Optimizing Match-LSTM for SQuAD v2.0

Stanford CS224N Default Project

Peter Nsaka
Department of Computer Science
Stanford University
nsaka@stanford.edu

James Dong
Department of Computer Science
Stanford University
xdong73@stanford.edu

Alex Lee
Department of Computer Science
Stanford University
alee8599@stanford.edu

Abstract

In 2016, Wang and Jiang built a novel Question Answering machine comprehension model[1], based on LSTM and Pointer Net using the SQuAD 1.0 datastet. This model was considered state-of-the-art at the time and capable of far outperforming previous logistic regression models. However, since its publication, SQuADv2.0 has been released, leaving the model somewhat antiquated. In this project, we aim to implement the model described in the original paper, adapt the model to work with the SQuAD 2.0 Data set and perform updates to the model architecture to improve its performance against the benchmark BiDAF model. Our implementation of the Match-LSTM model trained on SQuAD v2.0 with an F1 score of 51.88 and EM score of 54.51 performs a bit lower compared to the Baseline BiDAF Model. Throughout the project, there were mutliple struggles when attempting to adapt it to the SQuAD v2.0 dataset that we were unable to resolve such as preventing the model from marking too many answerable questions as unanswerable. Our initial thoughts were that the prevalence of unanswerable questions in the dataset skewed our model towards predicting all other questions as unanswerable. However, though it does seem to demonstrate some improvement, upsampling answerable questions still does not seem to completely fix this issue. Overall, this model produces reasonable performance with modest modifications and shows promise if given further time to adapt the model to the SQuAD v2.0 dataset. However, we realized that the Match-LSTM model has an immensely long training time of 15 hours when compared the baseline BiDAF model which completed training in about 3 hours. And this continued to hamstring our progress due to resource limitations as well as inability to rapidly iterate upon past attempts.

1 Key Information to include

- Mentor: Michihiro Yasunaga
- External Collaborators (if you have any): None
- Sharing project: N/A

2 Introduction

Machine comprehension (MC) and question answering (QA) have been gaining rapid popularity due to the complexity and practical implications of the task. One of the primary goals has been to try to point to a specific area within a piece of text as the answer when given a contextual question. As more

successful approaches began to be developed to tackle the task of MC and QA, there quickly rose a need for a unifying metric to compare against other models' performances. Therefore, in 2016, the SQuAD Dataset was introduced as a way of evaluating machine comprehension models. It quickly became the standard for testing question answering models and comparing a model's performance on the dataset to other models became the norm. Then, more recently, the SQuAD 2.0 dataset was introduced, bringing a new set of interesting additions, namely the addition of unanswerable questions. These newly included questions created new challenges for question answering by introducing instances where certain questions cannot be answered at all and expecting the model to recognize the non-existence of an answer.

To address these new complications, many different models and approaches have been introduced. One approach that has been particularly popular has been the BIDAF network, which utilizes various attention mechanisms to find the most relevant information within the context of the given text. And we will be using this model's performance on the v2.0 dataset as the baseline for our project. For this project, we have been primarily focused on the implementation introduced by Wang and Jiang [1] with the Match-LSTM and Pointer Network. The Match-LSTM network Our goal has been to adapt the approach originally meant for the SQuAD 1.0 dataset to the v2.0 dataset by adding new features that help the model identify which questions are unanswerable.

3 Related Work

The SQuAD dataset released by Rajpurkar et al. was unlike any other dataset before it. Previous datasets had previously forcused on multiple-choice setting fro Question answering or assumed teha nswers are single tokens. Given that the SQuAD dataset had answered consisting of multiple sequential tokens, new approaches to the problem of Question Answering had to be utilized. Although many neural network based models architectures have been applied to the problem, we decided to explore an approach which conditioned the end prediction based on the start prediction as seen in the paper Machine Comprehension Using Match-LSTM and Answer Pointer (Wang Jiang, 2016).

