Extending a BiDAF model with DCN for Question Answering

Stanford CS224N Default Project (IID SQuAD track)

Michelle Xing
mxing621@stanford.edu

Hanna Yip
hannayip@stanford.edu

Anooshree Sengupta
anoosh22@stanford.edu

Abstract

Our goal in this project is to improve the performance of the Bidirectional Attention Flow (BiDAF) model for the NLP task of question answering on the SQuAD 2.0 dataset. To do this, we 1) integrate character-level embeddings into the baseline BiDAF model and 2) replace the default attention layer with a coattention layer. While adding character-level embeddings has shown to improve the baseline BiDAF model's EM and F1 scores substantially, their addition to the DCN model actually decreased its scores slightly. Moreover, transforming the BiDAF model into a Dynamic Coattention Network (DCN) decreased the model's performance. Thus, the best model architecture we found is BiDAF with character-level embeddings. Future work includes tuning hyperparameters, experimenting with data processing techniques, adding optimizations like the Adam optimizer, and exploring different forms of attention.

1 Introduction

As researchers have begun to tackle the task of machine Question Answering (QA), available datasets have grown from small, hand-labeled ones to larger ones like the Stanford-released SQuAD. SQuAD's magnitude introduces new challenges to the field of QA: it encompasses a diverse set of logic that models need to accommodate to be successful. Training and testing a model on the SQuAD dataset can reveal how well it 'understands' human text, and can indicate the degree to which it can help humans accomplish tasks associated with information retrieval.

For this project, we built a question answering system based on SQuAD. First, we successfully trained a baseline Bidirectional Attention Flow (BiDAF) model. After doing so, we endeavoured to improve the BiDAF model's performance—in terms of EM and F1 scores—by implementing character-level embeddings, and by incorporating the encoder, coattention, and dynamic decoder layers introduced in *Dynamic Coattention Networks for Question Answering* [4].

We chose to add character-level embeddings to the existing word-level embeddings in the baseline model because they would allow our model to capture the internal structure of words in addition to sentence composition, and provide our model with a framework with which to handle previously unseen words [5]. Adding character-level embeddings improved our model's EM and F1 scores by 2.07 and 2.435, respectively.

We then investigated the extent to which adapting layers from the Dynamic Coattention Network (DCN) improved our model's performance. The authors of *Dynamic Coattention Networks for Question Answering* claim that a coattention layer is capable of contextualizing a given question within its document, and that continuously predicting the beginning and ending of an answer span with a dynamic decoder prevents predictions from being trapped within local maxima. They also report that these changes yield significant F1 score increases [4].

However, after replacing the baseline model's attention layer with a coattention layer and adding an encoder and dynamic decoder to our model, we found that the DCN performed worse than the BiDAF, both without character embeddings—which yielded an EM score of 47.811 and F1 score of

47.811—and with character embeddings—which yielded an EM score of 47.489 and an F1 score of 47.513. These results are unsurprising: despite the advancements made by Xiong et al., their model's EM and F1 scores are still lower than that of the best-performing BiDAF model constructed by Seo et al. Moving forward, we would like to explore hyperparameter tuning and preprocessing techniques to improve the DCN's performance.

2 Related Work

The SQuAD dataset was first presented and freely released by Rajpurkar et al. in 2016 [2], who created the dataset by posing over 100,000 questions, whose answers were contained in predesignated Wikipedia articles, to crowdworkers. Rajpurkar et al. claimed that SQuAD's magnitude could allow it to serve as a large, foundational dataset that could accelerate and advance the field of QA, just as ImageNet did for object recognition and Penn Treebank did for syntactic parsing [2]. The researchers built SQuAD such that models must find an answer within an entire document instead of from a limited list of options; this combined with SQuAD's unprecedented scale and its diversity of question and answer pairings indicate that training and testing with it could close the gap between human and model performance in QA [2].

Rajpurkar et al. present a logistic regression model that relies heavily on lexicalized and dependency tree path features, and that has an F1 score of 51%, which is significantly less than the human performance score of 86.8% [2]. However, subsequent research has yielded models with greatly improved performance: the baseline BiDAF model we use is based on the work done by Seo et al. in *Bi-Directional Attention Flow for Machine Comprehension*. Instead of using on an attention layer that summarizes context information in a fixed-size vector early in training, Seo et al. implement a bidirectional attention layer that 1) independently computes Context2Query and Query2Context attention at each time step, and 2) allows attention computations from every time step to flow through every layer of the model to preserve information. This technique achieves an F1 score of 81.1%, approaching human comprehension [3].

