Neural Methods for Question-Answering

Manik Dhar
dmanik@stanford.edu

Pratyaksh Sharma
praty@stanford.edu

Ayush Kanodia
akanodia@stanford.edu

Abstract

Machine comprehension is a challenging and important task in Natural Language Processing. We look at a particular instance of the task by targeting the question answering task on the SQuAD dataset. In particular we try variants of two current state of the art models and analyze them.

1 Introduction

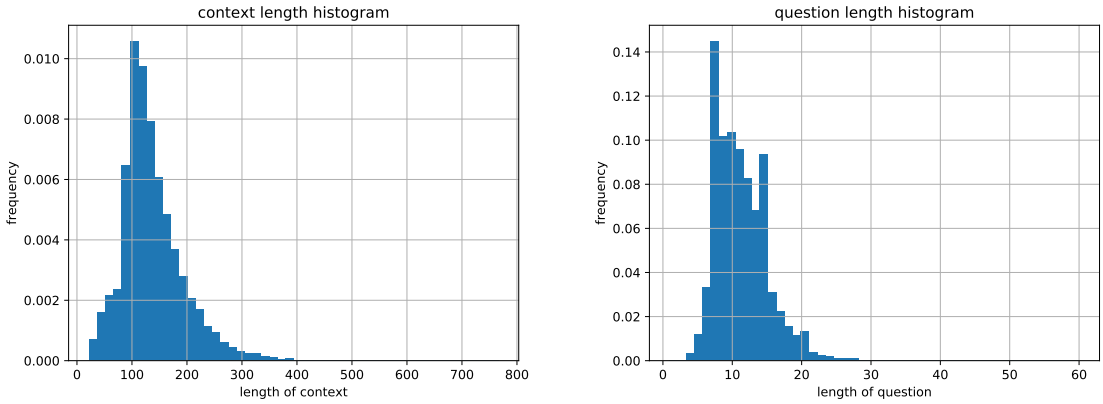
Question answering is quite a challenging problem. One needs to understand the question and understand the information provided with it to obtain the answer. To measure our progress in this task we need datasets to do tests on. Stanford Question Answering Dataset[4] (SQuAD) provides a large and challenging dataset for this task. There has been a lot of recent work on this problem and in the following section we will discuss them. In later sections we analyze the results we got after implementing certain variants of these models. In particular we look at Match-LSTM[6] and Dynamic Co-attention networks[7].

2 Dataset

Stanford Question Answering Dataset (SQuAD) consists of extracts from Wikipedia articles and questions posed on them. The answer to these questions are always a contiguous span in the paragraph provided. The articles cover a broad range of topics and provide a challenging task to test Machine Comprehension models on. The SQuAD dataset has a test set which is withheld from everyone and used to benchmark models trained on the remaining dataset.
We analyze certain distribution of the question and paragraphs sizes in the dataset. From Figures 1a, 1b we observe that almost all contexts in the dataset are within a length of 300, and almost all questions are within a length of 25. We shall use this information to reduce our model size in order to speed up training.

2.1 Evaluation Criterion

We use two evaluation criterion for this task F1 and Exact Match score EM. SQuAD can have multiple right spans for the same question. Exact match is a metric that measures the percentage of predictions that match one of the ground truth answers exactly.
F1 score is a metric that loosely measures the average overlap between the prediction and



(a) (b)

Figure 1: Distribution of context and question lengths on the SQuAD training set.

ground truth answer. We treat the prediction and ground truth as bags of tokens, and compute their F1 (the harmonic mean of precision and recall). We take the maximum F1 over all of the ground truth answers for a given question, and then average over all of the questions.
For reference humans score 90.5% F1 and 86.8 % EM and the state of the art models score 80% F1 score and 75% EM on the test set. A logistic regression benchmark was implemented in SQuAD paper which achieves 51% F1 and 40.4% EM on the test set.

2.2 Problem description

Given a paragraph P and question Q we want to find two positions in the paragraph s and e indicating the start and end location of the answer. P and Q are tokenized via the NLTK [2] library. We used word level embeddings obtained from GLoVE vectors [3]

3 Related Work

A lot of work has been done on this task. We will briefly describe variants of two state of the art models Match-LSTM and Dynamic Co-attention networks which we implemented.

