mathematics

MDPI

Article

Improving Machine Reading Comprehension with Multi-Task Learning and Self-Training

Jianquan Ouyang * and Mengen Fu

College of Computer Cyberspace Security, Xiangtan University, Xiangtan 411105, China; ccdf.mengen@gmail.com
* Correspondence: oyjq@xtu.edu.cn

Abstract: Machine Reading Comprehension (MRC) is an AI challenge that requires machines to determine the correct answer to a question based on a given passage, in which extractive MRC requires extracting an answer span to a question from a given passage, such as the task of span extraction. In contrast, non-extractive MRC infers answers from the content of reference passages, including Yes/No question answering to unanswerable questions. Due to the specificity of the two types of MRC tasks, researchers usually work on one type of task separately, but real-life application situations often require models that can handle many different types of tasks in parallel. Therefore, to meet the comprehensive requirements in such application situations, we construct a multi-task fusion training reading comprehension model based on the BERT pre-training model. The model uses the BERT pre-training model to obtain contextual representations, which is then shared by three downstream sub-modules for span extraction, Yes/No question answering, and unanswerable questions, next we fuse the outputs of the three sub-modules into a new span extraction output and use the fused cross-entropy loss function for global training. In the training phase, since our model requires a large amount of labeled training data, which is often expensive to obtain or unavailable in many tasks, we additionally use self-training to generate pseudo-labeled training data to train our model to improve its accuracy and generalization performance. We evaluated the SQuAD2.0 and CAIL2019 datasets. The experiments show that our model can efficiently handle different tasks. We achieved 83.2EM and 86.7F1 scores on the SQuAD2.0 dataset and 73.0EM and 85.3F1 scores on the CAIL2019 dataset.

Keywords: machine reading comprehension; Natural Language Processing; multi-task learning; Self Training; pre-trained model

check for updates

Citation: Ouyang, J.; Fu, M. Improving Machine Reading Comprehension with Multi-Task Learning and Self-Training. *Mathematics* **2022**, *10*, 310. https://doi.org/10.3390/math10030310

Academic Editors: Florentina Hristea and Cornelia Caragea

Received: 28 November 2021
Accepted: 16 January 2022
Published: 19 January 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Machine reading comprehension (MRC) aims to teach machines to answer questions after understanding a given passage [1,2], which can be broadly classified into two categories: Extractive MRC and Non-extractive MRC. Extractive MRC requires models to extract the answer span of a question from a reference text. For example, the tasks of close-test [3] and span extraction [4,5]. Figure 1 shows a span extraction sample in the SQuAD2.0 dataset [6]. In contrast, Non-extractive MRC infers answers to questions from the content of the referenced passage, including Yes/No question answering [7] and unanswerable question task [6]. Figure 2 shows an unanswerable question sample in the SQuAD2.0 dataset.

Due to the different task objectives, researchers usually research one type of task. However, realistic reading comprehension application scenarios usually require models that can handle many different types of tasks simultaneously. For example, the Chinese legal text dataset CAIL2019 reading comprehension dataset [8] consists of more than 39,000 sample data, which involve the content of civil and criminal first instance judgments. The dataset has three tasks: span extraction, yes/no questions answered, and unanswerable questions, where the span extraction task is the main task, accounting for 83% of total data.

Yes/no questions answering and unanswerable question task accounts for 14% and 3%, respectively. This requires multi-task learning.

Multi-task learning is a field of machine learning in which multiple tasks are learned in parallel while using a shared representation [9–11]. Compared with learning multiple tasks individually, this joint learning effectively increases the sample size for training the model, thus leading to performance improvement by increasing the generalization of the model [12]. To solve multi-tasking MRC is important, and some straightforward solutions have been proposed. Liu et al. [13] appended an empty word token to the context and added a simple classification layer for the MRC model. Hu et al. [14] used two auxiliary losses, independent span loss to predict plausible answers and independent no answer loss to determine the answerability of the question. Further, an extra verifier is used to determine whether the predicted answer is contained by the input snippets. Back et al. [15] developed an attention-based satisfaction score to compare question embeddings with candidate answer embeddings. Zhang et al. [16] proposed a verifier layer, which is a linear layer applied to context embeddings weighted by the start and end distributions over contextual word representations concatenated to [CLS] token representation for BERT. The above studies are based on the SQuAD2.0 dataset, but the SQuAD2.0 dataset only includes two different tasks. We want the model to be able to handle more tasks simultaneously, such as the CAIL2019 reading comprehension dataset, which contains three different tasks. Researchers usually set up three types of auxiliary losses to jointly train the model. We think that too much auxiliary loss may hurt the general model training, so we propose to fuse three different task outputs into a new span extraction output with only one loss function for global training.

