in Table 4, it's hard to tell if the phrase "Its growth was reversed by the loss of the American colonies in 1776" is a reference to British colonies in America or something else. Learning models may also have difficulty correctly predicting the correct answer if they are unable to produce an accurate How the question and the document's text are interpreted together.

VII.CONCLUSION AND FUTURE WORK

This project's primary goal is to implement DCN for question answering which becomes very useful for us to search for answers. In natural language processing (NLP), Natural language processing and machine learning are both required for effective QA understanding and knowledge of the world around us. QA systems are designed to be aware of text similarity and to respond to questions that are asked in a natural language context. It is possible to use a variety of deep learning models to answer questions. To recover from incorrect answers, they are unable to use their single-pass nature. As a solution to this problem, we came up with the Dynamic Coattention Network (DCN). For the DCN, the question and the document are brought together in a way that emphasises their interdependence. Iterating over the possible answer spans, a dynamic pointing decoder is then used. An initial local maxima associated with incorrect answers can be recovered using this iterative procedure. This DCN ensemble model scores 80.4 percent on the Stanford question-answering dataset.

The Future work for this project is it can be implemented for Visual Question Answering, so that it can answer for visual question answers. The proposed learning model has a maximum input sequence length for documents and questions. In order to create a model that can accept inputs of any length, additional research should be carried out. Recurrent neural networks are difficult to parallelize because they are sequential in nature. Documents and questions can be encoded using convolutional neural networks, or other architectures. Another important issue for future work will be the application to other large datasets, such as VQA and SemEval-cQA of the model proposed.

REFERENCES

[1] An application of natural language inference via dependency tree mapping: a question-answering example. Research in Computational Linguistics (9) Scholars can find this information on Google Scholar.

[2] Heilman M., Smith Textual entailment, paraphrase, and answer recognition using N.A.Tree edit models. Proceedings of the NAACL HLT 2010 Conference on Human Language Technologies, June 2010 in Los Angeles, CA, USA. pp. 1011–1019. [Google Scholar].

[3] Khan M., Kuhn F., Malkhi D., Pandurangan G., and Talwar K. are the researchers who conducted this study. Probabilistic tree embeddings allow for efficient distributed approximation algorithms. The Journal of Distributed Computing. 2012;25(3):189–205. Cite this article as: 10.1007/s00446-012-0157

[4] An answer extraction method that uses the tree edit distance is proposed by C., and other researchers. pp. 858–867 in the proceedings of the NAACL HLT 2013 Conference in Atlanta, Georgia, in June 2013. Scholars can find this information on Google Scholar.

[5] For retrieving community question answers, we used neural networks to learn semantic representations through training. There are 93 pages in the journal Knowledge-Based Systems in 2016. Knosys.2015.11.002 [CrossRef] Scholars can find this information on Google Scholar.

[6] semantic parsing for single-relation questions: Yih, He, and Meek (2006a; 2006b). Pages 643–648 in the proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL 2014), June 2014, Baltimore, MD, USA. Scholars can find this information on Google Scholar.

[7] Wang P., Ji L., Yan J., and colleagues Multi-view learning question retrieval using a concept and attention-based CNN. Doi: 10.1145/3151957, ACM Transactions on Intelligent Systems and Technology, Volume 9, Issue 4, Pages 1–24, April 2018, [CrossRef] Scholars can find this information on Google Scholar.

[8] Wang D., Nyberg E. An answer sentence selection model based on a long short-term memory. International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL-IJCNLP; July 2015; Beijing, China. ACL-IJCNLP Proceedings: 53rd Annual Meeting. pp. 707–712. [Cross Ref] [Google Scholar].

[9] Analysis of discriminant characteristics and a similarity measure. Research in Pattern Recognition, 2014, 47(1): 359–367 Use this citation: 10.1016/j.patcog." [CrossRef] Scholars can find this information on Google Scholar.

