Multi-Task Learning using Dynamic Task Weighting for Conversational Question Answering

Sarawoot Kongyoung
University of Glasgow
s.kongyoung.1@research.gla.ac.uk

Craig Macdonald, Iadh Ounis
University of Glasgow
{*first.lastname*}@glasgow.ac.uk

Abstract

Conversational Question Answering (ConvQA) is a Conversational Search task in a simplified setting, where an answer must be extracted from a given passage. Neural language models, such as BERT, fine-tuned on large-scale ConvQA datasets such as CoQA and QuAC have been used to address this task. Recently, Multi-Task Learning (MTL) has emerged as a particularly interesting approach for developing ConvQA models, where the objective is to enhance the performance of a primary task by sharing the learned structure across several related *auxiliary* tasks. However, existing ConvQA models that leverage MTL have not investigated the *dynamic* adjustment of the relative importance of the different tasks during learning, nor the resulting impact on the performance of the learned models. In this paper, we first study the effectiveness and efficiency of dynamic MTL methods including Evolving Weighting, Uncertainty Weighting, and Loss-Balanced Task Weighting, compared to *static* MTL methods such as the uniform weighting of tasks. Furthermore, we propose a novel hybrid dynamic method combining Abridged Linear for the main task with a Loss-Balanced Task Weighting (LBTW) for the auxiliary tasks, so as to automatically fine-tune task weighting during learning, ensuring that each of the tasks' weights is adjusted by the relative importance of the different tasks. We conduct experiments using QuAC, a large-scale ConvQA dataset. Our results demonstrate the effectiveness of our proposed method, which significantly outperforms both the single-task learning and static task weighting methods with improvements ranging from +2.72% to +3.20% in F1 scores. Finally, our findings show that the performance of using MTL in developing ConvQA model is sensitive to the correct selection of the auxiliary tasks as well as to an adequate balancing of the loss rates of these tasks during training by using LBTW.

1 Introduction

The task of Conversational Question Answering (ConvQA), which consists in answering a question from a given passage in the form of a dialogue has become a vital task for Machine Reading Comprehension (MRC). In the ConvQA task, in order to predict an answer, the system needs to extract text spans from a given passage and understand the question based on the given conversational history. Recently, the advancement in neural language modeling such as BERT (Devlin et al., 2019), and the introduction of two large-scale datasets, namely CoQA (Reddy et al., 2019) and QuAC (Choi et al., 2018) have further boosted research in the ConvQA task. In particular, QuAC introduces a main task, namely Answer Span prediction, which consists in answering a question by extracting text spans from a given passage as well as some *auxiliary* tasks, namely Yes/No prediction, Follow up prediction and Unanswerable prediction. Recently, Multi-Task Learning (MTL), which is a way to learn multiple different but related tasks simultaneously, has emerged as a popular solution to tackle all these tasks in a uniform model (Qu et al., 2019b). MTL can also be used to leverage the auxiliary tasks to improve the performance of a system on the main task. For example, for the QuAC dataset, (Qu et al., 2019b; Yeh and Chen, 2019) adopted an MTL approach that learns the *auxiliary* tasks and the main task by sharing the encoder, and showed an improvement in the used ConvQA model.

MTL methods can be categorised into static or dynamic methods. In the static MTL methods, each of the task's weights used to combine the loss functions of the various used tasks during training are unchanged throughout the learning phase, which may divert training resources to unnecessary tasks. In contrast, in the dynamic MTL methods, each of the task's weights are adjusted automatically

Proceedings of the 5th International Workshop on Search-Oriented Conversational AI (SCAI), pages 17–26

ISBN 978-1-952148-67-5

to balance the loss rate (Liu et al., 2019a) or to balance the weights across tasks (Kendall et al.). However, the implementation of a MTL dynamic method is more complicated and has a lower training efficiency than static methods.

Most existing (Qu et al., 2019b; Yeh and Chen, 2019) Conversational Question Answering (ConvQA) models that leverage Multi-Task Learning (MTL) use the static method with unchanged tasks' weights during the training epochs. For instance, the recently-proposed History Attention Mechanism (HAM) model (Qu et al., 2019b) attempted to apply Multi-Task Learning in order to improve the effectiveness of conversational QA. However, the tasks' weights in the model were unchanged during the training state and emphasise the main task. FlowDelta (Yeh and Chen, 2019) is a ConvQA model that also employed a static MTL method, which sets all tasks' weights equal to one. In the static MTL methods used in HAM and FlowDelta, all of the tasks' weights have not been adjusted throughout the learning phase. As a result, training resources could be diverted to unnecessary tasks with a negative impact on the performance of the learned models. To improve the effectiveness of Multi-Task Learning for Conversational Question Answering, we propose a novel method, called Hybrid Task Weighting, which focuses on adjusting the tasks' weights by modelling the difference between the tasks' weights, while still prioritising on the main task.

