SQuAD Question Answering Dataset: CS224N Assn 4

Xin Jin
Department of Economics
Stanford University
Stanford, CA 94305
xinj@stanford.edu

Milind Rao
Electrical Engineering
Stanford University
milind@stanford.edu

Abbas Kazerouni
Electrical Engineering
Stanford University
abbask@stanford.edu

Abstract

We solve the contextual question answering problem, which is an essential part in many automated question-answering datasets. Recently the SQuAD dataset [1] was uploaded and there were several deep learning approaches proposed to solve this. We implement a modified version of one of them, the Dynamic Coattention model as well as simple baseline.

1 Introduction and Problem Model

Question Answering is one of the important aspects of customer satisfaction in many businesses. In this project, we use deep learning and NLP techniques to build an automatic question answering machine which can find the answer of any given question in a given context. We do the training and experiments on the SQuAD dataset.

The SQuAD dataset consists of a context paragraph, and a question which have been crowdsourced. The task is to compute the the position of the starting and ending words in the answer based on reading comprehension. More precisely, a question is is given to the machine which can be modeled as a sequence of words such as $(x_1^Q, \cdots, x_n^Q)$. Also, a context or document is also given which is a longer sequences of words such as $(x_1^D, \cdots, x_m^D)$, and the answer is a part of this context. The machine's task is to find the answer by identifying its starting and ending position

In this paper, we first describe an implementation of the Dynamic Coattention model [2] which produced our best results, a comparison with a simple baseline, and conclude with our experimental results.

2 Dynamic Coattention Model

In this section, we describe the model we used based on [2]

2.1 Encoder

Fig. 1 carries the Encoder that we use. The essential steps are as follows:

1. The m embeddings (we used Glove vectors in this paper) of the context $x_1^D, \ldots, x_m^D$ are fed into a Bi-LSTM and the $2l \times m$ dimensional output is called D. We append a trainable

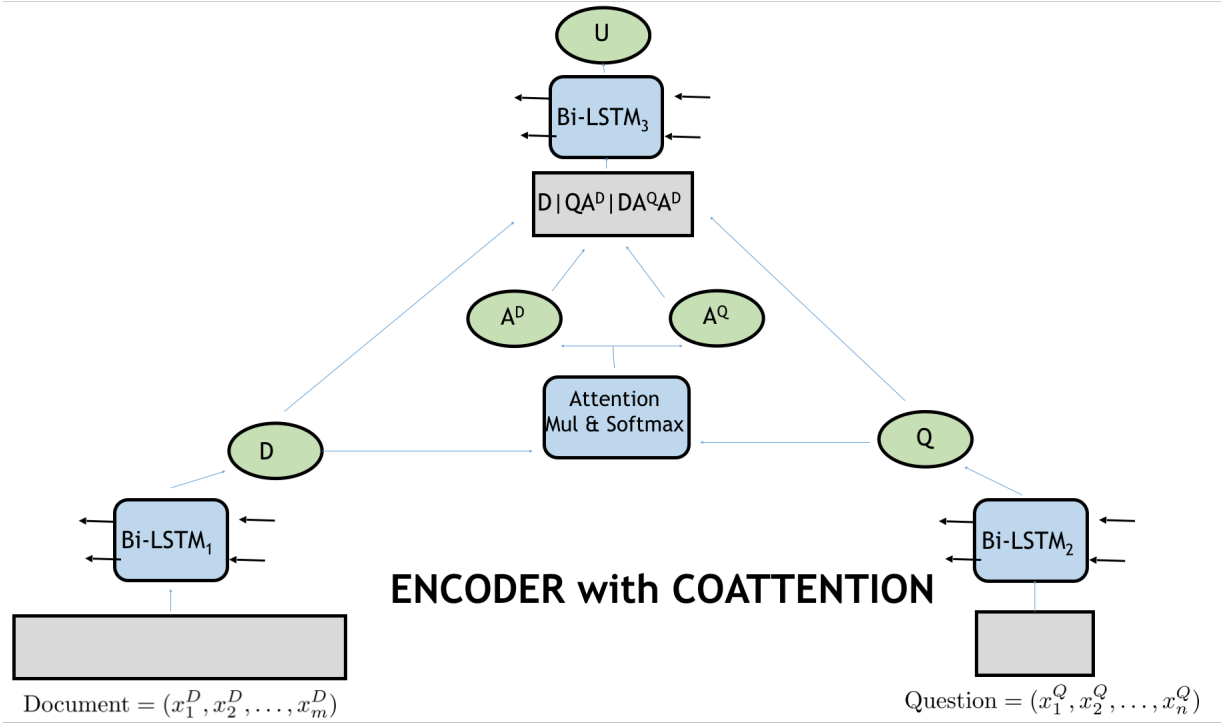

Figure 1: Attention encoder used.

