Natural Language Question Answering on Squad dataset

Chaikesh Chouragade
Systems Engineering
chaikesh@gmail.com

Prakash Kumar Uttam
Systems Engineering
prakashuttam@iisc.ac.in

Sonu Dixit
Systems Engineering
sonudixit2k@gmail.com

Abstract

Many deep learning models have been proposed for question answering. One such model is Dynamic Co-attention Network (DCN) for question answering. The DCN first fuses co-dependent representations of the question and the document in order to focus on relevant parts of both. In this Project we tried to improve over dynamic co-attention networks(DCN) encoder by adding more connections and layers of Bi-LSTM .Essentially we added one more co-attention module,and used the concatenated output of the two module to be fed to Bi-LSTM to generate final output for document representation.The motivation for such encoder comes from recent literature on Self Attention Network.We use same decoder presented in original DCN,where dynamic pointing decoder iterates over potential answer spans. This iterative procedure enables the model to recover from initial local maxima corresponding to incorrect answers. On the Stanford question answering dataset (SQUAD) our improvement gives F1 score of 76.7 which is an improvement over original DCN with F1 sore of 75.9.

1 Introduction

Question answering is a crucial task in natural language processing that requires both natural language understanding and world knowledge. This task is similar to reading comprehension exercise. Given a paragraph(context) and some questions(query), our aim is to answer the queries. Its an easy task for humans but for machines it requires modeling complex interaction between the paragraph and the question.

Here in SQUAD dataset, the answer for questions is a continuous sequence of words in paragraph, So we have to search for start and end index, hence its a bit easier than normal reading comprehension, which may involve abstractive summarization of one or more lines as answer. Several immensely practical applications can be built on top of modified version of this kind QA system.A few are: Automated Customer service system, AI system to answer queries from legal documents.

2 Related Work

2.1 Statistical QA

Traditional approaches to question answering typically involve rule-based algorithms or linear classifiers over hand-engineered feature sets. Richardson et al. (mct, 2013) proposed two baselines, one that uses simple lexical features such as a sliding window to match bags of words, and another that uses word-distances between words in the question and in the document.

Berant et al.(Berant and Liang, 2014) proposed an alternative approach in which one first learns a structured representation of the entities and relations in the document in the form of a knowledge base, then converts the question to a structured query with which to match the content of the knowledge base. Wang et al. (Wang et al., 2016) described a statistical model using frame semantic features as well as syntactic features such as part of speech tags and dependency parses. Chen et al. (Chen et al., 2016) proposed a competitive statistical baseline using a variety of carefully crafted lexical, syntactic, and word order features.

2.2 Neural models for QA

Current state-of-the-art approaches for question answering over unstructured text tend to be neural approaches. Wang & Jiang (Wang and Jiang, 2016) proposed one of the first conditional attention mechanisms in the Match-LSTM encoder. Coattention (Xiong et al., 2016), bidirectional attention flow (Seo et al., 2016), and self-matching attention (Microsoft Asia Natural Language Computing Group, 2017) all build codependent representations of the question and the document.

These approaches of conditionally encoding two sequences are widely used in question answering. After building codependent encodings, most models predict the answer by generating the start position and the end position corresponding to the estimated answer span. The generation process utilizes a pointer network (Vinyals et al., 2015) over the positions in the document. (Xiong et al., 2016) also introduced the dynamic decoder, which iteratively proposes answers by alternating between start position and end position estimates, and in some cases is able to recover from initially erroneous predictions.