Our contributions are summarised as follows: (1) We leverage dynamic Multi-Task Learning with BERT[1] to effectively address the task of learning Answer Span prediction with its auxiliary tasks including Yes/No prediction, Follow up prediction, and Unanswerable prediction; (2) To further enhance the performance of Multi-Task Learning, we introduce a hybrid strategy, which automatically fine-tunes the multiple tasks' weights along the learning steps. Our method uses Abridged Linear for the primary task and Loss-Balanced Task Weighting for the auxiliary tasks; (3) The proposed hybrid method yields the best performance improvements over the baselines on the QuAC dataset.

2 Related Work

In the following, we discuss related work about Multi-Task Learning, Conversational Question Answering, and using Multi-Task Learning in Conversational Question Answering.

Multi-Task Learning: MTL is a learning paradigm, which has achieved success in many machine learning applications, including Natural Language Processing (Liu et al., 2015, 2019b), Speech Processing (Hu et al., 2015; Wu et al., 2015), and Computer Vision (Leang et al., 2020). For further background on MTL, we refer the readers to the recent review by (Ruder, 2017; Zhang and Yang, 2017). The MTL methods can be classified into static methods or dynamic methods based on their weighting strategy (Ming et al., 2019). In a static method, before training the network, each of the task's weights is set manually, then these weights are fixed throughout the training of the network (Qu et al., 2019b; Yeh and Chen, 2019). In contrast, the dynamic methods initialise each of the task's weights at the beginning of the training and automatically update the weights during the training process (Belharbi et al., 2016; Chen et al., 2018; Liu et al., 2019a). Typically, the MTL networks can be classified into either *hard* or *soft* parameter sharing networks (Ruder, 2017). In hard parameter sharing, also known as multi-head, the network is applied by employing separate task-specific output layers on top of a shared encoder. In soft parameter sharing, all parameters are task-specific but all networks have mechanisms to handle the cross-task learning. Following (Liu et al., 2015, 2019b; Xu et al., 2019), we employ a hard parameter sharing MTL approach because this network type reduces the risk of overfitting (Ruder, 2017).

Conversational Question Answering: ConvQA is a Machine Reading Comprehension (MRC) task where questions are formed in conversations. Hence, a ConvQA approach needs to deal with the conversation history to accurately understand and answer the current question. To handle the conversation history, existing works (Zhu et al., 2018; Reddy et al., 2019) prepended previous questions and answers to the current question while (Qu et al., 2019a,b; Yeh and Chen, 2019) employed a history selection mechanism. Some prior studies have integrated the conversation history into neural language models such as BERT: For example, Qu et al. (2019b) proposed the Positional History Answer Embedding (PosHAE) approach, which uses a feature vector to encode the position of the answer in the conversation history in the current question; Similarly, Choi et al. (2018); Yeh and Chen (2019)

[1] Our preliminary experiments found BERT to be more effective than ALBERT or RoBERTa.

used a Context Feature to mark historical answers in the passage.

The two recent large-scale ConvQA datasets, QuAC (Choi et al., 2018) and CoQA (Reddy et al., 2019), have facilitated further research on this task. The differences between these datasets are that the questions in CoQA are predominantly factoid in nature, while most questions in QuAC are non-factoid. Moreover, QuAC also contains three auxiliary tasks; in contrast, CoQA only provides an Unanswerable prediction task as an auxiliary task. Hence, due to the presence of multiple auxiliary tasks, our MTL study focuses on the QuAC dataset.

Multi-Task Learning for Conversational Question Answering Models: Recently, existing works (Qu et al., 2019b; Yeh and Chen, 2019) on MTL for ConvQA have successfully adopted static MTL methods. However, there is still room for improvement since during the learning phase the weights for the auxiliary tasks are unchanged and therefore they not adjusted relative to the importance of the different tasks. We include these MTL methods as baselines in our present work.

Instead, in this paper, we take advantage of a dynamic method in MTL for ConvQA. The goal of our proposed model is to improve the effectiveness of the ConvQA task. As far we know, no prior work has addressed the use of dynamic MTL methods for the ConvQA task. In our proposed MTL approach, we employ the Abridged Linear (Belharbi et al., 2016) for the primary task and the Loss-Balanced Task Weighting (Liu et al., 2019a) for the auxiliary tasks, which prioritises the primary task after step t during training by setting the task's weight to one while also automatically fine-tuning the tasks' weights by balancing the loss ratio of the auxiliary tasks. In our model, we employ BERT (Devlin et al., 2019), which is still a widely used and popular pre-trained model, with customised features following (Qu et al., 2019b; Yeh and Chen, 2019). In the following section, we describe in detail our ConvQA model.

3 The BERT ConvQA Model

We first define the task in Section 3.1. An overview of the proposed ConvQA model is provided in Section 3.2. Section 3.3 describes how additional features are integrated with the BERT encoder. Then we explain how predictions are made for the main Answer Span prediction task as well as the auxiliary tasks in Sections 3.4 & 3.5, respectively.

