MaP: A Matrix-based Prediction Approach to Improve Span Extraction in Machine Reading Comprehension

Huaishao Luo[1]*, **Yu Shi**[2], **Ming Gong**[3], **Linjun Shou**[3], **Tianrui Li**[1]
[1]School of Information Science and Technology, Southwest Jiaotong University
[2]Microsoft Cognitive Services Research Group
[3]Microsoft STCA NLP Group
huaishaoluo@gmail.com, trli@swjtu.edu.cn
{yushi,migon,lisho}@microsoft.com

Abstract

Span extraction is an essential problem in machine reading comprehension. Most of the existing algorithms predict the start and end positions of an answer span in the given corresponding context by generating two probability vectors. In this paper, we propose a novel approach that extends the probability vector to a probability matrix. Such a matrix can cover more start-end position pairs. Precisely, to each possible start index, the method always generates an end probability vector. Besides, we propose a sampling-based training strategy to address the computational cost and memory issue in the matrix training phase. We evaluate our method on SQuAD 1.1 and three other question answering benchmarks. Leveraging the most competitive models BERT and BiDAF as the backbone, our proposed approach can get consistent improvements in all datasets, demonstrating the effectiveness of the proposed method.

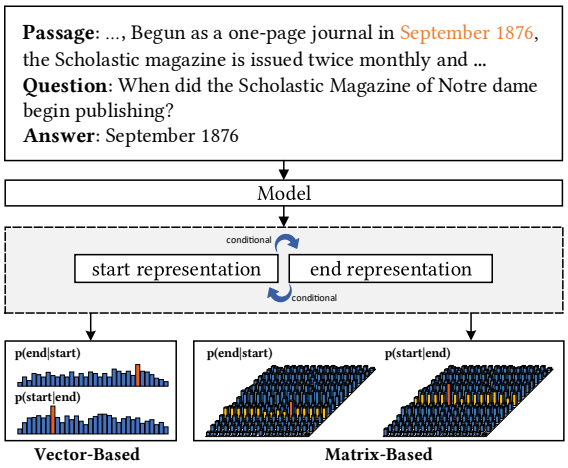

Figure 1: An illustration of a machine reading comprehension framework. Most of previous works are vector-based approaches shown as the left part. Our matrix-based conditional approach is shown in the right part. In our setting, every start (or end) position has an end (or start) probability vector, which leads that the output probabilities is a matrix (best seen in color).

1 Introduction

Machine reading comprehension (MRC), which requires the machine to answer comprehension questions based on the given passage of text, has been studied extensively in the past decades (Liu et al., 2019). Due to the increase of various large-scale datasets (e.g., SQuAD (Rajpurkar et al., 2016) and MS MARCO (Nguyen et al., 2016)), and the enhancement of pre-trained models (e.g., ELMo (Peters et al., 2018), BERT (Devlin et al., 2019), and XLNet (Yang et al., 2019)), remarkable advancements have been made recently in this area. Among various MRC tasks, span extraction is one of the essential tasks. Given the context and question, the span extraction task is to extract a span of the most plausible text from the corresponding context as a candidate answer. Although there exist unanswerable cases beyond the span extraction, the span-based task is still fundamental and significant in the MRC field.

Previous methods used to predict the start and end position of an answer span can be divided into two categories. The first one regards the generation of begin position and end position independently. We refer to this category as *independent approach*. It can be written as $p_* = p(*|\mathbf{H}^*)$, where $* \in \{s, e\}$, the s and e denote start and end, respectively. $\mathbf{H}^*$ is the hidden representation, in which $\mathbf{H}^s$ and $\mathbf{H}^e$ usually have shared features. The other one constructs a dependent route from the start position when predicting the end position. We refer to this category as *conditional approach*. It can be formalized as $p_s = p(s|\mathbf{H}^s), p_e = p(e|s, \mathbf{H}^e)$. This category usu-

*This work was done during the first author's internship at Microsoft

Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing, pages 687–695
December 4 - 7, 2020.

ally reuses the predicted position information (e.g., s) to assist in the subsequent prediction. The difference between these two approaches is that the conditional approach considers the relationship between start and end positions, but the independent approach does not. In the literature, AMANDA (Kundu and Ng, 2018b), QANet (Yu et al., 2018), and SEBert (Keskar et al., 2019) can be regarded as the independent approach, where the probabilities of the start and end positions are calculated separately with different representations. DCN (Xiong et al., 2017), R-NET (Wang et al., 2017), BiDAF[1] (Seo et al., 2017), Match-LSTM (Wang and Jiang, 2017), S-Net (Tan et al., 2018), SDNet (Zhu et al., 2018), and HAS-QA (Pang et al., 2019) belong to the conditional approach. The probabilities are generated in sequence.