Early neural reading models typically used various attention mechanisms to build interdependent representations of passages and questions, then predict answer boundaries in turn. A wide range of attention models have been employed, including Attention Sum Reader [17], Gated attention Reader [18], Self-matching Network [19], Attention over Attention Reader [20], and Bi-attention Network [21]. Recently, PrLMs have dominated the design of encoders for MRC with great success. These PrLMs include ELMo [22], GPT [23], BERT [24], XLNet [25], Roberta [26], ALBERT [27], and ELECTRA [28]. They bring impressive performance improvements for a wide range of NLP tasks for two main reasons: (1) language models are pre-trained on a large-scale text corpus, which allows the models to learn generic language features and serve as a knowledge base; (2) thanks to the Transformer architecture, language models enjoy a powerful feature representation learning capability to capture higher-order, long-range dependencies in text.

In the model training phase, since our model requires a large amount of labeled training data, which are often expensive to obtain or unavailable in many tasks, we additionally use self-training to generate pseudo-labeled training data to train our model to improve the accuracy and generalization performance of the model. Self-training [29,30] is a widely used semi-supervised learning method [31]. Most related studies follow the framework of traditional Self-Training and Co-Training and focus on designing better policies for selecting confident samples: trains the base model (student) on a small amount of labeled data, applies it to pseudo-labeled task-specific unlabeled data, uses pseudo-labels to augment the labeled data, and retrains the student model iteratively. Recently, self-training has been shown to obtain state-of-the-art performance in tasks such as image classification [32,33], few-shot text classification [34,35], and neural machine translation [36,37], and has shown complementary advantages to unsupervised pre-training [32].

In this paper, we propose a machine reading comprehension model based on multi-task fusion training, and we construct a multi-task fusion training reading comprehension model based on the BERT pre-training model. The model uses the BERT pre-training model to obtain contextual representations, which is shared by three downstream sub-modules of span extraction, yes/no question answering, and unanswerable questions. Next, we fuse the outputs of the three submodules into a new span extraction output, and we use a cross-entropy loss function for global training. We use a self-training method to generate

pseudo-labeled data from unlabeled data to expand labeled data iteratively, thus improving the model performance and achieving better generalization. The experiments show that our model can efficiently handle different tasks. We achieved 83.2 EM and 86.7 F1 scores on the SQuAD2.0 dataset and 73.0 EM and 85.3 F1 scores on the CAIL2019 dataset.

Passage:
The two halves of the city were actually founded and plotted as separate cities, but soon grew together. The north side is characterized by very diverse and fashionable urban neighborhoods near the city center and sprawling suburbs further north. South Oklahoma City is generally more blue collar working class and significantly more industrial, having grown up around the Stockyards and meat packing plants at the turn of the century, and is currently the center of the city's rapidly growing Latino community.

Question:
Which side is more urban and fashionable?

Gold Answer: North Oklahoma City

Figure 1. A extractive MRC example.

Passage:
The plague struck various countries in the Middle East during the pandemic, leading to serious depopulation and permanent change in both economic and social structures. As it spread to western Europe, the disease entered the region from southern Russia also. By autumn 1347, the plague reached Alexandria in Egypt, probably through the port's trade with Constantinople, and ports on the Black Sea. During 1347, the disease travelled eastward to Gaza, and north along the eastern coast to cities in Lebanon, Syria and Palestine.

Question:
What was one of the cities that had a port on the Black Sea?

Gold Answer: <no answer>

Figure 2. A non-extractive MRC example.

2. Materials and Methods

2.1. Materials

Two publicly available datasets are used in this study including SQuAD2.0 [6] and CAIL2019 Reading comprehension dataset [8]. SQuAD 2.0 is a widely used MRC benchmark dataset, which combines the 100,000 questions from SQuAD 1.1 [4] with over 50,000 new, unanswerable questions that are the crowd adversarially written to look similar to the answerable questions. The training dataset contains 87,000 answerable questions and 43,000 unanswerable questions. The CAIL2019 reading comprehension dataset is primarily concerned with civil and criminal first-instance judgments. The training set includes more than 39,000 questions, mainly span extraction, containing more than 5000 YES/NO question answering and more than 1000 unanswerable questions.

2.2. Method

We focus on three reading comprehension tasks: span extraction, Yes/No question answering, and unanswerable questions. They all can be described as a triplet <P, Q, A>, where P is a passage and Q is a question for P. When question Q is a span extraction task, the correct answer A is a span text in P; when question Q is a Yes/No question answering, the correct answer A is the text "YES" or "NO"; when question Q is an unanswerable question, the correct answer A is the null string. We have conducted experiments on both SQUAD2.0 and CAIL2019 reading comprehension datasets, and our model should be able to predict the begin and end position in passage P and extract the span text as answer A for a question which is the span extraction task and return the text "YES" or "NO" for the

question which is the Yes/No question answering. For the unanswerable question, the model can return a null string.

