Multi-task Learning with Sample Re-weighting for Machine Reading Comprehension

Yichong Xu[1]*, **Xiaodong Liu**[2], **Yelong Shen**[2], **Jingjing Liu**[2] **and Jianfeng Gao**[2]
[1] Carnegie Mellon University
[2] Microsoft Research
yichongx@cs.cmu.edu
{xiaodl, yeshen, jingjl,jfgao}@microsoft.com

Abstract

We propose a multi-task learning framework to learn a joint Machine Reading Comprehension (MRC) model that can be applied to a wide range of MRC tasks in different domains. Inspired by recent ideas of data selection in machine translation, we develop a novel sample re-weighting scheme to assign sample-specific weights to the loss. Empirical study shows that our approach can be applied to many existing MRC models. Combined with contextual representations from pre-trained language models (such as ELMo), we achieve new state-of-the-art results on a set of MRC benchmark datasets. We release our code at https://github.com/xycforgithub/MultiTask-MRC.

1 Introduction

Machine Reading Comprehension (MRC) has gained growing interest in the research community (Rajpurkar et al., 2016; Yu et al., 2018). In an MRC task, the machine reads a text passage and a question, and generates (or selects) an answer based on the passage. This requires the machine to possess strong comprehension, inference and reasoning capabilities. Over the past few years, there has been much progress in building end-to-end neural network models (Seo et al., 2016) for MRC. However, most public MRC datasets (e.g., SQuAD, MS MARCO, TriviaQA) are typically small (less than 100K) compared to the model size (such as SAN (Liu et al., 2018c,b) with around 10M parameters). To prevent over-fitting, recently there have been some studies on using pre-trained word embeddings (Pennington et al., 2014) and contextual embeddings in the MRC model training, as well as back-translation approaches (Yu et al., 2018) for data augmentation.

Multi-task learning (Caruana, 1997) is a widely studied area in machine learning, aiming at better model generalization by combining training datasets from multiple tasks. In this work, we explore a multi-task learning (MTL) framework to enable the training of one universal model across different MRC tasks for better generalization. Intuitively, this multi-task MRC model can be viewed as an implicit data augmentation technique, which can improve generalization on the target task by leveraging training data from auxiliary tasks.

We observe that merely adding more tasks cannot provide much improvement on the target task. Thus, we propose two MTL training algorithms to improve the performance. The first method simply adopts a sampling scheme, which randomly selects training data from the auxiliary tasks controlled by a ratio hyperparameter; The second algorithm incorporates recent ideas of data selection in machine translation (van der Wees et al., 2017). It learns the sample weights from the auxiliary tasks automatically through language models.

Prior to this work, many studies have used upstream datasets to augment the performance of MRC models, including word embedding (Pennington et al., 2014), language models (ELMo) (Peters et al., 2018) and machine translation (Yu et al., 2018). These methods aim to obtain a robust semantic encoding of both passages and questions. Our MTL method is orthogonal to these methods: rather than enriching semantic embedding with external knowledge, we leverage existing MRC datasets across different domains, which help make the whole comprehension process more robust and universal. Our experiments show that MTL can bring further performance boost when combined with contextual representations from pre-trained language models, e.g., ELMo (Peters et al., 2018).

*Most of this work was performed when the author was interning at Microsoft.

Proceedings of NAACL-HLT 2019, pages 2644–2655
Minneapolis, Minnesota, June 2 - June 7, 2019.

To the best of our knowledge, this is the first work that systematically explores multi-task learning for MRC. In previous methods that use language models and word embedding, the external embedding/language models are pre-trained separately and remain fixed during the training of the MRC model. Our model, on the other hand, can be trained with more flexibility on various MRC tasks. MTL is also faster and easier to train than embedding/LM methods: our approach requires no pre-trained models, whereas back translation and ELMo both rely on large models that would need days to train on multiple GPUs (Jozefowicz et al., 2016; Peters et al., 2018).

We validate our MTL framework with two state-of-the-art models on four datasets from different domains. Experiments show that our methods lead to a significant performance gain over single-task baselines on SQuAD (Rajpurkar et al., 2016), NewsQA (Trischler et al., 2017) and WhoDid-What (Onishi et al., 2016), while achieving state-of-the-art performance on the latter two. For example, on NewsQA (Trischler et al., 2017), our model surpassed human performance by **13.4** (46.5 vs 59.9) and **3.2** (72.6 vs 69.4) absolute points in terms of exact match and F1.

The contribution of this work is three-fold. First, we apply multi-task learning to the MRC task, which brings significant improvements over single-task baselines. Second, the performance gain from MTL can be easily combined with existing methods to obtain further performance gain. Third, the proposed sampling and re-weighting scheme can further improve the multi-task learning performance.

2 Related Work

Studies in machine reading comprehension mostly focus on architecture design of neural networks, such as bidirectional attention (Seo et al., 2016), dynamic reasoning (Xu et al., 2017), and parallelization (Yu et al., 2018). Some recent work has explored transfer learning that leverages outdomain data to learn MRC models when no training data is available for the target domain (Golub et al., 2017). In this work, we explore multi-task learning to make use of the data from other domains, while we still have access to target domain training data.

Multi-task learning (Caruana, 1997) has been widely used in machine learning to improve generalization using data from multiple tasks. For natural language processing, MTL has been successfully applied to low-level parsing tasks (Collobert et al., 2011), sequence-to-sequence learning (Luong et al., 2015), and web search (Liu et al., 2015). More recently, (McCann et al., 2018) proposes to cast all tasks from parsing to translation as a QA problem and use a single network to solve all of them. However, their results show that multi-task learning hurts the performance of most tasks when tackling them together. Differently, we focus on applying MTL to the MRC task and show significant improvement over single-task baselines.

