Iran University of Science and Technology
Deep Learning (Spring 2020)

DYNAMIC COATTENTION NETWORK FOR QUESTION ANSWERING

Mahsa Razavi, Houman Mehrafarin
Department of Computer Engineering
Iran University of Science
and Technology
Mahsasadat_razavi@comp.iust.ac.ir, H_mehrafarin@comp.iust.ac

Abstract

Several deep learning models have been proposed for question answering. However, due to their single-pass nature, they have no way to recover from local maxima corresponding to incorrect answers. The Dynamic Coattention Network [1] first fuses co-dependent representations of the question and the document in order to focus on relevant parts of both. Then dynamic pointing decoder iterates over potential answer spans. This iterative procedure enables the model to recover from initial local maxima corresponding to incorrect answers.

1 Introduction

Question answering (QA) systems are expecting strong increases in daily use now and in near future. This formulates a machine learning problem where the model receives a question and a passage and is tasked with answering the question using the passage. The training data consists of (question, paragraph, answer span) triplets. Due to the nature of the task, combining the information contained in the passage with the question posed is paramount to achieve good performance.

We implemented the Dynamic Coattention Network (DCN)[1], illustrated in Fig. 1, an end-to-end neural network for question answering. The model consists of a coattentive encoder that captures the interactions between the question and the document, as well as a dynamic pointing decoder that alternates between estimating the start and end of the answer span. To evaluate our model, we will use F1 score.

[1] Code available on github at https://github.com/hmehrafarin/DCN.git

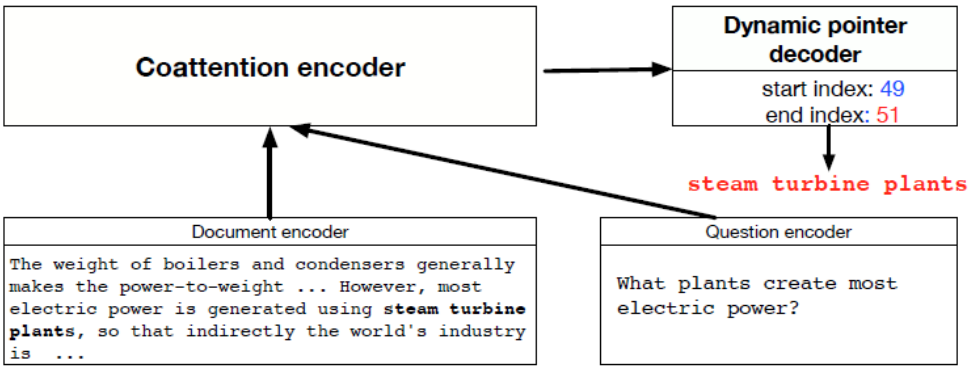


Figure 1 Overview of the Dynamic Coattention Network.

2 Related work/Background

Various neural models have been used to implement question answering systems. We discuss two of them here. It is worth noting that most successful models have used long short term memory units (LSTMs) to capture interactions over long sequences of words (necessary for context paragraphs that may run for hundreds of words.) Successful models also use some sort of attention mechanism, to capture relevance of words in the question and context paragraph as an intermediate step to answering the question.

2.1 Seq2seq Model

Our baseline is a simple seq2seq QA model. We need to know whether our attention mechanism improves the simple seq2seq QA model, thus we decided to compare the DCN model to a QA model consisting of 2 Bi-Directional LSTMs (one for answer spans and another for documents).

2.2 Dynamic Chunk Reader

Yang Yu et al[2]. use a similar attention mechanism (they calculate attention values for each pair of words in the question and context paragraph) but they aggressively filter possible answers by selecting an answer from candidate "chunks". The candidate chunks are determined by linguistic parsing of the roles (e.g. parts of speech) that the various words play in a sentence. Considering only these candidates substantially increases the training rate but leads to some correct answers not even being considered. But the authors report the correct answer was among the candidate chunks 92% of the time.

3 Proposed method

3.1 Preprocessing

The original data is available in Jason format. We first extract questions, document, answer and answer span files from the original json file. Then we used GloVe embedding layer with dimension of 100 to create vector of ids.

