LSTM Encoder-Decoder Architecture with Attention Mechanism for Machine Comprehension

Brian Higgins and Eugene Nho

March 23, 2017

Abstract

Machine intelligence is an important problem to be solved for artificial intelligence to be truly impactful in our lives. While many question answering models have been explored for existing machine comprehension datasets, there has been little work with the newly released MS Marco dataset, which poses many unique challenges. We explore an end-to-end neural architecture with attention mechanisms capable of comprehending relevant information and generating text answers for MS Marco.

1 Introduction

In this paper, we propose an end-to-end neural architecture for machine comprehension capable of ingesting text input and generating answers.

Our motivation is an increasingly widening information processing gap modern decision makers are facing; text information is generated at an ever increasing rate while the human cognitive ability to ingest text is largely constant over time. We believe effective machine comprehension systems can dramatically increase our ability to close that gap.

We chose MS Marco [4] as our dataset. Unlike previous reading comprehension and question answering datasets, MS Marco consists of real questions generated by anonymized queries from Bing, and has target answers in the form of sequence of words. Compared to other forms of answers like single-token answer or span of words, generating text is significantly more challenging, and as far as we are aware, no literature exists on end-to-end systems squarely addressing this dataset.

We built a multi-encoder-decoder architecture with attention mechanism between the encoders as well as between the encoder and the decoder. We achieved ROUGE-L of 12.8 and BLEU of 9.3 on an unseen dataset, which would put us in 7th place on the MS Marco leaderboard.

2 Related Work

Machine comprehension and question answering tasks went through multiple stages of evolution over the past decade. Traditionally, these tasks relied on complex NLP pipelines involving steps like syntactic parsing, semantic parsing, and question classification. With the rise of neural networks, end-to-end neural architecture has increasingly been applied to comprehension tasks [5][6][8][9]. This progression has been tightly tied with the evolution of available datasets–from the architectures for multiple choices [8][9] to those generating single-token answers [5][6] or identifying span of words [1].

Throughout this evolution, attention has emerged as a key concept, in particular as the task demanded by dataset became more complex. Attention was first utilized more in context of non-NLP tasks such as learning alignments between image objects and agent actions [2], or between visual features of an image and text description in the caption generation task. Bahdnau et al (2015)[10] applied attention mechanism to Neural Machine Translation (NMT), and Luong et al (2015)[2] explored different architectures for attention-based NMT, including a global approach which always looks at all source words, and a local approach that only focuses on a subset of source words.

Several approaches have been made to incorporate attention into machine comprehension and question answering tasks. Seo et al, 2017[11] applied bidirectional models with attention to achieve near state-of-the-art results for SQuAD. Wang & Jiang (2016)[1] combined match-LSTM, an attention model originally proposed for text entailment [1], and Pointer Net, a sequence-to-sequence model proposed by Vinyals et al. (2015)[12] constraining output words to be from the input sequences.

3 Dataset: MS Marco

MS Marco is a reading comprehension dataset with a number of characteristics that make it the most realistic available dataset for question answering. All questions are real anonymized queries from Bing, and the context passages, from which target answers are derived, are sampled from real web documents, closely mirroring the real-world scenario of finding an answer to a question on a search engine. The target answers, as mentioned earlier, are sequence of words and are human-generated.

The dataset has 100,000 queries. Each query consists of one question, approximately 10 context passages, and target answers. While most queries have one answer, some have many and some have none. The average length of the questions, passages, and target answers are approximately 15, 85, and 8 tokens, respectively. There are no particular topical focus, but the queries fall into one of the following five categories: description (52.6%), numeric (28.4%), entity (10.5%), location (5.7%) and person (2.7%).

Given the nature of text generation task required by the dataset, we use ROUGE-L and BLEU as evaluation metrics.

4 Methods

The problem we are trying to solve can be defined formally as follows. For each example, we are given a question and a set of potentially relevant context passages. The question is represented as $Q \in \mathbb{R}^{dxn}$ where d is the embedding size and n is the length of the question. The set of passages is reprented as S = {P1, P2, ... , Pm}, where m is the number of passages, and each $P_i \in \mathbb{R}^{dxl}$ where l is the length of P_i. Our objective is to (1) choose the most relevant passage P out of S, and (2) generate the answer consisting of a sequence of words, represented as $R \in \mathbb{R}^{dxc}$ where c is the length of the answer. Because both tasks require very similar architecture, this section will focus on the architecture of the model for the second objective.