[10] Deep and bidirectional representation learning for cross-modal retrieval by He, Xiang, Kang, Wang and Pan IEEE Transactions on Multimedia, Vol. 18, No. 7, July 2016, pp. 1363–1377, doi: 10.1109/tmm.2016.2558463. [Cross Ref] [Google Scholar].

[11] Alessandro Sordoni, Phillip Bachman, and Yoshua Bengio. Iterative alternating neural attention for machine reading. arXiv preprint arXiv:1606.02245, 2016.

[12] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. Journal of Machine Learning Research, 15(1):1929–1958, 2014.

[13] Rupesh K Srivastava, Klaus Greff, and Juergen Schmidhuber. Training very deep networks. In Advances in Neural Information Processing Systems 28, pp. 2377–2385, 2015. Seiya Tokui, Kenta Oono, Shohei Hido, and Justin Clayton. Chainer: a next-generation open-source framework for deep learning. In Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Twenty-ninth Annual Conference on Neural Information Processing Systems (NIPS), 2015.

[14] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In Advances in Neural Information Processing Systems, pp. 2692–2700, 2015.

[15] Hai Wang, Mohit Bansal, Kevin Gimpel, and David McAllester. Machine comprehension with syntax, frames, and semantics. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers), pp. 700–706. Association for Computational Linguistics, 2015.

[16] Shuohang Wang and Jing Jiang. Learning natural language inference with LSTM. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 1442–1451. Association for Computational Linguistics, 2016a.

[17] Shuohang Wang and Jing Jiang. Machine comprehension using match-LSTM and answer pointer. arXiv preprint arXiv:1608.07905, 2016b.

[18] Y. Yu, W. Zhang, K. Hasan, M. Yu, B. Xiang, and B. Zhou. End-to-End Reading Comprehension with Dynamic Answer Chunk Ranking. ArXiv e-prints, October 2016. Yang Yu, Wei Zhang, Kazi Hasan, Mo Yu, Bing Xiang, and Bowen Zhou. End-to-end answer chunk.

[19] C. Xiong, V. Zhong, and R. Socher, "Dynamic coattention networks for question answering," arXiv preprint arXiv:1611.01604, 2016.

[20] D. Chen, A. Fisch, J. Weston, and A. Bordes, "Reading wikipedia to answer open-domain questions," arXiv preprint arXiv:1704.00051, 2017.

[21] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," arXiv preprint arXiv:1409.0473, 2014.

[22] S. Sukhbaatar, J. Weston, R. Fergus et al., "End-to-end memory networks," in Advances in neural information processing systems, 2015, pp. 2440–2448.

[23] A. Graves, G. Wayne, and I. Danihelka, "Neural turing machines," arXiv preprint arXiv:1410.5401, 2014.

[24] C. Xiong, S. Merity, and R. Socher, "Dynamic memory networks for visual and textual question answering," in International Conference on Machine Learning, 2016, pp. 2397–2406.

[25] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "Squad: 100,000+ questions for machine comprehension of text," arXiv preprint arXiv:1606.05250, 2016.

[26] Y. Cui, Z. Chen, S. Wei, S. Wang, T. Liu, and G. Hu, "Attention-overattention neural networks for reading comprehension," arXiv preprint arXiv:1607.04423, 2016.

[27] G. A. Miller, "Wordnet: a lexical database for english," Communications of the ACM, vol. 38, no. 11, pp. 39–41, 1995.

[28] Z. Wu and M. Palmer, "Verbs semantics and lexical selection," in Proceedings of the 32nd annual meeting on Association for Computational Linguistics. Association for Computational Linguistics, 1994, pp. 133– 138.

[29] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural computation, vol. 9, no. 8, pp. 1735–1780, 1997. [12] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), 2014, pp. 1532–1543.

[30] M. Kusner, Y. Sun, N. Kolkin, and K. Weinberger, "From word embeddings to document distances," in International Conference on Machine Learning, 2015, pp. 957–966.

[31] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," arXiv preprint arXiv:1301.3781, 2013.

[32] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in Advances in neural information processing systems, 2013, pp. 3111–3119.