3 Method

Overall, We tried four models in this Project. Model-1 and Model-2 are there just to compare the performance of our encoder(Modified DCN).These four models uses three encoder module and two decoder module in total. Description of the models are as follows:

- Model 1
  - **Encoder** – baseline
  - **Decoder** – baseline
- Model 2
  - **Encoder** – DCN encoder
  - **Decoder** – baseline
- Model 3
  - **Encoder** – Modified DCN encoder
  - **Decoder** – baseline
- Model 4
  - **Encoder** – Modified DCN encoder
  - **Decoder** – DCN decoder

Description for Baseline Model and modules used in remaining Model is given below

3.1 Baseline Model(course cs224n, 2018)

Baseline model has three components.
1. RNN encoder layer : Every word is represented as pre-trained embedding from GLOVE. Bidirectional GRU is used to convert context and question to embedding.
2. Attention Layer: It applies standard attention with the context hidden state attending to the question hidden state. For each context hidden state, attention distribution is computed over each word in question and vice versa. Attention outputs are then concatenated with context hidden state to give blended representation
3. Output Layer: Blended representation are fed through a fully connected layer, followed by a ReLU non-linearity. Independent Probability distribution on start and end index is generated. Maximum probability index is selected as answer.

3.2 DCN Encoder(Xiong et al., 2016)

For description of DCN Encoder refer to DCN(Xiong et al., 2016)

3.3 Modified DCN Encoder

In DCN model, a single layer coattention Encoder is used. Since its single layer, it has limited ability to compose complex representation. So on top of coattention encoder we are applying a self attention by stacking coattention layers. This allows network to build richer representation over input. Vector representation of Words in context and question are as follows:

$$document = m\ words = L^D \in \mathbb{R}^{e \times m}$$
$$question = n\ words = L^Q \in \mathbb{R}^{e \times n}$$

where, e is the dimension of word embedding obtained using word2vec $GLOVE$ Final document and question embedding is obtained using BiLSTM.

$$E_1^D = biLSTM_1(L^D) \in \mathbb{R}^{h \times m+1} \quad (1)$$
$$E_1^Q = \tanh(W\ biLSTM_1(L^Q) + b) \in \mathbb{R}^{h \times n+1} \quad (2)$$

h denotes hidden state size of biLSTM. Like original DCN we have added one non-linear transform to the question encoder in equation (2).

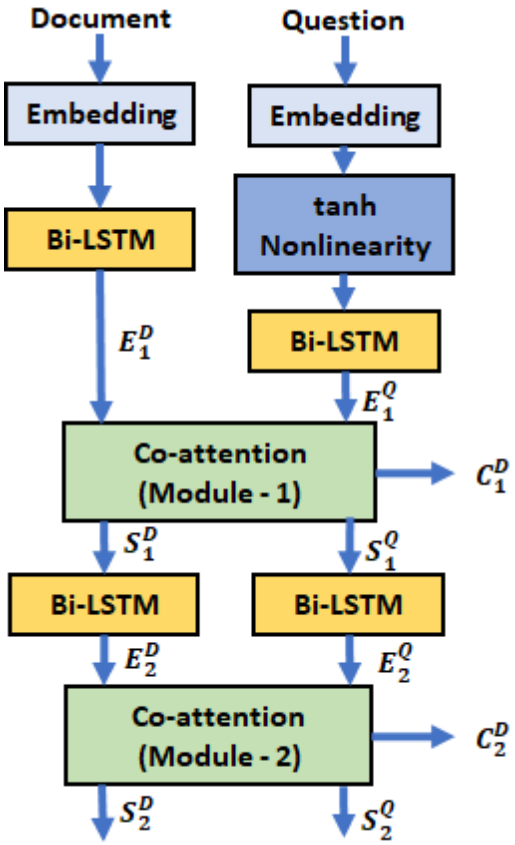

Figure 1: encoder (a)

In the next step we calculate Affinity matrix. Affinity matrix between document and question is calculated as:

$$A = (E_1^Q)^T E_1^D \in \mathbb{R}^{(m+1)\times(n+1)} \quad (3)$$

Now we calculate Summary for document and question as given below:

$$S_1^D = E_1^Q \text{ softmax}(A^T) \in \mathbb{R}^{h\times(m+1)} \quad (4)$$
$$S_1^Q = E_1^D \text{ softmax}(A) \in \mathbb{R}^{h\times(n+1)} \quad (5)$$

where softmax(X) is softmax operation over matrix X that normalizes X column wise.

