CS224n PA4: Extending Match-LSTM

Sebastian Goodman
sebastian.goodman@gmail.com

Keegan Mosley
keegan.r.mosley@gmail.com

Abstract

We propose two novel extensions to the Match-LSTM Boundary model for question answering on the SQuAD dataset. First we propose doing attention in the passage and question encoders. Second we propose adding a one-way conditional dependency between start-of-span and end-of-span prediction. In our evaluations, we show that these extensions result in a model that outperforms our implementation of vanilla Match-LSTM, suggesting a direction for future research.

1 Background

The task of automated question answering has received a great deal of attention from the modeling community. Answering a natural language question implies a deep understanding of a passage and question text, so it can be seen as an important problem in natural language processing.

The SQuAD dataset Rajpurkar et al. [2016] in particular is relevant to this paper. It is a dataset consisting of question, passage, and answer tuples, where the answer occurs verbatim in the passage. Existing approaches to this dataset include a logistic approach evaluated by that paper and the Match-LSTM approach described in Wang and Jiang [2016].

Neural models such as the Match-LSTM approach achieved near-state-of-the-art performance using attention and answer-pointer mechanisms. In this paper, we build on the Match-LSTM approach by trying out different architecture variations on the original model.

2 Problem Statement

In this research, we attempt to improve on the performance of the question-answering Match-LSTM model on SQuAD. We propose two architectural changes to the model and a new training technique. We evaluate these changes in terms of model performance on the F1 and EM metrics and report results.

3 Architecture: Extending Match-LSTM

In this section, we describe how we propose to modify the Match-LSTM model to address some of its limitations. We propose adding attention at the encoder layers, in order to allow contextual information to be used at encoding time. We also propose a new type of decoder layer to address the tenuous connection between span-start and span-end predictions. We do not propose any changes to the Match-LSTM layer itself; that component remains in place in all of our runs.

3.1 Extension 1: Attention at the encoder layer

The first extension we propose to Match-LSTM is attention at the encoder layer. In Wang and Jiang [2016], question and context are encoded separately, without any conditioning between them:

30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

$$\mathbf{H}^c = \overrightarrow{LSTM}(\mathbf{C})$$

$$\mathbf{H}^q = \overrightarrow{LSTM}(\mathbf{Q})$$

Here, $\mathbf{C}$ and $\mathbf{Q}$ are context and question embeddings, respectively. $\mathbf{H}^c$ and $\mathbf{H}^q$ are hidden representations of the context and question, i.e. the encoded representations at each timestep. We believe that not having context at encoding time misses a valuable opportunity to efficiently encode contextual information.

Our proposal modifies these two equations as follows:

$$\mathbf{H}^c = \overrightarrow{LSTM}(\mathbf{C}, attn(\mathbf{Q}))$$

$$\mathbf{H}^q = \overrightarrow{LSTM}(\mathbf{Q}, attn(\mathbf{C}))$$

This gives each encoders the ability to record contextual meaning for each word, as they relate to words in the other input text (see 3.1 for a detailed illustration).

We believe this will help when words in the question inform the interpretation of the context. For instance, in the case of a "When..." question, the encoder might want to encode dates such as "November 1986" differently. Or, consider the following instance, taken from the training set:

```
Q: What clauses does the 14th Amendment include ?

C: The Fourteenth Amendment to the United States Constitution ( Amendment XIV )
   is one of the post-Civil War amendments , intended to secure rights for former
   slaves . It includes the due process and equal protection clauses among others .
   The amendment introduces the concept of incorporation of all relevant federal
   rights against the states . While it has not been fully implemented , the
   doctrine of incorporation has been used to ensure , through the Due Process
   Clause and Privileges and Immunities Clause , the application of most of the
   rights enumerated in the Bill of Rights to the states .

A: due process and equal protection clauses
```

In this example, the answer is immediately followed and preceded by words in the question. We believe that this is a common pattern. We hypothesize that encoding with attention will be an improvement over encoding without it.