In the paper, Wang and Jiang[1] built a question answering model based on two important frameworks. The first a Match-LSTM layer which is able to determine is a "premise" scentence entails a "hypothesis" scentence by utilizing an attention to weight the premise and using the weighted vector on all tokens in the hypothesis. The other important piece is the pointer network originally proposed by Vinyals et al.[3] to learn about conditional probability of an output sequence with elements that are discrete tokens corresponding to positions in an input sequence.

4 Approach

Our model relies heavily on work introduced by (Wang Jiang, 2016) in the paper Machine comprehension using Match-LSTM and answer pointer. The original model is an end-to-end neural network model for machine comprehension which takes in a context paragraph and a related question as input and predicts a part of the context paragraph as an answers. We have slightly modified the model to also predicts no answers at all. In this section we'd be describing the Match-LSTM model on a high level. Please refer to the paper[1] for more detailed information.

4.1 Data Preprocessing

To get the SQuAD dataset ready to be used for training our model, we utilize the GloVe pre-trained word embedding(glove.840B.300d.txt). Using the GloVE embedding, we map each word to a 300-dimensional vector. GloVe is an unsupervised learning algorithm for obtaining vector representation for words. The word vectors utilized in our model are trained on the 840B word corpora from Common Crawl data. The dimensionality of the embedding at 300 is significantly higher that the other choices of 50, 100 and 200 it provides more contextual information about each word but with the added tradeoff of a larger input parameters and longer processing time.

During pre-processing, we permit using the following features:

- **POS Tagging:** If enabled we include part-of-speech(POS) tags of the tokens. The POS tags are retrieved from the en_core_web_sm model loaded on the Spacy library

• **NER Tagging:** If enabled we include Named Entity (NER) tags of the tokens.The NER tags are retrieved from the en_core_web_sm model loaded on the Spacy library.

Although our implementation closely followed that in the original paper, a few changes had to be made to adapt the model to predict "No Answers" for the SQuAD 2.0 dataset. During pre-processing of the dataset, a noans token was introduced with an "id" value of 1 to represent no answer scenarios. The value of 1 was prepended to each sentence's token list, part-of-speech tag list and entity tag list for every context. When predicting an answer, if $p_{start}(0) \cdot p_{end}(0)$ is greater than that of any predicted answer span, the model predicts no-answer. Otherwise we predict the highest probability span as usual.

4.2 The Match LSTM Model

The Match-LSTM model is made up of multiple layers capable of extracting varying levels of contextual information from each question-context dataset. On a high level our model is made up of an LSTM Preprocesing layer, a Match-LSTM layer and an output layer. First the questions and paragraphs are input into separate LSTM cells to incorporate contextual information in the encodings. Secondly, the encoded questions and paragraphs are input into a Match-LSTM later which uses a word to word attention mechanism to associate each word in the context paragraph with it's relevance to the related question. Lastly, an output layer responsible for generating and answer uses and attention mechanism to determine the probability of each word in the paragraph to be the start or end index. Below is a more detailed discussion of the model layers.

4.2.1 LSTM Preprocesing layer

The LSTM Preprocesing layer contextualizes information from the questions and paragraphs into our encoding. This means a word is encoded based on the meaning of the words next to it. This is achieve by using a bidirectional LSTM RNN on the question an paragrah to encode contextual information for each word in each direction. The output of this layer is:

$$H^p = LSTM(P) \quad H^q = LSTM(Q) \tag{1}$$

Here $H^p \in \mathbb{R}^{l \times p}$ and $H^q \in \mathbb{R}^{l \times q}$ where l is the size of the hidden state of the LSTM, p is the length of the context paragraph and q is the length of the question.