3.1 Match-LSTM

3.1.1 Encoder

The encoder involved an encoding layer which process the question and paragraphs embedding via an LSTM [1] to generate hidden representations.
Let the hidden representation of the paragraph be H^p and for the question be H^q. They will be of size lXp and lXq respectively (where l, p ,and q are the sizes of the hidden state, paragraph and question respectively).
Now we explain the Match-LSTM model which we runs over the paragraph representation. At the ith stage using H^q, h_i^p and the output of the Match-LSTM from the previous step to generate soft-attention vectors α_i over the question. We later concatenate h_i^p and $H^q\alpha_i^T$ which is the input for our Match-LSTM. This provides a new set of embeddings. We run another Match-LSTM in the backwards direction with tied weights. We concatenate the forward and backward output to generate our knowledge encoding.

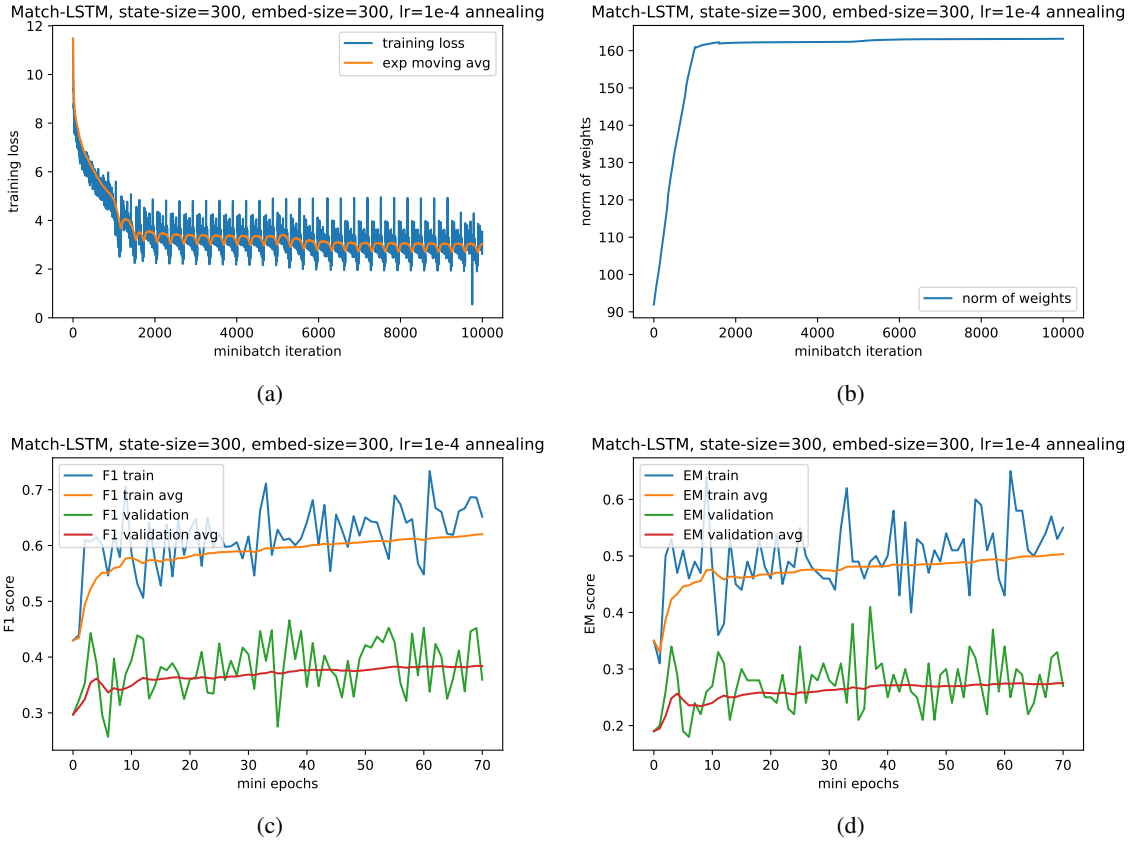


Figure 2: Diagnostic and evaluation plots for Match-LSTM model. (a) illustrates the progression of loss with number of training iterations, (b) depicts the increasing trend of the norm of parameters, (c) and (e) illustrate F1 and EM evaluation on both training and validation sets.