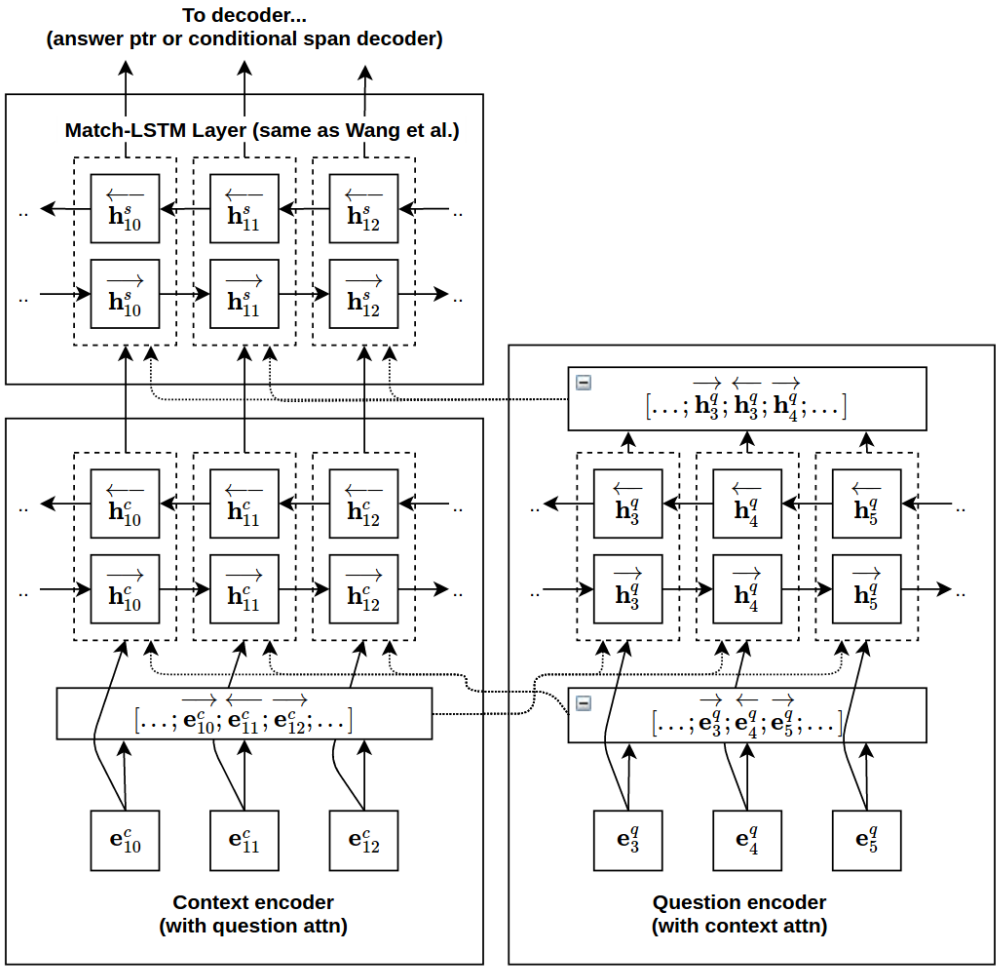

Figure 1: Our question and context encoders, based on the encoders in Wang and Jiang [2016]. The only difference is attention at the encoder/embedding level (dotted arrows indicate attention inputs).

3.2 Extension 2: Conditional Span Decoder

The second extension we propose to the Match-LSTM model is a new decoder architecture called Conditional Span Decoder, meant to replace the answer-pointer network described in Wang and Jiang [2016].

In the answer-pointer decoder, the span-start and span-end positions are predicted co-dependently (at least in the best-performing version of the model). In probabilistic terms, the Answer-Pointer boundary model chooses the most likely start index (a_s) and end index (a_e) based on the following formula:

$$P(a_e = i, a_s = j|C, Q) = P(a_e = i|C, Q)P(a_s = j|C, Q)$$

We propose a new decoder architecture that models a one-way conditional dependency between these probabilities, so that the start prediction informs the end prediction. Specifically, we rewrite the formula as:

$$P(a_e = i, a_s = j|C, Q) = P(a_e = i|a_s = j, C, Q)P(a_s = j|C, Q)$$

To accomplish this, we add a new unidirectional RNN for the span end. Its inputs are the predictions of the span begin decoder, i.e. $P_{pred}(a_s = j|C, Q)$, and the Match-LSTM outputs H^s. See Figure 3.2 for a detailed diagram.