We use an encoder-decoder architecture with multiple attention mechanisms to achieve this, as shown in Figure 1. The components of this model are:

1. **Question Encoder** is an LSTM layer mapping question information to a vector space.
2. **Passage Encoder with Attention** is an LSTM layer with an attention mechanism connecting passage information with the encoded knowledge about the question.
3. **Passage-Attention Encoder** is an extra LSTM layer further distilling the output from the Passage Encoder.
4. **Decoder** is an LSTM layer that takes in information from the three encoders in the form of the initial hidden state and generates answers.

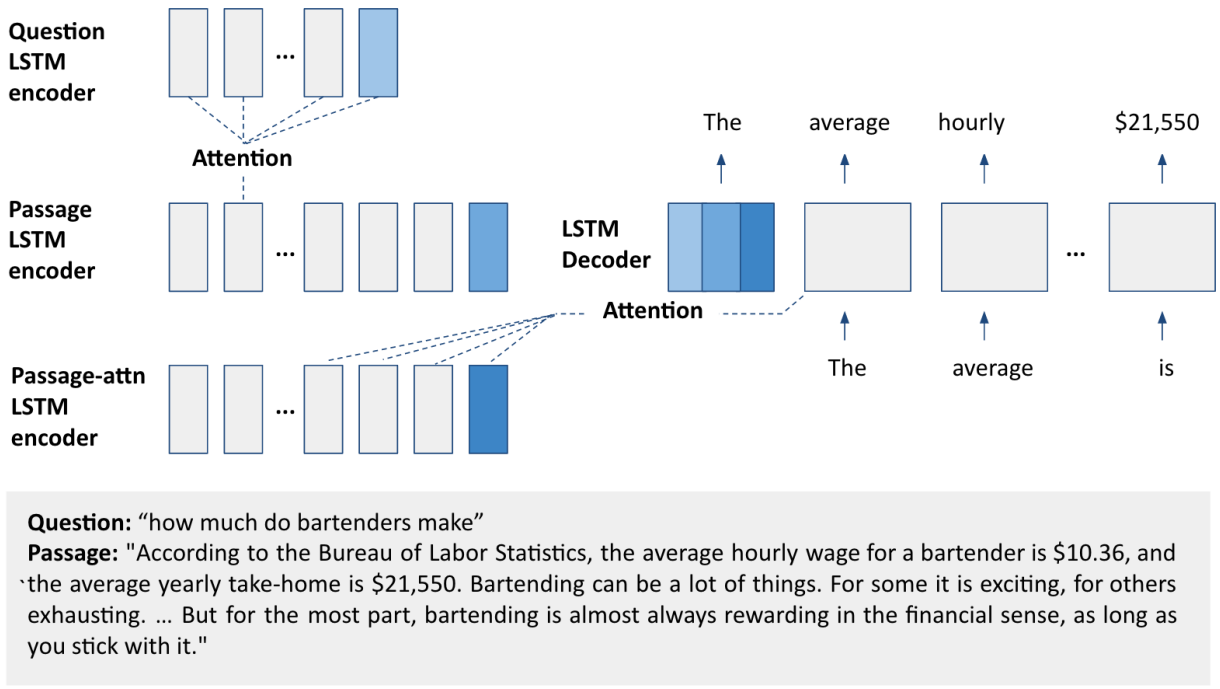

Figure 1: Architecture of the encoder-decoder model with attention

4.1 Question Encoder

The purpose of the question encoder is to incorporate contextual information from each token of the question to a vector space. We use a standard unidirectional LSTM [13] layer to process the question, represented as follows:

$$H^q = \hat{LSTM}(Q) \tag{1}$$

The output matrix $H^q \in \mathbb{R}^{hxk}$ is the hidden state representation of the question, where h is the size of the hidden state dimension. h_i^q represents the i^{th} column of H^q, and encapsulates the contextual information up to the i^{th} token of the question (q_i).

4.2 Passage Encoder with Attention

The objective of this layer is to capture the information from the passage tokens that is relevant to the contents of the question. To that end, it employs a standard LSTM layer along with the global attention mechanism proposed by Luong et al. [14] At each time step t, it uses the embedding of t^{th} token of the passage and the entire hidden state representation from the question encoder to capture contextual information up to the t^{th} token that is relevant to the question. This is represented as follows:

$$\tilde{h}_t^p = \vec{Attention LSTM}(p_t, H^q) \tag{2}$$

where $\tilde{h}_t^p \in \mathbb{R}^h$ is the hidden vector capturing the information, $p_t \in \mathbb{R}^d$ is the t^{th} token of the passage, and H^q is the matrix representing all the hidden states of the question encoder.