Our MRC model mainly includes two aspects of improvement: the multi-task fusion training model and the self-training method. The multi-task fusion training model mainly focuses on the fusion of multi-task outputs and training methods. Self-training mainly focuses on generating pseudo-labeled training data with high confidence from unlabeled data to expand the training data.

2.2.1. Embedding

We concatenate question and passage texts as the input, which is firstly represented as embedding vectors to feed an encoder layer. In detail, the input texts are first tokenized to word pieces(subword tokens). Let $T = \{t_1, t_2, ..., t_n\}$ denote as a sequence of subword tokens of length N. For each token, the embedding vector of the input text is the sum vector of its token embedding, position embedding, and token-type embedding.

We take $X = \{x_1, x_2, ..., x_n\}$ as the output of the encoding layer, which is an embedding feature vector of encoded sentence tokens of length n. Then, the embedded feature vector is fed to the interaction layer to obtain the contextual representation vector.

2.2.2. Interaction

Following Devlin at all. [24], the encoded sequence X is processed using multiple layers of Transformers [38] to learn the context representation vector. In the next section, we use $H = \{h_1, h_2, ..., h_n\}$ to denote the last layer of hidden states of the input sequence. H_c is denoted as the last layer of hidden states of the first token (classification token) in the sequence.

2.2.3. Multi-Tasking Fusion Training

As is shown Figure 3 depicts the whole architecture of our model. The unanswerable question task is to return an empty string as the answer after discriminating the question from the text, so we use a linear layer into which H_c is fed as input to obtain the probability value *unk* of the unanswerable question. The span extraction task is to find a span in the passage as an answer. We feed the last_layer hidden states H into a linear layer with softmax operations to obtain two probability sequences L_s and L_e. The Yes/No question answering task is to return the text "YES" or "NO" as the answer after understanding the question and the passage. We use sum-pooling to compress the hidden state H of the last layer into a vector as input to a linear layer to obtain predicted probability values *yes* and *no*.

We use the predicted probability values of the three tasks *Ls*, *unk*, *yes*, and *no* spliced together as the new start probability sequence S. In the same way, we can obtain the new end probability sequence E.

The training target for our model is defined as the cross-entropy loss of start S and end E:

$$L = -\frac{1}{N}\sum_{i=0}^{N}\left[log\left(p^s_{y^s_i}\right) + log\left(p^e_{y^e_i}\right)\right], \tag{1}$$

where y^s_i and y^e_i represent the true start and end positions of sample i respectively, and N is the number of samples.

For prediction, given output start and end probability sequences S and E, we calculate the span score $score_{span}$, the unanswer score $score_{ua}$, the Yes-answer score $score_{yes}$ and the no-answer score $score_{no}$:

$$\begin{aligned} score_{span} &= max(S_k + E_l), 0 \le k \le l \le n-2, \\ score_{ua} &= S_0 + E_0, \\ score_{yes} &= S_{n-1} + E_{n-1}, \\ score_{no} &= S_n + E_n. \end{aligned} \tag{2}$$

We get 4 answer scores, and we choose the answer with the largest score as the final answer by comparing them.