Seo et al. include character-level, word-level, and contextual embeddings [3], but our baseline model did not contain character-level embeddings. In order to replicate the results from Seo et al., we also followed the technique proposed by Kim et al. in *Convolutional Neural Networks for Sentence Classification*. Kim et al. demonstrated in their 2014 paper that passing character vectors through a CNN then max-pooling the CNN's output improves model performance in a wide range of tasks, from QA to sentiment analysis [7].

Finally, our DCN implementation is based on *Dynamic Coattention Networks for Question Answering*, where Xiong et al. propose a coattention method that analyzes user-posed questions and reference documents simultaneously, and a dynamic pointer decoder that constantly updates predictions for the start and end of a answer through iterative calculations and a Highway Maxout Network [4]. Xiong et al. report an F1 score of 80.4%.

3 Approach

3.1 Baseline Model

The provided baseline model, based on a Bidirectional Attention Flow (BiDAF) architecture [3], was downloaded from github (**https:://github.com/mingg/squad.git**).

The baseline model consists of 5 layers: the word embedding layer, the phrase or context embedding layer, the attention layer, the modeling layer, and the output layer where the phrase embedding layer and the modeling layer are bidirectional RNNs (see Figure 1 in Appendex). Note that the BiDAF model diverges from unidirectional attention models by representing the context of a query at different granularity levels, and by performing both Question-to-Context and Context-to-Question attention.

3.2 Character-Level Embeddings

For our first improvement to the model, we implemented character-level embeddings from scratch. We chose this because character-level embeddings enable us to capture information about the internal structure of words–their morphology–and handle out-of-vocabulary words more effectively [3].

The baseline model represents each word in the vocabulary with a GLoVe word embedding of length 300 [5]. For each word, we lookup the individual character embeddings of length 64 and combine them into a single 64 length vector using a 2-dimensional convolutional neural network (CNN). Then, we perform a max pooling. Finally, we concatenate the word embedding vectors with their respective character embedding vectors to obtain final embedding vectors of length 364 (see Figures 2 and 4 in Appendix).

3.3 Dynamic Coattention Network

For our next enhancement, we implemented from scratch the DCN encoder, coattention layer, and dynamic pointing decoder, which are based on the architecture of the Dynamic Coattention Network (DCN) in the original paper [4]. Integrating these into the given baseline BiDAF model, we end up replacing the attention flow layer, modeling layer, and the output layer of the baseline BiDAF model (see Figure 3 in Appendix). Thus, our final DCN architecture consists of 5 layers: the embedding layer, the phrase embedding layer, the DCN encoder layer, the coattention layer, and the dynamic pointing decoder layer.

3.4 DCN Encoder Layer

The DCN Encoder takes as input from the phrase embedding layer the context hidden states $[c_1, ..., c_N] \in \mathbb{R}^l$ and question hidden states $[q_1, ..., q_M] \in \mathbb{R}^l$.

We first calculate Q' which is the final projected representation for the question hidden states by applying a linear layer then a tanh nonlinearity.

$$Q' = tanh(W * q_{enc} + b) = [q'_1, ..., q'_n] \tag{1}$$

Then, we concatenate sentinel vectors $q_\emptyset$ and $c_\emptyset$ to the context and question hidden states. This allows the model to not attend to any particular word in the input.

$$C = [c_1, ..., c_n, c_\emptyset] \tag{2}$$

$$Q' = [q'_1, ..., q'_n, q'_\emptyset] \tag{3}$$

Finally, using the final context hidden states C and question hidden states Q' shown above, we compute the affinity matrix L which contains affinity scores corresponding to all pairs of document and question words.

$$L = C^T Q' \tag{4}$$

3.5 DCN Coattention Layer

The DCN Coattention takes as input from the DCN Encoder layer the affinity matrix L, context hidden states C, and new question hidden states Q'.

On the high level, the coattention encoder simultaneously attends to the question and document and then fuses the attention contexts.

