From Answer Extraction to Answer Generation for Machine Reading Comprehension

Bhavya Patwa, *Research Intern,IIT-Bombay,*
Pankaj Singh, *Project Research Associate, IIT-Bombay,*
and Ganesh Ramakrishnan, *Associate Professor, IIT-Bombay*

Abstract—In this paper, we present results and analysis on our implementation of machine reading comprehension for the MS-MARCO dataset. The MS-MARCO dataset defines the task as answering a question from multiple passages and words in the answer are not necessarily in the passages. We have implemented from the state-of-the-art models which proposes different frameworks to answer the questions from a given set of passages. An extraction-then-synthesis framework to synthesize answers from extraction results has been proposed by the aforementioned models. Specifically, the answer extraction model is first employed to predict the most important sub-spans from the passage as evidence, and the answer synthesis model takes the evidence as additional features. The answer extraction model is built on state-of-the-art neural networks for single passage reading comprehension, and an additional task of passage ranking to help answer extraction in multiple passages. The answer synthesis model is based on sequence-to-sequence neural networks with extracted evidences as features. We conduct extensive experiments on the dataset through hyperparameter tuning, profiling ranging from individual models to end-to-end neural networks model for reading comprehension style question answering.

Index Terms—MicroSoft MAchine Reading COmprehension (MS-MARCO), Pointer Networks, Sequence-to-Sequence Neural Networks

1 Introduction

MACHINE reading comprehension [1], [2], which attempts to enable machines to answer questions after reading a passage or a set of passages, attracts great attentions from both research and industry communities in recent years. The release of the Stanford Question Answering Dataset (SQuAD) [1] and the Microsoft MAchine Reading COmprehension Dataset (MS-MARCO) [2] provides large-scale manually created datasets for model training and testing of machine learning (especially deep learning) algorithms for this task. We'll be pertaining only to the MS-MARCO dataset in this paper.

In this paper, we present extensive experiments on the MS-MARCO dataset based on two models: R-NET [3] and S-NET [4]. R-NET is an evidence extraction model with attention based neural networks which predicts the start and end positions of evidence snippets. While in S-NET, it first extracts evidence snippets by matching the question and passage while passage ranking as an additional task, and then generates the answer by synthesizing the question, passage, and evidence snippets. The evaluation of the experiments are in terms of ROUGE-L F1 scores.

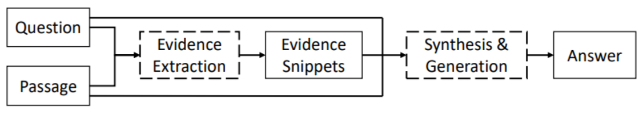


Fig. 1: Overview of S-NET. It first extracts evidence snippets by matching the question and passage, and then generates the answer by synthesizing the question, passage and evidence snippets

2 Related Work

The existing works on the MS-MARCO dataset follows their methods on SQuAD. Wang & Jiang [5] combine match-LSTM and pointer networks to produce the boundary of the answer. Wang et al. (2017) [3] apply an additional gate to the attention-based recurrent networks amd propose a self-matching mechanism for aggregating evidence from the whole passage, which achieves the state-of-the-art result on SQuAD dataset.

The sequence-to-sequence model is widely-used in many tasks such as machine translation (Bahdanau et al., 2014) [6] (Luong et al., 2015) [7]. We use it to generate the synthetic answer with the start and end positions of the evidence snippet as features.

3 RNET + SNET model approach

The proposed approach as defined in [3] consists of only evidence snippet prediction. Following the overview in Figure 1, the proposed approach as defined in [4] consists of evidence extraction and answer synthesis. The evidence extraction part aims to extract evidence snippets related to the question and passage. The answer synthesis part aims to generate the answer based on the extracted evidence snippets. We propose a multi-task learning framework for the evidence extraction shown in Figure 2, and use the sequence-to-sequence model with additional features of the start and end positions of the evidence snippet for the answer synthesis shown in Figure 3.