4.2.2 Match-LSTM layer

In summary, the Match-LSTM layer is an attention layer which measures how much each token in the paragraph is related to the question. At each step during training, the Match-LSTM derives and attention vector over the question using it's previous state and the input layers. The attention vector is computed as:

$$\vec{G}_i = \tanh\left(W^q H^q + \left(W^p h_i^p + W^r \vec{h}_{i-1}^r + b^p\right) \otimes e_q\right) \tag{2}$$

$$\vec{\alpha}_i = softmax\left(w^T \vec{G}_i + b \otimes e_q\right) \tag{3}$$

where $W^q \in \mathbb{R}^{l \times 2l}, W^p, W^r \in \mathbb{R}^{l \times l}, b^p, w \in \mathbb{R}^l$, and $b \in \mathbb{R}$ are parameters to be learned and $\vec{G}_i \in \mathbb{R}^{l \times q}, \vec{\alpha}_i \in \mathbb{R}^q$. The $\otimes e_q$ operator indicates that the argument to the left is tiled vertically q times in order to make the dimensions line up correctly. The attention vector $\vec{\alpha}_i$ is then used to collapse the question into a single feature vector $H^q \vec{\alpha}_i^T$, and this single feature vector is concatenated with the initial encoding of the word in the paragraph to create a new vector $\vec{z}_i$ serving as the input to generate the next state of the Match LSTM $\overrightarrow{\mathbf{H}}^r$. Based on this, Match-LSTM is able to learn a sequential representation of the paragraph that is semantically relevant to the particular question.

$$\vec{z}_i = \begin{bmatrix} h_i^p \\ H^q \ \vec{\alpha}_i^T \end{bmatrix} \tag{4}$$

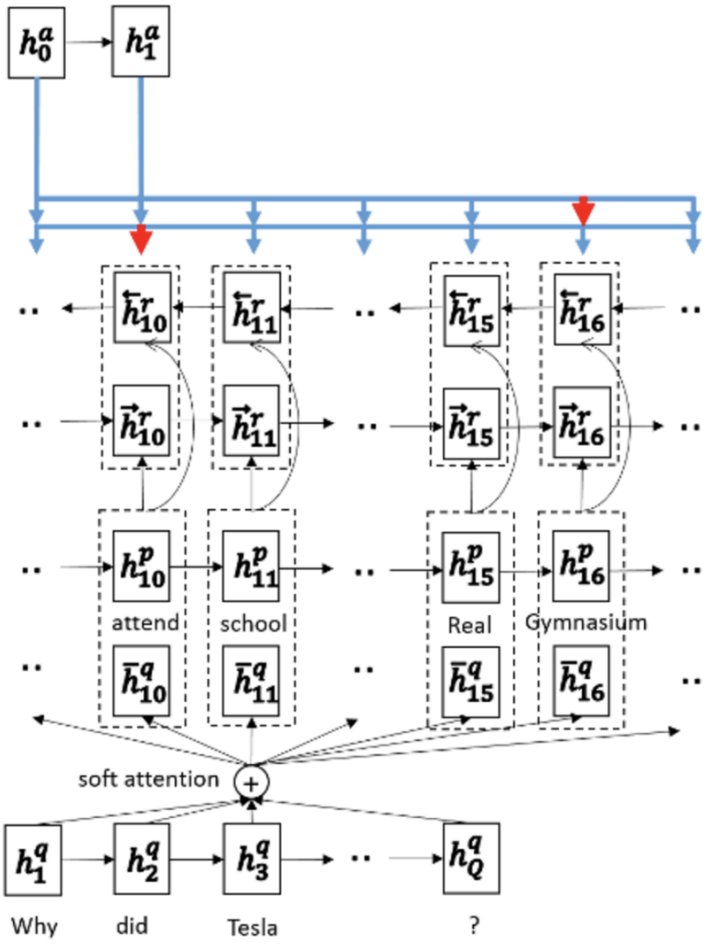

Figure 1: Match-LSTM with Answer Pointer

Match LSTM is also run in the backward direction by reversing the input and feeding it through the same Match-LSTM resulting in a vector $\overleftarrow{\mathbf{H}}^r$. The outputs from both directions are concatenated to attain our final matrix H^r

$$\mathbf{H}^r = \left[\overrightarrow{\mathbf{H}}^r, \overleftarrow{\mathbf{H}}^r\right] \tag{5}$$