Our sample re-weighting scheme bears some resemblance to previous MTL techniques that assign weights to tasks (Kendall et al., 2018). However, our method gives a more granular score for each sample and provides better performance for multi-task learning MRC.

3 Model Architecture

We call our model Multi-Task-SAN (MT-SAN), which is a variation of SAN (Liu et al., 2018c) model with two main differences: i) we add a highway network layer after the embedding layer, the encoding layer and the attention layer; ii) we use exponential moving average (Seo et al., 2016) during evaluation. The SAN architecture and our modifications are briefly described below and in Section 5.2, and detailed description can be found in (Liu et al., 2018c).

3.1 Input Format

For most tasks we consider, our MRC model takes a triplet (Q, P, A) as input, where $Q = (q_1, ..., q_m), P = (p_1, ..., p_n)$ are the word index representations of a question and a passage, respectively , and $A = (a_{\text{begin}}, a_{\text{end}})$ is the index of the answer span. The goal is to predict A given (Q, P).

3.2 Lexicon Encoding Layer

We map the word indices of P and Q into their 300-dim Glove vectors (Pennington et al., 2014). We also use the following additional information for embedding words: i) 16-dim part-of-speech (POS) tagging embedding; ii) 8-dim named-entity-recognition (NER) embedding; iii) 3-dim exact match embedding: $f_{\text{exact\_match}}(p_i) = \mathbb{I}(p_i \in Q)$, where matching is determined based on the original word, lower case, and lemma form,

respectively; iv) Question enhanced passage word embeddings: $f_{\text{align}}(p_i) = \sum_j \gamma_{i,j} h(\text{GloVe}(q_j))$, where

$$\gamma_{i,j} = \frac{\exp(h(\text{GloVe}(p_j)), h(\text{GloVe}(q_i)))}{\sum_{j'} \exp(h(\text{GloVe}(p_{j'})), h(\text{GloVe}(q_i)))} \quad (1)$$

is the similarity between word p_j and q_i, and $g(\cdot)$ is a 300-dim single layer neural net with Rectified Linear Unit (ReLU) $g(x) = \text{ReLU}(W_1 x)$; v) Passage-enhanced question word embeddings: the same as iv) but computed in the reverse direction. To reduce the dimension of the input to the next layer, the 624-dim input vectors of passages and questions are passed through a ReLu layer to reduce their dimensions to 125.

After the ReLU network, we pass the 125-dim vectors through a highway network (Srivastava et al., 2015), to adapt to the multi-task setting: $g_i = \text{sigmoid}(W_2 p_i^t), p_i^t = \text{ReLU}(W_3 p_i^t) \odot g_i + g_i \odot p_i^t$, where p_i^t is the vector after ReLU transformation. Intuitively, the highway network here provides a neuron-wise weighting, which can potentially handle the large variation in data introduced by multiple datasets.

3.3 Contextual Encoding Layer

Both the passage and question encodings go through a 2-layer Bidirectional Long-Short Term Memory (BiLSTM, Hochreiter and Schmidhuber, 1997) network in this layer. We append a 600-dim CoVe vector (McCann et al., 2017) to the output of the lexicon encoding layer as input to the contextual encoders. For the experiments with ELMo, we also append a 1024-dim ELMo vector. Similar to the lexicon encoding layer, the outputs of both layers are passed through a highway network for multi-tasking. Then we concatenate the output of the two layers to obtain $H^q \in \mathbb{R}^{2d \times m}$ for the question and $H^p = \mathbb{R}^{2d \times n}$ the passage, where d is the dimension of the BiLSTM.

3.4 Memory/Cross Attention Layer

We fuse H^p and H^q through cross attention and generate a working memory in this layer. We adopt the attention function from (Vaswani et al., 2017) and compute the attention matrix as $C = \text{dropout}\left(f_{\text{attention}}(\hat{H}^q, \hat{H}^p)\right) \in \mathbb{R}^{m \times n}$. We then use C to compute a question-aware passage representation as $U^p = \text{concat}(H^p, H^q C)$. Since a passage usually includes several hundred tokens, we use the method of (Lin et al., 2017) to apply self attention to the representations of passage to rearrange its information: $\hat{U}^p = U^p \text{drop}_{\text{diag}}(f_{\text{attention}}(U^p, U^p))$, where $\text{drop}_{\text{diag}}$ means that we only drop diagonal elements on the similarity matrix (i.e., attention with itself). Then, we concatenate U^p and $\hat{U}^p$ and pass them through a BiLSTM: $M = \text{BiLSTM}([U^p]; \hat{U}^p])$. Finally, output of the BiLSTM (after concatenating two directions) goes through a highway layer to produce the memory.

3.5 Answer Module

The base answer module is the same as SAN, which computes a distribution over spans in the passage. Firstly, we compute an initial state s_0 by self attention on H^q: $s_0 \leftarrow \text{Highway}\left(\sum_j \frac{\exp(w_4 H_j^q)}{\sum_{j'} \exp w_4 H_{j'}^q} \cdot H_j^q\right)$. The final answer is computed through T time steps. At step $t \in \{1, ..., T-1\}$, we compute the new state using a Gated Recurrent Unit (GRU, Cho et al., 2014) $s_t = \text{GRU}(s_{t-1}, x_t)$, where x_t is computed by attention between M and s_{t-1}: $x_t = \sum_j \beta_j M_j, \beta_j = \text{softmax}(s_{t-1} W_5 M)$. Then each step produces a prediction of the start and end of answer spans through a bilinear function: $P_t^{\text{begin}} = \text{softmax}(s_t W_6 M)$, $P_t^{\text{end}} = \text{softmax}(s_t W_7 M)$. The final prediction is the average of each time step: $P^{\text{begin}} = \frac{1}{T} \sum_t P_t^{\text{begin}}, P^{\text{end}} = \frac{1}{T} \sum_t P_t^{\text{end}}$. We randomly apply dropout on the step level in each time step during training, as done in (Liu et al., 2018c). During training, the objective is the log-likelihood of the ground truth: $l(Q, P, A) = \log P^{\text{begin}}(a_{\text{begin}}) + \log P^{\text{end}}(a_{\text{end}})$.