3.1 Gated Recurrent Unit

We use Gated Recurrent Unit (GRU) (Cho et al., 2014) [8] instead of basic RNN. Equation 1 describes the mathematical model of the GRU. r_t and z_t are the gates and h_t is the hidden state.

$$\begin{aligned} z_t &= \sigma(W_{hz}h_{t-1} + W_{xz}x_t + b_z) \\ r_t &= \sigma(W_{hr}h_{t-1} + W_{xr}x_t + b_r) \\ \hat{h_t} &= \phi(W_h(r_t \odot h_{t-1}) + W_x x_t + b) \\ h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot \hat{h_t} \end{aligned} \tag{1}$$

3.2 Evidence Extraction

R-NET [3] and S-NET [4] proposes a single-task and a multi-task learning framework respectively for evidence extraction. Unlike the SQuAD dataset, which only has one passage given a question, there are several related passages for each question in the MS-MARCO dataset. In addition to annotating the answer, MS-MARCO also annotates which passage is correct. To this end, [4] proposes improving text span prediction with passage ranking. Specifically, as shown in Figure 2, in addition to predicting a text span, we apply another task to rank candidate passages with the passage-level representation.

3.2.1 Evidence Snippet Prediction

Consider a question Q = $\{w_t^Q\}_{t=1}^m$ and a passage Q = $\{w_t^P\}_{t=1}^n$, we first convert the words to their respective word-level embeddings and character-level embeddings. The character-level embeddings are generated by taking the final hidden states of a bi-directional GRU applied to embeddings of characters in the token. We then use a bi-directional GRU to produce new representation $u_1^Q, \ldots, u_m^Q$ and $u_1^P, \ldots, u_n^P$ of all words in the question and passage respectively:

$$\begin{aligned} u_t^Q &= \text{BiGRU}_Q(u_{t-1}^Q, [e_t^Q, char_t^Q]) \\ u_t^P &= \text{BiGRU}_P(u_{t-1}^P, [e_t^P, char_t^P]) \end{aligned} \tag{2}$$

Given question and passage representation $\{u_t^Q\}_{t=1}^m$ and $\{u_t^P\}_{t=1}^n$, Rocktaschel et al. (2015) propose generating sentence-pair representation $\{v_t^P\}_{t=1}^n$ via soft-alignment of words in the question and passage as follows:

$$v_t^P = \text{GRU}(v_{t-1}^P, c_t^Q) \tag{3}$$

where $c_t^Q = att(u^Q, [u_t^P, v_{t1}^P])$ is an attention-pooling vector of the whole question (u^Q):

$$\begin{aligned} s_j^t &= \mathrm{v}^{\mathrm{T}} tanh(W_u^Q u_j^Q + W_u^P u_t^P) \\ a_i^t &= \frac{\exp\{s_i^t\}}{\sum_{j=1}^m \exp s_j^t} \\ c_t^Q &= \sum_{i=1}^m a_i^t u_i^Q \end{aligned} \tag{4}$$

Wang & Jiang (2016a) introduce match-LSTM, which takes u_j^P as an additional input into the recurrent network. Wang et al. (2017) propose adding gate to the input ($[u_t^P, c_t^Q]$) of RNN to determine the importance of passage parts.

$$\begin{aligned} g_t &= sigmoid(W_g[u_t^P, c_t^Q]) \\ [u_t^P, c_t^Q]^* &= g_t \odot [u_t^P, c_t^Q] \\ v_t^P &= \text{GRU}(v_{t-1}^P, [u_t^P, c_t^Q]^*) \end{aligned} \quad (5)$$