Table 1: An example dialog from the ConvQA dataset.

p	In 1934 he batted .344 with 18 home runs, 104 RBI, 102 runs scored and 192 hits in 138 games. After a disappointing final season with the White Sox which saw Simmons bat just .267 with 16 home runs and 79 RBI in 128 game (first time in his 11-year career he did not reach .300+ & 100 RBI) he rebounded by hitting .327 with 13 home runs, 112 RBI and 96 runs scored in 1936 for the Detroit Tigers. In 1937 he struggled again, this time with the Washington Senators, batting just .279 with 8 home runs and 84 RBIs in 103 games. He rebounded with a stellar season in 1938, batting .302 with 21 home runs and 95 RBI in just 125 games for Washington. His 21 home runs that year gave Simmons the distinction of being the first player to hit 20 home runs in a year for the Senators. CANNOTANSWER
q_1	Where was he playing in 1933?
a_1	CANNOTANSWER
q_2	What did he do between 1933 and 1938?
a_2	In 1934 he batted .344 with 18 home runs, 104 RBI, 102 runs scored and 192 hits in 138 games.
q_3	Did he lead the league in hitting?
a_3	After a disappointing final season with the White Sox

3.1 Task Definition

Following Choi et al. (2018), we describe the ConvQA task as follows: given a passage p, a conversation history H_k consisting of a list of k questions and ground truth answer pairs, i.e. $H_k = [\langle q, a \rangle]$, and a new query q_{k+1}, the task is to predict answer a_{k+1} by predicting answer span indices i, j within passage p. Table 1 exemplifies the ConvQA task, showing an example passage p, and a history of length $k = 2$ with corresponding questions and answers; In particular, in response to question q_3, the aim of a ConvQA system is to correctly predict the right answer a_3 from all possible sentences in p.

Moreover, as mentioned in Section 2, the QuAC dataset (Choi et al., 2018) provides labels for auxiliary tasks that are relevant to the ConvQA task, namely the affirmation (Yes/No) and continuation (Follow up) classification tasks. For example, Yeh and Chen (2019) showed how to leverage the unanswerable questions as another auxiliary task called Unanswerable prediction. In the next section, we provide an overview of a BERT-based model that can be used for the ConvQA task; Moreover as an MTL model, it can benefit from learning using the auxiliary tasks. Later, in Section 4, we describe different MTL methods for weighting the ConvQA and auxiliary tasks during learning, which we apply and evaluate. We describe all auxiliary tasks in detail in Section 6.1.

3.2 Model Overview

To tackle the tasks described in Section 3.1, we present our ConvQA model by adopting a Multi-Task Learning approach. Figure 1 illustrates the architecture of our model, which consists of three components: an encoder, an answer span predictor and the auxiliary tasks predictor. For the encoder, we deploy a BERT model that encodes the question q_{k+1}, the passage p, and the conversation history H_k as a sequence of m words $C = \{c1, c2, ..., cm\}$

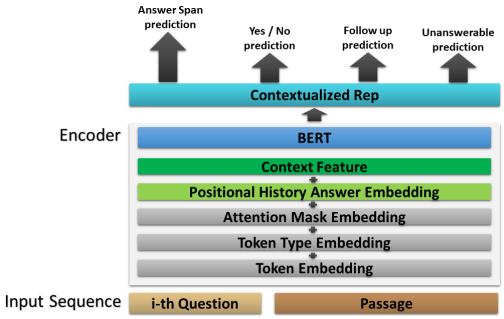

Figure 1: The model architecture.

into contextualised token-level representations i.e., $\hat{T}_k = F(q_{k+1}, p, H_k)$, where $F(\cdot)$ is BERT's encoder transformation function. These encodings are customised to the task by integrating conversation history features (Section 3.3). Finally, these representations are fed into the predictors' modules in a Multi-Task Learning setting (Sections 3.4 & 3.5).

3.3 BERT Encoder Features

In our model, we modify the BERT input to encapsulate two features – the Positional History Answer Embedding (PosHAE) and the Context Features:

PosHAE: We use this modification feature introduced by Qu et al. (2019b) to capture the conversation history into BERT. As exemplified by the example in Table 1, questions in the QuAC dataset often refer to entities in the previous answer(s). Consequently, PosHAE was introduced to embed the relative position of the terms that occur in previous answers within the conversational history H_k.

Context Feature: We integrate contextual knowledge of the previous answer within the passage into BERT by following Yeh and Chen (2019) who, applied BiDAF++ (Choi et al., 2018). Indeed, BiDAF++ learns a passage embedding that denotes whether a token in a recent answer is part of passage p.