4 Multi-task Learning Algorithms

We describe our MTL training algorithms in this section. We start with a very simple and straightforward algorithm that samples one task and one mini-batch from that task at each iteration. To improve the performance of MTL on a target dataset, we propose two methods to re-weight samples according to their importance. The first proposed method directly lowers the probability of sampling from a particular auxiliary task; however, this probability has to be chosen using grid search. We then propose another method that avoids such search by using a language model.

Suppose we have K different tasks, the simplest version of our MTL training procedure is shown in Algorithm 1. In each epoch, we take all the mini-batches from all datasets and shuffle them for

Algorithm 1 Multi-task Learning of MRC

Input: k different datasets $\mathcal{D}_1, ..., \mathcal{D}_K$, max_epoch

Initialize the model $\mathcal{M}$
for epoch= $1, 2, ...,$ max_epoch **do**
  Divide each dataset $\mathcal{D}_k$ into N_k mini-batches $\mathcal{D}_k = \{b_1^k, ..., b_{N_k}^k\}, 1 \leq k \leq K$
  Put all mini-batches together and randomly shuffle the order of them, to obtain a sequence $B = (b_1, ..., b_L)$, where $L = \sum_k N_k$
  for each mini-batch $b \in B$ **do**
    Perform gradient update on $\mathcal{M}$ with loss $l(b) = \sum_{(Q,P,A)\in b} l(Q, P, A)$
  end for
  Evaluate development set performance
end for

Output: Model with best evaluation performance

Algorithm 2 Multi-task Learning of MRC with mixture ratio, targeting $\mathcal{D}_1$

Input: K different datasets $\mathcal{D}_1, ..., \mathcal{D}_K$, max_epoch, mixture ratio α

Initialize the model $\mathcal{M}$
for epoch= $1, 2, ...,$ max_epoch **do**
  Divide each dataset $\mathcal{D}_k$ into N_k mini-batches $\mathcal{D}_k = \{b_1^k, ..., b_{N_k}^k\}, 1 \leq k \leq K$
  $S \leftarrow \{b_1^1, ..., b_{N_1}^1\}$
  Randomly pick $\lfloor \alpha N_1 \rfloor$ mini-batches from $\bigcup_{k=2}^K \mathcal{D}_k$ and add to S
  Assign mini-batches in S in a random order to obtain a sequence $B = (b_1, ..., b_L)$, where $L = N_1 + \lfloor \alpha N_1 \rfloor$
  for each mini-batch $b \in B$ **do**
    Perform gradient update on $\mathcal{M}$ with loss $l(b) = \sum_{(Q,P,A)\in b} l(Q, P, A)$
  end for
  Evaluate development set performance
end for

Output: Model with best evaluation performance

model training, and the same set of parameters is used for all tasks. Perhaps surprisingly, as we will show in the experiment results, this simple baseline method can already lead to a considerable improvement over the single-task baselines.

4.1 Mixture Ratio

One observation is that the performance of our model using Algorithm 1 starts to deteriorate as we add more and more data from other tasks into our training pool. We hypothesize that the external data will inevitably bias the model towards auxiliary tasks instead of the target task.

To avoid such adverse effect, we introduce a mixture ratio parameter during training. The training algorithm with the mixture ratio is presented in Algorithm 2, with $\mathcal{D}_1$ being the target dataset. In each epoch, we use all mini-batches from $\mathcal{D}_1$, while only a ratio α of mini-batches from external datasets are used to train the model. In our experiment, we use hyperparameter search to find the best α for each dataset combination. This method resembles previous methods in multi-task learning to weight losses differently (e.g., Kendall et al., 2018), and is very easy to implement. In our experiments, we use Algorithm 2 to train our network when we only use 2 datasets for MTL.

4.2 Sample Re-Weighting

The mixture ratio (Algorithm 2) dramatically improves the performance of our system. However, it requires to find an ideal ratio by hyperparameter search which is time-consuming. Furthermore, the ratio gives the same weight to every auxiliary data, but the relevance of every data point to the target task can vary greatly.

We develop a novel re-weighting method to resolve these problems, using ideas inspired by data selection in machine translation (Axelrod et al., 2011; van der Wees et al., 2017). We use (Q^k, P^k, A^k) to represent a data point from the k-th task for $1 \leq k \leq K$, with $k = 1$ being the target task. Since the passage styles are hard to evaluate, we only evaluate data points based on Q^k and A^k. Note that only data from auxiliary task ($2 \leq k \leq K$) is re-weighted; target task data always have weight 1.

Our scores consist of two parts, one for questions and one for answers. For questions, we create language models (detailed in Section 5.2) using questions from each task, which we represent as LM_k for the k-th task. For each question Q^k from auxiliary tasks, we compute a cross-entropy score:

$$H_{C,Q}(Q^k) = -\frac{1}{m} \sum_{w \in Q^k} \log(LM_C(w)), \quad (2)$$

where $C \in \{1, k\}$ is the target or auxiliary task, m is the length of question Q^k, and w iterates over all words in Q^k.