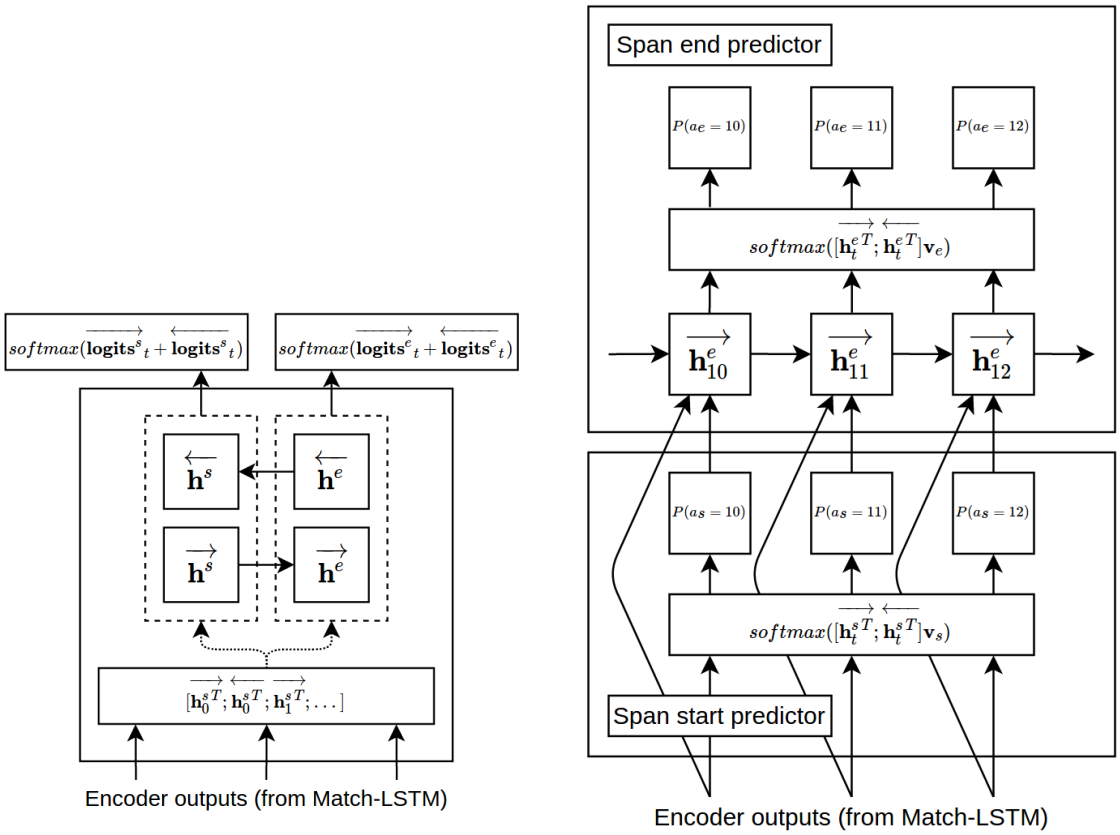

Figure 2: Answer-Pointer from Wang and Jiang [2016] (left) vs our Conditional Span Decoder (right).

We hypothesize that reformulating the model with this conditional dependency will improve prediction quality. Specifically, we believe that this will cut down on cases where the start and end predictions are "out-of-sync", i.e. cases where the predicted span is far too long. In addition, we hypothesize that overall quality of predictions will improve, due to the additional power of a full RNN for the span end decoding, rather than the 2-cell LSTM used by Answer-Pointer.