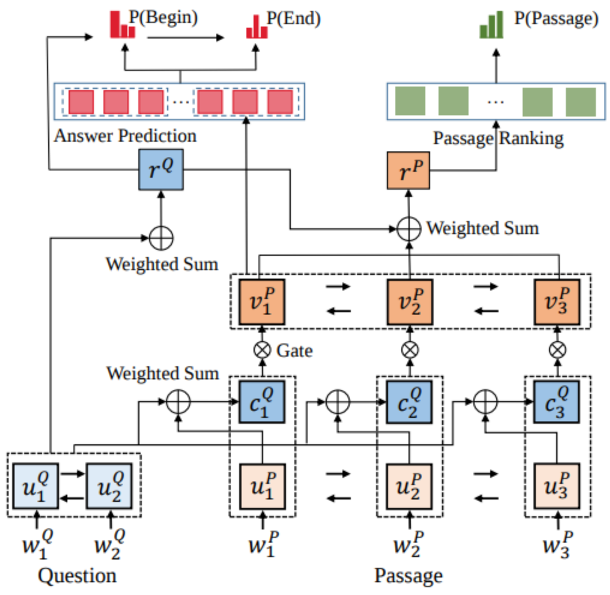


Fig. 2: Evidence Extraction Model

We use pointer networks (Vinyals et al., 2015a) to predict the position of evidence snippets. Following the previous work (Wang & Jiang, 2016b), we concatenate all passages to predict one span for the evidence snippet prediction. Given the representation $\{v_t^P\}_{t=1}^N$ where N is the sum of the length of all passages, the attention mechanism is utilized as a pointer to select the start position (p^1) and end position (p^2), which can be formulated as follows:

$$\begin{aligned} s_j^t &= \mathrm{v}^{\mathrm{T}}(W_h v_j^P + W_h^a h_{t-1}^a) \\ a_i^t &= \frac{exp(s_i^t)}{\sum_{j=1}^m \exp s_j^t} \\ p^t &= argmax(a_1^t, ..., a_N^t) \end{aligned} \quad (6)$$

Here h_{t1}^a represents the last hidden state of the answer recurrent network (pointer network). The input of the answer recurrent network is the attention-pooling vector based on current predicted probability a_t:

$$\begin{aligned} c_t &= \sum_{i=1}^N a_i^t v_i^P \\ h_t^a &= \text{GRU}(h_{t-1}^a, c_t) \end{aligned} \quad (7)$$

When predicting the start position, h_{t1}^a represents the initial hidden state of the answer recurrent network. We utilize the question vector r^Q as the initial state of the answer recurrent network. $r^Q = att(u^Q, v_r^Q)$ is an attention-pooling vector of the question based on the parameter v_r^Q:

$$\begin{aligned} s_j &= \mathrm{v}^{\mathrm{T}} tanh(W_u^Q u_j^Q + W_v^Q v_r^Q) \\ a_i &= \frac{exp(s_i)}{\sum_{j=1}^m exp(s_j)} \\ r^Q &= \Sigma_{i=1}^m a_i u_i^Q \end{aligned} \quad (8)$$

For this part, the objective function is to minimize the following cross entropy:

$$\mathcal{L}_{AP} = -\sum_{t=1}^{2}\sum_{i=1}^{N}[y_i^t \log a_i^t + (1-y_i^t)\log(1-a_i^t)] \quad (9)$$

where $y_i^t \in 0, 1$ denotes a label. $y_i^t = 1$ means i is a correct position, otherwise $y_i^t = 0$.

3.2.2 Passage Ranking

In this part, we match the question and each passage from word level to passage level. Firstly, we use the question representation r^Q to attend words in each passage to obtain the passage representation r^P where $r^P = att(v^P, r^Q)$.

$$\begin{aligned} s_j &= \mathrm{v}^{\mathrm{T}} tanh(W_v^P v_j^P + W_v^Q r^Q) \\ a_i &= \frac{exp(s_i)}{\sum_{j=1}^n exp(s_j)} \\ r^P &= \sum_{i=1}^n a_i v_i^P \end{aligned} \quad (10)$$

Next, the question representation r^Q and the passage representation r^P are combined to pass two fully connected layers for a matching score,

$$g = v_g^{\mathrm{T}}(tanh(W_g[r^Q, r^P])) \tag{11}$$

For one question, each candidate passage P_i has a matching scoreg_i. We normalize their scores and optimize following objective function:

$$\hat{g}_i = \frac{exp(g_i)}{\sum_{j=1}^{k} exp(g_j)}$$

$$\mathcal{L}_{PR} = -\sum_{i=1}^{k}[y_i \log \hat{g}_i + (1 - y_i) log(1 - \hat{g}_i)] \tag{12}$$

where k is the number of passages. $y_i \epsilon \{0, 1\}$ denotes a label. $y_i = 1$ means P_i is the correct passage, otherwise $y_i = 0$.