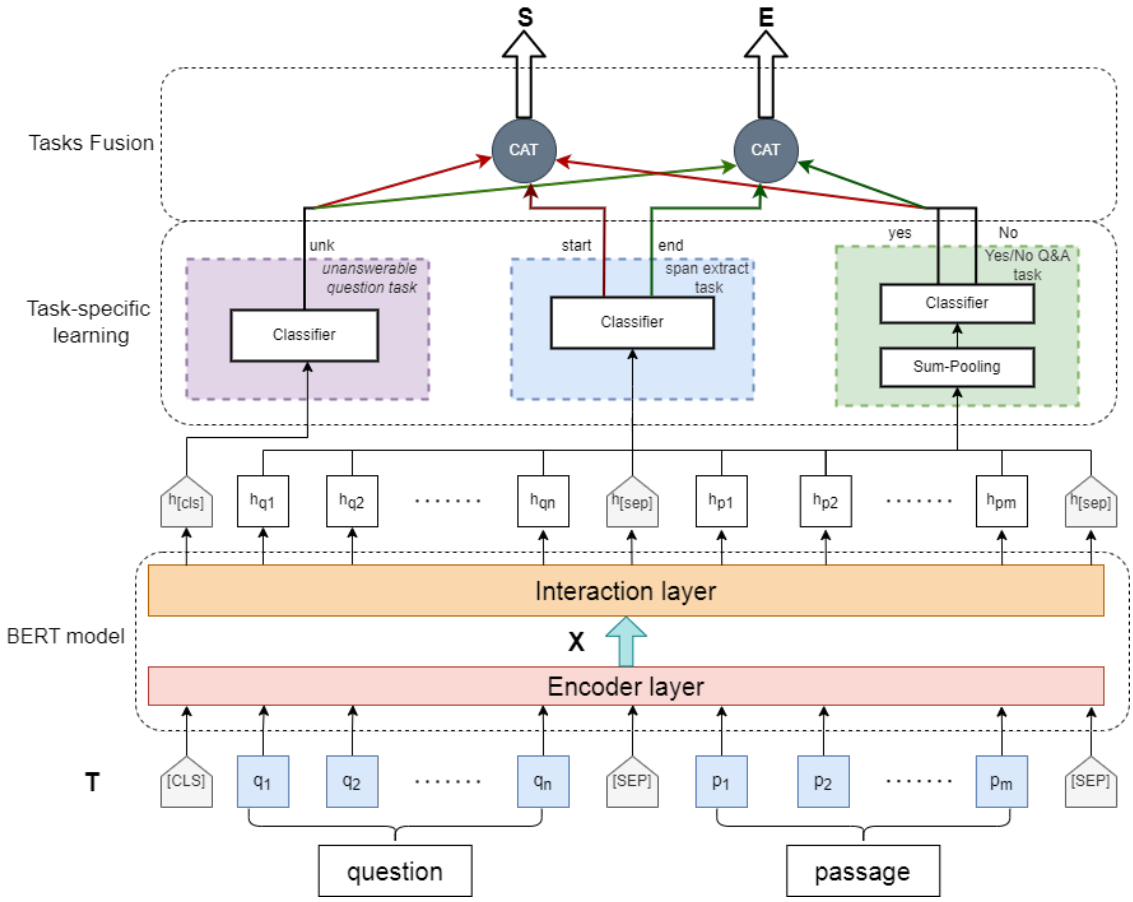


Figure 3. An overview of our base model. T is the sequence of subword tokens after embedding, X is the outputs of the encoder layer. Task-specific learning Module sets up three sub-modules to generate probability sequences or values of different answers for each sample. CAT means concatenates the input sequence or value.

2.2.4. Self Training

The complete two-stage process is as specified in Algorithm 1. First, in the fine-tuning phase, the pre-trained MRC model is fine-tuned in a supervised manner on the labeled dataset D_s. The model is named M_0 and is then used as the starting point for the next phase. In the self-training phase, the training and labeling process is iterated several times. For each iteration i, we first generated pseudo-labels for the unlabeled dataset D_t using the model M_i and then trained M_0 on the unlabeled dataset using these labels.

Algorithm 1 MRC self-training process

Require: Pre-trained MRC model, M_p; Labeled dataset, D_s; Unlabeled dataset, D_t; Number of self-training iterations, N; Threshold for filtering pseudo-labels, δ

$M_0 \leftarrow train(M_p, D_s)$
$D_0 \leftarrow label(M_0, D_t, \delta)$
for $i \in [1, n]$ **do**
    $M_i \leftarrow train(M_{i-1}, D_{i-1})$
    $D_i \leftarrow label(M_i, D_t, \delta)$
end for
return M_n

The unlabeled dataset can be described as a tuple $< P_t, Q_t >$, where $P_t = \{p_1, p_2, ..., p_n\}$ is a passage, $Q_t = \{q_{11}, q_{12}, ..., q_{nm}\}$ is a question over P_t. To obtain pseudo-labels, we use the model running on the unlabeled dataset D_t, and Algorithm 2 explains the labeling process. For each unlabeled example (c_i, q_i), the model gives t predicted answers $A_{ij} = \{a_{ij1}, a_{ij2}, ..., a_{ijp}\}$ and the corresponding confidence $E_{ij} = \{e_{ij1}, e_{ij2}, ..., e_{ijp}\}$. Then we use a threshold to filter the data, only the examples with maximum confidence above the threshold are retained. For each question, the answer with the maximum confidence is used as a pseudo-label.

Algorithm 2 Pseudo-label generation

Require: Model used in self-training process, M; Passages in unlabeled dataset, C_t; Questions in unlabeled dataset, Q_t; Unlabeled dataset, D_t; Threshold for filtering pseudo-labels, δ

$D^{'} \leftarrow \emptyset$
for $c_i \in C_t$ **do**
  for $q_{ij} \in Q_t$ **do**
    $A_{ij}, E_{ij} \in M(c_i, q_{ij})$
    if $max_\tau\ E_{ij} \geq \delta$ **then**
      $k \leftarrow argmax\ E_{ij}$
      $D^{'} \leftarrow D^{'} \cup (p_i, q_{ij}, a_{ijk})$
    end if
  end for
end for
return D'

In the span extraction task, the model answers questions by generating two probability distributions for contextual tokens. One is the start position and the other is the end position. The answer is then extracted by choosing the span between the start and end positions.