The conditional approach empirically has an advantage over the independent approach. However, the output distributions of the previous conditional approaches are two probability vectors. It ignores some more possible start-end pairs. As an extension, every possible start (or end) position should have an end (or start) probability vector. Thus, the output conditional probabilities is a matrix.

We propose a **Ma**trix-based **P**rediction approach (MaP) based on the above consideration in this paper. As Figure 1 shown, the key point is to consider as many probabilities as possible in *training* and *inference* phases. Specifically, we calculate a conditional probability matrix instead of a probability vector to expand the choices of start-end pairs. Because of more values contained in a matrix than a vector, there is a big challenge in the training phase of the MaP. That is the high computational cost and memory issues if the input sequence is long. As an instance, the matrix contains $262,144$ probability values if the sequence length is 512. Therefore, we propose a sampling-based training strategy to speed up the training and reduce the memory cost.

The main contributions of our work are fourfold.

- A novel conditional approach is proposed to address the limitation of the probability vector generated by the vector-based conditional approach. It increases the likelihood of hitting the ground-truth start and end positions.
- A sampling-based training strategy is proposed to overcome the computation and memory issues in the training phase of the matrix-based conditional approach.
- An ensemble approach on both start-to-end and end-to-start directions of conditional probability is investigated to improve the accuracy of the answer span.
- We evaluate our strategy on SQuAD 1.1 and three other question answering benchmarks. The implementation of the matrix-based conditional approach is designed based on the BERT and BiDAF, which are the most competitive models, to test the generalization of our strategy. The consistent improvements in all datasets demonstrate the effectiveness of the strategy.

[1]We classify BiDAF as a conditional approach by its official implementation: https://github.com/allenai/bi-att-flow

2 Methodology

In this section, we first give the problem definition. Then we introduce a typical vector-based conditional approach. Next, we mainly introduce our matrix-based conditional approach and sampling-based training strategy. Finally, an ensemble approach on both start-to-end and end-to-start directions of conditional probability is discussed.

2.1 Problem Statement

Given the passage $P = \{t_1, t_2, \cdots, t_n\}$ and the question $Q = \{q_1, q_2, \cdots, q_m\}$, the span extraction task needs to extract the continuous subsequence $A = \{t_s, \cdots, t_e\}$ $(1 \leq s \leq e \leq n)$ from the passage as the right answer to the question, where n and m are the length of the passage and question respectively, s and e are the start and end position in the passage. Usually, the objective to predict $a = (s, e)$ is maximizing the conditional probability $p(a|P, Q)$.

2.2 A Typical Vector-based Approach

We summarize a typical implementation of the vector-based conditional approach shown in Figure 2. Previous mentioned R-NET, BiDAF, Match-LSTM, S-Net, and SDNet can be regarded as such implementation. Its backbone is the Pointer Network proposed by Vinyals et al. (2015). The interactive representation $\mathbf{H} \in \mathbb{R}^{n \times d}$ between the given question Q and passage P is calculated as follows,

$$\mathbf{H} = \mathfrak{M}(Q, P), \tag{1}$$

where $\mathfrak{M}$ is a neural network, e.g., Match-LSTM, QANet, BERT, and XLNet, d is the dimension size

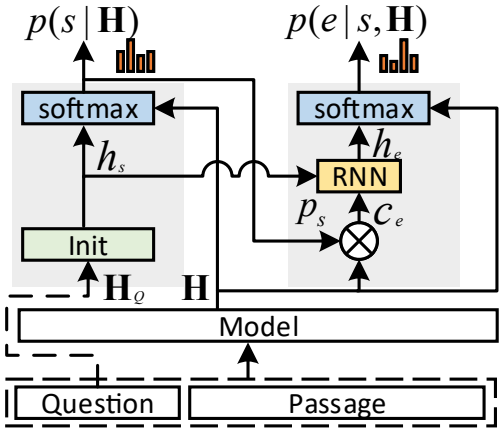

Figure 2: A typical implementation of the vector-based conditional approach.

of the representation. After generating the interactive representation, the next step is to predict the answer span.

The main architecture of the span prediction is an RNN. As an instance, LSTM is used in (Wang and Jiang, 2017), and GRU is adopted in (Tan et al., 2018; Zhu et al., 2018). Take the hidden representation $h_e \in \mathbb{R}^k$ of end position as an example, which is calculated as follows,

$$h_e = \mathrm{RNN}(h_s, c_e), \tag{2}$$

$$c_e = \mathbf{H}^\top p_s, \tag{3}$$

where $p_s = p(s|\mathbf{H})$ is the start probability and $p_s \in \mathbb{R}^n$, k is the dimension size of h_e. Then $p_e = p(e|s,\mathbf{H})(p_e \in \mathbb{R}^n)$ can be calculated using h_e as follows,

$$p(e|s,\mathbf{H}) = \mathrm{softmax}\Big(\mathbf{v}^\top \tanh\big(\mathbf{V}\mathbf{H}^\top + [\![\mathbf{W}_e h_e]\!]^n\big)\Big) \tag{4}$$

where $[\![\cdot]\!]^n$ is an operation that generates a matrix by repeating the vector on the left n times, $\mathbf{v} \in \mathbb{R}^l$, $\mathbf{V} \in \mathbb{R}^{l\times d}$, and $\mathbf{W}_e \in \mathbb{R}^{l\times k}$ are parameters to be learned.