3.2.3 Joint Learning

The evident extraction part is trained by minimizing joint objective functions:

$$\mathcal{L}_E = rL_{AP} + (1 - r)\mathcal{L}_{PR} \tag{13}$$

where r is the hyper-parameter for weights of two loss functions.

3.3 Answer Synthesis

As shown in Figure 3, we use the sequence-to-sequence model to synthesize the answer with the extracted evidences as features. We first produce the representation h_t^P and h_t^Q of all words in the passage and question respectively. When producing the answer representation, we combine the basic word embedding e_t^p with additional features f_t^s and f_t^e to indicate the start and end positions of the evidence snippet respectively predicted by evidence extraction model. $f_t^s = 1$ and $f_t^e = 1$ mean the position t is the start and end of the evidence span, respectively.

$$h_t^P = \text{BiGRU}(h_{t-1}^P, [e_t^p, f_t^s, f_t^e])$$
$$h_t^Q = \text{BiGRU}(h_{t-1}^Q, e_t^Q) \tag{14}$$

On top of the encoder, we use GRU with attention as the decoder to produce the answer. At each decoding time step t, the GRU reads the previous word embedding w_{t1} and previous context vector c_{t1} as inputs to compute the new hidden state d_t. To initialize the GRU hidden state, we use a linear layer with the last backward encoder hidden state $\overleftarrow{h_1}^P$ and $\overleftarrow{h_1}^P$ as input:

$$d_t = \text{GRU}(w_{t-1}, c_{t-1}, d_{t-1})$$
$$d_0 = tanh(W_d[\overleftarrow{h_1}^P, \overleftarrow{h_1}^Q] + b) \tag{15}$$

where W_d is the weight matrix and b is the bias vector.

The context vector c_t for current time step t is computed through the concatenate attention mechanism [7], which matches the current decoder state d_t with each encoder hidden state ht to get the weighted sum representation. Here h_i consists of the passage representation h_t^P and the question representation h_t^Q.

$$s_j = v_a^T tanh(W_a d_{t-1} + U_a h_j)$$
$$a_i = \frac{exp(s_i)}{\sum_{j=1}^{n} exp(s_j)}$$
$$c_t = \sum_{i=1} n a_i^t h_i \tag{16}$$

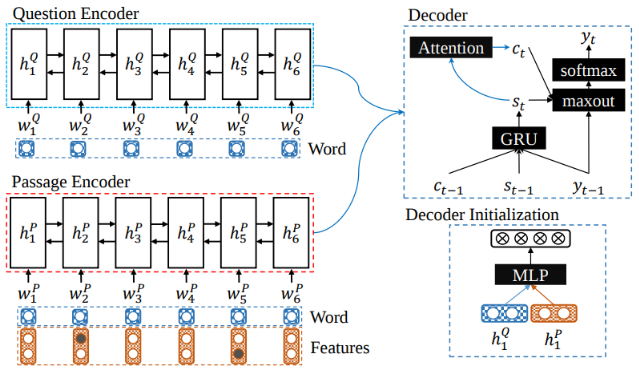

Question Encoder
h_1^Q h_2^Q h_3^Q h_4^Q h_5^Q h_6^Q
w_1^Q w_2^Q w_3^Q w_4^Q w_5^Q w_6^Q
Word
Passage Encoder
h_1^P h_2^P h_3^P h_4^P h_5^P h_6^P
w_1^P w_2^P w_3^P w_4^P w_5^P w_6^P
Word
Features
Decoder
Attention
c_t
y_t
softmax
maxout
s_t
GRU
c_{t-1}
s_{t-1}
y_{t-1}
Decoder Initialization
MLP
h_1^Q
h_1^P