We normalize L in both dimensions to get the attention weights across the document for each word in the question (α) and weights across the question for each word in the document (β). We also obtain the Context-to-Question (C2Q) attention outputs A and Question-to-Context (Q2C) attention outputs B.

$$\alpha = softmax(L) \tag{5}$$

$$A = \alpha Q' \tag{6}$$

$$\beta = softmax(L^T) \tag{7}$$

$$B = \beta C \tag{8}$$

Next, we calculate the second-level attention outputs S by taking the weighted sum of Q2C attention outputs B with the C2Q weights α:

$$S = \alpha B \tag{9}$$

Finally, we concatenate the second level attention outputs S with the first-level C2Q outputs A and feed it through a bidirectional LSTM. The output U is the coattention encoding.

$$U = biLSTM([S; A]) \tag{10}$$

3.6 Dynamic Pointing Decoder

The Dynamic Pointing Decoder takes as input from the DCN Coattention layer the coattention encodings U. The goal of the Dynamic Pointing Decoder is to predict the start and end indices of the answer in the input context document. Consistent with the original DCN paper, the decoder performs many iterations to select an answer span by alternating between predicting the start point and end point of the span [4].

In each iteration, the decoder updates the hidden state of the LSTM, the estimate of the start position, and estimate of the end position. It does so by taking into account the coattention encoding corresponding to current estimates of the start and end positions and uses a Highway Maxout Network (HMN), a multilayer neural network, to produce new estimates.

We start by initializing the start and end positions in the coattention encoding U: s_{prev} and e_{prev}. These positions correspond to representations $u_{s_{prev}}$ and $u_{e_{prev}}$. We also initialize the hidden state of the LSTM h_i.

In the beginning of each iteration, we update the hidden state by inputting the concatenation of $u_{s_{prev}}$ and $u_{e_{prev}}$ into an LSTM:

$$h_{current} = LSTM([u_{s_{prev}}; u_{e_{prev}}], h_{prev}) \tag{11}$$

Then we initialize tensors of the start scores αs and end scores βs for each word in the document. We iterate through the document twice, once to calculate the score α_t and once to calculate the score β_t corresponding to the t^{th} word in the document. We calculate both these scores using an HMN, and then concatenate them to our growing list of scores. The position scores correspond to the likelihood of each word being the actual start or end position of the answer span.

$$\alpha_t = HMN_{alpha}(u_t, h_{current}, u_{s_{prev}}, u_{e_{prev}}) \tag{12}$$

$$\beta_t = HMN_{beta}(u_t, h_{current}, u_{s_{prev}}, u_{e_{prev}}) \tag{13}$$

We calculate new estimates for the start and end positions by taking the argmax of αs and βs.

At the end of the iteration, we update s_{prev}, e_{prev}, $u_{s_{prev}}$ and $u_{e_{prev}}$, and begin a new iteration.

Finally, after all the iterations, we perform a masked softmax on the αs and βs and output our results.

3.7 DCN Loss

We use a softmax cross entropy loss on both the start position scores and the end position scores outputted by the HMN in the last iteration of the dynamic pointing decoder.

$$loss_{start} = CrossEntropyLoss(y_{start|pred}, y_{start|actual}) \tag{14}$$

$$loss_{end} = CrossEntropyLoss(y_{end|pred}, y_{end|actual}) \tag{15}$$

The overall loss is the sum of the losses.

$$loss = loss_{start} + loss_{end} \tag{16}$$

4 Experiments

4.1 Data

For our dataset, we use Stanford Question Answering Dataset (SQuAD 2.0) [2], which includes both answerable and unanswerable questions. The official SQuAD dataset is split into train, dev, and test sets [2]. We use the official SQuAD train set as our train set, half of the official SQuAD dev set as our dev set, and the remaining half of the official SQuAD dev set as our test set. This leaves us with 129,941 examples (context, question, answer) triples for our train set, 6078 examples for our dev set, and 5915 examples for our test set. To preprocess the datasets, we convert the raw data in the files to example and evaluation sets. For the training set, we obtain word/character vocabularies which contain their pretrained GloVe embeddings [5]. For the dev set, we drop examples that are too long in either context, question, or answer size.

Context	In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under gravity. The main forms of precipitation include drizzle, rain, sleet, snow, graupel and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals within a cloud. Short, intense periods of rain in scattered locations are called "showers".
Question	Where do water droplets collide with ice crystals to form precipitation?
Answer	within a cloud

Table 1: Example of a (context, question, answer) triplet data entry of SQuAD

4.2 Evaluation method

We use the Exact Match (EM) score, which calculates the exact match between the model's predicted answer and the ground truth answer, the F1 score, which quantifies the overlap between them. Both scores are official SQuAD 2.0 evaluation metrics.