It is hard to build language models for answers since they are typically very short (e.g., answers

Dataset	SQuAD(v1)	NewsQA	MS MARCO(v1)	WDW
# Training Questions	87,599	92,549	78,905	127,786
Text Domain	Wikipedia	CNN News	Web Search	Gigaword Corpus
Avg. Document Tokens	130	638	71	365
Answer type	Text span	Text span	Natural sentence	Cloze
Avg. Answer Tokens	3.5	4.5	16.4	N/A

Table 1: Statistics of the datasets. Some numbers come from (Sugawara et al., 2017).

on SQuAD includes only one or two words in most cases). We instead just use the length of answers as a signal for scores. Let l_a^k be the length of A^k, the cross-entropy answer score is defined as:

$$H_{C,A}(A^k) = -\log \text{freq}_C(l_a^k), \quad (3)$$

where freq_C is the frequency of answer lengths in task $C \in \{1, k\}$.

The cross entropy scores are then normalized over all samples in task C to create a comparable metric across all auxiliary tasks:

$$H'_{C,Q}(Q^k) = \frac{H_{C,Q}(Q^k) - \min(H_{C,Q})}{\max(H_{C,Q}) - \min(H_{C,Q})} \quad (4)$$

$$H'_{C,A}(A^k) = \frac{H_{C,A}(A^k) - \min(H_{C,A})}{\max(H_{C,A}) - \min(H_{C,A})} \quad (5)$$

for $C \in \{1, 2, ..., K\}$. For $C \in \{2, ..., K\}$, the maximum and minimum are taken over all samples in task k. For $C = 1$ (target task), they are taken over all available samples.

Intuitively, $H'_{C,Q}$ and $H'_{C,A}$ represents the similarity of text Q, A to task C; a low $H'_{C,Q}$ (resp. $H'_{C,A}$) means that Q^k (resp. A^k) is easy to predict and similar to C, and vice versa. We would like samples that are most similar from data in the target domain (low H'_1), and most different (informative) from data in the auxiliary task (high H'_k). We thus compute the following cross-entropy difference for each external data:

$$\begin{aligned}\text{CED}(Q^k, A^k) =& (H'_{1,Q}(Q^k) - H'_{k,Q}(Q^k)) + \\ & (H'_{1,A}(A^k) - H'_{k,A}(A^k)) \quad (6)\end{aligned}$$

for $k \in \{2, ..., K\}$. Note that a low CED score indicates high importance. Finally, we transform the scores to weights by taking negative, and normalize between $[0, 1]$:

$$\text{CED}'(Q^k, A^k) = 1 - \frac{\text{CED}(Q^k, A^k) - \min(\text{CED})}{\max(\text{CED}) - \min(\text{CED})}. \quad (7)$$

Here the maximum and minimum are taken over all available samples and task. Our training algorithm is the same as Algorithm 1, but for mini-batch b we instead use the loss

$$l(b) = \sum_{(P,Q,A) \in b} \text{CED}'(Q, A) l(P, Q, A) \quad (8)$$

in step 6. We define $\text{CED}'(Q^1, A^1) \equiv 1$ for all target samples (P^1, Q^1, A^1).

5 Experiments

Our experiments are designed to answer the following questions on multi-task learning for MRC:
1. Can we improve the performance of existing MRC systems using multi-task learning?
2. How does multi-task learning affect the performance if we combine it with other external data?
3. How does the learning algorithm change the performance of multi-task MRC?
4. How does our method compare with existing MTL methods?

We first present our experiment details and results for MT-SAN. Then, we provide a comprehensive study on the effectiveness of various MTL algorithms in Section 5.4. At last, we provide some additional results on combining MTL with DrQA (Chen et al., 2017) to show the flexibility of our approach [1].

5.1 Datasets

We conducted experiments on SQuAD (Rajpurkar et al., 2016), NewsQA(Trischler et al., 2017), MS MARCO (v1, Nguyen et al.,2016) and WDW (Onishi et al., 2016). Dataset statistics is shown in Table 1. Although similar in size, these datasets are quite different in domains, lengths of text, and types of task. In the following experiments, we will validate whether including external datasets as additional input information (e.g., pre-trained language model on these datasets) helps boost the performance of MRC systems.

[1] We include the results in the appendix due to space limitations.

Model	Dev Set Performance
Single Model without Language Models	EM,F1
BiDAF (Seo et al., 2016)	67.7, 77.3
SAN (Liu et al., 2018c)	76.24, 84.06
MT-SAN on SQuAD (single task, ours)	76.84, 84.54
MT-SAN on SQuAD+NewsQA(ours)	78.60, 85.87
MT-SAN on SQuAD+MARCO(ours)	77.79, 85.23
MT-SAN on SQuAD+NewsQA+MARCO(ours)	**78.72, 86.10**
Single Model with ELMo	
SLQA+ (Wang et al., 2018a)	80.0, 87.0
MT-SAN on SQuAD (single task, ours)	80.04, 86.54
MT-SAN on SQuAD+NewsQA(ours)	81.36, 87.71
MT-SAN on SQuAD+MARCO(ours)	80.37, 87.17
MT-SAN on SQuAD+NewsQA+MARCO(ours)	**81.58, 88.19**
BERT (Devlin et al., 2018)	**84.2, 91.1**
Human Performance (test set)	82.30, 91.22

Table 2: Performance of our method to train SAN in multi-task setting, competing published results, leaderboard results and human performance, on SQuAD dataset (single model). Note that BERT uses a much larger language model, and is not directly comparable with our results. We expect our test performance is roughly similar or a bit higher than our dev performance, as is the case with other competing models.