Fig. 3: Answer Synthesis Model

We then combine the previous word embedding w_{t1}, the current context vector c_t, and the decoder state d_t to construct the readout state r_t. The readout state is then passed through a maxout hidden layer (Goodfellow et al., 2013) to predict the next word with a softmax layer over the decoder vocabulary.

$$r_t = W_r w_{t-1} + U_r c_t + V_r d_t$$
$$m_t = [max\{r_{t,2j-1}, r_{t,2j}\}]$$
$$p(y_t|y_1, ..., y_{t-1}) = \text{softmax}(W_o m_t) \quad (17)$$

where W_a, U_a, W_r, U_r, V_r and W_o are parameters to be learned. Readout state rt is a $2d$-dimensional vector, and the maxout layer (Equation 17) picks the max value for every two numbers in r_t and produces a d-dimensional vector m_t.

Our goal is to maximize the output probability given the input sentence. Therefore, we optimize the negative log-likelihood loss function:

$$\mathcal{L}_S = -\frac{1}{|D|}\Sigma_{(X,Y)\in D} \log p(Y|X) \quad (18)$$

where D is the set of data. X represents the question and passage including evidence snippets, and Y represents the answer.

4 Experiment

4.1 Dataset and Evaluation Metrics

For the MS-MARCO dataset, the questions are user queries issued to the Bing search engine and the context passages are from real web documents. The data has been split into a training set (82,326 pairs), a development set (10,047 pairs) and a test set (9,650 pairs). The answers are human-generated and not necessarily sub-spans of the passages so that the metrics in the official tool of MS-MARCO evaluation are BLEU (Papineni et al., 2002) and ROUGE-L (Lin, 2004). In the official evaluation tool, the ROUGE-L is calculated by averaging the score per question, however, the BLEU is normalized with all questions. We hold that the answer should be evaluated case-by-case in the reading comprehension task. Therefore, we mainly focus on the result in the ROUGE-L.

4.2 Implementation Details

4.2.1 Training

The evidence extraction and the answer synthesis are trained in two stages.

For evidence extraction, since the answers are not necessarily sub-spans of the passages, we choose the span with the highest ROUGE-L score with the reference answer as the gold span in the training. Moreover, we only use the data whose ROUGE-L score of chosen text span is higher than 0.7, therefore we only use 71,417 training pairs in our experiments.

For answer synthesis, the training data consists of two parts. First, for all passages in the training data, we choose the best span with highest ROUGE-L score as the evidence, and use the corresponding reference answer as the output. We only use the data whose ROUGE-L score of chosen evidence snippet is higher than 0.5. Second, we apply our evidence extraction model to all training data to obtain the extracted span. Then we treat the passage to which this span belongs as the input.