random sentinel column which is useful for the attention described below. We do not set a maximum size (perhaps training with long contexts can be discarded to improve results)

2. A similar operation is done with the n question embeddings $x_1^Q, \ldots, x_n^Q$ to produce $2l \times n$ dimensional Q. Again, a trainable random sentinel is added. We differ from the original paper on two fronts - we first use BiLSTMs (for both the context and the question) to produce a more expressive function class where words at the end of the question also play an impact. Secondly, while the original paper uses the same LSTM for both D and Q and passes the latter through a linear transformation and a tanh filter, we use different Bi-LSTMs to maximise expressive power.

3. We find the attention between every word of the context and every word of the question. This follows the original paper and simply does the dot product of $D_i Q_j$ (subscript refers to columns) to find the attention between context word i and question word j. We then run a softmax (across both rows and columns) to find probability distributions of the most relevant word for words in the question/context. The sentinel that we append would correspond to not paying attention to any particular word. In other words $L = D^T Q \in \mathbb{R}^{m+1 \times n+1}$. $A^Q = \text{softmax}(L)$, $A^D = \text{softmax}(L^T)$. Other alternatives that we would have explored had we had more time would have been more expressive attention models where the attention is $tanh(D_i^T F Q_j)$.

4. We weight the context words by how relevant they are the question with QA^D Similarly, we also repeat this by using the question weighted by the context $(DA^Q)A^D$. We stack this along with the document embedding to produce $[D; QA^D; DA^Q A^D]$ and feed it into a BiLSTM to obtain our $2l \times m + 1$ dimensional embedding U.

2.2 Decoder

We largely use the dynamic decoder model of the original paper. Fig. 2 carries an overview. The steps are as follows:

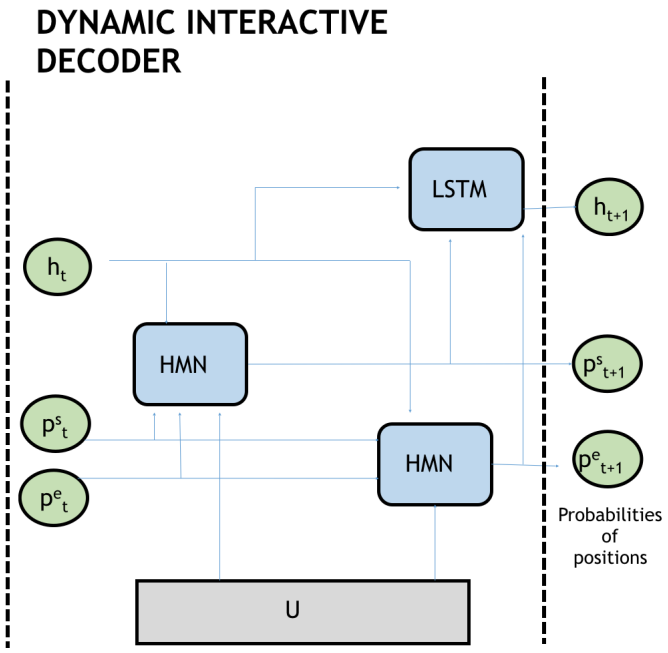

Figure 2: The iterative dynamic decoder.