3.4 Answer Span prediction

Given the token-level representation $\hat{T}_k$ produced by BERT, we compute the probability of each token being the start token or the end token in order to predict the answer span. In particular, to map a token representation $\hat{T}_k$ to a logit, two sets of parameters are learned for the start vector and the end vector, respectively. After that the softmax function is applied to obtain probabilities across all tokens in the sequence C (see Section 3.2). From this, we obtain p_m^S, and p_m^E, which are the probabilities of token m being the start token or end token, as follows:

$$p_m^S = \text{softmax}(\hat{T}_k(m)), p_m^E = \text{softmax}(\hat{T}_k(m)). \quad (1)$$

Then for the Answer Span prediction task, we compute the cross-entropy loss as follows:

$$\mathcal{L}_S = -\sum_M \mathbb{1}\{m = m_S\} \log(p_m^S),$$
$$\mathcal{L}_E = -\sum_M \mathbb{1}\{m = m_E\} \log(p_m^E),$$
$$\mathcal{L}_{ans} = \frac{1}{2}(\mathcal{L}_S + \mathcal{L}_E) \quad (2)$$

where the ground truth of the start token and end token are m_S and m_E, respectively, and $\mathbb{1}\{\cdot\}$ is and indicator function to show that the predicted token m is in the ground truth. Then the loss of the Answer Span prediction $\mathcal{L}_{ans}$ is calculated by averaging the loss of the start and end tokens, $\mathcal{L}_S$ and $\mathcal{L}_E$.

3.5 Auxiliary Task Prediction

All auxiliary tasks in our datasets are formulated as binary or multi-label classification tasks. To address each auxiliary task, we take the sequence-level representation $\hat{s}_k$ that is obtained from the $[CLS]$ token (which is the first token of the sequence, produced by BERT). We apply a softmax function on $\hat{s}_k$ to compute the posterior probabilities across the true and false labels for the multi-label tasks; for the binary tasks, we use a sigmoid function. After that, we compute cross-entropy loss for the multi-label tasks and the binary cross-entropy loss functions for the binary tasks. Next, we describe the MTL approaches to combine the loss functions from the auxiliary tasks with the loss calculated on the main task.

4 Multi-Task Learning for ConvQA

We now describe and categorise existing loss weighting approaches as either *static* or *dynamic*, depending on whether the importance they place on the loss of each task during learning is fixed or varied. In the following, we describe existing static and dynamic approaches (Sections 4.1 & 4.2), before describing our hybrid approach (Section 4.3).

4.1 Static MTL

Static MTL methods, which are the most frequently used MTL approaches for ConvQA, apply a fixed weighting of the different loss functions of the auxiliary tasks throughout the training process. This

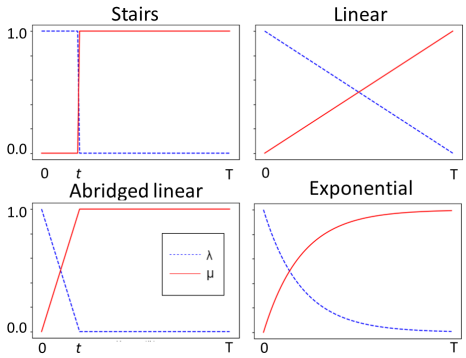

Figure 2: Dynamic Evolving Weighting approaches.

strategy is simple but yet expensive to fine-tune. Instead, many previous studies just report the use of uniform weights for tasks, such as setting all of them to 1.0 (Yeh and Chen, 2019), or setting their sum to 1 (Qu et al., 2019b). The total loss function of this method is defined as follows:

$$\mathcal{L}_{total} = \mu\mathcal{L}_{ans} + \lambda \sum_{a \in A} \mathcal{L}_a \tag{3}$$

where A is the set of auxiliary tasks, μ is the weight for the main task and λ is the weight for A.

4.2 Dynamic MTL

Applying static weighting to the auxiliary tasks can unnecessarily apply learning resources to the auxiliary tasks, instead of the main task. Indeed, this can lead to an overfitting to the wrong task and hence to underfitting on the main task (Chen et al., 2018). On the other hand, in the dynamic MTL approaches, the loss weighting of the tasks is instead continually adjusted during learning. Examples of dynamic approaches are Evolving Weighting (Belharbi et al., 2016), Loss-Balanced Task Weighting (Liu et al., 2019a), and Uncertainty Weighting (Kendall et al.), discussed further below.

Evolving Weighting: Belharbi et al. (2016) proposed to evolve the loss weighting during the training steps according to a *schedule*. A training step is defined as the number of batches of the training data, such that the total number of steps is the number of batches multiplied by the number of training epochs. Four different schedules were proposed. Figure 2 gives an overview of how the four schedules vary the weights of the main and auxiliary tasks – μ and λ, respectively – across the training steps. These four schedules are described below:

Stairs schedule: The initial emphasis is on the auxiliary task, with $\mu = 0$ and $\lambda = 1$. At a given training step t, $\mu = 1$ and $\lambda = 0$.