4.2.2 Parameter

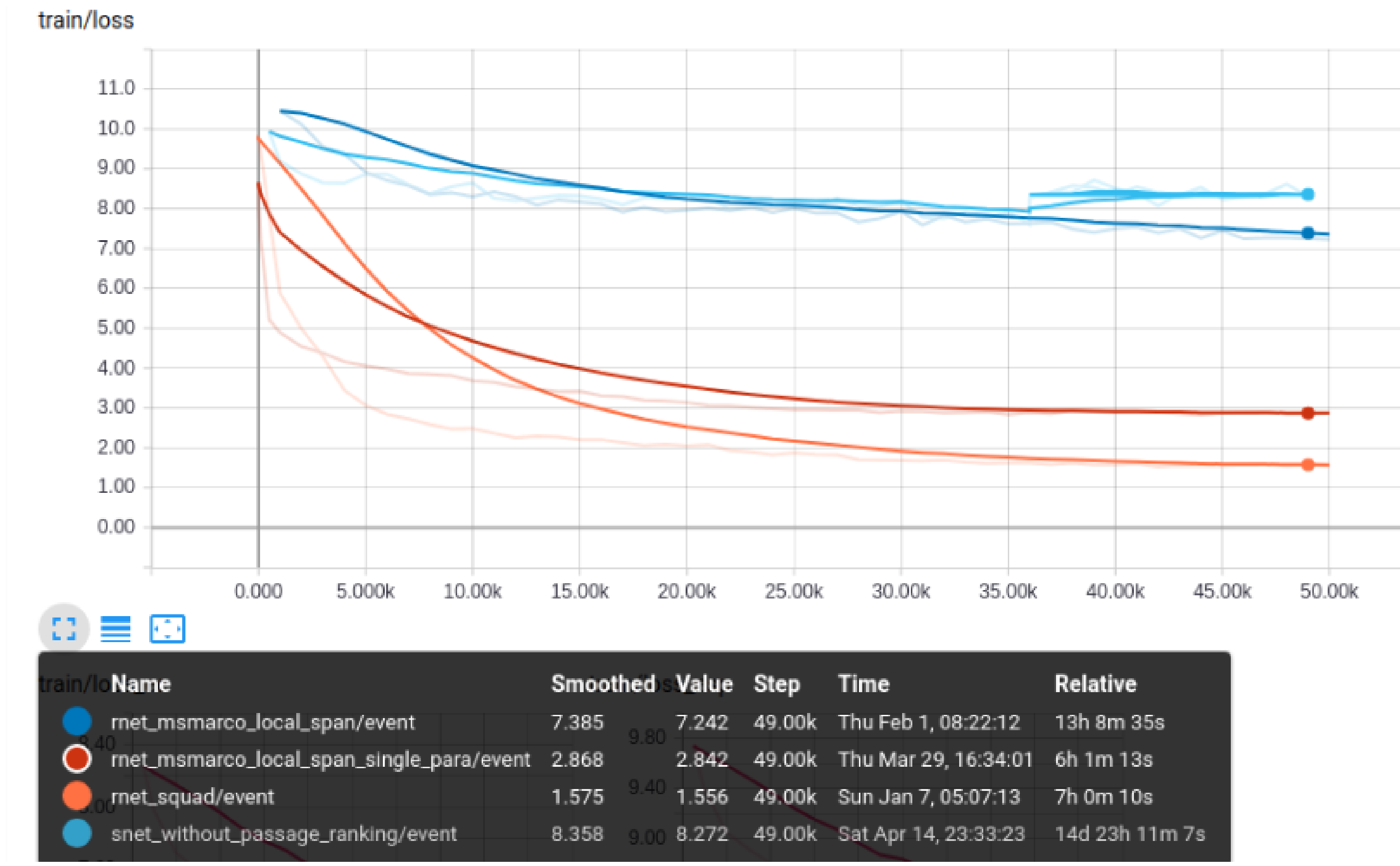
For answer extraction, we use 300-dimensional uncased pre-trained GloVe embeddings (Pennington et al., 2014)2 for both question and passage without update during training. We use zero vectors to represent all out-of-vocabulary words. Hidden vector length is set to 150 for all layers. We also apply dropout (Srivastava et al., 2014) between layers, with dropout rate 0.1. The weight r is set to 0.8.

For answer synthesis, we use an identical vocabulary set for the input and output collected from the training data. We set the vocabulary size to 30,000 according to the frequency and the other words are set to ¡unk¿. All word embeddings are updated during the training. We set the word embedding size to 300, set the feature embedding size of start and end positions of the extracted snippet to 50, and set all GRU hidden state sizes to 150.

The model is optimized using AdaDelta (Zeiler, 2012) with initial learning rate of 1.0. All hyperparameters are selected on the MS-MARCO development set.