5.2 Experiment Details

We mostly focus on span-based datasets for MT-SAN, namely SQuAD, NewsQA, and MS MARCO. We convert MS MARCO into an answer-span dataset to be consistent with SQuAD and NewsQA, following (Liu et al., 2018c). For each question, we search for the best span using ROUGE-L score in all passage texts and use the span to train our model. We exclude questions with maximal ROUGE-L score less than 0.5 during training. For evaluation, we use our model to find a span in all passages. The prediction score is multiplied with the ranking score, trained following Liu et al. (2018a)'s method to determine the final answer.

We train our networks using algorithms in Section 4, using SQuAD as the target task. For experiments with two datasets, we use Algorithm 2; for experiments with three datasets we find the re-weighting mechanism in Section 4.2 to have a better performance (a detailed comparison will be presented in Section 5.4).

For generating sample weights, we build a LSTM language model on questions following the implementation of Merity et al. (2017) with the same hyperparameters. We only keep the 10,000 most frequent words, and replace the other words with a special out-of-vocabulary token.

Parameters of MT-SAN are mostly the same as in the original paper (Liu et al., 2018c). We utilize spaCy[2] to tokenize the text and generate part-of-speech and named entity labels. We use a 2-layer BiLSTM with 125 hidden units as the BiLSTM throughout the model. During training, we drop the activation of each neuron with 0.3 probability. For optimization, we use Adamax (Kingma and Ba, 2014) with a batch size of 32 and a learning rate of 0.002. For prediction, we compute an exponential moving average (EMA, Seo et al. 2016) of model parameters with a decay rate of 0.995 and use it to compute the model performance. For experiments with ELMo, we use the model implemented by AllenNLP [3]. We truncate passage to contain at most 1000 tokens during training and eliminate those data with answers located after the 1000th token. The training converges in around 50 epochs for models without ELMo (similar to the single-task SAN); For models with ELMo, the convergence is much faster (around 30 epochs).

5.3 Performance of MT-SAN

In the following sub-sections, we report our results on SQuAD and MARCO development sets,

[2]https://spacy.io

[3]https://allennlp.org/

as well as on the development and test sets of NewsQA [4]. All results are single-model performance unless otherwise noted.

The multi-task learning results of SAN on SQuAD are summarized in Table 2. By using MTL on SQuAD and NewsQA, we can improve the exact-match (EM) and F1 score by (2%, 1.5%), respectively, both with and without ELMo. The similar gain indicates that our method is orthogonal to ELMo. Note that our single-model performance is slightly higher than the original SAN, by incorporating EMA and highway networks. By incorporating with multi-task learning, it further improves the performance. The performance gain by adding MARCO is relatively smaller, with 1% in EM and 0.5% in F1. We conjecture that MARCO is less helpful due to its differences in both the question and answer style. For example, questions in MS MARCO are real web search queries, which are short and may have typos or abbreviations; while questions in SQuAD and NewsQA are more formal and well written.

Using 3 datasets altogether provides another marginal improvement. Our model obtains the best results among existing methods that do not use a large language model (e.g., ELMo). Our ELMo version also outperforms any other models which are under the same setting. We note that BERT (Devlin et al., 2018) uses a much larger model than ours(around 20x), and we leave the performance of combining BERT with MTL as interesting future work.

The results of multi-task learning on NewsQA are in Table 3. The performance gain with multi-task learning is even larger on NewsQA, with over 2% in both EM and F1. Experiments with and without ELMo give similar results. What is worth noting is that our approach not only achieves new state-of-art results with a large margin but also surpasses human performance on NewsQA.

Finally we report MT-SAN performance on MS MARCO in Table 4. Multi-tasking on SQuAD and NewsQA provides a similar performance boost in terms of BLEU-1 and ROUGE-L score as in the case of NewsQA and SQuAD. Our method does not achieve very high performance compared to previous work, probably because we do not apply common techniques like yes/no classification or cross-passage ranking (Wang et al., 2018b).

We also test the robustness of our algorithm by performing another set of experiments on SQuAD and WDW. WDW is much more different than the other three datasets (SQuAD, NewsQA, MS MARCO): WDW guarantees that the answer is always a person, whereas the percentage of

[4] The official submission for SQuAD v1.1 and MARCO v1.1 are closed, so we report results on the development set. According to their leaderboards, performances on development and test sets are usually similar.

Model	Dev Set	Test Set
Model W/o ELMo	EM,F1	EM, F1
Match-LSTM[1]	34.4, 49.6	34.9, 50.0
FastQA[2]	43.7, 56.1	42.8, 56.1
AMANDA[3]	48.4, 63.3	48.4, 63.7
MT-SAN (Single task)	55.8, 67.9	55.6, 68.0
MT-SAN (S+N)	**57.8, 69.9**	**58.3, 70.7**
Model With ELMo		
MT-SAN (Single task)	57.7, 70.4	57.0, 70.4
MT-SAN (S+N)	**60.1, 72.5**	**59.9, 72.6**
Human Performance	-,-	46.5, 69.4

Table 3: Performance of our method to train SAN in multi-task setting, with published results and human performance on NewsQA dataset. All SAN results are from our models. "S+N" means jointly training on SQuAD and NewsQA References: [1]: implemented by Trischler et al. (2017). [2]:Weissenborn et al.(2017). [3]: Kundu and Ng(2018).