Document co-attention context is calculated as,

$$C_1^D = S_1^Q \text{softmax}(A^T) \in \mathbb{R}^{h\times m} \quad (6)$$

Now, we encode the summary matrices using another biLSTM,

$$E_2^D = biLSTM_2(S_1^D) \in \mathbb{R}^{2h\times m} \quad (7)$$
$$E_2^Q = biLSTM_2(S_1^Q) \in \mathbb{R}^{2h\times n} \quad (8)$$

Now, we compute the second co-attention layer,

$$\text{Co-attn}_1(E_1^D, E_1^Q) \Rightarrow S_1^D, S_1^Q, C_1^D \quad (9)$$
$$\text{Co-attn}_2(E_2^D, E_2^Q) \Rightarrow S_2^D, S_2^Q, C_2^D \quad (10)$$

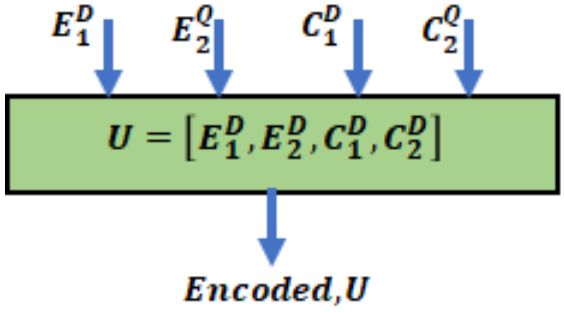

Figure 3: encoder (c)

Output of Encoder,

$$U = \text{biLSTM}(\text{concat}(E_1^D, E_2^D, C_1^D, C_2^D)) \in \mathbb{R}^{2h\times m}$$

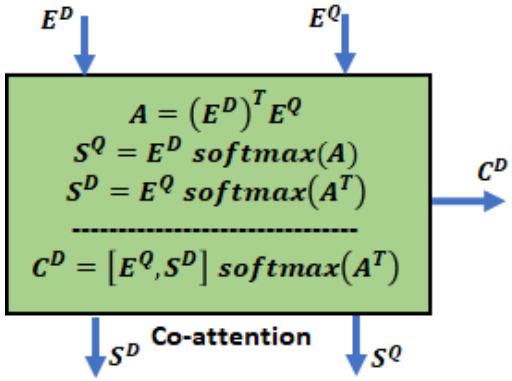

Figure 2: encoder (b)

3.4 DCN Decoder

We have used the same decoder that was used in (Xiong et al., 2016) The decoder should provide start and end index of the answer in the paragraph. Finding these indexes is an iterative process. During each iteration, based on current state of decoder and coattention encoding of current estimate of start and end index, decoder produces next estimate and updates its hidden state. On top of decoder we are using highway maxout network(Xiong et al., 2016) to produce start and end indexes.

$$h_i = \text{LSTM}_{dec}(h_{i-1}, [u_{s_i-1}; u_{e_i-1}]) \quad (11)$$

Where, u is the representation corresponding to previous estimation of index.

$$s_i = \arg\max_t(\alpha_1, \alpha_2, ..., \alpha_m) \quad (12)$$
$$e_i = \arg\max_t(\beta_1, \beta_2, ..., \beta_m) \quad (13)$$

here, α and β are calculated using two different highway maxout network (HMN) (Xiong et al., 2016). Both, the neural network has same network structure, but different parameters. HMN provides simple and effective method to pool across multiple model variation.