3.1.2 Decoder

The decoder is implemented as an answer pointer layer inspired from the pointer net architecture [5]. We implement a boundary model, that is we output two vectors representing the softmax probabilities of the context words being start and end tags for the answer.
We use a simple 2-layer feed forward network over the encoding from the encoder (let us call it H) to generate the predictions for the start position β_s. Then we use these probabilities as attention vectors over H to obtain $H\beta_s^T$. We use a new feedforward network over H and the $H\beta_s^T$ to obtain the end probabilities, β_e. This forward network has tied weights with the first one. The new network just one extra set of weights corresponding to the $H\beta_s^T$ input. We apply a simple cross-entropy loss on these predictions. We also run another model where the start and end decoder weights are untied.

3.2 Dynamic Coattention

Dynamic Co-attention for question answering [7] uses a encoder which applies dynamic co-attention. The decoder is a highway maxout network. We did implement both the encoder

and decoder but this model took too long to train so we used the coattention encoder, along with the Match-LSTM decoder with untied weights.

3.3 Coattention encoder

The coattention encoder first calculates the affinity matrix L which is the product between the paragraph and question encoding $L = P^T Q$. This matrix has softmax applied on it row-wise to obtain attention weights for the document and column wise for the question ($A^P = softmax(L)$ and $A^Q = softmax(L)$).
We apply the attention to the paragraph to obtain $C^Q = PA^Q$ and next we find the $C^D = [Q; C^Q]A^P$ ([;] is the concat operation). Finally, we run a Bi-LSTM over a concatenation of C^D and P to obtain our knowledge encoding.