1. The decoder is iterative and run a maximum of 4-5 (a hyperparameter) times. This ensures that we have multiple passes over the document and attention-with-question embeddings. At each stage t, we have inputs p_t^S which denotes the probability of the starting word in the context, p_t^E denotes the probability of the ending word, and finally h_t is the state of the decoder block.
2. We pass these inputs as well as the encoding U to two Highway Maxout Network (HMN) blocks to produce as the output p_{t+1}^S and p_{t+1}^E.
3. A highway maxout network [3] is described as follows and in Fig. 3:
   - First we use p_t^S and U to produce the expected embedding of the starting word. This is $u_t^S = \mathbb{E}_{p_t^S}[U_j]$. We differ from the original paper which uses the mode instead of the mean. This is similarly done for u_t^E, the mean embedding of the final word.
   - A linearity-tanh block summarises the previous inputs as
   $$r = \tanh(W^D[h_t; u_t^S; u_t^E]) \in \mathbb{R}^l$$
   - A maxout block, like a multilayer neural network can express any arbitrary function given enough hidden layers. As suggested by the original paper, we use this
   $$m_i^1 = \max\{W_1^1[U_i; r] + b_1^1, \ldots, W_p^1[U_i; r] + b_p^1\} \in \mathbb{R}^l$$
   $$m_i^2 = \max\{W_1^2 m_i^1 + b_1^2, \ldots, W_p^2 m_i^1 + b_p^2\} \in \mathbb{R}^l$$
   $$HMN(U, h_t, p_t^S, p_t^E)_i = \max\{W_1^3[m_i^1; m_i^2] + b_1^3, \ldots, W_p^3[m_i^1; m_i^2] + b_p^3\}$$
   There is a highway connection across a layer. Instead of using a non-linearity like tanh or sigmoid after a linear transformation, we use the maximum of p linear transformations. This allows for the modeling of a very expressive function class.
4. Finally, $[u_{t+1}^S; u_{t+1}^E]$ is the input to an LSTM that has h_{t+1} as the resultant state. Ideally, we run this iterative procedure for a much larger number of iterations and then allow the LSTM to learn when to stop. This allows for that to some extent.
5. The loss function is sum of cross-entropy loss for starting word and ending word positions
$$J = \sum_{i \in m} p_{t,i}^{real,S} - \log p_{t,i}^S + \sum_{j \in m} p_{t,j}^{real,E} - \log p_{t,j}^E$$

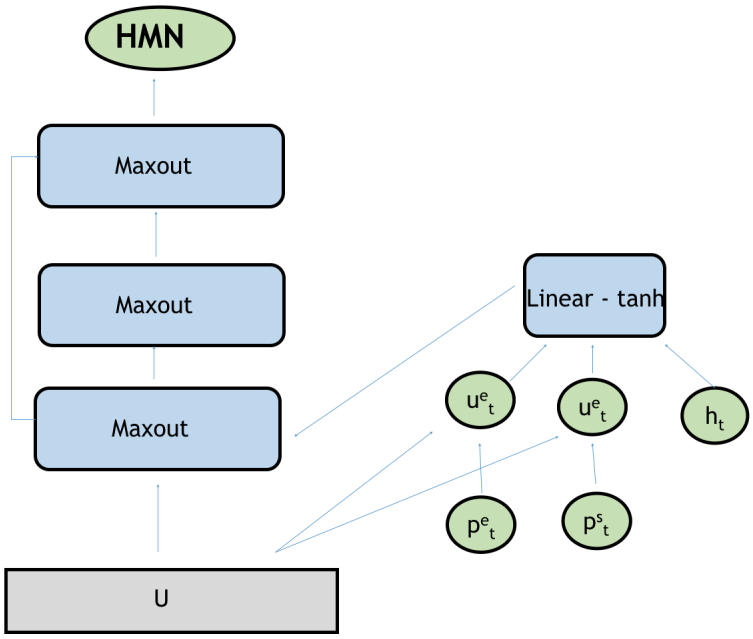

Figure 3: Schematic of the Highway Maxout Network block.

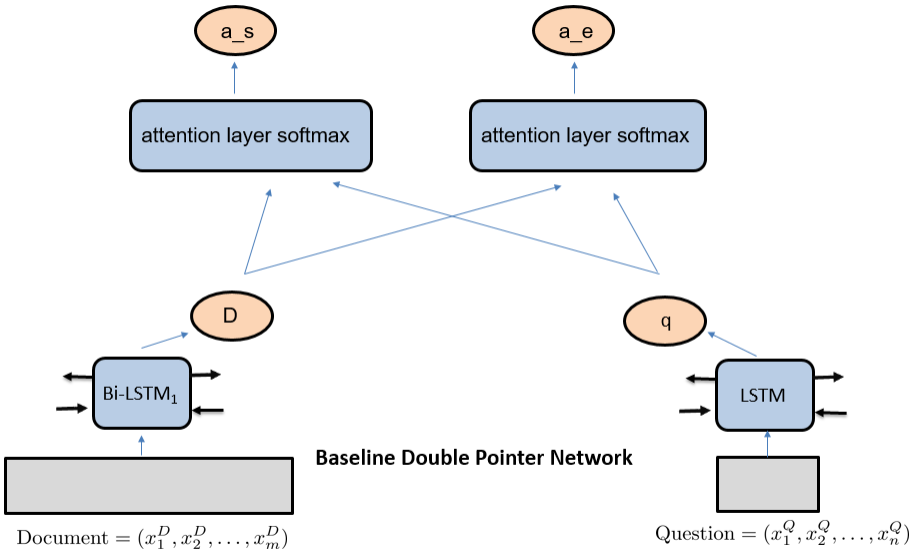

Figure 4: Baseline model.