4.3 Experimental details

Experiments were run on Microsoft Azure.

As a benchmark, we trained the given baseline model, and evaluated it on the dev and test sets. Then we trained the given baseline model + character-level embeddings and evaluated it.

We then trained our DCN model with and without character-level embeddings separately using a maxout pool size of 16 and running the dynamic pointing decoder for 4 iterations.

For all experiments, a learning rate of 0.5 and a dropout probability of 0.2 was used. We processed data in batches of 64. The models were trained for 30 epochs each.

4.4 Results

Model	EM Score	F1 Score
Baseline BiDAF	56.895	60.058
Baseline BiDAF + Character-level Embeddings	58.965	62.493
Dynamic Coattention Network	52.092	52.092
Dynamic Coattention Network + Character-level Embeddings	51.672	51.708

Table 2: Model performance scores on Dev Set

The performance of the DCN was worse than the baseline BiDAF model, achieving on the dev set an EM score that is 4.803 less and a F1 score that is 7.966 less, which mirrors the relative decrease in scores on the dev set by the state of the art BiDAF model (EM: 68.0, F1: 77.3) [3] and the state of the art DCN model (EM: 65.4, F1: 75.6) [4]. However, the decrease that we saw in our implementation is larger than the decrease in the state of the art models.

Model	EM Score	F1 Score
Baseline BiDAF + Character-level Embeddings	59.51	63.17
Dynamic Coattention Network	47.811	47.811
Dynamic Coattention Network + Character-level Embeddings	47.489	47.513

Table 3: Model performance scores on Test Set

We see that we achieved the best results when adding character-level embeddings to the baseline model, improving the EM score from 56.895 to 58.965 and the F1 score from 60.058 to 62.493 (Table 3), resulting in 2.07 and 2.435 increases, respectively. Another aspect to note is when examining the performance of our best model, we notice that the model actually did better on the test set (EM: 59.510, F1: 63.170) than it did on the dev set (EM: 58.965, F1: 62.493). This is an indication that our model is striking a good balance between variance and bias because it's generalizing well to unseen examples in the test set. However, this makes sense because both the dev and the test set are apportioned from the official SQuAD dev set.

While adding character-level embeddings improved the baseline BiDAF model, adding character-level embeddings to the DCN model did not improve the models performance. In fact, we see that the character-level embeddings caused the DCN model to perform slightly worse. This can be explained in that the original DCN paper actually foregoes character-level embeddings and instead uses only word embeddings. This may be potentially because character-level embeddings further increases the complexity of the embedding and attention layers while a coattention architecture tries to extract the right context.

These results tell us that our approach of extending the baseline BiDAF model to a DCN is a good effort in exploring the dynamics and implications of adding character-level embeddings to the embedding layers of different Question Answering architectures and how the BiDAF model and the DCN model differ.

5 Analysis

Below are examples of when the baseline model with character embeddings and the DCN model succeed and fail.

Context	The simplest valve gears give events of fixed length during the engine cycle and often make the engine rotate in only one direction. Most however have a reversing mechanism which additionally can provide means for saving steam as speed and momentum are gained by gradually "shortening the cutoff" or rather, shortening the admission event; this in turn proportionately lengthens the expansion period. However, as one and the same valve usually controls both steam flows, a short cutoff at admission adversely affects the exhaust and compression periods which should ideally always be kept fairly constant; if the exhaust event is too brief, the totality of the exhaust steam cannot evacuate the cylinder, choking it and giving excessive compression ("kick back").[citation needed]
Question	What can the exhaust steam not fully do when the exhaust event is insufficiently long?
Prediction	evacuate the cylinder
Answer	evacuate the cylinder

Table 4: Successful example for baseline model with character embeddings succeeds

The baseline model with character embeddings successfully captured the answer to a multi-word answer (Table 4), but failed to predict correctly when there are more than one word in the context that matches the correct answer in terms of part of speech or format (Table 5). In the answer to a "What King...", at first glance, Louis XII and Henry IV both seem like valid answers. But the model fails to recognize the nuance in the rest of the question. This could be due to the fact that the BiDAF model does not perform second-level attention, as the DCN model does, and thus does not capture as

much context in larger documents, or the fact that the BiDAF model is more susceptible to being trapped in local maxima when making predictions.