Model	Scores
Single Model W/o ELMo	
FastQAExt[1] (test set)	33.99, 32.09
Reasonet++[2]	38.62, 38.01
V-Net[3]	-, 45.65
SAN[4]	43.85, 46.14
MT-SAN	34.13, 42.65
MT-SAN: SQuAD+MARCO	34.29, 43.47
MT-SAN: 3 datasets	**36.99, 43.64**
Single Model With ELMo	
MT-SAN	34.57, 42.88
MT-SAN: SQuAD+MARCO	37.02, 43.89
MT-SAN: 3 datasets	**37.12, 44.12**
Human Performance (test set)	48.02, 49.72

Table 4: Performance of our method to train SAN in multi-task setting, competing published results and human performance, on MS MARCO dataset. The scores stand for (BLEU-1, ROUGE-L) respectively. All SAN results are our results. "3 dataset" means we train using SQuAD+NewsQA+MARCO. References: [1]: (Weissenborn et al., 2017). [2]: implemented by (Shen et al., 2017). [3]:(Wang et al., 2018b). [4]: (Liu et al., 2018c)

Model	SQuAD	WDW
MT-SAN (Single Task)	76.8, 84.5	77.5
MT-SAN (S+W)	**77.6, 85.1**	**78.5**
SOTA(Yang et al., 2016).	86.2, 92.2	71.7
Human Performance	82.3, 91.2	84

Table 5: Performance of MT-SAN on SQuAD Dev and WDW test set. Accuracy is used to evaluate WDW. "S+W" means jointly training on SQuAD and WDW.

Model	EM, F1	+/-
QANet	73.6, 82.7	0.0, 0.0
QANet + BT	75.1, 83.8	+1.5,+1.1
SAN	76.8, 84.5	0.0, 0.0
MT-SAN	**78.7, 86.0**	**+1.9,+1.5**
SAN + ELMo	80.0, 86.5	+3.2,+2.0
MT-SAN + ELMo	**81.6, 88.2**	**+4.8, +3.7**

Table 6: Comparison of methods to use external data. BT stands for back translation (Yu et al., 2018).

such questions in SQuAD is 12.9%. Moreover, WDW is a cloze dataset, whereas in SQuAD and NewsQA answers are spans in the passage. We use a task-specific answer layer in this experiment and use Algorithm 2; the WDW answer module is the same as in AS Reader (Kadlec et al., 2016), which we describe in the appendix for completeness. Despite these large difference between datasets, our results (Table 5) show that MTL can still provide a moderate performance boost when jointly training on SQuAD (around 0.7%) and WDW (around 1%).

Comparison of methods using external data. As a method of data augmentation, we compare our approach to previous methods for MRC in Table 6. Our model achieves better performance than back translation. We also observe that language models such as ELMo obtain a higher performance gain than multi-task learning, however, combining it with multi-task learning leads to the most significant performance gain. This validates our assumption that multi-task learning is more robust and is different from previous methods such as language modeling.

5.4 Comparison of Different MTL Algorithms

In this section, we provide ablation studies as well as comparisons with other existing algorithms on the MTL strategy. We focus on MT-SAN without ELMo for efficient training.

Model	Performance
SQuAD + MARCO	EM,F1
Simple Combine (Alg. 1)	77.1, 84.6
Loss Uncertainty	77.3, 84.7
Mixture Ratio	77.8, 85.2
Sample Re-weighting	**77.9,85.3**
SQuAD + NewsQA + MARCO	
Simple Combine (Alg. 1)	77.6, 85.2
Loss Uncertainty	78.2, 85.6
Mixture Ratio	78.4, 85.7
Sample Re-weighting	**78.8**, **86.0**

Table 7: Comparison of different MTL strategies on MT-SAN. Performance is on SQuAD. Loss Uncertainty is from Kendall et al. (2018).

Table 7 compares different multi-task learning strategies for MRC. Both the mixture ratio (Sec 4.1) and sample re-weighting (Sec 4.2) improves over the naive baseline of simply combining all the data (Algorithm 1). On SQuAD+MARCO, they provide around 0.6% performance boost in terms of both EM and F1, and around 1% on all 3 datasets. We note that this accounts for around a half of our overall improvement. Although sample re-weighting performs similar as mixture ratio, it significantly reduces the amount of training time as it eliminates the need for a grid searching the best ratio. Kendal et al., (2018) use task uncertainty to weight tasks differently for MTL; our experiments show that this has some positive effect, but does not perform as well as our proposed two techniques. We note that Kendal et al. (as well as other previous MTL methods) optimizes the network to perform well for all the tasks, whereas our method focuses on the target domain which we are interested in, e.g., SQuAD.

Sensitivity of mixture ratio. We also investigate the effect of mixture ratio on the model performance. We plot the EM/F1 score on SQuAD dev set vs. mixture ratio in Figure 1 for MT-SAN when trained on all three datasets. The curve peaks at $\alpha = 0.4$; however if we use $\alpha = 0.2$ or $\alpha = 0.5$, the performance drops by around 0.5%, well behind the performance of sample re-weighting. This shows that the performance of MT-SAN is sensitive to changes in α, making the hyperparameter search even more difficult. Such sensitivity suggests a preference for using our sample re-weighting technique. On the other hand, the ratio

	Samples/Groups	CED′	H_Q	H_A
Examples	(NewsQA) Q: Where is the drought hitting? A: Argentina	0.824	0.732	0.951
	(MARCO) Q: thoracic cavity definition A: is the chamber of the human body ... and fascia.	0.265	0.332	0.240
Averages	Samples in NewsQA	0.710	0.593	0.895
	Samples in MARCO	0.587	0.550	0.669
	MARCO Questions that start with “When” or “Who”	0.662	0.605	0.761
	All samples	0.654	0.573	0.791

Table 8: Scores for examples from NewsQA and MS MARCO and average scores for specific groups of samples. CED′ is as in (7), while H_Q and H_A are normalized version of question and sample scores. “Sum” are the actual scores we use, and “LM”, “Answer” are scores from language models and answer lengths.