3.2.1 Label-Mixing

We evaluated a training technique which we refer to as label-mixing, which is only possible due to the structure of our Conditional Span Decoder. During label-mixing, ground-truth labels are fed into the span end decoder in addition to or instead of the predictions of the span start predictor.

We evaluated two flavors of label-mixing. First, we try a probabilistic approach where the ground truth labels, i.e. one-hot vectors which we will call $P_{gt}(a_s = j|C, Q)$, are sometimes fed instead of the span start decoder outputs, according to an annealed probability $z(t)$:

$$z(t) = max(0, 1 - training\_step/\lambda)$$

Where λ is a hyperparameter which determines the annealing schedule.

We also try a weighted approach:

$$P(a_s = j|C, Q) = z(t)P_{gt}(a_s = j|C, Q) + (1 - z(t))P_{pred}(a_s = j|C, Q)$$

We hypothesize that the loss of the span end decoder in either of these configurations will be less than the loss of Answer-Pointer (note that losses are directly comparable since they both involve

softmax cross-entropy over the context length for both a_e and a_s). In addition, we hypothesize that this will lead to better predictions because the span end predictor will a better idea of what it is trying to model, since periodically it is fed "clean" input probabilities, rather than the outputs of another RNN.

3.3 Inference

At inference time, we use two different approaches, depending on the decoder.

When using Answer-Pointer, we use the same inference used in the Match-LSTM paper. We do an exhaustive $O(C^2)$ search to find $argmax_{i,j}(P(a_s = i|C, Q)P(a_e = j|C, Q))$, where C is the context size.

When using Conditional Span Decoder, we first select $argmax_j(P(a_s = j|C, Q))$ and from there search forward to find $argmax_i(P(a_e = i|a_s = j, C, Q))$. This reflects the modeling decision of choosing the span end as a surrogate problem.

4 Methodology

We ran several experiments to showcase our architecture changes, first we established our baseline implementation of the Match LSTM paper (a simple question/context encoding layer, followed by the Match LSTM attention layer, then the Answer Pointer decoder). We then ran an experiment enabling attention in the initial encoding layer. This was followed by another experiment where we replaced the decoder with our Conditional Span Decoder (CSD). Finally, we ran a network utilizing both attention in the encoder and CSD. The experiments evaluated F1[1] and Exact Match (EM) percentages over a holdout dataset, results can be seen in Table 1

The SQuAD dataset contains 81386 training examples, 4284 validation examples, and a "secret" test set used for final evaluation. We evaluate the performance of our model using the F1 and EM (exact match) metrics. We use GloVe 100-d word embeddings for initialization.

We use the same hyperparameters used in Wang and Jiang [2016], 150-unit LSTMs, learning rate of .001, adaptive optimizer (adam; adamax was used in the paper but was not available in TensorFlow), context limit of 300 and question limit of 30.

We compute the F1 metric across the evaluation set as

$$F1 = \frac{2\mathbf{E}[recall]\mathbf{E}[precision]}{\mathbf{E}[recall] + \mathbf{E}[precision]}$$

We compute the EM metric as a simple accuracy

$$EM = \frac{num\_correct\_spans}{num\_spans}$$

.

In terms of EM, a correctly predicted span is defined as one where the model predicts the words of the answer exactly correctly, in the correct order. F1 is more lenient in that it allows partial credit to be given for predictions that partially overlap the ground truth.

5 Results

The detailed results of our experiments can be found in Tables 1 and 2. The first shows the performance of our four primary tests showcasing our proposed architecture changes. The second shows the results of a battery of tuning experiments we ran plus the results of using the label-mixing technique.

The experiments show that adding attention to the encoders (i.e. Extension 1) yields about a 35% boost to the F1 score and 57% boost to Exact Match vs the baseline model and and 14% F1 and 20% EM vs the CSD-only experiment.

[1]Our $F1$ score is computed as $F1 = 2\mathbf{E}[recall]\mathbf{E}[precision]/(\mathbf{E}[recall] + \mathbf{E}[precision])$, not as $F1 = \mathbf{E}[2 * recall * precision/(recall + precision)]$, so that is why our F1 numbers are higher. We discovered this bug very late so we kept the results to keep things consistent.