3 Simple Baseline

We also implemented a simple baseline as described in Fig. 4. This is a basic attention network which generates the associated hidden states for each of the words in the context and the question, by running a BI-LSTM, and then passing these to an attention layer followed by a softmax, generates the probability of each word in the context being the starting or ending position of the answer.

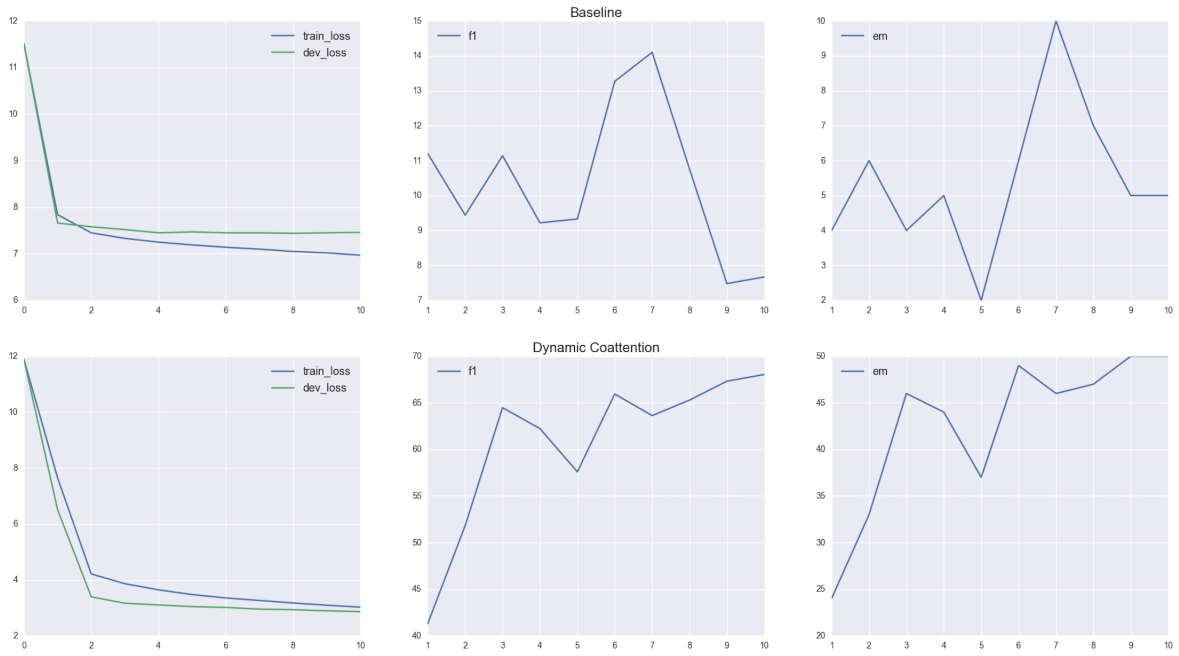

Figure 5: Experimental results

4 Experimental Results

We ran 10 epochs, batch size 32, hidden layer size 100 everywhere, dropout 0.5. We did not have time to play with hyper-parameters or train ensembles.

Our results are summarized in Fig. 5. Our implemented dynamic coattention network achieves an F1 score of 68.04 and an EM of 50 on the validation data set after being trained on 10 epochs.

5 Conclusions

We implemented and modified the Dynamic Coattention model and obtained an F1 score of 68.04 and an EM of 50 on the SQuAD dataset. We find that complex networks with coattention mechanisms, and iterative maxout networks boost performance, especially in the presence of large amounts of data.

Acknowledgments

We thank the CS224N team for a great course and the Azure team for providing access to GPUs.

References

[1] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. Squad: 100,000+ questions for machine comprehension. *EMNLP 2016* of text. In Empirical Methods in Natural Language Processing (EMNLP), 2016.

[2] Caiming Xiong, Victor Zhong, Richard Socher. Dynamic Coattention Networks for Question Answering, *ICLR 2017*.

[3] Ian J Goodfellow, DavidWarde-Farley, Mehdi Mirza, Aaron C Courville, and Yoshua Bengio. Maxout networks. *ICML (3), 28:13191327, 2013*.