The calculation of $p(s|\mathbf{H})$ is similar to $p(e|s,\mathbf{H})$. The key is to obtain the hidden state h_s. A choice is to use an attention approach to condense the question representation into a vector. The process is as follows,

$$p_{init} = \mathrm{softmax}\Big(\mathbf{v}_Q^\top \tanh\big(\mathbf{V}_Q \mathbf{H}_Q^\top\big)\Big), \tag{5}$$

$$h_s = \mathbf{H}_Q^\top p_{init}, \tag{6}$$

where $\mathbf{H}_Q \in \mathbb{R}^{m\times d}$ is the representation corresponding to Q, $\mathbf{v}_Q \in \mathbb{R}^l$, and $\mathbf{V}_Q \in \mathbb{R}^{l\times d}$ are parameters.

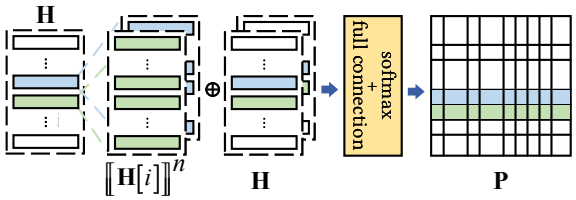

Figure 3: Matrix-based conditional approach.

There is a vast number of works on MRC. However, most of these works focus on the design of $\mathfrak{M}$ and generate the answer span based on the vector-based conditional approach. In this paper, we expand the vector to a probability matrix. Thus, many more possibilities can be covered. It is also a natural manner because that every start (or end) position should have an end (or start) probability vector.

2.3 Matrix-based Conditional approach

As the previous description, the implementation of the vector-based conditional approach has a unified and important implementation step: create a 'condition'. Take the forward direction ('condition' constructed from the start position to end position) of the vector-based conditional approach as an example, the 'condition' is the probability vector p_s. The end probability vector p_e can not be calculated until generating p_s. However, there is only one probability vector p_e whatever the start position is. In this paper, we keep the 'condition' step but propose calculating an individual p_e for each start position. Specifically, the probability matrix $\mathbf{P}_e \in \mathbb{R}^{n\times n}$ is calculated as follows,

$$\mathbf{P}_e^{(i)} = \mathrm{softmax}\left(\mathbf{v}^\top \tanh\left(\mathbf{V}\left[\mathbf{H}^\top; \left[\!\!\left[\left(\mathbf{H}[i]\right)^\top\right]\!\!\right]^n\right]\right)\right) \tag{7}$$

where $\mathbf{P}_e^{(i)}$ denotes the i-th row of $\mathbf{P}_e$, $[;]$ is a concatenate operation, $[\![\cdot]\!]^n$ is an operation that generates a matrix by repeating the vector on the left n times, $[i]$ means to choose the i-th row from the matrix $\mathbf{H}$, $\mathbf{v} \in \mathbb{R}^l$ and $\mathbf{V} \in \mathbb{R}^{l\times 2d}$ are parameters. Figure 3 illustrates the calculation process of Eq. (7).

Although the calculation is brief and can cover more probabilities than the vector-based approach, there is a big question on computation cost and memory occupation. The main computation cost comes from the matrix multiplication between $\mathbf{V}$ and $\left[\mathbf{H}^\top; \left[\!\!\left[\left(\mathbf{H}[i]\right)^\top\right]\!\!\right]^n\right]$ in Eq. (7), totally n times

such computation for $\mathbf{P}_e$. The number of probabilities is also n times bigger than the vector-based conditional approach. It also causes the issue of out of memory (OOM), especially with a big n, due to intermediate gradient values needing cache in the training phase. We propose a sampling-based training strategy to solve the above issues.

2.4 Sampling-based Training Strategy

In order to train the probability matrix effectively, we propose a sampling-based strategy in the training phase. Given the hyper-parameter k, we first choose the indexes $\hat{\mathcal{I}}$ of top $k-1$ possibilities from $p_s^{(\text{-}\hat{s})}$,

$$\hat{\mathcal{I}} = \text{top}\left(p_s^{(\text{-}\hat{s})}, k-1\right), \tag{8}$$

where $\text{top}(p, v)$ is an operation used to get the indexes of top v values in p, $p^{(\text{-}w)}$ contains all but w-th value of p, and $\hat{s}$ is the truth start position used as the supervised information in the training phase. Then, the $\hat{s}$ must merge to $\hat{\mathcal{I}}$,

$$\mathcal{I} = \hat{\mathcal{I}} + \{\hat{s}\}, \tag{9}$$

where $\mathcal{I}$ contains k indexes.