3.2 Model Architecture

Our model is purely based on the paper "Dynamic Coattention Networks for Question Answering" with some changes in the base model. For implementing this project, we used TensorFlow 2.0 and model sub classing for creating the Encoder, Coattention Encoder, Decoder and Highway Maxout Network (HMN). This part will be divided into 4 separate parts to cover each part independently.

3.2.1 Encoder

In this part we encode the input questions and documents with 2 different embedding layers with pre-trained GloVe embedding weights. To keep the model's training simple, the training of these two layers are set to False. We then pass these 2 embedded inputs to an LSTM and pass the question output of the LSTM to a dense layer to get $\acute{Q}$. Now we are ready to move on to the next layer called the coattention layer.

3.2.2 Coattention Encoder

This part is rather important as it is the main part of this paper and model. The basic idea behind this layer is to attend to both questions and documents simultaneously time rather than attending to each separately. This lets the model to comprehend both questions and documents at the same time. Now we will proceed to the implementation. We first create an affinity matrix by multiplying the document states from the LSTM encoder and $\acute{Q}$. The affinity matrix is then used to create attention weights A^Q, A^D by applying softmax on the affinity matrix. Now we have the attention vectors of both documents and questions. Thus, we can easily create attention context C^Q by multiplying A^Q and the document states. C^Q is actually attending to the documents using the attention weights from the questions. We then concatenate C^Q with $\acute{Q}$ and apply the same strategy to the concatenation output, only this time with the attention weights from the documents. As of now, we have attended to both questions and documents simultaneously, all we need is to pass them to a Bi-LSTM layer and we would be ready for decoding. Below is the summary of this part.

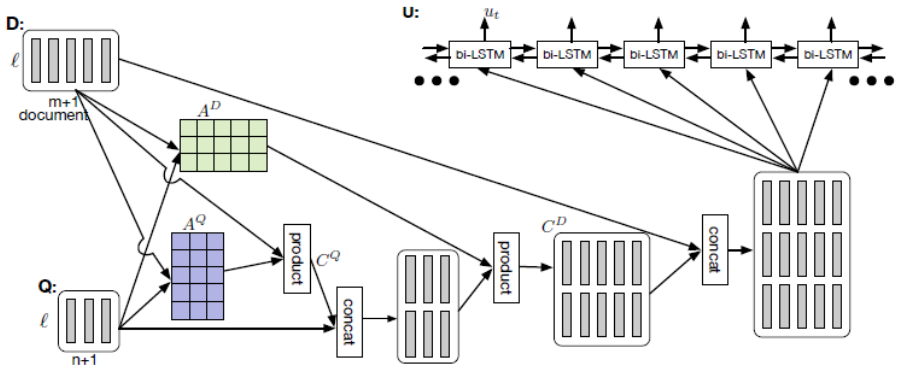


Figure 2 Coattention Encoder

3.2.3 Decoder

Due to the nature of SQuaD dataset, a Question-Answer pair, there may exist several intuitive answer spans within the document, each corresponding to a local maxima. This is solved by iterating between the prediction of start and end indexes in the document. This iterative procedure allows us to recover from the local maximum and lead us to the optimum answer. The decoder takes the coattention encoder states as its input and estimates the current start and end position. This is done by passing by the current coattention encoder states to a GRU cell. This is the initial estimation of start and end positions. This process repeats itself four times, to give the model that dynamic positioning behavior. In each iteration the GRU cell creates a cell state h_i, h_i is then passed to the Highway Maxout Network alongside two other parameters, starting position encoding and the ending position encoding. This step is for predicting the next starting and ending position encodings. Highway Maxout Network layer is used within the decoder and will be covered in the following part.