Here, we consider probability as a measure of confidence. More specifically, we take the distributions of the start and end tokens as input, then filter out unlikely candidates (for example, candidates whose end token precedes the start token) and perform a beam search with the sum of the start/end distributions as the confidence. The threshold was set to 0.95, which means that only those examples with the most confident answers scoring greater than 0.95 were used in the self-training phase.

3. Experiments and Results

3.1. Experiments

3.1.1. Evaluation

Metrics: Two official metrics are used to evaluate the performance of the model. Exact Match (EM) and a softer metric F1 score, which measures the average overlap between predicted and ground-truth answers at the token level.

Exact Match (EM): This metric measures the percentage of predictions that match any one of the ground truth answers exactly. For each question+answer pair, if the characters of the model's prediction exactly match the characters of (one of) the True Answer(s), EM = 1, otherwise EM = 0. This is a strict all-or-nothing metric; being off by a single character results in a score of 0. When assessing against a negative example, if the model predicts any text at all, it automatically receives a 0 for that example.

F1 Score: It is a common metric for classification problems, and is widely used in Question Answering. It is appropriate when we care equally about *precision* and *recall*. In this case, it's computed over the individual words in the prediction against those in the True Answer. The number of shared words between the prediction and the truth is the basis of the F1 score: *precision* is the ratio of the number of shared words to the total number of words in the prediction, and *recall* is the ratio of the number of shared words to the total number of words in the ground truth. The formula for the F1 score is the following:

$$F1\ score = 2 * \frac{precision * recall}{precision + recall} \quad (3)$$

3.1.2. Setup

We use the available pre-trained models as encoders to build the baseline MRC models: BERT, ALBERT, ELECTRA. Our implementations for BERT are based on Transformers' public Pytorch version of the scheme. We use pre-trained language model weights in the encoder module of the model, using all official hyperparameters. For fine-tuning in our task, we set the initial learning rate in $\{2 \times 10^{-5}, 3 \times 10^{-5}\}$ with a warmup rate of 0.1, and L2 weight decay of 0.01. Specifically, 2×10^{-5} for BERT, 3×10^{-5} for ALBERT and ELECTRA. For the batch size, BERT is set to 20, ALBERT and ELECTRA are set to 8. In all experiments, the maximum number of epochs was set to 3. Texts are tokenized using wordpieces [39], with a maximum length of 512. Hyperparameters were selected using the development set.

3.2. Results

Tables 1 and 2 compare our model with the current more advanced models on the SQuAD2.0 and CAIL2019 datasets. In the English dataset SQuAD2.0, our model improves compared to the NeurQuRI and BERT models but lags behind the ALBERT and ELECTRA models. This is because our model is constructed based on the pre-trained language model, and the ability of the pre-trained model to learn generic language representations is important for the overall model performance. Currently available pre-trained language models for machine-reading comprehension tasks include BERT, ALBERT, and ELECTRA. The ALBERT model as an improved model of the BERT model can effectively improve the downstream performance of multi-sentence coding tasks through three improvements: factorized embedding parameterization, cross-layer parameter sharing, and inter-sentence coherence loss. The ELECTRA model proposes a model training framework borrowed from the design of GAN networks and a new pre-training task of replaced token detection. These two improvements allow ELECTRA to learn more contextual representations than BERT. Theoretically, our model can be designed based on the ALBERT or ELECTRA pre-trained model. However, the problem is that the training time of the ALBERT or ELECTRA pre-trained model is much longer than that of the BERT model, and our self-training method requires iterative training of the model. As such, the time cost is unacceptable. In the Chinese reading comprehension dataset CAIL2019, we did not find an available Chinese ALBERT pre-training model, so we did not conduct a comparison experiment of ALBERT models. Our model improves compared to NeurQuRI, BERT, and ELECTRA models. Benefiting from our designed task-specific learning module, our model can identify different task samples and make correct answers better than other models.

Table 1. The results (%) for SQuAD2.0 dataset. Time means Training time of model, *ST* means our Self Training method (Section 2.2.4).

Model	EM (%)	*F*1 (%)	Time (h)
NeurQuRI	81.3	84.3	7
BERT	82.1	84.8	8.5
ALBERT	86.9	89.8	22.5
ELECTRA	87.0	89.9	20
Our model	82.9	86.4	8.5
Our model + *ST*	83.2 (+0.3)	86.7 (+0.3)	42.5 (5 iters)

We fuse the outputs of different tasks into a new span extract output, train the model using a cross-entropy loss function. It can effectively prevent the imbalance from different task losses that confuse the whole model training. Our self-training method is also applied to the model successfully. The experiments show that the self-training method increases

0.3EM and 0.3F1 scores in the SQuAD2.0 dataset, and 0.9EM and 1.1F1 scores in the CAIL2019 reading comprehension dataset compared to our base model.

Table 2. The results (%) for CAIL2019 Reading comprehension dataset. Time means Training time of model, *ST* means our Self Training method (Section 2.2.4).