Eq. (8) and Eq. (9) promise that the sampled start probabilities must contain and only contain the target probability which we need to train in each iteration. The target probability is the $\hat{s}$-th value in p_s, and the bigger, the better.

After sampling the start probability vector, the computation cost of $\mathbf{P}_e$ decrease. For each $i \in \mathcal{I}$, executing Eq. (7) repeatedly can generate a sampling-based end probability matrix. It is noted that this sampling-based matrix is a part of the original $\mathbf{P}_e$. We refer to it as $\tilde{\mathbf{P}}_e$, and $\tilde{\mathbf{P}}_e \in \mathbb{R}^{k \times n}$. It is still a big issue of computation cost and memory occupation for $\tilde{\mathbf{P}}_e$ with a long sequence. So, we carry out similar operations in Eq. (8) and Eq. (9) for each row of $\tilde{\mathbf{P}}_e$ using $\hat{e}$ instead of $\hat{s}$, where $\hat{e}$ is the end truth position. Finally, the sampling-based matrix $\hat{\mathbf{P}}_e \in \mathbb{R}^{k \times k}$ is generated. It is small enough to train compared with $\mathbf{P}_e$. Figure 4 shows the sampling results colored with a yellow background on the left and corresponding ground truth matrix on the right.

2.5 Training

In the training phase, the objective function is to minimize the cross-entropy error averaged over start and end positions,

$$\mathcal{L} = \frac{1}{2}(\mathcal{L}_s + \mathcal{L}_e), \tag{10}$$

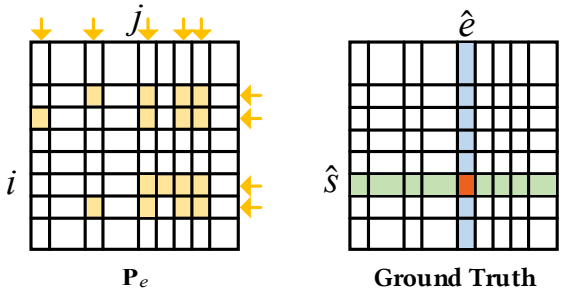

Figure 4: A sampling of probability matrix. Left: the calculated probability matrix with sampled top four positions (in both row and column directions colored with yellow background). Right: the ground truth matrix, where position $(\hat{s}, \hat{e})$ with the red background has probability 1.

$$\mathcal{L}_s = -\frac{1}{N}\sum_{i=1}^{N}\left(\mathbb{I}(\hat{s})\left(\log(p_s)\right)^\top\right), \tag{11}$$

$$\mathcal{L}_e = -\frac{1}{N}\sum_{i=1}^{N}\left(\mathcal{T}\left(\mathbb{I}(\hat{s},\hat{e})\right)\left(\log\left(\mathcal{T}(\hat{\mathbf{P}}_e)\right)\right)^\top\right), \tag{12}$$

where N is the number of data, $\mathbb{I}(\hat{s})$ means the one-hot vector of $\hat{s}$, $\mathbb{I}(\hat{s},\hat{e})$ means a zero matrix with a value of 1 in row $\hat{s}$ and column $\hat{e}$, and $\mathcal{T}()$ is a row wise flatten operation. The flatten operation makes the loss function on matrix-based distribution similar to that on vector-based distribution.

As the introduction of the sampling-based training strategy, there are limited end probabilities that could be trained in each iteration. The extreme situation is k equals to n, which makes all probability matrix calculate each time. As our previous argumentation, it is almost impossible for time and memory limitations. However, there is a question of what makes sampling strategy works. The following content gives some explanation based on gradient backpropagation.

The gradient of the cross-entropy $\mathcal{L}_*$ to the predicted logits z_* is,

$$\frac{\partial \mathcal{L}_*}{\partial z_*} = \begin{cases} p_*^{(i)} - 1, & \text{if } i \text{ is the ground-truth;} \\ p_*^{(j)}, & \textit{others} \end{cases} \tag{13}$$

where $p_* = \text{softmax}(z_*)$ is probabilities in which values are between 0 and 1 (exclusion). Thus $p_*^{(i)} - 1$ is negative, and $p_*^{(j)}$ is positive in most cases. As the parameters θ update usually follows $\theta_t = \theta_{t-1} - \eta \cdot \nabla_\theta \mathcal{L}(\theta)$ and learning rate η is a positive value, the probability in ground-truth position should go up, and the probabilities in other sampled positions should go down.