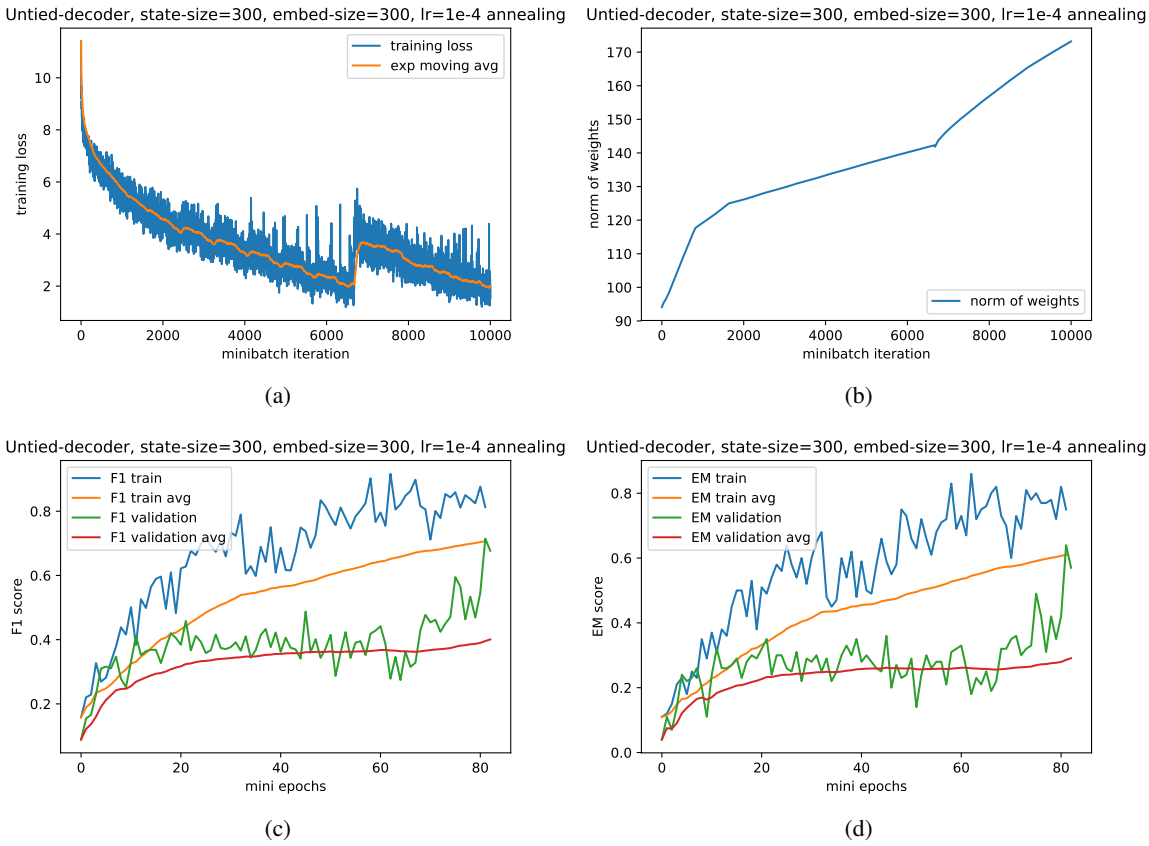


Figure 3: Diagnostic and evaluation plots for Untied-decoder model. (a) illustrates the progression of loss with number of training iterations, (b) depicts the increasing trend of the norm of parameters, (c) and (e) illustrate F1 and EM evaluation on both training and validation sets.

4 Hyper-parameter Selection

For our model we use 300 size glove embeddings. We limit the paragraph and answer size in our model to 300 and 25 respectively. On analyzing the historgram of word paragraph

distribution we note that most paragraphs are smaller than 300 size. A similar observation holds for question as well. We clip any answers which go beyond the limit while training. We start from a learning rate of 0.01 and reduce it by a fixed factor of 0.9 with each epoch. We used a fixed batch size of 80 in all experiments.
Once we noticed our models were overfitting we applied dropout on the non-recurrent layers with keep probability 0.85. We used Adam optimizer with 0.9 β_1 and 0.999 β_2.