Model	EM (%)	*F*1 (%)	Time (h)
NeurQuRI	67.2	79.8	5
BERT	69.5	81.7	6
ELECTRA	71.2	83.6	11
Our model	72.1	84.2	6
Our model + *ST*	73.0 (+0.9)	85.3 (+1.1)	30 (5 iters)

4. Discussion

This paper compares the performance of three methods, including MLT with Fusion training, MLT with three auxiliary losses, and a pipeline approach. We implemented three comparative experiments in the CAIL2019 dataset. Based on the idea of Hu et al. [28], we adopt a BERT model to obtain context representation and use three auxiliary loss functions to process the output of different task modules. The training loss of the whole model is the weighted sum of the losses of the three modules. The pipeline approach uses a BERT model for training the extraction MRC task, turns non-extracted MRC tasks into a classification task, and learns the classification task directly for the sentences in each passage. Finally, the predicted results of the two models are combined and the scores are calculated.

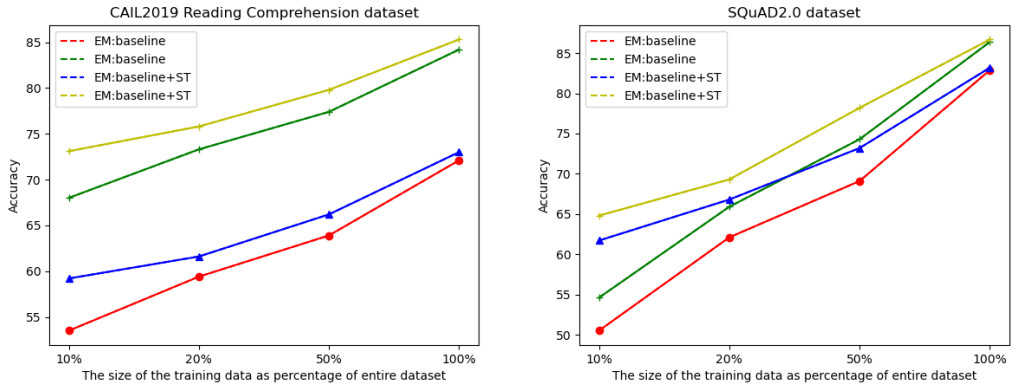
The specific experimental comparison results are shown in Table 3. It can be seen that the performance of the pipeline approach is lower compared to the other two approaches. It may be the pipeline approach only optimizes the loss of one task and lacks the interaction of other tasks. However, the pipeline approach also has its advantages, such as better controllability of the results for each subtask and the convenience of manual observation for each subtask. In multi-task learning, the magnitude of task loss is considered a perceived similarity measure. Imbalanced tasks in this regard produce largely varied gradients which can confuse model training. MLT with three auxiliary losses uses a task weighting coefficient to help normalize the gradient magnitudes across the tasks. However, its specific implementation employing an average task loss weighting coefficient does not achieve the balance among tasks, which may lead to a bias of the model toward a particular task during training. Our research revealed that the outputs of the different MRC task modules can be fused into a sequence, and we put it into a loss function for training. The model can automatically learn the weight coefficient between different tasks during training and make reasonable judgments. Experiments show that our model, although lower than MLT with three auxiliary losses in the EM metric, exceeds it by 0.8% in the *F*1 metric.

Table 3. Comparative experimental results of three multi-task learning methods.

Training Method	EM	*F*1
MLT with Fusion training	72.1	84.2
MLT with Three Auxiliary losses	72.6	83.4
Pipeline method	70.6	81.0

It is well known that the size of the labeled dataset affects the performance of the pre-trained model for downstream task fine-tuning. After that, we apply self-training to the models trained above to investigate the effect of labeled datasets of different sizes on the effectiveness of the self-training method for the base model. We set the size of the unlabeled dataset used for the self-training approach to 100%. Figure 4 shows that the evaluation performance of base model fine-tuning is highly dependent on the size of the domain labeled data, and the self-training method always improves the evaluation performance of

the base model. However, as the evaluation performance of the base model improves, the self-training method has less and less improvement on the effectiveness of the base model.



(**a**) (**b**)

Figure 4. The results of different sized labeled datasets for self-training improvement. (**a**) SQuAD 2.0 dataset. (**b**) CAIL2019 Reading Comprehension dataset.

5. Conclusions

In this paper, we construct a multi-task fusion training reading comprehension model based on a BERT pre-training model. The model uses the BERT pre-training model to obtain contextual representations, which are then shared by three downstream sub-modules for span extraction, yes/no question answering and unanswerable questions, and we next fuse the outputs of the three sub-modules into a new span extraction output and use the fused cross-entropy loss function for global training. We use self-training to generate pseudo-labeled training data to train our model to improve its accuracy and generalization performance. However, our self-training approach requires iteratively training a model, which consumes a lot of time, and we consider it needs further optimization. Our model is designed for only three specific tasks and cannot be widely applied to more machine reading comprehension task scenarios. We hope to explore more effective multi-task methods for machine-reading comprehension in the future.