Algorithm 1: Hybrid Task Weighting

```
Given S tasks and parameter α.
Initialise neural network weights W.
for each epoch i do
    for each batch of data B do
        Get the loss on each task ℓ_B ∈ R^S
        Store the first batch loss as ℓ_(0,i) ∈ R^S
        if step t ≤ t_τ
            Set the main task weight μ = (t/T)
        else
            Set the main task weight μ = 1
        for each auxiliary task s do
            Set the auxiliary task weight λ = (ℓ_(B,s) / ℓ_(0,i,s))^α
            Update weighted loss ℓ_(B,s) = ℓ_(B,s) × λ
        Update weighted loss ℓ_(B,m) = ℓ_(B,m) × μ
        Set the total loss ℓ_total = ℓ_(B,m) + Σ_{i=1}^{s} ℓ_i
```

Linear schedule: The weight of the auxiliary task decreases linearly at each training step, such that the auxiliary weight $\lambda = 1$ tends to 0; in contrast, the weight of the main task increases linearly, i.e. $\lambda = (1 - \mu)$. In particular, given that the total number of steps T is known in advance, $\lambda_t = \frac{t}{\mathrm{T}}$.

Abridged Linear schedule: In a linear schedule, μ rises over the full training schedule to step T. This may not place sufficient emphasis on the main task during training. Instead, in the Abridged Linear schedule the weight on the auxiliary task λ falls linearly to 0 by a threshold step t_τ. After t_τ, all emphasis is on the main task (i.e. $\mu = 1$).

Exponential schedule: The weights evolve exponentially to the step number, i.e. $\mu = exp(\frac{-t}{\sigma})$, where t is the current number of training steps, and σ is the slope, as shown in Figure 2.

Loss-Balanced Task Weighting (LBTW) (Liu et al., 2019a): This MTL method aims to reduce *negative transfer* by using the task-specific loss to balance the different auxiliary tasks. Negative transfer is when the performance of the task is decreased by Multi-Task Learning compared to the single-task learning. This method employs the *loss ratio* between the current loss and the initial loss of each task to adjust the task's weight. The task with the loss ratio closest to one needs to contribute more to the total loss. By increasing the weight of the task with loss ratio that is closest to one, this method attempts to *balance* the task importances.

Uncertainty Weighting (Kendall et al.): This method is the most often used Multi-Task Learning approach, which is a weighting strategy that consists in analysing the uncertainty of each task. In this method, each of the task's weights is adjusted by deriving a multi-task loss function when maximising the Gaussian likelihood (Ruder, 2017).

4.3 Hybrid Task Weighting

Among the existing dynamic MTL methods, Uncertainty Weighting (Kendall et al.), and Loss-Balanced Task Weighting (Liu et al., 2019a) both weight all tasks without prioritising on the main task, such that resources are unnecessarily allocated to other tasks, thereby leading to a possible underfitting on the main task (Guo et al.). For this reason, we propose a Hybrid Task Weighting approach, which applies an Abridged Linear schedule for weighting the main task and LBTW (Liu et al., 2019a) for weighting the auxiliary tasks. In particular, for the Abridged Linear schedule, we take a step threshold $t_\tau = \mathrm{T}/10$, i.e. 10% of all steps, which is the same as the warm-up ratio we use (see Section 6.4 for further details). To apply LBTW for the auxiliary tasks, a hyperparameter α is used to balance the influence of the task-specific weights, i.e. α=0.5 (Liu et al., 2019a). For each batch, the weight of each task is calculated by using the loss ratio between the loss at step t and the loss at t=0, thereby balancing the loss rates of the auxiliary tasks. Algorithm 1 provides further details about the implementation of our hybrid approach.

5 Research Questions

In this paper, we address two key research questions. Firstly, one of our central contributions is the comparison of existing Multi-Task Learning (MTL) strategies, when used in the same Conversational Question Answering (ConvQA) model both in terms of effectiveness and efficiency. By doing this, we investigate whether there is an actual difference between the static and dynamic loss weighting methods, in guiding the learning process. Moreover, to the best of our knowledge, there has been no previous study that investigated dynamic loss weighting for the ConvQA task on the QuAC dataset. Hence, our first research question is:

RQ1: What is the most effective and efficient Multi-Task Learning method for ConvQA?

Secondly, we investigate the effectiveness of the combination of the auxiliary tasks to improve the performance of the main QuAC task, namely we posit the following research question:

RQ2: Does applying the proposed MTL ConvQA model using each of the auxiliary tasks result in effectiveness improvements over learning using only the main task?

6 Experimental Setup

In this section, we describe the used dataset, QuAC, and its auxiliary tasks in Section 6.1. We present the list of our baselines in Section 6.2. We discuss the used evaluation metrics in Section 6.3, and the applied hyper-parameter settings in Section 6.4.