Context	By 1620 the Huguenots were on the defensive, and the government increasingly applied pressure. A series of three small civil wars known as the Huguenot rebellions broke out, mainly in southwestern France, between 1621 and 1629. revolted against royal authority. The uprising occurred a decade following the death of Henry IV, a Huguenot before converting to Catholicism, who had protected Protestants through the Edict of Nantes. His successor Louis XIII, under the regency of his Italian Catholic mother Marie de' Medici, became more intolerant of Protestantism. The Huguenots respond by establishing independent political and military structures, establishing diplomatic contacts with foreign powers, and openly revolting against central power. The rebellions were implacably suppressed by the French Crown.[citation needed]
Question	What King and former Huguenot looked out for the welfare of the group?
Prediction	Louis XIII
Answer	Henry IV

Table 5: Unsuccessful example for baseline model with character embeddings

Context	Throughout the 18th century, Enlightenment ideas of the power of reason and free will became widespread among Congregationalist ministers, putting those ministers and their congregations in tension with more traditionalist, Calvinist parties.:1–4 When the Hollis Professor of Divinity David Tappan died in 1803 and the president of Harvard Joseph Willard died a year later, in 1804, a struggle broke out over their replacements. Henry Ware was elected to the chair in 1805, and the liberal Samuel Webber was appointed to the presidency of Harvard two years later, which signaled the changing of the tide from the dominance of traditional ideas at Harvard to the dominance of liberal, Arminian ideas (defined by traditionalists as Unitarian ideas).:4–5:24
Question	In what year was Henry Ware elected to chair?
Prediction	1805
Answer	1805

Table 6: Successful example for DCN

Context	Today, Warsaw has some of the best medical facilities in Poland and East-Central Europe. The city is home to the Children's Memorial Health Institute (CMHI), the highest-reference hospital in all of Poland, as well as an active research and education center. While the Maria Skłodowska-Curie Institute of Oncology it is one of the largest and most modern oncological institutions in Europe. The clinical section is located in a 10-floor building with 700 beds, 10 operating theatres, an intensive care unit, several diagnostic departments as well as an outpatient clinic. The infrastructure has developed a lot over the past years.
Question	What is the highest reference hospital in all of Germany?
Prediction	700
Answer	N/A

Table 7: Unsuccessful example for DCN

It appears that the DCN model performs well on one-word answers (Table 6), but oftentimes makes a prediction even if the answer is not found in the context (Table 7). When the question asked for the "highest reference hospital in all of Germany", the DCN model seemed to just search for a large number in the context and predict that as the answer regardless of the meaning of that number. This could be attributed to the fact that the DCN is "highly bimodal": Xiong et al. reported that it is either right or wrong for the majority of questions posed and rarely provides a partial or incomplete answer [4]. Our results mirror these findings and suggest that the model may be providing the wrong answer

type (ex. a number instead of a word) for the sake of providing an output, indicating that, despite the coattention matrix, the DCN model is not capturing the diversity of question and context types in the SQuAD dataset.

6 Conclusion

We successfully implemented three models: the baseline BiDAF with added character-level embeddings, the DCN model, and DCN model with character-level embeddings.

The baseline with character embeddings improved upon the baseline EM and F1 scores, however both DCN models performed worse than the baseline BiDAF. It appears that the DCN encoder, coattention layer, and the decoder are not necessarily compatible with the RNN Encoder used in the context embedding layer used in the baseline model. Furthermore, it's harder to train a the complicated architecture of coattention and extract appropriate information given the relatively small amount of data. Adding character-level embeddings to an under-performing DCN model intuitively won't improve results.

Despite the challenges in implementing a high-performing DCN model, this project allowed us to transform our intuition behind the deep learning ideas of layers, encoders, decoders, and attention into a fully functioning model, helping us understand how to translate every equation, diagram, and flow into code.

There are many avenues for future work. We could go beyond the paper's implementation and examine the effect of changing the preprocessing techniques and tuning hyperparameters such as the learning rate, dropout probability, max sequence length, the hidden state size, the L2 weight decay, the number of epochs to train for, the maximum number of iterations for the dynamic encoder, and the max pool size. We could also add optimizations such as the Adam optimizer which implements an adaptive learning rate. Finally, we could explore different types of attention.