Author Contributions: Conceptualization J.O. and M.F.; methodology, J.O. and M.F.; software, M.F.; validation, J.O. and M.F.; formal analysis, J.O. and M.F.; investigation, J.O. and M.F.; resources, J.O.; data curation, J.O. and M.F.; writing—original draft preparation, M.F.; writing—review and editing, J.O. and M.F.; visualization, M.F.; supervision, J.O.; project administration, J.O. and M.F.; funding acquisition, J.O. and M.F. All authors have read and agreed to the published version of the manuscript.

Funding: This research has been supported by Key Projects of the Ministry of Science and Technology of the People Republic of China (2020YFC0832401).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The CAIL2019 reading comprehension datasets used in our study are available at https://github.com/china-ai-law-challenge/CAIL2019. The SQuAD 2.0 datasets are available at https://rajpurkar.github.io/SQuAD-explorer.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Hermann, K.M.; Kocisky, T.; Grefenstette, E.; Espeholt, L.; Kay, W.; Suleyman, M.; Blunsom, P. Teaching machines to read and comprehend. *Adv. Neural Inf. Process. Syst.* **2015**, *28*, 1693–1701.
2. Zhang, Z.; Yang, J.; Zhao, H. Retrospective Reader for Machine Reading Comprehension. In *Proceedings of the AAAI Conference on Artificial Intelligence, Virtual, 2–9 February 2021*; AAAI Press: Palo Alto, CA, USA, 2021; Volume 35, pp. 14506–14514.
3. Xie, Q.; Lai, G.; Dai, Z.; Hovy, E. Large-scale Cloze Test Dataset Created by Teachers. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, 31 October–4 November 2018; pp. 2344–2356.
4. Rajpurkar, P.; Zhang, J.; Lopyrev, K.; Liang, P. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, Austin, Texas, USA, 1–5 November 2016; pp. 2383–2392.
5. Inoue, N.; Stenetorp, P.; Inui, K. R4C: A Benchmark for Evaluating RC Systems to Get the Right Answer for the Right Reason. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Stroudsburg, PA, USA, 6–8 July 2020; Association for Computational Linguistics: Stroudsburg, PA, USA, 2020; pp. 6740–6750.
6. Rajpurkar, P.; Jia, R.; Liang, P. Know what you don't know: Unanswerable questions for SQuAD. *arXiv* **2018**, arXiv:1806.03822.
7. Reddy, S.; Chen, D.; Manning, C.D. Coqa: A conversational question answering challenge. *Trans. Assoc. Comput. Linguist.* **2019**, *7*, 249–266. [CrossRef]
8. Xiao, C.; Zhong, H.; Guo, Z.; Tu, C.; Liu, Z.; Sun, M.; Xu, J. Cail2019-scm: A dataset of similar case matching in legal domain. *arXiv* **2019**, arXiv:1911.08962.
9. Jacob, I.J. Performance evaluation of caps-net based multitask learning architecture for text classification. *J. Artif. Intell.* **2020**, *2*, 1–10.
10. Peng, Y.; Chen, Q.; Lu, Z. An empirical study of multi-task learning on BERT for biomedical text mining. *arXiv* **2020**, arXiv:2005.02799.
11. Ruder, S.; Bingel, J.; Augenstein, I.; Søgaard, A. Latent multi-task architecture learning. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January 2019; AAAI: Honolulu, HI, USA, 2019; Volume 33, pp. 4822–4829.
12. Zhang, Y.; Yang, Q. A survey on multi-task learning. *IEEE Trans. Knowl. Data Eng.* **2021**, *2*, 1–10. [CrossRef]
13. Liu, X.; Li, W.; Fang, Y.; Kim, A.; Duh, K.; Gao, J. Stochastic answer networks for squad 2.0. *arXiv* **2018**, arXiv:1809.09194.
14. Hu, M.; Wei, F.; Peng, Y.; Huang, Z.; Yang, N.; Li, D. Read+ verify: Machine reading comprehension with unanswerable questions. In Proceedings of the AAAI Conference on Artificial Intelligence, Palo Alto, CA, USA, 2–9 February 2019; Volume 33, pp. 6529–6537.
15. Back, S.; Chinthakindi, S.C.; Kedia, A.; Lee, H.; Choo, J. NeurQuRI: Neural question requirement inspector for answerability prediction in machine reading comprehension. In Proceedings of the International Conference on Learning Representations, New Orleans, LA, USA, 6–9 May 2019.
16. Zhang, Z.; Wu, Y.; Zhou, J.; Duan, S.; Zhao, H.; Wang, R. SG-Net: Syntax-guided machine reading comprehension. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 34, pp. 9636–9643.
17. Kadlec, R.; Schmid, M.; Bajgar, O.; Kleindienst, J. Text Understanding with the Attention Sum Reader Network. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Berlin, Germany, 7–12 August 2016; pp. 908–918.
18. Dhingra, B.; Liu, H.; Yang, Z.; Cohen, W.W.; Salakhutdinov, R. Gated-Attention Readers for Text Comprehension. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Vancouver, BC, Canada, 30 July–4 August 2017; pp. 1832–1846.
19. Park, C.; Song, H.; Lee, C. S3-NET: SRU-based sentence and self-matching networks for machine reading comprehension. *ACM Trans. Asian -Low-Resour. Lang. Inf. Process. (TALLIP)* **2020**, *19*, 1–14. [CrossRef]
20. Cui, Y.; Chen, Z.; Wei, S.; Wang, S.; Liu, T.; Hu, G. Attention-over-Attention Neural Networks for Reading Comprehension. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Vancouver, BC, Canada, 30 July–4 August 2017; pp. 593–602.
21. Kim, J.H.; Jun, J.; Zhang, B.T. Bilinear attention networks. *arXiv* **2018**, arXiv:1805.07932.
22. Sarzynska-Wawer, J.; Wawer, A.; Pawlak, A.; Szymanowska, J.; Stefaniak, I.; Jarkiewicz, M.; Okruszek, L. Detecting formal thought disorder by deep contextualized word representations. *Psychiatry Res.* **2021**, *304*, 114135. [CrossRef] [PubMed]
23. Brown, T.B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Amodei, D. Language models are few-shot learners. *arXiv* **2020**, arXiv:2005.14165.
24. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv* **2018**, arXiv:1810.04805.
25. Yang, Z.; Dai, Z.; Yang, Y.; Carbonell, J.; Salakhutdinov, R.R.; Le, Q.V. Xlnet: Generalized autoregressive pretraining for language understanding. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 5753–5763.
26. Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Stoyanov, V. Roberta: A robustly optimized bert pretraining approach. *arXiv* **2019**, arXiv:1907.11692.
27. Lan, Z.; Chen, M.; Goodman, S.; Gimpel, K.; Sharma, P.; Soricut, R. Albert: A lite bert for self-supervised learning of language representations. *arXiv* **2019**, arXiv:1909.11942.