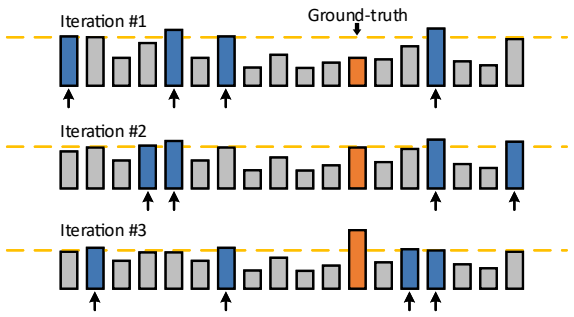

Figure 5: Sampling-based probabilities training ($k = 5$). Block with red color is the ground-truth, blocks with blue color are the sampled probabilities. Probabilities with a gray background will not change their values in each iteration.

Figure 5 illustrates the sampling-based training process, where the parameter k is set to 5. It means that there are extra top-4 probabilities (blue background) except ground-truth (red background) will be chosen to calculate. With the iteration going from #1 to #3, the probability in ground-truth position goes up, and that in sampled top-4 positions goes down. Such a sampling-based training approach has the same goal with the training on the whole probabilities, thus should have proximity results.

2.6 Ensemble for Inference

The vector-based conditional approach usually searches the span (s,e) via the computation of $p_s^{(i)} \times p_e^{(j)}$ under the condition of $i \leq j$, and choices the (i^*, j^*) with the highest $p_s^{(i^*)} \times p_e^{(j^*)}$ as the output in the inference phase. The matrix-based conditional approach follows the same idea, but the calculation of the probability is $p_s^{(i)} \times \mathbf{P}_e^{(i,j)}$ instead of $p_s^{(i)} \times p_e^{(j)}$. The $p_s^{(i)}$ is the i-th probability in p_s, and $\mathbf{P}_e^{(i,j)}$ is the probability in row i, column j of $\mathbf{P}_e$.

The above inference strategy only involves one direction, e.g., start-to-end direction (generate start position firstly, then generate end position), which is the most cases in previous works. An ensemble of both start-to-end and end-to-start directions is a good choice to improve the performance. The difference in end-to-start direction is that Eqs. (7-12) should be repeated in the opposite direction. In other words, the start is replaced by e, and the end is replaced by s. Totally, there are two groups of probabilities, $(p_s, \mathbf{P}_e)$ and $(p_e, \mathbf{P}_s)$. In this paper, we design a type of ensemble strategy, which first chooses top k pairs $\mathbf{F} = \{(i_f, j_f)\}$ with highest probability $p_s^{(i_f)} \times \mathbf{P}_e^{(i_f,j_f)}$, then chooses top k pairs $\mathbf{B} = \{(j_b, i_b)\}$ with highest probability $p_e^{(j_b)} \times \mathbf{P}_s^{(j_b,i_b)}$. It is noted that some pairs may have the same position, e.g., $(3_f, 5_f)$ and $(5_b, 3_b)$. If there are the same elements, we prune away them in $\mathbf{B}$. Then, we choose the (i^*, j^*) with highest probability in $\mathbf{F} \cup \mathbf{B}$.

The overall training procedure of MaP is summarized in Algorithm 1.

Algorithm 1 MaP Training Algorithm

Input: N pairs of passage P and question Q, k used to choose top probabilities;
Output: Learned MaP model

```
Initialize all learnable parameters Θ;
repeat
    Select a batch of pairs from corpus;
    for each pair (P, Q) do
        Use a neural network 𝔐 to generate the representation H;  (Eq. 1)
        Compute start probability vector p_s;  (Eqs. 4-6)
        Sample indexes 𝓘 by choosing top k − 1 probabilities of p_s;  (Eqs. 8,9)
        Compute end probability matrix P_e;  (Eq. 7)
        Compute objective 𝓛;  (Eq. 10-12)
    end for
    Use the backpropagation algorithm to update parameters Θ by minimizing the objective with the batch update mode
until stopping criteria is met
```

3 Experiments

In this section, we conduct experiments to evaluate the effectiveness of the proposed MaP.

3.1 Datasets

We first evaluate our strategy on SQuAD 1.1, which is a reading comprehension benchmark. The benchmark benefits to our evaluation compared with its augmented version SQuAD 2.0 due to its questions always have a corresponding answer in the given passages. We also evaluate our strategy on three other datasets from the MRQA 2019 Shared Task[2]: NewsQA (Trischler et al., 2017), HotpotQA (Yang et al., 2018), Natural Questions (Kwiatkowski et al., 2019). As the SQuAD 1.1 dataset, the format of