4.3 Output layer

The output layer utilizes a modified version of the original Pointer-Net to predict a start and end of answer within the context paragraph by utilizing the output of the previous Match-LSTM later H^r

5 Experiments

5.1 Data

We used the Stanford Question Answering Dataset 2.0(SQuAD v2.0) for model training, validation and testing. SQuAD is a collection of question-answer pairs derived from Wikipedia articles. In total we have 141,934 Question-Answer pairs. The data has been split into a training set (with 129,941

Question-Answer Pairs), a development set (with 6078 Question-Answer Pairs) and a test set of 5915 Questions. It is worth noting that dev and test sets are a subset of the Offical SQuAD dev set obtained by splitting the official dev set in half. The answerable vs unanswerable questions distributions are listed in 5.1. We also upsampled the answerable questions and unanswerable questions by 100 percent separately. The distribution after upsampling is also listed in 5.1.

	answerable	**unanswerable**
Train	66.6	33.4
Dev	47.9	52.1
Ans Upsampled	80.0	20.0
Unans Upsampled	50.0	50.0

Table 1: Dataset distribution.

5.2 Evaluation method

We use three main metrics(EM, F1 and AvNA) to evaluate how well our model does compared to the benchmark. EM measures the exact match percentage of the model output in comparison with the ground truth answers. F1 measures the average overlap between the prediction and the ground truth answer. AvNA measures the accuracy of answerable predicted versus unanswerable predicted.

Additionally, to gain more insights into how the models perform in different scenarios, we introduced several more metrics to help understand, namely, precision and recall of answerable and unanswerable predicted. We also reported the unanswerable predicted percentage on the dev set to help understand how likely a model is to predict a question as unanswerable.

5.3 Experimental details

For the BiDAF model, we followed the default setting, batch size 64, epoch 30, starting learning rate 0.5. For Match-LSTM, we encountered cuda memory issue with batch size 64, so we downsized it a bit to 50. Learning rate is set to 0.002 constant. And we trained it for 10 epochs. 30 epochs training of the BiDAF and 10 epochs training of Match-LSTM takes around 5 hours and 6 hours on a Tesla V-100, respectively. It shows that the Match-LSTM model is about 3 times slower than the BiDAF model unfortunately.

5.4 Results

Having validated the correctness of the implementation of the Match-LSTM model on SQuAD v1.0 which is what it is designed for when the model was released. We adapted the model to predict no answers for SQuAD v2.0. The dev result is listed in 5.4. Unfortunately, the table shows that Match-LSTM significantly underperforms the BiDAF model. To better understand why this happens, we ran some extra analysis on the predictions. The metrics we looked at in the analysis include answerable prediction precision, answerable prediction recall, unanswerable prediction precision, unanswerable prediction recall, EM and F1 scores on answerable questions that the model also predicts as answerable, and the unanswerable prediction distribution compared to the actual unanswerable question distribution. The analysis result is reported in 5.4 and 5.4.

model	**EM**	**F1**	**AvNA**
BiDAF	**57.91**	**61.4**	**68.32**
Match-LSTM	48.63	51.6	59.38

Table 2: BiDAF versus Match-LSTM.

The analysis shows that Match-LSTM model performs decently well in answerable questions which are also predicted as answerable by the model. Even though still worse than the BiDAF model, but we see a **+19.33** increase in EM, and a **+25.2** increase in F1. The gap between BiDAF and Match-LSTM gets narrowed from **-9.28** and **-9.8** to **-6.35** and **-6.01**, EM and F1 respectively when only looking

model	ans prec	ans rec	unans prec	unans rec
BiDAF	**62.89**	**81.46**	**76.66**	**55.88**
Match-LSTM	56.07	70.03	64.31	49.59

Table 3: BiDAF versus Match-LSTM detailed analysis.

model	EM ans	F1 ans	unans pred	unans actual
BiDAF	**74.31**	**82.81**	37.21	52.12
Match-LSTM	67.96	76.8	**40.19**	52.12

Table 4: BiDAF versus Match-LSTM detailed analysis cntd.

at the answerable predicted. At the same time, the analysis shows that the answerable precision is worse than the unanswerable precision, standing at a merely **56.07**. The gap between Match-LSTM and BiDAF in terms of answerable recall is also larger than that in terms of unanswerable recall. So we decided to upsample the answerable questions to try to make the model learn to predict more questions as answerable and more precisely.