The experiments also show no significant gains for using CSD instead of Answer Pointer (i.e. Extension 2) when you are also using attention encoding (Extension 1). The slight gain in F1 and loss in EM can be attributed to experimental noise.

We ran several tuning experiments for our best model. We have included the results for some of these runs in Table 2. For reference, "Init RNN States" refers to setting the initial state on the initial context encoder to the output state of the question encoder. "Dropout" refers to adding 20% dropout to the initial encoder and Match LSTM outputs. "No LR Decay" refers to removing the polynomial decay from our learning rate. "Layer Norm LSTM Cell" refers to replacing LSTM cells with the LayerNormBasicLSTMCell Tensorflow implementation of [Lei Ba et al. [2016], Semeniuta et al. [2016]].

We also experimented a little with manual post-training answer calibrations. We applied a prior probability onto answer lengths to adjust span selection from the learned outputs and also blocking unlikely words from appearing at the start or end of answers. Both of these yielded neutral-worse performance, which we're happy about since it implies less low hanging fruit for improving obviously incorrect answers.

Table 1: Base Experiment Results

Name	$F1$[1]	Exact Match
Match LSTM + Answer Pointer	.5088	.2830
Match LSTM + CSD	.5997	.3704
Match LSTM + Answer Pointer + Attention Encoding	**.6865**	.4440
Match LSTM + CSD + Attention Encoding	.6822	**.4460**

Table 2: Tuning Experiment Results

Name	$F1$[1]	Exact Match
Attention Encoding + Match LSTM + CSD + Weighted Label Mixing	.6593	.4260
Attention Encoding + Match LSTM + CSD + Probabilistic Label Mixing	.6420	.4260
Attention Encoding + Match LSTM + CSD + Init RNN States	**.6817**	.4510
Attention Encoding + Match LSTM + CSD + Dropout	.6573	.4320
Attention Encoding + Match LSTM + CSD + No LR Decay	.6670	**.4560**
Attention Encoding + Match LSTM + CSD + Layer Norm LSTM Cell	.6656	.4330

6 Answer Analysis

See Table 3 for a breakdown of scores by first question word. We show that the performance on "when" questions is very high, as expected. The when questions are probably of the easiest variety, mostly dates. The who questions are also fairly easy for our model to predict and represent a decent bulk of the questions. Our model has a harder time with the more generic "what"-style questions, but interestingly this represents approximately half of the questions we sampled, so further analysis should include a drill down into this category.

Table 3: Scores by First Question Word

Word	Count (out of 1k)	F1	Precision	Recall
why	17	0.591072	0.521314	0.682382
which	29	0.565751	0.542529	0.591051
where	39	0.654287	0.636518	0.673077
how	73	0.647615	0.660850	0.634899
when	80	0.840939	0.837122	0.844792
who	94	0.708966	0.684929	0.734752
what	500	0.592297	0.556406	0.633138

We extracted some sample instances from one of our models for further analysis.

```
Q: What despcription was assigned to minority leader in part ?
C: The roles and responsibilities of the minority leader are not well-defined . To a large
   extent , the functions of the minority leader are defined by tradition and custom . A
   minority leader from 1931 to 1939 , Representative Bertrand Snell , R-N.Y. , provided
   this " job description " : " He is spokesman for his party and enunciates its policies .
   He is required to be alert and vigilant in defense of the minority 's rights . It is his
   function and duty to criticize constructively the policies and programs of the majority ,
   and to this end employ parliamentary tactics and give close attention to all proposed
   legislation . "
A: He is spokesman for his party and enunciates its policies .
?: Representative Bertrand Snell
precision=0.000000 recall=0.000000
```