[2] https://github.com/mrqa/MRQA-Shared-Task-2019

Models		SQuAD		NewsQA		HotpotQA		Natural Questions	
		EM	F1	EM	F1	EM	F1	EM	F1
BERT-Base	InD	81.24	88.38	52.59	67.12	59.01	75.69	67.31	78.96
	MaP$_F$	81.78	88.59	52.66	66.50	59.82	75.81	67.68	78.99
	MaP$_E$	**82.12**	**88.63**	**53.06**	**67.37**	**60.55**	**76.12**	**68.21**	**79.09**
BERT-Large	InD	84.05	90.85	54.46	69.61	62.26	78.18	69.44	80.93
	MaP$_F$	84.50	90.89	54.84	68.73	63.19	78.99	69.56	80.49
	MaP$_E$	**84.79**	**90.89**	**55.29**	**69.98**	**63.70**	**79.25**	**69.91**	**81.22**
BiDAF	VCP	68.57	78.23	44.04	58.07	47.31	62.42	56.95	68.79
	MaP$_F$	68.85	78.06	44.19	58.65	50.25	65.21	57.04	68.87
	MaP$_E$	**69.55**	**78.91**	**44.25**	**58.91**	**51.45**	**66.74**	**57.21**	**69.08**

Table 1: The performance (%) of EM and F1 on SQuAD 1.1 and three MRQA extractive question answering tasks. MaP$_F$ is the matrix-based conditional approach calculating on start-to-end direction. MaP$_E$ means the ensemble of both directions of matrix-based conditional approach. InD denotes the independent approach. VCP is vector-based conditional approach.

the task is extractive question answering. It contains no unanswerable or non-span answer questions. Besides, the fact that these datasets vary in both domain and collection pattern benefits for the evaluation of our strategy on generalization across different data distributions. Table 2 shows the statistics of these datasets.

Dataset	Training	Development
SQuAD 1.1	86,588	10,507
NewsQA	74,160	4,212
HotpotQA	72,928	5,904
Natural Questions	104,071	12,836

Table 2: The statistics of datasets.

3.2 Baselines

To validate the effectiveness and generalization of our proposed strategy on the span extraction, we implement it using two strong backbones, BERT and BiDAF. Specifically, we borrow their main bodies except the top layer to implement the proposed strategy to finish the span extraction on different datasets. Some more tests on other models, e.g., XLNet (Yang et al., 2019) and SpanBERT (Joshi et al., 2019), and datasets will be our future work.

- **BERT**: BERT is an empirically powerful language model, which obtained state-of-the-art results on eleven natural language processing tasks in the past (Devlin et al., 2019). The original implementation in their paper on the span prediction task belongs to the independent approach. Both BERT-base and BERT-large with uncased pre-trained weights are used in comparison to investigating the effect of the ability of language model on span extraction with different prediction approaches.

- **BiDAF**: BiDAF is used as a baseline of the vector-based conditional approach (Seo et al., 2017). The use of a multi-stage hierarchical process and a bidirectional attention flow mechanism makes its representation powerful.

There are four strategies of span extraction involved in our comparison: **InD** denotes the independent approach; **VCP** is the vector-based conditional approach; **MaP**$_F$ is our matrix-based conditional approach calculating on start-to-end direction; **MaP**$_E$ means the ensemble of both directions of matrix-based conditional approach. The InD is used to compare with MaP$_F$ and MaP$_E$ in BERT, and the VCP is used to compare with MaP$_F$ and MaP$_E$ in BiDAF.

3.3 Experimental Settings

We implement the BERT and BiDAF following the official settings for a fair comparison. For the BERT, we train for 3 epochs with a learning rate of 5e-5 and a batch size of 32. The max sequence length is 384 for SQuAD 1.1 and 512 for other datasets, and a sliding window of size 128 is used for all datasets is the sentence is longer than the max length. For the BiDAF, we keep all original settings except a difference that we use ADAM (Kingma and Ba, 2015) optimizer with a learning rate of 1e-3 in the training phase instead of AdaDelta (Zeiler, 2012) for a stable performance.

Following the work from (Rajpurkar et al., 2016), we evaluate the results using Exact Match (EM) and Macro-averaged F1 score. The sampling parameter k is set to 20 for our strategy. We implement our model in python using the pytorch-transformers library[3] for BERT and the AllenNLP library[4] for BiDAF. The reported results are average scores of 5 runs with different random seeds. All computations are done on 4 NVIDIA Tesla V100 GPUs.

3.4 Main Results

The results of our strategies as well as the baselines are shown in Table 1. All these values come from the evaluation of the development sets in each dataset due to the test sets are withheld. Nevertheless, our strategy achieves a consistent improvement compared with the independent approach and the vector-based conditional approach. The values with a bold type mean the winner across all strategies. As we can observe, the MaP$_E$ wins 16 out of 16 in both BERT-base and BERT-large groups. It proves that the ensemble of both directions is helpful for the span extraction. In the BiDAF group, The MaP$_E$ is also the best on all datasets compared with VCP. It shows the robustness of our matrix-based conditional approach in language models. The fact that the MaP$_F$ wins 12 out of 12 in EM, and 8 out of 12 in F1 demonstrates that the matrix-based conditional approach is capable of predicting a clean answer span that matches human judgment exactly. We suppose the reason is that more start-end position pairs considered in the probability matrix can enhance the interaction and constraint between the start and end, thus, make the MaP$_F$ perform more consistently in EM than in F1.