$$\text{HMN}\left(u_t, h_i, u_{s_{i-1}}, u_{e_{i-1}}\right)$$

$$= \max\left(W^{(3)}\left[m_t^{(1)}; m_t^{(2)}\right] + b^{(3)}\right) \quad (14)$$

$$r = \tanh\left(W^{(D)}\left[h_i; u_{s_{i-1}}; u_{e_{i-1}}\right]\right) \quad (15)$$

$$m_t^{(1)} = \max\left(W^{(1)}\left[u_t; r\right] + b^{(1)}\right) \quad (16)$$

$$m_t^{(2)} = \max\left(W^{(2)} m_t^{(1)} + b^{(2)}\right) \quad (17)$$

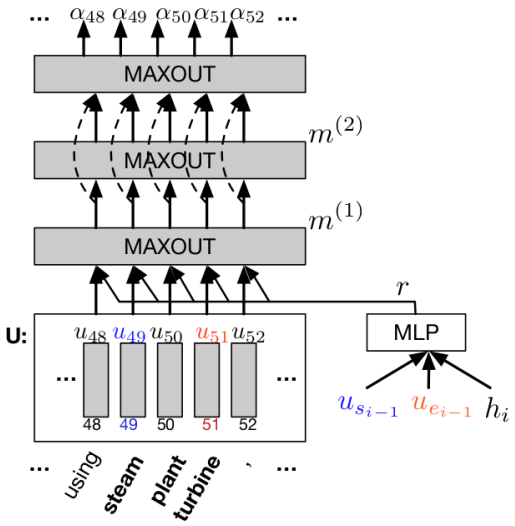

Figure 4: HMN(Highway Maxout Network)

We are minimizing cross entropy loss for predicted start and end index in a batch size of 20. Final start and end index is given if they don't change in consecutive iterations or when maximum number of user defined iterations is over.

4 Results

First we trained Model-1 (baseline model),which we took from stanford project page (course cs224n, 2018) ,which uses naive encoder and naive decoder.
then,we trained our Model-2 and Model-3 .Model-2 is used to asses whether our modified DCN encoder gives any improvement over DCN encoder,in both the model we have used naive decoder from baseline model.

At last we trained Model-4 only change here is ,we used DCN decoder.The results of this model indicate whether our model improves over original DCN model (Xiong et al., 2016) or not.

4.1 F1-score and EM(Exact Match)

The results of above mentioned models are tabulated below.

Model	Dev-Set(F1-score)	Dev-Set(EM)
Model-1(baseline)	39	28
Model-2	43.4	32
Model-3	44.8	34
Model-4	**76.7**	**66.5**
Original DCN	75.9	66.2

4.2 Plots

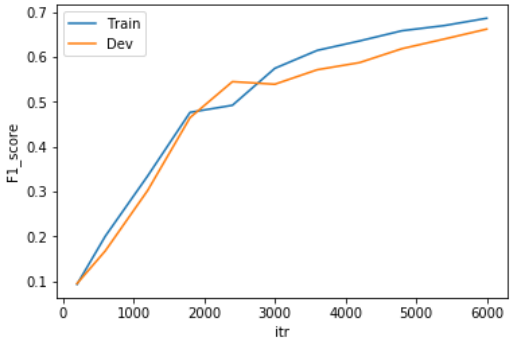

Figure 5: F1 Score comparision

We can see in the figure that our model's F1 score on development(test) set is around 0.66 after 6000 iterations using batch size of 100. Test set data for this SQUAD has not been released. We have used Dev Set data for our all the results.