7 References

[1] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for SQuAD. In *Association for Computational Linguistics* (ACL), 2018.

[2] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machines comprehension of text. In *Association for Computational Linguistics (ACL)*, 2016.

[3] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. arXiv preprint arXiv:1611.01603, 2016.

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

[5] 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)*, 2014.

[6] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. arXiv preprint arXiv:1505.00387, 2015.

[7] Yoon Kim. Convolutional neural networks for sentence classification. In *EMNLP*, 2014

8 Appendix

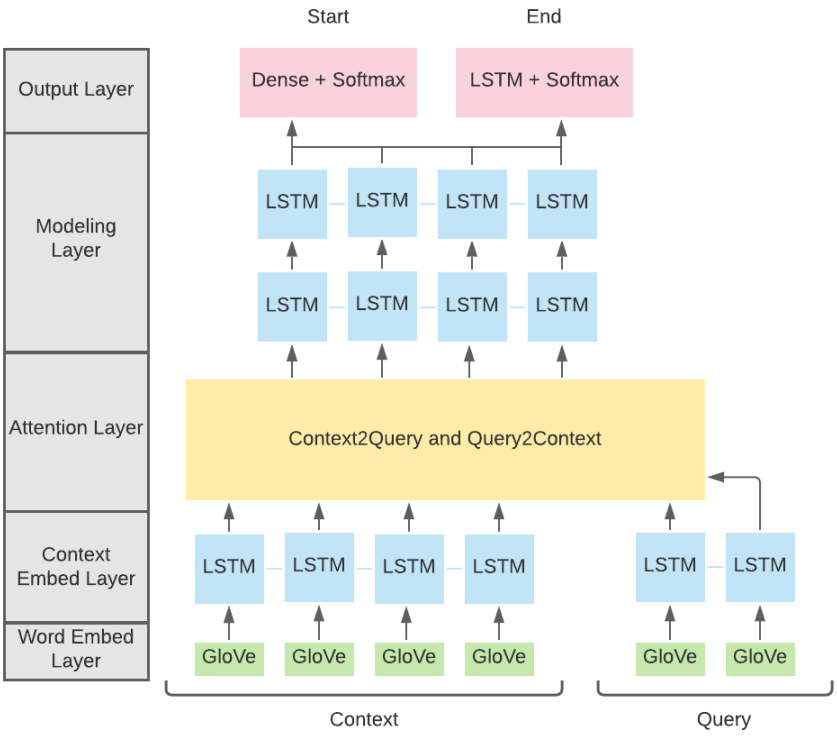

Figure 1: Baseline BiDAF Model Architecture

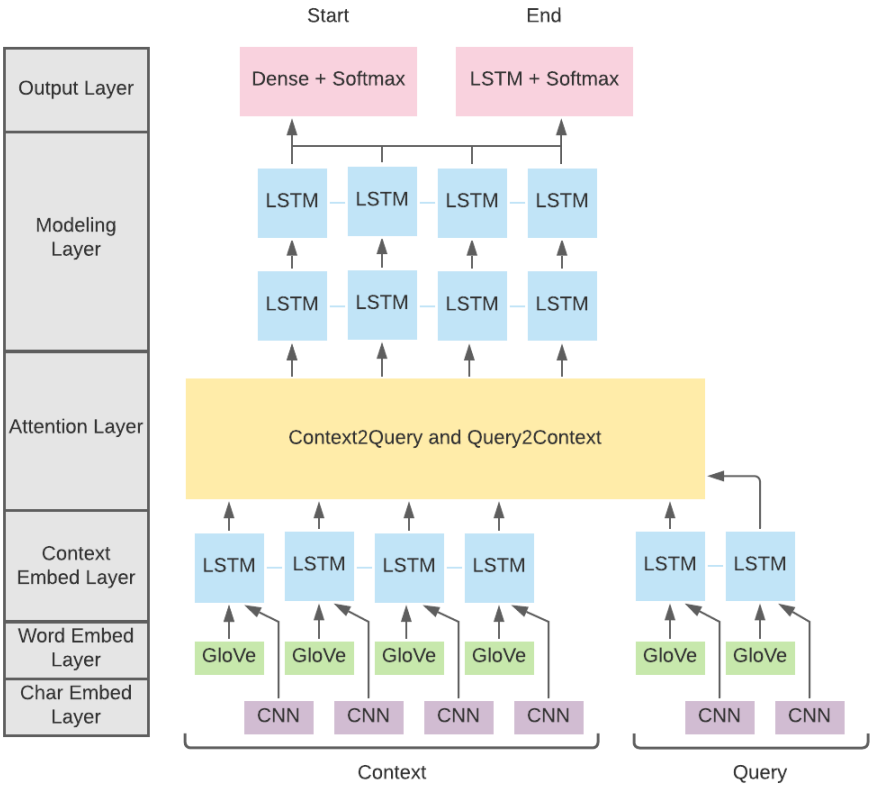

Figure 2: BiDAF with Character Embeddings Model Architecture

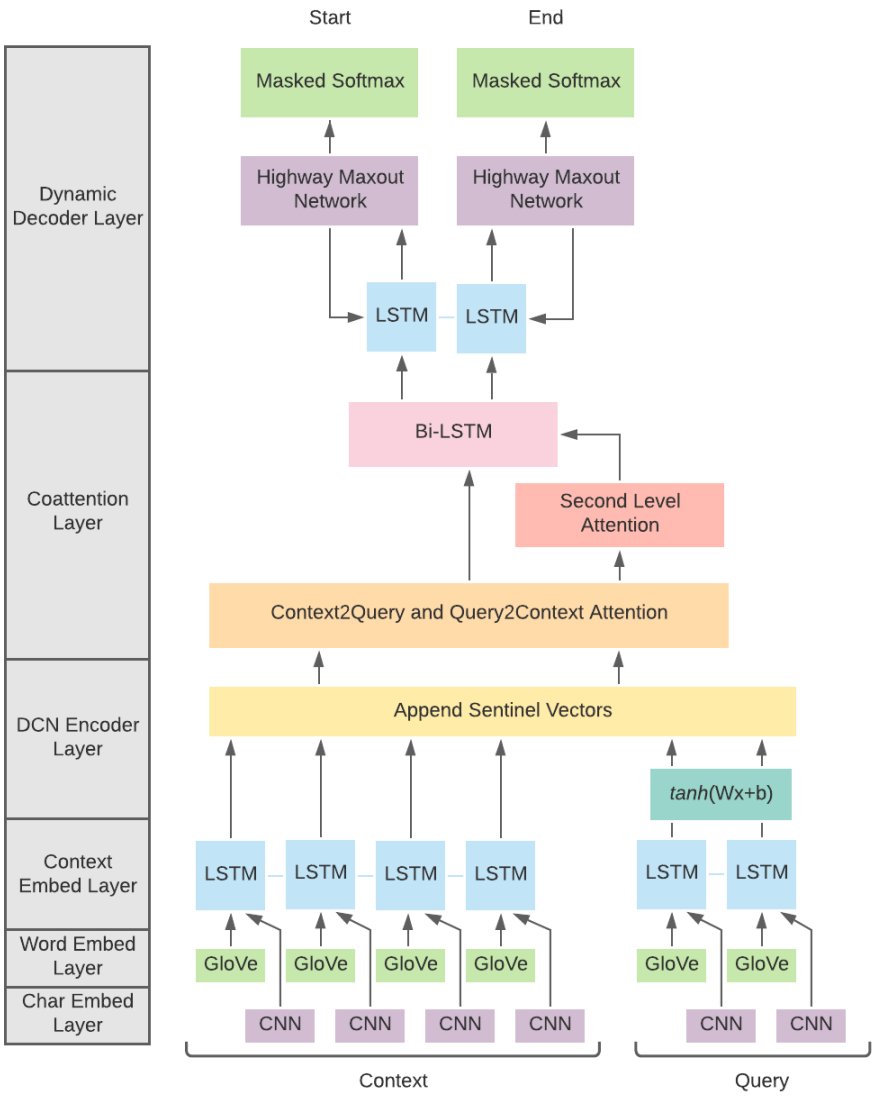

Figure 3: Dynamic Coattention Network Architecture

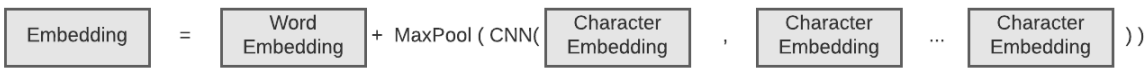

Figure 4: Construction of Full Embeddings