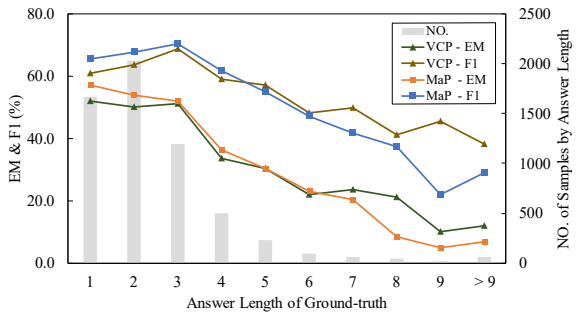

Figure 6: EM and F1 of MaP$_F$ and VCP based on BiDAF under different answer length.

[3] https://github.com/huggingface/pytorch-transformers

[4] https://github.com/allenai/allennlp

3.5 Strategy Analysis

Figure 6 shows how the performance changes with respect to the answer length, which is designed on HotpotQA. We can see that the matrix-based conditional approach works better than the vector-based conditional approach as the span decrease in length. Since the short answers have a high rate in all answer spans, so the matrix-based conditional approach is better for the answer span task. In other words, this observation supports the ensemble of both directions as $_E$ does. The MaP$_E$ combining the MaP$_F$'s advantage in short answers and the VCP's advantage in long answers can get a better result than any of them.

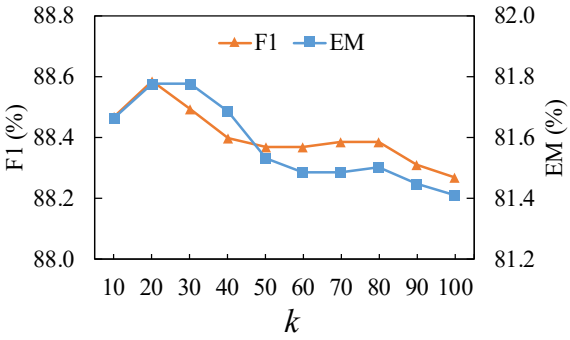

Figure 7: Impact of hyper-parameter k in MaP$_F$ on SQuAD 1.1 with BERT-base as the backbone.

We investigate the impact of k used to choose the top probabilities in the training phase. The results are shown in Figure 7. With the increase of k, the EM and F1 show a downtrend. The best performance happens at $k = 20$. We guess that choosing more probabilities makes the training difficult and brings extra noises to the candidate positions. E.g., if k is set to 30, the number of candidate probabilities will be 900, which is larger than the sequence length 512 in vector-based conditional approach.

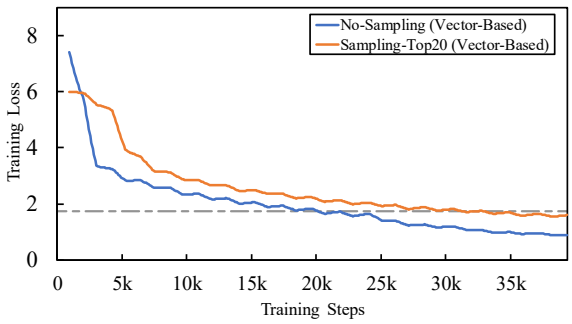

Figure 8: Convergence of sampling-based training strategy on BERT.

We analyze the convergence of the sampling-

based training strategy on SQuAD 1.1. Due to the effectiveness of the sampling-based training strategy is proved in MaP, we conduct an further experiment under the VCP to prove its generalization. Figure 8 demonstrates the results. As our expectation, the sampling-based training strategy optimizes the model as training in whole samples. However, it will cost longer training steps to get the same loss compared with standard training. So our sampling-based training strategy is good for the training of the matrix-based conditional approach.

4 Related Work

Machine reading comprehension is an important topic in the NLP community. More and more neural network models are proposed to tackle this problem, including DCN (Xiong et al., 2017), R-NET (Wang et al., 2017), BiDAF (Seo et al., 2017), Match-LSTM (Wang and Jiang, 2017), S-Net (Tan et al., 2018), SDNet (Zhu et al., 2018), QANet (Yu et al., 2018), HAS-QA (Pang et al., 2019). Among various MRC tasks, span extraction is a typical task that extracting a span of text from the corresponding passage as the answer of a given question. It can well overcome the weakness that words or entities are not sufficient to answer questions (Liu et al., 2019).

Previous models proposed for span extraction mostly focus on the design of architecture, especially on the representation of question and passage, and the interaction between them. There are few works devoted to the top-level design of span output, which refers to the probabilities generation from the representation. We divide the previous top-level design into two categories, independent approach and conditional approach. The independent approach is to predict the start and end positions in the given passage independently (Kundu and Ng, 2018a; Yu et al., 2018). Although the independent approach has a simple assumption, it works well when the input features are strong enough, e.g., combining with BERT (Devlin et al., 2019), XL-Net (Yang and Song, 2019), and SpanBERT (Joshi et al., 2019). Nevertheless, since there is a kind of dependency relationship between start and end positions, the conditional approach has advancements over the independent approach.