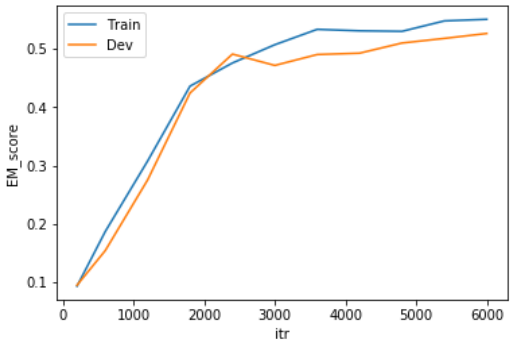

Figure 6: EM Score comparision

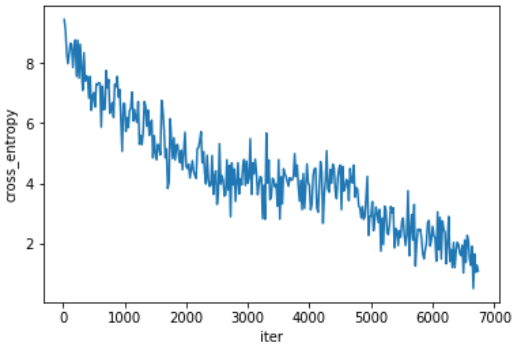

Figure 7: Cross Entropy loss(Model-4)

5 Discussion

There are several hyper parameter to tune in our model,choosing the best hyperparameter is crucial for model performance.In this project we performed few experiments on dev set to tune the hyperparameters of our model to get to best results.

5.1 List of hyperparameters

Hyper Parameter	value
Learning rate	0.001 with exponential decay
Gradient cliping	5.0
Dropout	0.15
Batch size	32
Hidden state size	200
Fixed question length	30
Fixed document length	600

5.2 Other aspects

Apart from hyperparameter tuning following aspects were also important for better tuning the model.some of them are listed below

Word embedding vector size

Deciding on word embedding vetor size is also crucial.There is option for specifying the embedding size for pre-trained GLOVE embedding.

We experimented with 100,200,300 length vector embedding .In our experiment we found out that the word embedding size of 100 gave us the best result .

Combining forwards and backwards states(Bi-LSTM)

The final hidden state output of Bi-LSTM contains forward output hidden state and backward output hidden state. There are a few options to use both the hidden state in some way as

1. Concatenation: Both the hidden state output is concatenated side by side, to feed to next layer in our network.
2. Average: In this we take average of both the hidden state.
3. Max Pool: In this we take maximum of two hidden state component wise.
4. Addition: In this we add the two state component wise.

6 Future Work

6.1 Feature Engineering

Several additional features (alongside word vectors) to represent a context token can be incorporated,few of them are listed below

1. Exact Match: Whether a token in question exactly matches with a token in context.
2. Token Features: Part of Speech tag, Named entity recognition tag, normalized term frequency can be used as input.

6.2 Ensembling

Ensembling almost always boosts performance, so combining several models together for final result will improve the performance . However, ensembles are more computationally expensive to run,thats why we didnt implement it.

6.3 Regularization

In our model we only used dropout for training.Regularizaion alongside the dropout can give some improvement in performance.

References

2013. *MCTest: A Challenge Dataset for the Open-Domain Machine Comprehension of Text*. https://www.microsoft.com/en-us/research/publication/mctest-challenge-dataset-open-domain-machine-comprehension-text/.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of the*

52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Association for Computational Linguistics, pages 1415–1425. https://doi.org/10.3115/v1/P14-1133.

Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. *CoRR* abs/1606.02858. http://arxiv.org/abs/1606.02858.

Stanford NLP course cs224n. 2018. baseline model as default project. `http://web.stanford.edu/class/cs224n/default_project/index.html`. [Online; accessed 25-04-2018].

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *CoRR* abs/1611.01603. http://arxiv.org/abs/1611.01603.

O. Vinyals, M. Fortunato, and N. Jaitly. 2015. Pointer Networks. *ArXiv e-prints* .

Shuohang Wang and Jing Jiang. 2016. Machine comprehension using match-lstm and answer pointer. *CoRR* abs/1608.07905. http://arxiv.org/abs/1608.07905.

Zhiguo Wang, Haitao Mi, Wael Hamza, and Radu Florian. 2016. Multi-perspective context matching for machine comprehension. *CoRR* abs/1612.04211. http://arxiv.org/abs/1612.04211.

Caiming Xiong, Victor Zhong, and Richard Socher. 2016. Dynamic coattention networks for question answering. *CoRR* abs/1611.01604. http://arxiv.org/abs/1611.01604.