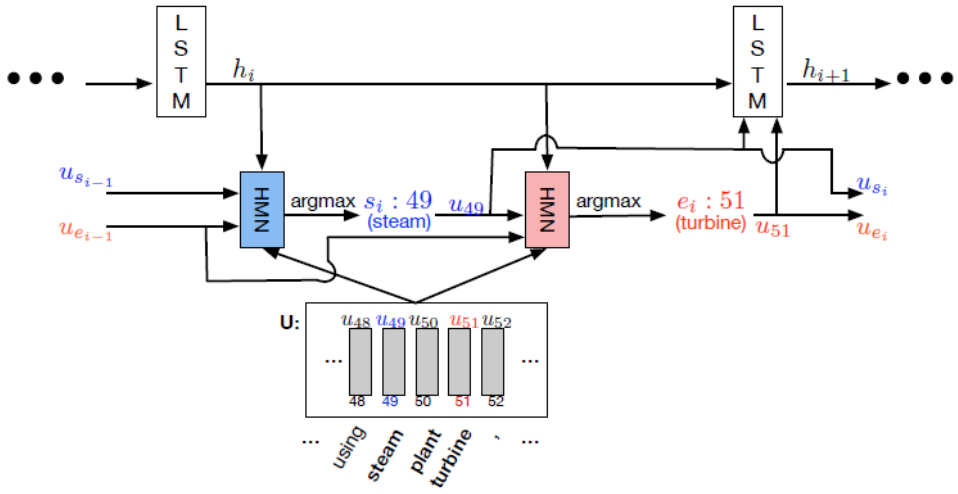


Figure 3 Dynamic Decoder

3.2.4 Highway Maxout Network

This part is proposed by the paper and is combined of two parts, the Highway Network[3] and Maxout Networks. Its soul purpose is to predict $\alpha_i$and β_i, which is the starting and ending position respectively. There are two separate networks with same architectures for predicting α_i and β_i. In this layer, we first create a vector by concatenating GRU cell state, h_i, start and end position encodings. Highway Maxout Network basically consists of 4 dense layers, however we have implemented these dense layers manually and did not use tf.keras.layers.Dense to better convey the purpose of the authors. Firstly, we pass the created vector to a dense layer with hyperbolic tangent activation function. Secondly, we pass the concatenation of the first layer's output and GRU cell state to another dense layer, however, this time we do not use any activation functions because we need to apply reduce max on the output. Thirdly, we pass the previous output to another dense layer and apply reduce max on the output. Lastly, we concatenate the outputs of the second and third layer and pass them to a final dense layer. The last layer chooses the index with maximum probability to be the star/end position.

The intuition behind using such model is that the QA task consists of multiple question types and document topics. These variations may require different models to estimate the answer span. Maxout provides a simple and effective way to pool across multiple model variations.

4 Results

The data used in this project is the SQuaD[2] 1.1 dataset. The reason behind this choice is that squad 2.0 offers unanswered questions, which the model has to decide not to answer them. However, our model is basically a question answering model with a different attention mechanism and it is not focused on this specific task. As result we keep our model simple and train it with SQuaD 1.1 dataset.
The model has been trained on a small fraction of the original SQuaD dataset to avoid resource exhaustion. The metrics for this model is MSE to show the distance between the true indexes and the predicted indexes of the answer span.

The following is the graph provided by TensorBoard:

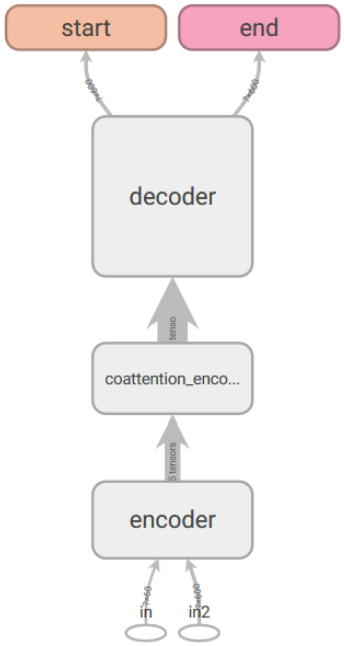
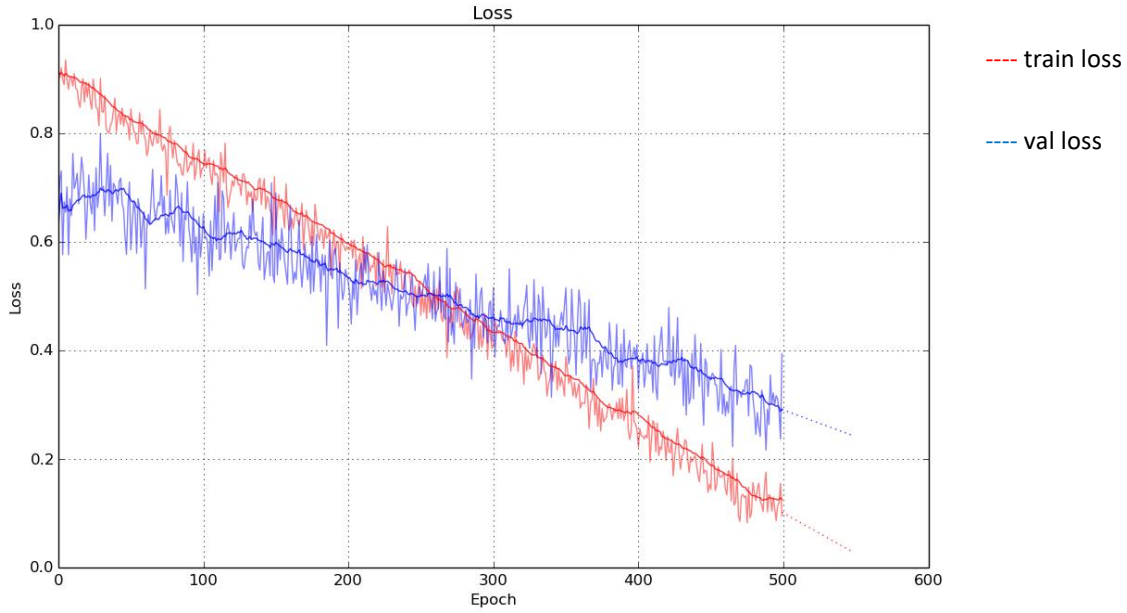


Figure 4 Model Graph

Time was a major obstacle for us in training, improving, and testing our models. Our model took hours per epoch to train on the GPU, which made it extremely difficult to test incremental improvements or even tune hyperparameters.

[2] https://www.kaggle.com/stanfordu/stanford-question-answering-dataset

With that said, the greatest limiting factor of our experimentation was out-of-memory errors. In hindsight, we should have used smaller GloVe vectors or limited the context paragraph size to only a few hundred, to be able to run much larger batches.



Model	F1-score
DCN	70.3
Baseline (Seq2Seq)	61.7

5 Discussion

We implemented the Dynamic Coattention Network, an end-to-end neural network architecture for question answering. The DCN consists of a coattention encoder which learns co-dependent representations of the question and of the document, and a dynamic decoder which iteratively estimates the answer span. We conclude that the iterative nature of the model allows it to recover from initial local maxima corresponding to incorrect predictions. According to the main paper, on the SQuAD dataset, the DCN achieves the state of the art results at 75.9% F1 with a single model. The DCN significantly outperforms all other models.

The results show that our DCN model outperforms the baseline model. As mentioned earlier the DCN is capable of finding the optimum answer to the question and recover from the local maxima. Our baseline's performance is highly dependent on the length of the document. The longer the document, the worse the results become. However, due to DCN's attention mechanism, the performance would be the same regardless of the document's length. To be more specific, the

Coattention encoder is robust to the document's length because it focuses on parts of the document, which are relevant to the question. As a result, the document's length would not be an issue to this model. Our dataset consists of many types of questions and documents, questions with "when", "what" and the more complex type "why". The results show us that, these different question types are handled pretty well. This is mostly because we use the HMN network in our model. As mentioned earlier the HMN handles different question and document types with different models and pooling between them. Moreover, increasing the pool size of the Highway Maxout Network, improves the results. Our result is with pool size of 16.

6 Contributions

All group members collaborated equally on all parts of the project. We spent dozens of hours together working on this project, and we all contributed extensively to the design, coding, and debugging processes.

References

[1] Xiong Caiming, Zhong Victor, Socher Richard (2017) Dynamic coattention networks for question answering. In 5th International Conference on Learning Representations, ICLR 2017.

[2] Yu, et al. "End-to-End Answer Chunk Extraction and Ranking for Reading Comprehension."IBM Watson.2016.https://arxiv.org/pdf/1610.09996.pdf

[3] Srivastava, et al. "Highway Networks."The Swiss AI Lab IDSIA.2015.https://arxiv.org/pdf/1505.00387.pdf