A typical work on the conditional approach comes from Wang and Jiang (2017). They proposed two different models based on the Pointer Network. One is the sequence model which produces a sequence of answer tokens as the final output, and another is the boundary model which produces only the start token and the end token of the answer. The experimental results demonstrate that the boundary model (span extraction) is superior to the sequence model on both EM and F1. The R-NET (Wang et al., 2017), BiDAF (Seo et al., 2017), S-Net (Tan et al., 2018), SDNet (Zhu et al., 2018) have the same output layer and inference phase with the boundary model in (Wang and Jiang, 2017). Lee et al. (2016) presented an architecture that builds fixed length representations of all spans in the passage with a recurrent network to address the answer extraction task. The computation cost is decided by the max-length of the possible span and the sequence length. The experimental results show an improvement on EM compared with the endpoints prediction that independently predicts the two endpoints of the answer span.

However, previous works related to the conditional approach are always based on a probability vector. We investigate another possible matrix-based conditional approach in this paper. Besides, a well-matched training strategy is proposed to our approach, and forward and backward conditional possibilities are also integrated to improve the performance.

5 Conclusion

In this paper, we first investigate different approaches of span extraction in MRC. To improve the current vector-based conditional approach, we propose a matrix-based conditional approach. More careful consideration of the dependencies between the start and end positions of the answer span can predict their values better. We also propose a sampling-based training strategy to address the training process of the matrix-based conditional approach. The final experimental results on a wide of datasets demonstrate the effectiveness of our approach and training strategy.

Acknowledgments

This work was supported by National Key R&D Program of China (2019YFB2101802) and Sichuan Key R&D project (2020YFG0035).

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of

deep bidirectional transformers for language understanding. In *NAACL-HLT*, pages 4171–4186.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2019. Spanbert: Improving pre-training by representing and predicting spans. *arXiv:1907.10529*.

Nitish Shirish Keskar, Bryan McCann, Caiming Xiong, and Richard Socher. 2019. Unifying question answering and text classification via span extraction. *arXiv:1904.09286*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Souvik Kundu and Hwee Tou Ng. 2018a. A nil-aware answer extraction framework for question answering. In *EMNLP*, pages 4243–4252.

Souvik Kundu and Hwee Tou Ng. 2018b. A question-focused multi-factor attention network for question answering. In *AAAI*, pages 5828–5835.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *TACL*, 7:453–466.

Kenton Lee, Shimi Salant, Tom Kwiatkowski, Ankur Parikh, Dipanjan Das, and Jonathan Berant. 2016. Learning recurrent span representations for extractive question answering. *arXiv:1611.01436*.

Shanshan Liu, Xin Zhang, Sheng Zhang, Hui Wang, and Weiming Zhang. 2019. Neural machine reading comprehension: Methods and trends. *arXiv:1907.01118*.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. In *Proceedings of the Workshop on Cognitive Computation: Integrating neural and symbolic approaches 2016 colocated with the 30th Annual Conference on Neural Information Processing Systems (NIPS 2016)*, volume 1773.

Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Lixin Su, and Xueqi Cheng. 2019. HAS-QA: hierarchical answer spans model for open-domain question answering. In *AAAI*, pages 6875–6882.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL-HLT*, pages 2227–2237.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In *EMNLP*, pages 2383–2392.

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *ICLR*.

Chuanqi Tan, Furu Wei, Nan Yang, Weifeng Lv, and Ming Zhou. 2018. S-net: From answer extraction to answer generation for machine reading comprehension. In *AAAI*.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2017. Newsqa: A machine comprehension dataset. In *Rep4NLP@ACL*, pages 191–200.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems*, pages 2692–2700.

Shuohang Wang and Jing Jiang. 2017. Machine comprehension using match-lstm and answer pointer. In *ICLR 2017: International Conference on Learning Representations, Toulon, France, April 24-26: Proceedings*, pages 1–15.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *ACL*, pages 189–198.

Caiming Xiong, Victor Zhong, and Richard Socher. 2017. Dynamic coattention networks for question answering. In *ICLR*.

Liu Yang and Lijing Song. 2019. Contextual aware joint probability model towards question answering system. *arXiv:1904.08109*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv:1906.08237*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *EMNLP*, pages 2369–2380.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V. Le. 2018. Qanet: Combining local convolution with global self-attention for reading comprehension. In *ICLR*.

Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *arXiv:1212.5701*.

Chenguang Zhu, Michael Zeng, and Xuedong Huang. 2018. Sdnet: Contextualized attention-based deep network for conversational question answering. *arXiv:1812.03593*.