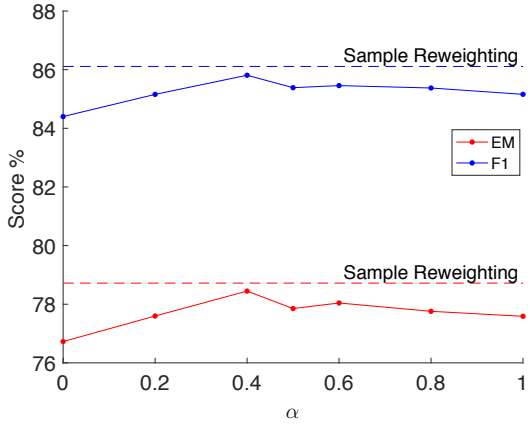

Figure 1: Effect of the mixture ratio on the performance of MT-SAN. Note that $\alpha = 0$ is equivalent to single task learning, and $\alpha = 1$ is equivalent to simple combining.

based approach is pretty straightforward to implement.

Analysis of sample weights. Dataset comparisons in Table 1 and performance in Table 2 suggests that NewsQA share more similarity with SQuAD than MARCO. Therefore, a MTL system should weight NewsQA samples more than MARCO samples for higher performance. We try to verify this in Table 8 by showing examples and statistics of the sample weights. We present the CED′ scores, as well as normalized version of question and answer scores (resp. $(H'_{1,Q} - H'_{k,Q})$ and $(H'_{1,A} - H'_{k,A})$ in (6), and then negated and normalized over all samples in NewsQA and MARCO in the same way as in (7)). A high H_Q score indicates high importance of the question, and H_A of the answer; CED′ is a summary of the two. We first show one example from NewsQA and one from MARCO. The NewsQA question is a natural question (similar to SQuAD) with a short answer, leading to high scores both in questions and answers. The MARCO question is a phrase, with a very long answer, leading to lower scores. From overall statistics, we also find samples in NewsQA have a higher score than those in MARCO. However, if we look at MARCO questions that start with “when” or “who” (i.e., probability natural questions with short answers), the scores go up dramatically.

6 Conclusion

We proposed a multi-task learning framework to train MRC systems using datasets from different domains and developed two approaches to re-weight the samples for multi-task learning on MRC tasks. Empirical results demonstrated our approaches outperform existing MTL methods and the single-task baselines as well. Interesting future directions include combining with larger language models such as BERT, and MTL with broader tasks such as language inference (Liu et al., 2019) and machine translation.

Acknowledgements

Yichong Xu has been partially supported by DARPA (FA8750-17-2-0130).

References

Amittai Axelrod, Xiaodong He, and Jianfeng Gao. 2011. Domain adaptation via pseudo in-domain data selection. In *Proceedings of the conference on empirical methods in natural language processing*, pages 355–362. Association for Computational Linguistics.

Rich Caruana. 1997. Multitask learning. *Machine learning*, 28(1):41–75.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1870–1879.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

David Golub, Po-Sen Huang, Xiaodong He, and Li Deng. 2017. Two-stage synthesis networks for transfer learning in machine comprehension. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 835–844.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Minghao Hu, Yuxing Peng, Zhen Huang, Nan Yang, Ming Zhou, et al. 2018. Read+ verify: Machine reading comprehension with unanswerable questions. *arXiv preprint arXiv:1808.05759*.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*.

Rudolf Kadlec, Martin Schmid, Ondřej Bajgar, and Jan Kleindienst. 2016. 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)*, volume 1, pages 908–918.

Alex Kendall, Yarin Gal, and Roberto Cipolla. 2018. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7482–7491.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Souvik Kundu and Hwee Tou Ng. 2018. A question-focused multi-factor attention network for question answering. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*.

Xiaodong Liu, Kevin Duh, and Jianfeng Gao. 2018a. Stochastic answer networks for natural language inference. *arXiv preprint arXiv:1804.07888*.

Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang. 2015. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 912–921.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding. *arXiv preprint arXiv:1901.11504*.

Xiaodong Liu, Wei Li, Yuwei Fang, Aerin Kim, Kevin Duh, and Jianfeng Gao. 2018b. Stochastic answer networks for squad 2.0. *arXiv preprint arXiv:1809.09194*.

Xiaodong Liu, Yelong Shen, Kevin Duh, and Jianfeng Gao. 2018c. Stochastic answer networks for machine reading comprehension. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1694–1704. Association for Computational Linguistics.

Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2015. Multi-task sequence to sequence learning. *arXiv preprint arXiv:1511.06114*.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*, pages 6294–6305.

Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2018. The natural language decathlon: Multitask learning as question answering. *arXiv preprint arXiv:1806.08730*.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2017. Regularizing and Optimizing LSTM Language Models. *arXiv preprint arXiv:1708.02182*.

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. *arXiv preprint arXiv:1611.09268*.

Takeshi Onishi, Hai Wang, Mohit Bansal, Kevin Gimpel, and David McAllester. 2016. Who did what:

A large-scale person-centered cloze dataset. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2230–2235.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 2227–2237.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you dont know: Unanswerable questions for squad. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 784–789.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

Yelong Shen, Xiaodong Liu, Kevin Duh, and Jianfeng Gao. 2017. An empirical analysis of multiple-turn reasoning strategies in reading comprehension tasks. *arXiv preprint arXiv:1711.03230*.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *arXiv preprint arXiv:1505.00387*.

Saku Sugawara, Yusuke Kido, Hikaru Yokono, and Akiko Aizawa. 2017. Evaluation metrics for machine reading comprehension: Prerequisite skills and readability. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 806–817.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2017. Newsqa: A machine comprehension dataset. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 191–200.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Wei Wang, Ming Yan, and Chen Wu. 2018a. Multi-granularity hierarchical attention fusion networks for reading comprehension and question answering. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1705–1714.