6.1 Dataset

To conduct our evaluation of the MLT methods when integrated into the BERT ConvQA model, we choose QuAC (Choi et al., 2018), a large-scale dataset for ConvQA over passages extracted from Wikipedia articles. Unlike other Machine Reading datasets such as SQuAD (Rajpurkar et al., 2016, 2018), this dataset is considered to be a multi-turn dataset where the questions and answers simulate conversations. The main reason for choosing this dataset for our experiments is that it provides not only an Answer span prediction as the main task but it also provides other auxiliary tasks namely, the affirmation (Yes/No prediction) and continuation (Follow up prediction) classification tasks. Moreover, we also observe that if an answer in QuAC is tagged as CANNOTANSWER, then this means that the corresponding question cannot be answered. Hence, from these kind of answers, we define another Unanswerable prediction task as an additional auxiliary task to use in our MTL method. We describe below each of the used auxiliary tasks:

Yes/No prediction: This task consists of three possible labels: yes, no, neither where yes or no are represented as the sought answer to this question type; otherwise it will be 'neither'. Choi et al. (2018) observed that there were 25.8% of yes/no questions in the QuAC dataset.

Follow up prediction: This classification task consists in predicting the continuation of a given question, and has three possible labels: follow up, maybe follow up, don't follow up.

Unanswerable prediction: This task has two possible labels: yes/no allocated by inspecting the answer text associated to each question in the dataset. If the answer text is CANNOTANSWER, the label is yes otherwise it is no. 20.2% of all questions in the QuAC dataset are unanswerable.

6.2 Baselines

We use as baselines all methods described in Section 4. Hence, our baselines consist of the Static MTL methods from Section 4.1, namely *sum to 1* and *equal to 1*, and the dynamic MTL meth-

ods from Section 4.2, namely Evolving Weighting (Stair, Linear, Abridged Linear, and Exponential), Loss-Balanced Task Weighting, and Uncertainty Weighting as baselines. In addition, we also include Single-Task Learning as a baseline to illustrate the effectiveness of Multi-Task Learning as well as our proposed Hybrid Task Weighting method.

6.3 Evaluation Metrics.

Since we are using the QuAC dataset, we naturally adopt the two evaluation metrics in the corresponding challenge, which consist of the word-level F1, and the human equivalence score (HEQ). The word-level F1, commonly used in Machine Comprehension and in the ConvQA tasks (Rajpurkar et al., 2016, 2018; Choi et al., 2018), evaluates the overlap between the system's prediction and the ground truth answer span. Meanwhile, the HEQ metric is used to evaluate the percentage of examples for which the deployed model's F1 is equivalent to or higher than the human F1. This metric is composed of HEQ-Q, computed on the question level, and HEQ-D, computed at the dialogue level. The QuAC challenge defines the human performance to have an HEQ-Q and HEQ-D of 100%. Finally, we use the McNemar's test to measure statistical significance between the prediction performances.

6.4 Hyper-parameter Settings.

We implement all models using the Pytorch version of BERT from HuggingFace (Wolf et al., 2019), namely using the `bert-base-uncased`[2] model as our encoder. Following Qu et al. (2019b), the model configuration is as follows: the max sequence length is set to 12, the stride in the sliding window is set to 128, the max question length is set to 64, the max answer length set to 35, the number of training epochs is set to 5 and the batch size is set to 12. To train our BERT ConvQA model, we use the BertAdam weight decay optimiser, with an initial learning rate of 5e-5 while the learning rate warming up portion is 10%. For all our experiments, we use a single Nvidia TITAN RTX GPU.

7 Experimental Results

We first report our evaluation results for various MTL methods using our ConvQA model in Section 7.1. Our findings for the usefulness of the auxiliary tasks in MTL are detailed in Section 7.2.

[2] https://huggingface.co/transformers/pretrained_models.html

7.1 RQ1: Effectiveness and Efficiency of the MTL Methods

We investigate the performance of the baselines in comparison to our proposed hybrid method for Multi-Task Learning on the validation set[3] of the QuAC dataset. All MTL methods are trained on the provided QuAC training set by using all the auxiliary tasks, namely the Yes/No prediction, the Follow up prediction and the Unanswerable prediction classification tasks. In this section, we focus on the performance of the system on the main task (i.e. the Answer Span prediction task).

First, we examine the effectiveness of the MTL methods, including our proposed methods and those baselines listed in Section 4. Table 2 illustrates the single-task learning baseline (denoted STL) in the first column and the MTL methods in the following columns. Within Table 2, the best result in each row is highlighted in bold. From this table, we observe that the F1 performance of all the MTL methods is better than the STL baseline. Indeed, our proposed method, Hybrid Task Weighting, achieves the best F1 and HEQ-Q performances, at 72.28 and 68.71, respectively. The best reported HEQ-D score is achieved by the Exponential Evolving Weighting method at 13.1 followed by our Hybrid Task Weighting method at 13.0. Indeed, our proposed method is more effective than the Abridged Linear and the Loss-Balanced Task Weighting dynamic methods, showing that while it emphasises the main tasks (c.f. Abridged Linear), it also balances the auxiliary tasks through use of the LBTW method. Moreover, all of the dynamic task weighting methods significantly outperform the STL model, except for the Stair and Uncertainty Weighting methods (McNemar's test, $p < 0.05$).