28. Clark, K.; Luong, M.T.; Le, Q.V.; Manning, C.D. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv* **2020**, arXiv:2003.10555.
29. Lee, D.H. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on Challenges in Representation Learning*; ICML: Atlanta, GA, USA, 2013; Volume 3, p. 896.
30. Yarowsky, D. Unsupervised word sense disambiguation rivaling supervised methods. In Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics, Cambridge, MA, USA, 26–30 June 1995; pp. 189–196.
31. Zhu, X.; Goldberg, A.B. Introduction to semi-supervised learning. *Synth. Lect. Artif. Intell. Mach. Learn.* **2009**, *3*, 1–130. [CrossRef]
32. Zoph, B.; Ghiasi, G.; Lin, T.Y.; Cui, Y.; Liu, H.; Cubuk, E.D.; Le, Q.V. Rethinking pre-training and self-training. *arXiv* **2020**, arXiv:2006.06882.
33. Zhao, R.; Liu, T.; Xiao, J.; Lun, D.P.; Lam, K.M. Deep multi-task learning for facial expression recognition and synthesis based on selective feature sharing. In Proceedings of the 2020 25th International Conference on Pattern Recognition (ICPR), Milan, Italy, 10–15 January 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 4412–4419.
34. Wang, Y.; Mukherjee, S.; Chu, H.; Tu, Y.; Wu, M.; Gao, J.; Awadallah, A.H. Adaptive self-training for few-shot neural sequence labeling. *arXiv* **2020**, arXiv:2010.03680.
35. Li, C.; Li, X.; Ouyang, J. Semi-Supervised Text Classification with Balanced Deep Representation Distributions. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, Bangkok, Thailand, 1–6 August 2021; pp. 5044–5053.
36. He, J.; Gu, J.; Shen, J.; Ranzato, M.A. Revisiting self-training for neural sequence generation. *arXiv* **2019**, arXiv:1909.13788.
37. Jiao, W.; Wang, X.; Tu, Z.; Shi, S.; Lyu, M.R.; King, I. Self-Training Sampling with Monolingual Data Uncertainty for Neural Machine Translation. *arXiv* **2021**, arXiv:2106.00941.
38. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems*; NeurIPS Proceedings: Long Beach, CA, USA, 2017; pp. 5998–6008.
39. Wu, Y.; Schuster, M.; Chen, Z.; Le, Q.V.; Norouzi, M.; Macherey, W.; Dean, J. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv* **2016**, arXiv:1609.08144.