AttentionLSTM is an abstraction of the following two steps: First, it captures information from the t^{th} token using a regular LSTM cell, represented as $h_t^p = \vec{LSTM}(p_t)$, where $h_t^p \in \mathbb{R}^h$. Second, using H^q and h_t, it derives a context vector $c_t \in \mathbb{R}^h$ that captures relevant information from the questions, which in turn c_t

is concatenated with h_t^p to generate $\tilde{h}_t^p$.

$$h_t^p = \vec{LSTM}(p_t) \tag{3}$$

The context vector c_t captures all the hidden states from the question encoder weighted by how relevant each question word is to the t^{th} passage word. To derive c_t we first calculate the relevance score:

$$\text{Score}(h_t, h_s^q) = (W_s h_t + b) \odot h_s^q \tag{4}$$

We then create an attention vector a, which has the same dimension as H^q, where each element is a softmax value of score(h_t, h_s^q). c_t is the weighted average of H^q based on the attention vector a.

$$g_t = H_q \odot \bar{a} \tag{5}$$

$$c_t = \sum_i^k g_t^{(i)} \tag{6}$$

where $\bar{a} \in \mathbb{R}^{hxk}$ is the horizontally broadcasted matrix of a, and $g_t^{(i)}$ is the ith column of g_t. The output of $\vec{AttentionLSTM}(\tilde{h}_t^p)$ is calculated as follows:

$$\tilde{h}_t^p = \tanh(W_c[h_t; C_t] + b) \tag{7}$$

4.3 Passage-Attention Encoder

The purpose of this LSTM layer is to further distill the contextual information captured by the passage encoder. It is a variant of the Match LSTM layer, first proposed by Wang & Jiang (2016)[1] for text entailment and later applied to Question Answering by the same authors (Wang & Jiang (2016))[1]. We utilize a standard unidirectional LSTM layer taking as input all the hidden states of the passage encoder with attention, as shown below:

$$H^m = \vec{LSTM}(\tilde{H}^p) \tag{8}$$

where $\tilde{H}^p \in \mathbb{R}^{hxm}$ is the hidden state representation from the passage encoder.

4.4 Decoder

We use an LSTM layer with the global attention between the decoder and the Passage-Attention Encoder to generate the answer. The purpose of the attention mechanism is to let the decoder "peek" at the relevant information encapsulating the passage and the question as it generates the answer. In addition, to pass along the contextual information captured by the encoders, we concatenate the last hidden state from all three encoders and feed it as the initial hidden state to the decoder. At each time step t, the decoder takes as input the embedding of the generated token from the previous step, and uses the same attention mechanism described in the earlier section to derive the hidden state representation h_t^r (dim: 3h):

$$h_t^r = \vec{AttentionLSTM}(r_{t-1}, H^m) \tag{9}$$

where r_{t-1} is the embedding of the word generated in the last time step. Then we apply matrix multiplication and softmax to generate the output vector o_t.

$$O_t = softmax(Uh_t^r + b) \tag{10}$$

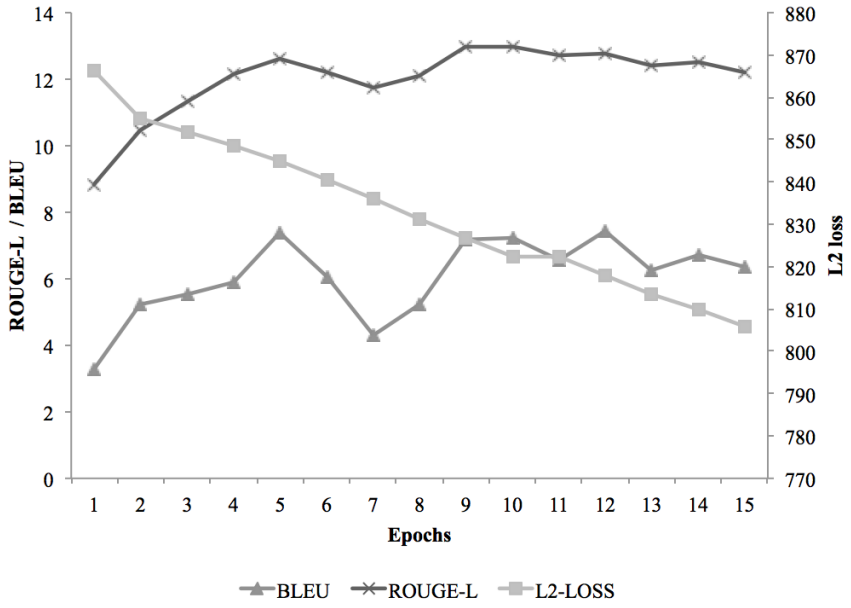

Figure 2: Loss and evaluation metrics on best model

5 Experiments

5.1 Results

We achieved ROUGE-L of 12.8 and BLEU of 9.3 with a variant of the model described earlier with a double-layer LSTM decoder. While this result is not sate-of-the-art, our model outperformed several benchmark models such as Seq2Seq model with Memory Networks (MS Marco).