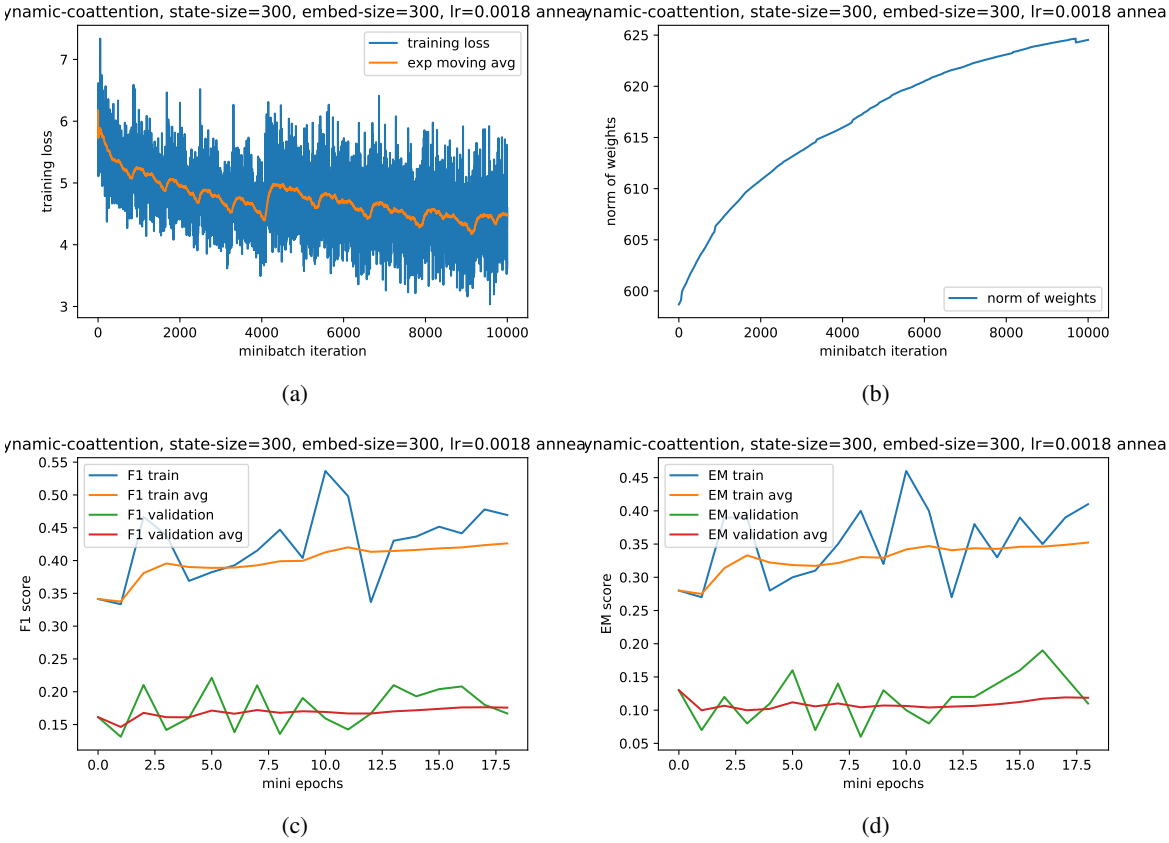


Figure 4: Diagnostic and evaluation plots for the Dynamic-Coattention model. (a) illustrates the progression of loss with number of training iterations, (b) depicts the increasing trend of the norm of parameters, (c) and (e) illustrate F1 and EM evaluation on both training and validation sets.

5 Evaluation

Figure 2a indicates that Match-LSTM converges to a loss of around 3 and although it provides a decent training F1 and EM score, the validation score is too low in comparison. This indicates that the model is overfitting. See Figures 2c, 2d.

In Figures 3a, 3c, 3d, we see that untying the weights lets the model drop the loss even further and gives much better F1 and EM performance but still there is a large gap between the two even after applying dropout. The slight spike in the validation set performance near the end is because at that point we were also training on the validation set in preparation for

submitting to the test set and it is not indicative of the model generalizing in later epochs. Dynamic Coattention encoder with Match-LSTM decoder doesn't train very well and its training has been quite noisy. This is probably because of not tuning our hyperparameters well enough. See figures 4a-4d.

6 Analysis and Future Work

In this section we look at some of the question answer, outputs and suggest improvements to the model.

One clear room for improvement is applying better regularization by either increasing dropout or applying weight decay. From the graphs we can see that the norm of the parameter does steadily increase while training, forcing the weights to be small will prevent overfitting.

Now we look at some of the questions and their answers.

```
context= The Broncos took an early lead in Super Bowl 50 and never trailed .
Newton was limited by Denver 's defense , which sacked him seven times
and forced him into three <unk> , including a <unk>
which they recovered for a touchdown . Denver linebacker Von Miller was named
Super Bowl MVP , recording five solo <unk> , <unk> sacks , and two forced <unk> .
question= How many forced <unk> did Von Miller have during the Super Bowl 50 game ?
answer= seven times
```

We can see that the reason that the model gives the wrong answer here is because of not having all the tokens deciphered. Using a more comprehensive tokensizer will give us better results.
There were some empty answers which indicate the start position has been predicted to be after the end. We can encourage that not to happen by putting a soft constraint on this.

```
context= The collection of drawings includes over 10,000 British and
2,000 old master works , including works by : <unk> , Giovanni Benedetto ....
question= Approximately how many old masters works
are included in the V & A collection ?
answer= 10,000
```

As we can see the model did figure out to pick the numeral for the question "how many?" but didn't pick the right way value. Giving extra information like parts of speech tag may help the model.
The ideas and insights from this analysis is an interesting direction to look for further improver our models for this task. The good news there is room for improvement.

References

[1] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[2] Edward Loper and Steven Bird. Nltk: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective tools and methodologies for teaching natural language processing and computational linguistics-Volume 1*, pages 63–70. Association for Computational Linguistics, 2002.

[3] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[4] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

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

[6] Shuohang Wang and Jing Jiang. Machine comprehension using match-lstm and answer pointer. *arXiv preprint arXiv:1608.07905*, 2016.

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