4.3 BASELINE METHODS

We conduct experiments on the datasets with following settings:



	Name	Smoothed	Value	Step	Time	Relative
●	rnet_msmarco_local_span/event	7.385	7.242	49.00k	Thu Feb 1, 08:22:12	13h 8m 35s
●	rnet_msmarco_local_span_single_para/event	2.868	2.842	49.00k	Thu Mar 29, 16:34:01	6h 1m 13s
●	rnet_squad/event	1.575	1.556	49.00k	Sun Jan 7, 05:07:13	7h 0m 10s
●	snet_without_passage_ranking/event	8.358	8.272	49.00k	Sat Apr 14, 23:33:23	14d 23h 11m 7s

Fig. 4: Loss Convergence on different models

4.3.1 SQuAD

R-NET: The original rnet model taken from [?]

4.3.2 MS-MARCO

R-NET (on single para): The model is trained on the passages where the answer belongs to, from the MS-MARCO dataset.
R-NET (on multi para): The model is trained on the concatenation of all the passages provided for the particular query.
S-NET (Extraction w/o Passage Ranking): The model only has the evidence snippet prediction part.
S-NET (Extraction): The model which only has the evidence extraction part.
S-NET: The model that consists of the evidence extraction part and the answer synthesis part.

Method	**ROUGE-L paper / ours**
RNET (Single Para)	N.A/51.46
RNET (Multi Para)	42.89/27.93
S-NET (w/o Passage Ranking)	39.62/20.30
S-NET (Extraction)	42.23/19.87
S-NET	47.76/15.3

TABLE 1: Our performance on the MS-MARCO dev dataset

4.4 Result

Table 1 and Figure 4 shows the results on the MS-MARCO development data. The R-NET model on MS-MARCO performs modestly compared to the scores reported in the R-NET paper [3]. Our S-NET Extraction Model achieves only 19.87 in terms of ROUGE-L compared to what is reported by the authors, even after taking the exact hyperparameters as mentioned in [4]. On answer synthesis, it performs even worse on the outputs of the evidence extraction model compared to the slight improvement as reported in the S-NET paper. By observing the loss curve in Figure 4, it is quite evident that the models converges very earlier (blue curves) compared to the models trained on single-para datasets.

5 Conclusion

In this technical report, we present results on the experiments on R-NET and S-NET. The model achieves modest results on R-NET on MS-MARCO but the improvement in not reflected on the S-NET on MS-MARCO as it's supposed to. For future work, we will explore and identify the bottlenecks, one of which is figuring out appropriate hyperparameters for these models, due to which the results can improve.

Acknowledgments

The authors would like to thank the MS-MARCO organizers, R-NET and S-NET authors, HKUST-KnowComp [9] for the initial R-NET implementation and Google for the TensorFlow framework and the example codes for NMT [10] which were used to build the answer synthesis model.

References

[1] Konstantin Lopyrev Percy Liang Pranav Rajpurkar, Jian Zhang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250v3*, 2016.

[2] Xia Song Jianfeng Gao Saurabh Tiwary Rangan Majumder Li Deng Tri Nguyen, Mir Rosenberg. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268v2*, 2016.

[3] Furu Wei Baobao Chang Ming Zhou Wenhui Wang, Nan Yang. Gated self-matching networks for reading comprehension and question answering. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, 1(8):189–198, 2017.

[4] Nan Yang Bowen Du Weifeng Lv Ming Zhou Chuanqi Tan, Furu Wei. S-net: From answer extraction to answer generation for machine reading comprehension. *arXiv preprint arXiv:1706.04815v6*, 2017.

[5] Konstantin Lopyrev Percy Liang Pranav Rajpurkar, Jian Zhang. Shuohang wang and jing jiang. *arXiv preprint arXiv:1608.07905*, 2016b.

[6] Yoshua Bengio Dzmitry Bahdanau, Kyunghyun Cho. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473v7*, 2014.

[7] Christopher D. Manning Minh-Thang Luong, Hieu Pham. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025v5*, 2015.

[8] Caglar Gulcehre Dzmitry Bahdanau Fethi Bougares Holger Schwenk Yoshua Bengio Kyunghyun Cho, Bart van Merrienboer. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078v3*, 2014.

[9] R-net. https://github.com/HKUST-KnowComp/R-Net, 2016.

[10] Nmt. https://github.com/tensorflow/nmt, 2016.