Figure 2 shows the training progression for our model. Loss steadily declines, but the model starts overfitting around the 10th epoch, where both evaluation metrics peaked. Our classifier for selecting the most relevant context passage achieved an accuracy of 100% on both validation and unseen development set.

For hyperparameter optimization, we conducted approximately 40 runs, and found the following:

- Larger batch size generally performed better on our model, with a batch size of 256 outperforming 64 (9.3 vs 6.4 BLEU)
- L2 loss outperformed cross entropy (9.3 vs 7.0 BLEU)
- Our model was sensitive to learning rate, with our metrics dropping precipitously when the rate approached 0.01 range (vs 0.001 to 0.0001)

5.2 Error Analysis

As illustrated by Table 1, our model was much more effective in generating descriptions and numeric answers than people and locations. Its relative weakness on people and locations most likely occurs because of the rarity of their names. The dataset contained a total of approximately 650,000 tokens. Because of memory restrictions, we used a vocabulary size of 20,000 during training, and the rest of the words were cast to the $< unk >$ token. This clearly appeared as an error with names of people and locations, and to some degree entities.

	Description	Person	Numeric	Location	Entity	Full Data Set
BLEU-1	9.1	3.2	7.4	3.8	7.4	9.3
ROUGE-L	15.1	2.8	13.7	3.4	5.5	12.8

Table 1: Error by question type in the hidden development set

We also found that longer answers were much harder to predict. This is well known and reported in text generation tasks because the farther into the decoder the model gets, the more diluted the original hidden state becomes. Because of this, it is very challenging to predict long sequences of words.

To highlight another source of error for our model, it comes from learning the end token. At training time, we made the design decision to mask based on the length of ground truth in the model. Our hypothesis was that this would encourage our model to generate predictions that are of the same length as the target answers. However, because of this decision, the model was not penalized for generating often irrelevant tokens beyond the length of the target answer.

6 Conclusion

We see a few areas of improvement for future work. First, we would like to reduce the burden on the decoder softmax by limiting the vocabulary to only the tokens used in that particular query's question and passage. Predicting the right token out of 20,000 options is challenging, especially when the model is required to get that right 20 to 30 times in a row. Second, we would like to train a language model to provide a warm start for our decoder, so the decoder does not have to learn how to generate sensible sequence of words and the content of the passages and questions at the same time.

References

[1] Shuohang Wang and Jing Jiang. Machine comprehension using Match-LSTM and answer pointer. Under review as a conference paper at ICLR 2017, 2016.

[2] Minh-Thang Luong, Hieu Pham, Christopher D. Manning, Effective Approaches to Attention-based Neural Machine Translation. In arXiv:1508.04025, 2015.

[3] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, Hannaneh Hajishirzi, Bidirectional Attention Flow for Machine Comprehension. In Proceedings of ICLR 2017, 2017.

[4] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder and Li Deng. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. In 30th Conference on Neural Information Processing Systems, 2016

[5] Hermann, Karl Moritz et al. "Teaching Machines To Read And Comprehend". Arxiv.org. N. p., 2015. Web. 23 Mar. 2017.

[6] Kadlec, Rudolf et al. "Text Understanding With The Attention Sum Reader Network". Arxiv.org. N. p., 2016. Web. 23 Mar. 2017.

[7] Shen, Yelong et al. "Reasonet: Learning To Stop Reading In Machine Comprehension". Arxiv.org. N. p., 2016. Web. 23 Mar. 2017.

[8] Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. the Goldilocks principle: Reading childrens books with explicit memory representations. In Proceedings of the International Conference on Learning Representations, 2016.

[9] Yin, Wenpeng, Sebastian Ebert, and Hinrich Schutze. "Attention-Based Convolutional Neural Network For Machine Comprehension". Arxiv.org. N. p., 2016. Web. 23 Mar. 2017.

[10] Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural Machine Translation By Jointly Learning To Align And Translate". Arxiv.org. N. p., 2014. Web. 23 Mar. 2017.

[11] Dhingra, Bhuwan et al. "A Comparative Study Of Word Embeddings For Reading Comprehension". Arxiv.org. N. p., 2017. Web. 23 Mar. 2017.

[12] Vinyals, Oriol, Meire Fortunato, and Navdeep Jaitly. "Pointer Networks". Arxiv.org. N. p., 2015. Web. 23 Mar. 2017.

[13] Hochreiter, Sepp, and Jrgen Schmidhuber. "Long Short-Term Memory". Neural Computation 9.8 (1997): 1735-1780. Web. 23 Mar. 2017.

[14] Luong, Thang, Richard Socher, and Christopher D. Manning. "Better word representations with recursive neural networks for morphology." CoNLL. 2013.