Yizhong Wang, Kai Liu, Jing Liu, Wei He, Yajuan Lyu, Hua Wu, Sujian Li, and Haifeng Wang. 2018b. Multi-passage machine reading comprehension with cross-passage answer verification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1918–1927.

Marlies van der Wees, Arianna Bisazza, and Christof Monz. 2017. Dynamic data selection for neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1400–1410.

Dirk Weissenborn, Georg Wiese, and Laura Seiffe. 2017. Making neural qa as simple as possible but not simpler. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 271–280.

Yichong Xu, Jingjing Liu, Jianfeng Gao, Yelong Shen, and Xiaodong Liu. 2017. Dynamic fusion networks for machine reading comprehension. *arXiv preprint arXiv:1711.04964*.

Zhilin Yang, Bhuwan Dhingra, Ye Yuan, Junjie Hu, William W Cohen, and Ruslan Salakhutdinov. 2016. Words or characters? fine-grained gating for reading comprehension. *arXiv preprint arXiv:1611.01724*.

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. *arXiv preprint arXiv:1804.09541*.

A Answer Module for WDW

We describe the answer module for WDW here for completeness. For WDW we need to choose an answer from a list of candidates; the candidates are people names that have appeared in the passage. We use the same way to summary information in questions as in span-based models: $s_0 \leftarrow \text{Highway}\left(\sum_j \frac{\exp(w_4 H_j^q)}{\sum_{j'} \exp w_4 H_{j'}^q} \cdot H_j^q\right)$. We then compute an attention score via simple dot product: $s = \text{softmax}(s_0^T M)$. The probability of a candidate being the true answer is the aggregation of attention scores for all appearances of the candidate:

$$\Pr(c|Q,P) \propto \sum_{1 \le i \le n} s_i \mathbb{I}(p_i \in C)$$

Setup	SQuAD (v1)	SQuAD (v2)	NewsQA	WDW
Single Dataset	69.5,78.8 (paper) 68.6, 77.8 (ours)	61.9, 65.2	51.9, 64.6	**75.8**
MT-DrQA on Sv1+NA	70.2, 79.3	-,-	52.8, 65.8	-
MT-DrQA on Sv1+W	69.2, 78.4	-,-	-,-	75.7
MT-DrQA on Sv1+N+W	**70.2, 79.3**	-,-	**53.1, 65.7**	75.4
MT-DrQA on Sv2+N	-,-	**63.6, 66.7**	52.7, 65.7	-
MT-DrQA on Sv2+W	-,-	63.5, 66.3	-,-	75.4
MT-DrQA on Sv2+N+W	-,-	63.1, 66.3	52.5, 65.6	75.3
SOTA (Single Model)	80.0, 87.0	72.3, 74.8	48.4, 63.7 (test)	71.7 (test)
MT-DrQA Best Performance	**70.2, 79.3**	**63.6, 66.7**	**53.0, 66.2(test)**	**75.4 (test)**
Human Performance (test set)	82.3, 91.2	86.8, 89.5	46.5, 69.4	84

Table 9: Single model performance of our method to train DrQA on multi-task setting, as well as state-of-the-art (SOTA) results and human performance. SQuAD and NewsQA performance are measured by (EM, F1), and WDW by accuracy percentage. All results are on development set unless otherwise noted. Published SOTA results come from (Wang et al., 2018a; Hu et al., 2018; Kundu and Ng, 2018; Yang et al., 2016) respectively.

for each candidate C. Recall that n is the length of passage P, and p_i is the i-th word; therefore $\mathbb{I}(p_i \in C)$ is the indicator function of p_i appears in candidate C. The candidate with the largest probability is chosen as the predicted answer.

B Experiment Results on DrQA

To demonstrate the flexibility of our approach, we also adapt DrQA (Chen et al., 2017) into our MTL framework. We only test DrQA using the basic Algorithm 2, since our goal is mainly to test the MTL framework.

B.1 Model Architecture

Similar to MT-SAN, we add a highway network after the lexicon encoding layer and the contextual encoding layer and use a different answer module for each dataset. We apply MT-DrQA to a broader range of datasets. For span-detection datasets such as SQuAD, we use the same answer module as DrQA. For cloze-style datasets like Who-Did-What, we use the attention-sum reader (Kadlec et al., 2016) as the answer module. For classification tasks required by SQuAD v2.0 (Rajpurkar et al., 2018), we apply a softmax to the last state in the memory layer and use it as the prediction.

B.2 Performance of MT-DrQA

We apply MT-DrQA to SQuAD (v1.1 and v2.0), NewsQA, and WDW. We follow the setup of (Chen et al., 2017) for model architecture and hyperparameter setup. We use Algorithm 1 to train all MT-DrQA models. Different than (Rajpurkar et al., 2018), we do not optimize the evaluation score by changing the threshold to predict unanswerable question for SQuAD v2.0; we just use the argmax prediction. As a result, we expect the gap between dev and test performance to be lower for our model. The results of MT-DrQA are presented in Table 9. The results of combining SQuAD and NewsQA obtain similar performance boost as our SAN experiment, with a performance boost between 1-2% in both EM and F1 for the two datasets. The results of MTL including WDW is different: although adding WDW to SQuAD still brings a marginal performance boost to SQuAD, the performance on WDW drops after we add SQuAD and NewsQA into the training process. We conjecture that this negative transfer phenomenon is probably because of the drastic difference between WDW and SQuAD/NewsQA, both in their domain, answer type, and task type; and DrQA might not be capable of caputuring all these features using just one network. We leave the problem of further preventing such negative transfer to future work.