5.5 Answerable Upsampled

To do that, we upsampled the answerable questions by 100 percent on the training set and reran the experiment. The result is reported in table 5.5, 5.5 and 5.5 along with other ablation experiments that will be mentioned later. Underlined and bold text represents the best score of the table (except for the greyed out rows); bold text represents the second best score. Surprisingly, the overall performance of the model trained on answerable upsampled is actually worse than the original one. Taking a closer look at the analysis, we can see that the performance on the answerable ones predicted does enjoy a **3.49** and **3.45** gain. The recall on answerables also jumps to **84.02**, a **13.99** gain, with only a slight decrease on the precision, **-1.77**. It's safe to say that upsampling does help the model performs better in terms of learning a more precise answer. The downside, however, is that the model predicts too few as unanswerable. The unanswerable prediction distribution drops to **25.91**, a **-14.28** drop, unanswerable recall also drops **14.55**. Therefore, the overall performance actually drops.

model	EM	F1	AvNA
Match-LSTM	48.63	51.6	59.38
Match-LSTM features off	**50.3**	**53.33**	**60.5**
Match-LSTM ans upsampled	47.01	50.55	58.49
Match-LSTM ans upsampled*[1]	**51.88**	**54.51**	**60.46**
Match-LSTM unans upsampled	52.07	52.11	52.4
Match-LSTM unans upsampled w/o search	52.02	52.1	52.42

Table 5: Match-LSTM ablation study.

model	ans prec	ans rec	unans prec	unans rec
Match-LSTM	56.07	**70.03**	64.31	49.59
Match-LSTM features off	**57.18**	69.7	**65.15**	**52.05**
Match-LSTM ans upsampled	54.3	**84.02**	**70.48**	35.04
Match-LSTM ans upsampled*	**57.68**	65.43	63.77	**55.90**
Match-LSTM unans upsampled	61.04	1.62	52.29	99.05
Match-LSTM unans upsampled w/o search	59.78	1.89	52.31	98.83

Table 6: Match-LSTM ablation detailed study.

[1]Loss is doubled for unanswerable questions

model	EM ans	F1 ans	unans pred	unans actual
Match-LSTM	67.96	76.8	40.19	52.12
Match-LSTM features off	69.43	78.52	**41.64**	52.12
Match-LSTM ans upsampled	**71.45**	**80.25**	25.91	52.12
Match-LSTM ans upsampled*	**72.58**	**80.98**	**45.69**	52.12
Match-LSTM unans upsampled	57.45	61.98	98.73	52.12
Match-LSTM unans upsampled w/o search	56.36	64.25	98.49	52.12

Table 7: Match-LSTM ablation detailed study ctnd.

5.6 Unanswerable Upsampled

Having learned this, we decided to try to replicate the dev set distribution in the training set to see if we can get better at distinguishing answerable versus unanswerables. Therefore, we upsampled the unanswerable questions by 100 percent in the training set, which results in a 50-50 split between answerable and unanswerable in the training set. The result on the unanswerable upsampled training set is included in the same table, highlighed in light grey. The result is a little disguising just looking at the EM and F1 score, **52.07** and **52.11** as they are the highest score we achieved so far. However, when we take a closer look at the analysis result, it becomes clear why that's the case. The model simply predicts almost all questions **98.73** percent, as unanswerable. Therefore the EM and F1 score simply represent the unanswerable distribution in the dev set. After giving it a second thought, we realize it is indeed much easier for the model to default to predict a question as unanswerable since all unanswerable questions' ground truths are simply [0,0] while the answerable questions' ground truths are randomly distributed across all possible indexes. Therefore, it is much more likely for the model to predict [0,0] as the answer span.