In this instance, the model has trouble understanding the meaning of the passage/question, and guesses the most likely answer to be the name of a person. Our model is clearly focused on extracting likely "answer" candidates rather than a true understanding of what is being asked and what is in the passage.

```
Q: What denomination do these small groups belong to ?
C: There are many other Protestant denominations that do not fit neatly into the
   mentioned branches , and are far smaller in membership . Some groups of individuals
   who hold basic Protestant tenets identify themselves simply as " Christians " or "
   born-again Christians " . They typically distance themselves from the confessionalism
   and/or creedalism of other Christian communities by calling themselves "
   non-denominational " or " evangelical " . Often founded by individual pastors , they
   have little affiliation with historic denominations .
A: " non-denominational " or " evangelical "
?: Protestant
precision=0.000000 recall=0.000000
```

In this instance, the model provides a decent answer, and one could argue that the model actually did a decent job since Protestant is more of a "denomination" than "non-denominational".

```
Q: Who established the Theotokos Paregoritissa Church in 1294-96 ?
C: The Church of the Holy Apostles in Thessaloniki was built in 1310-14 . Although some
   vandal systematically removed the gold tesserae of the background it can be seen that
   the Pantokrator and the prophets in the dome follow the traditional Byzantine pattern .
   Many details are similar to the Pammakaristos mosaics so it is supposed that the same
   team of mosaicists worked in both buildings . Another building with a related mosaic
   decoration is the Theotokos Paregoritissa Church in Arta . The church was established
   by the Despot of Epirus in 1294-96 . In the dome is the traditional stern Pantokrator ,
   with prophets and cherubim below .
A: the Despot of Epirus
?: Despot of Epirus
precision=1.000000 recall=0.750000
```

In this case, the model receives a small penalty to recall for not including "the" in its answer. This seems more or less like an issue with the evaluation approach. Perhaps having multiple ground truths and matching the closest one or implementing forgiveness for stop-words would improve the quality of comparisons between models.

7 Conclusions

7.1 Use Embedding Attention [3.1] During Initial Encoding of Question/Context

Adding attention in the initial encoding layer showed a pretty clear win over both the baseline (35% F1, 57% EM) and the CSD only model (14% F1, 20% EM). Even though our model tuning didn't

yield substantially better results the fact that it never dipped as low as the baseline or CSD-only models shows that these gains are somewhat resilient to tweaks.

7.2 Don't Use CSD [3.2] for the Decoder

Although the CSD-only model did show gains over the baseline (18% F1, 31% EM), the difference between the attention-only model and the model using both changes was negligible. This implies that although there was information gain over the baseline using CSD, that information is a subset of what is gained by using embedding attention during encoding making the CSD change unnecessary.

Furthermore, we actually recommend sticking with Answer Pointer instead of CSD because we saw a model training time increase of roughly 50% because of the added RNN and softmax layers in CSD.

7.3 Don't Mix Labels Into Training CSD

Both our experiments with mixing label information into the decoders to guide their learning yielded worse models. The theory that this aids in early training when the begin decoder is noisy is probably serves more to prevent the network from getting the gradients it needs early to put it onto the path of a better convergence point.

7.4 Double Check Baseline Model

The Match LSTM paper we are emulating reported F1 results significantly higher than our baseline achieved. We're not sure right now how our implementation diverged from the paper, but it stands to reason there is some difference. In order for our results to truly be validated, we need to be able to fully reproduce the results of Match LSTM and alter that to show our attention encoder gains still hold.

8 Team Member Contributions

Sebastian wrote the architecture section of this paper and did the diagrams. Keegan gathered our experiment results together and wrote up the analysis and conclusions portions of this paper. We both ran many experiments and we both have dozens of commits into the git repo with bugfixes, model development, scripting, etc..

References

J. Lei Ba, J. R. Kiros, and G. E. Hinton. Layer Normalization. *ArXiv e-prints*, July 2016.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. *CoRR*, abs/1606.05250, 2016. URL `http://arxiv.org/abs/1606.05250`.

Stanislau Semeniuta, Aliaksei Severyn, and Erhardt Barth. Recurrent dropout without memory loss. *CoRR*, abs/1603.05118, 2016. URL `http://arxiv.org/abs/1603.05118`.

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