Next, we investigate the efficiency of the tested MTL methods by comparing the average number of iterations per second needed during training and evaluation. Table 3 depicts the efficiency of the MTL methods for the BERT ConvQA model. In this table, the higher the number, the higher the efficiency, while the best result is highlighted in bold. We observe that the Linear Evolving Weighting yields the best efficiency in comparison to all other methods – at 2.31 iterations per second during learning – while the static task weighting method (equal to 1) exhibits the best evaluation efficiency

Overall the efficiency of most models during

[3] QuAC's test set is only accessible by submitting to their leaderboard. Hence, we provide results using the provided validation set.

Table 2: Effectiveness of various task-weighting methods for Conversational Question Answering. † denotes a result statistically different from that of our proposed Hybrid Task Weighting model (McNemar's test, $p < 0.05$); ‡ denotes a significant improvement over the STL baseline. The highest value for each measure is highlighted.

	Single-task learning	Static task weighting		Evolving Weighting				Loss-Balanced Task Weighting	Uncertainty Weighting	Hybrid Task Weighting
		Sum to 1	Eq.1	Linear	Exponential	Abridged Linear	Stair			
F1	69.08 †	69.56 †	69.40 †	70.97 †‡	71.16 †‡	71.37 †‡	69.73 †	69.41 †‡	69.20 †	**72.28** ‡
HEQ-Q	65.51	66.06	65.65	67.49	67.88	67.78	66.24	65.71	65.43	**68.71**
HEQ-D	11.6	11.10	10.7	12.8	**13.1**	12.10	11.80	11.6	11.3	13.00

Table 3: Efficiency of different MTL methods. The highest value for each phase is highlighted.

Model		#iters/sec	
		Training	Evaluating
Single-task learning		2.23	3.95
Static task weigthing	Sum to 1.	2.13	4.05
	All eq. 1	2.25	**4.12**
Evolving Weighting	Linear	**2.31**	3.99
	Exponential	2.26	3.90
	Abridged Linear	2.20	3.84
	Stair	2.15	4.07
Loss-Balanced Task Weighting		2.05	3.91
Uncertainty Weighting		1.5	3.94
Hybrid Task Weighting		2.04	4.04

Table 4: Comparison of different combinations of auxiliary tasks. † denotes a statistically significant improvement over STL with $p < 0.05$ using the McNemar's test. The highest value for each measure is highlighted.

	F1	HEQ-Q	HEQ-D
STL	69.08	65.51	11.6
Yes/No	68.60	64.98	11.6
Follow up	69.76 †	66.74	12.0
Unanswerable	72.27 †	68.87	13.5
Yes/No + Follow up	**72.66** †	68.87	11.7
Yes/No + Unanswerable	72.48 †	**69.02**	13.2
Follow up + Unanswerable	71.91 †	68.52	**14.2**
All	72.28 †	68.71	13.0

evaluation is fairly similar, at around 3.8 to 4.1 iterations per second. We argue that this is because during the evaluation phase, all models have the same structure, and only differ in terms of weights. On the other hand, during learning, the Evolving Weighting method is slightly faster than the other baseline methods including our own proposed method due to the simple manner in which it calculates the task weight. Moreover, training the ConvQA model using the Uncertainty Weighting method exhibits more training time than other methods. Indeed, this approach has the most complex implementation.

In response to RQ1, we find that our BERT ConvQA model learned through Multi-Task Learning by using a hybrid approach has the best effectiveness, yielding statistically significant improvements over the baselines. Moreover, we observe that there is little difference between the efficiency of our proposed method, and that of the static task weighting methods, or the single-task learning in both the training and evaluation phases even though our approach has a more complex implementation.

7.2 RQ2: Combination of Auxiliary Tasks vs. Single-Task Learning

Next, we conduct experiments to determine the best combination of auxiliary tasks, which helps to improve the performance of the main task. In these experiments, all models are learned by using our proposed method as the Multi-Task Learning strategy for the BERT ConvQA model. We vary the choice of auxiliary tasks from those detailed in Section 6.1, namely Yes/No prediction, Follow up prediction and Unanswerable prediction. Single-task learning acts as a baseline for these experiments.

Table 4 presents the effectiveness of the different combinations of auxiliary tasks (each row is a different combination). We observe that the highest scores for the F1, HEQ-Q and HEQ-D measures are not obtained from the same combination. In particular, applying Multi-Task Learning using the Yes/No and Follow up tasks achieves the best F1 performance compared to the other combinations. However, when using the HEQ-Q metric, it is apparent that the combination of the Yes/No prediction and Unanswerable prediction is the best. Furthermore, the combination of the Follow up prediction and Unanswerable prediction yields the best model in terms of the HEQ-D metric. From these results, we further analyse why the models that include Unanswerable prediction as one of the auxiliary tasks, have higher HEQ scores in comparison to models that use either the Yes/No prediction or the Follow up prediction as the auxiliary tasks. We found that a key issue is the number of correct predictions for the unanswerable questions. The more correct answers achieved on this type of questions, the more likely the performance will be higher in terms of HEQ. From the table, we also observe that the model that fused all the auxiliary tasks is not the best choice for MTL, and its performance on all metrics is similar to the model that used only

the Unanswerable prediction auxiliary task.