5.7 Answerable Upsampled + double loss calculation on unanswerables

Realizing that upsampling unanswerable questions is not an actual solution to the drop of unanswerable recall problem introduced by upsampling answerable questions, if we still want to enjoy the benefit of a more precise answer span prediction from upsampling answerable questions, we need another solution. To do so, it comes to us that the biggest problem of the Match-LSTM model, worsened by upsampling is the incapability of distinguishing answerable from unanswerable questions. To alleviate that issue, we doubled the loss for unanswerable questions during training so that the model is punished more for wrong answerable/unanswerable classification instead of imprecise answer span prediction. And the result, included as Match-LSTM ans upsampled* in the table, shows that indeed our idea works to some extent. It achieves the best EM and F1 score of all the experiments we have run on Match-LSTM, with the second best AvNA score. Compared to answerable sampling without doubled loss on unanswerables, it achieves a **4.87** gain of EM, a **3.96** gain of F1, and a **1.97** gain of AvNA. It also significantly predicts more questions as unanswerables, from 25.91 to 45.69 percent of all dev questions. The unanswerable questions recall also jumps from 35.04 to **55.90**.

5.8 Misc

We also ran the experiments with the feature flags turned off. Surprisingly, the performance seems to be better than incorporating the additional features. We do not have a very good explanation for it except for that the additional features point the model to the opposite direction in the hypothesis space.

The analysis for without grid search one does not really add much value to the overall analysis since we did not anticipate that the model is simply going to predict everything as unanswerable. However, the time log does show us that doing grid search does not add extra training time.

6 Analysis

Besides the analysis we covered in the result section, we ran some additional analysis on the prediction result. The table below 6 shows that even though our model does not perform very well overall, it

does achieve a very good performance on short answerable questions. It achieves an EM score of **71.44** and an F1 score of **74.12**. At the same time, it performs poorly on answerable questions, getting a mere 25.91 EM score on those questions. Therefore, it is safe to say that our model is pretty reliable when it comes to short one word answerable question; however, for questions with long answers or especially unanswerable questions, it is better to turn to other models.

	EM	**F1**
Ans len = 1	**71.44**	**74.12**
Ans len > 1	54.27	64.05
No ans	25.91	25.91

Table 8: Prediction analysis on Match-LSTM model trained on answerable questions upsampled.

Furthermore, as the results from previous tables show, our model getting an AvNA score around 60, is not very good at telling answerable questions from unanswerable questions. However, once it predicts the question as answerable, if the question is indeed answerable, it can capture the answer very precisely. Therefore, we will say that our model is in general very suited for short answerable questions.

7 Conclusion

In conclusion, we have successfully adapted Match-LSTM from SQuAD v1.0 to SQuAD v2.0 and predict 'no answer' for unanswerable questions. We have also done a lot of valuable experiments that helped us develop many insights into the capabilities and limitations of the Match-LSTM model. According to the experimental result from the original Match-LSTM paper, the model performs better on SQuAD v1.0 than SQuAD v2.0. The main reason of its failure on SQuAD v2.0 is its incapability of distinguishing between the answerable and the unanswerable questions. One of our biggest finding is that the model is actually very good at predicting a precise answer span for the answerable questions, but since it's incapable of telling the answerables from the unanswerables, it also predicts a lot of wrong answers for the unanswerable questions. Overall, given its performance on short answerable questions, we think it is safe to say that Match-LSTM is still a reliable model for short answerable questions.

For the future work, we suggest keep looking into improving Match-LSTM's capability of distinguishing the answerables from the unanswerables. Our solution is to double count the loss on unanswerable questions, but it only works to some extent and does not fully resolve the issue.

References

[1] Shuohang Wang and Jing Jiang, "Machine Comprehension Using Match-LSTM and Answer Pointer". arxiv.org/abs/1608.07905

[2] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang." Squad: 100, 000+ questions for machine comprehension of text". https://arxiv.org/abs/1606.05250

[3] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks, 2015.