In answer to RQ2, we conclude that most of the combination models are better than just learning the main task, except the model that solely used the Yes/No prediction as an auxiliary task. This raises the question as to why the model that combines all the auxiliary tasks does not outperform the models that includes Unanswerable as an auxiliary task. We conjecture that negative transfer (see Section 4.2) might be a possible reason explaining the drop in the performance of MTL. We leave the investigation of this issue to future work.

8 Conclusions

We have proposed a method for Conversational Question Answering, which learns to predict the correct answer span, by applying Multi-Task Learning (MTL). Our proposed hybrid MTL method makes use of Evolving Weighting by Abridged Linear for learning the main task, while the auxiliary tasks are addressed using Loss-Balanced Task Weighting. Our experiments on the QuAC dataset demonstrated that our ConvQA model learned through Multi-Task Learning by using a hybrid approach has the best effectiveness, yielding statistically significant improvements over the baselines. Furthermore, we showed that the use of a combination of the auxiliary tasks resulted in an enhancement to the main task performance compared to single-task learning. For future work, we plan to consider the integration of a question re-writer as well as the use of an attention mechanism for capturing the dialog context.

References

Soufiane Belharbi, Romain Hérault, Clément Chatelain, and Sébastien Adam. 2016. Deep multi-task learning with evolving weights. In *Proc. ESANN*.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. 2018. GradNorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *Proc. ICML*, pages 794–803.

Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wentau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. 2018. QuAC: Question answering in context. In *Proc. EMNLP.*, page 2174–2184.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. *Proc. NAACL*, pages 4171–4186.

Michelle Guo, Albert Haque, De-An Huang, Serena Yeung, and Li Fei-Fei. Dynamic task prioritization for multitask learning. In *Proc. ECCV*, pages 270–287.

Qiong Hu, Zhizheng Wu, Korin Richmond, Junichi Yamagishi, Yannis Stylianou, and Ranniery Maia. 2015. Fusion of multiple parameterisations for DNN-based sinusoidal speech synthesis with multitask learning. In *Proc. Interspeech*, pages 854–858.

Alex Kendall, Yarin Gal, and Roberto Cipolla. Multitask learning using uncertainty to weigh losses for scene geometry and semantics. In *Proc. CVPR*, pages 7482–7491.

Isabelle Leang, Ganesh Sistu, Fabian Burger, Andrei Bursuc, and Senthil Yogamani. 2020. Dynamic task weighting methods for multi-task networks in autonomous driving systems. *arXiv preprint arXiv:2001.02223*.

Shengchao Liu, Yingyu Liang, and Anthony Gitter. 2019a. Loss-balanced task weighting to reduce negative transfer in multi-task learning. In *Proc. AAAI*, volume 33, pages 9977–9978.

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 *Proc. NAACL*, pages 912–921.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019b. Multi-task deep neural networks for natural language understanding. In *Proc. ACL*, pages 4487–4496.

Zuheng Ming, Junshi Xia, Muhammad Muzzamil Luqman, Jean-Christophe Burie, and Kaixing Zhao. 2019. Dynamic Multi-Task Learning for Face Recognition with Facial Expression. In *Proc. ICCV*.

Chen Qu, Liu Yang, Minghui Qiu, W. Bruce Croft, Yongfeng Zhang, and Mohit Iyyer. 2019a. Bert with history answer embedding for conversational question answering. In *Proc. SIGIR*, pages 1133–1136.

Chen Qu, Liu Yang, Minghui Qiu, Yongfeng Zhang, Cen Chen, W. Bruce Croft, and Mohit Iyyer. 2019b. Attentive history selection for conversational question answering. In *Proc. CIKM*, pages 1391–1400.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proc. ACL*, pages 784–789.

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

Siva Reddy, Danqi Chen, and Christopher D Manning. 2019. CoQA: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, pages 249–266.

Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Zhizheng Wu, Cassia Valentini-Botinhao, Oliver Watts, and Simon King. 2015. Deep neural networks employing multi-task learning and stacked bottleneck features for speech synthesis. In *Proc. ICASSP*, pages 4460–4464.

Yichong Xu, Xiaodong Liu, Yelong Shen, Jingjing Liu, and Jianfeng Gao. 2019. Multi-task learning with sample re-weighting for machine reading comprehension. In *Proc. NAACL*, pages 2644–2655.

Yi-Ting Yeh and Yun-Nung Chen. 2019. FlowDelta: Modeling flow information gain in reasoning for conversational machine comprehension. In *Proc. MRQA*, pages 86–90.

Yu Zhang and Qiang Yang. 2017. A survey on multi-task learning. *arXiv preprint arXiv:1707.08114*.

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